Information Freshness for Monitoring and Control over Wireless Networks

by

Vishrant Tripathi

B.Tech., Indian Institute of Technology, Bombay (2017)
S.M., Massachusetts Institute of Technology (2019)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2023

Authored by: Vishrant Tripathi
Department of Electrical Engineering and Computer Science
August 29, 2023

Certified by: Eytan H. Modiano
Richard C. Maclaurin Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by: Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Information Freshness for Monitoring and Control over Wireless Networks

by

Vishrant Tripathi

Submitted to the Department of Electrical Engineering and Computer Science on August 29, 2023, in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Abstract

In this thesis, we study the optimization of general information freshness metrics in wireless networks, with the goal of applying our theoretical results to problems in real-time monitoring and control. We make contributions in three directions.

First, we consider the optimization of general cost functions of Age of Information (AoI). Here, we develop computationally efficient scheduling algorithms for optimizing information freshness in both single-hop and multi-hop wireless networks. We further develop an online learning formulation when the cost functions of AoI are unknown and propose a new online learning algorithm for this setting called Follow-the-Perturbed-Whittle-Index.

Second, we consider weighted-sum AoI minimization. In this setting, we study how correlation impacts information freshness. We also propose a near-optimal distributed scheduling protocol called Fresh-CSMA for AoI minimization, that has provable performance guarantees.

Third, we apply our theoretical results to problems in multi-agent robotics and monitoring – both via simulations and practical system implementations. We use simulations to demonstrate significant performance improvements in the collection of time-varying occupancy grid maps using multiple robots via the Whittle Index framework. Further, to demonstrate the benefits of our theoretical contributions, we built a real system (WiSwarm) for mobility tracking using a swarm of UAVs, communicating with a central controller over WiFi. Our experimental results show that, when compared to the standard IEEE 802.11 MAC layer + TCP/UDP, our system can reduce AoI by a factor of 109x/48x and improve tracking accuracy by a factor of 4x/6x, respectively.

Thesis Supervisor: Eytan H. Modiano
Title: Richard C. Maclaurin Professor of Aeronautics and Astronautics

Acknowledgments

First and foremost, I am thankful to Prof. Eytan Modiano for giving me the time and freedom to work on problems that I find interesting; and providing crucial help and guidance, whenever I got stuck. His expertise, advice, mentorship and careful attention to detail were essential to the completion of this thesis and are much appreciated.

I would also like to thank my thesis committee - Prof. John Tsitsiklis, Prof. Jon How and Prof. Yury Polyanskiy for their useful comments, suggestions and questions that pushed me to think more deeply about my work.

The work in this thesis is a result of collaborations with many people, and their contributions were essential to its completion. I would like to thank Prof. Luca Carlone, Prof. Sertac Karaman, Igor Kadota, Ezra Tal, Rajat Talak, Luca Ballotta, Nicholas Jones, M. Shahir Rehman, Alexander Warren, and Vallabh Ramakanth. I would especially like to thank Rajat Talak, Igor Kadota, and Prof. Carlone for many useful discussions on a wide variety of topics.

I sincerely thank people who have made my life in graduate school very enjoyable. Bai Liu, Xinzhe Fu and Anurag Rai have been close friends throughout my time here and their support is much appreciated. In addition, I would like to thank my flatmate Aniket Patankar, especially for the countless dinners we cooked together through our PhD journeys. I would also like to thank Mrinalini Singha for all the support - my time with her has been the happiest I have been at MIT.

I was lucky to have been part of a very warm and welcoming community throughout my time at CNRG, LIDS, Ashdown, Sangam, MIT and beyond. I am thankful to the MIT Cricket Club, and friends from soccer, badminton and hiking, for pushing me to go out every once in a while. I would like to thank all of my good friends over the years: I owe a lot to my community of friends here in Boston and I am going to miss them greatly.

Finally, I would like to thank my parents Sheela and Vipul Tripathi, for being a constant source of support and encouragement. I know that this thank you note does not do justice to everything they have done for me over the years. None of this would have been possible without their love and belief in me. Thank you for everything!

Funding

This work was supported in part by the US National Science Foundation (NSF) through grants CNS-1524317, CNS-1713725, CNS-1701964 and CNS-1907905; and by the Army Research Office (ARO) through grants W911NF-17-1-0508 and W911NF-19-1-0322.

THIS PAGE INTENTIONALLY LEFT BLANK

Previously Published Material

Chapter 2 includes prior work:

- [1] **Vishrant Tripathi**, and Eytan Modiano. "A Whittle Index Approach to Minimizing Functions of Age of Information," in Proceedings of Allerton; 2019.

Chapter 3 includes prior work:

- [2] **Vishrant Tripathi**, and Eytan Modiano. "An Online Learning Approach to Optimizing Time-Varying Costs of AoI," in Proceedings of ACM MobiHoc, 2021.

Chapter 4 includes prior work:

- [3] **Vishrant Tripathi, Luca Ballotta**, Luca Carlone and Eytan Modiano. "Computation and Communication Co-Design for Real-Time Monitoring and Control in Multi-Agent Systems," in Proceedings of IFIP WiOpt, 2021.

Chapter 5 includes prior work:

- [4] **Vishrant Tripathi**, and Eytan Modiano. "Age Debt: A General Framework for Minimizing Age of Fnformation," in Proceedings of IEEE INFOCOM Workshop on AoI, 2021.

- [5] **Vishrant Tripathi**, Rajat Talak, and Eytan Modiano. "Information Freshness in Multihop Wireless Networks," in IEEE/ACM Transactions on Networking, 2022.

Chapter 6 includes prior work:

- [6] **Vishrant Tripathi**, Nicholas Jones, and Eytan Modiano. "Fresh-CSMA: A Distributed Protocol for Minimizing Age of Information," in Proceedings of IEEE INFOCOM, 2023.

Chapter 7 includes prior work:

- [7] **Vishrant Tripathi**, and Eytan Modiano. "Optimizing Age of Information with Correlated Sources," in Proceedings of ACM MobiHoc, 2022.

Chapter 8 includes prior work:

- [8] **Vishrant Tripathi, Igor Kadota, Ezra Tal**, M. Shahir Rehman, Alexander Warren, Sertac Karaman, and Eytan Modiano. "WiSwarm: Age-of-Information-based Wireless Networking for Collaborative Teams of UAVs," in Proceedings of IEEE INFOCOM, 2023.

Contents

1 Introduction **23**

1.1 Motivating Example 26

1.2 Contributions and Prior Work 28

1.2.A Single-Hop Networks 29

1.2.B Online Learning of AoI Cost Functions 31

1.2.C Computation Communication Trade-Offs 32

1.2.D Multi-Hop Networks 33

1.2.E Distributed Scheduling 33

1.2.F Correlated Sources 35

1.2.G System Design 35

1.3 Outline 37

2 Information Freshness in Single-Hop Networks **43**

2.1 Model 44

2.2 Restless Multi-Armed Bandit Formulation 46

2.3 Reliable Channels 48

2.3.A Properties of an Optimal Policy 52

2.4 Unreliable Channels 57

2.5 Simulations 60

2.6 Applications 62

2.6.A Monitoring LTI systems 62

2.6.B Monitoring Markov Chains 64

2.7 Summary 67

2.8 Appendix . . . 68
2.8.A Proof of Theorem 1 . . . 68
2.8.B Proof of Theorem 2 . . . 71
2.8.C Proof of Theorem 4 . . . 72
2.8.D Proof of Theorem 5 . . . 76
2.8.E Proof of Theorem 6 . . . 77
2.8.F Proof of Theorem 3 . . . 80
2.8.G Proof of Theorem 7 . . . 82
2.8.H Proof of Theorem 8 . . . 86
2.8.I Proof of Theorem 9 . . . 87
2.8.J Proof of Theorem 10 . . . 89

3 Online Learning of AoI Cost Functions 93
3.1 Single Source Monitoring . . . 94
3.1.A An Epoch Based Formulation . . . 96
3.2 Multiple Sources . . . 103
3.2.A Online Whittle-Index Scheduling . . . 107
3.3 Mobility Tracking . . . 114
3.3.A Levy Mobility . . . 115
3.3.B Adversarial Mobility . . . 117
3.4 Summary . . . 118
3.5 Appendix . . . 119
3.5.A Proof of Lemma 2 . . . 119
3.5.B Proof of Corollary 2 . . . 120
3.5.C Closeness of Whittle and Optimal Policies . . . 120
3.5.D Proof of Theorem 13 . . . 121
3.5.E Proof of Lemma 5 . . . 130

4 Computation and Communication Trade-offs 133
4.1 Model . . . 135
4.2 A Lagrangian Relaxation . . . 140

4.2.A Solving the Decoupled Problem . . . 142
4.2.B Optimizing Processing Times . . . 143
4.3 Whittle Index Scheduling . . . 145
4.4 Applications . . . 148
4.4.A Multi-agent Mapping of Time-Varying Environments . . . 148
4.4.B Smart Ride Sharing Control in Vehicle Networks . . . 151
4.5 Summary . . . 155
4.6 Appendix . . . 156
4.6.A Proof of Theorem 14 . . . 156
4.6.B Restless Multi-Armed Bandit Formulation . . . 164
4.6.C Proof of Lemma 7 . . . 166
4.6.D Entropy Cost as Function of AoI . . . 168

5 Information Freshness in Multi-Hop Networks 173
5.1 Model . . . 174
5.2 Age Debt . . . 177
5.3 Lyapunov Drift Approach . . . 180
5.3.A Single-Hop Broadcast . . . 180
5.3.B General Networks . . . 183
5.3.C Intermediate Debt Queues . . . 185
5.3.D An Example . . . 186
5.4 Choosing Target Vectors . . . 189
5.4.A Gradient Descent . . . 190
5.4.B Flow Control . . . 191
5.5 Numerical Results . . . 192
5.6 Summary . . . 200
5.7 Appendix . . . 200
5.7.A Proof of Lemma 8 . . . 200
5.7.B Proof of Remark 1 . . . 201

6 Fresh-CSMA: A Distributed Protocol for AoI **203**

6.1 System Model . . . 204

6.2 Distributed Scheduling Design . . . 205

6.2.A Fresh-CSMA . . . 206

6.3 Near-Realistic Multiple Access Model . . . 219

6.3.A Collisions . . . 222

6.3.B Backoff Timer Overhead . . . 224

6.4 Going Beyond Age of Information . . . 227

6.5 Numerical Results . . . 231

6.6 Summary . . . 240

7 Optimizing Age of Information with Correlated Sources **241**

7.1 System Model . . . 243

7.1.A Correlated Age of Information . . . 244

7.1.B Goal . . . 246

7.2 Scheduling Policies . . . 246

7.2.A Stationary Randomized Policies . . . 246

7.2.B Max-Weight Policy . . . 249

7.3 Scaling . . . 251

7.3.A An Upper Bound . . . 252

7.3.B Random Geometric Graphs . . . 256

7.4 Robustness . . . 256

7.5 Learning The Correlation Matrix . . . 259

7.5.A Online Setting . . . 259

7.6 Numerical Results . . . 261

7.7 Summary . . . 266

7.8 Appendix . . . 266

7.8.A Proof of Theorem 20 . . . 266

7.8.B Proof of Theorem 21 . . . 268

7.8.C Proof of Theorem 22 . . . 270

7.8.D Proof of Theorem 23 . . . 274
7.8.E Proof of Theorem 24 . . . 277
7.8.F Proof of Theorem 25 . . . 278
7.8.G Proof of Theorem 26 . . . 282
7.8.H Proof of Theorem 27 . . . 285
7.8.I Proof of Theorem 28 . . . 286

8 WiSwarm: A System for AoI-based Scheduling 289
8.1 Design . . . 292
8.1.A Rate Control and Queuing . . . 292
8.1.B Scheduling . . . 294
8.1.C Middleware . . . 296
8.2 Implementation . . . 299
8.2.A Mobility Tracking Application . . . 299
8.2.B Mobile Objects . . . 300
8.2.C Follower Sensing-UAVs . . . 301
8.2.D Leader Compute Node . . . 302
8.3 Evaluation . . . 304
8.3.A Stationary Experiments . . . 305
8.3.B Flight Experiments . . . 310
8.4 Summary . . . 314

9 Concluding Remarks 317
9.1 Open Questions . . . 318
9.1.A Complexity of Age Optimal Scheduling . . . 318
9.1.B Multi-Hop AoI Optimization . . . 319
9.1.C Correlated or Coupled Sources . . . 319
9.1.D From Freshness to Semantics . . . 320
9.1.E Applications . . . 320
9.1.F Hardware Implementations . . . 321

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

1-1 Illustration of the AoI evolution for sample generation and delivery processes. The first update is generated at the source at time t_1 and is delivered to the destination at time t_1'. The destination now has information about the source that is $t_1' - t_1$ old, so AoI drops to $A(t_1') = t_1' - t_1$. The second update is generated at time t_2 and delivered at t_2'. So, $A(t)$ increases linearly until time t_2' and then drops to $A(t_2') = t_2' - t_2$. 24

1-2 Single source monitoring over a costly wireless channel. 26

2-1 Linear, quadratic, logarithmic and indicator cost functions for a sample age process. The linear process tracks the actual values of AoI. 44

2-2 N sources transmitting updates to a base station over a wireless channel, with different reliabilities. 45

2-3 Symmetric two-state Markov chain, representing the state of the ith process. 65

3-1 Single source monitoring 94

3-2 Multiple source monitoring 104

3-3 Levy Mobility: Average Tracking Error v/s number of nodes 116

3-4 Adversarial Mobility: Average Tracking Error v/s number of nodes . 118

4-1 Example: four drones monitor different regions and send updates to a base station over a wireless channel. Each agent spends time τ_i processing the collected measurements before sending. A scheduling algorithm prioritizes transmissions to the base station. This paper focuses on the co-design of the processing times τ_i and the scheduling policy. 134

4-2 AoI evolution for agent i. The agent acquires and processes new samples every τ_i time-slots. When the base station (B.S.) requests a new update, the agent sends the most recent sample that has finished processing, taking $r_i(\tau_i)$ time-slots for transmission. The variable $\delta_i^{(k)}$ represents the waiting time in the buffer for update k. Upon a new update delivery, the AoI at the base station $A_i(t)$ drops to the age of the delivered update. 137

4-3 **Multi-agent mapping over 9 regions:** each agent monitors and builds a local grid map of a region, and sends map updates to a base station. The occupancy in the regions is time-varying. A scheduling policy specifies how to share the communication channel among the agents. Processing times specify how much time each agent spends in generating new map updates. 148

4-4 Transition probabilities and optimal processing time allocations plotted for each region. The probabilities are plotted on a logarithmic scale while the processing times are plotted in number of time-slots. 150

4-5 Performance of different scheduling policies vs. processing times τ. Solid lines represent performance of different classes of scheduling policies as the processing time τ varies. The dotted lines represent the scheduling performance with processing times computed using Algorithm 5. 152

4-6 Drivers calculate their route by processing the oldest R requests (green queue portion). The TSP solver starts from the current driver location and involves pick ups (P) and drop offs (D) of the processed requests. 153

4-7 Left: long processing may cause large gaps between new routes calculated by the driver (solid gray) and the outdated ones stored at the scheduler (dashed gray), yielding bad matches (red dots). Right: with short processing, the matched requests are close to the actual routes (green dots). 153

4-8 Average service time with varying processing time τ_s. 155

4-9 The average entropy of a region v/s AoI. Solid lines represent p = 0.0005 and dashed lines represent p = 0.001 170

5-1 Example of a line network with a unicast flow from node 1 to node 3 183

5-2 Example of a five node multihop network with two competing flows. 187

5-3 Evolution of three debt queue quantities with time, along with scheduling decisions involving link a. 188

5-4 Weighted-sum AoI minimization in broadcast networks with unreliable channels 193

5-5 Functions of Age minimization in broadcast networks with reliable channels 194

5-6 Sum of virtual debt queues vs time 195

5-7 A single unicast flow on a line network (neighboring nodes interfere) 196

5-8 A single unicast flow on a line network (all nodes interfere) 197

5-9 Broadcast flows in multihop networks with 5 and 6 nodes 198

5-10 Weighted Age minimization of broadcast flows in multihop networks with 5 nodes 199

6-1 Single-hop broadcast network with N sources sending updates to a base station over a shared channel. 204

6-2 Events within frame t in the near-realistic multiple access model. Four sources choose backoff timers $D_1(t), ..., D_4(t)$. Source 1's timer runs out first, after which it transmits its update for M minislots. . . 220
6-3 Normalized average AoI vs system size (N) for symmetric weights. . 232
6-4 Normalized average AoI vs system size (N) for unequal weights. . . 232
6-5 Normalized average AoI vs α. 234
6-6 Probability of collisions vs β 236
6-7 Probability of collisions vs B 236
6-8 Average backoff overhead vs β 237
6-9 Average backoff overhead vs B 237
6-10 Symmetric two-state Markov chain representing the ith source. . . 238
6-11 Normalized average AoII vs system size N when symmetric Markov sources 238
6-12 Normalized average AoI vs system size N when monitoring symmetric Markov sources 239

7-1 Sources share information locally and send updates to a base station.243
7-2 Correlated AoI evolution. 245
7-3 Average AoI vs network size N for random geometric graphs. 261
7-4 Average AoI vs network size N for hyperbolic geometric graphs. . . . 262
7-5 Average AoI vs correlation probability p. 264
7-6 Moving window average AoI vs time. 265
7-7 Grid of size $\frac{r}{\sqrt{2}}$ on the unit square. 285

8-1 Setup for flight experiments with 5 UAVs. 291
8-2 Overview of the Networking Middleware that provides information freshness for applications with a team of followers and a single leader. 292
8-3 Mobility tracking using a swarm of sensor-UAVs and a leader compute node. 299

8-4 (a) Sensing-UAV. (b) Autonomous car with an identifying ArUco marker on top. 301

8-5 Screenshot from the videos used to simulate car movement during stationary experiments. The tags are programmed to perform random walks with time-varying velocities. The virtual UAVs need to keep track of the tags. On the right, two examples of 224x224 frames sent to the Compute Node by the RasPis based on their current virtual UAV locations. 305

8-6 (a) AoI and (b) tracking error of baseline WiFi-TCP and WiFi-UDP plotted against the update generation rate of each of the $N = 6$ emulated UAVs. 306

8-7 Mean AoI per UAV plotted for (a) fixed-rate (50 fps) and (b) optimized rate WiFi, as well as WiSwarm, as the number of UAVs increases. 308

8-8 Tail (95^{th} percentile) AoI per UAV plotted for (a) fixed-rate (50 fps) and (b) optimized rate WiFi, as well as WiSwarm, as the number of UAVs increases. 309

8-9 Mean Throughput per UAV plotted for (a) fixed-rate (50 fps) and (b) optimized rate WiFi, as well as WiSwarm, as the number of UAVs increases. 310

8-10 Mean Tracking Error per UAV plotted for (a) fixed-rate (50 fps) and (b) optimized rate WiFi, as well as WiSwarm, as the number of UAVs increases. 311

8-11 Two Examples of 160x160 grayscale video frames sent by the RasPis during flight experiments. 312

8-12 Coordinates of sensing-UAVs and target cars in 2-D and 3-D, for a two drone flight experiment running WiSwarm. 313

8-13 Coordinates of sensing-UAVs and target cars in 2-D and 3-D, for a two drone flight experiment running optimized WiFi-UDP. 314

8-14 Histograms of (a) AoI and (b) tracking error for flight experiments with two UAVs, comparing WiSwarm with WiFi. 315

List of Tables

2.1 Cost of the Whittle index policy and the optimal dynamic programming policy for 2 sources. 60
2.2 Cost of the Whittle index policy and the optimal policy for more than 2 sources. 61

8.1 Average tracking error per sensing-UAV (in meters). 314
8.2 Average AoI per sensing-UAV (in seconds). 314

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

Monitoring and control of dynamical systems are fundamental and well-studied problems. Many emerging applications involve performing these tasks over communication networks. Examples include: sensing for IoT, control of robot swarms, real-time surveillance, and environmental monitoring by sensor networks. Such systems typically involve multiple agents collecting and sending information to a central entity where data is stored, aggregated, analyzed, and then possibly used to send back control commands. Due to the dramatic improvements both in on-device and edge computing, and in wireless communication over the past two decades, there has been a rapid growth in the size and scale of such networked systems.

Age of Information (AoI) is a metric that captures timeliness of received information at a destination [9, 10, 11]. Unlike packet delay, AoI measures the lag in obtaining information at a destination node, and is therefore suited for applications involving time sensitive updates. Age of information, at a destination, is defined as the time that has elapsed since the last received information update was generated at the source. AoI, upon reception of a new update, drops to the time elapsed since generation of the update, and grows linearly otherwise. Over the past decade, there has been a rapidly growing body of work on analyzing AoI

for queuing systems [9, 10, 11, 12, 13, 14], using AoI as a metric for scheduling policies in networks [15, 16, 17, 18, 19, 1, 20, 21] and for monitoring or controlling systems over networks [22, 23, 24, 25, 26]. For detailed surveys of AoI literature, we point the reader to [27] and [28].

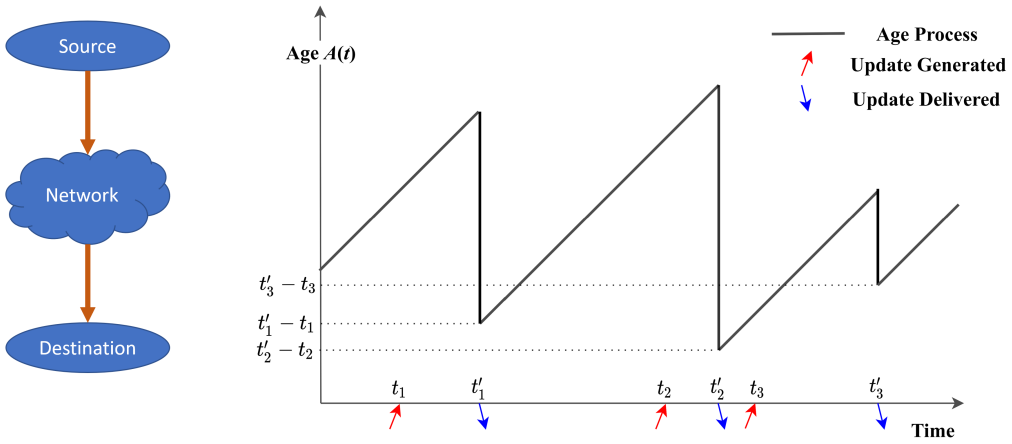


Figure 1-1: Illustration of the AoI evolution for sample generation and delivery processes. The first update is generated at the source at time t_1 and is delivered to the destination at time t_1'. The destination now has information about the source that is $t_1' - t_1$ old, so AoI drops to $A(t_1') = t_1' - t_1$. The second update is generated at time t_2 and delivered at t_2'. So, $A(t)$ increases linearly until time t_2' and then drops to $A(t_2') = t_2' - t_2$.

Age-of-Information (AoI) is an end-to-end metric that characterizes *how old the information is from the perspective of the destination.* Consider a destination receiving time-stamped updates from a source over a network. Let $\tau(t)$ be the time-stamp of the *latest update* received at the destination by time t. The AoI associated with this source-destination pair is then defined as $A(t) := t - \tau(t)$. The AoI increases linearly with time when no updates are delivered, representing the information getting older. At the moment a *fresher* update from the source is received at the destination, the value of $\tau(t)$ increases and the AoI reduces to the delay of the received update. This evolution of the AoI metric with time is illustrated in Fig. 1-1.

Typically, AoI represents a measure of distortion between the state of the system that is expected at the monitor based on past updates and the actual current state of the system. Thus, a larger age corresponds to the monitor having a higher uncertainty about the current state of the system being observed. This, in turn, means that ensuring a low average AoI can lead to higher monitoring accuracy or better control performance. While AoI is a proxy for measuring the cost of having out-of-date information, it may not properly reflect the impact of stale information on system performance.

Illustrative example: Consider a mobility tracking application where a monitor wants to keep track of the location of multiple nodes. If the AoI associated with the last known position of a moving object is 1.5 seconds, this means that the object has been moving around for 1.5 seconds without the monitor knowing about it. Clearly, a larger AoI corresponds to the monitor having a higher uncertainty about the current position of the object. Similarly, a larger average object velocity also corresponds to the monitor having a higher position uncertainty. Therefore, to support mobility tracking applications, the underlying communication network should strive to keep the AoI associated with the position of every moving object as low as possible, further prioritizing objects with higher velocities.

When multiple systems or sources are being observed at the same time, there arises a need to differentiate between them based on their relative importance. Namely, if a system evolves slowly, then a higher AoI does not necessarily mean higher uncertainty or cost. Conversely, if a system has fast dynamics, then even a small AoI might mean higher distortion and poor performance. Further, certain sources might be more critical to the system performance than others. Many works on AoI-based scheduling for multiple sources consider weighted-sum AoI minimization [15, 16, 17], where weights represent the relative importance of each source. Typical assumptions involve the weights being fixed and known in

advance, based on the underlying application or systems being monitored.

Further, recent works on networked control systems [25, 26] and remote estimation [22, 23, 24] emphasize that even for very simple systems, linear AoI is not a sufficiently accurate metric to track accuracy or overall system performance. This has motivated interest in using general, possibly non-linear cost functions of AoI that reflect the cost of delayed information more accurately [29, 30, 1, 25]. Next, we discuss a simple example that motivates why non-linear functions of AoI are key to performing monitoring tasks over wireless networks.

1.1 Motivating Example

Consider a linear dynamical system that is being observed over a costly wireless channel (see Fig. 1-2). The source to be monitored evolves as follows:

$$\boldsymbol{x}(t+1) = \boldsymbol{G}\boldsymbol{x}(t) + \boldsymbol{w}(t), \tag{1.1}$$

where $\boldsymbol{G}$ describes the source dynamics and $\boldsymbol{w}(t) \sim \mathcal{N}(0, \Sigma)$ is i.i.d. Gaussian noise in every time-slot.

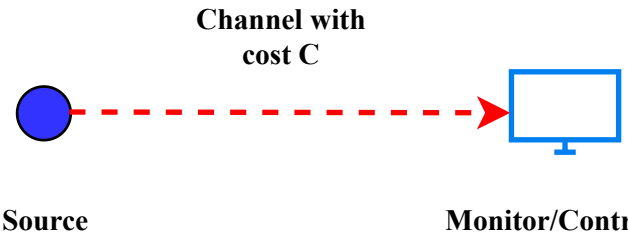


Figure 1-2: Single source monitoring over a costly wireless channel.

The monitor can, in every time-slot, decided to observe the state of the source exactly and pay an observation cost C or estimate the current state of the source based on prior observations. The goal of the monitor is to minimize the sum of the monitoring error and observation cost, averaged over time.

Suppose that the monitor last observer the state of the source τ time-slots ago. Then, the MMSE estimate of the state in the current time-slot is given by:

$$\hat{\boldsymbol{x}}(t) \triangleq \mathbb{E}\Big[\boldsymbol{x}(t)|\boldsymbol{x}(t-\tau)=\boldsymbol{x}\Big] = \boldsymbol{G}^{\tau}\boldsymbol{x}. \tag{1.2}$$

The last equality above follows from the linearity of expectation and the fact that the noise is zero-mean in every time-slot.

Now consider a scheduling policy π that specifies whether the monitor should sample the source at time-slot t or not, for every time-slot. This decision is represented by the indicator variable $u(t)$. The optimization problem faced by the monitor can then be formulated as:

$$\arg\min_{\pi}\left(\limsup_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\left(\mathbb{E}\Big[||\boldsymbol{x}(t)-\hat{\boldsymbol{x}}(t)||^2\Big]+Cu(t)\right)\right). \tag{1.3}$$

Interestingly, this problem can be converted to an equivalent problem of the form

$$\arg\min_{\pi}\left(\limsup_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\Big(f(A(t))+Cu(t)\Big)\right), \tag{1.4}$$

where $A(t)$ is the AoI at the monitor of information regarding the source and $f(\cdot)$ is a monotone increasing function given by $f(h)=\sum_{k=0}^{h} Tr(\boldsymbol{G}^{k^T}\boldsymbol{G}^{k}\Sigma)$ [25, 26]. We will derive this result in detail in Chapter 2.

Similar results can be derived for monitoring discrete Markov chains and controlling Linear systems with Quadratic costs and Gaussian noise (LQG) over a wireless channel. **Thus, the design of policies that achieve optimal monitoring and control of linear systems over constrained communication networks requires optimization of general AoI cost functions**. These AoI cost functions will be one of the main focuses of this thesis.

1.2 Contributions and Prior Work

At a high level, our contributions in this thesis can be split into three parts -

1. **Cost Functions of AoI.** A large portion of this thesis is dedicated to optimizing general AoI cost functions. In Chapter 2 we optimize AoI cost functions in single-hop networks. In Chapter 3, we consider the setting where AoI cost functions are not known in advance, time-varying and possibly adversarial. In Chapter 4, we extend the single-hop framework with cost functions to consider computation-communication tradeoffs, and sources that generate updates of different sizes and at different rates. In Chapter 5, we consider the optimization of general AoI cost functions for multi-hop networks.

2. **Weighted-Sum AoI.** We study two specific aspects of the weighted-sum AoI minimization problem in this thesis that haven't been looked at in great depth in AoI literature before. In Chapter 6, we explore how to design a distributed scheduling policy that replicates the performance of centralized policies proposed in [15, 16, 17] and in Chapter 7, we explore how to model correlation between sources in this setting.

3. **Applications and Implementation.** A crucial part of this thesis is the focus on applying ideas from AoI optimization to problems in monitoring, control, and robotics. We demonstrate the equivalence between monitoring of LTI systems and monitoring of Markov chains to minimizing AoI cost funtions in Chapter 2. We apply the Whittle framework to two problems in robotics - multi-agent occupancy grid mapping, and ride-sharing with computational offloading - in Chapter 4. Finally, in Chapter 8, we describe the implementation of an application layer scheduling system that utilizes key results from AoI literature over the past decade to deliver significant performance and scalability gains in a multi-UAV mobility tracking task.

Next, we provide a detailed literature review of prior works.

1.2.A Single-Hop Networks

The problem of minimizing age of information in single-hop networks was first considered in [31] and [15]. In these works, the authors considered a base station collecting time-sensitive information from a number of sources over a wireless broadcast network, where only one source can send an update at any given time. They looked at weighted linear combinations of AoI of all sources as the metric to be optimized. This prompted the design of low complexity scheduling policies that provably minimize weighted sum AoI at the base station, up to a constant multiplicative factor. These results crucially depend on the fact that for linear AoI, one can find a stationary randomized policy that is factor-2 optimal. As we will see in Chapter 2, this observation does not hold for general functions of AoI. In fact, stationary randomized policies can be arbitrarily worse than simple heuristic policies.

Scheduling problems with weighted linear combinations of age have also been considered with throughput constraints in [16] and with general interference constraints in [17]. AoI-based scheduling with stochastic arrivals was considered in [20], where a Whittle Index policy was shown to have good performance. In [21], the authors extended prior results to different sampling behaviors, update sizes and transmission times in the same weighted-sum AoI context.

On the other hand, nonlinear cost functions of age were introduced as a natural extension to the AoI metric in [11] for characterizing how the level of dissatisfaction depends on data staleness in a more general manner. Nonlinear functions of age of information were also discussed in the context of queuing systems in [27] and [32]. These papers develop the notion of value of information and use nonlinear cost of update delays, which correspond to nonlinear age cost functions.

Nonlinear functions of age have also been discussed in the context of networked control systems in [25],[26] and [33]. In [25], the authors discuss a real time networked control system and show that the cost function is characterized as a non-decreasing, possibly nonlinear, function of AoI. In [26], the authors formulated the state estimation problem for an LTI system, where the state of a discrete-time LTI system can be observed in any time-slot by paying a fixed transmission cost. The problem of minimizing the time-average of the sum of the estimation error and transmission cost reduces to minimizing a non-decreasing age-cost function for a single source with a fixed transmission cost (our motivating example from Section 1.1). We explore this relationship more closely in Section 2.6, where we derive a similar equivalence for monitoring *multiple* LTI systems or Markov chains.

Scheduling to minimize functions of age has also been considered in [18] and [30]. In [18], the authors deals with minimizing symmetric functions of age of sources over multiple orthogonal unreliable channels and show that simple greedy schemes are asymptotically optimal. In [30], the authors formulate the general functions of age problem with reliable channels and develop a high complexity algorithm that achieves minimum age. They also derive a key structural property of the optimal policy in this setting - the optimal policy is always periodic. However, their approach does not extend to the setting with unreliable channels. In Chapter 2, we consider unreliable channels and also build upon results from [30] and [20] to derive stronger structural properties for optimal policies. These properties hint at why the performance of the heuristic Whittle index policy may be close to optimal. In more recent work [34], published subsequent to our original conference paper [1], it has been shown that the Whittle policy is indeed asymptotically optimal for *linear* functions of AoI.

1.2.B Online Learning of AoI Cost Functions

Many works on AoI-based scheduling for multiple sources consider weighted-sum AoI minimization [15, 16, 17], where weights represent the relative importance of each source. Typical assumptions involve the weights being fixed and known in advance, based on the underlying application or systems being monitored.

Recent works on networked control systems [25, 26] and remote estimation [22, 23, 24] emphasize that linear AoI is not a sufficiently accurate metric to track accuracy or overall system performance. This has motivated interest in using general, possibly non-linear cost functions of AoI that reflect the cost of delayed information more accurately [29, 30, 1, 25]. Typical assumptions in works studying non-linear AoI include knowing the cost functions in advance [27, 30, 1, 26], assuming that cost functions increase monotonically with AoI [27, 18, 23, 26] and decoupled costs across multiple systems [1, 26]. Importantly, we observe that all of these works assume there is some *fixed and known* cost function mapping the AoI to system performance and that the source dynamics are stationary. In Chapter 3, we ask the following question - *what happens when the AoI cost functions are time-varying, not known in advance, and possibly adversarial?* Related to this, a context-aware notion of AoI was proposed in [35], where the authors considered sources with *known* time-varying context that influences the AoI cost function.

There has been some recent work at the intersection of AoI and learning to sample or schedule sources. Learning how to sample a source through a network with an unknown delay profile while minimizing AoI has been considered in [36]. Minimizing AoI with unknown and adversarial channel processes has also been considered in [37] and [38], respectively. However, these works do not look at learning of the AoI cost functions themselves.

1.2.C Computation Communication Trade-Offs

Two key directions of innovation in multi-agent networked systems involve a) pushing the computation to be distributed across the network, such that all agents perform local processing of the collected measurements, and b) designing scheduling algorithms that efficiently share limited communication resources across all devices and ensure timely delivery of information. However, existing work on communication scheduling [9, 11, 15, 17, 1] disregards distributed processing, while related work on sensor fusion [39, 40, 41] focuses on designing distributed algorithms, rather than allocating computational resources at each node.

Our processing and scheduling co-design problem in Chapter 4 is motivated by recent advances in embedded electronics, as well as the development of efficient estimation and inference algorithms for real-time applications on low-powered devices [42, 43, 44, 45]. The output accuracy of such algorithms increases with the runtime, in line with the delay-accuracy trade-off we consider in this chapter. Another application of such a trade-off involves deciding on computation offloading in cloud robotics, which has been the focus of recent works on real-time inference by resource-constrained robots [46, 47]. In this context, sending raw data can induce long transmission delays, but allow better inference by shifting the computational burden the cloud.

Our main theoretical contribution involves extending the Whittle frame work from Chapter 2 to sources with different update generation rates and update sizes, and solving a joint optimization problem involving wireless scheduling and distributed processing. We apply this framework to two practical problems in robotics via simulations to demonstrate the benefits of our approach.

1.2.D Multi-Hop Networks

While there has been significant work on minimizing AoI in single-hop wireless networks, optimizing AoI or its cost functions over multi-hop networks has received limited attention in the literature so far. In [48], a switch type network was considered under physical interference constraints, and the problem of scheduling finitely many update packets was shown to be NP-hard for this network. AoI in multi-hop networks of queues was studied in [49], where LIFO queue service was shown to reduce age. AoI minimization in multihop wireless networks with all-to-all broadcast flows was considered in [50, 51]. Scaling of AoI in multihop multicast networks was studied in [52].

Finding low complexity near optimal scheduling and routing schemes for AoI minimization which handle general network topologies, interference constraints, cost functions, different types of flows and link reliabilities *has remained an open problem.* We target this problem in Chapter 5.

1.2.E Distributed Scheduling

A large majority of the works on weighted-sum AoI minimization [31, 15, 16, 17, 20, 21] focus on the design of *centralized scheduling algorithms.* Specifically, at the beginning of each time-slot, the base station looks at the AoI values for each node in the network and then decides which node to poll for an update. This requires support for polling protocols, which might not be available at the MAC layer and might involve excessive overhead for networks with many nodes. This has motivated the need to study distributed schemes for information freshness in wireless networks.

In [53], the authors consider a simple class of distributed algorithms - each node transmits using a fixed attempt probability in each time-slot, and they design a scheme to find the attempt probabilities that minimize weighted sum AoI.

In [54], the authors consider a single-hop setting with stochastic arrivals and solve the AoI minimization problem by deriving a Whittle index and propose a heuristic ALOHA-like scheme called Index-Prioritized-Random-Access (IPRA) where a node is active with a fixed probability but only when its AoI exceeds a specified threshold. The idea of ALOHA with thresholds has been explored in further detail in subsequent works. In [55, 56], the authors study the performance of ALOHA style random access protocols for information freshness and propose the idea of "thinning", where only nodes with AoI greater than a specified threshold remain active. Along similar lines, the performance of threshold-ALOHA for AoI minimization is also analyzed in [57, 58] with performance bounds derived in a symmetric setting.

Another class of distributed protocols commonly used in wireless networks for medium access is Carrier Sense Multiple Access with Collision Avoidance, also known as CSMA/CA. Throughout this thesis, when we use the term CSMA, we use it to denote CSMA/CA style protocols. Age of information has also been analyzed in settings where nodes employ CSMA [59, 60, 61, 55, 62, 63]. In [59], the authors analyze an idealized version of IEEE 802.11 CSMA and optimize the backoff timer parameters to minimize AoI. They show that this version of CSMA has poor delay and freshness performance in certain settings and suggest the need for new distributed scheduling schemes for AoI. In [61, 60], the authors analyze AoI under standard CSMA in broadcast environments. In [62, 63], the authors study sleep-wake carrier sensing based scheduling with the goal of minimizing energy consumption together with AoI.

The CSMA protocol has been well studied in wireless networks for a long time, especially for optimizing throughput and utility. It was shown in [64] that CSMA tends to outperform ALOHA in terms of both throughput and delay. In [65], the author developed an approximation that allows closed-form analysis for the IEEE 802.11 implementation of CSMA. More recently, there has been work on through-

put and utility optimization by trying to replicate centralized scheduling policies' behavior using CSMA style schemes [66, 67, 68, 69]. Typically, these works involve modifying the way CSMA backoff timers work by adding dependence on the current network state (e.g. queue lengths) and then analyzing performance guarantees by comparing to a centralized scheduling scheme. Our analysis and approach in Chapter 6 are motivated by this line of work, in particular the Fast-CSMA protocol proposed in [69].

1.2.F Correlated Sources

Most AoI optimization works assume that information and updates from different sources are uncorrelated, i.e. different sources do not provide information about each other. In Chapter 7, we explore how to model information freshness in the presence of correlated sources.

There has been some prior work in trying to understand how correlation influences information freshness. In [70], the authors consider updates from a single source that are temporally correlated. In [71, 72], the authors consider a network of cameras with overlapping fields of view and formulate a joint optimization problem that looks at processing and scheduling. In [73], the authors consider a setting with multiple sensors partially monitoring a single source, where updates from at least M sensors are required to reconstruct the state of the source. In [74], the authors consider spatially correlated updates from a random field and study the optimal density at which to place sensors. In [75], the authors consider a two hop setting where sources can send updates to multiple sensors.

1.2.G System Design

WiFi is a common choice for deploying time-sensitive applications. Some examples include: automated fulfilment warehouses at Amazon [76], vehicle-to-

everything communication in New York City [77, 78, 79], and various multi-agent systems using teams of UAVs [80, 81, 82, 83, 84, 85, 86, 87] and/or ground robots [88, 89, 90, 80]. WiFi is attractive for deploying such systems because it is inexpensive, tried-and-true, and readily available in sensors, cameras, UAVs, and robotic platforms. However, it is well-known that WiFi's performance degrades sharply as the network size scales and traffic load increases. This is due to WiFi's Carrier-Sense Multiple Access (CSMA) distributed random access mechanism that works well for small-scale underloaded networks, but not for large-scale systems with stringent latency or freshness requirements. When a larger number of sources attempt to transmit using distributed random access, the higher probability of packet collisions leads to lower throughput and higher latency, which can result in degraded performance (or even failure) of the time-sensitive application. This motivates our work on designing an application layer system for efficient AoI-based scheduling in Chapter 8.

Cellular networks employ a centralized resource allocation mechanism [91] that prevents packet collisions and prioritizes traffic according to its preassigned Quality-of-Service (QoS) level. Major drawbacks of cellular networks include their high cost and the fact that the technology is proprietary and vendor specific, meaning that customizing resource allocation to the needs of distinct applications is prohibitively complex. An effort to make cellular networks open, virtualized, and programmable is underway by the O-RAN Alliance [92, 93]. Programmability is seen as key to enabling the deployment of *custom solutions that satisfy application-specific performance requirements* [94]. However, this is a long-term approach that requires substantial efforts and negotiations with vendors, network operators, and hardware/software developers. Moreover, it is limited to environments with cellular infrastructure. Applications such as search and rescue following a disaster and automated exploration of remote environments may not be able to use such infrastructure.

1.3 Outline

We describe a detailed the outline of the rest of the thesis below.

- **Chapter 2** is organized as follows. In Section 2.1, we describe the general system model for minimizing functions of AoI over single-hop wireless networks. In Section 2.2, we describe the equivalent restless multi-armed bandit formulation and discuss why we use the Whittle Index approach to solve the problem. In Section 2.3, we discuss the functions of age problem with reliable channels, develop the Whittle Index solution for this setting, and also prove key structural properties that an optimal policy must satisfy. In Section 2.4, we find the Whittle Index policy for the functions of age problem with unreliable channels. In Section 2.5, we provide simulation results that verify our theoretical results. In Section 2.6, we show that the problem of minimizing monitoring error for linear time-invariant systems when observing them over a wireless channel is equivalent to minimizing functions of AoI. We also show a similar result for monitoring symmetric Markov chains over a wireless channel. This shows the direct applicability of our Whittle framework to a large class of wireless monitoring problems.

- In **Chapter 3** we consider online learning of AoI cost functions. In Section 3.1 we formulate a problem that involves monitoring a single non-stationary source over a costly communication channel. We design an epoch based framework in which the AoI cost functions change across epochs in an unknown time-varying manner, but remain fixed within an epoch. At the end of each epoch, the scheduler receives feedback (either partial or full) regarding the cost in the previous epoch and uses it to decide a policy for the next epoch. We provide simple scheduling algorithms that have sublinear worst-case regret compared to the best fixed policy in hindsight. Our main contribution here is formulating the problem in such a way that

we can apply techniques from online optimization. In Section 3.2, we use insights from the single source model to develop an epoch based framework for online scheduling of multiple sources. In each epoch, the scheduler needs to decide on a scheduling policy that specifies which source gets to send an update in every time-slot. The goal is to dynamically adapt the scheduling policy to optimize for overall monitoring cost, as the AoI cost functions change across epochs in an unknown manner. Since the number of scheduling policies of a given length grows exponentially in the number of sources, it becomes computationally infeasible to implement traditional online learning algorithms directly in the multiple source setting. We design a new online learning algorithm called *Follow the Perturbed Whittle Leader* (FPWL) for this setting that is computationally feasible while also achieving low regret. Here, analyzing regret is especially challenging due to the combinatorial nature of the scheduling problem and since the Whittle index is only an approximately optimal solution for the offline problem. Our algorithm and its regret analysis are novel and of independent interest to the study of online learning for restless multi-armed bandits with time-varying costs. In Section 3.3, we apply the algorithms that we develop to a mobility tracking problem and illustrate the performance benefits of using online learning for scheduling.

- In **Chapter 4**, we explore the joint optimization of computation and communication resources for monitoring and control tasks. In Section 4.1, we develop a general framework to jointly optimize computation and communication for real-time monitoring and decision-making. This framework extends existing work by a) considering joint optimization of scheduling in addition to processing, and b) addressing a general model that goes beyond linear systems. In Sections 4.2-4.3, we develop low-complexity scheduling and processing allocation schemes that perform well in practice. The co-

design problem is a multi-period resource allocation problem and is hard to solve in general due to its combinatorial nature. To solve the scheduling problem, we generalize the Whittle index framework proposed in Chapter 2 for sources that generate updates at different rates and of different sizes. Finally, in Section 4.4, we demonstrate the benefits of using our methods in two practical applications from robotics and autonomous systems: multi-agent occupancy grid mapping in time-varying environments and ride-sharing systems with local route optimization. Our simulations show that we can achieve performance improvements of 18 – 35% in the mapping application and 75 – 82% in the ride-sharing application with respect to baseline approaches.

- In **Chapter 5**, we consider minimization of AoI costs in general multihop networks. In Section 5.2, we provide a recipe to transform AoI optimization problems into network stability problems. Instead of trying to solve for the best scheduling and routing policies directly, we assume that we have access to a set of target values which represent the average age cost for every flow in the network. We introduce the notion of *Age Debt* and set up a virtual queuing network that is stable if and only if there exists a feasible network control policy that can achieve the specified target costs. In Section 5.3, we use Lyapunov drift based methods to stabilize this system of virtual queues and achieve the desired target age costs. In Section 5.4, we further discuss how to choose the right age cost targets, when there is no access to either an optimization oracle or a system administrator specifying requirements for each flow. Finally, in Section 5.5, we provide detailed simulation results that compare our proposed AoI optimization methods with prior works. We find that Age Debt and its variants perform as well as or better than the best known scheduling and routing schemes in a wide variety of network settings.

- In **Chapter 6**, we propose Fresh-CSMA to replicate the behavior of centralized scheduling schemes that minimize AoI. In Section 6.1 we discuss our system model and set up the single-hop weighted age minimization problem. In Section 6.2 we introduce the Fresh-CSMA protocol in an idealized setting and provide performance guarantees that show that it can closely match the centralized max-weight scheduling policy both per time-slot and over the entire time horizon. In Section 6.3, we relax some of the assumptions from our idealized model and study the Fresh-CSMA protocol under a more realistic setting. We analyze two keys aspects - the probability of collision and the total time lost due to the backoff timers during which the channel remains idle. In Section 6.4, we consider the recently proposed information freshness metric called Age of Incorrect Information (AoII) and extend our CSMA design to incorporate this metric. In Section 6.5, we provide simulations that support our theoretical results.

- The focus of **Chapter 7** is to understand to role of correlation in designing scheduling policies for information freshness in wireless networks. In Section 7.1, we formulate a simple model to analyze weighted-sum average AoI in the presence of correlated sources under wireless interference constraints. In Section 7.2, we use this model to design scheduling policies that can utilize the correlation structure between sources. We formulate a convex problem that solves for the optimal stationary randomized policy and show that it is factor-2 optimal in general. We then develop a Lyapunov drift-based max-weight policy that works well in practice and show that it is also constant factor optimal. In Section 7.3, we provide scaling results that allow us to understand how the degree of correlation affects information freshness. In Section 7.4, we discuss some alternate ways to model correlation and show that the average AoI for these models remains the same as our proposed model under randomized policies. This highlights the ro-

bustness of our results to the way in which correlation is modeled. In Section 7.5, we consider the setting where correlation parameters are unknown and possibly time-varying. Here, we propose a heuristic algorithm called *EMA-max-weight* based on exponential moving averages. This algorithm attempts to both keep track of the correlation parameters and adjust the scheduling decisions in an online manner so as to keep information fresh at the base station. Finally, in Section 7.6, we show numerically that our proposed policies outperform scheduling schemes that ignore the correlation structure inherent in the problem and verify our theoretical results.

- In **Chapter 8**, we describe the design of an application layer networking middleware that customizes WiFi to the needs of time-sensitive applications that rely on multi-agent systems. In Section 8.2, we describe the design and implementation of WiSwarm which is an instantiation of the networking middleware for information freshness tailored to a mobility tracking application. In Section 8.3, we evaluate the performance of both WiFi and WiSwarm for the mobility tracking application. We perform our experiments in a *dynamic indoor campus space with multiple external sources of interference* such as WiFi base stations, mobile phones, and laptops. Through stationary and flight experiments involving multiple sources, we demonstrate significant performance gains of WiSwarm over traditional WiFi.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2

Information Freshness in Single-Hop Networks

In this chapter, we consider a wireless broadcast network with N sources generating real-time updates that need to be sent to a monitoring station. In any timeslot, only one source can attempt a transmission to the base station. Instead of weighted sum AoI, we are interested in minimizing the time-average of *general non-decreasing cost functions* of AoI, summed over all sources. Examples of such functions include $f(x) = 2^x$, $f(x) = \log(x)$, $f(x) = \mathbb{1}_{\{x \geq 10\}}$, etc. See Fig.2-1 for examples. We develop a restless mutli-armed bandit formulation for the problem and use a Whittle Index based approach to find low complexity scheduling policies that have good performance.

The rest of the chapter is organized as follows. In Section 2.1, we describe the general system model. In Section 2.2, we describe the equivalent restless multi-armed bandit formulation and discuss why we use the Whittle Index approach to solve the problem. In Section 2.3, we discuss the functions of age problem with reliable channels, develop the Whittle Index solution for this setting, and also prove key structural properties that an optimal policy must satisfy. In Section 2.4, we find the Whittle Index policy for the functions of age problem with unreliable

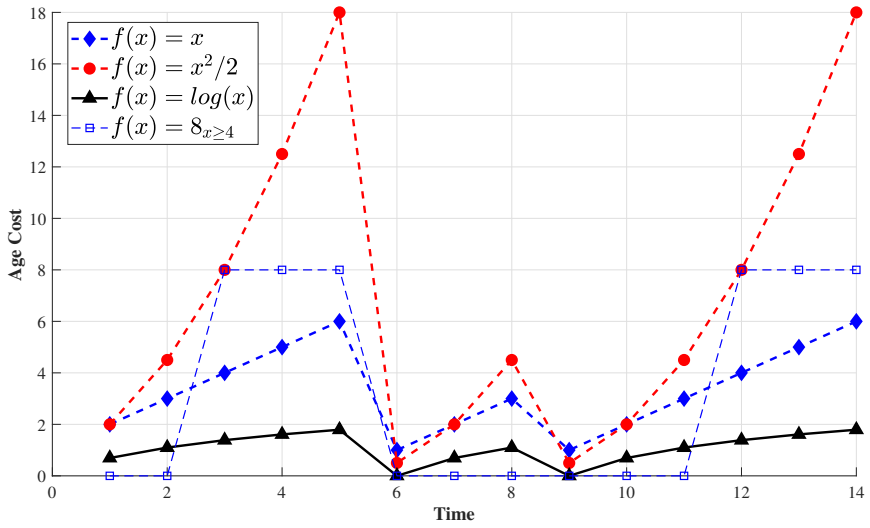


Figure 2-1: Linear, quadratic, logarithmic and indicator cost functions for a sample age process. The linear process tracks the actual values of AoI.

channels. In Section 2.5, we provide simulation results that verify our theoretical results. In Section 2.6, we show that the problem of minimizing monitoring error for linear time-invariant systems when observing them over a wireless channel is equivalent to minimizing functions of AoI. We also show a similar result for monitoring symmetric Markov chains over a wireless channel. This shows the direct applicability of our Whittle framework to a large class of wireless monitoring problems.

2.1 Model

Consider a single-hop wireless network with N active sources generating real-time status updates that need to be sent to a base station. We consider a slotted system in which each source takes a single time-slot to transmit an update to the base station. Due to interference, only one of the sources can transmit in any given time-slot.

For every source i, the age of information at the base station $A_i(t)$ measures

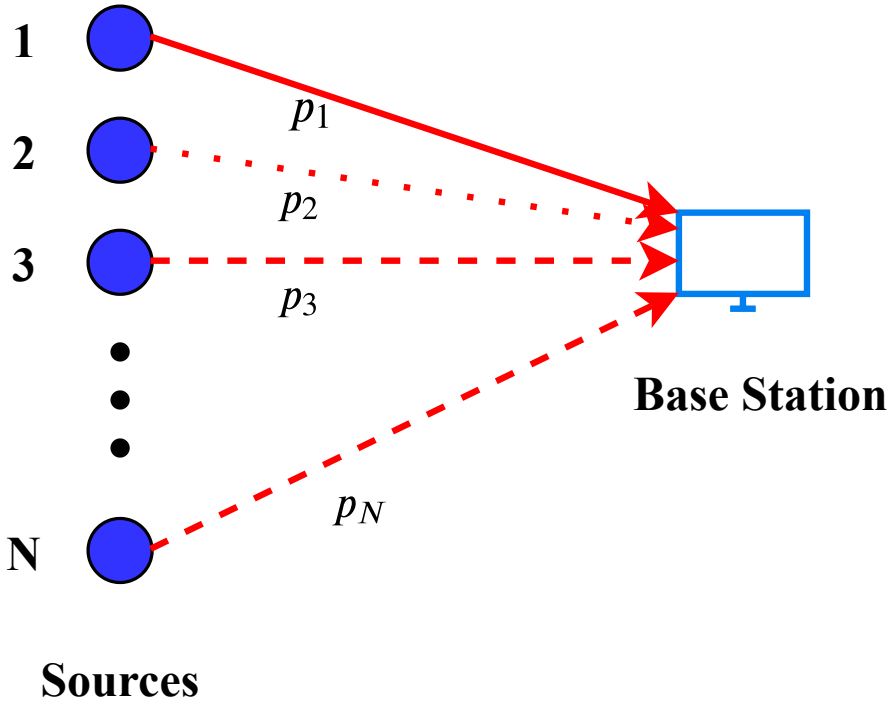


Figure 2-2: N sources transmitting updates to a base station over a wireless channel, with different reliabilities.

the time elapsed since it received a fresh information update from the source. We assume active sources, i.e. in any time-slot, sources can generate fresh updates at will. Let $s(t)$ be the source activated in time-slot t and $u_i(t)$ be a Bernoulli random variable with parameter p_i that denotes channel reliability between the i^{th} source and the base station. Then, we have

$$A_i(t+1) = \begin{cases} A_i(t)+1, & \text{if } s(t) \neq i \text{ or } u_i(t) = 0, \\ 1, & \text{if } s(t) = i \text{ and } u_i(t) = 1. \end{cases} \tag{2.1}$$

We consider general cost functions of age as our metric of interest. For each source i, let $f_i(\cdot)$ denote a positive *non-decreasing* cost function.

Let π be a scheduling scheme that decides which sources to schedule in every time-slot. The age process $A_i(t)$ depends on π and the channel processes. Then, the expected average cost of age for source i is given by

$$C_i^{\text{ave}}(\pi) \triangleq \limsup_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T} f_i(A_i^{\pi}(t))\right], \tag{2.2}$$

where $A_i^{\pi}(t)$ is age process for the i^{th} source under policy π.

Our goal is find a schedule π that minimizes the sum of average costs of age of sources. Let Π denote the set of causal scheduling policies, then we want to solve the following optimization problem

$$C^* = \min_{\pi\in\Pi} \sum_{i=1}^{N} C_i^{\text{ave}}(\pi), \tag{2.3}$$

where C^* is minimum average cost and π^* is an optimal scheduling policy.

2.2 Restless Multi-Armed Bandit Formulation

In this section, we show that scheduling to minimize such a metric can be reformulated as a restless multi-armed bandit (RMAB).

Consider a restless multi-armed bandit problem with N arms. The state space for every arm i is the set of positive integers $\mathbb{Z}^+$. The state evolution of the arm depends on whether it is currently active or not. Let the state of arm i at time t be denoted by $A_i(t)$. If arm i is active in time-slot t then the state evolution is given by

$$A_i(t+1) = \begin{cases} A_i(t)+1, & \text{w.p. } 1-p_i \\ 1, & \text{w.p. } p_i. \end{cases} \tag{2.4}$$

If the arm is not active in time-slot t, then the state evolution is given by

$$A_i(t+1) = A_i(t) + 1. \tag{2.5}$$

For every arm i, there is a cost function $f_i : \mathbb{Z}^+ \to \mathbb{R}^+$ which maps the states of the arm to their associated costs. Thus, the cost of a state $\boldsymbol{x} \in \mathbb{Z}^{+^N}$ is given by $\sum_{i=1}^{N} f_i(x_i)$, where $\boldsymbol{x}$ is a vector of ages and x_i is the age (state) of the i^{th} source. Given that only one arm can be activated in any time-slot, the goal of the RMAB framework is to find a scheduling policy that minimizes the total time average cost of running this system.

This establishes the equivalence between the functions of age problem discussed earlier and a corresponding restless multi-armed bandit. Observe that the "restless" part of our construction cannot be dropped, since the states of the arms do not freeze when they are not active and there is no way to reformulate our problem as a simple (non-restless) multi-armed bandit problem. If that were the case, we could have found an optimal policy by solving for the Gittins index [95]. However, finding optimal policies for restless bandits is much harder. The usual approach is to find the Whittle Index policy which provides good performance under certain conditions, namely *indexability* of the RMAB problem.

In [31] and [15], the authors develop three methods to solve the minimum age scheduling problem. First, they look at stationary randomized policies, where a source i is scheduled at random with a fixed probability p_i. They find a stationary randomized policy that is factor-2 optimal for weighted sum AoI. However, this result does not hold for general functions: even the best stationary randomized policies in our setting can lead to an unbounded overall cost, despite there being very simple policies that have bounded cost. We demonstrate this with a simple example.

Consider two identical sources with cost functions given by $f(x) = 3^x$ and reliable channels, i.e $p_1 = p_2 = 1$. Any stationary randomized policy schedules at least one of the sources with probability less than or equal to 0.5. For this source, the average cost is lower bounded by $\lim_{T\to\infty} \sum_{t=1}^{T} (3^t)\frac{0.5^T}{T}$ since with probability at least 0.5, it does not get to transmit and its age increases by 1 in every time-slot.

Observe that this lower bound goes to ∞ and hence the average cost also goes to ∞ for all stationary randomized policies. On the other hand, a simple round-robin scheme that schedules the two sources in alternating time-slots guarantees bounded cost for both sensors. Thus, stationary randomized policies can be infinitely worse than the optimal policy for the functions of age problem.

The second method developed for age-based scheduling in [31, 15] uses a Max-Weight approach. The authors design a quadratic Lyapunov function for the weighted sum of linear functions of AoI and find the max-weight policy - the policy that maximizes the amount of negative drift in the Lyapunov function in every time-slot. Performance guarantees for the max-weight policy crucially rely on the fact that there exists a stationary randomized policy that is factor-2 optimal for linear functions of age. Since this is not the case for general functions of age, we cannot develop similar performance bounds using the Max-Weight policy for the general functions of age problem.

This finally leaves us with the third method - using a Whittle Index based approach. In the following two sections, we use the RMAB formulation to establish indexability for the functions of age problem and derive a Whittle Index policy. We also show that for the case with 2 sources and reliable channels, the Whittle index policy is exactly optimal. This is a novel result since the optimality of Whittle Index policies is typically shown either only asymptotically, or in symmetric settings for finite systems. On the other hand, our optimality result holds for two asymmetric sources.

2.3 Reliable Channels

We first look at the problem with reliable channels between the sources and the base station. This leads to simpler analysis and a better understanding of the problem. Consider the setup described in Section I with channel reliability $u_i(t) =$

1, for all i and t. In other words, the probability of success $p_i = 1, \forall i$.

In Section 2.2, we showed that the functions of age minimization problem is equivalent to a restless multi-armed bandit problem. Next, we use a Whittle Index based approach to try and solve the problem.

The first step in the Whittle Index approach is to formulate the *decoupled problem*, where we consider a single arm in isolation with a fixed charge required to activate the arm.

Definition *Decoupled Problem*
Consider a single arm with the state space $\mathbb{Z}^+$ and an associated non-decreasing cost function $f : \mathbb{Z}^+ \rightarrow \mathbb{R}^+$. Let the state of the arm be $A(t)$. Its evolution is given by

$$A(t+1) = \begin{cases} A(t)+1, & \text{if not active at time t} \\ 1, & \text{otherwise.} \end{cases}$$

There is a strictly positive activation charge C to be paid in every time-slot that the arm is pulled.

Our goal is to find a scheduling policy that minimizes the time-average cost of running this system. Assuming that the cost function $f(\cdot)$ is non-negative and non-decreasing, we solve the decoupled problem using dynamic programming. The case when the activation charge is set to zero is trivial. The optimal policy is to always activate the arm. So, we consider C to be strictly positive. The single source decoupled problem has also been solved in a slightly different setting in [26].

Theorem 1. *The optimal policy for the decoupled problem is a stationary threshold policy. Let H satisfy*

$$f(H) \leq \frac{\sum_{j=1}^{H} f(j) + C}{H} \leq f(H+1). \tag{2.6}$$

Then, the optimal policy is to activate the arm at time-slot t if $A(t) \geq H$ and to let it rest otherwise. If no such H exists, the optimal policy is to never activate the arm.

Proof. See Appendix 2.8.A. □

Theorem 1 establishes that the optimal policy for the decoupled problem has a threshold structure. We now want to show that the *indexability* property also holds for the decoupled problem. The indexability property states that as the activation charge C increases from 0 to ∞, the set of states for which it is optimal to activate the arm decreases monotonically from the entire set $\mathbb{Z}^+$ to the empty set $\{\phi\}$.

Theorem 2. *The indexability property holds for the decoupled problem.*

Proof. See Appendix 2.8.B. □

The Whittle index approach states that if the decoupled problem satisfies the indexability property, we can formulate a heuristic index policy called the Whittle Index Policy that has good performance.

Definition *Whittle Index*
Consider the decoupled problem and denote by $W(h)$ the Whittle index in state h. Given indexability, $W(h)$ is the infimum charge C that makes both decisions (activate, not activate) equally desirable in state h. The expression for $W(h)$ is given by

$$W(h) = hf\big(h+1\big) - \sum_{j=1}^{h} f(j). \tag{2.7}$$

Observe that using (2.6), $C = W(h)$ is the minimum value of the activation charge that makes both actions equally desirable in state h. This gives us the expression for the Whittle index.

Let $W_i(x) := xf_i\big(x+1\big) - \sum_{j=1}^{x} f_i(j)$ represent the index function for the i^{th} decoupled problem. By the monotonicity of $f_i(\cdot)$, it is easy to see that the functions $W_i(\cdot)$ are also monotonically non-decreasing. This is because $W_i(h) - W_i(h-1) = h\big(f_i(h+1) - f_i(h)\big) \geq 0, \forall h$ since $f_i(\cdot)$ is non-decreasing. Using these functions, we define the Whittle Index Policy.

Definition *Whittle Index Policy*
Let $\pi^W(t)$ be the action taken by the Whittle Index Policy at time t. Then $\pi^W(t)$ is given by

$$\begin{aligned}\pi^W(t) &= \arg\max_{1\leq i\leq N}\Big\{W_i\big(A_i(t)\big)\Big\} \\ &= \arg\max_{1\leq i\leq N}\Big\{A_i(t)f_i\Big(A_i(t)+1\Big) - \sum_{j=1}^{A_i(t)} f_i(j)\Big\}.\end{aligned} \tag{2.8}$$

Consider the case when cost functions are weighted linear functions of AoI,

i.e let $f_i(A_i(t)) = w_i A_i(t)$, with positive weights w_i. This is the setting considered in [31] and [15]. The Whittle Index for source i is then given by $W_i(A_i(t)) = w_i(A_i^2(t) + A_i(t))/2$. This is the same as the Whittle index found in [31], where the authors showed that the Whittle policy is optimal for symmetric settings when all the weights are equal. We also establish that for $N = 2$, the Whittle index policy is optimal even for asymmetric settings.

Theorem 3. *For the functions of age problem with reliable channels and two sources, the Whittle index policy is exactly optimal.*

Proof. See Appendix 2.8.F. □

This is an atypical result for restless multi-armed bandit problems which typically only have optimality results for symmetric or asymptotic settings. Our result is valid for finite ($N = 2$) asymmetric settings. To the best of our knowledge, ours was the first work to prove such a result for a restless multi-armed bandit problem. Next, we discuss some general properties that an optimal policy satisfies even for larger size systems. These properties help us establish the optimality of the Whittle index policy for $N = 2$ and provide insight as to why the Whittle index policy has good performance in general.

2.3.A Properties of an Optimal Policy

For the functions of age problem, a policy is stationary if it depends only on the current values of age. A cyclic policy is one that repeats a finite sequence of actions in a fixed order. We define the space of policies that are stationary and periodic.

Definition *Stationary Cyclic Policies*
A stationary cyclic policy is a stationary policy that cycles through a finite subset of points in the state space, repeating a fixed sequence of actions in a particular order.

In [30], the authors show that for reliable channels there exists an optimal policy that is stationary, cyclic and can be found by solving the minimum average cost cycle problem over a large graph.

We look at this cyclic policy and analyze its properties. If there are multiple such cycles, we consider a cycle with the shortest length. We denote the length of the cycle by T and age vectors on the cycle to be $\boldsymbol{x_1}, \ldots, \boldsymbol{x_T}$. Let the corresponding scheduling decisions be $d_1, \ldots, d_T$. This implies that for state $\boldsymbol{x_k}$, taking action d_k leads to the state $\boldsymbol{x_{k+1}}$, where the subscripts cycle back to $1, 2, \ldots$ after T.

We establish an important structural property that such an optimal policy must satisfy, which we call the *strong-switch-type* property. We call the policies that satisfy this property *strong-switch-type* policies.

Definition *Strong-switch-type Policies*
Consider a stationary policy π that maps every point in the state space $\mathbb{Z}^{+^N}$ to the set of arms $\{1, \ldots, N\}$. We say that such a policy is strong-switch-type if

$$\pi(x_1, \ldots, x_N) = i$$

implies

$$\pi(x'_1, \ldots, x'_N) = i,$$

for all $\boldsymbol{x}$ and $\boldsymbol{x'}$ such that $x'_i \geq x_i$ and $x'_j \leq x_j, \forall j \neq i$.

In words, the strong-switch-type property implies that if a policy decides to activate arm i for a state vector $\boldsymbol{x}$, then for a state vector $\boldsymbol{x}'$ with a higher age for the i^{th} source and lower ages for all the other sources, it still decides to activate source i. Note that our definition of strong-switch-type policies is a stronger version of the switch-type policies introduced in [20].

Theorem 4. *For the functions of age problem with reliable channels, no state-action pairs that are a part of the shortest length optimal cyclic policy can violate the strong-switch-type property.*

Proof. See Appendix 2.8.C. □

We can prove this result for general values of N. However, to extend the strong-switch-type property over the entire state-space, we consider systems with up to three sources.

Theorem 5. *There exists an optimal stationary policy for the functions of age problem with reliable channels and up to three sources that has the strong-switch-type property over the entire state-space.*

Proof. We have already established that points on the minimum average cost cycle satisfy the strong-switch-type property. In Appendix 2.8.D, we extend this policy over the entire state space while maintaining the strong-switch property to obtain a well defined stationary policy. □

While we prove this result for up to three source and reliable channels, we believe that the strong-switch-type property is a natural property that an optimal policy must have in general, due to monotonicity of cost functions.

We now define the space of policies that can be found as a result of the Whittle Index based approach.

Definition *Index Policies*
Consider a stationary policy π that maps every point in the state space $\mathbb{Z}^{+^N}$ to the set of arms $\{1,\ldots,N\}$. We say that such a policy is an index policy if $\exists\, F_1(\cdot), ..., F_N(\cdot)$ such that

$$\pi(x_1,\ldots,x_N) = \arg\max_{1\leq i\leq N}\left\{F_i(x_i)\right\}$$

for all $\boldsymbol{x}$, where $F_i : \mathbb{Z}^+ \to \mathbb{R}$ are monotonically non-decreasing functions for all i.

Observe that if F_i are the same as W_i in the above definition, then we get back the Whittle Index Policy. Also, note that an index policy always satisfies the strong-switch-type property by definition. This is because the index functions $F_i(\cdot)$ are monotonically non-decreasing. We now show that index policies are in fact the same as strong-switch-type policies.

Theorem 6. *For the functions of age problem, every policy that is strong-switch-type is also an index policy.*

Proof. The proof is based on induction on the number of sources. We assume that every strong-switch-type policy can be represented as an index policy for systems with N sources. Using this fact, we show that strong-switch-type policies can also be represented as index policies for systems with $N+1$ sources. We also

show that the two types of policies are equivalent for the single source decoupled problem, thus completing the proof. The details are in Appendix 2.8.E. □

An important point to notice is that while we use the reliability of channels in the proof of Theorem 5, we do not use any such condition for the proof of Theorem 6. Thus, strong-switch-type policies are equivalent to index policies regardless of channel connectivity.

Theorems 5 and 6 together imply the following corollary.

Corollary 1. *For the functions of age problem with reliable channels and up to three source, there exists a stationary optimal policy that is an index policy.*

In other words, there exists an optimal policy that looks like the Whittle Index policy in that the arm to be activated has the maximum value among *monotone index functions* that take as arguments only the states of individual arms. This hints at why the performance of Whittle Index policies may be close to optimal.

Observe that the Whittle Index policy would be optimal in general if we could show that it achieves a cost that is the minimum cost among the space of index policies and that the strong-switch-type property holds for some optimal policy. We show that this is indeed the case for $N = 2$. However, we later provide an example that shows that the Whittle policy is not optimal, but only close to optimal, for $N = 4$.

We leave the question of whether the Whittle index policy is at most a constant factor away from optimal in general to future work. We believe that the structural properties introduced here provide a recipe to proving constant factor optimality of the Whittle index policy, even for general bandit problems with similar underlying structure.

2.4 Unreliable Channels

We now consider independent Bernoulli channels between every source and the base station, with probability of success p_i for source i. We derive a Whittle index in this setting and establish indexability of the RMAB problem by enforcing a bounded cost condition on the functions $f_i(\cdot)$.

An important fact to notice is that monotonicity in itself is not sufficient to ensure that the system has finite average cost even for $N = 1$, in the case of unreliable channels. Consider a single source case where $f(a) = 3^a$ and the probability of success $p = 0.5$. If the source attempts a transmission in every time-slot, the expected average cost satisfies

$$\limsup_{T\to\infty} \sum_{t=1}^{T} (3^t)\frac{0.5^T}{T} \leq \limsup_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T} 3^{A(t)}\right], \tag{2.9}$$

since with probability 0.5, the transmission fails and age increases by 1 in every time-slot. However, observe that the summation on the left goes to infinity and thus the expected average cost goes to infinity. This happens despite the source attempting a transmission in every time-slot. To prevent such a situation from happening we enforce the following *bounded cost* condition on the age cost functions f_i in addition to monotonicity

$$\sum_{h=1}^{\infty} f_i(h)(1-p_i)^h < \infty. \tag{2.10}$$

It can be shown that this condition ensures that the single source case has bounded cost. We define the decoupled problem in this case as follows:

Definition *Decoupled Problem*
Consider a single arm with the state space $\mathbb{Z}^+$, probability of success p and

an associated non-decreasing cost function $f : \mathbb{Z}^+ \to \mathbb{R}^+$ that satisfies the *bounded cost* condition. Let the state of the arm be $A(t)$. If the arm is active at time t, its evolution is given by

$$A(t+1) = \begin{cases} A(t)+1, & \text{w.p. } 1-p \\ 1, & \text{w.p. } p. \end{cases}$$

If the arm is not active in time-slot t, then the state evolution is given by

$$A(t+1) = A(t)+1.$$

There is a strictly positive activation charge C to be paid in every time-slot that the arm is pulled.

As before, our goal is to find a scheduling policy that minimizes the time-average cost of running this system.

Theorem 7. *The optimal policy for the decoupled problem is a stationary threshold policy. Let H satisfy*

$$\begin{aligned} p^2(H-1)\left(\sum_{k=H}^{\infty} f(k)(1-p)^{k-H}\right) - p\left(\sum_{j=1}^{H-1} f(j)\right) \\ \leq C \\ \leq p^2 H\left(\sum_{k=H+1}^{\infty} f(k)(1-p)^{k-H-1}\right) - p\left(\sum_{j=1}^{H} f(j)\right) \end{aligned} \tag{2.11}$$

Then, the optimal policy is to activate the arm at time-slot t if $A(t) \geq H$ and to let it rest otherwise. If no such H exists, the optimal policy is to never activate

the arm.

Proof. See Appendix 2.8.G. □

Observe that taking the limit as $p \to 1$ in Theorem 7, we get back the threshold policy for reliable channels derived in Theorem 1. We now establish indexability and derive the functional form of the Whittle Index.

Theorem 8. *The indexability property holds for the decoupled problem. Denote by $W(h)$ the Whittle index in state h. Given indexability, $W(h)$ is the infimum charge C that makes both decisions (activate, not activate) equally desirable in state h. The expression for $W(h)$ is given by*

$$W(h) = p^2 h\Big(\sum_{k=1}^{\infty} f(k+h)(1-p)^{k-1}\Big) - p\Big(\sum_{j=1}^{h} f(j)\Big). \tag{2.12}$$

Proof. See Appendix 2.8.H. □

Again, observe that taking the limit as $p \to 1$, we get back the Whittle Index derived in Section 2.3. Further, if we assume that the cost functions are weighted linear functions of AoI, i.e. $f_i(A_i(t)) = w_i A_i(t)$ where all the weights are positive, then the index functions for the Whittle policy are given by $W_i(A_i(t)) = w_i p_i A_i(t)(A_i(t) + \frac{1+(1-p_i)}{1-(1-p_i)})/2$. This corresponds to the index policy developed in [31], where the authors showed that for symmetric settings when all the weights and channels probabilities are equal, the Whittle index policy is optimal.

2.5 Simulations

First, we compare the optimal policy, found using dynamic programming, with the Whittle index policy for two sources. We consider six different settings in total - 3 sets of functions, each with reliable and unreliable channels.

For settings A_1 and A_2, the cost functions are chosen to be $f_1(x) = 13x$ and $f_2(x) = x^2$. In A_1, we consider reliable channels, i.e. $p_1 = p_2 = 1$. In A_2, we consider unreliable channels, specifically $p_1 = 0.9$ and $p_2 = 0.5$. For settings B_1 and B_2, the cost functions are chosen to be $f_1(x) = x^2$ and $f_2(x) = 3^x$. In B_1, we consider reliable channels, i.e. $p_1 = p_2 = 1$. In B_2, we consider unreliable channels, specifically $p_1 = 0.65$ and $p_2 = 0.8$. For settings C_1 and C_2, the cost functions are chosen to be $f_1(x) = x^3/2$ and $f_2(x) = 10\log(x)$. In C_1, we consider reliable channels, i.e. $p_1 = p_2 = 1$. In C_2, we consider unreliable channels, specifically $p_1 = 0.55$ and $p_2 = 0.75$. Simulation results are presented in Table 2.1.

Setting	Optimal Cost	Whittle Index Cost
A_1 (reliable)	21.95	21.95
A_2 (unreliable)	36.12	36.28
B_1 (reliable)	8.48	8.48
B_2 (unreliable)	23.16	23.37
C_1 (reliable)	5.69	5.69
C_2 (unreliable)	21.54	21.54

Table 2.1: Cost of the Whittle index policy and the optimal dynamic programming policy for 2 sources.

We find the optimal cost for each setting using finite horizon dynamic programming over a horizon of 500 time-slots. For reliable channels, we find the cost of the Whittle index policy by simply implementing it once over 500 time-slots. For unreliable channels, we estimate the expected Whittle index cost by averaging the performance of the Whittle policy over 500 independent runs.

Observe that the Whittle index policy is exactly optimal when the channels are reliable, as expected from our theoretical results. The expected cost for the

Whittle index policy is very close to the optimal cost for unreliable channels as well. Also, for the same set of functions, having unreliable channels increases the cost compared to reliable channels, as expected.

Next, we compare the optimal policy with the Whittle index policy for more than two sources. Simulation results are presented in Table 2.2.

For settings D_1 and D_2, we consider 3 sources. The cost functions are chosen to be $f_1(x) = x^2$, $f_2(x) = 3^x$ and $f_3(x) = x^4$. In D_1, we consider reliable channels, i.e. $p_1 = p_2 = p_3 = 1$. In D_2, we consider unreliable channels, specifically $p_1 = 0.66$, $p_2 = 0.8$ and $p_3 = 0.75$.

For settings E_1 and E_2, we consider 4 sources. The cost functions are chosen to be $f_1(x) = x^3$, $f_2(x) = 2^x$, $f_3(x) = 15x$ and $f_4(x) = x^2$. In E_1, we consider reliable channels, i.e. $p_1 = p_2 = p_3 = 1$. In E_2, we consider unreliable channels, specifically $p_1 = 0.7$, $p_2 = 0.9$, $p_3 = 0.67$ and $p_4 = 0.8$.

No. of Sources	Setting	Optimal Cost	Whittle Index Cost
3	D_1 (reliable)	44.23	44.23
	D_2 (unreliable)	161.19	161.39
4	E_1 (reliable)	73.36	73.36
	E_2 (unreliable)	129.02	130.94
4	F_1 (reliable)	87.66	88.27
	F_2 (unreliable)	158.35	159.81

Table 2.2: Cost of the Whittle index policy and the optimal policy for more than 2 sources.

For settings F_1 and F_2, we consider 4 sources. The cost functions are chosen to be $f_1(x) = x^3$, $f_2(x) = e^x$, $f_3(x) = 15x$ and $f_4(x) = x^2$. In F_1, we consider reliable channels, i.e. $p_1 = p_2 = p_3 = 1$. In F_2, we consider unreliable channels, specifically $p_1 = 0.8$, $p_2 = 0.85$, $p_3 = 0.75$ and $p_4 = 0.66$.

We observe that the cost of the Whittle index policy is the same as that obtained using dynamic programming for settings D_1 and E_1. However, for setting

F_1, we observe a small gap in performance between the two policies, thus giving us an example that shows that the *Whittle index policy need not be optimal*, in general. We also verify that the optimal policy found using dynamic programming follows a cyclic pattern that satisfies the strong-switch-type property and is distinct from the Whittle index policy. This is also in line with our discussion on structural properties.

We also note that computing the optimal policy using dynamic programming becomes progressively harder in terms of space and time complexity for larger values of N, as the state-space to be considered grows exponentially with N. The Whittle index policy, on the other hand, is very easy to compute and implement with only a linear increase in space and time complexity with the number of sources. Also, as is evident from simulations, the performance of the Whittle policy is close to optimal in every setting considered, thus making it a very good low complexity heuristic.

2.6 Applications

In this section, we will apply the framework we have developed to two problems in remote monitoring and control to show that optimizing general functions of AoI arise naturally in many practical settings.

2.6.A Monitoring LTI systems

First, we consider the remote monitoring of linear time-invariant (LTI) systems over a wireless channel. Suppose that there are N such systems, where the ith system evolves over time as follows

$$x_i(t+1) = G_i x_i(t) + w_i(t), \tag{2.13}$$

where $x_i(t) \in \mathbb{R}^{d_i}, G_i \in \mathbb{R}^{d_i \times d_i}$ is the system matrix and $w_i(t) \sim \mathcal{N}(0, \Sigma_i)$ is multivariate zero-mean Gaussian noise, i.i.d. across time. We further assume that the noise increments $w_i(t)$ are independent across sources, so their evolution is decoupled.

Suppose that a central agent wants to monitor the state of each of the N systems with as little monitoring error as possible. However, due to wireless interference constraints, it can only observe the state of one system at any given time-slot. How should the agent design a wireless scheduling policy that minimizes expected monitoring error?

Let $\hat{x}_i(t)$ represent the maximum likelihood estimate of the state of the ith system at the monitor at any given time-slot t, given past observations. We define monitoring error for the ith system as

$$e_i(t) \triangleq \mathbb{E}\left[\left\|x_i(t) - \hat{x}_i(t)\right\|_2^2\right]. \tag{2.14}$$

The following theorem relates the expected monitoring error of the ith system to its AoI. Specifically, we compute the expected error if the ith system has not been observed for the last Δ time-slots.

Theorem 9. *Suppose that the ith system evolves according to* (2.13)*. Further suppose that the monitor last observed the state of the system at time $t = \tau$. Then, the expected monitoring error for the ith system at time $t = \tau + \Delta$ is given by*

$$\begin{aligned} e_i(\tau+\Delta) &= \mathbb{E}\left[\left\|x_i(\tau+\Delta) - \hat{x}_i(\tau+\Delta)\right\|_2^2\right] \\ &= \sum_{k=0}^{\Delta-1} Tr\left((G_i^k)^T (G_i^k)\Sigma_i\right) \triangleq f_i(\Delta). \end{aligned} \tag{2.15}$$

Proof. See Appendix 2.8.I. □

Using this observation, we can establish an equivalence between minimizing monitoring error and minimizing functions of AoI. To find the scheduling policy π that minimizes expected time-average monitoring error, we need to solve the following optimization problem

$$\min_{\pi\in\Pi} \limsup_{T\to\infty} \frac{1}{T}\mathbb{E}\Big[\sum_{t=1}^{T}\sum_{i=1}^{N} e_i(t)\Big], \tag{2.16}$$

where $e_i(t)$ is defined as in (2.14). This optimization problem is equivalent to solving the following functions of AoI problem

$$\min_{\pi\in\Pi} \limsup_{T\to\infty} \frac{1}{T}\Big[\sum_{t=1}^{T}\sum_{i=1}^{N} f_i(A_i(t))\Big], \tag{2.17}$$

where $A_i(t)$ is the AoI of the ith system and the functions $f_i(\cdot)$ are as defined in (2.15).

We also show in Appendix 2.8.I that the functions $f_i(\cdot)$ are monotonically increasing, so we can indeed apply our Whittle index approach to solve this problem. The rate at which the functions $f_i(\cdot)$ increase depends on the eigenvalues of the system matrices G_i. If the largest eigenvalue of G_i lies inside (outside) the unit circle, then $f_i(\cdot)$ increases slower (faster) than a linear function. If the largest eigenvalue of G_i lies on the unit circle, then $f_i(\cdot)$ increases linearly.

2.6.B Monitoring Markov Chains

Consider N symmetric two state Markov chains of the form drawn in Fig. 2-3 running in discrete-time. As for the previous example, we assume that only one system out of the N can be observed in any given time-slot. We denote the distribution of the ith Markov chain at time t by $x_i(t) \in \mathbb{R}^2$, where $x_i(t) = [1\ \ 0]$ if the Markov chain is in state 0 and $x_i(t) = [0\ \ 1]$ if the Markov chain is in state 1.

We assume that the base station knows the transition probability q_i and the

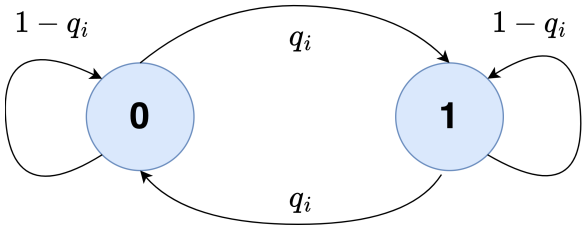


Figure 2-3: Symmetric two-state Markov chain, representing the state of the ith process.

transition matrix

$$Q_i = \begin{bmatrix} 1-q_i & q_i \\ q_i & 1-q_i \end{bmatrix}$$

associated with the ith Markov chain and uses this to maintain the estimated distribution of the ith chain, based on the most recent observation. Suppose that the base station knew that the ith Markov chain had the distribution $x_i(\tau)$ at time τ. Using the transition matrix Q_i for the ith chain, the base station can compute the distribution of the Markov chain at time $\tau + \Delta$ given the information at time τ. We denote this estimated distribution of the actual state by $\hat{x}_i(\tau + \Delta)$ and it is given by

$$\hat{x}_i(\tau + \Delta) = x_i(\tau) \begin{bmatrix} 1-q_i & q_i \\ q_i & 1-q_i \end{bmatrix}^{\Delta} = x_i(\tau) Q_i^{\Delta}. \tag{2.18}$$

We are interested in minimizing the monitoring error, defined as a notion of distance between the estimated distribution and the actual state of the Markov chain. We define error for the ith system as follows -

$$e_i(t) = \mathbb{E}\Big[D\big(x_i(t) || \hat{x}_i(t)\big) \Big], \tag{2.19}$$

where D is a notion of divergence between the two probability distributions. In this work, we will discuss our results for Kullback-Liebler (KL) divergence and total variation (TV) distance, however, the general ideas should work for other

divergences as well. The KL divergence for discrete distributions is defined as

$$D_{KL}(P||Q) = -\sum_{x\in\mathcal{X}} P(x)\log(\frac{P(x)}{Q(x)}).$$

The total variation (TV) distance for discrete distributions is defined as

$$D_{TV}(P||Q) = \sum_{x\in\mathcal{X}} \frac{1}{2}\big|P(x) - Q(x)\big|.$$

The following theorem relates the expected monitoring error of the ith system to its AoI. Specifically, we compute the expected error if the ith system has not been observed for the last Δ time-slots.

Theorem 10. *Suppose that the ith system evolves according to Markov chain in Fig. 2-3. Further suppose that the monitor last observed the state of the system at time $t = \tau$. Then, the expected monitoring error for the ith system at time $t = \tau + \Delta$ is given by*

$$\begin{aligned} e_i(\tau+\Delta) &= \mathbb{E}\Big[D\big(x_i(\tau+\Delta)||\hat{x}_i(\tau+\Delta)\big)\Big] \\ &= \begin{cases} H\big([Q_i^\Delta]_{00}\big), & \text{if } D \text{ is KL Divergence,} \\ 2[Q_i^\Delta]_{00}\big(1-[Q_i^\Delta]_{00}\big), & \text{if } D \text{ is TV Distance.} \end{cases} \\ &= f_i(\Delta). \end{aligned} \tag{2.20}$$

Here $[Q_i^\Delta]_{00}$ is the top diagonal element of the transition matrix raised to the power Δ, i.e. Q_i^Δ and $H(q) \triangleq -q\log(q) - (1-q)\log(1-q)$ is the binary entropy function.

Proof. See Appendix 2.8.J. □

Using the result above, it is straightforward to establish an equivalence between minimizing monitoring error for Markov chains and minimizing functions of AoI. As for the case with the LTI systems, we further show in Appendix 2.8.J that the functions $f_i(\cdot)$ are monotonically increasing, so we can indeed apply our Whittle index approach to solve this problem.

An interesting observation for the KL divergence case is that the monitoring error cost ends up being the entropy of the estimated distribution of the Markov chain. This can be interpreted as the amount of uncertainty that the base station has about the Markov chain, which increases with the number of time-slots that the chain remains unobserved. We use this Markov model and our Whittle framework to solve a robotics problem involving time-varying multi-agent occupancy grid mapping in Chapter 4.

2.7 Summary

In this chapter, we presented the problem of minimizing functions of age of information over a wireless broadcast network. We used a restless multi-armed bandit approach to establish indexability of the problem and found the Whittle index policy. For the case with two sources and reliable channels, we were able to show that the Whittle index policy is exactly optimal. We also established structural properties of an optimal policy, for the case with reliable channels. These properties hint at why the performance of the Whittle index policy is close to optimal in general.

In Chapter 3, we extend our framework to consider unknown, possibly time-varying functions of Age of Information using ideas from online learning. In Chapter 4, we further extend our Whittle framework to consider computation communication trade-offs and apply our results to problems in multi-agent robotics. An interesting direction of future work involves proving constant factor optimal-

ity of the Whittle index policy in general, using the structural properties developed in this chapter.

2.8 Appendix

2.8.A Proof of Theorem 1

Consider the decoupled problem described in Section 2.3. Let $u(t)$ be an indicator variable that denotes whether the arm is pulled or not at time t. Under a scheduling policy π that specifies the value of $u(t)$ for all instants of time, the average cost is given by

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \left[f\left(A^{\pi}(t)\right) + Cu^{\pi}(t) \right]. \tag{2.21}$$

We want to find a policy that minimizes this cost over the space of all policies. Let $S : \mathbb{Z}^{+} \to \mathbb{R}$ denote the differential cost-to-go function for this problem, let $u : \mathbb{Z}^{+} \to \{1, 0\}$ be the stationary optimal policy and let λ denote the optimal cost. Then, the Bellman equations are given by

$$S(h) = f(h) + \min_{u(h)\in\{1,0\}} \{C, S(h+1)\} - \lambda, \forall h \in \mathbb{Z}^{+}. \tag{2.22}$$

Without loss of generality we set $S(1) = 0$. Assume that the optimal policy has a threshold structure, i.e. there exists H such that it is optimal to pull the arm ($u(h) = 1$) for all states $h \geq H$ and let it rest otherwise ($u(h) = 0$). If this the case, then the Bellman equations reduce to

$$S(h) = f(h) + C - \lambda, \forall h \geq H. \tag{2.23}$$

Using the monotonicity of $f(\cdot)$, we conclude that $S(h+1) \geq S(h), \forall h \geq H$. We will use this fact later. For the state $H-1$, we get

$$\begin{aligned} S(H-1) &= f\big(H-1\big) - \lambda + S(H) \\ &= f\big(H-1\big) - \lambda + f(H) - \lambda + C. \end{aligned} \tag{2.24}$$

Repeating this k times, we get

$$S(H-k) = \sum_{j=0}^{k} f\big(H-j\big) - (k+1)\lambda + C, \tag{2.25}$$

for all k in $\{1, \ldots, H-1\}$. Observe that since we set $S(1) = 0$, we get

$$\lambda = \frac{\sum_{j=1}^{H} f(j) + C}{H}, \tag{2.26}$$

by putting $k = H-1$ in (2.25). Now assume that H further satisfies the relation given in Theorem 1, i.e.

$$f(H) \leq \frac{\sum_{j=1}^{H} f(j) + C}{H} \leq f(H+1). \tag{2.27}$$

Using (2.26), we can simplify (2.27) as

$$f(H) \leq \lambda \leq f(H+1). \tag{2.28}$$

Adding $C - \lambda$ to every term above, we get

$$\begin{aligned} & f(H) + C - \lambda \leq C \leq f(H+1) + C - \lambda \\ \implies & S(H) \leq C \leq S(H+1). \end{aligned} \tag{2.29}$$

Observe that we assumed $f(\cdot)$ to be non-decreasing. This combined with (2.28) and the Bellman equations (2.23) and (2.25) ensures that $S(\cdot)$ is also non-decreasing.

Thus, if there exists a state H that satisfies (2.27), then the threshold policy with threshold H satisfies the Bellman equations and is hence optimal.

The one thing that remains to be shown is the case in which we cannot find some H that satisfies (2.27). Consider the function $W : \mathbb{Z}^+ \to \mathbb{R}$ given by

$$W(h) = hf(h) - \sum_{j=1}^{h} f(j). \tag{2.30}$$

Observe that $W(h+1) - W(h) = h(f(h+1) - f(h)) \geq 0$ since $f(\cdot)$ is non-decreasing. Thus, $W(\cdot)$ is also non-decreasing. Also, by definition, $W(1) = 0$, while we had assumed that $C > 0$. Thus, $W(1) < C$. Now, if there exists some $h > 1$ such that $W(h) \geq C$, then we know that there also exists some H such that $W(H) \leq C \leq W(H+1)$ using monotonicity of $W(\cdot)$. Observe that this implies that there exists some H satisfying

$$Hf\big(H\big) - \sum_{j=1}^{H} f(j) \leq C \leq (H+1)f\big(H+1\big) - \sum_{j=1}^{H+1} f(j).$$

Rearranging and dividing by H, we get back (2.27). Thus, if there exists no H satisfying (2.27), then $W(h) < C, \forall h$.

Since $W(\cdot)$ is a bounded monotone sequence, it converges to a finite value. It is easy to see that this implies that $f(\cdot)$ is also bounded and hence converges. We set $\lambda = \lim_{h\to\infty} f(h)$ and the cost-to-go function $S(h)$ to be

$$S(h) = \sum_{j=h}^{\infty} \big(f(j) - \lambda\big) + C. \tag{2.31}$$

Clearly, $S(h)$ satisfies the recurrence relation

$$S(h) = f(h) - \lambda + S(h+1), \forall h. \tag{2.32}$$

By the monotonicity of $f(\cdot)$, we know that $f(h) \leq \lambda, \forall h$. Thus, using (2.31) we conclude that $S(h) \leq C, \forall h$. This implies that $S(\cdot)$ satisfies the Bellman equations, with the optimal policy being to never activate the arm. This completes our proof.

2.8.B Proof of Theorem 2

For $C = 0$, it is obvious that the optimal policy is to always activate the arm since there is no charge for activating it and the cost function is monotone and positive. For larger values of C, consider the function $W : \mathbb{Z}^+ \to \mathbb{R}^+$ given by

$$W(h) = hf(h) - \sum_{j=1}^{h} f(j). \tag{2.33}$$

Observe that since $f(\cdot)$ is non-decreasing, $W(\cdot)$ is also non-decreasing. This is because $W(h+1) - W(h) = h\big(f(h+1) - f(h)\big) \geq 0, \forall h$ since $f(\cdot)$ is non-decreasing. Also, by definition, $W(1) = 0$, while we had assumed that $C > 0$. Thus, $W(1) < C$. Now, if there exists some $h > 1$ such that $W(h) \geq C$, then we know that there also exists some H such that $W(H) \leq C \leq W(H+1)$ using monotonicity of $W(\cdot)$. Observe that this implies that there exists some H satisfying

$$Hf\big(H\big) - \sum_{j=1}^{H} f(j) \leq C \leq (H+1)f\big(H+1\big) - \sum_{j=1}^{H+1} f(j). \tag{2.34}$$

Rearranging and dividing by H, we get back (2.27).

Using this, we can relate the optimal threshold values to values of activation charge. Let C be such that it lies in the interval $\big[W(h), W(h+1)\big)$, then the optimal policy is of threshold type with the threshold at h. Observe that if W is strictly increasing then there can only be one such interval in which C can lie. If $W(\cdot)$ is non-decreasing, then there could be multiple such intervals in which C could lie. In this case, we choose the smallest h such that the condition holds.

The monotonicity of $W(\cdot)$ ensures that the the threshold value is also monotone non-decreasing with increasing values of C. When $W(h) < C, \forall h$, we choose h to be ∞, as done in Appendix 2.8.A. This completes the proof of *indexability* for the decoupled problem. Observe that $C = W(h+1)$ is the minimum value of the activation charge that makes both actions equally desirable in state h. This gives us the expression for the Whittle index.

2.8.C Proof of Theorem 4

We look at the optimal cyclical policy and analyze its properties. If there are multiple such cycles, we consider the cycle with the shortest length. We denote the length of the cycle by T, points on the cycle to be $\boldsymbol{x_1}, \ldots, \boldsymbol{x_T}$, and the average cost of this cycle to be C^*. The point $\boldsymbol{x_i}$ is an age vector in $\mathbb{Z}^{+^N}$, where $x_i^{(j)}$ represents the age of the j^{th} source. Let the corresponding scheduling decisions be $d_1, \ldots, d_T$. This implies that for age vector $\boldsymbol{x_k}$, taking action d_k leads to the age vector $\boldsymbol{x_{k+1}}$, where the subscripts cycle back to $1, 2, \ldots$ after T. Assume that there exists some pair of states in this cycle that violate the strong-switch-type property. If not, then our claim that the cycle satisfies the strong-switch-type property is true.

Without loss of generality, we assume that the pair of states that violates strong-switch is given by $\boldsymbol{x_1}$ and $\boldsymbol{x_k}$ for some $k \in \{2, \ldots, T\}$. This is because the cyclical policy is same up to cyclical permutations, so we can always ensure that one member of the violating pairs is at the front of the cycle. Also without loss of generality, we assume that $d_1 = 1$ and $d_k = 2$, since we can always relabel the sources. Observe that d_1 and d_k cannot be the same since they violate the strong-switch property. In fact, we know that $x_k^{(j)} \leq x_1^{(j)}, \forall j \neq 1$ and $x_k^{(1)} \geq x_1^{(1)}$. If the strong-switch property was satisfied, d_k must have been 1 since d_1 is 1.

We now construct two new cyclical policies out of which at least one has a better cost or the same cost but a smaller length compared to the original optimal policy. This contradicts our original assumption that the cycle we had started

with was the shortest policy with the lowest average cost. Starting with the state $\boldsymbol{x_1}$, we take the action d_1, following the original cycle up to $\boldsymbol{x_k}$. At $\boldsymbol{x_k}$, instead of taking action d_k, we take the action d_1 leading to the state $\boldsymbol{y_{k+1}}$. Observe that $\boldsymbol{y_{k+1}} \leq \boldsymbol{x_2}$, where the inequality is element-wise. Since $d_1 = 1$, we schedule source 1 at both $\boldsymbol{x_1}$ and $\boldsymbol{x_k}$, which guarantees that its age goes to 1. Thus, $x_2^{(1)} = y_{k+1}^{(1)} = 1$. Also, since $x_k^{(j)} + 1 \leq x_1^{(j)} + 1, \forall j \neq 1$ and none of the other sources are scheduled, so $y_{k+1}^{(j)} \leq x_2^{(j)}, \forall j \neq 1$. Together, this implies $\boldsymbol{y_{k+1}} \leq \boldsymbol{x_2}$.

Now, we follow the original cycle starting from $d_2, \ldots, d_T$. Action d_2 at state $\boldsymbol{y_{k+1}}$ leads to state $\boldsymbol{y_{k+2}}$ and so on, up to action d_T at state $\boldsymbol{y_{k+T-1}}$. Since the channels are reliable and $\boldsymbol{y_{k+1}} \leq \boldsymbol{x_2}$, it is easy to see that $\boldsymbol{y_{k+i}} \leq \boldsymbol{x_{i+1}}, \forall i \in \{1, \ldots, T-1\}$.

Also, observe that starting at $\boldsymbol{x_k}$, we have repeated an entire period of the original cycle, i.e. $d_1, \ldots, d_T$. Every source gets activated at least once during the original cycle, otherwise, its age goes to infinity, and we might as well remove it from the system. Starting at any age vector and following the actions $d_1, \ldots, d_T$ in sequence ensures that the state reached after these T steps equals $\boldsymbol{x_1}$. Thus, the actions $\{d_1, \ldots, d_{k-1}, d_1, \ldots, d_T\}$ and the age vectors $\{\boldsymbol{y_1}, \ldots, \boldsymbol{y_k}, \boldsymbol{y_{k+1}}, \ldots, \boldsymbol{y_{k+T-1}}\}$ form a cycle of length $k + T - 1$. Here $\boldsymbol{y_i} = \boldsymbol{x_i}, \forall i \in 1, \ldots, k$ and $\boldsymbol{y_{k+i}} \leq \boldsymbol{x_{i+1}}, \forall i \geq 1$. We denote the average cost of this cycle by C_1.

Now, we perform a cyclic permutation of the original optimal policy to get a new optimal policy with the actions $\{d_k, \ldots, d_T, d_1, \ldots, d_{k-1}\}$ and the corresponding states $\{\boldsymbol{x_k}, \ldots, \boldsymbol{x_T}, \boldsymbol{x_1}, \ldots, \boldsymbol{x_{k-1}}\}$. We repeat the process of constructing a new cyclical policy of length $2T - k + 1$ as done above, but using the new cyclic permutation of the optimal policy. That is, instead of choosing action d_1 at $\boldsymbol{x_k}$, we choose action d_k at $\boldsymbol{x_1}$.

This new cyclical policy consists of actions $\{d_k, \ldots, d_T, d_k, \ldots, d_T, d_1, \ldots, d_{k-1}\}$ and the corresponding age vectors $\{\boldsymbol{z_1}, \ldots, \boldsymbol{z_{2T-k+1}}\}$, forming a cycle of length $2T - k + 1$. Using exactly the same argument as earlier, it is easy to see that $\boldsymbol{z_j} = \boldsymbol{x_{j+k-1}}, \forall j \in \{1, \ldots, T-k+1\}$ and $\boldsymbol{z_j} \leq \boldsymbol{x_j}, \forall j \in \{T-k+2, \ldots, 2T-k+1\}$. We

denote the average cost of this cycle by C_2.

We know that the cost of the optimal policy C^* is minimum over the space of all policies, and hence less than or equal to cost of the first cyclical policy that we created C_1. Thus,

$$\begin{aligned}
& C^* \leq C_1 \\
\implies & \frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{N} f_j(x_t^{(j)}) \leq \frac{1}{k+T-1}\sum_{t=1}^{k+T-1}\sum_{j=1}^{N} f_j(y_t^{(j)}) \\
\leq & \frac{1}{k+T-1}\left(\sum_{t=1}^{k}\sum_{j=1}^{N} f_j(x_t^{(j)}) + \sum_{t=k+1}^{k+T-1}\sum_{j=1}^{N} f_j(y_t^{(j)})\right) \\
\leq & \frac{1}{k+T-1}\left(\sum_{t=1}^{k}\sum_{j=1}^{N} f_j(x_t^{(j)}) + \sum_{t=2}^{T}\sum_{j=1}^{N} f_j(x_t^{(j)})\right)
\end{aligned}$$

Simplifying this inequality, we get

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{N} f_j(x_t^{(j)}) \leq \frac{1}{k-1}\sum_{t=2}^{k}\sum_{j=1}^{N} f_j(x_t^{(j)}). \tag{2.35}$$

Similarly, the average cost of the optimal cycle * is also less than or equal to the average cost of the second cycle C_2. Thus,

$$\begin{aligned}
& C^* \leq C_2 \\
\implies & \frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{N} f_j(x_t^{(j)}) \leq \frac{1}{2T-k+1}\sum_{t=1}^{2T-k+1}\sum_{j=1}^{N} f_j(z_t^{(j)}) \\
& \leq \frac{1}{2T-k+1}\left(\sum_{t=k}^{T}\sum_{j=1}^{N} f_j(x_t^{(j)}) + \sum_{j=1}^{N} f_1(x_1^{(j)}) + \sum_{t=T-k+3}^{2T-k+1}\sum_{j=1}^{N} f_j(z_t^{(j)})\right) \\
& \leq \frac{1}{2T-k+1}\left(\sum_{t=k+1}^{T}\sum_{j=1}^{N} f_j(x_t^{(j)}) + \sum_{j=1}^{N} f_1(x_1^{(j)}) + \sum_{t=1}^{T}\sum_{j=1}^{N} f_j(x_t^{(j)})\right)
\end{aligned}$$

Simplifying this inequality, we get

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{N} f_j(x_t^{(j)}) \leq \frac{1}{T-k+1}\Bigg(\sum_{t=k+1}^{T}\sum_{j=1}^{N} f_j(x_t^{(j)}) + \sum_{j=1}^{N} f_1(x_1^{(j)})\Bigg).$$

Rearranging and simplifying again, we get

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{N} f_j(x_t^{(j)}) \geq \frac{1}{k-1}\sum_{t=2}^{k}\sum_{j=1}^{N} f_j(x_t^{(j)}). \tag{2.36}$$

From the analysis above, we observe that (2.35) and (2.36) must hold simultaneously. However, if that's the case then the inequalities cannot be strict. Also, observe that the cyclical policy given by actions $\{d_2, ..., d_k\}$ has average cost C_3 that satisfies

$$C_3 \leq \frac{1}{k-1}\sum_{t=2}^{k}\sum_{j=1}^{N} f_j(x_t^{(j)}).$$

This is because starting at state $\boldsymbol{x_2}$ and following the policy we end up at state $\boldsymbol{x_k}$, where using the exact same argument as earlier, taking action d_k leads us to a state $\boldsymbol{y_{k+1}}$ such that $\boldsymbol{y_{k+1}} \leq x_2$. The upper bound follows directly. Also, since (2.35) is tight, we get that

$$C_3 \leq \frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{N} f_j(x_t^{(j)}).$$

This is a contradiction, since if the above inequality is strict, our original policy is no longer optimal and if the inequality is tight, we have a smaller length cycle with the same cost, which still contradicts our original assumption that we started with an optimal cost cycle with minimum length.

2.8.D Proof of Theorem 5

We have shown that the points on the optimal cycle satisfy the strong-switch-type property. We need to show that we can assign actions to states that are not on the optimal cycle while maintaining the strong-switch property, for $N \leq 3$.

This can be done in an iterative manner. Consider the set of points in the state-space that have been assigned an action, and which satisfy the strong-switch property to be D. Let $\boldsymbol{x} \notin D$, be a new point that we want to assign an action to. There are three possible scenarios - 1) there exists $\boldsymbol{y} \in D$ such that the strong-switch-type property implies a unique action to be taken at $\boldsymbol{x}$, 2) there exists no such $\boldsymbol{y} \in D$ and so an arbitrary action can be chosen at $\boldsymbol{x}$, and 3) there exist multiple such points in D, which suggest different actions to be taken at $\boldsymbol{x}$.

Clearly, for scenarios 1 and 2 above, we can assign an action to the point $\boldsymbol{x}$, increase our set to $D \cup \{\boldsymbol{x}\}$ and repeat the procedure for a new point. We claim that if $N \leq 3$ then scenario 3 never occurs. This is sufficient to prove that we can extend the strong-switch-type property over the entire state-space.

To prove that scenario 3 doesn't happen, we start by assuming the contrary. Let $\boldsymbol{y_1}$ and $\boldsymbol{y_2} \in D$ and without loss of generality, assume that the action taken at $\boldsymbol{y_1}$ is 1 and the action taken at $\boldsymbol{y_2}$ is 2. Also, to satisfy our assumption of scenario 3 for $\boldsymbol{x}$, we require that $x^{(1)} \geq y_1^{(1)}, x^{(2)} \geq y_2^{(2)}, x^{(j)} \leq y_1^{(j)}, \forall j \neq 1$ and $x^{(k)} \leq y_1^{(k)}, \forall k \neq 2$. For these inequalities to be feasible simultaneously, we need $y_2^{(1)} \geq y_1^{(1)}$ and $y_1^{(2)} \geq y_2^{(2)}$.

Now, if there are only two sources, i.e. $N = 2$, then the fact that $y_2^{(1)} \geq y_1^{(1)}$ and $y_1^{(2)} \geq y_2^{(2)}$ together with the assumption that the action taken at $\boldsymbol{y_1}$ is 1 and the action taken at $\boldsymbol{y_2}$ is 2, we get that $\boldsymbol{y_1}$ and $\boldsymbol{y_2}$ violate the strong-switch property, despite being in the set D. This is a contradiction and completes our proof.

Similarly, consider the setting with three sources (N=3). Now, there are two possibilities - either $y_1^{(3)} \leq y_2^{(3)}$ or $y_1^{(3)} > y_2^{(3)}$. If $y_1^{(3)} \leq y_2^{(3)}$, then using the fact that $y_1^{(1)} \leq y_2^{(1)}$ and $y_1^{(2)} \geq y_2^{(2)}$, the strong-switch-type property implies that the

action taken at $\boldsymbol{y_1}$ must be 2. However, we assumed that the action taken at state $\boldsymbol{y_1}$ is 1. Thus, this violates the strong-switch-type property. Similarly, if $y_1^{(3)} > y_2^{(3)}$, then using the fact that $y_1^{(1)} \leq y_2^{(1)}$ and $y_1^{(2)} \geq y_2^{(2)}$, the strong-switch-type property implies that the action taken at $\boldsymbol{y_2}$ must be 1. This again violates our assumption that $\boldsymbol{y_1}$ and $\boldsymbol{y_2}$ satisfy the strong-switch-type property.

Thus, we have proved that for $N \leq 3$, if the optimal cycle is strong-switch-type then we can find a stationary optimal policy that is strong-switch-type over the entire state-space.

2.8.E Proof of Theorem 6

We use an inductive argument to prove this result. Assume that for a scheduling setup with $N-1$ sources, every strong-switch type policy can also be written as an index policy. Now, consider a functions of age setup with N sources and reliable channels. Using Theorem 5 we know that there exists an optimal policy that is strong-switch-type. Let this policy be $\pi : \mathbb{Z}^{+^N} \to \{1, \ldots, N\}$.

Let x_i denote the age of the i^{th} source when the current state is $\boldsymbol{x}$ and let $\boldsymbol{x_{-i}}$ denote the vector comprising of ages of all sources except i. Consider the minimum age x_1 at source 1 such that $\pi(x_1, \boldsymbol{x_{-1}}) = 1$, for any fixed $\boldsymbol{x_{-1}}$. That is, for any fixed value of ages for all other sources, consider the age at source 1 for which the optimal policy schedules the first source. This value of x_1 may depend on $\boldsymbol{x_{-1}}$, so we denote it by $x_{\text{th}}(\boldsymbol{x_{-1}})$. Observe that for all values of x_1 such that $x_1 \geq x_{\text{th}}(\boldsymbol{x_{-1}})$, the strong-switch-type property implies that $\pi(x_1, \boldsymbol{x_{-1}}) = 1$. In other words, $x_{\text{th}}(\boldsymbol{x_{-1}})$ acts like a threshold value such that for all values of age at source 1 above it, the optimal policy schedules the first source. If no such threshold exists, we let $x_{\text{th}}(\boldsymbol{x_{-1}}) \to \infty$.

We append the state space of the first arm by zero, i.e. let the state space of source 1 be $\mathbb{Z}_0 = \mathbb{Z}^+ \cup \{0\}$. Zero is the minimum age that this source can have and without loss of generality, we can set $f_1(0) = 0$. If at any time-slot the age

of this source is zero, we let it increases to one in the next time-slot. Scheduling this source when its age is zero gives us no benefit, as the age increases by one no matter what our scheduling decision is. We extend the policy π over this new state space as follows. Let $\pi' : \mathbb{Z}_0 \times \mathbb{Z}^{+^{N-1}} \to \{1, \ldots, N\}$ be a mapping that satisfies

- $\pi'(\boldsymbol{x}) = \pi(\boldsymbol{x}), \forall \boldsymbol{x}$ s.t. $x_1 \neq 0$.
- if $x_{\text{th}}(\boldsymbol{x_{-1}}) > 1$, $\pi'(0, \boldsymbol{x_{-1}}) = \pi(x_{\text{th}}(\boldsymbol{x_{-1}}) - 1, \boldsymbol{x_{-1}})$
- if $x_{\text{th}}(\boldsymbol{x_{-1}}) = 1$, $\pi'(0, \boldsymbol{x_{-1}}) = \pi_{N-1}(\boldsymbol{x_{-1}})$,

where $\pi_{N-1}(\boldsymbol{x_{-1}})$ is an optimal strong-switch-type policy for the functions of age problem with just the sources $2, \ldots, N$. It is easy to see that this new extended policy still satisfies the strong-switch-type property and is still optimal for the original problem with all N sources over the extended state space.

Now, we project this new optimal policy π' on to $\mathbb{Z}^{+^{N-1}}$ to get a new policy $\pi'' : \mathbb{Z}^{+^{N-1}} \to \{2, \ldots, N\}$ such that $\pi''(\boldsymbol{x_{-1}}) = \pi'(x_{\text{th}}(\boldsymbol{x_{-1}}) - 1, \boldsymbol{x_{-1}})$. This is well defined since $x_{\text{th}}(\boldsymbol{x_{-1}}) \geq 1$ and $\pi'(x_{\text{th}}(\boldsymbol{x_{-1}}) - 1, \boldsymbol{x_{-1}}) \in \{2, \ldots, N\}$, by construction. Also, π'' is strong-switch-type by construction, since it is a projection of a strong-switch-type policy onto a lower dimensional space. If not, then π' would also violate the strong-switch-type property.

Now, using our induction assumption, we can find index functions such that

$$\pi''(x_2, \ldots, x_N) = \arg\max_{2 \leq i \leq N} \Big\{ F_i(x_i) \Big\}$$

for all $\boldsymbol{x}$, where $F_i : \mathbb{Z}^+ \to \mathbb{R}$ are monotonically non-decreasing functions for all i.

We partition the $N-1$ dimensional state space of policy π'' into a countable number of sets. Let

$$S_k \triangleq \{\boldsymbol{x} : \boldsymbol{x} \in \mathbb{Z}^{+^{N-1}}, x_{\text{th}}(\boldsymbol{x}) = k\}, \forall k \in \mathbb{Z}^+.$$

Then, $\mathbb{Z}^{+^{N-1}} = \cup_{k=1}^{\infty} S_k$ and $S_k \cap S_j = \{\phi\}, \forall k, j$. Consider $\boldsymbol{x} \in S_j$ and $\boldsymbol{y} \in S_k$ such that $k > j$. Then,

$$\max_{2\leq i\leq N}\Big\{F_i(x_i)\Big\} \leq \max_{2\leq i\leq N}\Big\{F_i(y_i)\Big\}. \tag{2.37}$$

Suppose the opposite is true, i.e. $\max_{2\leq i\leq N}\Big\{F_i(x_i)\Big\} > \max_{2\leq i\leq N}\Big\{F_i(y_i)\Big\}$. Let $m = \arg\max_{2\leq i\leq N}\Big\{F_i(x_i)\Big\}$. Clearly, for the opposite of (2.37) to hold we need $x_m > y_m$. If we define $\boldsymbol{z}$ such that $z_i \triangleq \max\{x_i, y_i\}, \forall i \in 2, \ldots, N$, then using the index property of $\pi''(\cdot)$ we get

$$\pi''(\boldsymbol{z}) = m.$$

Also, $x_{\text{th}}(\boldsymbol{z}) \geq k$. If not, then since $\boldsymbol{z} \geq \boldsymbol{y}$, the strong-switch property implies $\pi'(x_{\text{th}}(\boldsymbol{z}), \boldsymbol{y}) = 1$, where $x_{\text{th}}(\boldsymbol{z}) < k$. This violates our assumption that $\boldsymbol{y} \in S_k$.

Now, observe that $\pi'(x_{\text{th}}(\boldsymbol{z})-1, \boldsymbol{z}) = m$, since $\pi''(\boldsymbol{z}) = m$. Also $(j, x_2, \ldots, x_m, \ldots, x_N) \leq (x_{\text{th}}(\boldsymbol{z})-1, z_2, \ldots, x_m, \ldots, z_N)$ where the inequality holds element-wise. This is because $z_i = \max\{x_i, y_i\}$, and $j \leq k-1 \leq x_{\text{th}}(\boldsymbol{z}) - 1$, and $x_m > y_m$. Thus, using the strong-switch-type property of $\pi'(\cdot)$, we get

$$\pi'(j, \boldsymbol{x}) = m.$$

This contradicts our initial assumption that $x \in S_j$, since that would imply $\pi'(j, \boldsymbol{x}) = 1$. Thus, we conclude that (2.37) must be satisfied.

We now construct a monotone function based on the above discussion. Let

$$F_1(j) \triangleq \sup_{\boldsymbol{z}\in S_j} \max_{2\leq i\leq N}\Big\{F_i(z_i)\Big\}, \forall j \in \mathbb{Z}^+. \tag{2.38}$$

Clearly, since the condition (2.37) is satisfied, the function $F_1(\cdot)$ is monotone. Also, let

$$\pi'''(\boldsymbol{x}) = \arg\max_{1\leq i\leq N}\Big\{F_i(x_i)\Big\}, \forall \boldsymbol{x} \in \mathbb{Z}^{+^N},$$

where we break ties in lexicographic order. Then π''' is the same our original pol-

icy π. This is because for every state $\boldsymbol{x} \in S_j$, the construction of F_1 forces us to schedule source 1 for values of $x_1 \geq j$ and not schedule source 1 for values below j. This holds for all values of j, which means we replicate the original scheduling policy $\pi(\cdot)$. Thus, if we assume strong-switch-type policies can be written as index policies for a problem with $N-1$ sources, we can also prove the same fact for N sources.

It is trivial to see that strong-switch-type policies and index policies are equivalent for the single source decoupled problem. This is because strong-switch-type policies and index policies both correspond to monotone threshold policies for the decoupled problem. Hence, using the principle of induction, we have the required result.

2.8.F Proof of Theorem 3

Using Corollary 1, we know that there exists some index policy which is optimal.We observe that for $N = 2$ index policies have a specific structure.

Let $F_1(\cdot)$ and $F_2(\cdot)$ represent the index functions for the optimal index policy. We set the ages of the two sources to $(1,1)$ at time $t = 1$ and assume that the optimal index functions are such that $F_1(1) \geq F_2(1)$. Then, the policy schedules source 1 at time $t = 1$. The new state at time $t = 2$ is given by $(1,2)$. Again, assume that $F_1(1) \geq F_2(2)$. Then, the policy schedules source 1 at time $t = 2$. The new state at time $t = 3$ is given by $(1,3)$. We keep repeating this process until we reach state $(1,k)$ at time $t = k$ for which $F_1(1) < F_2(k)$. The policy then schedules source 2 and reaches state $(2,1)$ at time $k+1$. Now, since we assumed that $F_1(1) \geq F_2(1)$, then using monotonicity we get $F_1(2) \geq F_2(1)$. Thus, the policy schedules source 1 again and we reach state $(1,2)$ at time $t = k+2$.

From the above discussion, we see that any index policy for $N = 2$ has a cyclic form and the cycle consists one of the sources being scheduled repeatedly followed by the second source once. To find the best index policy, which is also the

best policy overall, we just need to find the best policy with this specific structure.

Without loss of generality, assume that an optimal cyclical policy is given by scheduling source 1 k times followed by source 2 once, and repeating this sequence of actions. Now, consider two cases.

Case 1

($k > 1$) We compare the cost of the optimal cycle with a cycle that schedules source 1 $k-1$ times followed by source 2 once.

$$\frac{\sum_{j=1}^{k} f_2(j) + (k-1)f_1(1) + f(2)}{k} \geq \frac{\sum_{j=1}^{k+1} f_2(j) + kf_1(1) + f(2)}{k+1}$$

Simplifying, we get

$$f_1(2) - f_1(1) \geq kf_2(k+1) - \sum_{j=1}^{k} f_2(j),$$

i.e. $W_1(1) \geq W_2(k)$. The Whittle index policy follows the optimal policy till the state $(1,k)$.

We then compare the cost of the optimal cycle with a cycle that schedules source 1 $k+1$ times followed by source 2 once.

$$\frac{\sum_{j=1}^{k+2} f_2(j) + (k+1)f_1(1) + f(2)}{k+2} > \frac{\sum_{j=1}^{k+1} f_2(j) + kf_1(1) + f(2)}{k+1}$$

Simplifying, we get

$$(k+1)f_2(k+2) - \sum_{j=1}^{k+1} f_2(j) > f_1(2) - f_1(1),$$

i.e. $W_2(k+1) > W_1(1)$.

Together, this implies that the Whittle Index policy must also schedule source

1 k times followed by source 2 once, and repeat this sequence of actions. Hence, the Whittle index policy is optimal.

Case 2

($k = 1$) We compare the optimal policy with a cycle that schedules source 1 twice and source 2 once. Then, we get

$$\frac{2f_1(1) + f_1(2) + f_2(1) + f_2(2) + f_2(3)}{3} > \frac{f_1(1) + f_2(1) + f_1(2) + f_2(2)}{2}$$

Simplifying, we get

$$2f_2(3) - f_2(1) - f_2(2) > f_1(2) - f_1(1),$$

i.e. $W_2(2) > W_1(1)$. Using a symmetrical argument, it is easy to see that $W_1(2) > W_2(1)$. Thus, the Whittle Index policy also schedules each source exactly once, and repeats this sequence of actions.

Combining the two cases, we conclude that for $N = 2$ and reliable channels, the Whittle Index policy is exactly optimal.

2.8.G Proof of Theorem 7

Let $S : \mathbb{Z}^+ \to \mathbb{R}$ denote the differential cost-to-go function for this problem, let $u : \mathbb{Z}^+ \to \{1, 0\}$ be the stationary optimal policy and let λ denote the optimal cost. Then, the Bellman equations are given by

$$\begin{aligned} S(h) = f(h) + \min_{u(h)\in\{1,0\}} \{C + (1-p)S(h+1), S(h+1)\} \\ -\lambda, \forall h \in \mathbb{Z}^+. \end{aligned} \tag{2.39}$$

Without loss of generality we set $S(1) = 0$. Assume that the optimal policy has a threshold structure, i.e. there exists H such that it is optimal to pull the arm ($u(h) = 1$) for all states $h \geq H$ and let it rest otherwise ($u(h) = 0$). If this the case, then the Bellman equations for values above the threshold H reduce to

$$S(h) = f(h) + C + (1-p)S(h+1) - \lambda, \forall h \geq H. \tag{2.40}$$

Solving this recursion and assuming $\lim_{h\to\infty}(1-p)^h S(h) = 0$, we get

$$S(H+j) = \sum_{k=j}^{\infty} f(k+H)(1-p)^{k-j} + \frac{C-\lambda}{p}, \forall j \geq 0. \tag{2.41}$$

Since $f(\cdot)$ is non-decreasing, it is easy to see that $S(h)$ is also non-decreasing for all values of h above the threshold H, using (2.41). We will use this fact later. Now, observe that

$$\lim_{h\to\infty}(1-p)^h S(h) = \lim_{h\to\infty}\sum_{j=h}^{\infty} f_i(j)(1-p_i)^j + \lim_{h\to\infty}\frac{C-\lambda}{h}(1-p_i)^h, \forall h \geq H. \tag{2.42}$$

By the bounded cost assumption, the first term is the limit of the partial sums of a convergent series, thus it goes to zero. The second term also goes to zero since $p < 1$ and λ is finite, again using the bounded cost assumption. This confirms that our assumption $\lim_{h\to\infty}(1-p)^h S(h) = 0$ was indeed correct.

For $h = H-1$, the Bellman equation is given by

$$S(H-1) = f\big(H-1\big) - \lambda + S(H) = f\big(H-1\big) - \lambda + \sum_{k=0}^{\infty} f(k+H)(1-p)^k + \frac{C-\lambda}{p}. \tag{2.43}$$

Repeating this k times, we get

$$S(H-k) = \sum_{j=H-k}^{H-1} (f(j)-\lambda) + \sum_{j=0}^{\infty} f(j+H)(1-p)^j + \frac{C-\lambda}{p}, \forall k \in \{1,\dots,H-1\}. \tag{2.44}$$

Now, putting $k = H-1$ in the above equation and using the fact that $S(1) = 0$, we get

$$\lambda = \frac{p\left(\sum_{j=1}^{H} f(j) + \sum_{k=1}^{\infty} f(k+H)(1-p)^k\right) + C}{1+p(H-1)}. \tag{2.45}$$

If we further assume that the threshold value H satisfies the condition (2.11) given in Theorem 7, then we get that

$$\begin{aligned} p^2(H-1)\left(\sum_{k=H}^{\infty} f(k)(1-p)^{k-H}\right) - p\left(\sum_{j=1}^{H-1} f(j)\right) \\ \leq C \\ \leq p^2 H\left(\sum_{k=H+1}^{\infty} f(k)(1-p)^{k-H-1}\right) - p\left(\sum_{j=1}^{H} f(j)\right). \end{aligned} \tag{2.46}$$

Rearranging terms, dividing by $1+p(H-1)$ and using the expression for λ from (2.45), we get

$$p\left(\sum_{k=0}^{\infty} f(H+k)(1-p)^k\right) \leq \lambda \leq p\left(\sum_{k=1}^{\infty} f(H+k)(1-p)^{k-1}\right). \tag{2.47}$$

Simplifying the inequalities in (2.47), the expression for λ from (2.45) and the Bellman solutions (2.41), we get

$$S(H) \leq \frac{C}{p} \leq S(H+1). \tag{2.48}$$

Using (2.44), we note that for $h < H$, $S(h) - S(h-1) = \lambda - f(h-1)$. Also, using the

monotonicity of $f(\cdot)$ and (2.47), we get

$$\begin{aligned}\lambda &\geq p\left(\sum_{k=0}^{\infty} f(H+k)(1-p)^k\right) \geq p\left(\sum_{k=0}^{\infty} f(H)(1-p)^k\right) \\ &\geq f(H) \geq f(h), \forall h < H.\end{aligned} \tag{2.49}$$

Thus, $S(h) - S(h-1) \geq 0, \forall h$ since we already established monotonocity for $h \geq H$. Since $S(\cdot)$ is non-decreasing, (2.48) implies that

$$\begin{aligned}&S(h) \leq C + (1-p)S(h), \forall h \leq H, \text{ and} \\ &S(h) \geq C + (1-p)S(h), \forall h > H.\end{aligned} \tag{2.50}$$

Thus, if we find an H that satisfies (2.11), the threshold policy using H as a threshold satisfies the Bellman equations and is optimal.

The one thing that remains to be shown is the case in which we cannot find some H that satisfies (2.27). As done earlier, we define a function $W : \mathbb{Z}^+ \to \mathbb{R}$ given by

$$W(h) = p^2(h-1)\left(\sum_{k=h}^{\infty} f(k)(1-p)^{k-h}\right) - p\left(\sum_{j=1}^{h-1} f(j)\right). \tag{2.51}$$

Observe that

$$\begin{aligned}W(h+1) - W(h) =& p^2 h\left[\sum_{k=0}^{\infty}\big(f(h+1+k) - f(h+k)\big)(1-p)^k\right] \\ &+ p^2\left[\sum_{k=0}^{\infty} f(h+k)(1-p)^k\right] - pf(h) \\ &\geq 0, \forall h\end{aligned} \tag{2.52}$$

since $f(\cdot)$ is non-decreasing. Thus, $W(\cdot)$ is also non-decreasing. Also, putting $h = 1$ in the definition of $W(h)$ we get $W(1) = 0$, while we had assumed that $C > 0$. Thus, $W(1) < C$. Now, if there exists some $h > 1$ such that $W(h) \geq C$, then we know that there also exists some H such that $W(H) \leq C \leq W(H+1)$ using monotonicity

of $W(\cdot)$. Observe that this implies that there exists some H satisfying (2.11) and hence the threshold policy is optimal. If there exists no H satisfying (2.11), then $W(h) < C, \forall h$.

Since $W(\cdot)$ is a bounded monotone sequence, it converges to a finite value. It is easy to see that this implies that $f(\cdot)$ is also bounded and hence converges. We set $\lambda = \lim_{h\to\infty} f(h)$ and the cost-to-go function $S(h)$ to be

$$S(h) = \sum_{j=h}^{\infty} \big(f(j) - \lambda\big) + C. \tag{2.53}$$

Clearly, $S(h)$ satisfies the recurrence relation

$$S(h) = f(h) - \lambda + S(h+1), \forall h. \tag{2.54}$$

By the monotonicity of $f(\cdot)$, we know that $f(h) \leq \lambda, \forall h$. Thus, using (2.53) we conclude that $S(h) \leq C, \forall h$. This implies that $S(\cdot)$ satisfies the Bellman equations, with the optimal policy being to never activate the arm. This completes our proof.

2.8.H Proof of Theorem 8

This proof is very similar to the indexability proof for the reliable channels case. For $C = 0$, it is obvious that the optimal policy is to always activate the arm since there is no charge for activating it and the cost function is monotone and positive. For larger values of C, consider the function $W : \mathbb{Z}^+ \to \mathbb{R}^+$ given by

$$W(h) = p^2(h-1)\bigg(\sum_{k=h}^{\infty} f(k)(1-p)^{k-h}\bigg) - p\bigg(\sum_{j=1}^{h-1} f(j)\bigg).$$

Observe that since $f(\cdot)$ is non-decreasing, $W(\cdot)$ is also non-decreasing, as discussed in Appendix 2.8.G. Also, by definition, $W(1) = 0$, while we had assumed

that $C > 0$. Thus, $W(1) < C$. Now, if there exists some $h > 1$ such that $W(h) \geq C$, then we know that there also exists some H such that $W(H) \leq C \leq W(H+1)$ using monotonicity of $W(\cdot)$. Observe that this implies that there exists some H satisfying (2.11).

Using this, we can relate the optimal threshold values to values of activation charge. Let C be such that it lies in the interval $\big[W(h), W(h+1)\big)$, then the optimal policy is of threshold type with the threshold at h. Observe that if W is strictly increasing then there can only be one such interval in which C can lie. If $W(\cdot)$ is non-decreasing, then there could be multiple such intervals in which C could lie. In this case, we choose the smallest h such that the condition holds.

The monotonicity of $W(\cdot)$ ensures that the the threshold value is also monotone non-decreasing with increasing values of C. When $W(h) < C, \forall h$, we choose h to be ∞, as done in Appendix 2.8.G. This completes the proof of *indexability* for the decoupled problem. Observe that $C = W(h+1)$ is the minimum value of the activation charge that makes both actions equally desirable in state h. This gives us the expression for the Whittle index.

2.8.I Proof of Theorem 9

Suppose that the base station knows that the ith process was at state $x_i(\tau) = x_0$ at time τ. Further, suppose that it received no additional updates regarding the ith process up to time-slot $\tau + \Delta$. Without loss of generality, we can set $\tau = 0$, since we can always offset the time-slots by a fixed constant.

Then, using the state evolution equation 2.13, we know that

$$\begin{aligned} x_i(1) &= G_i x_0 + w_i(0) \\ x_i(2) &= G_i^2 x_0 + G_i w_i(0) + w_i(1) \\ &\dots \\ x_i(\Delta) &= G_i^{\Delta} x_0 + \sum_{k=0}^{\Delta-1} G_i^{\Delta-k-1} w_i(k). \end{aligned} \tag{2.55}$$

The base station does not have access to the increments $w_i(0), ..., w_i(\Delta-1)$. However, it knows that each of them is i.i.d. and $\mathcal{N}(0,\Sigma_i)$. Thus, the maximum likelihood of the state at time Δ is given by

$$\hat{x}_i(\Delta) = \mathbb{E}[x_i(\Delta)|x_i(0) = x_0] = G_i^{\Delta} x_0. \tag{2.56}$$

Using this, we can now compute the difference between the actual state and the estimate at the base station

$$x_i(\Delta) - \hat{x}_i(\Delta) = \sum_{k=0}^{\Delta-1} G_i^{\Delta-k-1} w_i(k). \tag{2.57}$$

Observe that this is simply a sum of zero-mean independent multi-variate normal random variables. Thus, $x_i(\Delta) - \hat{x}_i(\Delta)$ is also a zero-mean multi-variate normal random variable. Recall the following standard properties of multivariate normal random variables. If $X \sim \mathcal{N}(0,\Sigma)$ and $Y = GX$ is some linear transformation of X, then $Y \sim \mathcal{N}(0, G\Sigma G^T)$. Further, if $X_1 \sim \mathcal{N}(0,\Sigma_1)$ and $X_2 \sim \mathcal{N}(0,\Sigma_2)$ are independent, then $Z = X_1 + X_2$ is distributed as $\mathcal{N}(0,\Sigma_1+\Sigma_2)$. Finally, if $X \sim \mathcal{N}(0,\Sigma)$, then $\mathbb{E}[X^T X] = Tr(\Sigma)$. Putting the first two properties together, we observe that

$$x_i(\Delta) - \hat{x}_i(\Delta) \sim \mathcal{N}\left(0, \sum_{k=0}^{\Delta-1} G_i^k \Sigma_i (G_i^k)^T\right). \tag{2.58}$$

Using the last property, we get

$$\begin{aligned} e_i(\Delta) &= \mathbb{E}\big[(x_i(\Delta) - \hat{x}_i(\Delta))^T (x_i(\Delta) - \hat{x}_i(\Delta))\big] \\ &= \mathbb{E}\big[||x_i(\Delta) - \hat{x}_i(\Delta)||_2^2\big] \\ &= Tr\left(\sum_{k=0}^{\Delta-1} G_i^k \Sigma_i (G_i^k)^T\right) \\ &= \sum_{k=0}^{\Delta-1} Tr\big((G_i^k)\Sigma_i (G_i^k)^T\big) \\ &= \sum_{k=0}^{\Delta-1} Tr\big((G_i^k)^T (G_i^k)\Sigma_i\big) \triangleq f_i(\Delta). \end{aligned} \tag{2.59}$$

The last two equalities follow from the linearity of the trace operator and the fact that $Tr(AB) = Tr(BA)$. This completes the proof of Theorem 9.

We also want to show that $f_i(\Delta)$ increases monotonically in Δ. This is straightforward to show, since $G_i^k \Sigma_i (G_i^k)^T$ is a covariance matrix for any $k \in \mathbb{Z}^+$. This implies that it must be positive semi-definite, and in turn, must have a non-negative trace. Now, we consider the difference

$$f_i(\Delta + 1) - f_i(\Delta) = Tr\big((G_i^\Delta)\Sigma_i (G_i^\Delta)^T\big) \geq 0. \tag{2.60}$$

The last inequality follows due to the non-negativity of trace for a positive semi-definite matrix. This shows that $f_i(\Delta + 1) \geq f_i(\Delta)$, which allows us to conclude monotonicity of the AoI cost functions.

2.8.J Proof of Theorem 10

As we discussed earlier, the estimate distribution at time $\tau + \Delta$, given the last observation at time τ is $x_i(\tau)$, is given by

$$\hat{x}_i(\tau + \Delta) = x_i(\tau) Q_i^\Delta.$$

Without loss of generality, we can assume that $x_i(\tau) = [1 \;\; 0]$, i.e. the chain at time τ is in state 0. This is because the chain is symmetric, so it does not matter which state we start from. Using this, we get the estimate distribution to be

$$\hat{x}_i(\tau+\Delta) = \left[[Q_i^\Delta]_{00} \;\; 1 - [Q_i^\Delta]_{00}\right]. \tag{2.61}$$

Further, at time $\tau+\Delta$ the actual state of the chain is 0 with probability $[Q_i^\Delta]_{00}$ and 1 with probability $1-[Q_i^\Delta]_{00}$. Thus, the distribution of the actual state at time $\tau+\Delta$ is given by

$$x_i(\tau+\Delta) = \begin{cases} [1 \;\; 0], & \text{with probability } [Q_i^\Delta]_{00} \\ [0 \;\; 1], & \text{with probability } 1-[Q_i^\Delta]_{00}. \end{cases} \tag{2.62}$$

Now, suppose that the distance between the actual and estimate distributions is measured using the Kullback-Leibler (KL) divergence. Then,

$$\begin{aligned} &\mathbb{E}\left[D_{KL}\left(x_i(\tau+\Delta)||\hat{x}_i(\tau+\Delta)\right)\right] \\ &= [Q_i^\Delta]_{00} D_{KL}\left([1 \;\; 0]\middle\|\left[[Q_i^\Delta]_{00} \;\; 1-[Q_i^\Delta]_{00}\right]\right) \\ &+ (1-[Q_i^\Delta]_{00}) D_{KL}\left([0 \;\; 1]\middle\|\left[[Q_i^\Delta]_{00} \;\; 1-[Q_i^\Delta]_{00}\right]\right) \\ &= -[Q_i^\Delta]_{00}\log\left([Q_i^\Delta]_{00}\right) - (1-[Q_i^\Delta]_{00})\log\left(1-[Q_i^\Delta]_{00}\right) \\ &= H\left([Q_i^\Delta]_{00}\right). \end{aligned} \tag{2.63}$$

Here $H(q) \triangleq -q\log(q) - (1-q)\log(1-q)$ is the binary entropy function.

Now, suppose that the distance between the actual and estimate distributions

is measured using the total variation (TV) distance. Then,

$$\begin{aligned}
&\mathbb{E}\Big[D_{TV}\big(x_i(\tau+\Delta)||\hat{x}_i(\tau+\Delta)\big)\Big] \\
&= [Q_i^\Delta]_{00} D_{TV}\Big([1\ \ 0]\Big\|\big[[Q_i^\Delta]_{00}\ \ 1-[Q_i^\Delta]_{00}\big]\Big) \\
&+ (1-[Q_i^\Delta]_{00}) D_{TV}\Big([0\ \ 1]\Big\|\big[[Q_i^\Delta]_{00}\ \ 1-[Q_i^\Delta]_{00}\big]\Big) \\
&= [Q_i^\Delta]_{00}\big(1-[Q_i^\Delta]_{00}\big) + (1-[Q_i^\Delta]_{00})[Q_i^\Delta]_{00} \\
&= 2[Q_i^\Delta]_{00}\big(1-[Q_i^\Delta]_{00}\big) \\
&\triangleq g\big([Q_i^\Delta]_{00}\big).
\end{aligned} \tag{2.64}$$

Here $g(x) = 2x(1-x)$. This completes the proof of Theorem 10.

In addition, we also need to show that the two functions derived above are monotonically increasing. To do so, we will simplify our notation a bit. Let $\mu_0 = [Q_i^\Delta]_{00}$ by μ_0 and $\mu_1 = 1-\mu_0 = [Q_i^\Delta]_{01}$. Further, let $\nu_0 = [Q_i^{\Delta+1}]_{00} = \mu_0(1-q_i) + (1-\mu_0)q_i$ and $\nu_1 = [Q_i^{\Delta+1}]_{01} = 1-\nu_0 = \mu_1(1-q_i) + (1-\mu_1)q_i$. We will split the proof into two cases.

Case 1 (KL Divergence): Note that the function $x\log(x)$ is convex for all $x > 0$, since $\frac{d^2}{dx^2}(x\log(x)) = \frac{1}{x} > 0, \forall x > 0$. Using this fact and the definitions of ν_0 and ν_1, we obtain the following inequalities:

$$(1-q_i)\mu_0\log(\mu_0) + q_i\mu_1\log(\mu_1) \geq \nu_0\log(\nu_0), \tag{2.65}$$

$$(1-q_i)\mu_1\log(\mu_1) + q_i\mu_0\log(\mu_0) \geq \nu_1\log(\nu_1), \tag{2.66}$$

Now, we look at the difference:

$$H\big([Q_i^{\Delta+1}]_{00}\big) - H\big([Q_i^{\Delta}]_{00}\big) = \Big((1-q_i)\mu_0\log(\mu_0) + q_i\mu_1\log(\mu_1) - \nu_0\log(\nu_0)\Big) + \Big((1-q_i)\mu_1\log(\mu_1) + q_i\mu_0\log(\mu_0) - \nu_1\log(\nu_1)\Big) \geq 0. \quad (2.67)$$

The inequality above follows by applying (2.65) and (2.66). This proves that the monitoring error grows monotonically with the AoI for KL divergence.

Case 2 (TV distance): Note that the function $2x(1-x)$ is concave for all x, since $\frac{d^2}{dx^2}(2x(1-x)) = -2 < 0, \forall x$. Using this fact and the definitions of ν_0 and ν_1, we obtain the following inequalities:

$$(1-q_i)2\mu_0(1-\mu_0) + q_i 2\mu_1(1-\mu_1) \leq 2\nu_0(1-\nu_0), \quad (2.68)$$

$$(1-q_i)2\mu_1(1-\mu_1) + q_i 2\mu_0(1-\mu_0) \leq 2\nu_1(1-\nu_1), \quad (2.69)$$

Now, we look at the difference:

$$g\big([Q_i^{\Delta+1}]_{00}\big) - g\big([Q_i^{\Delta}]_{00}\big) = \Big(2\nu_0(1-\nu_0) - (1-q_i)2\mu_0(1-\mu_0) - q_i 2\mu_1(1-\mu_1)\Big) + \Big(2\nu_1(1-\nu_1) - (1-q_i)2\mu_1(1-\mu_1) - q_i 2\mu_0(1-\mu_0)\Big) \geq 0. \quad (2.70)$$

The inequality above follows by applying (2.68) and (2.69). This proves that the monitoring error grows monotonically with the AoI for TV distance as well.

Chapter 3

Online Learning of AoI Cost Functions

In this chapter, we ask the question: *what if the AoI cost functions are not known in advance, time-varying and possibly adversarial?* How does one go about designing scheduling policies that lead to good monitoring accuracy or control?

Our goal is to model applications where delivering timely information is of essence but the costs for delayed information are not completely known beforehand and hard to model, including non-stationary settings and adversarial dynamics. A broad range of networked control and monitoring applications fit this description. An example is designing scheduling schemes for real-time monitoring of power grids which have nonlinear and complicated dynamics that cannot be easily modeled. Another example is scheduling for mobility tracking. Mobility traces in the real world are often highly non-stationary and hard to explain via models. A third example is monitoring queue length information in data centers for load balancing, where only a small number of queues are sampled every few time-steps, and traffic flows, server outages and job sizes may be non-stationary and possibly adversarial. All of the above problems require optimization of unknown time-varying cost functions of AoI in an online fashion.

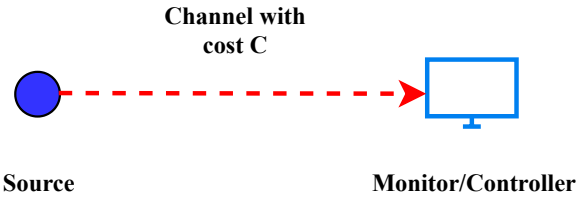


Figure 3-1: Single source monitoring

3.1 Single Source Monitoring

We start by discussing the single source setting with a known AoI cost function that remains fixed throughout. This will provide important insight and reveal key technical issues while formulating an online version of the problem.

Consider a single source sending updates to a monitoring station over a costly wireless channel as in Figure 3-1. In every time-slot, the scheduler decides whether the source sends a new update to the monitor. If it does, the monitor receives a new update in the next time-slot. The monitor maintains an age of information $A(t)$ which tracks how long it has been since it received a new update from the source. The evolution of $A(t)$ can be written as:

$$A(t+1) = \begin{cases} A(t)+1, & \text{if } u(t) = 0 \\ 1, & \text{if } u(t) = 1, \end{cases} \tag{3.1}$$

where $u(t)$ indicates whether a new update was sent in time-slot t.

There is a known cost function of AoI $f(\cdot)$ which models the cost of having stale information at the monitor. Thus, the cost at any time-slot is:

$$\text{Cost}(t) = f(A(t)) + Cu(t), \tag{3.2}$$

where $C > 0$ is the cost of sending a new update from the source. Our goal is to design a monitoring policy that minimizes the long term average cost. The

average cost under a policy π is:

$$\text{Cost}_{\text{ave.}}(\pi) \triangleq \limsup_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} f(A^{\pi}(t)) + Cu^{\pi}(t). \tag{3.3}$$

This problem and the optimal policy have been analyzed in [1] and [26]. We describe the key result from these works below.

Lemma 1. *The optimal policy for the single source monitoring problem is a stationary threshold policy. Let H satisfy*

$$f(H) \leq \frac{\sum_{j=1}^{H} f(j) + C}{H} \leq f(H+1). \tag{3.4}$$

Then, the optimal policy is to send an update at time-slot t only if $A(t) \geq H$. If no such H exists, the optimal policy is to never send an update.

Proof. See Chapter 2. □

Lemma 1 implies that the optimal monitoring policy is to send an update only if the current AoI gets above a threshold H. Similar threshold based schemes have appeared in many different settings for online sampling and remote estimation [22, 23, 24, 96].

Now consider a naive reformulation of the problem where the cost function $f(\cdot)$ is time-varying and unknown. We represent it by $f_t(\cdot)$ where the subscript indicates its time varying nature. If the function were fixed and unknown, we could have used reinforcement learning to solve the problem as done in [36]. However, this cannot be done directly for settings with time-varying costs.

On the other hand, there is a large amount of literature on online learning and optimization where the goal is to solve a sequence of optimization problems

which vary in an unknown, possibly adversarial manner (see [97] and [98] for a detailed introduction to the field). In these problems there is no system state or history, so decisions in the current time-slot do not affect the optimization problem or decisions in a future time-slot. This is not true of our monitoring problem which has a system state (AoI), and where the state evolution depends on decisions taken in the past.

To overcome these difficulties, we reformulate the single source monitoring problem in an epoch based setting.

3.1.A An Epoch Based Formulation

We observe that the AoI of the source resets to 1 after every new update delivery. Thus, AoI evolution within an update inter-delivery period does not depend on the AoI evolution in any other period. We use this observation to formulate an epoch based problem.

We divide time into T epochs, where each epoch further consists of M time-slots. As before, when a new update is sent it gets delivered in the next time-slot. At the beginning of epoch k, we choose an AoI threshold $x_k \in \{1, ..., M\}$. Within the epoch, the source generates a new sample and sends it to the monitor whenever the AoI reaches the threshold x_k. In the last time-slot of the epoch, a new sample is sent regardless of the AoI. This ensures that the next epoch begins with AoI at 1. A cost is observed for sending samples every x_k time-slots based on the current system dynamics and communication costs. Then, epoch $k+1$ starts. Using cost information about previous decisions, a new sampling threshold x_{k+1} is chosen for epoch $k+1$ and the process repeats itself.

In each epoch k there is a function $f_k(\cdot)$ that represents the current cost for age of information and remains fixed for the duration of the epoch. So, for any time-slot t in epoch k, the current cost is given by $f_k(A(t)) + Cu(t)$. The total cost incurred in epoch k denoted by $C_k(x)$ is simply the sum of the cost in the

individual time-slots.

Lemma 2. *If a sampling threshold of x is chosen in epoch k and the AoI cost function is $f_k(\cdot)$ then the loss function $C_k(x)$ is given by:*

$$C_k(x) = \left\lfloor \frac{M}{x} \right\rfloor \left(\sum_{j=1}^{x} f_k(j) + C \right) + \mathbb{1}_{r>0} \left(\sum_{j=1}^{r} f_k(j) + C \right), \tag{3.5}$$

where $r = M \mod x$. This is the sum total AoI cost of monitoring over the epoch k.

Proof. See Appendix 3.5.A. □

If we knew $f_k(\cdot)$ at the beginning of epoch k, we could use (3.5) to find the optimal sampling threshold x_k^*. In our online framework, the goal is to learn the best sampling thresholds without knowing any information about the sequence of cost functions that we are going to face.

While we have motivated the setting above using cost that splits into a sum of AoI cost and communication cost, our setup allows for general cost functions $C_k(x_k)$ that map the choice of sampling threshold x_k to a cost in epoch k. For the remainder of this section, we will deal with these general cost functions $C_k(\cdot)$.

In our online setting, an **unconstrained adversary** chooses the sequence of **bounded** cost functions $C_k(\cdot)$ for each epoch k. The designer does not have access to the sequence of cost functions beforehand and must learn a suitable transmission/sampling policy in an online manner. We make no assumptions on how the underlying system dynamics or resulting cost functions change across epochs.

Note that the cost function $C_k(\cdot)$ in epoch k can be seen as an M dimensional vector where the cost for choosing the sampling threshold x is represented by the $C_k(x)$ which is the xth element of the vector. Going forward, when we use the notation C_k, we refer to the M dimensional vector of costs for each threshold in epoch t, while $C_k(x)$ represents its xth element.

The boundedness of the cost functions C_k is crucial to proving any meaningful results in this setting and is standard in online learning literature. Without loss of generality, we further assume that the cost functions are normalized such that $C_k(x) \in [0,1]$ for all sampling thresholds x and epochs k.

Feedback Structure

For the setup described above, we will look at two kinds of feedback structure for observing the costs. Note that x_k represents the decision made at the beginning of epoch k.

- Full Feedback - the scheduler observes the entire cost function $C_k(x), \forall x \in \{1,...,M\}$ at the end of epoch k.
- Bandit Feedback - the scheduler observes only $C_k(x_k)$ at the end of epoch k.

Objective (Regret Minimization)

: For any sequence of cost functions $C_1(\cdot), C_2(\cdot), ..., C_T(\cdot)$, x^* is defined as the best fixed sampling threshold that minimizes sum AoI cost. It is given by the following equation.

$$x^* \triangleq \arg\min_{x\in\{1,...,M\}} \sum_{k=1}^{T} C_k(x). \tag{3.6}$$

Our goal is to find an online policy that achieves sublinear regret compared to the best fixed sampling threshold x^* for any sequence. This is known as *worst-case static regret*. For any policy π, it is defined as follows:

$$\text{Regret}_T(\pi) = \sup_{C_1,...,C_T} \left\{ \sum_{k=1}^{T} C_k(x_k^\pi) - \min_{x\in\{1,...,M\}} \sum_{k=1}^{T} C_k(x) \right\}. \tag{3.7}$$

Note that regret is defined over epochs rather than time-slots since we assume that cost functions can change only across epochs.

We will now show that our online sampling problem formulation is equivalent to the prediction with expert advice setting that is well studied in online learning literature. This will allow us to apply policies and regret bounds derived for this setting to our problem.

Prediction With Expert Advice:

A decision maker has to choose among the advice of n given experts. After making a choice, a bounded loss is incurred. This scenario is repeated iteratively, and at each iteration the costs of choosing the various experts are arbitrary (possibly even adversarial, trying to mislead the decision maker). The goal of the decision maker is to do as well as the best expert in hindsight.

In our setting, the role of experts is played by the AoI thresholds $x \in \{1, ..., M\}$. In each epoch, the scheduler decides an AoI sampling threshold x and observes an associated cost. This process repeats iteratively with time-varying, possibly adversarial changes to costs. Thus, our setting corresponds with the expert advice setting with M experts.

Sublinear Regret

We now discuss in detail a policy that achieves sublinear static regret for the full feedback setting. This will illustrate how regret bounds from online learning literature can be applied to our single source online monitoring setup.

The full feedback assumption in our setting means that we observe costs for all possible sampling thresholds in every epoch. This makes sense when the scheduler has information about the current source dynamics and communication costs by the end of an epoch. Knowing this information is often sufficient to construct the current cost function for any possible sampling threshold.

We describe an online monitoring policy based on Follow the Perturbed Leader (FTPL) style algorithms. The FTPL method was first analyzed in the online set-

ting in [99] and is based on an algorithm first proposed in [100]. The key idea of the FTPL algorithm is to maintain the sum of cost functions observed until now and perturb it slightly. Choosing the best AoI threshold based on this perturbed history is sufficient to get sublinear regret.

Algorithm 1: FTPL for Online Monitoring

Input : parameter $\eta > 0$, number of thresholds M

Set $\Theta_1 \leftarrow 0$
while $t \in 1, ..., T$ **do**
  Sample $\gamma_t \sim \mathcal{N}(0, I)$
  Choose sampling threshold $x_t \in \arg\min_{x \in \{1,...,M\}} \Theta_t(x) + \eta\gamma_t(x)$
  Incur loss $C_t(x_t)$ and update $\Theta_{t+1} = \Theta_t + C_t$
end

Theorem 11. *FTPL online monitoring described by Algorithm 1 with $\eta = \sqrt{T}$ achieves the following upper bound for expected regret:*

$$\mathbb{E}[Regret_T(FTPL)] \leq 2\sqrt{2T \log M},$$

where the expectation is taken over the random perturbations.

Proof. The proof is based on [101]. A lower bound of the form $\Omega(\sqrt{T \log M})$ is also available in [97]. □

Bandit Feedback

The bandit feedback assumption implies that we only observe the cost associated with the chosen sampling threshold. This is a realistic assumption especially when no other information about the system dynamics and communica-

tion costs is available to the scheduler. However, the single point feedback means learning happens slowly and regret bounds are worse in this setting.

The online bandit setting has also been well studied in literature. Notably, the EXP3 algorithm was first proposed in the seminal paper [102] and is known to have near optimal expected regret under bandit feedback. We describe online monitoring based on EXP3 below.

Algorithm 2: EXP3 for Online Monitoring

Input : parameter $\epsilon > 0$, distribution $p_1 = \mathbf{1}/M$

1 **while** $t \in 1, ..., T$ **do**

2 Choose sampling threshold $x_t \sim p_t$

3 Incur loss $C_t(x_t)$ and observe $C_t(x_t)$

4 Let

$$\hat{C}_t(i) = \begin{cases} C_t(i)/p_t(i), & \text{if } i = x_t \\ 0, & \text{otherwise.} \end{cases}$$

5 Update $y_{t+1}(i) = p_t(i)e^{-\epsilon \hat{C}_t(i)}$, $p_{t+1} = \frac{y_{t+1}}{||y_{t+1}||_1}$

6 **end**

The key idea of Algorithm 2 is to maintain an unbiased estimate of the cost C_t via importance sampling (line 4). In every epoch, the algorithm samples a threshold x_t from a probability distribution p_t over the M thresholds. At the end of the epoch, p_t is updated with the current cost function estimate using exponential weights. It can be shown that the expected regret of EXP3 is sublinear and near optimal. Theorem 12 provides an upper bound on the expected regret in the bandit feedback setting.

Theorem 12. *The EXP3 online sampling policy described by Algorithm 2 with* $\epsilon = \sqrt{\frac{\log M}{TM}}$ *achieves the following upper bound for expected regret:*

$$\mathbb{E}[Regret_T(EXP3)] \leq 2\sqrt{TM \log M}.$$

The expectation is taken over the random sampling decisions made in each epoch.

Proof. The proof and a lower bound of the form $\Omega(\sqrt{TM})$ follow from discussion in [102]. □

Next, we discuss what sublinear epoch regret means for AoI cost averaged over time-slots. To do so, we note that the sum total AoI cost over all time-slots equals the cost summed over individual epochs, by definition. Let π denote an online algorithm which specifies threshold x_k^π to be chosen in epoch k and let E_k be the set of times-slots in epoch k. Then,

$$\sum_{k=1}^{T} C_k(x_k^\pi) = \sum_{k=1}^{T} \sum_{t \in E_k} f_k(A^\pi(t)) + Cu^\pi(t). \tag{3.8}$$

The relation above immediately implies that epoch regret also equals regret over time-slots. We describe this in the lemma below.

Lemma 3. *Suppose an online algorithm π has an upper bound on its expected static epoch regret of the form $f(M,T)$. Let π^* denote the policy corresponding to the best fixed AoI threshold x^* given the entire sequence of AoI cost functions $f_1, ..., f_T$ and the sampling cost C. Then for any bounded sequence of cost functions, the same upper bound holds for regret over time-slots, i.e.*

$$\begin{aligned} \mathbb{E}\Bigg[\sup_{f_1,\ldots,f_T} \Bigg\{ \sum_{k=1}^{T} \sum_{t \in E_k} f_k(A^\pi(t)) + Cu^\pi(t) - \\ \sum_{k=1}^{T} \sum_{t \in E_k} f_k(A^{\pi^*}(t)) + Cu^{\pi^*}(t) \Bigg\} \Bigg] \leq f(M,T), \end{aligned} \tag{3.9}$$

Proof. Substituting $\sum_{k=1}^{T} C_k(x_k^\pi)$ in (3.7) using (3.8) gives us the required result. □

If $f(M,T)$ is sublinear in the number of epochs T, then it is also sublinear in the number of time-slots MT since we assume that M is fixed to be a large constant. Thus, using Lemma 3, sublinear epoch regret implies sublinear time-slot regret. As a direct corollary of this, any online algorithm with sublinear static epoch regret achieves an expected time-average AoI cost which is at least as good as that under the best fixed sampling threshold.

Corollary 2. *Suppose an online algorithm π has an upper bound on its expected static regret that grows sublinearly in T. Let π^* denote the policy corresponding to the best fixed AoI threshold x^* given the entire sequence of AoI cost functions $f_1, ..., f_T$. Then for any sequence of bounded cost functions the following holds:*

$$\begin{aligned} &\limsup_{T\to\infty} \frac{1}{MT}\mathbb{E}\left[\sum_{k=1}^{T}\sum_{t\in E_k} f_k(A^{\pi}(t)) + Cu^{\pi}(t)\right] \leq \\ &\limsup_{T\to\infty} \frac{1}{MT}\mathbb{E}\left[\sum_{k=1}^{T}\sum_{t\in E_k} f_k(A^{\pi^*}(t)) + Cu^{\pi^*}(t)\right]. \end{aligned} \tag{3.10}$$

Proof. See Appendix 3.5.B. □

Note that the relation in Corollary 1 is an inequality and not an equality because we are comparing to the best static threshold policy across epochs and it is possible that our online monitoring policy performs better.

3.2 Multiple Sources

Motivated by the single-source discussion, we study a more challenging problem. Now, multiple sources are sending information to a monitoring station over a network as in Figure 3-2. In this setting, the scheduler needs to decide which source gets to send an update in every time-slot to optimize for overall monitoring accuracy and performance, and the goal is to learn good scheduling policies.

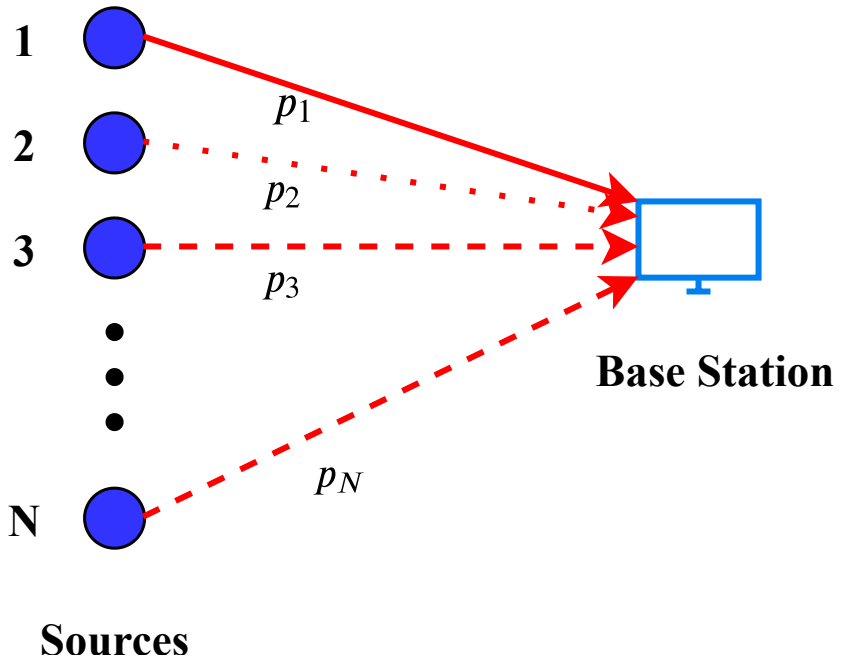


Figure 3-2: Multiple source monitoring

Consider a system with N sources sending updates over a network such that only one source can transmit at any given time-slot (due to interference/capacity constraints). We assume reliable channels, i.e. when a source is chosen to transmit an update, it is delivered to the monitor without fail in the next time-slot. Freshness aware scheduling in such single-hop wireless networks has been the focus of a lot of recent work in the AoI community [15, 16, 17, 18, 19, 1, 30].

We now create an epoch-based structure and set up an online learning formulation for multiple sources as we did in the single source setting. As before, we divide the time horizon into T epochs, where each epoch is of length M time-slots. At the beginning of epoch k, the scheduler needs to decide a scheduling policy π_k which specifies when to schedule each sensor. Once the epoch is over, a cost of the form $C_k(\pi_k)$ is incurred and a new epoch begins. Using cost information about previous decisions, we again choose a scheduling policy π_{k+1} for epoch $k+1$ and the process repeats itself.

We maintain variables $A^{(1)}, ..., A^{(N)}$ which track the evolution of AoI for each

source within an epoch. The evolution of AoI for source i in epoch k is described by the following equation:

$$A^{(i)}(j+1) = \begin{cases} A^{(i)}(j)+1, & \text{if } i \notin \pi_k(j) \\ 1, & \text{if } i \in \pi_k(j), \end{cases} \tag{3.11}$$

where j is an index denoting the current time-slot within the epoch and $\pi_k(j)$ is the scheduling decision set in time-slot j of epoch k.

Similar to the single-source formulation, we relax the interference constraint in the last time-slot of every epoch. This ensures that the AoI of every source is set to 1 at the beginning of each epoch and we do not need to maintain history of AoI across different epochs. Practically, we justify this as a two time-scale assumption. A scheduling policy remains fixed over an epoch (the longer time-scale) and specifies how to take decisions over time-slots (the shorter time-scale). Once the epoch ends, the system resets. The system designer observes the performance of the scheduling policy that was chosen and specifies a new scheduling policy for the next epoch.

We consider scheduling policies as a sequence of M scheduling decisions, specifying which source gets to transmit in each time-slot within an epoch. We denote this space of scheduling policies by Π^M.

We assume that the cost for delayed information in any time-slot can be represented as a general function of the current AoIs. Let $f_k(A^{(1)}, ..., A^{(N)})$ represent this AoI cost function in epoch k, where $f_k : \mathbb{Z}^{+N} \to [0, D]$ is a **bounded** mapping from the set of AoI vectors to costs. The total cost of choosing a policy π in epoch k is given by

$$\sum_{j=1}^{M} f_k(A^{(1)}(j), ..., A^{(N)}(j)), \tag{3.12}$$

where the AoIs evolve under policy π according to (3.11).

We have an **unconstrained adversary** who chooses the sequence of bounded cost functions $f_k(\cdot)$ for each epoch k and we need to learn the best scheduling policy in response to any sequence of cost functions. Without loss of generality we assume that $f_k(\cdot)$ are normalized such that the total cost of any policy $C_k(\cdot)$, given by (3.12), lies in the set $[0,1]$.

At the end of every epoch, the scheduler receives feedback in terms of $C_k(\cdot)$. In the case of full feedback, the entire function $C_k(\cdot)$ is revealed, meaning cost for all scheduling policies is known when the epoch ends. For the case of bandit feedback, only the cost for the chosen scheduling policy $C_k(\pi_k)$ is revealed.

Observe that the multiple source problem with the feedback structure as set up above can also be viewed as prediction with expert advice. Now, instead of AoI thresholds as experts, we have scheduling policies as experts and our goal is to *compete with the best scheduling policy in hindsight.*

Thus, we can directly apply online learning algorithms as done in Section 3.1 to the multiple source setting. The regret bounds, however, are not the same. This is because the number of scheduling policies of length M time-slots scales as $\Theta(N^M)$.

Lemma 4. *Consider the multiple source online scheduling problem with $N \geq 2$. If an online algorithm Alg. has an upper bound $f(M,T)$ on its expected regret in the single source setting, then the same algorithm run using scheduling policies as experts for the multiple source problem has the following regret bound:*

$$\mathbb{E}[Regret_T(Alg.)] \leq \bar{C} f(N^M, T),$$

where $\bar{C} > 0$ is a constant that does not depend on any other parameters.

We observe that while the dependence of regret on T remains the same, it becomes exponentially worse in M for the multiple source setting. This also highlights a key computational challenge in the multiple source setting. The number

of policies scales exponentially with M, the length of an epoch. Thus the optimization step in FTPL (Algorithm 1) has computational complexity that scales exponentially with M. Similar computational challenges are faced in implementing exponential weight algorithms like EXP3 for the bandit feedback case of the multiple source setting. This makes it hard to implement these online scheduling schemes in practice.

This is not surprising, given that the offline problem of finding the best scheduling policy of length M time-slots in the setting with cost functions known beforehand also requires computation that scales as $O(N^M)$ (see [30]). In [1], the authors analyzed the setting where cost functions can be represented as sums of separate cost functions that depend only on the AoI of each source individually. If the individual cost functions of AoI for each source are monotone increasing, then a low complexity heuristic based on the Whittle index approach can be found which is nearly optimal. We will use this observation to design low complexity online policies that keep track of the best scheduling policy in hindsight.

3.2.A Online Whittle-Index Scheduling

We modify the general multiple source setting so as to solve the computational challenge discussed above.

First, we consider scheduling policies as mappings from the set of AoI vectors $A^{(1)}, ..., A^{(N)}$ to the set of sources, i.e. $\pi : \mathbb{Z}^{+N} \to \{1, ..N\}$. Given the AoIs of all sources at time-slot j within an epoch, a policy π specifies which source gets to transmit. We denote this space of scheduling policies by Π.

Second, we assume that the cost function splits as a sum of individual cost functions of AoI, where $f_k^{(1)}, ..., f_k^{(N)}$ represent individual AoI cost functions for each source in epoch k. Then, the total cost of choosing a policy π in epoch k is

given by

$$C_k(\pi) = \frac{1}{NM} \sum_{j=1}^{M} \sum_{i=1}^{N} f_k^{(i)}(A^{(i)}(j)), \tag{3.13}$$

where the AoIs evolve under policy π according to (3.11). We multiply a normalizing constant $\frac{1}{NM}$ to the sum AoI cost to make regret analysis neater.

We assume that the cost functions $f_k^{(i)} : \mathbb{Z}^+ \to \mathbb{R}^+$ are fixed during an epoch and bounded monotone increasing functions of AoI, i.e. if $x > y$ then $f_k^{(i)}(x) \geq f_k^{(i)}(y)$ and $f_k^{(i)}(\cdot) \leq D$. An unconstrained adversary is free to change these bounded cost functions arbitrarily across epochs.

Finally, instead of receiving feedback directly in terms of cost of scheduling policies $C_k : \Pi^M \to [0,1]$, we consider feedback in terms of individual cost functions of AoI. So, at the end of epoch k, a cost $C_k(\pi)$ is incurred (given by (3.13)) and AoI cost functions $f_k^{(1)}, ..., f_k^{(N)}$ are revealed to the scheduler, either completely or partially. In the case of bandit feedback, we will construct estimates of the entire cost functions $\hat{f}_k^{(1)}, ..., \hat{f}_k^{(N)}$.

Note that within an epoch, the scheduling problem that we want to solve is an instantiation of the functions of age problem described in [1].

We briefly review the multiple source setting with fixed AoI cost functions studied in [1]. Consider N sources and a given set of increasing AoI cost functions $f^{(1)}, ..., f^{(N)}$. Our goal is to minimize average age cost over an infinite horizon. The Whittle index policy maps the current vector of source AoIs to a scheduling decision. If the current AoI for source i is $A^{(i)}$ then the Whittle policy is given by

$$\pi^{\text{Whittle}}(A^{(1)}, ..., A^{(N)}) \triangleq \arg\max_{i \in \{1,...,N\}} \{W^{(i)}(A^{(i)})\}, \tag{3.14}$$

where

$$W^{(i)}(x) \triangleq x f^{(i)}(x+1) - \sum_{k=1}^{x} f^{(i)}(k)$$

are Whittle index functions. It was shown in [1] that this Whittle policy is optimal

for $N = 2$ and near optimal in general. For cost functions $f^{(1)}, ..., f^{(N)}$, we denote the Whittle policy given by (3.14) as Whittle$\left(f^{(1)}, ..., f^{(N)}\right)$. Next, we describe how to design a low-complexity online algorithm using Whittle index policies.

Full Feedback

In this setting, we assume that the entire M dimensional AoI cost function $f_k^{(i)}$ for each source i is revealed to the scheduler at the end of the epoch. Instead of looking for the best schedule in every epoch which is computationally expensive, we will use the Whittle index policy as an approximate minimizer. This leads to Algorithm 3, which we call *Follow the Perturbed Whittle Leader* (FPWL).

Algorithm 3: Follow the Perturbed Whittle Leader

Input : parameter $\epsilon > 0$

Set $F_1^{(i)}(j) = j, \forall i \in \{1, ..., N\}, \forall j \in \{1, ..., M\}$

while $t \in 1, ..., T$ **do**

  Set $A^{(1)}, ..., A^{(N)} = \mathbf{1}$

  Sample $\delta_t^{(i)}(j) \sim$ uniform in $[0, 1/\epsilon]$, i.i.d. $\forall i \in \{1, ..., N\}$ and $\forall j \in \{1, ..., M\}$

  Compute $\gamma_t^{(i)}(j) = \sum_{k=1}^{j} \delta_t^{(i)}(j), \forall i, j$

  Choose scheduling policy $\pi_t = \text{Whittle}\left(F_t^{(1)} + \gamma_t^{(1)}, ..., F_t^{(N)} + \gamma_t^{(N)}\right)$

  Incur loss $= C_t(\pi_t)$ over epoch t and observe feedback on $f_t^{(1)}, ..., f_t^{(N)}$

  In case of bandit feedback, construct cost estimates $\hat{f}_t^{(i)}, \forall i \in \{1, ..., N\}$ using linear interpolation

  Update

$$F_{t+1}^{(i)} = \begin{cases} F_t^{(i)} + f_t^{(i)}, \forall i \in \{1, ..., N\}, \text{ if full feedback} \\ F_t^{(i)} + \hat{f}_t^{(i)}, \forall i \in \{1, ..., N\}, \text{ if bandit feedback.} \end{cases}$$

end

FPWL can be divided into three major steps. First, accumulate the entire history of cost functions that the scheduler has seen until the current epoch in $F_t^{(1)}, ..., F_t^{(N)}$. Since cost functions in each epoch are increasing in terms of AoI,

their sums $F_t^{(i)}$ are also increasing. Second, perturb these accumulated cost functions in a manner such that they remain increasing functions of AoI but are still amenable for FTPL style analysis. Third, instead of computing the best possible scheduling policy for these accumulated and perturbed cost functions, use the Whittle index policy as an approximate minimizer.

Computing the Whittle policy has complexity $O(NM)$ since it involves a maximization over N quantities for at most M steps. Further, generating the random perturbations γ_t in steps 4 and 5 also takes at most $O(NM)$ computation. Thus, the algorithm above resolves the computational challenge involved in implementing FTPL directly for the online scheduling problem.

Proving regret bounds our proposed algorithm is much harder than in the single or multiple source settings studied earlier. We overcome three significant problems: 1) perturbations in Algorithm 3 are made to the AoI cost functions rather than policies, unlike regular FTPL; 2) because of this, cost perturbations are not i.i.d. across policies; 3) the Whittle index policy is only an approximate minimizer rather than an exact minimizer of the average AoI cost. Despite these challenges, we are able to show that FPWL achieves low regret compared to any fixed scheduling policy, if the Whittle policy is “close" to the actual optimal policy. Theorem 13 describes an upper bound on the regret of FPWL when compared to the best fixed scheduling policy in hindsight. The parameter α measures the closeness between the Whittle policy and an optimal policy. For a detailed definition of α see Appendix 3.5.C.

Theorem 13. *Follow the perturbed Whittle leader (FPWL) based scheduling described by Algorithm 3 with* $\epsilon = \sqrt{\frac{2M}{ND^2T}}$ *achieves the following upper bound on expected regret:*

$$\mathbb{E}[Regret_T(FPWL)] \leq \alpha T + 2D\sqrt{2MNT},$$

where the expectation is taken over the random perturbations.

Proof. See Appendix 3.5.D. □

We proved in Chapter 2 that the Whittle index policy is optimal for $N = 2$, meaning $\alpha = 0$ and so we can achieve sublinear regret with respect to the best fixed scheduling policy when there are 2 sources. Further, recent work in [34] suggests that $\alpha \to 0$ as $N \to \infty$ meaning that FPWL can achieve sublinear regret for large system sizes. Simulations in both [1] and [34] indicate that α is very small for most problems of practical interest.

Importantly, note that there is no way to get sublinear static regret by using FPWL if the Whittle solution is not exactly optimal for the offline problem. In this case, even if the cost functions are the same in every epoch, there would be a small gap $\alpha > 0$ between the cost of the Whittle policy and the optimal policy in every epoch. The small constant gap will add up to give linear regret. Thus, the term αT in the regret bound above accounts for this cost of using an approximate optimization oracle rather than an exact one, and cannot be eliminated.

Dynamic Regret

A drawback of the online learning formulation is that sublinear regret is only possible when comparing to a simple class of policies since there are no constraints on the adversary choosing the cost functions. A more general notion of regret is *dynamic regret* where cost is compared to an algorithm which chooses the best scheduling policy in *each* epoch rather than the best fixed policy across epochs. Dynamic regret of an algorithm that chooses scheduling policy π_k in epoch k is defined as follows:

$$\text{D-Regret}_T(\text{Alg.}, C) \triangleq \sup_{C_{1,\ldots,T} \in C} \left\{ \sum_{k=1}^{T} C_k(\pi_k) - \sum_{k=1}^{T} \min_{\pi \in \Pi} C_k(\pi) \right\}, \tag{3.15}$$

where C incorporates constraints on the adversary. It is easy to show that if there are no constraints on how an adversary is allowed to choose the cost functions $C_1, ..., C_T$ then achieving sublinear dynamic regret is not possible. Thus, the definition of dynamic regret includes C which is the class of cost function sequences over which the regret is being considered and incorporates constraints on the adversary.

A number of recent works on online learning consider the problem of minimizing dynamic regret by constraining how the sequence of cost functions change over time (see [103, 104, 105, 106]). We follow the approach of [103] and [105] by defining the quantity V_T which measures the variation of a given sequence of cost functions as follows:

$$\sum_{k=2}^{T} \max_{\pi} \left| C_{k-1}(\pi) - C_k(\pi) \right| \leq V_T. \tag{3.16}$$

Suppose we know that any sequence of cost functions chosen by the adversary is going to satisfy the inequality (3.16). Then, we denote the set of allowable sequence of cost functions by $C(V_T)$ and define the quantity V_T as the variation budget given to the adversary.

We can also use the Whittle index approach to achieve low dynamic regret. If V_T is known to be sublinear in T beforehand, then simply using the Whittle index policy for the cost functions revealed in the previous epoch is sufficient to get low dynamic regret. Specifically, set $f_0^{(i)} = \{1, ..., M\}, \forall i \in \{1, ..., N\}$ and let the scheduling policy in epochs k be given by:

$$\pi_k = \text{Whittle}\left(f_{k-1}^{(1)}, ..., f_{k-1}^{(N)} \right). \tag{3.17}$$

We call this algorithm *Follow the Dynamic Whittle Leader* (FDWL).

Lemma 5. *The dynamic regret of FDWL satisfies*

$$D\text{-}Regret_T(FDWL, C(V_T)) \leq \alpha T + V_T + D,$$

where V_T is the variation budget as defined in (3.16) *and D is the upper-bound on AoI cost functions.*

Proof. See Appendix 3.5.E. □

An important point to note here is that FDWL should only be used when an upper bound on V_T that grows sublinearly with T is known *a priori*. If no such upper bound is known and V_T grows linearly with T, then it can be shown that FDWL incurs static regret that is linear in T meaning it performs worse than FPWL (Algorithm 3). This neatly splits the full feedback setting into two regimes. If V_T is known to be sublinear use FDWL to get sublinear dynamic regret. Otherwise, use the entire history of cost functions as in FPWL to get sublinear static regret.

Algorithm 3 also highlights the strength of follow-the-leader style algorithms in solving online optimization problems with combinatorial structure. If a low complexity solution is known to the offline problem as with the Whittle index then it can be incorporated into FTPL as an optimization oracle. On the other hand, exponential weight update based algorithms like EXP3 [102] or EXP3.S [105] are standard in the bandit feedback case. Incorporating a Whittle index solution directly in these algorithms is not possible. This makes designing computationally efficient online learning algorithms for bandit feedback harder in the multiple source setting. We develop a heuristic solution for this below.

Bandit Feedback

For bandit feedback, the cost function of AoI associated with source i is only revealed during the time-slots in which it sends an update.Specifically, if at time-slot j within epoch k the policy π_k schedules sensor i, then $f_k^{(i)}(A^{(i)}(j))$ is revealed to the scheduler. This happens for every time-slot in the epoch.

To run FPWL and FDWL on this incomplete feedback we need to construct estimates of the cost functions denoted by $\hat{f}_k^{(1)}, ..., \hat{f}_k^{(N)}$. We do this by linearly

interpolating between the revealed values of $f_k^{(i)}$ for each source i. Algorithm 4 describes the details. Importantly, constructing the linear interpolating cost estimates for a single source requires a single pass over $1, ..., M$. Thus, constructing $\hat{f}_k^{(1)}, ..., \hat{f}_k^{(N)}$ has a computational complexity $O(NM)$. So, our modified versions of FPWL and FDWL for bandit feedback remain computationally efficient. However, since these estimates are not guaranteed to be unbiased, regret analysis in the bandit feedback case becomes challenging.

Algorithm 4: Linearly Interpolating Cost Function Estimate for source i

Input : $X \subseteq \{1, ..., M\}$ for which $f_k^{(i)}$ is known, D known upper bound on $f_k^{(i)}$

Output: Estimate $\hat{f}_k^{(i)}$ that is an increasing AoI cost function

Add 0 to X and set $f_k^{(i)}(0) = 0$
if $M \notin X$ **then**
    set $f_k^{(i)}(M) = D$ and add M to X
end
Sort X in increasing order $\{0, x_1, ..., x_l, M\}$
while $h \in 1, ..., M$ **do**
    if $h \notin X$ **then**
        Find k such that $x_k < h < x_{k+1}$ and $x_k, x_{k+1} \in X$
        Set $\hat{f}_k^{(i)}(h) = f_k^{(i)}(x_k) + (h - x_k)\frac{f_k^{(i)}(x_{k+1}) - f_k^{(i)}(x_k)}{x_{k+1} - x_k}$
    else
        Set $\hat{f}_k^{(i)}(h) = f_k^{(i)}(h)$.
    end
end

3.3 Mobility Tracking

We now apply the results we have developed to a mobility tracking problem. Consider N nodes moving around in the two dimensional plane whose positions needs to be tracked by a central base station (BS). At any given time, only one of these nodes can send an update about its current location and velocity to the

BS. The BS keeps track of the location of the nodes by storing the most recently received update from each node. Our goal is to design a scheduling policy that minimizes total tracking error between the location estimates at the BS and the actual locations of the nodes.

Observe that if the current velocity of a node i is v_i, then its tracking error grows linearly with its AoI. That is, if the BS hasn't received an update from node i for time A_i, then the tracking error is $v_i A_i$. In practical scenarios, node mobility patterns and velocities are often unknown beforehand, non-stationary, and possibly adversarial. Thus, our mobility tracking problem can be viewed as a weighted-AoI minimization problem with time-varying velocities acting as weights. Since a new update from a node only contains information about its current location and velocity, *this fits into the multiple source bandit feedback setting*.

We will discuss two specific mobility models and apply our online algorithms to show that they outperform static AoI based scheduling. Note that while cost functions being static within an epoch and resetting of AoIs at the beginning of every epoch are necessary for regret analysis, these assumptions are not required to implement our algorithms in practice.

3.3.A Levy Mobility

In this scenario, we simulate the nodes' motion using Levy mobility. This is a realistic mobility model that closely matches human mobility in practice [107]. A node's motion is described in a series of *steps*. A step is represented by the tuple (v, θ, t_f, t_p) - a velocity v picked randomly in the interval $[0, v_{\max}]$, an angle θ picked uniformly from the interval $[0, 2\pi]$, a flight time t_f picked at random from $\{1, ..., T_{f\max}\}$ and a pause time t_p picked at random from $\{1, ..., T_{p\max}\}$. The node then moves with the velocity v, in the direction θ for t_f time-slots and then pauses at its location for t_p time-slots. This leads to a bursty random walk pattern with time-varying velocities.

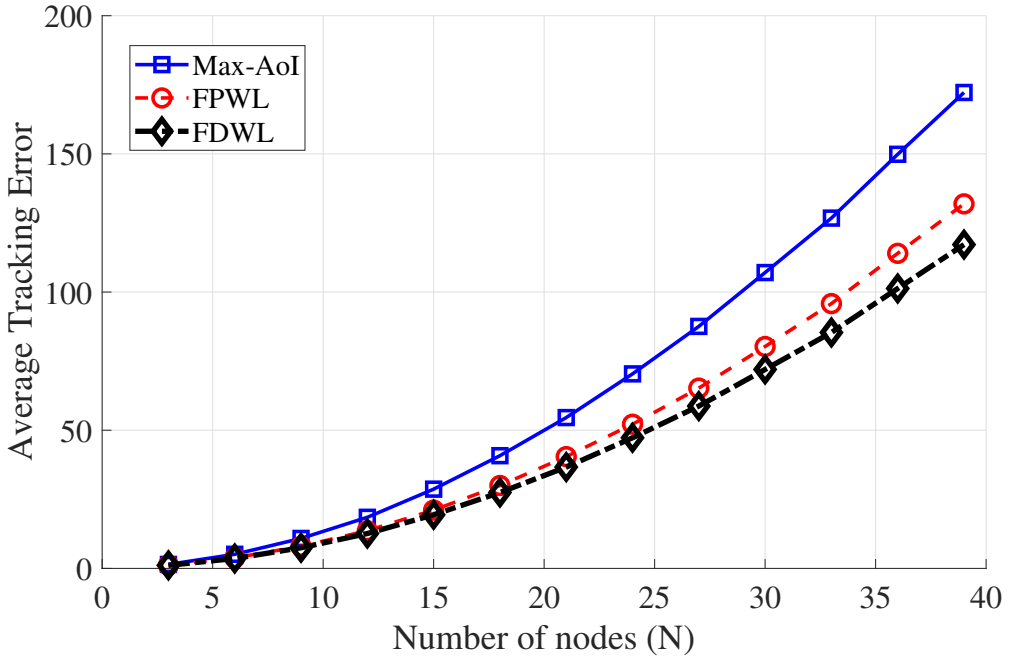


Figure 3-3: Levy Mobility: Average Tracking Error v/s number of nodes

We consider N nodes executing Levy mobility. An adversary sets the values of $v_{\max}$ for each node from the set $\{0.1, 0.5, 5\}$ designating it as a slow, medium or fast node. Overall, $N/3$ nodes each are designated as fast, medium and slow, but the scheduler doesn't know which. We set $T_{f\max} = 50$ and $T_{p\max} = 30$ for all nodes.

The scheduler does not know beforehand that there is inherent asymmetry in the motion of the nodes. An oblivious static scheduling policy is max-AoI: let the node with the maximum AoI transmit in every time-slot. From Figure 3-3, we observe that using FPWL in this setting outperforms the max-AoI scheduling policy (by about 25%). Further, FDWL outperforms both max-AoI (by about 33%) and FPWL (by about 10%). This is because velocities under Levy mobility are slowly varying in time and not adversarial, allowing a dynamic regret based algorithm such as FDWL to work better than FPWL. We set the epoch length M to 200 time-slots for both FPWL and FDWL, and the number of epochs T to 500, thus running the simulation for 100000 time-slots.

3.3.B Adversarial Mobility

In this scenario, we assume that the nodes execute a mobility pattern that is chosen by a reactive adversary in response to the scheduling policies. In every epoch, the nodes execute Brownian motion (moving in random directions at a fixed velocity). An adversary assigns velocities to nodes such that they are inversely proportional to their scheduling priorities.

For FPWL, the scheduling policy in epoch t is given by $\pi_t = \text{Whittle}\big(F_t^{(1)} + \gamma_t^{(1)}, ..., F_t^{(N)} + \gamma_t^{(N)}\big)$. So, the velocity $v_t^{(i)}$ of node i in epoch t is chosen to satisfy $v_k^{(i)} \propto c^{(i)}||F_t^{(i)}||^{-2}$. Similarly, for FDWL, the scheduling policy in epoch t is given by $\pi_t = \text{Whittle}\big(\hat{f}_{t-1}^{(1)}, ..., \hat{f}_{t-1}^{(N)}\big)$, where we use estimated cost functions since our setting involves bandit feedback. So, $v_t^{(i)}$ is chosen to satisfy $v_k^{(i)} \propto c^{(i)}||\hat{f}_{t-1}^{(i)}||^{-2}$. For the max-AoI policy, the velocity $v_t^{(i)}$ is chosen to satisfy $v_k^{(i)} \propto c^{(i)}$. Here $c^{(i)}$ are parameters which are fixed across epochs and also chosen by the adversary to ensure that the motion of nodes has inherent asymmetry unknown to the scheduler. Overall, $N/3$ nodes each are assigned $c^{(i)} = 0.1$, $c^{(i)} = 0.4$ and $c^{(i)} = 40$. If the scheduler observes a node was moving fast in the previous epochs and assigns it a larger cost, then the adversary assigns it a low velocity in the next epoch so as to confuse the scheduler. The sum total of velocities is normalized and remains fixed in every time-slot ensuring that the *adversary is equally powerful irrespective of scheduling policies.*

Under this adversarial model, we observe in Figure 3-4 that while FPWL still outperforms max-AoI (by about 8%), FDWL performs significantly worse than both FPWL and max-AoI (about 50% worse). This is consistent with our results from theory - when cost functions are quickly varying and adversaries are unconstrained and reactive, dynamic regret based algorithms like FDWL perform worse than static regret algorithms like FPWL.

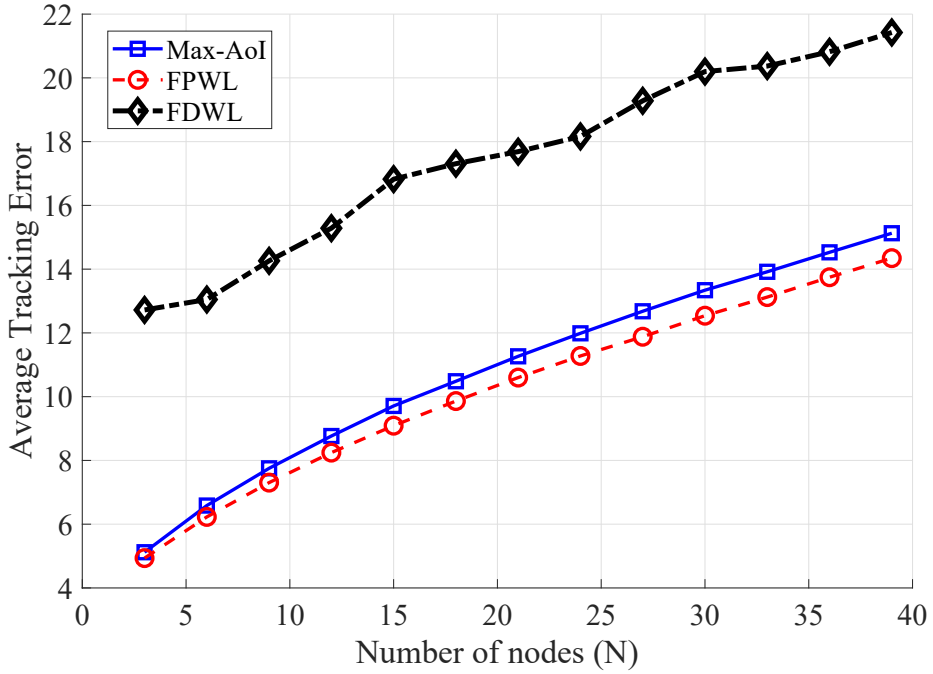


Figure 3-4: Adversarial Mobility: Average Tracking Error v/s number of nodes

3.4 Summary

In this chapter, we have formulated a general framework for online monitoring and scheduling for non-stationary sources. Specifically, we handle unknown, time-varying, and possibly adversarial cost functions of AoI and design algorithms that attempt to learn the best scheduling policies in an online fashion. We apply our results to a mobility tracking problem and show that our online learning algorithms outperform oblivious AoI based schemes and are able to learn information about the underlying source dynamics.

Possible directions of future work involve applying our online scheduling framework to different problems of practical interest, and incorporating unreliable channels and noisy feedback about the costs into our framework.

3.5 Appendix

3.5.A Proof of Lemma 2

Let the AoI cost function in epoch k be $f_k(\cdot)$, let the transmission cost be C and let the chosen sampling threshold be x. We set $t = 1$ at the beginning of the epoch. Then,

$$C_k(x) = \sum_{t=1}^{M} f_k(A(t)) + Cu(t). \tag{3.18}$$

Note that the AoI at time $t = 1$ is $A(1) = 1$, since each epoch begins after a new transmission. Since the threshold is set to x, no new update is sent till time-slot x at which point the AoI reaches x. Now, a new sample is generated and sent, so the AoI drops to 1 in the next time-slot. This process repeats in cycles of x time-slots. Since the epoch consists of M time-slots, there are $\lfloor \frac{M}{x} \rfloor$ complete cycles of length x. The sum of costs over each of these cycles is $\left(\sum_{j=1}^{x} f_k(j) + C \right)$ since the AoI goes from 1 to x and there is a transmission at the end.

The final cycle is of length $r = M \mod x$ where $a \mod b$ is the remainder when a is divided by b. There is a mandatory transmission in the final time-slot regardless of the AoI exceeding the threshold to finish the epoch. Thus,

$$C_k(x) = \left\lfloor \frac{M}{x} \right\rfloor \left(\sum_{j=1}^{x} f_k(j) + C \right) + \mathbb{1}_{r>0} \left(\sum_{j=1}^{r} f_k(j) + C \right). \tag{3.19}$$

This completes the proof.

3.5.B Proof of Corollary 2

Let the regret of algorithm π be $f(M,T)$. From Lemma 3, we know that for any bounded sequence $f_1, ..., f_T$

$$\mathbb{E}\left[\left\{\sum_{k=1}^{T}\sum_{t\in E_k} f_k(A^{\pi}(t)) \;+\; Cu^{\pi}(t) \;-\; \sum_{k=1}^{T}\sum_{t\in E_k} f_k(A^{\pi^*}(t)) \;+\; Cu^{\pi^*}(t)\right\}\right] \;\leq\; f(M,T).$$

Dividing the equation about by MT, we get

$$\frac{1}{MT}\mathbb{E}\left[\sum_{k=1}^{T}\sum_{t\in E_k} f_k(A^{\pi}(t))+Cu^{\pi}(t)\right] \leq \frac{1}{MT}\mathbb{E}\left[\sum_{k=1}^{T}\sum_{t\in E_k} f_k(A^{\pi^*}(t))+Cu^{\pi^*}(t)\right]+\frac{f(M,T)}{MT}.$$

Taking the limit supremum as T goes to infinity and using the fact that $f(M,T)$ grows sublinearly in T, we get the required result.

3.5.C Closeness of Whittle and Optimal Policies

Here, we define α, the parameter that measures the closeness of the Whittle index policy to an optimal policy within an epoch.

Consider a set of monotone and bounded AoI cost functions $f^{(1)}, ..., f^{(N)}$ such that for all i, if $x > y$ then $f^{(i)}(x) \geq f^{(i)}(y)$ and $f^{(i)}(M) \leq D$. Let Whittle(f) denote the Whittle policy for this set of cost functions, as defined in (3.14). Let Opt(f) denote an optimal policy for this set of cost functions.

Now consider another set of monotone bounded AoI cost functions $g^{(1)}, ..., g^{(N)}$ with the same upper bound D. Given a scheduling policy π, let

$$C_g(\pi) \triangleq \frac{1}{NM}\sum_{j=1}^{M}\sum_{i=1}^{N} g^{(i)}(A^{(i)}(j)), \tag{3.20}$$

where the AoIs evolve under policy π. This is the total sum cost of policy π under the cost functions $g^{(1)}, ..., g^{(N)}$. We make the following assumption on the struc-

ture of Whittle index and optimal policies when the epoch length M is long.

Assumption 1. *For any two sets of bounded monotone sets of cost functions $f^{(1)}, ..., f^{(N)}$ and $g^{(1)}, ..., g^{(N)}$ with a fixed known upper bound D, the following holds:*

$$\left| C_g\big(Whittle(f)\big) - C_g\big(Opt(f)\big) \right| \leq \alpha, \tag{3.21}$$

where α is a small constant that can depend on N, M and D.

Note that this assumption is stronger than just assuming that the Whittle index policy has near optimal performance over long epochs. We assume that the Whittle policy is also close to the optimal policy in its sequence of scheduling decisions. Thus, given arbitrary bounded cost functions, the two policies $C_g\big(\text{Whittle}(f)\big)$ and $C_g\big(\text{Opt}(f)\big)$ have average costs that are close to each other. This is a Lipschitz like assumption on the policy space and cost functions for the scheduling problem. The motivation for this comes from results in [1], where it was shown that the Whittle policy is exactly optimal for $N = 2$ as $M \to \infty$, meaning that we can set $\alpha = 0$. It was also observed via simulations that the Whittle policies are structurally similar to optimal policies for general N. Results on asymptotic optimality of the Whittle policy [34] further suggest that $\alpha \to 0$ as $N \to \infty$.

3.5.D Proof of Theorem 13

Suppose $f_k^{(1)}, ..., f_k^{(N)}$ are the AoI cost functions during epoch k. In each epoch, the cost functions $f_k^{(i)} : \{1, ..., M\} \to \mathbb{R}^+$ are bounded monotone increasing functions of AoI, i.e. if $x > y$ then $f_k^{(i)}(x) \geq f_k^{(i)}(y)$ and $f_k^{(i)}(\cdot) \leq D$. D is fixed and known beforehand. Let $C_k(\pi)$ be the cost incurred in epoch k by using scheduling policy π, given by (3.13). For a set of cost functions $f^{(1)}, ..., f^{(N)}$, the Whittle scheduling policy is represented by Whittle$(f^{(1)}, ..., f^{(N)})$. For the same set of cost

functions, an optimal scheduling policy is represented by $\text{Opt}(f^{(1)}, ..., f^{(N)})$. We will use these notations throughout the proof.

Similar to [99], we will divide our proof into three steps.

Be-the-Whittle-Leader has low regret

First, we define a hypothetical algorithm called Be-the-Whittle-Leader (BWL). In epoch k, a scheduling policy π_k^{BWL} is chosen as follows:

$$\pi_k^{\text{BWL}} = \text{Whittle}\left(\sum_{t=1}^{k} f_t^{(1)}, ..., \sum_{t=1}^{k} f_t^{(N)}\right). \tag{3.22}$$

BWL applies the Whittle procedure to the sum of cost functions seen from epoch 1 through k and uses this as the scheduling policy in epoch k. Clearly, this requires knowledge of the cost functions in the current epoch k and hence, it is not an online learning algorithm. In this step, we will show that this algorithm, which looks ahead one epoch into the future, achieves low regret. In the next two steps, we will show that the gap between FPWL and BWL increases only sublinearly in T, completing the proof.

Note from (3.14) that if all cost functions are multiplied by a fixed positive constant, the Whittle and optimal policies remain unchanged. So, we rewrite BWL as:

$$\pi_k^{\text{BWL}} = \text{Whittle}\left(\frac{1}{k}\sum_{t=1}^{k} f_t^{(1)}, ..., \frac{1}{k}\sum_{t=1}^{k} f_t^{(N)}\right). \tag{3.23}$$

Since AoI cost functions in each epoch are upper-bounded by D, their averages are also upper-bounded by D. Thus, we can apply Assumption 1 to the BWL policy. This results in the following inequality $\forall k \in 1, ..., T$

$$C_k(\pi_k^{\text{BWL}}) \leq C_k\left(\text{Opt}\left(\frac{1}{k}\sum_{t=1}^{k} f_t^{(1)}, ..., \frac{1}{k}\sum_{t=1}^{k} f_t^{(N)}\right)\right) + \alpha. \tag{3.24}$$

Summing the equation above for $k = 1, ..., T$, we get

$$\sum_{k=1}^{T} C_k(\pi_k^{\text{BWL}}) \leq \sum_{k=1}^{T} C_k\left(\text{Opt}\left(\frac{1}{k}\sum_{t=1}^{k} f_t^{(1)}, ..., \frac{1}{k}\sum_{t=1}^{k} f_t^{(N)}\right)\right) + \alpha T. \tag{3.25}$$

Now, we claim that

$$\sum_{k=1}^{T} C_k\left(\text{Opt}\left(\frac{1}{k}\sum_{t=1}^{k} f_t^{(1)}, ..., \frac{1}{k}\sum_{t=1}^{k} f_t^{(N)}\right)\right) \leq \min_{\pi\in\Pi} \sum_{k=1}^{T} C_k(\pi). \tag{3.26}$$

To prove this, we use induction. For the base case, observe that the following holds by the definition of $\text{Opt}(\cdot)$.

$$C_1\left(\text{Opt}(f_1^{(1)}, ..., f_N^{(1)})\right) = \min_{\pi\in\Pi} C_1(\pi) \tag{3.27}$$

Further, since costs across epochs are additive, we have the following for any $l \in 1, ..., T$:

$$\sum_{k=1}^{l} C_k\left(\text{Opt}\left(\frac{1}{l}\sum_{t=1}^{l} f_t^{(1)}, ..., \frac{1}{l}\sum_{t=1}^{l} f_t^{(N)}\right)\right) = \min_{\pi\in\Pi} \sum_{k=1}^{l} C_k(\pi). \tag{3.28}$$

The above equation simply states that a policy that is optimal for the sum of cost functions from $1, ..., l$ is also the best fixed scheduling policy to be used over the epochs $1, ..., l$.

Let's assume the following for some l:

$$\sum_{k=1}^{l} C_k\left(\text{Opt}\left(\frac{1}{k}\sum_{t=1}^{k} f_t^{(1)}, ..., \frac{1}{k}\sum_{t=1}^{k} f_t^{(N)}\right)\right) \leq \min_{\pi\in\Pi} \sum_{k=1}^{l} C_k(\pi). \tag{3.29}$$

Then, adding the term $C_{l+1}\left(\text{Opt}\left(\frac{1}{l+1}\sum_{t=1}^{l+1} f_t^{(1)}, ..., \frac{1}{l+1}\sum_{t=1}^{l+1} f_t^{(N)}\right)\right)$ to both sides we get:

$$\sum_{k=1}^{l+1} C_k\left(\text{Opt}\left(\frac{1}{k}\sum_{t=1}^{k} f_t^{(1)}, ..., \frac{1}{k}\sum_{t=1}^{k} f_t^{(N)}\right)\right) \leq \min_{\pi\in\Pi}\left\{\sum_{k=1}^{l} C_k(\pi)\right\} + C_{l+1}\left(\text{Opt}\left(\frac{1}{l+1}\sum_{t=1}^{l+1} f_t^{(1)}, ..., \frac{1}{l+1}\sum_{t=1}^{l+1} f_t^{(N)}\right)\right). \quad (3.30)$$

Note that the first term in the RHS is a minimum over all policies, so it can be upper bounded by replacing π with any policy. This implies:

$$\sum_{k=1}^{l+1} C_k\left(\text{Opt}\left(\frac{1}{k}\sum_{t=1}^{k} f_t^{(1)}, ..., \frac{1}{k}\sum_{t=1}^{k} f_t^{(N)}\right)\right) \leq \sum_{k=1}^{l+1} C_k\left(\text{Opt}\left(\frac{1}{l+1}\sum_{t=1}^{l+1} f_t^{(1)}, ..., \frac{1}{l+1}\sum_{t=1}^{l+1} f_t^{(N)}\right)\right). \quad (3.31)$$

Using (3.28) we can rewrite this as:

$$\sum_{k=1}^{l+1} C_k\left(\text{Opt}\left(\frac{1}{k}\sum_{t=1}^{k} f_t^{(1)}, ..., \frac{1}{k}\sum_{t=1}^{k} f_t^{(N)}\right)\right) \leq \min_{\pi\in\Pi}\sum_{k=1}^{l+1} C_k(\pi). \quad (3.32)$$

Thus, assuming (3.29), we were able to prove (3.32). By induction on l, this proves (3.26). Combining (3.26) with (3.25), we get:

$$\sum_{k=1}^{T} C_k(\pi_k^{\text{BWL}}) \leq \min_{\pi\in\Pi}\sum_{k=1}^{T} C_k(\pi) + \alpha T. \quad (3.33)$$

Finally, (3.33) together with the definition of static regret (3.7) implies that:

$$\text{Regret}_T(\text{BWL}) \leq \alpha T. \quad (3.34)$$

Be-the-Perturbed-Whittle-Leader has low regret

Now, we consider a policy called Be-the-Perturbed-Whittle-Leader (BPWL). This is similar to the BWL policy, but it involves adding an extra perturbation to the cost functions before computing the Whittle index.

We first describe how the perturbation is generated. First, we generate NM i.i.d. random variables $\delta^{(i)}(j) \sim \text{Uniform}([0, 1/\epsilon]), \forall i \in 1, ..., N$ and $\forall j \in 1, ..., M$. We collect these random variables into N vectors $\delta^{(1)}, ..., \delta^{(N)}$, where each vector $\delta^{(i)} \in \mathbb{R}^M$. Using these, we create monotonically increasing random vectors $\gamma^{(1)}, ..., \gamma^{(N)}$ as follows:

$$\gamma^{(i)}(j) = \sum_{k=1}^{j} \delta^{(i)}(k), \forall i \in 1, ..., N \text{ and } \forall j \in 1, ..., M. \tag{3.35}$$

Now, we have N M-dimensional random vectors that are monotonically increasing. Given any set of AoI cost functions $f^{(1)}, ..., f^{(N)}$, the perturbation procedure is given by:

$$\text{Perturb}\Big(f^{(1)}, ..., f^{(N)}\Big) = \Big(f^{(1)} + \gamma^{(1)}, ..., f^{(N)} + \gamma^{(N)}\Big). \tag{3.36}$$

Now, we can describe the hypothetical algorithm called Be-the-Perturbed-Whittle-Leader (BPWL). In epoch k, a scheduling policy π_k^{BPWL} is chosen as follows:

$$\pi_k^{\text{BPWL}} = \text{Whittle}\Big(\text{Perturb}\Big(\sum_{t=1}^{k} f_t^{(1)}, ..., \sum_{t=1}^{k} f_t^{(N)}\Big)\Big), \tag{3.37}$$

where the perturbations are generated i.i.d. for every epoch k. We denote the the perturbations in epoch k by $\gamma_k^{(1)}, ..., \gamma_k^{(N)}$. Since $\gamma_k^{(1)}, ..., \gamma_k^{(N)}$ are monotone increasing functions, they can themselves be viewed as AoI costs. The cost of a policy π with the AoI cost functions $\gamma_k^{(1)}, ..., \gamma_k^{(N)}$ is denoted by $C_{\gamma_k}(\pi)$. We will use this notation later.

Now, consider a sequence such that in epoch k, the AoI cost functions are given by:

$$\left(\tilde{f}_k^{(1)}, ..., \tilde{f}_k^{(N)}\right) = \left(f_k^{(1)} + \gamma_k^{(1)} - \gamma_{k-1}^{(1)}, ..., f_k^{(N)} + \gamma_k^{(N)} - \gamma_{k-1}^{(N)}\right), \tag{3.38}$$

where $\gamma_0^{(i)} = \mathbf{0}$ for all i. Let $\tilde{C}_k(\pi)$ denote the cost of using scheduling policy π in epoch k where the AoI cost functions are $\tilde{f}_k^{(1)}, ..., \tilde{f}_k^{(N)}$.

Observe that the cumulative cost functions in epoch k for this hypothetical sequence are given by:

$$\begin{aligned}\left(\sum_{t=1}^{k} \tilde{f}_t^{(1)}, ..., \sum_{t=1}^{k} \tilde{f}_t^{(N)}\right) &= \left(\sum_{t=1}^{k} f_t^{(1)} + \gamma_k^{(1)}, ..., \sum_{t=1}^{k} f_t^{(N)} + \gamma_k^{(N)}\right) \\ &= \text{Perturb}\left(\sum_{t=1}^{k} f_t^{(1)}, ..., \sum_{t=1}^{k} f_t^{(N)}\right).\end{aligned} \tag{3.39}$$

Because of the way the perturbations are created the cumulative cost functions $\left(\sum_{t=1}^{k} \tilde{f}_t^{(1)}, ..., \sum_{t=1}^{k} \tilde{f}_t^{(N)}\right)$ are monotone increasing functions of AoI in every epoch k. Thus, we can apply (3.33) to this sequence of cost functions to get:

$$\sum_{k=1}^{T} \tilde{C}_k\left(\text{Whittle}\left(\sum_{t=1}^{k} \tilde{f}_t^{(1)}, ..., \sum_{t=1}^{k} \tilde{f}_t^{(N)}\right)\right) \leq \min_{\pi \in \Pi} \sum_{k=1}^{T} \tilde{C}_k(\pi) + \alpha T. \tag{3.40}$$

Now using (3.39) and the definition of BPWL (3.37), we get:

$$\sum_{k=1}^{T} \tilde{C}_k(\pi_k^{\text{BPWL}}) \leq \min_{\pi \in \Pi} \sum_{k=1}^{T} \tilde{C}_k(\pi) + \alpha T. \tag{3.41}$$

Observe that the first term in the RHS is a minimization over all policies π, so we can replace π with $\text{Opt}\left(\sum_{t=1}^{T} f_t^{(1)}, ..., \sum_{t=1}^{T} f_t^{(N)}\right)$. This is the best fixed scheduling

policy for the *original* sequence of cost functions.

$$\sum_{k=1}^{T} \tilde{C}_k(\pi_k^{\text{BPWL}}) \leq \sum_{k=1}^{T} \tilde{C}_k\left(\text{Opt}\left(\sum_{t=1}^{T} f_t^{(1)}, ..., \sum_{t=1}^{T} f_t^{(N)}\right)\right) + \alpha T. \tag{3.42}$$

Note that costs across epochs are additive. So, using (3.38) for any fixed policy π, we get:

$$\sum_{k=1}^{T} \tilde{C}_k(\pi) = \sum_{k=1}^{T} \left(C_k(\pi) + C_{\gamma_k}(\pi) - C_{\gamma_{k-1}}(\pi)\right). \tag{3.43}$$

This further simplifies to:

$$\sum_{k=1}^{T} \tilde{C}_k(\pi) = \sum_{k=1}^{T} \left(C_k(\pi)\right) + C_{\gamma_T}(\pi). \tag{3.44}$$

Applying (3.44) to (3.42) and using the definition of $\text{Opt}(\cdot)$ we get:

$$\sum_{k=1}^{T} \tilde{C}_k(\pi_k^{\text{BPWL}}) \leq \min_{\pi\in\Pi} \sum_{k=1}^{T} C_k(\pi) + \max_{\pi\in\Pi} C_{\gamma_T}(\pi) + \alpha T. \tag{3.45}$$

Using (3.38), we can also conclude that:

$$\sum_{k=1}^{T} C_k(\pi_k^{\text{BPWL}}) \quad \leq \quad \sum_{k=1}^{T} \tilde{C}_k(\pi_k^{\text{BPWL}}) \; + \; \sum_{k=1}^{T} \left| C_{\gamma_k}(\pi_k^{\text{BPWL}}) \; - \; C_{\gamma_{k-1}}(\pi_k^{\text{BPWL}}) \right| \tag{3.46}$$

Combining (3.45) and (3.46), we get:

$$\sum_{k=1}^{T} C_k(\pi_k^{\text{BPWL}}) \leq \min_{\pi\in\Pi} \sum_{k=1}^{T} C_k(\pi) + \sum_{k=1}^{T} \max_{\pi\in\Pi} \left| C_{\gamma_k}(\pi) - C_{\gamma_{k-1}}(\pi) \right| + \max_{\pi\in\Pi} C_{\gamma_T}(\pi) + \alpha T. \tag{3.47}$$

Now, we will use a trick that is standard in online learning literature. We will assume that the adversary choosing the sequence of bounded cost functions is

non-reactive, i.e the sequence of cost functions is chosen in advance. Thus, for the purposes of expected regret, it is sufficient to use the same perturbations $\gamma_1^{(1)}, ..., \gamma_1^{(N)}$ in every epoch (since the adversary cannot learn the perturbations). For this choice of perturbations, (3.47) simplifies to:

$$\mathbb{E}\Big[\sum_{k=1}^{T} C_k(\pi_k^{\text{BPWL}})\Big] \leq \min_{\pi\in\Pi} \sum_{k=1}^{T} C_k(\pi) + 2\max_{\pi\in\Pi} C_{\gamma_1}(\pi) + \alpha T. \tag{3.48}$$

Observe that the maximum value that $\gamma_1^{(i)}(j)$ can have for any value of i and j is M/ϵ. Thus, by the definition of average cost in an epoch (3.13), we know that:

$$\max_{\pi\in\Pi} C_{\gamma_1}(\pi) \leq \frac{M}{\epsilon}. \tag{3.49}$$

Putting everything together, we have:

$$\mathbb{E}\big[\text{Regret}_T(\text{BPWL})\big] \leq 2\frac{M}{\epsilon} + \alpha T. \tag{3.50}$$

While we proved this by assuming an oblivious adversary, the extension to a reactive or non-oblivious adversary is straightforward from Lemma 4.1 in [97].

Follow-the-Perturbed-Whittle-Leader has low regret

In this step, we consider the regret of Follow-the-Perturbed-Whittle-Leader (FPWL) described in Algorithm 3. In epoch k, a scheduling policy is chosen as follows:

$$\pi_k^{\text{FPWL}} = \text{Whittle}\Big(\text{Perturb}\Big(\sum_{t=1}^{k-1} f_t^{(1)}, ..., \sum_{t=1}^{k-1} f_t^{(N)}\Big)\Big), \tag{3.51}$$

Unlike BWL and BPWL, this is a valid online learning algorithm in the full feedback setting since it does not require the cost functions in the current epoch and only uses past information. Now, we will bound the gap between the per-

formance of FPWL and BPWL.

To do this, we state the following Lemma from [99].

Lemma 6. *For any $v \in \mathbb{R}^n$, the cubes $\left[0, \frac{1}{\epsilon}\right]^n$ and $\left[0, \frac{1}{\epsilon}\right]^n + v$ overlap in at least a $(1 - \epsilon|v|_1)$ fraction.*

We define the increment function $f'(\cdot)$ for an AoI cost function $f : \{1, ..., M\} \to \mathbb{R}^+$ as follows:

$$f'(i) = f(i) - f(i-1), \forall i \in 1, ..., M. \tag{3.52}$$

$f(0)$ is set to zero to have a valid definition for $i = 1$. Now, we can rewrite the Perturb$(\cdot)$ using increment functions rather than cost functions. Thus,

$$\text{Perturb}\left(f'^{(1)}, ..., f'^{(N)}\right) = \left(f'^{(1)} + \delta^{(1)}, ..., f'^{(N)} + \delta^{(N)}\right), \tag{3.53}$$

where $\delta^{(i)}(j) \sim \text{Uniform}\big([0, 1/\epsilon]\big), \forall i \in 1, ..., N$ and $\forall j \in 1, ..., M$ are i.i.d. random variables. This allows us to write the perturbation procedure as an addition of i.i.d. uniform random vectors $\delta^{(i)}$. The earlier definition had $\gamma^{(i)}$ which were not element-wise i.i.d.

Now applying Lemma 6 we observe that
$\text{Perturb}\left(\sum_{t=1}^{k-1} f_t^{(1)}, ..., \sum_{t=1}^{k-1} f_t^{(N)}\right)$ and $\text{Perturb}\left(\sum_{t=1}^{k} f_t^{(1)}, ..., \sum_{t=1}^{k} f_t^{(N)}\right)$ have the same expectation with probability

$$\geq (1 - \epsilon \sum_{i=1}^{N} |f_k'^{(i)}|_1).$$

Using linearity of expectation and cost functions, $\mathbb{E}[C_k(\pi_k^{\text{FPWL}})]$ and $\mathbb{E}[C_k(\pi_k^{\text{BPWL}})]$ are also the same with probability

$$\geq (1 - \epsilon \sum_{i=1}^{N} |f_k'^{(i)}|_1).$$

On the non-overlapping fraction we assume the worst possible cost difference between the algorithms, which can be upper bounded by D since we assume that

the AoI cost functions are upper bounded by D. Combining all of this together, we get:

$$\mathbb{E}[C_k(\pi_k^{\text{FPWL}})] \leq \mathbb{E}[C_k(\pi_k^{\text{BPWL}})] + D\epsilon \max_{f_k^{(1)},\ldots,f_k^{(N)}} \sum_{i=1}^{N} |f_k'^{(i)}|_1. \tag{3.54}$$

Observe that since $f_k^{(i)}(\cdot) \leq D$, so $|f_k'^{(i)}|_1 \leq D$, for all i. Thus,

$$\mathbb{E}[C_k(\pi_k^{\text{FPWL}})] \leq \mathbb{E}[C_k(\pi_k^{\text{BPWL}})] + \epsilon N D^2. \tag{3.55}$$

Adding the above equation for all $k \in 1, \ldots, T$:

$$\sum_{k=1}^{T} \mathbb{E}[C_k(\pi_k^{\text{FPWL}})] \leq \sum_{k=1}^{T} \mathbb{E}[C_k(\pi_k^{\text{BPWL}})] + \epsilon N D^2 T. \tag{3.56}$$

Using (3.50), we finally have regret of the FPWL algorithm:

$$\mathbb{E}\big[\text{Regret}_T(\text{FPWL})\big] \leq \epsilon N D^2 T + 2\frac{M}{\epsilon} + \alpha T. \tag{3.57}$$

Setting $\epsilon = \sqrt{\frac{2M}{ND^2T}}$, we get:

$$\mathbb{E}\big[\text{Regret}_T(\text{FPWL})\big] \leq \alpha T + 2D\sqrt{2MNT}. \tag{3.58}$$

This completes our proof.

3.5.E Proof of Lemma 5

For any given sequence of cost functions $C_1, \ldots, C_T$ that satisfy (3.16), the performance gap between the decisions π_k given by (3.17) and choosing the optimal policy in each epoch is given by:

$$\sum_{k=1}^{T} C_k(\pi_k) - \sum_{k=1}^{T} \min_{\pi} C_k(\pi) \tag{3.59}$$

We rewrite this as:

$$\sum_{k=2}^{T}\Bigg(C_k\big(\arg\min_{\pi\in\Pi} C_{k-1}(\pi)\big) - C_{k-1}\big(\arg\min_{\pi\in\Pi} C_{k-1}(\pi)\big)\Bigg)+$$
$$C_1(\pi_1) - \min_x C_T(\pi) + \sum_{k=2}^{T}\Bigg(C_k(\pi_k) - C_k\big(\arg\min_{\pi\in\Pi} C_{k-1}(\pi)\big)\Bigg) \quad (3.60)$$

Using Assumption 1 and the definition of π_k, we get:

$$C_k(\pi_k) - C_k\big(\arg\min_{\pi\in\Pi} C_{k-1}(\pi)\big) \leq \alpha, \forall k = 2, ..., T. \quad (3.61)$$

This is because

$$\arg\min_{\pi\in\Pi} C_{k-1}(\pi) = \text{Opt}(f_{k-1}^{(1)}, ..., f_{k-1}^{(N)}),$$

while $\pi_k = \text{Whittle}(f_{k-1}^{(1)}, ..., f_{k-1}^{(N)})$.

Using the defintion of V_T (3.16), the fact that $C_k(\cdot) \in [0, D]$ and the inequality (3.61), we get:

$$\sum_{k=1}^{T} C_k(\pi_k) - \sum_{k=1}^{T} \min_{\pi} C_k(\pi) \leq \alpha T + V_T + D. \quad (3.62)$$

Since the above equation is true for any sequence of cost functions that satisfy the V_T constraint, it completes the proof.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4

Computation and Communication Trade-offs

In this Chapter, we explore the joint optimization of computation and communication resources for monitoring and control tasks. We consider a multi-agent system where each agent is in charge of monitoring a time-varying phenomenon and sending information to a central base station. For instance, this setup can model a team of robots mapping a dynamic environment and sending map updates to a base station, which aggregates a global map for centralized decision-making (Figure 4-1).

The agents are capable of local processing before transmitting the acquired information. This could involve operations such as refining, denoising, or compressing the data or simply gathering more informative updates. We assume that the more time an agent spends in processing locally, the higher the quality of the generated update. However, longer processing also induces a delay in between subsequent updates. This yields a *delay-accuracy trade-off*: is it better to send outdated but high-quality updates, or to reduce the overall latency by communicating low-quality information?

We consider the realistic scenario where the total communication resources

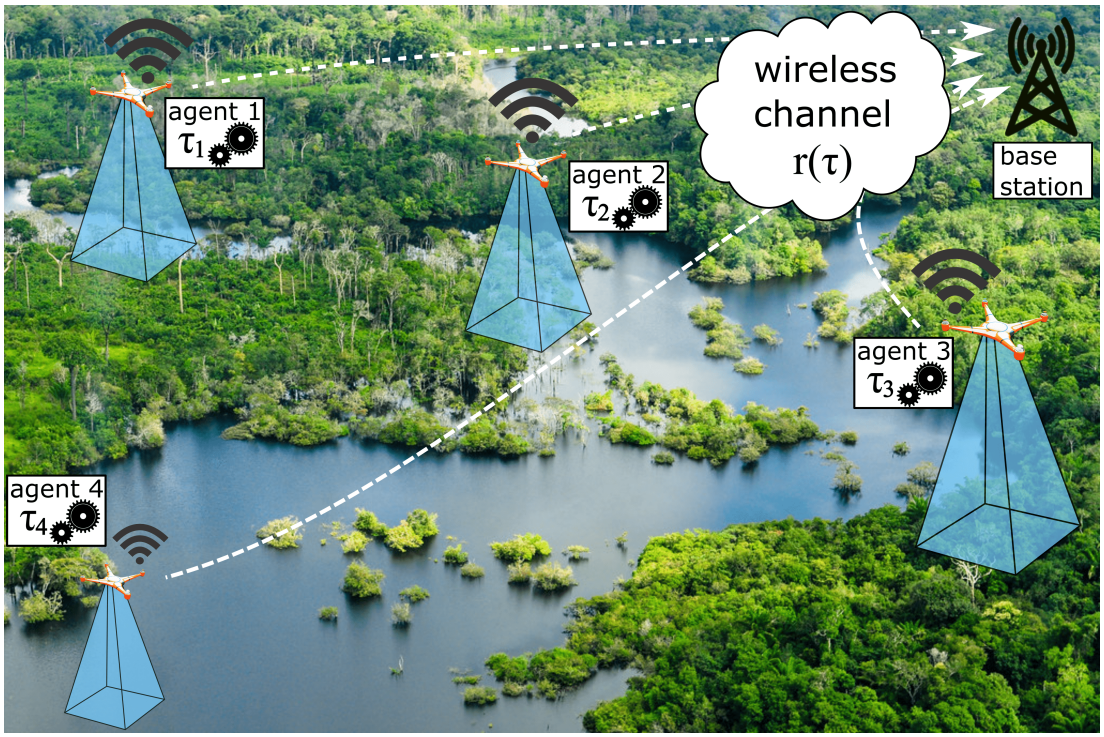


Figure 4-1: Example: four drones monitor different regions and send updates to a base station over a wireless channel. Each agent spends time τ_i processing the collected measurements before sending. A scheduling algorithm prioritizes transmissions to the base station. This paper focuses on the co-design of the processing times τ_i and the scheduling policy.

available are limited due to interference, limited bandwidth, and/or power constraints. Thus, in any given time-slot, only one of the agents is allowed to communicate with the base station. The communication constraints mean that, in addition to optimizing the local processing times, a *scheduling policy* needs to be designed to specify which agents can communicate in every time-slot.

Therefore, the goal of this chapter is to develop a general framework to determine the optimal amount of local processing at each agent in the network and design a scheduling policy to prioritize communication in order to maximize performance.

Contributions. We address the computation and communication co-design problem and develop a) a scheduling policy that ensures timely delivery of up-

dates, and b) an algorithm to determine the optimal amount of local processing at each agent. To do so, we use AoI to measure the lag in obtaining information for monitoring and control of time-critical systems. Our contribution is threefold. First, we develop a general framework to jointly optimize computation and communication for real-time monitoring and decision-making (Section 4.1). This framework extends existing work [108] by a) considering joint optimization of scheduling in addition to processing, and b) addressing a general model that goes beyond linear systems.

Second, we develop low-complexity scheduling and processing allocation schemes that perform well in practice (Sections 4.2-4.3). The co-design problem is a multi-period resource allocation problem and is hard to solve in general due to its combinatorial nature. We resolve this by considering a Lagrangian relaxation that decouples the problem into multiple single-agent problems, which can be solved effectively. To solve the scheduling problem, we generalize the Whittle index framework proposed in Chapter 2 for sources that generate updates at different rates and of different sizes.

Finally, we demonstrate the benefits of using our methods in two practical applications from robotics and autonomous systems: multi-agent occupancy grid mapping in time-varying environments and ride-sharing systems with local route optimization. Our simulations in Section 4.4 show that we can achieve performance improvements of 18 – 35% in the mapping application and 75 – 82% in the ride-sharing application with respect to baseline approaches.

4.1 Model

We consider a discrete-time setting with N agents in a networked system, where each agent is in charge of monitoring a time-varying phenomenon and sending information updates to a base station. Each agent processes the collected mea-

surements locally, before sending its updates. The i-th agent spends τ_i time slots to process a new update. We refer to this quantity as the *processing time* associated with agent i.

We assume that sensing and processing happen sequentially at each agent. Thus, agent i acquires a new sample every τ_i time slots. Further, each agent stores in a buffer the freshest processed measurement. We will assume that the processing time allocations $\tau_i, \forall i$ are constant during operation.

To communicate the acquired and processed updates, the agents use a wireless communication channel. We assume that, due to interference and bandwidth constraints, only one of the agents can transmit to the base station in any given time-slot. At every transmission opportunity, the base station polls one of the agents regarding the state of its system and receives the most recent measurement that has been processed.

Scheduling decisions are modeled as indicator variables $u_i(t)$ where $u_i(t) = 1$ if the i-th agent is scheduled at time t and zero otherwise. We assume that a transmission from the i-th agent takes $r_i(\tau_i)$ time slots, with $r_i(\cdot)$ a monotone sequence. This captures one aspect of the delay-accuracy trade-off, namely that the size of the update depends on the amount of time spent in processing it. When the agents spend local processing to collect more detailed information, *e.g.*in exploration tasks, the measurements get larger overtime and $r_i(\cdot)$ is increasing. Conversely, when the agents compress the collected data, *e.g.*extracting visual features from images, $r_i(\cdot)$ is decreasing.

To measure the freshness of the information at the base station, we use a metric called Age of Information (AoI). The AoI $A_i(t)$ measures how old the information at the base station is regarding agent i at time t. Upon receiving a new update, it drops to the age of the delivered update. Otherwise, it increases linearly. The

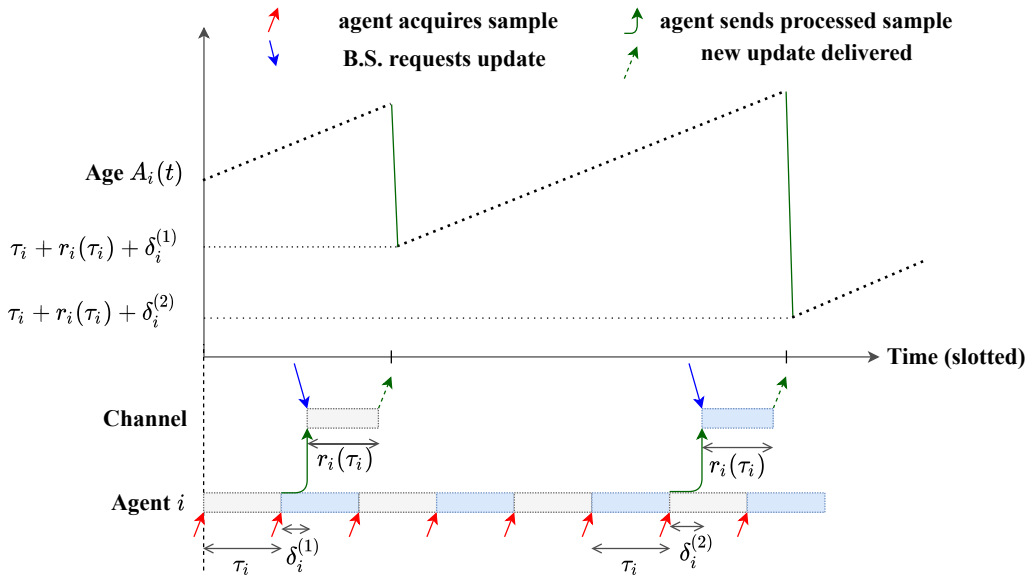


Figure 4-2: AoI evolution for agent i. The agent acquires and processes new samples every τ_i time-slots. When the base station (B.S.) requests a new update, the agent sends the most recent sample that has finished processing, taking $r_i(\tau_i)$ time-slots for transmission. The variable $\delta_i^{(k)}$ represents the waiting time in the buffer for update k. Upon a new update delivery, the AoI at the base station $A_i(t)$ drops to the age of the delivered update.

evolution is described below:

$$A_i(t+1) = \begin{cases} \tau_i + r_i(\tau_i) + \delta_i^{(k)}, & \text{if update } k \text{ is delivered,} \\ A_i(t) + 1, & \text{otherwise.} \end{cases} \tag{4.1}$$

Here $\delta_i^{(k)}$ is the *waiting time* spent by the k-th update from agent i in the buffer, *i.e.*the delay from the time the update was processed to the time it was actually transmitted. Since a new processed update is generated every τ_i time-slots, the waiting time $\delta_i^{(k)}$ ranges from 0 to $\tau_i - 1$ time-slots. Figure 4-2 depicts the AoI process for agent i. Observe that the lowest value that the AoI can drop to is $\tau_i + r_i(\tau_i)$, since every update spends time τ_i in processing and time $r_i(\tau_i)$ in communication.

The AoI evolution in (4.1) is involved since it requires analyzing waiting times that vary with each update. To simplify the analysis, while still capturing the relevant features of the AoI dynamics, we assume that the sequences $\delta_i^{(k)}$ are constant over time, *i.e.*$\delta_i^{(k)} \equiv \delta_i \, \forall k$, $\forall i \in \mathcal{V}$. Each δ_i accounts for the average waiting time accumulated by a processed measurement before it is sent by the i-th agent. We are interested in the practical setting where processing times τ_i are small, and the number of agents N is large. Thus, our assumption of constant waiting times is reasonable, since the waiting time's contribution to the overall AoI is negligible on average (being upper bounded by τ_i) as compared to the time between subsequent requests from the base station, which grows linearly with the number of agents N [15]. The smallest AoI for agent i is defined as $\Delta_i \triangleq \tau_i + r_i(\tau_i) + \delta_i$, which is the value that AoI resets to upon a new update delivery.

It has been shown in recent works [22, 24, 25, 26] that real-time monitoring error for linear dynamical systems can be seen as an increasing function of the AoI. Intuitively, fresher updates lead to higher monitoring accuracy and better control performance. Motivated by this, we assume that each agent has an as-

sociated cost function $J_i(\tau_i, A_i(t))$ that maps the processing time and the current AoI to a cost that reflects how useful the current information at the base station is for monitoring or control.

Assumption 1 (Delay-Accuracy Trade-off)**.** *The cost functions $J_i(\tau_i, A_i(t))$ are increasing with the AoI $A_i(t)$ and decreasing with the processing time τ_i. Thus, longer processing leads to more useful measurements (for a fixed age), while fresher information induces a lower cost than outdated information.*

Remark 1 (Task-related cost function). The functional form of $J_i(\tau_i, A_i(t))$ depends on the underlying dynamics of the system i and on the impact of agent processing on the quality of updates. These functions are typically estimated using domain knowledge or learned from data offline. The approach in this paper holds for any functions $J_i(\tau_i, A_i(t))$ that satisfy the above assumption. We discuss numerical examples in section 3.3.

Our goal is to design a causal scheduling policy π and find the processing times $\tau_1, ..., \tau_N$ for every agent so as to minimize the sum of the time-average costs.

Problem 1 (Computation and Computation Co-design)**.** *Given the set of agents $\mathcal{V} = \{1, \ldots, N\}$, cost functions $\{J_i(\cdot, \cdot)\}_{i \in \mathcal{V}}$, and AoI evolution* (4.1)*, find the processing times $\{\tau_i\}_{i \in \mathcal{V}}$ and the scheduling policy π that minimize the*

infinite-horizon time-averaged cost:

$$\begin{aligned} \min_{\substack{\tau_i \in \mathcal{T}_i \, \forall i \in \mathcal{V} \\ \pi \in \Pi}} \quad & \sum_{i \in \mathcal{V}} \limsup_{T \to +\infty} \mathbb{E}_\pi \left[\frac{1}{T} \sum_{t=t_0}^{T} J_i \left(\tau_i, A_i^\pi(t) \right) \right] \\ s.t. \quad & \sum_{i \in \mathcal{V}} u_i^\pi(t) \leq 1, \forall t \end{aligned} \tag{P1}$$

where Π *is the set of causal scheduling policies,* $u_i^\pi(t) = 1$ *if policy* π *schedules agent* i *at time* t *and* $u_i^\pi(t) = 0$ *otherwise.* $\mathcal{T}_i$ *is the set of admissible processing times for agent* i*, and* $A_i^\pi(t)$ *is the AoI of the* i*-th agent at time* t *under policy* π*.*

Finding the optimal processing times requires iterating over the combinatorial space $\mathcal{T}_i \times \ldots \times \mathcal{T}_N$, while finding the optimal scheduling policy requires solving a dynamic program which suffers from the curse of dimensionality.

4.2 A Lagrangian Relaxation

We now discuss a relaxation of (P1) that enables us to develop efficient algorithms. This approach is motivated by the work of Whittle [109] and its applications to network scheduling, as we saw in Chapter 2. The relaxation will be useful not only for finding a scheduling policy, but also in optimizing the processing times.

We start by considering a relaxation of (P1) where the scheduling constraint is to be satisfied on average, rather than at each time slot. The relaxed problem is

given by

$$\begin{aligned} \min_{\substack{\tau_i \in \mathcal{T}_i \, \forall i \in \mathcal{V} \\ \pi \in \Pi}} \quad & \sum_{i \in \mathcal{V}} \limsup_{T \to +\infty} \mathbb{E}_\pi \left[\frac{1}{T} \sum_{t=t_0}^{T} J_i \left(\tau_i, A_i^\pi(t) \right) \right] \\ \text{s.t.} \quad & \sum_{i \in \mathcal{V}} \limsup_{T \to +\infty} \frac{\sum_{t=t_0}^{T} u_i^\pi(t)}{T} \leq 1. \end{aligned} \tag{4.2}$$

To solve (4.2), we introduce a Lagrange multiplier $C > 0$ for the average scheduling constraint. The Lagrange optimization is given by the following equation:

$$\max_{C>0} \min_{\substack{\tau_i \in \mathcal{T}_i \, \forall i \in \mathcal{V} \\ \pi \in \Pi}} \sum_{i \in \mathcal{V}} \bar{J}_i(\tau_i, C) - C \tag{4.3}$$

$$\bar{J}_i(\tau_i, C) \triangleq \limsup_{T \to +\infty} \mathbb{E}_\pi \left[\frac{1}{T} \sum_{t=t_0}^{T} \left(J_i \left(\tau_i, A_i^\pi(t) \right) + C u_i^\pi(t) \right) \right]$$

Due to the Lagrangian relaxation, the inner minimization can be decoupled as the sum of N independent problems.

Problem 1 (Decoupled Problem i)**.** *Given a constant cost $C > 0$, find a scheduling policy $\pi_i = \{u_i(t)\}_{t \geq t_0}$ and a processing time $\tau_i \in \mathcal{T}_i$ that minimize the infinite-horizon time-averaged cost of agent i:*

$$\min_{\substack{\tau_i \in \mathcal{T}_i \\ \pi_i \in \Pi}} \limsup_{T \to +\infty} \mathbb{E}_{\pi_i} \left[\frac{1}{T} \sum_{t=t_0}^{T} \left(J_i \left(\tau_i, A_i^{\pi_i}(t) \right) + C u_i(t) \right) \right] \tag{P2}$$

In (P2), the multiplier C can be interpreted as a transmission cost: whenever $u_i(t) = 1$, agent i has to pay a cost of C for using the channel. Further, transmitting an entire update costs $C r_i(\tau_i)$, since i transmits for $r_i(\tau_i)$ time-slots.
In the next section, we look at the single-agent problem (P2) in greater detail, and

show how to solve it exactly. Since the problem involves a single agent, it is much easier to solve than the original combinatorial formulation. The solution also provides key insights in choosing both the scheduling policy and the processing times for the original problem (P1).

4.2.A Solving the Decoupled Problem

We now solve (P2) for each agent separately. First, we characterize the structure of the optimal scheduling policy π_i^* given a fixed value of τ_i. Then, we optimize over the latter.

Theorem 14. *The solution to* (P2)*, given a fixed value of τ_i, is a stationary threshold-based policy: let $\widetilde{H}_i \triangleq H_i + r_i(\tau_i)$ and suppose there exists an age H_i that satisfies*

$$J_i(\tau_i, \widetilde{H}_i - 1) \leq J_i^W(\tau_i, H_i) \leq J_i\big(\tau_i, \widetilde{H}_i\big) \tag{4.4}$$

where

$$J_i^W(\tau_i, H_i) \triangleq \frac{\sum_{h=\Delta_i}^{\widetilde{H}_i - 1} J_i(\tau_i, h) + C r_i(\tau_i)}{\widetilde{H}_i - \Delta_i}. \tag{4.5}$$

Then, an optimal scheduling policy π_i^* is to start sending an update whenever $A_i(t) \geq H_i$ and to not transmit otherwise. If no such H_i exists, the optimal policy is to never transmit. The quantity $J_i^W(\tau, H_i)$ represents the time-average cost of using a threshold policy with the AoI threshold H_i.

Proof. See Appendix 4.6.A. □

The structure of the optimal scheduling policy π_i^* according to Theorem 14 is intuitive, due to the monotonicity of the cost functions $J_i(\tau_i, \cdot)$ in the AoI. If it is optimal to transmit and pay the cost C for $r_i(\tau_i)$ time-slots at a particular AoI, it

should be also be optimal to do so when the AoI is higher, since the gain from AoI reduction would be even more. Given τ_i and C , a way to compute the optimal threshold is to start from $H_i = \Delta_i$ and increase H_i until condition (4.4) is satisfied. Let the value that this procedure terminates at be denoted by $H_i(\tau_i)$. Then, $H_i(\tau_i)$ is an optimal threshold for agent i.

Next, we look at how to compute the optimal processing time τ_i^* to solve Problem 1. To do so, given the admissible set $\mathcal{T}_i$, we find the value of $\tau_i \in \mathcal{T}_i$ that induces the lowest time-averaged cost for agent i by enumerating over the set $\mathcal{T}_i$:

$$\tau_i^* = \arg\min_{\tau_i \in \mathcal{T}_i} \tilde{J}_i^W(\tau_i). \tag{4.6}$$

where $\tilde{J}_i^W(\tau_i) \triangleq J_i^W\big(\tau_i, H_i(\tau_i)\big)$. The optimal processing times τ_i^* and policies π_i^*, with thresholds $H_i(\tau_i^*)$, computed for each decoupled problem provide an optimal solution to the inner minimization of (4.3).

4.2.B Optimizing Processing Times

Leveraging the solution of the decoupled problems found in subsection 4.2.A, we now design a procedure to optimize the processing times for the original multi-agent problem (4.8).

Given a cost $C > 0$, we can use (4.4) and (4.6) to compute the optimal processing times τ_i^* and the corresponding AoI thresholds $H_i(\tau_i^*)$ for the N decoupled problems in (4.3). Further, observe that, for the i-th decoupled problem, the optimal scheduling policy for agent i chooses to send a new update every time the AoI exceeds $H_i(\tau_i^*)$ and the AoI drops to Δ_i after each update delivery. Thus, the fraction of time that agent i occupies the channel (on average) is given by

$$f_i(\tau_i^*) = \frac{r_i(\tau_i^*)}{H_i(\tau_i^*) + r_i(\tau_i^*) - \Delta_i}. \tag{4.7}$$

The total channel utilization given the Lagrange multiplier C is $f = \sum_{i\in\mathcal{V}} f_i(\tau_i^*)$. From (4.2), f must lie in the interval $[0,1]$ to represent a feasible allocation of computation and communication resources. If not, then more than one agent is transmitting in every time-slot *on average*, which is not possible given the (relaxed) interference constraint.

This suggests a natural way to optimize over both the Lagrange cost C and the processing times τ_i, which is presented in Algorithm 5. In particular, we optimize the processing times τ_i by using (4.4) and (4.6) (line 4 in Algorithm 5), and update C via a dual-ascent scheme (lines 6–7) using the average channel utilization f_{curr}.

Algorithm 5: Optimizing Processing Times

Input : Costs $J_i^W(\cdot)$, set of admissible processing times $\mathcal{T}_i$ for each agent $i \in \mathcal{V}$, stepsize $\alpha > 0$.
Output: Locally optimal processing times $\{\tau_i^*\}_{i\in\mathcal{V}}$.

1 Set $C \leftarrow C_0$
2 **while** $f_{curr} > 1$ **do**
3 **for** *sensor* $i \in \mathcal{V}$ **do**
4 $\tau_i^* \leftarrow \arg\min_{\tau_i\in\mathcal{T}_i} \tilde{J}_i^W(\tau_i)$ ▷ optimization (4.6)
5 **end**
6 $f_{curr} \leftarrow \sum_{i\in\mathcal{V}} f_i(\tau_i^*)$
7 $C \leftarrow C + \alpha(f_{curr} - 1)$
8 **end**
9 Return $\{\tau_i^*\}_{i\in\mathcal{V}}$

Intuitively, the algorithm keeps increasing the virtual communication cost (quantified by the Lagrange multiplier C) until the processing times computed in line 4 become compatible with the scheduling constraint. The decoupling reduces the complexity of finding the optimal processing times from combinatorial $O\left(\prod_{i\in\mathcal{V}} |\mathcal{T}_i|\right)$ to linear search $O\left(\sum_{i\in\mathcal{V}} |\mathcal{T}_i|\right)$.

4.3 Whittle Index Scheduling

In the previous section, we established a threshold structure for the optimal scheduling policy of the relaxed problem (4.2), where each agent transmits when its AoI exceeds $H_i(\tau_i^*)$. Next, we exploit this threshold structure to design an efficient scheduling policy for the original optimization problem (1). Given the processing times τ_i^* computed via Algorithm 5, we need to solve:

$$\begin{aligned} \min_{\pi\in\Pi} \quad & \sum_{i\in\mathcal{V}} \limsup_{T\to+\infty} \mathbb{E}_\pi \left[\frac{1}{T} \sum_{t=t_0}^{T} J_i\left(\tau_i^*, A_i^\pi(t)\right) \right] \\ \text{s.t.} \quad & \sum_{i\in\mathcal{V}} u_i(t) \leq 1, \forall t. \end{aligned} \tag{4.8}$$

We considered minimizing the time-average of increasing functions of AoI in Chapter 2. However, unlike the setting in Chapter 2, our agents generate updates at different rates (every τ_i time-slots for agent i) and induce different communication delays ($r_i(\tau_i)$ time-slots). We now generalize the Whittle index approach for our setting.

Recall that the Whittle index approach consists of four steps: 1) converting the problem into an equivalent restless multi-armed bandit (RMAB) formulation, 2) decoupling the problem via a Lagrange relaxation, 3) establishing a structural property called *indexability* for the decoupled problems, and 4) using this structure to formulate a Whittle index policy for the original scheduling problem. We go through these steps below.

Step 1. We first need to establish (4.8) can be equivalently formulated as a restless multi-armed bandit problem.

Step 2. As we observed in section 4.2, the original scheduling problem can be split into N decoupled problems of the form (P2) via a Lagrange relaxation. Further, through Theorem 14, we know that the optimal scheduling policy for each decoupled problem has a threshold structure, *i.e.*agent i should transmit

only if its associated AoI $A_i(t)$ exceeds the threshold $H_i(\tau_i^*)$.

Step 3. Whittle showed in [109] that when there is added structure in the form of a property called *indexability* for the decoupled problems, then the RMAB admits a low-complexity solution called the Whittle index, that is known to be near optimal [110]. The *indexability* property for the i-th decoupled problem requires that, as the transmission cost C increases from 0 to ∞, the set of AoI values for which it is optimal for agent i to transmit must decrease monotonically from the entire set (all ages $A_i(t) \geq \Delta_i$) to the empty set (never transmit). In other words, the optimal threshold $H_i(\tau_i^*)$ should increase as the transmission cost C increases. Next, we use Theorem 14 and the monotonicity of the cost functions $J_i(\tau_i^*, \cdot)$ to establish that the decoupled problems are indeed indexable.

Lemma 7. *The indexability property holds for the decoupled problems* (P2)*, given an allocation of processing times* τ_i*.*

Step 4. Having established indexability for the decoupled problem (P2), we can derive a functional form for the Whittle index which solves the scheduling for the original optimization problem (4.8).

Definition For the i-th decoupled problem, the Whittle index $W_i(H)$ is defined as the minimum cost C that makes both scheduling decisions (transmit, not transmit) equally preferable at AoI H. Let $\widetilde{H} \triangleq H + r_i(\tau_i)$. The expression for $W_i(H)$, given a processing time τ_i, is:

$$W_i(H) \triangleq \frac{\left(\widetilde{H} - \Delta_i\right) J\left(\tau_i, \widetilde{H}\right) - \sum_{k=\Delta_i}^{\widetilde{H}-1} J(\tau_i, k)}{r_i(\tau_i)}. \tag{4.9}$$

Using (4.9), we can now design the Whittle index policy to solve (4.8). Whenever the channel is unoccupied, the agent with the most critical update should be asked for an update. This leads to the scheduling policy presented in Algorithm 6. The Whittle index policy chooses the agent with the highest index (line 5), since it represents the minimum cost each agent would be willing to pay to transmit at the current time-slot. When the channel is occupied, no other transmission is allowed (line 9). The variable z keeps track of ongoing communication and drops to zero when a new transmission can be scheduled.

Algorithm 6: Whittle Index Scheduling

Input : Processing time τ_i, communication delay $r_i(\cdot)$, and cost $J_i(\cdot,\cdot)$ for each agent $i \in \mathcal{V}$, time horizon T.

1 $t = t_0$, $z = 0$
2 **while** $t \leq T$ **do**
3 **if** $z = 0$ ▷ *schedule new transmission at time* t
4 **then**
5 $i^* \leftarrow \arg\max_{i \in \mathcal{V}} W_i(A_i(t))$ ▷ trigger agent i^*
6 $z \leftarrow r_{i^*}(\tau_{i^*}) - 1$
7 **end**
8 **else**
9 $z \leftarrow z - 1$ ▷ continue ongoing transmission
10 **end**
11 **end**

The Whittle index is known to be asymptotically optimal as $N \to \infty$, if a fluid limit condition is satisfied [110, 34]. These results, suggest that the Whittle index is a very good low-complexity heuristic for scheduling in real-time monitoring and control applications. In the following section, we demonstrate our co-design algorithms in two applications: multi-agent occupancy grid mapping in time-varying environments and ride sharing in autonomous vehicle networks. The results show that we can achieve performance improvements of 18 – 35% for grid mapping and 75 – 82% for ride-sharing compared to baseline approaches.

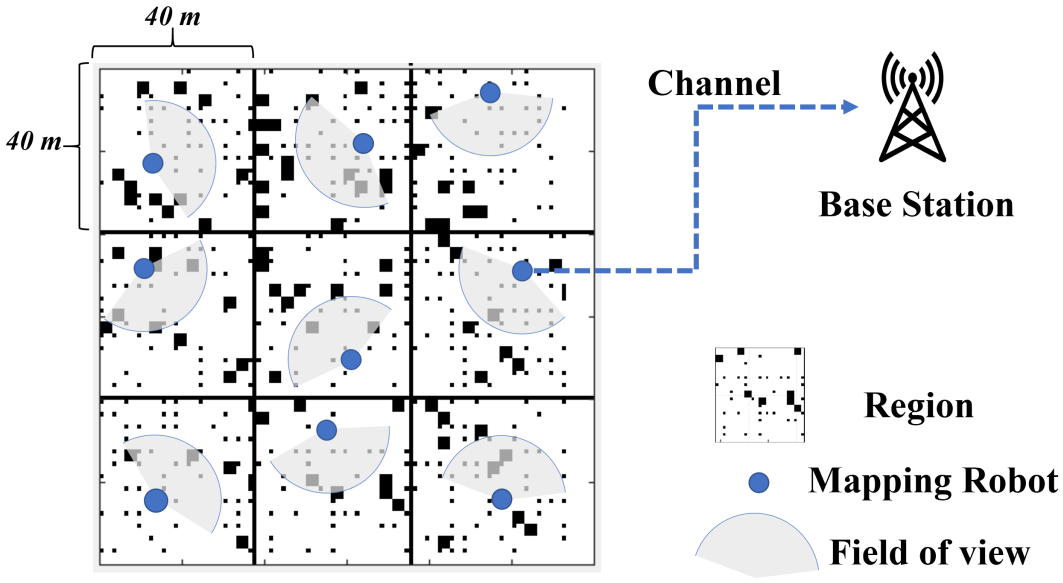


Figure 4-3: **Multi-agent mapping over 9 regions:** each agent monitors and builds a local grid map of a region, and sends map updates to a base station. The occupancy in the regions is time-varying. A scheduling policy specifies how to share the communication channel among the agents. Processing times specify how much time each agent spends in generating new map updates.

4.4 Applications

We demonstrate our co-design algorithms in two applications: multi-agent occupancy grid mapping in time-varying environments (Section 4.4.A), and ride sharing in autonomous vehicle networks (Section 4.4.B). The results show that we can achieve performance improvements of 18 – 35% for grid mapping and 75 – 82% for ride-sharing compared to baseline approaches. We also provide a video briefly summarizing and visualizing our simulation results [111].

4.4.A Multi-agent Mapping of Time-Varying Environments

Setup We co-design computation and communication for a multi-agent mapping problem. We assume there are N separate regions each of which is being mapped by an agent. The agents send updates —in the form of occupancy

grid maps of their surroundings— to a base station over a single communication channel, where the local maps are aggregated into a global map for centralized monitoring (Figure 4-3).

In our tests, each region is 40m $\times$ 40m in size and is represented by an occupancy grid map with 1m $\times$ 1m cells. The state of each cell can be either *occupied* (1) or *unoccupied* (0). We consider a dynamic environment where the state of each cell within region i evolves according to a Markov chain, with cells remaining in their original state with probability $1 - p_i$ and switching from occupied to unoccupied and vice-versa with probability p_i. This is a common model for grid mapping in dynamic environments in the robotics community [112, 113].

Each agent is equipped with a range-bearing sensor (*e.g.*lidar), with a fixed maximum scanning distance (25m) and angular range $[-\pi/2, \pi/2]$. The agents move around the regions randomly, taking scans of the area round them. Scanning an entire region takes an agent multiple time-slots. We use the Navigation toolbox in MATLAB to create sensors such that the resolution of the readings $\theta_{\min}$ improves with the processing time. We set $\theta_{\min} = 0.5/\tau$. We also set the noise variance in angle and distance measurements to be inversely proportional to τ. These settings capture the delay-accuracy trade-off. We further set the update communication times to increase linearly with the amount of processing, *i.e.*$r(\tau) = 5 + \lceil \tau/2 \rceil$.

The base station maintains an estimate of the current map for each region based on the most recent update it received and the Markov transition probabilities $\{p_i\}_{i \in \mathcal{V}}$ associated with each region. As is common in mapping literature [114, 115], we measure uncertainty at the base station in terms of entropy of the current estimated occupancy grid map for each region and set the cost functions $J_i(\cdot, \cdot)$ to be the entropy of region i. In Section 4.6.D, we show that the entropy cost increases monotonically with the AoI of a region and satisfies the assumptions of our framework. It drops to a lower value if more time was spent in process-

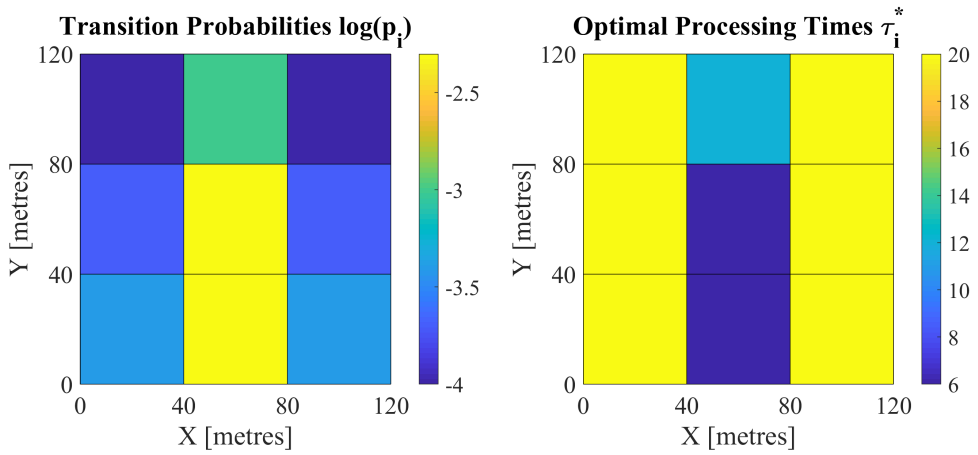


Figure 4-4: Transition probabilities and optimal processing time allocations plotted for each region. The probabilities are plotted on a logarithmic scale while the processing times are plotted in number of time-slots.

ing, since the base station is more certain about the quality of the received update. Our goal is to minimize the time-average of the entropies summed across each region through the joint optimization of processing times and the scheduling policy.

Results. Figure 4-4 shows an example of transition probabilities p_i (for each of the 9 regions) and the corresponding optimal processing times τ_i^* found using Algorithm 5. We observe that for regions that change quickly (*i.e.*have large value of p_i), the corresponding processing time allocated is smaller. This is because there is not much benefit to spending large amounts of time generating high quality updates if they become outdated very quickly. Conversely, for slowly changing regions (with low values of p_i), Algorithm 5 assigns much longer processing times. In this case, high quality useful updates can be created by taking longer time since the regions don't change quickly.

Further, we compare the performance of various scheduling algorithms in Figure 4-5. We consider the setting where the processing times τ_i are fixed to be the same parameter τ for every region (uniform processing allocation). We then plot the performance of three scheduling algorithms –a uniform stationary random-

ized policy, a round-robin policy, and the proposed Whittle index-based policy– for different values of τ. We also plot the performance of the Whittle index policy and the stationary randomized policy under the optimized processing times, computed using Algorithm 5, shown via dotted lines in Figure 4-5. We observe that Algorithm 5 can find processing times that perform well in practice. We also observe that the Whittle index policy outperforms the two "traditional" classes of scheduling policies for every value of the parameter τ.

Overall, choosing the processing times using Algorithm 5 and using the Whittle schedule from Algorithm 6 together leads to a performance improvement of 28 – 35% over the baseline versions of randomized policies. Similarly, our proposed approach leads to a performance improvement of 17 – 28% over the baseline versions of round-robin policies.

4.4.B Smart Ride Sharing Control in Vehicle Networks

Setup. We consider the scenario in which a ride-sharing taxi fleet serves a city coordinated by a central scheduler, which receives riding requests and assigns them to the drivers. Assigned requests are enqueued into a FIFO-like queue for each driver. In particular, a rider is matched to the driver whose predicted route has the shortest distance to the pick-up location.

In our setup, routes are calculated locally by drivers and transmitted on demand to the scheduler, which uses this information to match future requests. Such distributed processing for route optimization is different from current architectures, which are usually centralized. However, it allows for much greater scalability and is envisioned as a key component in increasing efficiency and scale of future ride-sharing systems [116, 117, 118].

Given communication constraints, only one driver can transmit at a time. Drivers update their route periodically to embed real-time road conditions and remove served requests from the queue. Routes are calculated via the Travelling

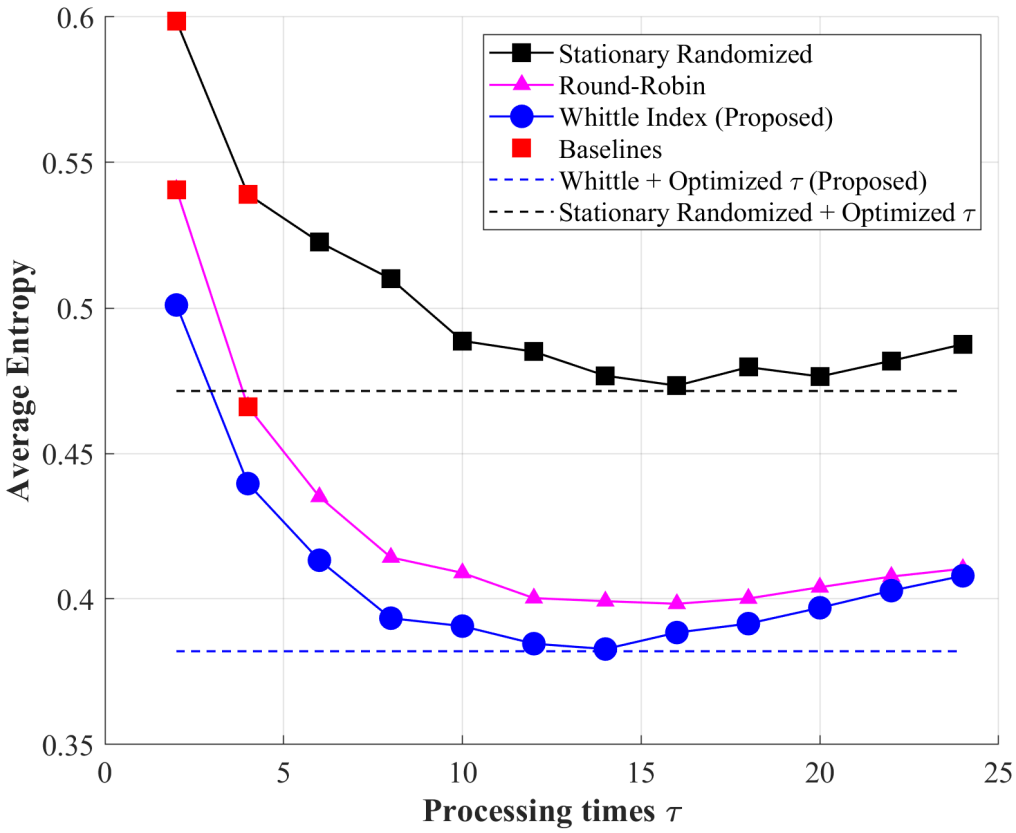


Figure 4-5: Performance of different scheduling policies vs. processing times τ. Solid lines represent performance of different classes of scheduling policies as the processing time τ varies. The dotted lines represent the scheduling performance with processing times computed using Algorithm 5.

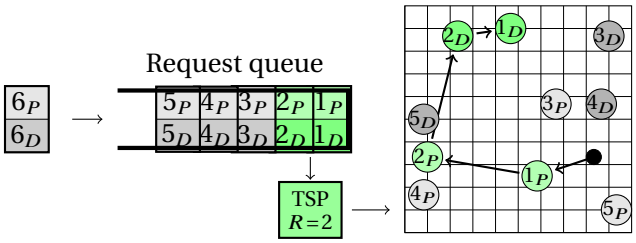


Figure 4-6: Drivers calculate their route by processing the oldest R requests (green queue portion). The TSP solver starts from the current driver location and involves pick ups (P) and drop offs (D) of the processed requests.

Figure 4-7: Left: long processing may cause large gaps between new routes calculated by the driver (solid gray) and the outdated ones stored at the scheduler (dashed gray), yielding bad matches (red dots). Right: with short processing, the matched requests are close to the actual routes (green dots).

Salesman Problem (TSP) involving the first R pick-up and drop-off locations in the request queue (Figure 4-6). Processing many requests ensures more efficient paths for enqueued riders, thus shortening their *travel time* from pick up to drop off. Conversely, the complexity of the TSP (*i.e.*its processing time) increases with the amount of processed requests R. As a consequence, the information collected by the scheduler is usually older, inducing larger gaps with the actual route followed by the driver (Figure 4-7). This leads to worse driver-request matching and increases the *waiting time* experienced by riders before they are actually picked up. Since the overall Quality of Service (QoS) is measured through the *service time*, given by the sum of travel and waiting times of riders, the drivers face a trade-off: processing many requests shortens the travels, while processing few reduces the waiting time.

In our tests, we model the city as a 200-node graph where each driver travels

one edge per time slot. Requests are randomly generated according to a Poisson process of unit intensity and assigned immediately to the matching driver by the scheduler. Each request contributes one time slot to the processing time of the TSP (*e.g.*$\tau = 2$ corresponds to processing two requests) and we set $r(\tau) = \tau$ (longer processing yields longer routes to transmit). To exploit the advantage of the Whittle index, we simulate an heterogeneous fleet with five "myopic" drivers, which can only process the oldest request ($\tau_m = 1$), and five "smart" drivers whose processing can be designed: in particular, we assign the same processing time τ_s to all such "smart" drivers. The cost of each driver, given by its *average service time* (AST), is modeled as

$$J_i\left(\tau_i, A_i(t)\right) = P_i(\tau_i) + A_i(t) \tag{4.10}$$

where the estimated contribution of the local processing (TSPs)

$$P_i(\tau_i) \doteq \left(2 + 2\mathrm{e}^{-0.2\tau_i}\right) \hat{q}_i(t) \tag{4.11}$$

was fitted from simulations with an initial queue and no assignments. Because the number of enqueued requests affects the AST but cannot be computed offline, we modeled $P_i(\tau_i)$ as linear with the queue length. The scheduler approximates the queue length at time t with the latest received value $\hat{q}_i(t)$. The dependence on $A_i(t)$ is hard to assess and we let it linear.[1]

Results. We compute statistics over 1000 Monte Carlo runs. Figure 4-8 shows the AST with 10000 requests assigned during the simulation for $\tau_s \in \{1, ..., 7\}$. The circles refer to the performance obtained with the Whittle index policy, while the squares to Stationary Randomized which is used as a benchmark. Combining Whittle index-based scheduling with processing optimization (green circle)

[1]Other cost functions decreasing with τ_i and increasing with $A_i(t)$ also yield good performance, suggesting that our approach is indeed robust.

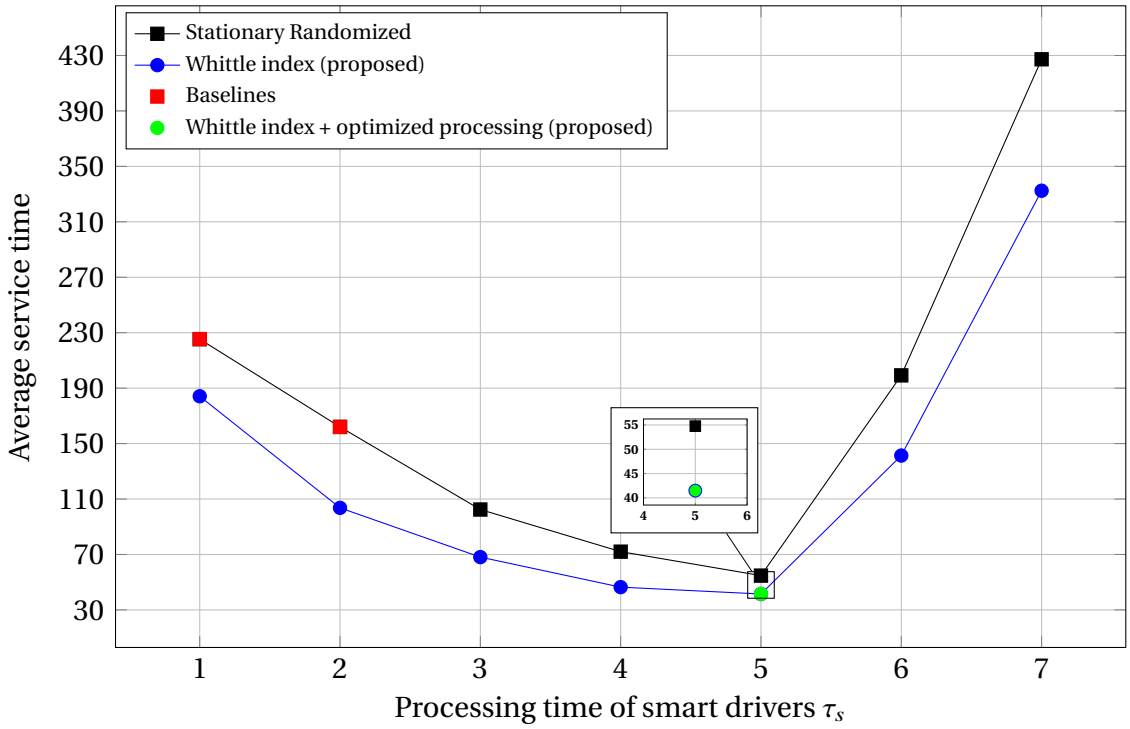


Figure 4-8: Average service time with varying processing time τ_s.

yields a striking improvement of the QoS (AST = 41) compared to the Stationary Randomized with standards policies (red squares) such as FIFO request service ($\tau_s = 1$, AST = 225), or back-to-back trips [119] ($\tau_s = 2$, AST = 165). In particular, the minimum at $\tau_s^* = 5$ indicates that it is optimal to process the five oldest requests in the queue. Also, the Whittle index outperforms Stationary Randomized for all values of the processing time, with a decrease at the optimum of 25%.

4.5 Summary

In this chapter, we developed a novel framework for computation and communication co-design for real-time multi-agent monitoring and control. We designed efficient algorithms that jointly allocate the processing time for each agent and schedule the available network communication resources. Through simulations,

we further demonstrated that the proposed approach works well for two different applications: multi-agent occupancy grid mapping in time-varying environments and distributed ride sharing in autonomous vehicle networks.

Possible directions of future work involve extending the theoretical framework to consider more complex and realistic cost functions that are coupled across multiple agents, time-varying or unknown, requiring learning-based approaches. We consider a simple model for coupling/correlation in Chapter 7 and time-varying cost functions in Chapter 3.

4.6 Appendix

4.6.A Proof of Theorem 14

We first establish that the decoupled (P2) is equivalent to a Markov decision process (MDP). We then solve the MDP using dynamic programming. Since the analysis looks similar for each of the N decoupled problems, we drop the subscript i and solve the problem for a generic agent.

The state of the MDP describing (P2) consists of two non-negative integers $\big(A(t), z(t)\big)$. $A(t)$ denotes the AoI of the agent at time t while $z(t)$ denotes how much time is left in the ongoing transmission from this agent. When the agent is not transmitting, $z(t)$ is set to be 0.

The variable $u(t)$ is an indicator variable that denotes the action of the agent: whether it is transmitting in time-slot t or not. Its value is chosen from the action set $\{0, 1\}$, 0 meaning the agent is at rest and 1 meaning an ongoing transmission. When $z(t) > 0$, that means a transmission is ongoing and $u(t)$ can only be set to 1. This ensures that an entire update must be finished by the agent before making the next scheduling decision. Whenever $z(t) = 0$, the scheduler can choose $u(t)$ to be either 0 or 1, indicating the beginning of a new transmission.

The MDP evolution can be split into 2 cases. When the agent is not transmitting ($u(t) = 0$), AoI increases by 1 and $z(t)$ remains at 0.

$$\big(A(t+1), z(t+1)\big)_{u=0} = (A(t)+1, 0). \tag{4.12}$$

When the agent is transmitting ($u(t) = 1$), the AoI drops when a new update completes delivery. Otherwise, it keeps increasing by 1. The variable $z(t)$ is set to $r(\tau) - 1$ at the beginning of a new transmission to indicate the time left in completing it. It decreases by 1 in every time-slot thereon, until the transmission completes and z becomes 0. Thus, the state evolution is given by:

$$\big(z(t+1)\big)_{u=1} = \begin{cases} r(\tau) - 1, & \text{if } z(t) = 0 \\ z(t) - 1, & \text{otherwise.} \end{cases} \tag{4.13}$$

$$\big(A(t+1)\big)_{u=1} = \begin{cases} \Delta, & \text{if } z(t+1) = 0. \\ A(t) + 1, & \text{otherwise.} \end{cases} \tag{4.14}$$

Now that we have specified the state space, the action space and the evolution equations; we also need to specify a cost function. We assume that in each time-slot the scheduler pays a cost of the form $Cu(t) + J(\tau, A(t))$. This maps the current state and action to a cost, where C acts like a transmission charge and $J(\tau, A(t))$ is an increasing function of the AoI, given a fixed value of τ.

Note that the decision process we have set up above is Markov since the state evolution depends only on the states and the actions taken in the previous time-slot. We wouldn't have been able to make this conclusion without assuming a fixed value of the waiting times δ_i, since that would have required us to maintain history per update.

Next, we aim to minimize the infinite horizon time-average cost for this MDP

using dynamic programming. We follow the standard approach by first setting up the Bellman recursions. The case when $r(\tau) = 1$ is a direct application of Theorem 1 in [1], but with an adjusted minimum AoI value. For the discussion that follows, we assume the more interesting case of $r(\tau) > 1$.

We start from a state where the AoI $A(t) = h$ and there is no ongoing transmission ($z = 0$), so a scheduling decision needs to be made. We denote the differential cost-to-go function by $S(h,z)$ and the time-average cost by λ. Then, the Bellman equation is given by:

$$S(h,0) = J(\tau,h) + \min_{u\in\{0,1\}} \Big\{ S(h+1,0), C + S\big(h+1, r(\tau)-1\big) \Big\} - \lambda. \quad (4.15)$$

Similarly, we write down the Bellman equation when there is an ongoing transmission. In this case, no scheduling decision needs to be made. When $z > 1$, the AoI keeps increasing and the Bellman equation is given by:

$$S(h,z) = J(\tau,h) + C + S\big(h+1, z-1\big) - \lambda. \quad (4.16)$$

When $z = 1$, the AoI drops in the next time-slot and the Bellman recursion is given by:

$$S(h,1) = J(\tau,h) + C + S\big(\Delta, 0\big) - \lambda. \quad (4.17)$$

Using (4.16), we expand the term $S\big(h+1, r(\tau)-1\big)$:

$$S\big(h+1, r(\tau)-1\big) = J(\tau,h+1) + C + S\big(h+2, r(\tau)-2\big) - \lambda. \quad (4.18)$$

Applying (4.16) recursively to the right-hand side till we reach $S\big(\Delta,0\big)$, we get:

$$S\big(h+1, r(\tau)-1\big) = \sum_{k=1}^{r(\tau)-1} J(\tau,h+k) + C\big(r(\tau)-1\big) + S\big(\Delta,0\big) - \lambda\big(r(\tau)-1\big). \quad (4.19)$$

Replacing $S\big(h+1, r(\tau)-1\big)$ in (4.15) with (4.19), we get:

$$S(h,0) = J(\tau,h) + \min_{u\in\{0,1\}} \Bigg\{ S(h+1,0), \\ Cr(\tau) + S\big(\Delta,0\big) - \lambda\big(r(\tau)-1\big) + \sum_{k=1}^{r(\tau)-1} J(\tau,h+k) \Bigg\} - \lambda. \quad (4.20)$$

Note that now we can simplify the differential cost-to-go function to depend on the AoI only. Let $S'(h) \triangleq S(h,0)$. Then, we get the simplified Bellman equation for our setting:

$$S'(h) = J(\tau,h) + \min_{u\in\{0,1\}} \Bigg\{ S'(h+1), Cr(\tau) + S'\big(\Delta\big) \\ - \lambda\big(r(\tau)-1\big) + \sum_{k=1}^{r(\tau)-1} J(\tau,h+k) \Bigg\} - \lambda. \quad (4.21)$$

Without loss of generality, we can set $S'(\Delta) = 0$, since $S'(\cdot)$ is a differential cost-to-go function.

Part 1. We consider the case when there exists a threshold H that satisfies the condition (4.4).

We start by looking at a policy with an arbitrary transmission threshold H, i.e. transmit if and only if the AoI $h \geq H$. We will show that if H satisfies (4.4) then this policy's differential cost-to-go function satisfies the optimal Bellman recursion (4.21).

To do so, we first compute the differential cost-to-go function for this policy. For all $h \geq H$, we set $u = 1$ in (4.21) to get:

$$\begin{aligned} S'(h) &= J(\tau,h) + Cr(\tau) + \sum_{k=1}^{r(\tau)-1} \big(J(h+k) - \lambda\big) - \lambda \\ &= (C-\lambda)r(\tau) + \sum_{k=0}^{r(\tau)-1} J(\tau,h+k) \end{aligned} \quad (4.22)$$

For $h = H - 1$ we again use (4.21) and set $u = 0$ to get:

$$\begin{aligned} S'(H-1) &= J(\tau, H-1) + S'(H) - \lambda \\ &= \sum_{k=-1}^{r(\tau)-1} J(\tau, H+k) - \lambda + (C-\lambda) r(\tau) \end{aligned} \tag{4.23}$$

where the second equality follows by expanding $S'(H)$ using (4.22). Repeating this process j times gives us:

$$S'(H-j) = \sum_{k=H-j}^{H+r(\tau)-1} J(\tau, k) - j\lambda + (C-\lambda) r(\tau) \tag{4.24}$$

Setting $H - j = \Delta$ in the equation above, we obtain the following equality:

$$\sum_{k=\Delta}^{H+r(\tau)-1} J(\tau, k) - (H-\Delta)\lambda + C r(\tau) - \lambda r(\tau) = 0. \tag{4.25}$$

Using this, we can compute λ:

$$\lambda = \frac{\sum_{k=\Delta}^{H+r(\tau)-1} J(\tau, k) + C r(\tau)}{H + r(\tau) - \Delta}. \tag{4.26}$$

For this threshold policy to be optimal, it has to satisfy the Bellman equation (4.21) such that the minimization procedure over action u computed for each value of AoI h matches the threshold structure.

Thus, for $h = H - 1$, the optimal decision must be to not transmit, i.e.

$$S'(H) \leq C r(\tau) + \sum_{k=1}^{r(\tau)-1} \Big(J(\tau, H-1+k) - \lambda \Big) \tag{4.27}$$

Plugging in the expression of $S'(H)$ using (4.22), we get:

$$(C - \lambda)r(\tau) + \sum_{k=0}^{r(\tau)-1} J(\tau, H + k) \leq Cr(\tau) + \sum_{k=1}^{r(\tau)-1} \Big(J(\tau, H - 1 + k) - \lambda \Big) \quad (4.28)$$

Simplifying the above yields:

$$J(\tau, H + r(\tau) - 1) \leq \lambda \quad (4.29)$$

Similarly, for $h = H - 2$, we get:

$$(C - \lambda)r(\tau) - \lambda + \sum_{k=-1}^{r(\tau)-1} J(\tau, H + k) \leq Cr(\tau) + \sum_{k=1}^{r(\tau)-1} \Big(J(\tau, H - 2 + k) - \lambda \Big). \quad (4.30)$$

Simplifying, we get:

$$J(\tau, H + r(\tau) - 1) + J(\tau, H + r(\tau) - 2) \leq 2\lambda. \quad (4.31)$$

Repeating the above procedure for any $h < H$, we get:

$$\sum_{k=1}^{j} J(\tau, H + r(\tau) - k) \leq j\lambda. \quad (4.32)$$

Observe that due to the monotonicity of the cost function $J(\tau, \cdot)$, the most restrictive of these conditions is (4.29), since $J(\tau, H + r(\tau) - 1) \leq \lambda$ implies $J(\tau, H + r(\tau) - k) \leq \lambda, \forall k > 1$ as well. Thus, for it to be optimal to not transmit at any AoI values below the threshold H, it is *sufficient* for the following to hold:

$$J(\tau, H + r(\tau) - 1) \leq \lambda \quad (4.33)$$

For AoI $h = H$, we instead require that the optimal choice be to transmit, i.e.

$u = 1$. Thus, the following must hold:

$$Cr(\tau) + \sum_{k=1}^{r(\tau)-1} (J(\tau, H+k) - \lambda) \leq S'(H+1) \tag{4.34}$$

Using (4.22) to expand $S(H+1)$, we get:

$$Cr(\tau) + \sum_{k=1}^{r(\tau)-1} (J(\tau, H+k) - \lambda) \leq (C-\lambda)r(\tau) + \sum_{k=0}^{r(\tau)-1} J(\tau, H+1+k). \tag{4.35}$$

Simplifying the above yields

$$\lambda \leq J(\tau, H + r(\tau)). \tag{4.36}$$

Similarly, for $h = H+1$, we require the optimal decision to be transmit and get:

$$Cr(\tau) + \sum_{k=1}^{r(\tau)-1} (J(H+1+k) - \lambda) \leq S'(H+2) = (C-\lambda)r(\tau) + \sum_{k=0}^{r(\tau)-1} J(\tau, H+2+k). \tag{4.37}$$

Simplifying the above yields

$$\lambda \leq J(\tau, H + r(\tau) + 1). \tag{4.38}$$

Repeating the above procedure for any value of $h \geq H$, we obtain similar inequalities:

$$\lambda \leq J(\tau, h + r(\tau)), \forall h \geq H. \tag{4.39}$$

Clearly, the most restrictive of these upper bounds is $\lambda \leq J(H + r(\tau))$. Thus, for it to be optimal to transmit at all AoI values $\geq H$, it is sufficient for the following to

hold:

$$\lambda \leq J(\tau, H + r(\tau)). \tag{4.40}$$

The two conditions (4.33) and (4.40) together imply that if there exists a threshold H that satisfies (4.41), then an optimal policy is to transmit only when the AoI is $\geq H$.

$$J\big(\tau, H + r(\tau) - 1\big) \leq \lambda \leq J\big(\tau, H + r(\tau)\big). \tag{4.41}$$

Observe that this is identical to the optimal threshold condition (4.4) presented in Theorem 14. This completes one part of the proof.

Part 2. It still remains to be shown that in case no such threshold can be found, then the optimal policy is to never transmit. For ease of notation, we denote $h + r(\tau)$ as $\widetilde{h}$. Consider the function $V : \mathbb{Z}^+ \to \mathbb{R}$, for all AoI values $h \geq \Delta - r(\tau) + 1$, given by:

$$V(h) \triangleq (\widetilde{h} - \Delta) J\big(\tau, \widetilde{h} - 1\big) - \sum_{k=\Delta}^{\widetilde{h}-1} J(\tau, k), \forall h. \tag{4.42}$$

Observe that for all values of $h \geq \Delta - r(\tau) + 1$, we have $V(h+1) - V(h) = (\widetilde{h} + 1 - \Delta)(J(\tau, \widetilde{h} + 1) - J(\tau, \widetilde{h})) \geq 0$. Thus, $V(\cdot)$ is an increasing function. Further, $V(\Delta - r(\tau) + 1) = J(\tau, \Delta) - J(\tau, \Delta) = 0$. Thus, $V(h)$ is a non-negative function for all values of AoI $\geq \Delta - r(\tau) + 1$.

Using the function $V(\cdot)$ and the expression for λ (4.26), we can rewrite the condition (4.41) as follows:

$$V(H) \leq Cr(\tau) \leq V(H + 1). \tag{4.43}$$

Suppose there exists some h such that $Cr(\tau) \leq V(h+1)$. Then, clearly (4.43) has a solution at $H = h$, since $V(\cdot)$ is a non-decreasing function. Since we are interested in the case when (4.43) does not have a solution, we can safely assume

$Cr(\tau) > V(h), \forall h$.

Since $V(h) \in [0, Cr(\tau)], \forall h \geq \Delta$, so $V(h)$ converges to a finite value (bounded sequences always converge). The relation $V(h+1) - V(h) = (\widetilde{h} + 1 - \Delta)(J(\tau, \widetilde{h} + 1) - J(\tau, \widetilde{h})) \geq 0$ also ensures that the function $J(\tau, \cdot)$ is bounded. This is because $J(\tau, \widetilde{h})$ is a non-decreasing sequence and has smaller increments than $V(h)$ for each value of h. So, we can set $\lambda = \lim_{h \to \infty} J(\tau, h)$ and λ is well-defined.

We also set the differential cost-to-go function to be:

$$S'(h) = \sum_{k=h}^{\infty} (J(\tau, k) - \lambda) + Cr(\tau), \forall h \geq \Delta. \tag{4.44}$$

Clearly, $S'(h)$ satisfies the following Bellman recurrence for never transmitting, i.e.

$$S'(h) = J(\tau, h) + S'(h+1) - \lambda, \forall h \geq \Delta. \tag{4.45}$$

By the monotonicity of $J(\tau, \cdot)$, we know that $J(\tau, h) \leq \lambda, \forall h \geq \Delta$. This, together with (4.44) implies

$$S'(h+1) \leq Cr(\tau) + \sum_{k=1}^{r(\tau)-1} (J(\tau, h+k) - \lambda), \forall h \geq \Delta. \tag{4.46}$$

The condition above implies that the minimization procedure to choose $u \in \{0, 1\}$ will always select 0, i.e. never transmit. Thus, our choice of λ and $S'(h)$ satisfies the Bellman equations and is optimal. This completes the proof of Theorem 14.

4.6.B Restless Multi-Armed Bandit Formulation

We establish that the scheduling optimization described by (4.8) is equivalent to a restless multi-armed bandit problem (RMAB). A restless multi-armed bandit problem [109] consists of N "arms". Each arm is a Markov decision process (MDP) with two actions (activate, rest). There are two transition matrices per arm, one

describing how the states evolve when the arm is active and one describing how the states evolve when the arm is at rest. Each arm has a cost function mapping states to costs. In the classic RMAB formulation, only one arm can be activated in each time-slot, similar to our scheduling constraint and the goal is to find the schedule that minimizes the long-term time-average cost.

To create a RMAB from (4.8), we first define the arms to represent each agent in the network. The state of every arm i consists of two non-negative integers $(A_i(t), z_i(t)) \in \mathbb{Z}^2$. Here, $A_i(t)$ is the AoI of the i-th agent while $z_i(t)$ is variable that tracks the number of remaining time-slots to finish an ongoing transmission from agent i. Thus, $z_i(t)$ is set to $r_i(\tau_i) - 1$ at the start of a new transmission. It decreases by 1 in each time-slot as the transmission proceeds and is set to 0 when agent i is not transmitting.

The state evolution of the arm (agent) depends on whether it is currently active (transmitting) or not. If agent i is transmitting in time-slot t, then it either initiates a new transmission; or the time remaining to finish sending the current update decreases by 1. Under this condition, $z_i(t)$ evolves as follows:

$$(z_i(t+1))_{u_i(t)=1} = \begin{cases} r_i(\tau_i) - 1, & \text{if } z_i(t) = 0 \\ z_i(t) - 1, & \text{otherwise.} \end{cases} \tag{4.47}$$

If the agent is transmitting in time-slot t and a new update finished delivery at time-slot $t+1$, i.e. $z_i(t+1) = 0$, then the AoI drops to the age of the delivered packet. Otherwise, the AoI increases by 1 in every time-slot.

$$(A_i(t+1))_{u_i(t)=1} = \begin{cases} \Delta_i, & \text{if } z_i(t+1) = 0, \\ A_i(t) + 1, & \text{otherwise.} \end{cases} \tag{4.48}$$

If the agent is not transmitting in time-slot t, then there is no update to be

delivered and the state evolution is simply given by

$$\big(A_i(t+1), z_i(t+1)\big)_{u_i(t)=0} = \big(A_i(t)+1, 0\big)\,. \tag{4.49}$$

For every arm i, there is a cost function $J_i(\tau_i, A_i(t))$ which maps the state of the arm $(A_i(t), z_i(t))$ to its associated costs, given the processing time allocations τ_i. This completes the MDP specification for each arm.

Since only one arm (agent) can be activated in any time-slot, the goal of the RMAB framework is to find a scheduling policy that minimizes the total time-averaged cost of running the system. Clearly, Markov decision processes evolving as above along with the associated cost functions and activation constraint are equivalent to the scheduling problem (4.8).

4.6.C Proof of Lemma 7

As in Appendix 4.6.A, we drop the subscript i and establish indexability for a generic agent, since the analysis looks similar for each of the N decoupled problems.

The *indexability* property for the decoupled problem requires that, as the transmission cost C increases from 0 to ∞, the set of AoI values for which it is optimal to transmit must decrease monotonically from the entire set (all ages $A(t) \geq \Delta$) to the empty set (never transmit). In other words, the optimal threshold H should increase as the transmission cost C increases.

We start with the case when $C = 0$. Clearly, since there is no cost for transmission and the AoI cost function $J(\tau, \cdot)$ is a non-negative increasing function, it is optimal to transmit at every value of AoI ($\forall A(t) \geq \Delta$).

Let $\widetilde{h} = h + r(\tau)$, as we have used throughout the paper. For $C > 0$, we start by

defining the function $V : \mathbb{Z}^+ \to \mathbb{R}$, for all AoI values $h \geq \Delta - r(\tau) + 1$, as follows:

$$V(h) \triangleq (\widetilde{h} - \Delta)J\big(\tau, \widetilde{h} - 1\big) - \sum_{k=\Delta}^{\widetilde{h}-1} J(\tau, k). \tag{4.50}$$

Observe that for all values of $h \geq \Delta - r(\tau) + 1$, we have $V(h+1) - V(h) = (\widetilde{h} + 1 - \Delta)(J(\tau, \widetilde{h}+1) - J(\tau, \widetilde{h})) \geq 0$. Thus, $V(\cdot)$ is an increasing function. Further, $V(\Delta - r(\tau) + 1) = J(\tau, \Delta) - J(\tau, \Delta) = 0$. Thus, $V(h)$ is a non-negative function for all values of AoI $\geq \Delta - r(\tau) + 1$.

Since $C > 0$, there are two possible scenarios - a) there exists H such that $V(H) \leq Cr(\tau) \leq V(H+1)$ or b) $V(h) \leq Cr(\tau), \forall h$. As proved in Appendix 4.6.A, if $Cr(\tau) \in \big[V(H), V(H+1)\big)$, then the optimal policy is of threshold type with the threshold being H. To map the transmission cost C to a unique optimal threshold, we choose the minimum value of AoI H for which the relation $V(H) \leq Cr(\tau) < V(H+1)$ holds. We call this value $H^*(C)$. When there is no such value of H, i.e. $V(h) \leq Cr(\tau), \forall h$ then we set $H^*(C) = \infty$.

Clearly, since the function $V(\cdot)$ is monotone, the optimal threshold $H^*(C)$ is also a non-decreasing function of the transmission cost C. This completes the proof of indexability, since we have shown that the set of states for which it is optimal to activate the arm (transmit an update) decreases monotonically as the transmission cost C increases.

The last part of the proof is to derive an expression for the Whittle index. Observe that when $C < V(H+1)/r(\tau)$, the optimal threshold is at H or lower and scheduling decision at H is to always transmit. $C = V(H+1)/r(\tau)$ is the minimum value of the transmission cost that makes both H and $H+1$ be the optimal threshold, or in other words, makes the transmit and not transmit decisions at

AoI H look equally favorable. Thus, the Whittle index is given by:

$$\begin{aligned} W(H) &\triangleq \frac{V(H+1)}{r(\tau)} \\ &= \frac{\big(\widetilde{H}+1-\Delta\big)J\big(\tau,\widetilde{H}\big) - \sum\limits_{k=\Delta}^{\widetilde{H}} J(\tau,k)}{r(\tau)} \\ &= \frac{\big(\widetilde{H}-\Delta\big)J\big(\tau,\widetilde{H}\big) - \sum\limits_{k=\Delta}^{\widetilde{H}-1} J(\tau,k)}{r(\tau)}. \end{aligned} \tag{4.51}$$

This completes our derivation of the Whittle index.

4.6.D Entropy Cost as Function of AoI

In this section, we derive the entropy cost used for the mapping application as a function of the AoI and also establish that it is a monotone increasing function.

Consider the Markov chain describing the occupancy of a cell c in region i. Its transition matrix has the following form:

$$P_i = \begin{bmatrix} 1-p_i & p_i \\ p_i & 1-p_i \end{bmatrix} \tag{4.52}$$

The stationary distribution of this Markov chain is $\mu = [0.5, 0.5]$, since $\mu P_i = \mu$. When the base station does not have any update regarding the state of the cell, it sets the probability of occupancy to be 0.5. The corresponding entropy cost is given by $-\log_2(0.5) = 1$.

Suppose that the base station believes that the cell c is occupied at time t with probability q. At time $t+1$ it does not receive any new update and needs to update its belief about the occupancy of the cell. Using the transition matrix P_i, it updates the distribution to $[q \;\; 1-q]P_i$. This distribution simply reflects the fact that one time-slot has passed and the base station needs to multiply the original

distribution by the state transition matrix to find the current *estimated* state distribution of the cell. This corresponds to the prediction step of a standard Bayes filter.

In fact, this same process is repeated for any general value of AoI. If the last received update about region i says that cell c's state distribution was $[q \;\; 1-q]$, and the current AoI for the region is A_i, then the current estimated distribution for cell c at the base station is $\hat{\mu} = [\hat{\mu}_1 \;\; \hat{\mu}_2] \triangleq [q \;\; 1-q]P_i^{A_i}$. The entropy cost for cell c is defined as:

$$J_c(A_i) \triangleq -\hat{\mu}_1 \log_2(\hat{\mu}_1) - \hat{\mu}_2 \log_2(\hat{\mu}_2). \tag{4.53}$$

We will show that $J_c(\cdot)$ is an increasing function of the AoI A_i, given a fixed value of q. Let $\nu_1 \triangleq \hat{\mu}_1(1-p_i) + \hat{\mu}_2 p_i$, and $\nu_2 \triangleq \hat{\mu}_2(1-p_i) + \hat{\mu}_1 p_i$. Then, it is easy to see that:

$$J_c(A_i+1) = -\nu_1 \log_2(\nu_1) - \nu_2 \log_2(\nu_2). \tag{4.54}$$

Note that the function $x\log_2(x)$ is convex for all $x > 0$, since $\frac{d^2}{dx^2}(x\log_2(x)) = \frac{1}{x} > 0, \forall x > 0$. Using this fact and the definitions of ν_1 and ν_2, we obtain the following inequalities:

$$(1-p_i)\hat{\mu}_1 \log_2(\hat{\mu}_1) + p_i\hat{\mu}_2 \log_2(\hat{\mu}_2) \geq \nu_1 \log_2(\nu_1), \tag{4.55}$$

$$(1-p_i)\hat{\mu}_2 \log_2(\hat{\mu}_2) + p_i\hat{\mu}_1 \log_2(\hat{\mu}_1) \geq \nu_2 \log_2(\nu_2), \tag{4.56}$$

Now, we look at the difference:

$$\begin{aligned} J_c(A_i+1) - J_c(A_i) = \\ &\left((1-p_i)\hat{\mu}_1 \log_2(\hat{\mu}_1) + p_i\hat{\mu}_2 \log_2(\hat{\mu}_2) - \nu_1 \log_2(\nu_1)\right) \\ &+ \left(p_i\hat{\mu}_1 \log_2(\hat{\mu}_1) + (1-p_i)\hat{\mu}_2 \log_2(\hat{\mu}_2) - \nu_2 \log_2(\nu_2)\right) \\ &\geq 0. \end{aligned} \tag{4.57}$$

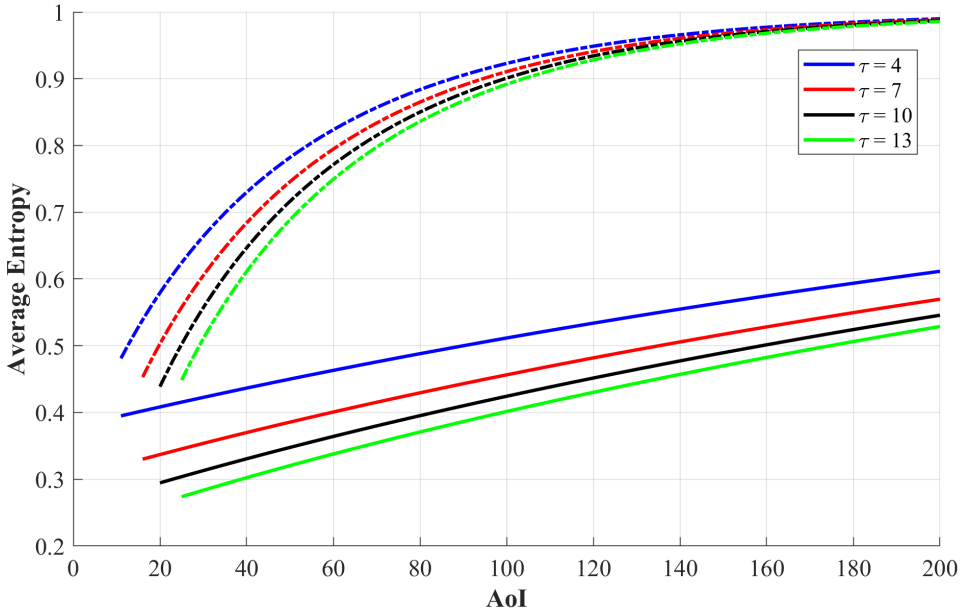


Figure 4-9: The average entropy of a region v/s AoI. Solid lines represent $p = 0.0005$ and dashed lines represent $p = 0.001$

The inequality above follows by applying (4.55) and (4.56). Since $J_c(A_i + 1) \geq J_c(A_i), \forall A_i$, so $J_c(\cdot)$ is a monotonic function of the AoI.

While we established this for a single cell in region i, the entropy cost of the entire region $J_i(A_i)$ is simply the sum of the entropies of each cell in the region. Thus, the entropy cost functions $J_i(\cdot)$ also grow monotonically with the AoI.

Another point to note is that the probability q reflects the quality of the sent update. If q is close to 0.5, the update doesn't convey much information about a cell and the entropy cost doesn't drop much on a new update. On the other hand, if q is close to 0 or 1, the update contains useful information and the entropy cost drops by a large amount. Since we use sensors that have a limited range and resolutions that improve with the processing time τ_i, the quality of updates also improves for a region with larger τ_i. This ensures that the entropy costs $J_i(\tau_i, A_i)$ satisfy the assumptions required in our co-design framework.

In Figure 4-9, we plot the entropy cost as a function of the AoI. We do so by

using our sensor for mapping a 40m $\times$ 40m region for different values of processing time τ and Markov transition probabilities p. We observe that the cost grows much more rapidly for the higher value of transition probability p. We also observe that for both values of p, the entropy cost function starts from a lower value for larger τ, denoting more useful updates for longer processing.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 5

Information Freshness in Multi-Hop Networks

In the previous chapters, we have looked at the minimization of general AoI cost functions for single-hop wireless networks. However, minimizing AoI or its cost functions over multi-hop networks has received limited attention in the literature. Finding low complexity near optimal scheduling and routing schemes for AoI minimization which handle general network topologies, interference constraints, cost functions, different types of flows and link reliabilities has remained an open problem.

In this chapter, we develop a unifying framework for making routing and scheduling decisions that minimize AoI cost in general multihop networks. In Section 5.1, we introduce the multihop problem in full generality - 1) with non-linear AoI cost functions; 2) unicast, multicast and broadcast flows and 3) considering both scheduling and routing decisions for optimization. In Section 5.2, we provide a recipe to transform AoI optimization problems into network stability problems. Instead of trying to solve for the best scheduling and routing policies directly, we assume that we have access to a set of target values which represent the average age cost for every flow in the network. These target values could be appli-

cation specific freshness requirements provided by a network administrator, or they could be the solution to an optimization program that optimizes some utility function of the average age costs.

In Section 5.2, we introduce the notion of *Age Debt* and set up a virtual queuing network that is stable if and only if there exists a feasible network control policy that can achieve the specified target costs. In Section 5.3, we use Lyapunov drift based methods to stabilize this system of virtual queues and achieve the desired target age costs. In Section 5.4, we further discuss how to choose the right age cost targets, when there is no access to either an optimization oracle or a system administrator specifying requirements for each flow. Finally, in Section 5.5, we provide detailed simulation results that compare our proposed AoI optimization methods with prior works. We find that Age Debt and its variants perform as well as or better than the best known scheduling and routing schemes in a wide variety of network settings.

5.1 Model

Consider a network with N nodes connected by a fixed undirected graph $G(V,E)$. An edge (i,j) means that nodes i and j can send packets to one another directly. We assume that at most one update can be sent over an edge in any given time-slot and takes exactly one time-slot to get delivered. We normalize the time-slot duration to unity.

Flows. The network consists of K ($\leq N$) source nodes that generate information updates. All the sources are active, i.e. they generate fresh updates on demand. A source node k has to send these updates to a set of $D_k \subset N$ destination nodes in the network. We assume that a set of nodes $C_k \subset N$ is commissioned for each flow k to forward its update packets to destination nodes. For example, a network administrator could restrict the paths over which certain flows are al-

lowed. A flow is characterized by the triplet of source node, its commissioned nodes, and the destination nodes, namely (k, C_k, D_k). For simplicity, we use k to denote both the source node k and the flow corresponding to source node k. Note that a node could be both a destination node and also a commissioned node forwarding packets for flow k, i.e. $C_k \cap D_k$ is not necessarily empty.

Flows can be of three types depending on the number of destination nodes: (1) unicast: the flow has a single destination node. (2) multicast: the flow has multiple destination nodes, which are a strict subset of the remaining nodes. (3) broadcast: every node other than the source itself is a destination node. The commissioned nodes can be either a small subset of nodes in the network needed to reach all the destination nodes, or the entire network. We assume there to be no queuing at any node and that each node maintains a single packet buffer for the freshest packet of each flow. [1]

Interference and Link States. We consider unreliable links as well as general interference constraints, i.e., transmission on all the links cannot happen simultaneously. We enumerate the set of all possible interference free choices of links and corresponding flow transmissions in the set $\mathcal{A}$. Thus, a member of set $\mathcal{A}$ contains a subset of links and corresponding flows which can be sent on these links in a single time-slot without interference. A valid network control policy must choose an action that is a member of the set $\mathcal{A}$ in every time-slot. Note that this description of $\mathcal{A}$ is very general and allows for interference constraints that depend on flow assignments. For example, consider a setting where a node is allowed to broadcast updates of a single flow to all of its neighbors in a single timestep but not send updates regarding *different* flows to each neighbor simultaneously.

For link $(i, j) \in E$, we use $U_{ij}^k(t)$ and $S_{ij}(t)$ (both $\in \{0, 1\}$) to denote the transmission decision and link state of the link (i, j) at time t. $U_{ij}^k(t)$ is 1 if a trans-

[1] Discarding older packets, or equivalently, preemptive LCFS (last come first serve) is known to be the optimal queuing discipline for AoI minimization [120].

mission of a flow-k update is scheduled on the link, at time t, and is 0 otherwise. Whereas, $S_{ij}(t)$ is 1 if a scheduled transmission on the link, at time t, will succeed; provided there is no interference. We assume $\{S_{ij}(t)\}_{t,(i,j)}$ to be independent and identically distributed processes across time t and links (i, j), with $\gamma_{ij} = \mathbb{P}\left[S_{ij}(t) = 1\right]$.

Age Evolution. For a flow k, each commissioned and destination node keeps track of the age of the freshest packet it has received. For a node $j \in C_k \cup D_k$, we denote its age for the kth flow by A_j^k(t) and it evolves as:

$$A_j^k(t+1) = \begin{cases} \min(A_j^k(t), A_i^k(t)) + 1 & \text{if } U_{ij}^k(t)S_{ij}(t) = 1 \\ A_j^k(t) + 1, & \text{if } U_{ij}^k(t)S_{ij}(t) = 0 \end{cases}, \tag{5.1}$$

for all $i \in \{k\} \cup C_k$, $j \in C_k \cup D_k$, and link $(i, j) \in E$. Note that for any flow, the source node and the commissioned nodes transmit the update packets, while other commissioned nodes and the destination nodes receive them.

Information Freshness. We consider two metrics of information freshness. The first is the average weighted sum AoI at the destination nodes:

$$A_{\text{ave}} = \lim_{T\to\infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\sum_{k=1}^{K}\sum_{j\in D_k} w_j^k A_j^k(t)\right], \tag{5.2}$$

where w_j^k denote constant weights, which determines the relative importance of a destination j and flow k, with respect to others. For the second metric, we consider general possibly non-linear functions of age. We associate a monotone increasing age cost function for each source-destination pair (k, j), where $j \in D_k$, denoted by $g_j^k(\cdot)$. We define the non-linear, effective age process to be:

$$B_j^k(t) \triangleq g_j^k(A_j^k(t)), \tag{5.3}$$

for all $t \geq 1$. The non-linear age metric is defined as:

$$B_{\text{ave}} = \lim_{T\to\infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\sum_{k=1}^{K}\sum_{j\in D_k} B_j^k(t)\right], \tag{5.4}$$

which is a generalized version of A_{ave} in (5.2).

Our goal is to minimize either A_{ave} or B_{ave} for a general, multi-hop network, by determining a policy that controls the link transmissions. A control policy needs to specify not only which links should be scheduled in each time-slot but also which flows should be transmitted along each link. We assume a centralized controller.

5.2 Age Debt

In this section, we introduce the notions of age-achievability and age debt virtual queues. We will then show how stabilizing this network of virtual queues leads to minimization of AoI. Finally, we will use quadratic Lyapunov drift to propose a heuristic scheme to achieve this stabilization in general multi-hop networks.

We consider settings with a) general increasing cost functions of AoI, b) no knowledge of fixed routing paths, i.e. the scheduler also needs to make routing decisions and c) unicast, multicast and broadcast flows in the same network. The general AoI optimization problem can be formulated as:

$$\pi^* = \underset{\pi}{\text{argmin}}\left(\lim_{T\to\infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\sum_{k=1}^{K}\sum_{j\in D_k} B_j^k(t)\right]\right), \tag{5.5}$$

where $B_j^k(t)$ are the effective age processes and $\pi(t) \in \mathcal{A}, \forall t$.

We start by assuming that we have been given a target value of time average age cost for each source-destination pair; denoted by α_j^k for the source-destination pair (k, j). We aggregate the target values associated with each source-destination

pair in the vector α. For any such target vector α, we define the notion of age-achievability below.

Definition A vector α is **age-achievable** if there exists a feasible network control policy π such that

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} B_j^k(t) \leq \alpha_j^k, \forall j \in D_k, \forall k \text{ w.p. } 1. \tag{5.6}$$

In other words, a vector α is age-achievable if the time-average of the effective age process for *every* source-destination pair (k, j) is upper bounded by the target value α_{kj}, under some feasible network control policy.

Note that the combination of general cost functions and achievability targets allows us to capture very general freshness requirements which might be useful in practical system specifications. For example, if an application requires that the empirical distribution of the age process $A_j^k(t)$ should satisfy $\mathbb{P}(A_j^k(t) \geq M) \leq \epsilon$, then we can capture this by setting the cost function $g_j^k(h) = \mathbf{1}_{\{h \geq M\}}$ and the corresponding target to be $\alpha_j^k = \epsilon$.

We now define a set of virtual queues called age-debt queues for every source-destination pair (k, j). These queues measure how much the effective age process exceeds its target value α_{kj}, summed over time. Our definition of debt is inspired by the notion of throughput debt as introduced in [121].

Definition Given a target vector α, the **age debt queue** for source-destination

pair (k, j) at time t, given by $Q_j^k(t)$, evolves as

$$Q_j^k(t+1) = \left[Q_j^k(t) + B_j^k(t+1) - \alpha_j^k \right]^+, \forall j \in D_k, \quad \text{and } \forall k \in \{1, ..., K\}. \tag{5.7}$$

To complete the definition, each age debt queue starts at zero, i.e. $Q_j^k(0) = 0, \forall j, k$.

We now introduce a notion of stability for these age debt queues. This is similar to how rate stability is typically defined in queueing networks [122].

Definition We say that the network of age debt queues is **stable** under a policy π and a given target vector $\boldsymbol{\alpha}$ if the following condition holds:

$$\lim_{T\to\infty} \mathbb{E}\left[\sum_{k=1}^{K} \sum_{j\in D_k} \frac{Q_j^k(T)}{T} \right] = 0, \tag{5.8}$$

where the expectation is taken over the randomness in the channel processes and the scheduling policy π.

We also establish an equivalence relationship between age-achievability of a vector $\boldsymbol{\alpha}$ and the stability of the corresponding network of age debt queues.

Lemma 8. *A target vector $\boldsymbol{\alpha}$ is age-achievable if and only if there exists a network control policy π, that stabilizes the network of source-destination age debt queues.*

Proof. See Appendix 5.7.A. □

Next, we define a debt-stable scheduling policy. Such a policy takes a target vector $\boldsymbol{\alpha}$ as an input and stabilizes the network of corresponding age debt queues.

Definition A **debt-stable** scheduling policy π stabilizes the set of age-debt queues for any given target vector $\boldsymbol{\alpha}$ that is age-achievable.

The notions introduced until now effectively allow us to convert the minimum age cost problem described by (5.5) into a network stability problem. Suppose π^* is a solution to the optimization problem (5.5). Further, suppose that the time average of the effective age process for pair (k, j) under π^* is given by

$$\lim_{T\to\infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} B_j^{k^*}(t)\right] = \alpha_j^{k^*}, \forall (k,j). \tag{5.9}$$

Clearly, if we have oracle access to an optimal age cost vector $\boldsymbol{\alpha}^* = \{\alpha_j^{k^*}\}_{(k,j)}$ and know how to design a debt-stable policy then we can perform minimum age cost scheduling. If the debt-stable policy is much lower in computational complexity than solving (5.5) directly, then we can also solve (5.5) at the same lower complexity (assuming oracle access to $\boldsymbol{\alpha}^*$).

5.3 Lyapunov Drift Approach

Next, we discuss a heuristic approach to designing debt-stable policies.

5.3.A Single-Hop Broadcast

We first consider the special case of single-hop broadcast networks. This setting is easier to analyze since it only requires scheduling and no routing and it also

highlights key structural properties of our proposed policy.

Consider a N node star network where each of the nodes $1, ..., N-1$ has an edge to node N. These nodes wish to send packets to the central node N. The edges are numbered $e_1, ..., e_{N-1}$. Due to broadcast interference constraints, only one node can transmit in any given time-slot. Since the destination for every flow is N, we can drop the destination in our notation. The age evolution is given by

$$A_i(t+1) = \begin{cases} A_i(t)+1, & \text{if } U_{e_i}(t)S_{e_i}(t) = 0 \\ 1, & \text{if } U_{e_i}(t)S_{e_i}(t) = 1. \end{cases} \tag{5.10}$$

Given an age-cost function $g_i(A_i(t))$ and a corresponding target value α_i, the debt queue evolution for node i is given by:

$$Q_i(t+1) = \Big[Q_i(t) + g_i(A_i(t+1)) - \alpha_i\Big]^+. \tag{5.11}$$

Given a target vector $\boldsymbol{\alpha}$, we will use a Lyapunov drift based scheduling scheme to try and achieve debt stability. To do so, we first define a Lyapunov function for our system of virtual queues:

$$L(t) \triangleq \sum_{i=1}^{N-1} Q_i^2(t). \tag{5.12}$$

Using this Lyapunov function, we then define the *age debt scheduling policy* π^{AD} as:

$$\pi^{\text{AD}}(t) = \underset{a\in\mathcal{A}}{\operatorname{argmin}}\Big(\mathbb{E}\big[L(t+1) - L(t)\big]\Big), \tag{5.13}$$

where the expectation is taken over the randomness in channel reliabilities $\boldsymbol{S}(t)$.

In the following remark, we consider a variant of the age-debt policy that minimizes an upper-bound on the Lyapunov drift instead of the actual Lyapunov drift as in (5.13). This is similar to the upper-bound drift minimization used in policies

such as the classical max-weight for throughput optimization [123] and allows us to compare the structure of age-debt to preivously proposed policies in literature.

***Remark* 1.** *Suppose that the links between each source i and the destination N are i.i.d. Bernoulli w.p. γ_i in every time-slot. Further, if each age cost function $g_i(\cdot)$ is upper bounded by a large constant D, then the policy $\pi(t)$ below minimizes an upper bound on the Lyapunov drift in every time-slot.*

$$\pi(t) = \underset{i\in 1,\ldots,N-1}{\operatorname{argmax}}\Big(\gamma_i Q_i(t)\big(g_i(A_i(t)+1) - g_i(1)\big)\Big). \tag{5.14}$$

Proof. See Appendix 5.7.B. □

In other words, an approximate drift minimizing policy chooses the source with the largest product of link reliability, current age debt and current age cost. This structure of the age-debt policy can be contrasted with the max-weight policy proposed in [124] which chooses the source with the largest value of $\gamma_i w_i A_i(t)(A_i(t)+2)$ given weights w_i. Similarly, the Whittle index policy proposed in Chapter 2, chooses the source with the largest value of $W_i(A_i(t))$, where $W_i(\cdot)$ is Whittle-index corresponding to the age cost $f_i(\cdot)$.

Note that to compute $\pi^{\mathrm{AD}}(t)$, the scheduler needs to iterate over the set of sources only once. So the per slot computational complexity of this policy grows linearly in N. This is similar to the complexity of the Whittle index policy we proposed in Chapter 2 and the max-weight policies proposed in [15, 17]. By contrast, a dynamic programming approach to solve (5.5) directly has per slot computational complexity that grows exponentially in N. This highlights the key strength of our approach. If the scheduler has some way to set the targets for each source

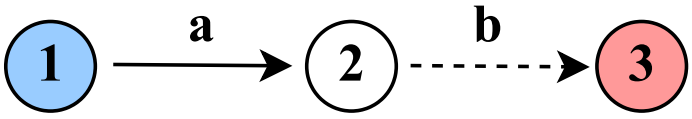


Figure 5-1: Example of a line network with a unicast flow from node 1 to node 3

optimally, then the age debt policy is a good low complexity heuristic for age minimization.

5.3.B General Networks

The general multihop setting is more challenging. Simply using one-slot Lyapunov drift to try and achieve debt stability does not work directly in the multihop setting. We highlight this with a simple example.

Consider the three node network described in Figure 5-1 with a single unicast flow from node 1 to node 3. The interference constraint enforces that only one of the two edges a and b can be activated in any time-slot. Suppose that we are interested in minimizing the time average of the age process $A_3^1(t)$. Given a target value α_3^1, we set up the age debt queue as follows:

$$Q_3^1(t+1) = \left[Q_3^1(t) + A_3^1(t+1) - \alpha_3^1\right]^+. \tag{5.15}$$

We will try to use the one slot Lyapunov drift minimizing policy to stabilize $Q_3^1(t)$ in this network. To do so, we solve the following optimization in every time-slot:

$$\pi^{\mathrm{AD}}(t) = \underset{x\in\{a,b\}}{\operatorname{argmin}}\left(\mathbb{E}\left[(Q_3^1(t+1))^2 - (Q_3^1(t))^2\right]\right). \tag{5.16}$$

At $t=1$, activating either edge a or edge b has no effect on the debt $Q_3^1(2)$ since node 2 does not have any packet from node 1. If we break ties in favour of edge b, then it is activated but no new packet is delivered to node 3. At $t=2$, since node 2 still does not have any new update from node 1, no action taken can affect the

debt $Q_3^1(3)$. Using the same tie-break rule, we would again schedule edge b. This process keeps on repeating and the age debt queue Q_3^1 blows up irrespective of the value of α_3^1, even though the age optimal policy in this setting is to simply alternate between a and b in every time-slot.

The example above illustrates why one-slot Lyapunov drift based techniques fail in stabilizing debt queues in multihop networks. The policy designer using Lyapunov drift is constrained to optimizing *only one time-step into the future*, similar to a greedy policy. So, if every possible scheduling and routing action has no effect on the age debt queues in the immediate next time-slot, the one step drift minimizing procedure does not provide any information on which action should be chosen to stabilize the debt queues.

This suggests that to be able to use one-slot drift minimizing techniques for stability, there should be a virtual queue for every intermediate node that tracks both the current age debt at the destination and the potential reduction in debt at the destination upon forwarding a fresh packet. If we can set up such queues, then large values of debt at intermediate nodes would lead to fresh packets being sent to the next hops via one-slot drift minimizing actions, eventually reaching the destination and stabilizing the age debt queues.

Let $Q_j^{k\to i}(t)$ denote such a debt queue corresponding to flow (k, j) at an intermediate node i. These additional queues at every intermediate node combined with the original debt queues form our virtual network. The Lyapunov function that we use for scheduling and routing is given by:

$$L(t) \triangleq \sum_{k=1}^{K} \sum_{j\in D_k} \left((Q_j^k(t))^2 + \sum_{i\notin D_k, i\neq k} (Q_j^{k\to i}(t))^2 \right) \tag{5.17}$$

The Age Debt scheduling and routing policy is to choose the activation set and

corresponding flows that minimizes the expected Lyapunov drift.

$$\pi^{\mathrm{AD}}(t) = \underset{a\in\mathcal{A}}{\operatorname{argmin}}\Big(\mathbb{E}\big[L(t+1) - L(t)\big]\Big), \tag{5.18}$$

where the expectation is taken over the randomness in channel reliabilities $\boldsymbol{S}(t)$.

5.3.C Intermediate Debt Queues

We now discuss how to set up the age debt queues $Q_j^{k\to i}(t)$ for intermediate nodes to augment the original network of queues. Note that there are no intermediate nodes for broadcast flows since every node other than the source is a destination.

Consider a source-destination pair (k, j) for a unicast/multicast flow k and an intermediate node i that is not a destination for the flow originating at k. We want to set up the age debt queue $Q_j^{k\to i}(t)$ at i for the pair (k, j). We maintain an age process for flow k at node i, even though there is no associated cost or target value for this age process.

$$A_i^k(t+1) = \begin{cases} \min(A_i^k(t), t - t_g) + 1, \text{if update generated} \\ \qquad\qquad \text{at time } t_g \text{ is delivered at time } t. \\ A_i^k(t) + 1, \text{if no new delivery at time } t. \end{cases} \tag{5.19}$$

Here $A_i^k(t)$ measures how old the information at node i is regarding node k. We split the debt queue's evolution into two cases.

Case 1: When node i forwards a flow k packet on a set of adjacent links L. Let h_{ij}^L be the minimum number of hops it takes to reach node j from node i, where the first hop can only include edges in the set L. Here, h_{ij}^L measures the minimum delay with which the packet that was forwarded by i gets delivered at j. The age debt queue $Q_j^{k\to i}(t)$, when node i is forwarding a flow k packet along the link set

L, evolves as:

$$Q_j^{k\to i}(t+1) = \left[Q_j^{k\to i}(t) + g_j^k\big(\min\{A_i^k(t), A_j^k(t)\} + h_{ij}^L\big) - \alpha_j^k\right]^+. \tag{5.20}$$

This measures the most optimistic change in age debt possible at the destination using the current packet transmission from node i.

Case 2: When node i does not forward a packet from node k along any of its adjacent edges, then the age debt queue evolves as below.

$$Q_j^{k\to i}(t+1) = \left[Q_j^{k\to i}(t) + B_j^k(t+1) - \alpha_j^k\right]^+. \tag{5.21}$$

This means that the intermediate queue simply tracks the change in debt at the destination when it is not forwarding a relevant packet. If the destination is not receiving fresh packets from anywhere in the network then this would increase the intermediate debt queue.

Thus, the debt at an intermediate node i for a source-destination pair (k, j) blows up if (a) either the destination has not received fresh packets for a long time and node i did not forward any packets from k (i.e. (5.21)) or if (b) node i keeps forwarding stale packets from k (i.e. (5.20)). A drift minimizing policy will then try to ensure that either the destination debt queue is small, or node i forwards fresh packets of flow k towards the destination.

5.3.D An Example

Consider the five node network depicted in Fig. 5-2. We consider two flows in this network - a multicast flow from source 1 to destination nodes 3 and 5 and a unicast flow from source 4 to destination node 1. We further consider interference constraints such that every node interferes with every other node, so in any given time-slot only one node can transmit successfully. Further, the transmitting node

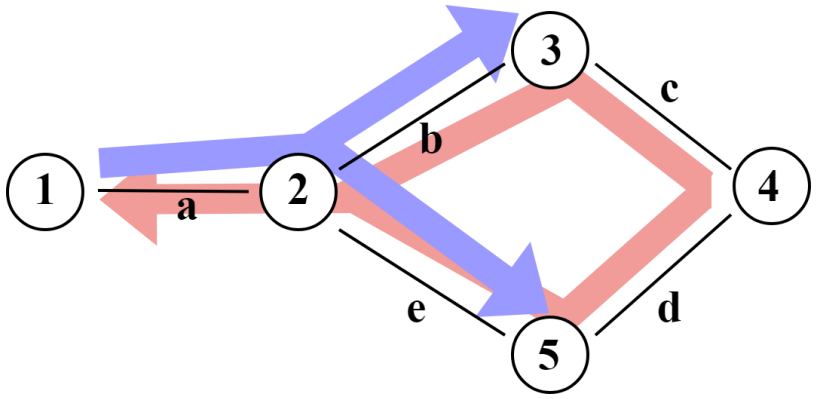


Figure 5-2: Example of a five node multihop network with two competing flows.

can only transmit a single flow's packet on all of it links, i.e. it cannot choose to send different flow packets on different links. Thus, the available choice of actions in each time-slot is to decide which node gets to transmit and regarding which flow.

Given average AoI targets α_3^1, α_5^1 and α_1^4, we will use our proposed age debt scheme to achieve them. Observe that edge a between nodes 1 and 2 must be used by both flows and hence acts like a bottleneck link. One possible schedule that we might want to replicate is to send packets for the first flow, and then send packets for the second flow, and keep alternating. If we represent actions as a tuple (i, j) denoting that node i forwards an update regarding node j, then this scheduling policy looks like $(1,1) \rightarrow (2,1) \rightarrow (4,4) \rightarrow (5,4) \rightarrow (2,4)$. It is easy to see that the average AoIs achieved by this policy are $\alpha_3^1 = 4.0, \alpha_5^1 = 4.0$ and $\alpha_1^4 = 5.0$.

Providing these targets to the age debt queues, we confirm via simulation that our one slot Lyapunov drift minimization method does indeed achieve the required average AoIs. To understand how age debt makes scheduling decisions, we will focus on the bottleneck link a. Recall that $Q_j^{k\rightarrow i}$ is the intermediate debt queue at a forwarding node i for the source k and destination j. The debt queues that primarily influence which flow gets transmitted on link a are $Q_3^{1\rightarrow 1}(t), Q_5^{1\rightarrow 1}(t)$,

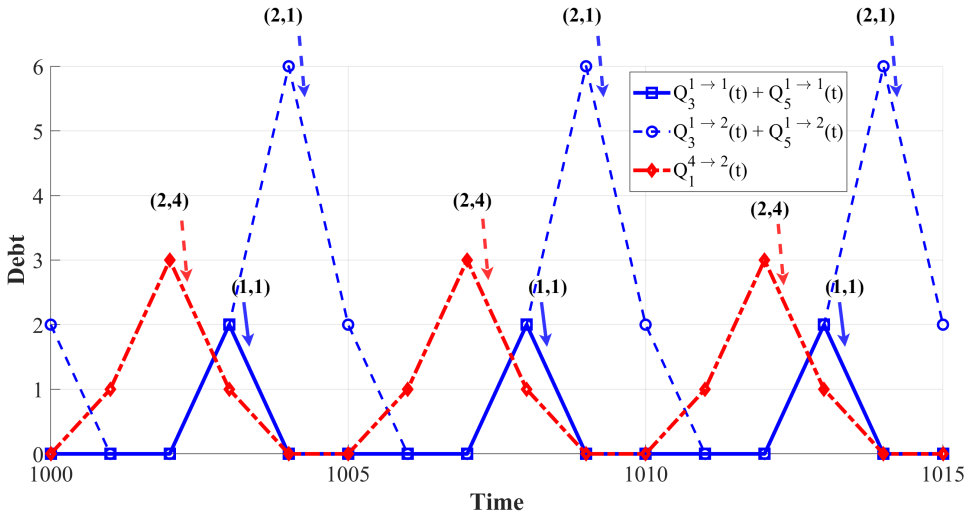


Figure 5-3: Evolution of three debt queue quantities with time, along with scheduling decisions involving link a.

$Q_3^{1\to 2}(t), Q_5^{1\to 2}(t),$ and $Q_1^{4\to 2}(t)$.

Given $\alpha_3^1 = 4.0, \alpha_5^1 = 4.0$ and $\alpha_1^4 = 5.0$, we will plot three debt queue quantities once the age debt policy has reached steady state. Specifically, we plot $Q_3^{1\to 1}(t) + Q_5^{1\to 1}(t), Q_3^{1\to 2}(t) + Q_5^{1\to 2}(t),$ and $Q_1^{4\to 2}(t)$.

There are three competing actions that the scheduling policy can take for the link a. These are $(1,1)$ - node 1 broadcasts an update about itself along link a; $(2,1)$ - node 2 broadcasts an update about node 1 along links a, b and e; and $(2,4)$ - node 2 broadcasts an update about node 4 along links a, b and e. In Fig. 5-3, we plot the three debt queue quantities of interest along with timeslots in which scheduling decisions involving link a were made.

We observe that the relative ordering of the three debt quantities explains why each scheduling decision is made. When $Q_3^{1\to 1}(t) + Q_5^{1\to 1}(t)$ is the largest, it suggests that sending an update from node 1 to node 2 along link a is the most valuable action. Similarly, when $Q_3^{1\to 2}(t) + Q_5^{1\to 2}(t)$ is the largest, it suggest that sending an update regarding node 1 from node 2 to nodes 3 and 5 is the most valuable action (and due to the interference constraint, it blocks any other node from us-

ing the link a). When $Q_1^{4\to 2}(t)$ is the largest, it suggests that sending an update regarding node 4 from node 2 to node 1 along link a is the most valuable action.

Note that scheduling decisions are not explicitly made based on relative ordering of the debt queues, but by finding the action that has the largest negative Lyapunov drift. We use the relative ordering to provide an intuitive explanation of age debt. Importantly, the evolution of the debt queues and the Lyapunov drift approach together lead to a scheduling policy that achieves the desired AoI targets. Adjusting the targets and/or AoI cost functions, the system designer can tradeoff the freshness of one flow against another.

5.4 Choosing Target Vectors

In the preceding sections, we have developed a general framework of age achievability where given a target average age cost for every source-destination pair, we formulate a corresponding network stability problem and attempt to solve it via one slot Lyapunov drift minimization. In this section, we discuss how to choose the right target vectors, such that they lead to minimum sum age cost.

In the absence of an optimization oracle that provides access to $\boldsymbol{\alpha}^*$ or a system administrator who specifies average age cost targets based on the underlying application requirements, we develop a simple heuristic to dynamically update $\boldsymbol{\alpha}$ in order to optimize utility based on the state of the underlying debt queues.

The following optimization problem needs to be solved to find the best target vector $\boldsymbol{\alpha}^*$.

$$\begin{aligned} &\underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \left(\sum_{k=1}^{K} \sum_{j \in D_k} \alpha_j^k \right), \\ &\text{s.t. } \boldsymbol{\alpha} \text{ is age-achievable.} \end{aligned} \tag{5.22}$$

Note that this problem has the same optimal value as (5.5).

5.4.A Gradient Descent

We want to use a gradient descent like approach to solve (5.22) and find $\boldsymbol{\alpha}^*$. The problem with doing so is that we do not have a simple characterization of the age-achievability region or a low complexity method to test whether a vector is achievable or not.

To resolve this, we use Lemma 8. If the network of source-destination age debt queues is unstable for a given value of $\boldsymbol{\alpha}$, then $\boldsymbol{\alpha}$ lies outside the age-achievability region. This immediately suggests the gradient descent like approach described in Algorithm 7.

Algorithm 7: Age Debt - Gradient Descent

Input : epoch size W, number of epochs E, step-size $\eta > 0$, threshold $\epsilon > 0$, initialization $\boldsymbol{\alpha}(1)$

while $e \in 1, ..., E$ **do**
  Set up age debt queues using $\boldsymbol{\alpha}(e)$ and initialize each queue to 0
  while $t \in 1, ..., W$ **do**
    Schedule and route using age debt
    $$\pi^{\mathrm{AD}}(t) = \underset{a \in \mathcal{A}}{\operatorname{argmin}} \Big(\mathbb{E}\big[L(t+1) - L(t)\big] \Big),$$
  end
  if $\exists$ *flow* k *and* $j \in D_k$ *s.t.* $Q_j^k(W) > \epsilon W$ **then**
    Increase target values for unstable queues:
    $\alpha_j^k(e+1) = \alpha_j^k(e) + \eta, \forall (k,j)$ s.t. $Q_j^k(W) > \epsilon W$
    Other targets remain unchanged:
    $\alpha_j^k(e+1) = \alpha_j^k(e), \forall (k,j)$ s.t. $Q_{kj}(W) \leq \epsilon W$
  end
  else
    Update all target values using gradients: $\alpha_j^k(e+1) = \alpha_j^k(e) - \eta$, $\forall (k,j)$.
  end
end

The algorithm above runs the age debt policy for epochs of length W time-slots. Within an epoch, the target vector $\boldsymbol{\alpha}$ remains fixed. At the end of the epoch,

we use the value of the source-destination age debt queues $Q_j^k(\cdot)$ to update the corresponding targets. If the network has at least one queue with debt larger than a threshold, it suggests that the current vector is not achievable. So, we increase the values of $\boldsymbol{\alpha}$ for the source-destination pairs with large values of debt. If the network has all queues with debt below a threshold, the current vector is likely achievable. So, we update the entire target vector using gradient descent. Note that this approach takes a large number of time-slots to converge to a good candidate target vector $\boldsymbol{\alpha}$.

5.4.B Flow Control

Another way to dynamically set the target vectors is to take a flow control approach for solving the optimization problem (5.22), similar to [123]. Algorithm 8 describes the details.

Algorithm 8: Age Debt - Flow Control

Input : parameter $V > 0$, upper bound $\alpha_{\max}$, initialization $\boldsymbol{\alpha}(1)$

while $t \in 1, ..., T$ **do**

- Use $\boldsymbol{\alpha}(t)$ to update debt queue values at time t
- Update $\boldsymbol{\alpha}$ by solving the optimization below:

$$\boldsymbol{\alpha}(t+1) = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \left(\sum_{k=1}^{K} \sum_{j \in D_k} V\alpha_j^k - \alpha_j^k Q_j^k(t) \right), \quad \text{s.t. } \boldsymbol{\alpha} \geq 1, \boldsymbol{\alpha} \leq \alpha_{\max}.$$

- Use $\boldsymbol{\alpha}(t+1)$ to compute the scheduling and routing decision that minimizes drift: $\pi(t) = \underset{a \in \mathcal{A}}{\operatorname{argmin}} \left(\mathbb{E}\left[L(t+1) - L(t) \right] \right)$

end

The flow control based age debt policy tries to tradeoff between the stability of the queueing network and the optimization of targets using a parameter $V > 0$. In

every time-slot, the flow control optimization sets the target α for the next time-slot and then the scheduling and routing decisions are computed by minimizing Lyapunov drift.

The update optimization in step 4 of Algorithm 8 can be simplified to the rule below:

$$\alpha_j^k(t+1) = \begin{cases} \alpha_{\max}, & \text{if } Q_j^k(t) > V \\ 1, & \text{if } Q_j^k(t) \leq V, \end{cases} \quad \forall (k,j). \tag{5.23}$$

Thus, instead of converging to a target vector as in the case with gradient descent, the flow control approach dynamically switches the value of targets in every time-slot. This means we do not need to wait a long period of time for convergence. When current debts are high, future targets are set to be high pushing the debts lower. Similarly, when the current debts are low, future targets are also set low, pushing the debts higher. The parameter V decides the threshold between high and low values of the debt queues.

5.5 Numerical Results

First, we consider the weighted-sum AoI problem in single-hop broadcast networks with unreliable channels. There are N nodes in the network and the weight of the ith node w_i is set to i/N. Link connection probabilities are chosen uniformly from the set $[0.6, 1]$. Figure 5-4 plots the performance of the age debt policy, the age difference policy proposed in [5], the optimal stationary randomized policy proposed in [15], and the max-weight and Whittle index policies proposed in [15]. The last two are known to be close to optimal for this setting.

First, we observe that the optimal stationary randomized policy performs much worse than the other classes of policies. Age difference performs better than the randomized policy but not as well as the Whittle-index or max-weight policies.

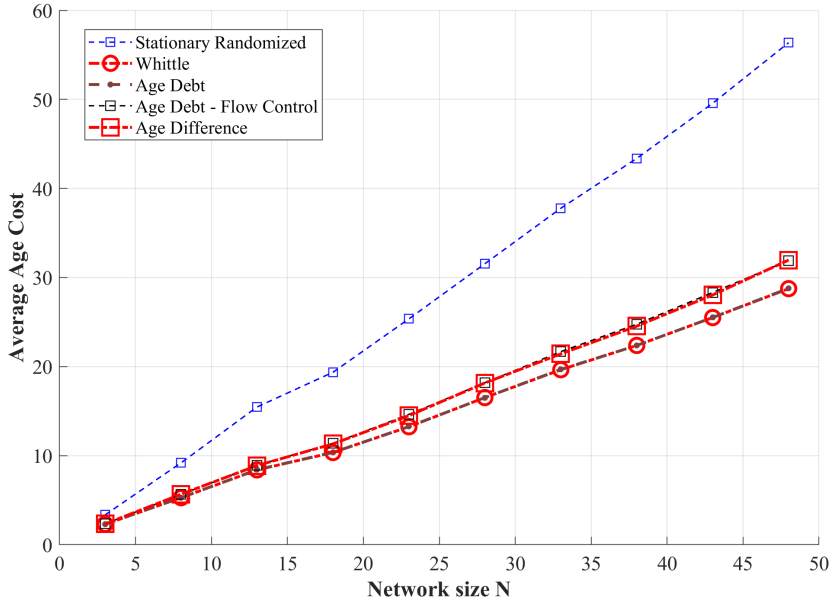


Figure 5-4: Weighted-sum AoI minimization in broadcast networks with unreliable channels

We further observe that when the age debt policy is provided the max-weight average cost as the target vector, it replicates near optimal performance. Also, the flow control and gradient descent versions of age debt have a small gap to the max-weight/Whittle policies despite not having access to α beforehand and perform as well as the age-difference policy.

Next, we consider general functions of age minimization in the single-hop wireless broadcast setting. There are N nodes in the network and the cost of AoI for each node is chosen from the set of functions $\{15A(t), e^{A(t)}, (A(t))^2$ and $(A(t))^3\}$. Figure 5-5 plots the performance of the age-debt policy and its variants along with the age difference policy proposed in [7] and the Whittle index policy proposed in [125]. As for the linear AoI case, we observe that age debt is able to replicate the Whittle policy's performance when provided its average cost as the target vector. The flow control and gradient descent variants are also only a small gap away in performance without knowing α beforehand. On the other hand,

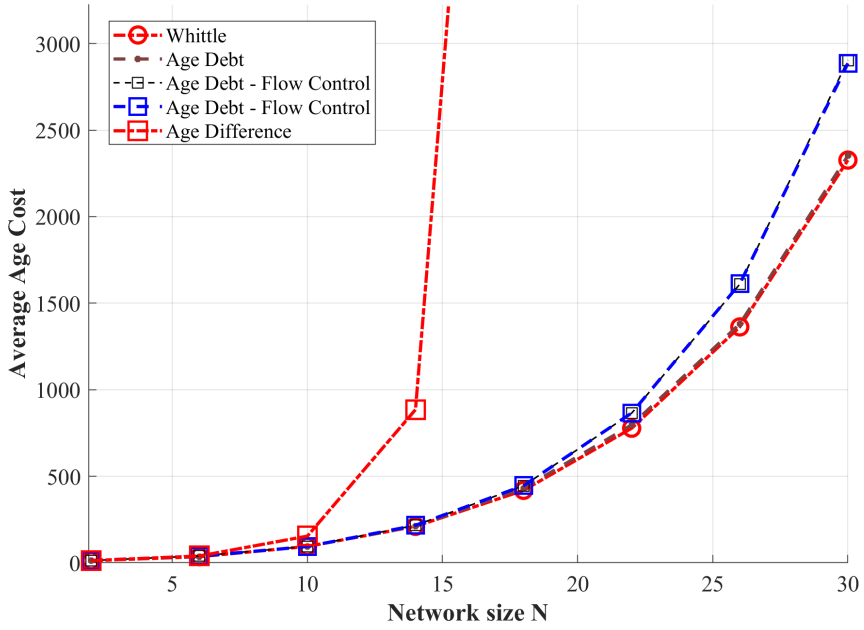


Figure 5-5: Functions of Age minimization in broadcast networks with reliable channels

the age difference policy performs much worse and the AoI cost rapidly grows very large even for moderate N. This is because the age difference policy is not designed to handle general AoI cost functions, so even though it tries to keep the AoIs small for all nodes, their actual impact to cost can become very large. We also showed in Chapter 2 that even the optimal stationary randomized policy can have unbounded AoI cost for systems as small as $N = 2$, given nonlinear AoI cost functions. So, we do not plot its performance in this scenario.

We also look at the functions of age problem with $N = 4$ in more detail. The age cost functions for each node are as follows $f_1(A_1(t)) = 15A_1(t)$, $f_2(A_2(t)) = e^{A_2(t)}$, $f_3(A_3(t)) = (A_3(t))^2$ and $f_4(A_4(t)) = (A_4(t))^3$. First, we use dynamic programming to compute the optimal policy π^* which minimizes average age cost. The time average age costs under this policy are given by $\alpha_1^* = 45.0, \alpha_2^* = 14.52, \alpha_3^* = 17.20$, and $\alpha_4^* = 11.0$, while the total sum cost is 87.72. Using these as target values, we set up debt queues and implement the age-debt policy.

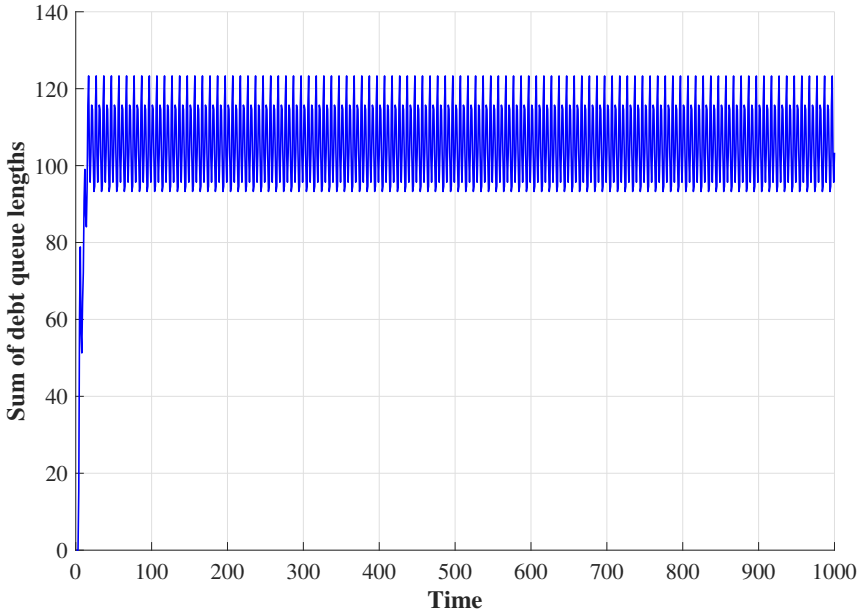


Figure 5-6: Sum of virtual debt queues vs time

Figure 5-6 plots the sum of the 4 age debt queues $\sum_{i=1}^{4} Q_i(t)$ under the age-debt policy implemented using the optimal α^* from above. We observe that the age debt policy indeed stabilizes the debt queues since queue lengths don't grow with time. As a corollary, it also achieves age cost optimality in this setting. On the other hand, our Whittle index policy from Chapter 2 achieves a total sum cost of 88.34, a fixed but small distance away from the optimal cost of 87.72. This suggests that age-debt might be a way to achieve exact optimality instead of near optimality when access to α^* is available.

Next, we consider scheduling for a single unicast flow on the line network. Consider N nodes arranged in a line network from 1 to N. Node 1 wants to sent packets to node N, however not all nodes can transmit simultaneously. We consider a simple interference constraint - in any given time-slot either all even numbered nodes or all odd numbered nodes can forward packets. This ensures that no two adjacent nodes send interfering transmissions. Figure 5-7 plots the performance of age-debt and its flow-control and gradient-descent variations along

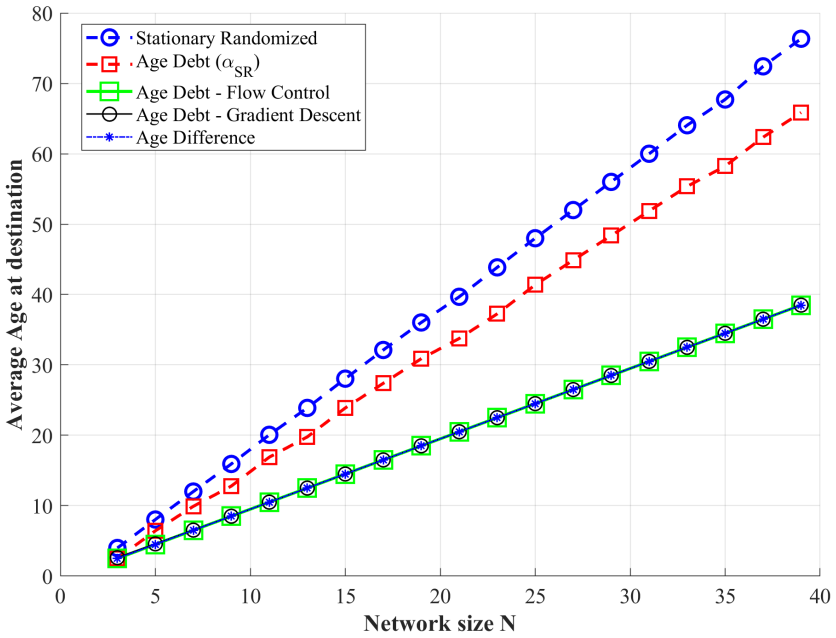


Figure 5-7: A single unicast flow on a line network (neighboring nodes interfere)

with the optimal stationary randomized policy and the age difference policy, both derived for this setting in [5]. We observe that age-debt outperforms the stationary randomized policy despite using its average cost α_{SR} as the target vector. The dynamic variants of age-debt significantly outperform the stationary randomized policy and match the performance of the age-difference policy. We also note that the gap in performance would increase in settings with multiple flows and paths available which age-debt can utilize for routing, unlike the stationary randomized and age difference approaches.

We also consider a different kind of interference constraint in the same line network example. Now, all nodes interfere with one another, and only one node can transmit successfully in any given time-slot. We plot the performance of the optimal stationary randomized policy, the age-debt policy (provided α_{SR}), our age-debt variants without any knowledge of α and the age difference policy against the number of nodes in the system in Figure 5-8. We again observe a large

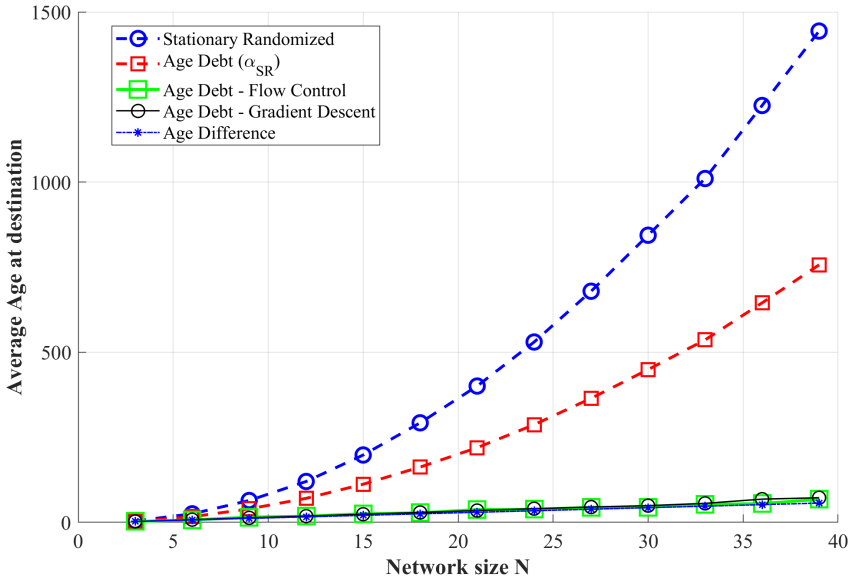


Figure 5-8: A single unicast flow on a line network (all nodes interfere)

gap in performance between the optimal randomized policy and our proposed methods. This is consistent with the line network AoI analysis from [7], where we showed that the best stationary randomized policy has performance that is $O(N^2)$ while the age difference policy has performance $O(N)$. We also observe that the age-debt variants match the performance of the age difference policy, which can be shown to be exactly optimal in this single source line network setting.

Finally, we consider average age minimization for all-to-all broadcast flows in multihop networks similar to [50]. Note that this is a broadcast setting that requires both scheduling and routing decisions to be made, so we cannot use the stationary randomized or age difference policies developed in [7]. We consider all possible connected network topologies with 5 or 6 nodes (a total of 133 graphs). Figure 5-9 plots the performance of the age-debt policy and its variants along with the near optimal minimum connected dominating set (MCDS) based scheme proposed in [50] for each of these networks. The x-axis represents the graph labels numbered from 1 to 133, sorted according to the average age

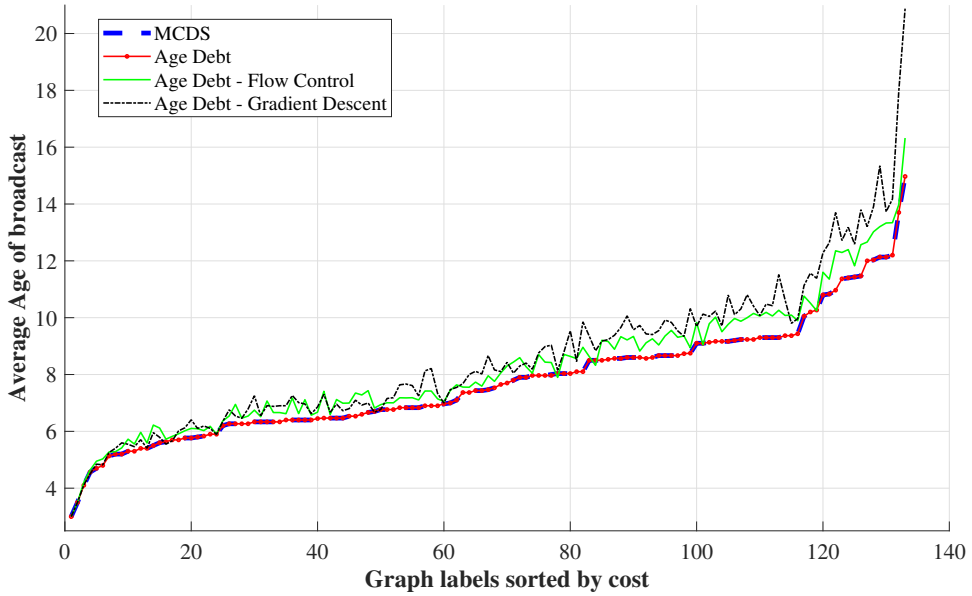


Figure 5-9: Broadcast flows in multihop networks with 5 and 6 nodes

achieved by the MCDS scheme.

We observe that age-debt achieves the same performance as the MCDS scheme when provided its average cost as the target vector. Further, age-debt with flow control achieves performance that is very close to that of the MCDS scheme without requiring knowledge of α. Importantly, the MCDS scheme can only be applied to this setting of all-to-all broadcast with one node transmitting at a time. Further, computing the optimal schedule using the MCDS scheme requires finding minimum size connected dominating sets, the complexity of which grows exponentially in the number of nodes.

We also consider the same broadcast setting but now with weighted-sum AoI as the minimization objective instead of just AoI. We consider all possible connected graphs with 5 nodes (21 in total). We set the importance weight of one node to 15 (giving it a higher priority) and the rest of the 4 nodes to 1. Figure 5-10 plots the performance of the MCDS scheme along with age-debt and its variants. As expected, age-debt policy replicates the performance of the MCDS scheme

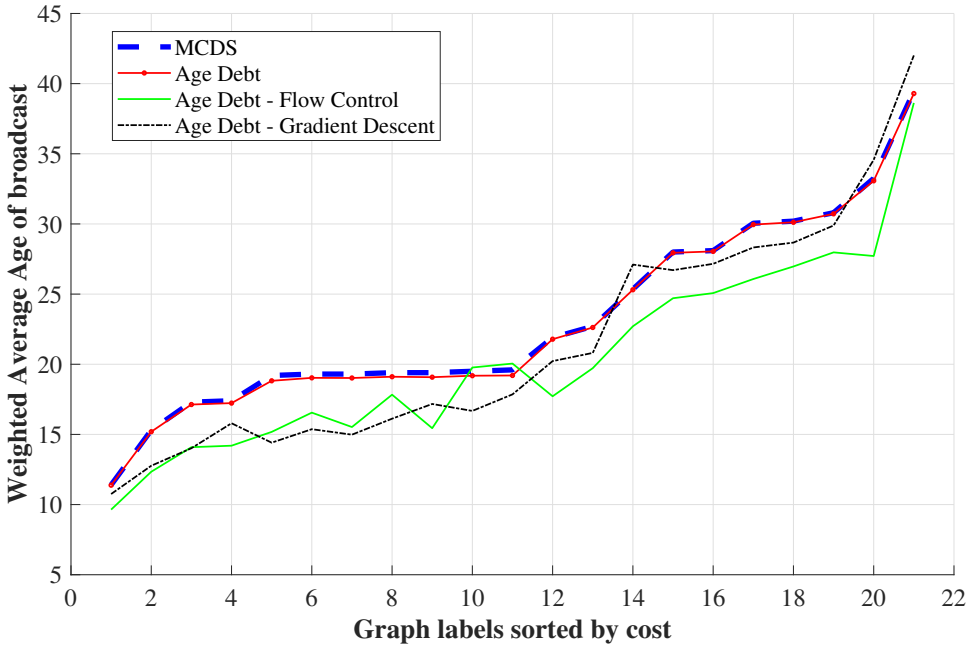


Figure 5-10: Weighted Age minimization of broadcast flows in multihop networks with 5 nodes

since it is provided the average age-cost realized by MCDS as the target. Interestingly, flow-control outperforms MCDS since it is able to adapt to a better target α in the presence of weights and asymmetry. This is consistent with the fact that the MCDS scheme is not designed for minimizing weighted-sum AoI. It also highlights the relative ease with which age-debt can be adapted to weights and general AoI cost functions.

Note that the complexity of implementing the flow-control scheme is polynomial in the network size per time-slot. This suggests that age-debt and its variants are a good candidate for low complexity near optimal age scheduling in general networks.

We also observe that the flow control variant of age-debt is the method of choice in the absence of known α. During our experiments, we found that the gradient descent variant has parameters that are hard to configure for networks of different sizes and takes a long time to converge. The flow-control method has

just two parameters V and $\alpha_{\max}$ that are relatively easy to configure and do not require any time for convergence.

5.6 Summary

Evidence from simulations suggests that Age-Debt and its variants that we propose in this chapter are good candidates to optimize AoI costs over general multi-hop networks. A natural next question to ask is whether there are performance guarantees for this class of policies. We believe that standard Lyapunov analysis is not sufficient to answer this question, and more work needs to be done in the multi-hop setting.

5.7 Appendix

5.7.A Proof of Lemma 8

We will prove this under the assumption that the AoI cost functions $g_j^k(\cdot)$ are upper-bounded by a fixed constant D for every source-destination pair (k, j). This is a mild assumption because D can be set to a very high value (in the order of years) which will never be attained in practical systems under any reasonable policy.

We note that the arrival process to the debt queue $Q_j^k(t)$ is given by the effective age process $B_j^k(t)$, while the departures in every time-slot are just α_j^k. Using the boundedness assumption, both arrivals and departures are strictly upper-bounded by D. The result immediately follows from Theorem 2(c) in [122] which relates mean-rate stability of a queue to time-averages of the arrival and departure processes.

5.7.B Proof of Remark 1

The debt queues in this setting evolve as follows:

$$Q_i(t+1) = \left[Q_i(t) + g_i(A_i(t+1)) - \alpha_i \right]^+, \forall i. \tag{5.24}$$

The AoI evolves as:

$$A_i(t+1) \begin{cases} A_i(t)+1, & \text{if } U_{e_i}(t)S_{e_i}(t) = 0 \\ 1, & \text{if } U_{e_i}(t)S_{e_i}(t) = 1. \end{cases} \tag{5.25}$$

Here $S_{e_i}(t) = 1$ i.i.d. with probability γ_i in every time-slot.

Let $\Delta(t) \triangleq L(t+1) - L(t)$. Then,

$$\begin{aligned} \mathbb{E}[\Delta(t)] &= \sum_i \mathbb{E}\left[(Q_i(t+1))^2 - (Q_i(t))^2 \right] \\ &\leq \sum_i \mathbb{E}\left[\alpha_i^2 - 2\alpha_i Q_i(t) + (g_i(A_i(t+1)))^2 + 2Q_i(t)g_i(A_i(t+1)) - 2\alpha_i g_i(A_i(t+1)) \right] \\ &\leq \sum_i \left[D^2 + 2Q_i(t)(\mathbb{E}[g_i(A_i(t+1))] - \alpha_i) \right] \end{aligned} \tag{5.26}$$

The first inequality follows from the evolution of debt queues. The second inequality follows from the boundedness assumption on $g_i(\cdot)$, i.e. $g_i(h) \leq D, \forall h$. Now, we will minimize the RHS of the expression above. We can drop the term

D^2 since it is a constant.

$$\begin{aligned}
&\underset{\pi(t)\in 1,\ldots,N-1}{\operatorname{argmin}} \sum_i Q_i(t)\big(\mathbb{E}[g_i(A_i(t+1))] - \alpha_i\big) \\
&= \underset{\pi(t)\in 1,\ldots,N-1}{\operatorname{argmin}} \sum_i Q_i(t)\mathbb{E}[g_i(A_i(t+1))] \\
&= \underset{j\in 1,\ldots,N-1}{\operatorname{argmin}} \left[\sum_i \Big(Q_i(t)g_i(A_i(t)+1)\Big) + \gamma_j Q_j(t)(g_j(1) - g_j(A_j(t)+1)) \right] \\
&= \underset{j\in 1,\ldots,N-1}{\operatorname{argmax}} \Big[\gamma_j Q_j(t)\big(g_j(A_j(t)+1) - g_j(1)\big) \Big]
\end{aligned} \tag{5.27}$$

The first equality follows since $Q_i(t)\alpha_i$ does not depend on the scheduling decision $\pi(t)$. The second equality follows from the evolution of AoI given $\pi(t) = j$. The third equality follows since the summation term does not depend on the scheduling choice j. This completes the proof.

Chapter 6

Fresh-CSMA: A Distributed Protocol for AoI

In this chapter, we propose Fresh-CSMA to replicate the behavior of centralized scheduling schemes that minimize AoI. In Section 6.1 we discuss our system model and set up the single-hop weighted age minimization problem. In Section 6.2 we introduce the Fresh-CSMA protocol in an idealized setting and provide performance guarantees that show that it can closely match the centralized max-weight scheduling policy both per time-slot and over the entire time horizon. In Section 6.3, we relax some of the assumptions from our idealized model and study the Fresh-CSMA protocol under a more realistic setting. We analyze two keys aspects - the probability of collision and the total time lost due to the backoff timers during which the channel remains idle. In Section 6.4, we consider the recently proposed information freshness metric called Age of Incorrect Information (AoII) and extend our CSMA design to incorporate this metric. In Section 6.5, we provide simulations that support our theoretical results.

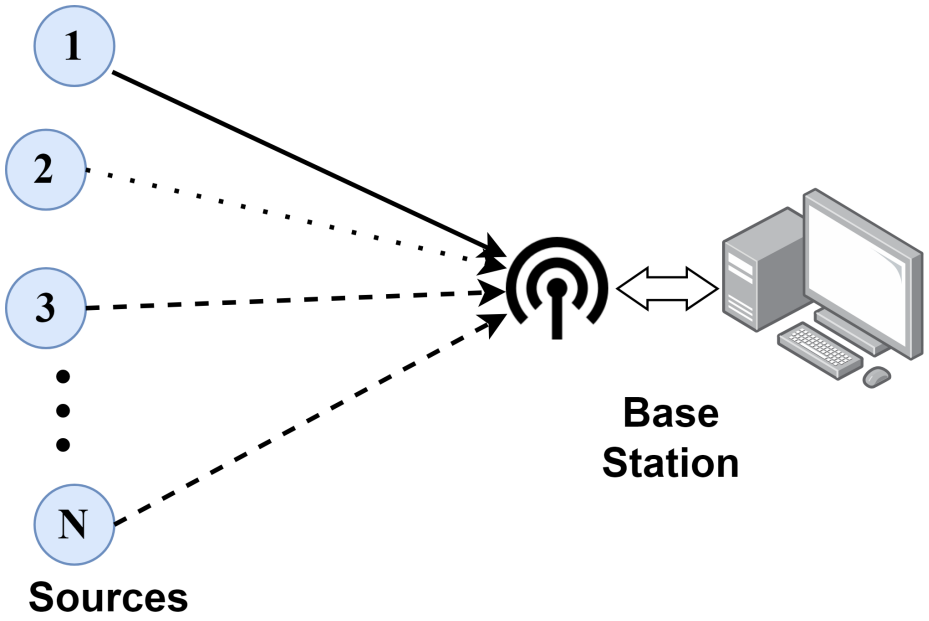


Figure 6-1: Single-hop broadcast network with N sources sending updates to a base station over a shared channel.

6.1 System Model

Consider a single-hop wireless network with N sources generating real-time status updates that need to be sent to a monitoring base station (see Fig. 6-1). We consider a slotted system in which each source takes a single time-slot to transmit an update to the base station. Due to interference, only one of the sources can transmit successfully in any given time-slot. If multiple sources decide to transmit, a collision occurs and the transmitted updates are lost.

For every source i, the age of information at the base station $A_i(t)$ measures the time elapsed since it received a fresh information update from the source. We assume active sources, i.e. in any time-slot, sources can generate fresh updates at will. Let $s(t)$ be the set of sources transmitting in time-slot t. Then, the age of

information $A_i(t)$ evolves as:

$$A_i(t+1) = \begin{cases} 1, & \text{if } s(t) = \{i\}, \\ A_i(t)+1, & \text{otherwise.} \end{cases} \tag{6.1}$$

The metric of interest in this chapter will be average AoI, which is simply the long-term time-average of the AoI process. Specifically,

$$\bar{A}_i \triangleq \limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} A_i(t). \tag{6.2}$$

Our goal in this chapter is to design a *distributed* wireless scheduling policy that minimizes the weighted sum of average AoI across all sources:

$$\underset{\pi}{\text{argmin}} \left(\limsup_{T\to\infty} \left[\frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{N} w_i A_i(t) \right] \right). \tag{6.3}$$

Here, the weights $\{w_1, w_2, ..., w_N\}$ are positive *integers* that denote the relative importance of each source to the overall monitoring or control application.

6.2 Distributed Scheduling Design

Before we introduce our distributed scheduling design, we briefly discuss key results from prior works that have looked at the same problem but from a centralized perspective.

In [15], the authors considered a similar single-hop network setting with the goal of minimizing weighted-sum average AoI, i.e. solving (6.3). First, they considered the class of stationary randomized policies. Each policy within this class is simply a probability distribution over the set of sources and the scheduling decision is sampled from this distribution i.i.d. at the beginning of every time-slot.

They showed that under the optimal stationary randomized policies that solve (6.3), each source i is scheduled with probability

$$\pi_i^* = \frac{\sqrt{w_i}}{\sum_{i=1}^{N} \sqrt{w_i}}, \forall i \in [N]. \tag{6.4}$$

They further showed that the best stationary randomized policies can be at most a factor of two away from the overall optimal policy.

They also proposed a centralized policy motivated by Lyapunov drift arguments called the max-weight policy. This policy, under reliable channels, makes scheduling decisions $\pi^{mw}(t)$ as follows:

$$\pi^{mw}(t) = \underset{j\in[N]}{\operatorname{argmax}}\Big(w_j A_j^2(t)\Big). \tag{6.5}$$

In [126], the authors showed that this policy is at most a factor of two away from optimal using Lyapunov drift arguments. However, unlike stationary randomized policies, this policy turns out to be *nearly optimal* in practice.

The goal of our distributed design is to replicate the decision making and performance guarantees of max-weight policies of the form (6.5).

6.2.A Fresh-CSMA

In general, a CSMA/CA style protocol involves the following steps. First, a node senses the channel to see if it is free. If the node determines the channel to be available, it starts transmitting a packet. Otherwise, if the channel is occupied, it generates a random backoff time and starts a timer counting down from this value. During this period, the node continuously senses the channel and only counts down *when the channel is free.* Once the timer hits zero, the node transmits a frame. Note that the timer can only hit zero when the channel is known to be free. Depending on whether an ACK is received or not from the receiving

station, the node updates its random backoff timer parameters for the next transmission.

The details of how to sample the backoff times, how to update them in case of a collision or re-transmission, and how to implement channel sensing determine the exact flavor of CSMA being implemented. To develop our scheme and provide tractable analysis, we will consider an idealized channel sensing setup that is commonly used in theoretical works addressing CSMA [66, 67, 68, 69]. This involves making the following key assumptions.

Assumption 2. *Backoff timers are implemented in continuous time.*

Assumption 3. *Carrier sensing happens instantly.*

Assumption 4. *There is a discrete slotted system and all nodes start their backoff timers at the beginning of each time-slot.*

Assumption 5. *Backoff timers are implemented with arbitrary precision, and can be made negligible in comparison to the duration of a time-slot.*

Under these assumptions, a version of the classic CSMA protocol that uses exponential backoff timers is described in Alg. 9. We use t to denote the discrete time-slots and τ to denote continuous time within each time-slot. We normalize the time-slot length to be 1 so τ is a continuous timer that increases from 0 to 1 within each time-slot.

As a consequence of our idealized assumptions, note that a packet collision happens only if two nodes choose the exact same backoff times. Since the probability that two exponential random variables take the exact same value is zero, so the probability of packet collisions is also zero in this idealized setup.

The protocol above consists of two key ideas. First, each source i generates

Algorithm 9: Idealized CSMA

```
Input : parameter α > 1
while t ∈ 1, ..., T do
    for i ∈ 1, ..., N do
        Generate a random timer Z_i(t) ~ exp(α).
        while τ < Z_i(t) do
            Stay silent
        end
        if Channel is free then
            Transmit
        end
    end
end
```

a random timer $Z_i(t)$ that is i.i.d. exponentially distributed. Second, the source with the timer that runs out first gets to transmit in the entire time-slot t, i.e.

$$\pi(t) = \underset{i\in[N]}{\operatorname{argmin}} \; Z_i(t). \tag{6.6}$$

Due to Assumption 5, we can scale the backoff timers by a factor δ such that they are negligible in comparison to the slot length, i.e. $\delta Z_i(t) \ll 1, \forall i$ with high probability.

Next, we modify this CSMA protocol to create Fresh-CSMA, described in Alg. 10. This is also an idealized distributed protocol, but with the goal of replicating the behavior of the max-weight scheduling policy (6.5) for AoI minimization. This style of CSMA is motivated by the fast-CSMA protocol proposed in [69].

Note that Fresh-CSMA consists of two key steps. First, each source i generates a random timer $\delta Z_i(t)$ where $Z_i(t)$ is exponentially distributed with the parameter $\alpha^{w_i A_i^2(t)}$. Then, the source with the timer that runs out first gets to transmit in the entire time-slot t, i.e. according to (6.6).

Importantly, each source only requires knowledge of its own AoI and scheduling weight to compute the backoff timer, thus maintaining the distributed nature

Algorithm 10: Idealized Fresh-CSMA

Input : parameters $\alpha > 1, \delta \ll 1$

while $t \in 1, ..., T$ **do**
  for $i \in 1, ..., N$ **do**
    Generate a random timer $Z_i(t) \sim \exp\left(\alpha^{w_i A_i^2(t)}\right)$.
    while $\tau < \delta Z_i(t)$ **do**
      Stay silent
    end
    if *Channel is free* **then**
      Transmit
    end
  end
end

of the protocol. The following lemma describes the structure of scheduling decisions made by this scheduling scheme.

Lemma 9. *For Fresh-CSMA at time-slot t, the probability that source i is scheduled is given by:*

$$r_i(t) = \frac{\alpha^{w_i A_i^2(t)}}{\sum_{j=1}^{N} \alpha^{w_j A_j^2(t)}}. \tag{6.7}$$

Proof. First, note that the scheduling decision at time-slot t is given by (6.6) where $Z_i(t)$ are independent and exponentially distributed with the parameters $\alpha^{w_i A_i^2(t)}$. Let $Z(t) \triangleq \min_{i \in [N]} Z_i(t)$. Then, $Z(t)$ is the minimum of N independent exponential random variables. Thus, it is also exponentially distributed and with the param-

eter $\sum_{j=1}^{N} \alpha^{w_j A_j^2(t)}$. Let $\lambda_i \triangleq \alpha^{w_i A_i^2(t)}$. We are interested in calculating the probability

$$\begin{aligned} r_i(t) &= \mathbb{P}\big(\pi(t) = i\big) = \mathbb{P}\big(Z(t) = Z_i(t)\big) \\ &= \int_0^\infty f_{Z_i(t)}(x) \prod_{k \neq i} \mathbb{P}\big(Z_k(t) > x\big) dx \\ &= \int_0^\infty \lambda_i e^{-\lambda_i x} \prod_{k \neq i} e^{-\lambda_k x} dx \\ &= \frac{\lambda_i}{\sum_{k=1}^{N} \lambda_k}. \end{aligned} \tag{6.8}$$

This completes the proof. □

Using Lemma 1, we next show that if the parameter α is set to be large enough, then in any particular time-slot, Fresh-CSMA will make the same scheduling decision as the max-weight policy with high probability.

Theorem 15. *Given any $\delta \in (0,1)$ and $A_1(t), ..., A_N(t)$; if we set $\alpha \geq (N-1)^{\frac{1-\delta}{\delta}}$, then the following holds*

$$\mathbb{P}\Big(\pi^{Fresh-CSMA}(t) = \pi^{mw}(t)\Big) \geq 1 - \delta. \tag{6.9}$$

Here, $\pi^{mw}(t)$ is the max-weight scheduling decision given by (6.5)*, while $\pi^{Fresh-CSMA}(t)$ is the scheduling decision made by Fresh-CSMA.*

Proof. We divide the proof into two parts.

Case 1: The expression $\underset{j \in [N]}{\operatorname{argmax}}\Big(w_j A_j^2(t)\Big)$ has a unique maximum. Let this maximum be the source 1 without loss of generality. Then, the max-weight decision is to schedule source 1. Since we have assumed all the weights w_i to be positive integers and the AoIs $A_i(t)$ are integers by definition, so the quantities

$w_i A_i^2(t)$ are also positive integers and the following must hold:

$$w_1 A_1^2(t) - w_i A_i^2(t) \geq 1, \forall i \neq 1. \tag{6.10}$$

Note that in the special case of $w_i = 1, \forall i$, (6.10) holds when the AoIs across the different sources are unique.

Now, applying Lemma 1, we can calculate the following probability

$$\begin{aligned}
\mathbb{P}\left(\pi^{Fresh-CSMA}(t) = 1\right) &= \frac{\alpha^{w_1 A_1^2(t)}}{\sum_{j=1}^{N} \alpha^{w_j A_j^2(t)}} \\
&= \frac{1}{1 + \sum_{i=2}^{N} \alpha^{w_i A_i^2(t) - w_1 A_1^2(t)}} \\
&\geq \frac{1}{1 + (N-1)\alpha^{-1}} \\
&\geq 1 - \delta.
\end{aligned} \tag{6.11}$$

The first inequality follows by using (6.10) while the second inequality follows due to the fact that $\alpha \geq (N-1)\frac{1-\delta}{\delta}$.

Case 2: The expression $\underset{j \in [N]}{\text{argmax}}\left(w_j A_j^2(t)\right)$ has multiple maxima. Suppose that the set of maxima is given by the nodes $\{1, ..., k\}$. Then, the max-weight policy will choose one of these sources to be scheduled. We want to lower bound the probability that Fresh-CSMA chooses a node from within this set. To do so, we first make a similar observation as in the case above.

$$w_1 A_1^2(t) - w_j A_j^2(t) \geq 1, \forall j = k+1, ..., N. \tag{6.12}$$

Using Lemma 1, we calculate the probability of interest

$$\begin{aligned}\mathbb{P}\left(\pi^{Fresh-CSMA}(t) \in \{1,...,k\}\right) &= \frac{k\alpha^{w_1 A_1^2(t)}}{\sum\limits_{j=1}^{N} \alpha^{w_j A_j^2(t)}} \\ &= \frac{k}{k + \sum_{i=k+1}^{N} \alpha^{w_i A_i^2(t) - w_1 A_1^2(t)}} \\ &\geq \frac{1}{1 + (N-k)\alpha^{-1}} \\ &\geq 1 - \delta.\end{aligned} \tag{6.13}$$

As before, the first inequality follows by using (6.12) while the second inequality follows due to the fact that $\alpha \geq (N-1)\frac{1-\delta}{\delta}$. This completes the proof. □

Next, we show that our idealized Fresh-CSMA protocol has the same theoretical long-term performance guarantees as the max-weight policy.

Theorem 16. *Given any set of integer weights $w_1,...,w_N$, if we set $\alpha > \left(\frac{(N-1)\sum_{i=1}^{N}\sqrt{w_i}}{\min_j \sqrt{w_j}}\right)$, then the following holds:*

$$\frac{\sum_{i=1}^{N} w_i \bar{A}_i^{csma}}{\sum_{i=1}^{N} w_i \bar{A}_i^{opt}} \leq 2. \tag{6.14}$$

Here, $\bar{A}_i^{csma}$ is the average AoI for source i under the Fresh-CSMA policy while $\bar{A}_i^{opt}$ is the average AoI of source i under an optimal policy π^{opt} that solves the age minimization problem (6.3).

Proof. Consider the linear Lyapunov function as defined below:

$$L(t) \triangleq \sum_{i=1}^{N} \sqrt{w_i} A_i(t). \tag{6.15}$$

We can define the one-slot Lyapunov drift $\Delta(t) \triangleq L(t+1) - L(t)$. The main challenge in proving the performance bound above, is to first show an intermediate result relating the Lyapunov drift of the Fresh-CSMA policy to that of the optimal stationary randomized policy described by (6.4).

Lemma 10. *Consider any $\boldsymbol{At} = \{A_1, ..., A_N(t)\}$. Let $\Delta^{csma}(t)$ be the one-slot Lyapunov drift of the Fresh-CSMA policy and $\Delta^{sr}(t)$ be the one-slot Lyapunov drift of the optimal stationary randomized policy. Then, the following holds:*

$$\mathbb{E}\left[\Delta^{csma}(t)\middle|\boldsymbol{At}\right] \leq \mathbb{E}\left[\Delta^{sr}(t)\middle|\boldsymbol{At}\right], \forall \boldsymbol{At}. \tag{6.16}$$

Proof. We first calculate an expression for the drift of the Fresh-CSMA policy. Recall that $r_j(t) \triangleq \mathbb{P}\left(\pi^{AoI-CMSA}(t) = j\right)$ and is given by (6.7).

$$\begin{aligned}
&\mathbb{E}\left[\Delta^{csma}(t)\middle|\boldsymbol{At}\right] \\
&= \sum_{j=1}^{N} r_j(t)\left(\sqrt{w_j} + \sum_{i\neq j}\sqrt{w_i}(A_i(t)+1)\right) - \sum_{j=1}^{N}\sqrt{w_j}A_j(t) \\
&= \sum_{j=1}^{N}\sqrt{w_j} - \sum_{j=1}^{N} r_j(t)\sqrt{w_j}A_j(t).
\end{aligned} \tag{6.17}$$

Repeating the above steps for the optimal stationary randomized policy, we get:

$$\mathbb{E}\left[\Delta^{sr}(t)\middle|\boldsymbol{At}\right] = \sum_{j=1}^{N}\sqrt{w_j} - \sum_{j=1}^{N}\pi_j^*\sqrt{w_j}A_j(t). \tag{6.18}$$

Recall that π_j^* are scheduling probabilities for the optimal stationary randomized policy, given by (6.4).

Consider the difference between (6.17) and (6.18)

$$\begin{aligned}
&\mathbb{E}\left[\Delta^{csma}(t)\middle|\boldsymbol{At}\right] - \mathbb{E}\left[\Delta^{sr}(t)\middle|\boldsymbol{At}\right] \\
&= \sum_{j=1}^{N} \sqrt{w_j} A_j(t)\left(\pi_j^* - r_j(t)\right) \\
&= \sum_{j=1}^{N} \sqrt{w_j} A_j(t)\left(\frac{\sqrt{w_j}}{\sum_{i=1}^{N}\sqrt{w_i}} - \frac{\alpha^{w_j A_j^2(t)}}{\sum_{i=1}^{N}\alpha^{w_i A_i^2(t)}}\right)
\end{aligned} \tag{6.19}$$

Note that we are only interested in the sign of (6.19), so we can instead look at

$$\begin{aligned}
&\sum_{j=1}^{N} \sqrt{w_j} A_j(t)\left(\sqrt{w_j}\left(\sum_{i=1}^{N}\alpha^{w_i A_i^2(t)}\right) - \alpha^{w_j A_j^2(t)}\left(\sum_{i=1}^{N}\sqrt{w_i}\right)\right) \\
&= \sum_{j=1}^{N} \alpha^{w_j A_j^2(t)}\left(\sum_{i=1}^{N} w_i A_i(t) - \sqrt{w_j} A_j(t)\sum_{i=1}^{N}\sqrt{w_i}\right) \\
&= \sum_{j=1}^{N} \alpha^{w_j A_j^2(t)}\left(\sum_{i=1}^{N}\sqrt{w_i}\left(\sqrt{w_i} A_i(t) - \sqrt{w_j} A_j(t)\right)\right)
\end{aligned} \tag{6.20}$$

Consider without loss of generality that sources are numbered such that $\sqrt{w_1}A_1(t) \geq \sqrt{w_2}A_2(t) \geq \ldots \geq \sqrt{w_N}A_N(t)$. This automatically implies that source 1 has the largest value of $\sqrt{w_j}A_j(t)$ among all sources.

Case 1: First we consider the case when Source 1 is the unique max-weight scheduling decision, i.e. $\pi^{mw}(t) = \underset{j\in[N]}{\operatorname{argmax}}\left(w_j A_j^2(t)\right) = 1$. Since we have assumed weights $w_i \in \mathbb{Z}^+$ and AoIs $A_i(t)$ are also positive integers, the above equation implies that

$$w_1 A_1^2(t) - w_i A_i^2(t) \geq 1, \forall i \neq 1. \tag{6.21}$$

This is because $w_i A_i^2(t) \in \mathbb{Z}^+$ for all sources i. Further, note that $f(x) = \sqrt{x}$ is Lipschitz for $x \in [1,\infty)$ with the Lipschitz constant 0.5. Applying this fact to (6.21), we get

$$\sqrt{w_1}A_1(t) - \sqrt{w_i}A_i(t) \geq 0.5, \forall i \neq 1. \tag{6.22}$$

Next, we define the following quantities

$$\gamma_j \triangleq \left(\sum_{i=1}^{N} \sqrt{w_i}\left(\sqrt{w_i}A_i(t) - \sqrt{w_j}A_j(t)\right)\right). \tag{6.23}$$

Using (6.22), it is easy to see that $\gamma_1 \leq -0.5 \sum_{i=1}^{N} \sqrt{w_i}$. We will bound the rest of the values γ_i in comparison to γ_1.

$$\begin{aligned}
\gamma_j &= \left(\sum_{i=1}^{N} \sqrt{w_i}\left(\sqrt{w_i}A_i(t) - \sqrt{w_j}A_j(t)\right)\right) \\
&\leq \Bigg(\sqrt{w_1}(\sqrt{w_1}A_1(t) - \sqrt{w_j}A_j(t)) \\
&+ \sum_{i=2}^{j-1} \sqrt{w_i}\left(\sqrt{w_1}A_1(t) - \sqrt{w_j}A_j(t)\right) \\
&+ \sum_{i=j}^{N} \sqrt{w_i}\left(\sqrt{w_i}A_i(t) - \sqrt{w_j}A_j(t)\right)\Bigg) \\
&\leq \frac{\sum_{i=1}^{N}\sqrt{w_i}}{\sqrt{w_j}}\Bigg(\sqrt{w_j}\left(\sqrt{w_1}A_1(t) - \sqrt{w_j}A_j(t)\right) \\
&+ \sum_{i\neq j} \sqrt{w_i}\left(\sqrt{w_1}A_1(t) - \sqrt{w_i}A_i(t)\right)\Bigg) \\
&\leq \frac{\sum_{i=1}^{N}\sqrt{w_i}}{\min_j \sqrt{w_j}}\left|\gamma_1\right|, \forall j \neq 1.
\end{aligned} \tag{6.24}$$

Using this result, (6.21) and the definition of γ_i we get

$$\begin{aligned}
&\sum_{j=1}^{N} \alpha^{w_j A_j^2(t)}\left(\sum_{i=1}^{N} \sqrt{w_i}\left(\sqrt{w_i}A_i(t) - \sqrt{w_j}A_j(t)\right)\right) \\
&\leq \alpha^{w_1 A_1^2(t)}\left(\gamma_1\right) + \sum_{i=2}^{N} \alpha^{w_2 A_2^2(t)} \frac{\sum_{i=1}^{N}\sqrt{w_i}}{\min_j \sqrt{w_j}}\left|\gamma_1\right| \\
&\leq \alpha^{w_1 A_1^2(t)}\left|\gamma_1\right|\left(-1 + \alpha^{-1}\frac{(N-1)\sum_{i=1}^{N}\sqrt{w_i}}{\min_j \sqrt{w_j}}\right).
\end{aligned} \tag{6.25}$$

Clearly, choosing

$$\alpha > \left(\frac{(N-1)\sum_{i=1}^{N} \sqrt{w_i}}{\min_j \sqrt{w_j}} \right) \tag{6.26}$$

is sufficient to guarantee that (6.25) is negative and hence (6.19) is negative. Thus, for this choice of α, we observe that

$$\mathbb{E}\left[\Delta^{CSMA}(t) \middle| \boldsymbol{At} \right] \leq \mathbb{E}\left[\Delta^{SR}(t) \middle| \boldsymbol{At} \right], \forall \boldsymbol{At}. \tag{6.27}$$

Case 2: Sources $1, ..., k$ are all solutions to the following maximization

$$\pi^{mw}(t) = \underset{j \in [N]}{\operatorname{argmax}} \left(w_j A_j^2(t) \right) \in \{1, ..., k\}. \tag{6.28}$$

We can repeat the exact same analysis as Case 1, but starting with

$$w_1 A_1^2(t) - w_i A_i^2(t) \geq 1, \forall i \in \{k+1, ..., N\}. \tag{6.29}$$

This is because $w_i A_i^2(t) \in \mathbb{Z}^+$ for all sources i. Further, note that $f(x) = \sqrt{x}$ is Lipschitz for $x \in [1, \infty)$ with the Lipschitz constant 0.5. Applying this fact to (6.21), we get

$$\sqrt{w_1} A_1(t) - \sqrt{w_i} A_i(t) \geq 0.5, \forall i \in \{k+1, ..., N\}. \tag{6.30}$$

Using these inequalities above, we obtain

$$\gamma_j \leq \frac{\sum_{i=1}^{N} \sqrt{w_i}}{\min_j \sqrt{w_j}} \left| \gamma_1 \right|, \forall j \in \{k+1, ..., N\}. \tag{6.31}$$

Finally putting all of the inequalities together, we get

$$\begin{aligned}
&\sum_{j=1}^{N} \alpha^{w_j A_j^2(t)} \left(\sum_{i=1}^{N} \sqrt{w_i} \left(\sqrt{w_i} A_i(t) - \sqrt{w_j} A_j(t) \right) \right) \\
&\leq k \alpha^{w_1 A_1^2(t)} (\gamma_1) + \sum_{i=k+1}^{N} \alpha^{w_{k+1} A_{k+1}^2(t)} \frac{\sum_{i=1}^{N} \sqrt{w_i}}{\min_j \sqrt{w_j}} |\gamma_1| \\
&\leq \alpha^{w_1 A_1^2(t)} |\gamma_1| \left(-k + \alpha^{-1} \frac{(N-k) \sum_{i=1}^{N} \sqrt{w_i}}{\min_j \sqrt{w_j}} \right).
\end{aligned} \tag{6.32}$$

Again, choosing

$$\alpha > \left(\frac{(N-1) \sum_{i=1}^{N} \sqrt{w_i}}{\min_j \sqrt{w_j}} \right) \tag{6.33}$$

is sufficient to guarantee that (6.32) is negative and hence (6.19) is negative. This completes the proof of Lemma 10. □

Next, we proceed to the proof of Theorem 16. The one-slot drift for the optimal stationary randomized policy is given by -

$$\begin{aligned}
&\mathbb{E}\left[\Delta^{sr}(t) \middle| \boldsymbol{At} \right] \\
&= \sum_{j=1}^{N} \pi_j^* \left(\sqrt{w_j} + \sum_{i \neq j} \sqrt{w_i} (A_i(t) + 1) \right) - \sum_{j=1}^{N} \sqrt{w_j} A_j(t) \\
&= \sum_{j=1}^{N} \sqrt{w_j} - \sum_{j=1}^{N} \pi_j^* \sqrt{w_j} A_j(t).
\end{aligned} \tag{6.34}$$

Putting together (6.16) and (6.34), we get

$$\mathbb{E}\left[\Delta^{csma}(t) \middle| \boldsymbol{At} \right] \leq \sum_{j=1}^{N} \sqrt{w_j} - \sum_{j=1}^{N} \pi_j^* \sqrt{w_j} A_j(t). \tag{6.35}$$

Summing (6.35) for $t = 1, ..., T$ and taking expectation, we get

$$\mathbb{E}\Big[L(T+1) - L(1)\Big] \leq \\ T\sum_{j=1}^{N}\sqrt{w_j} - \sum_{j=1}^{N}\mathbb{E}\Big[\sum_{t=1}^{T}\sqrt{w_j}\pi_j^* A_j(t)\Big]. \tag{6.36}$$

Substituting the expression for π_j^* from (6.4), rearranging and dividing by T, we get

$$\frac{1}{T\big(\sum_{j=1}^{N}\sqrt{w_j}\big)}\mathbb{E}\Big[\sum_{t=1}^{T}\sum_{j=1}^{N} w_j A_j(t)\Big] \leq \\ \sum_{j=1}^{N}\sqrt{w_j} - \frac{1}{T}\mathbb{E}\Big[L(T+1) - L(1)\Big]. \tag{6.37}$$

Since $L(T+1) \geq 0$ and $\frac{w_j}{\pi_j^*} = \sqrt{w_j}(\sum_{j=1}^{N}\sqrt{w_j})$, we can further simplify the equaiton above to

$$\frac{1}{T}\mathbb{E}\Big[\sum_{t=1}^{T}\sum_{j=1}^{N} w_j A_j(t)\Big] \leq \sum_{j=1}^{N}\frac{w_j}{\pi_j^*} + \frac{\mathbb{E}[L(1)]}{T}. \tag{6.38}$$

Now, we observe that the average AoI of a source j under the optimal stationary randomized policy is given by $\bar{A}_j^{sr} = \frac{1}{\pi_j^*}$, as shown in [15]. Using this fact and taking limsup as T goes to infinity in the equation above, we get

$$\sum_{j=1}^{N} w_j \bar{A}_j^{csma}(t) \leq \sum_{j=1}^{N} w_j \bar{A}_j^{sr}. \tag{6.39}$$

We also know from [15] that stationary randomized policies can be at most a factor of two away from optimal. Thus, we get

$$\frac{\sum_{i=1}^{N} w_i \bar{A}_i^{csma}}{\sum_{i=1}^{N} w_i \bar{A}_i^{opt}} \leq \frac{\sum_{i=1}^{N} w_i \bar{A}_i^{sr}}{\sum_{i=1}^{N} w_i \bar{A}_i^{opt}} \leq 2. \tag{6.40}$$

This completes the proof.

□

The factor of two optimality guarantee that we have derived is the same performance guarantee as the one shown for the max-weight policy in [126] and better than the factor of four bound derived in [15, 53]. Viewing Theorems 15 and 16 together, we conclude that the idealized Fresh-CSMA policy can replicate the behavior of the max-weight policy, both at each time-slot with high probability and in terms of long-term average AoI over the entire time-horizon if $\alpha > \max\left((N-1)\frac{1-\delta}{\delta}, \frac{(N-1)\sum_{i=1}^{N}\sqrt{w_i}}{\min_j \sqrt{w_j}}\right)$. In Section 6.5, we will see via simulations that this holds true even for small values of α, i.e. α doesn't need to be very large for Fresh-CSMA to be able to mimic the max-weight policy.

6.3 Near-Realistic Multiple Access Model

Until now, we have looked at distributed multiple access with idealized assumptions. In this section, we discuss the Fresh-CSMA protocol under a more realistic version of multiple access.

Our discussion is based on the IEEE 802.11 standard for wireless LAN [127]. This standard defines a distributed coordination function (DCF) for sharing access to the wireless medium based on a CSMA/CA style protocol. The 802.11 standard divides time into the basic units of mini-slots, where each mini-slot is the duration of time needed by a source to detect packet transmission from any another source, i.e. perform channel sensing. A typical value for the mini-slot duration in IEEE 802.11 g/n/ac protocols is $9\mu s$.

Fig. 6-2 describes the key elements of our near-realistic model. First, we consider that the backoff timers for source i at frame t, denoted by $D_i(t)$, can only run in multiples of minislot durations. This relaxes Assumptions 2 and 3, since the timers are now discrete, with finite precision and limited by the amount of time required to do channel sensing. Further, since timers are no longer contin-

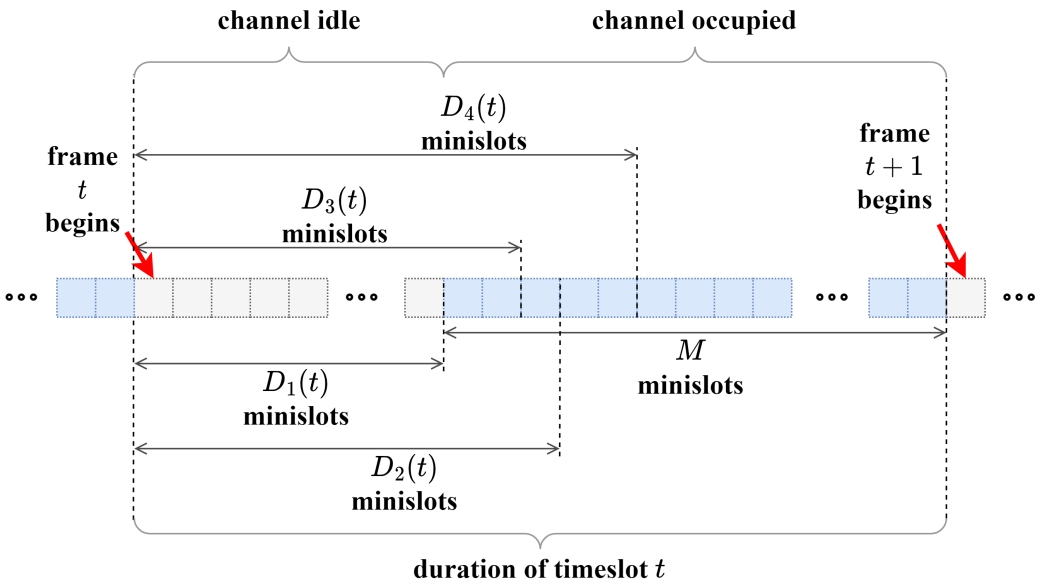


Figure 6-2: Events within frame t in the near-realistic multiple access model. Four sources choose backoff timers $D_1(t), ..., D_4(t)$. Source 1's timer runs out first, after which it transmits its update for M minislots.

uous, the probability of collision is also non-zero and has an effect on the average age.

As before, we assume that backoff timers for each source begin at the beginning of each frame and the source/sources whose timers run out first get to transmit an entire application layer update. Thus, we have not relaxed Assumption 4 in this model. We assume that a mini-slot takes $\frac{1}{M}$ units of time. So, transmitting an entire update takes M mini-slots. We will discuss in Section 6.5 that for typical values of update sizes and transmission rates M tends to be large (around 10000).

Finally, we also relax Assumption 5 and consider the amount of time that the channel remains idle in each frame. Consider the example frame depicted in Fig. 6-2 with four backoff timers $D_1(t), ..., D_4(t)$. The timer of source 1, denoted by $D_1(t)$, runs out first. Then, source 1 transmits its update for M minislots. Thus the total duration of frame t is $D_1(t) + M$ minislots or alternatively $1 + \frac{D_1(t)}{M}$. However, for the first $D_1(t)$ minislots in frame t, the channel remained idle. We term this

the *backoff overhead*. In general, given $D(t) \triangleq \min_{i\in[N]} D_i(t)$, frame t takes $M + D(t)$ mini-slots or alternatively $1 + \frac{D(t)}{M}$ units of time to complete. Compared to the idealized setting, the time $\frac{D(t)}{M}$ is the backoff overhead of the protocol, since it is the time that the channel must remain idle before any source starts transmitting. We take this overhead into account while calculating AoIs.

Our new model thus relaxes Assumptions 2, 3 and 5, while allowing us to study the effect of collision probabilities and backoff overheads on the AoI. For this near-realistic multiple access model, we provide a modified version of the Fresh-CSMA protocol below. As before, We use t to denote the discrete frames and τ to denote time within each frame, denoting the number of minislots that have passed within this frame.

Algorithm 11: Near-Realistic Fresh-CSMA

Input : parameters $\alpha > 1, \beta > 1, B \in \mathbb{Z}^+$

while $t \in 1, ..., T$ **do**
  for $i \in 1, ..., N$ **do**
    Generate a random variable $Z_i(t) \sim \exp\left(\alpha^{w_i A_i^2(t)}\right)$.
    Map it to a non-negative integer timer,
    $D_i(t) = \max\left(B + \lfloor \log_\beta \left(Z_i(t)\right) \rfloor, 0\right)$.
    while $\tau < D_i(t)$ **do**
      Stay silent
    end
    if *Channel is free* **then**
      Transmit
    end
  end
end

The key difference between the protocol described in Alg. 11 and Alg. 10 is mapping the continuous random variables $Z_i(t)$ to the discrete variables $D_i(t) \in \{0 \cup \mathbb{Z}^+\}$, which denote the number of mini-slots source i should count down before transmitting.

6.3.A Collisions

Note that when using the protocol above, a packet collision happens if two sources i and j choose the same discrete backoff timers $D_i(t)$ and $D_j(t)$ at frame t while also being the first timers to count down to zero, i.e. $\underset{j\in[N]}{\operatorname{argmin}}\Big(D_j(t)\Big)$ is not unique. When a collision happens, we assume that the base station fails to receive an update from any of the transmitting sources and the entire frame is wasted. The following theorem analyzes the probability of the event that two sources i and j choose different backoff timers at time t.

Theorem 17. *Let the AoIs at time t be given by $A_1(t), ..., A_N(t)$ and $\lambda_i \triangleq \alpha^{w_i A_i^2(t)}$. Then probability that any two sources i and j choose different backoff timers $D_i(t)$ and $D_j(t)$ can be lower bounded as follows*

$$\mathbb{P}\Big(D_i(t) \neq D_j(t)\Big) \geq \psi\big(B,\beta,\lambda_i,\lambda_j\big) + \psi\big(B,\beta,\lambda_j,\lambda_i\big) \tag{6.41}$$

where, the function $\psi(\cdot)$ is given by:

$$\psi\big(B,\beta,\lambda_i,\lambda_j\big) = \frac{\lambda_i e^{-\beta^{-B}(\lambda_i+\beta\lambda_j)}}{\lambda_i+\beta\lambda_j} + \Big(e^{\lambda_i\beta^{-B}} - 1\Big)e^{-\beta^{-B}(\lambda_i+\beta\lambda_j)}.$$

Proof. Consider the exponential random variables generated by the two sources $Z_i(t) \sim \exp(\lambda_i)$ and $Z_j(t) \sim \exp(\lambda_j)$, where $\lambda_i = \alpha^{w_i A_i^2(t)}, \forall i$. Suppose that the following inequality holds:

$$Z_j(t) > \begin{cases} \beta Z_i(t), & \text{if } Z_i(t) > \beta^{-B} \\ \beta^{-B+1}, & \text{otherwise.} \end{cases} \tag{6.42}$$

Then it is easy to see that

$$\max\left(B+\left\lfloor \log_\beta\left(Z_i(t)\right)\right\rfloor,0\right) < \max\left(B+\left\lfloor \log_\beta\left(Z_j(t)\right)\right\rfloor,0\right)$$

which in turn implies that $D_i(t) < D_j(t)$. Switching i and j in the inequality (6.42), we get $D_i(t) > D_j(t)$. Note that the two events are disjoint.

Thus, we can lower-bound our probability of interest as follows:

$$\begin{aligned}\mathbb{P}(D_i(t)\neq D_j(t)) \geq\ & \mathbb{P}\Big(Z_j(t) > \min\{\beta Z_i(t), \beta^{-B+1}\}\Big)\\ &+\mathbb{P}\Big(Z_i(t) > \min\{\beta Z_j(t), \beta^{-B+1}\}\Big)\end{aligned} \tag{6.43}$$

Simplifying the first term on the RHS, we get

$$\begin{aligned}&\mathbb{P}\Big(Z_j(t) > \min\{\beta Z_i(t), \beta^{-B+1}\}\Big)\\ =&\int_0^{\beta^{-B}} \lambda_i e^{-\lambda_i x} e^{-\lambda_j \beta^{-B+1}} dx + \int_{\beta^{-B}}^{\infty} \lambda_i e^{-\lambda_i x} e^{-\lambda_j \beta x} dx\\ =&\frac{\lambda_i e^{-\beta^{-B}(\lambda_i+\beta\lambda_j)}}{\lambda_i+\beta\lambda_j} + \left(e^{\lambda_i\beta^{-B}}-1\right)e^{-\beta^{-B}(\lambda_i+\beta\lambda_j)} \triangleq \psi\left(B,\beta,\lambda_i,\lambda_j\right).\end{aligned} \tag{6.44}$$

By the same argument, the second term on the RHS of (6.43) is equal to $\psi\left(B,\beta,\lambda_j,\lambda_i\right)$. Thus, we get

$$\mathbb{P}(D_i(t)\neq D_j(t)) \geq \psi\left(B,\beta,\lambda_i,\lambda_j\right) + \psi\left(B,\beta,\lambda_j,\lambda_i\right). \tag{6.45}$$

This completes the proof. □

As a corollary of this proof, note that

$$\begin{aligned}&\frac{\partial}{\partial B}\psi\left(B,\beta,\lambda_i,\lambda_j\right)\\ &\quad= \lambda_j\beta^{-B+1}\log(\beta)\left(e^{\lambda_i\beta^{-B}}-1\right)e^{-\beta^{-B}(\lambda_i+\beta\lambda_j)} \geq 0.\end{aligned} \tag{6.46}$$

The last inequality follows since $\beta > 1$ and $\lambda_i > 0$. Thus, the probability that two sources choose different backoff timers *increases with the parameter B*. In the limit as $B \to \infty$, we get

$$\lim_{B\to\infty} \mathbb{P}(D_i(t) \neq D_j(t)) \geq \frac{\lambda_i}{\lambda_i + \beta\lambda_j} + \frac{\lambda_j}{\lambda_j + \beta\lambda_i}. \tag{6.47}$$

Thus, when B is large, the probability that two sources occupy different minislots *decreases with* β. Putting the two observations together, we should choose a large value for B and a small value for β (close to 1) to reduce collisions. However, for finite B, (6.47) does not hold and the value of β cannot be too small, since in that case, all the discrete timers will map to the first minislot leading to collisions in almost every frame.

In the analysis above, we used the probability of two sources occupying different backoff timers as a proxy for analyzing the collision probability directly, since a tight bound for the actual collision probability is too involved to compute. In Section 6.5, we will see via simulations how the actual collision probability varies with the parameters B and β.

6.3.B Backoff Timer Overhead

Next, we analyze the overhead of the backoff timers in the Fresh-CSMA protocol in the near-realistic model. This is unlike the idealized setting where we ignored the time taken by the backoff timers to count down, during which the channel remains idle.

Recall that the quantity $\frac{D(t)}{M}$ is what we defined as the backoff overhead of a protocol at time t, since it is the time that the channel must remain idle before any source starts transmitting. The following theorem provides an upper-bound on the expected backoff overhead in frame t.

Theorem 18. *Let the AoIs at time t be $A_1(t), ..., A_N(t)$ and $\lambda_i \triangleq \alpha^{w_i A_i^2(t)}$. Then, the expected idle-time of the Fresh-CSMA protocol at time t can be upper-bounded by*

$$\frac{1}{M}\mathbb{E}\big[D(t)\big] \leq \frac{1}{M} + \frac{\Gamma\big(0, \lambda\beta^{-B}\big)}{M\log(\beta)}. \tag{6.48}$$

Here $\Gamma(\cdot,\cdot)$ is the upper incomplete gamma function and $\lambda \triangleq \sum_{i\in[N]} \lambda_i$.

Proof. Let $Z(t) = \min_{i\in[N]} Z_i(t)$. Since $Z(t)$ is the minimum of N independent exponential random variables, it is also exponentially distributed with the parameter $\lambda = \sum_{i\in[N]} \lambda_i$. Using this, we provide an upper bound for $\mathbb{E}[D(t)]$ below.

$$\begin{aligned}
\mathbb{E}\big[D(t)\big] &= \mathbb{E}\left[\min_{i\in[N]}\big(D_i(t)\big)\right] \\
&= \mathbb{E}\left[\min_{i\in[N]}\left(\max\big(B + \lfloor \log_\beta\big(Z_i(t)\big)\rfloor, 0\big)\right)\right] \\
&\leq B + 1 + \mathbb{E}\left[\max\left(\log_\beta\big(\min_{i\in[N]} Z_i(t)\big), -B\right)\right] \\
&\leq B + 1 + \mathbb{E}\left[\max\left(\log_\beta\big(Z(t)\big), -B\right)\right] \\
&\leq 1 + \frac{\Gamma\big(0, \lambda\beta^{-B}\big)}{\log(\beta)}.
\end{aligned} \tag{6.49}$$

The last inequality follows from the fact that $\mathbb{E}\big[\max(\log(Z(t)), -B)\big] = -B + \Gamma\big(0, \lambda\beta^{-B}\big)$, where $\Gamma(\cdot,\cdot)$ is the upper incomplete gamma function given by

$$\Gamma(s,x) = \int_x^\infty t^{s-1}e^{-t}dt. \tag{6.50}$$

□

Note that the function $\Gamma(0,x)$ is given by

$$\Gamma(0,x) = \int_x^{\infty} t^{-1}e^{-t}dt. \tag{6.51}$$

From (6.51), note that $\Gamma(0,x)$ is decreasing in x. Thus, for a fixed value of β, the expected idle time *increases as B increases*. The exact dependence on β is more tricky to evaluate, so we again consider the case of large B as an alternative. First, we make the following observation for the gamma function of large values of B

$$\Gamma(0,\lambda\beta^{-B}) \approx B\log(\beta) - log(\lambda) - \gamma, \tag{6.52}$$

where $\gamma \approx 0.58$ is the Euler-Mascheroni constant. Using this, it is easy to see that for large values of B, the expected idle time upper bound is approximately equal to $1 + B - \log_\beta(\lambda)$. Thus, the backoff overhead also *increases with* β, given a fixed large value of B.

Together this implies that we need to choose a relatively small value of B and a small value of β (close to 1) to reduce idle time. Importantly, there is a trade-off between the collision probability and the backoff overhead depending on the choice of the parameter B. A larger value of B reduces the probability of collision but at the cost of higher backoff overhead.

Theorem 18 also allows us to compute an approximate upper-bound for the average idle time over the entire horizon. Suppose the average AoI of source i under the Fresh-CSMA policy in the near-realistic model is denoted by $\bar{A}_i$. Using the average AoIs, we define the following quantity: $\bar{\lambda} \triangleq \sum_{i=1}^{N} \alpha^{w_i\bar{A}_i^2}$. Then, an approximate upper-bound of the average backoff overhead per frame over the entire time-horizon $\bar{D}^{ub}$ can be obtained as follows:

$$\bar{D}^{ub} \approx \frac{1}{M} + \frac{\Gamma\left(0,\bar{\lambda}\beta^{-B}\right)}{M\log(\beta)}. \tag{6.53}$$

In Section 6.5, we will see via simulations that this is a good bound for the average backoff overhead in the near-realistic setting.

6.4 Going Beyond Age of Information

While Age of Information has been used as a proxy for optimizing monitoring and control costs in real-time settings over the past decade, a recent line of work starting with [128] has proposed a new and more general metric to measure the impact of stale information on underlying real-time monitoring tasks. This metric is called the Age of Incorrect Information (AoII), and it has been used to study the monitoring of Markov sources in various kinds of settings [128, 129, 130, 131].

The key idea behind the AoII metric is that takes into account the actual error or distortion between the estimate of the process being monitored at the remote monitor and the actual value of the process at present. More precisely, suppose that $X(t)$ represents the state of the process that needs to be monitored and let $\hat{X}(t)$ be the estimate of the process at time t at the remote monitor. Let the function $g(X(t), \hat{X}(t))$ measure the distortion or error between the actual process and its remote estimate and let $f(\cdot)$ be a monotone increasing function. Let $V(t)$ represent the most recent time instant up to the current time t at which this distortion was zero. The AoII is then defined as

$$AoII(t) \triangleq f\big(t - V(t)\big)g(X(t), \hat{X}(t)). \tag{6.54}$$

Note that if the actual process and its estimate stay the same for some time despite no new updates being delivered to the monitor, the AoII remains zero while the AoI keeps increasing. Thus, the AoII can be viewed as a more accurate metric to measure information uncertainty at the monitor.

Now, consider a setting with N sources, sending updates to the base station,

where only one source can talk to the base station at any given time. Unlike the setting we have analyzed until now, we will now look at minimizing the sum of AoIIs instead of weighted AoIs. Each source is tracking a process $X_i(t), i \in \{1, ..., N\}$ and the base station maintains estimates for each process $\hat{X}_i(t), i \in \{1, ..., N\}$. Using these estimates and 6.54, we can compute the AoIIs for each source, denoted by $AoII_i(t), i \in \{1, ..., N\}$. We want to design a scheduling policy that minimizes the long term time-average of the AoIIs, i.e.

$$\underset{\pi}{\operatorname{argmin}}\left(\limsup_{T\to\infty} \left[\frac{1}{T}\frac{1}{N} \sum_{t=1}^{T} \sum_{i=1}^{N} AoII_i(t) \right] \right). \tag{6.55}$$

A good candidate policy to solve this problem would be to schedule the source with the highest AoII at each time-slot.

$$\pi^{max-AoII}(t) = \underset{j\in[N]}{\operatorname{argmax}}\Big(AoII_j(t) \Big). \tag{6.56}$$

However, a crucial drawback of the AoII metric is the fact that its computation requires knowledge of the actual current state of the process $X(t)$. Thus, AoIIs cannot be computed at the base station beforehand and a centralized multiple source scheduling policy like (6.56) cannot be implemented in reality, since the base station does not know the actual current states of each process. The CSMA based protocols we develop in this chapter provide a way out of this dilemma. Sources can compute their own AoIIs, since they have access to $X(t)$, $\hat{X}(t)$, t and $V(t)$. Then, a CSMA style policy that uses AoIIs instead of the AoIs can be implemented to pick the source that has the highest AoII and get better monitoring performance. We illustrate how to incorporate AoII into a CSMA style policy using the idealized version of Fresh-CSMA in Alg. 12. The near-realistic version follows immediately, by replacing the weighted AoI with the AoII.

For the setting involving monitoring Markov sources, the following choice of

Algorithm 12: Idealized Fresh-CSMA with AoIIs

Input : parameters $\alpha > 1, \delta \ll 1$

while $t \in 1, ..., T$ **do**
  for $i \in 1, ..., N$ **do**
    Generate a random timer $Z_i(t) \sim \exp\left(\alpha^{AoII_i(t)}\right)$.
    while $\tau < \delta Z_i(t)$ **do**
      Stay silent
    end
    if *Channel is free* **then**
      Transmit
    end
  end
end

the functions $f(\cdot)$ and $g(\cdot)$ is typically considered in literature

$$AoII(t) = \big(t - V(t)\big)\mathbb{1}_{\{X(t)\neq\hat{X}(t)\}}. \tag{6.57}$$

For this specific AoII metric, we can show a result similar to Theorem 15 in the case of weighted AoI.

Theorem 19. *Given any $\delta \in (0,1)$ and $AoII_1(t), ..., AoII_N(t)$ evolving according to 6.57; if we set $\alpha \geq (N-1)\frac{1-\delta}{\delta}$, then the following holds*

$$\mathbb{P}\left(\pi^{CSMA-AoII}(t) = \pi^{max-AoII}(t)\right) \geq 1 - \delta. \tag{6.58}$$

Here, $\pi^{max-AoII}(t)$ is the scheduling decision given by (6.56)*, while $\pi^{CSMA-AoII}(t)$ is the scheduling decision made by the idealized Fresh-CSMA policy that utilizes AoIIs (Alg. 12)*

Proof. The proof is identical to that of Theorem 15 - all we need to do is replace

the weighted AoIs with the AoIIs. The evolution of AoII according to 6.57 ensures that all the AoII values are integers. As before, we divide the proof into two parts.

Case 1: The expression $\underset{j\in[N]}{\text{argmax}}\Big(AoII_j(t)\Big)$ has a unique maximum. Let this maximum be the source 1 without loss of generality. Then, the max-AoII decision is to schedule source 1. Since we know that AoIIs are integers by (6.57), so the following must hold:

$$AoII_1(t) - AoII_i(t) \geq 1, \forall i \neq 1. \tag{6.59}$$

Now, applying Lemma 1, we can calculate the following probability

$$\begin{aligned}\mathbb{P}\Big(\pi^{CSMA-AoII}(t) = 1\Big) &= \frac{\alpha^{AoII_1(t)}}{\sum\limits_{j=1}^{N}\alpha^{AoII_j(t)}} \\ &= \frac{1}{1+\sum_{i=2}^{N}\alpha^{AoII_i(t)-AoII_1(t)}} \\ &\geq \frac{1}{1+(N-1)\alpha^{-1}} \\ &\geq 1-\delta.\end{aligned} \tag{6.60}$$

The first inequality follows by using (6.59) while the second inequality follows due to the fact that $\alpha \geq (N-1)\frac{1-\delta}{\delta}$.

Case 2: The expression $\underset{j\in[N]}{\text{argmax}}\Big(AoII_j(t)\Big)$ has multiple maxima. Suppose that the set of maxima is given by the nodes $\{1,...,k\}$. Then, the max-AoII policy will choose one of these sources to be scheduled. We want to lower bound the probability that Fresh-CSMA with AoIIs chooses a node from within this set. To do so, we first make a similar observation as in the case above.

$$AoII_1(t) - AoII_j(t) \geq 1, \forall j = k+1,...,N. \tag{6.61}$$

Using Lemma 1, we calculate the probability of interest

$$\begin{aligned}\mathbb{P}\left(\pi^{CSMA-AoII}(t) \in \{1,...,k\}\right) &= \frac{k\alpha^{AoII_1(t)}}{\sum_{j=1}^{N}\alpha^{AoII_j(t)}} \\ &= \frac{k}{k+\sum_{i=k+1}^{N}\alpha^{AoII_i(t)-AoII_1(t)}} \\ &\geq \frac{1}{1+(N-k)\alpha^{-1}} \\ &\geq 1-\delta.\end{aligned} \tag{6.62}$$

As before, the first inequality follows by using (6.61) while the second inequality follows due to the fact that $\alpha \geq (N-1)\frac{1-\delta}{\delta}$. This completes the proof. □

Theorem 19 shows that on a per-time-slot basis, Fresh-CSMA based on AoII matches the scheduling decisions made by the hypothetical centralized max-AoII policy (6.56), which cannot be implemented in reality. In Section 6.5, we will show via simulations that the Fresh-CSMA policy based on AoII can achieve lower time-average AoII across the network than even the centralized max-weight policy that utilizes only AoIs. This suggests that in certain settings distributed policies that utilize monitoring error can outperform centralized policies that have to make scheduling decisions while being oblivious to the actual processes.

6.5 Numerical Results

To verify the theoretical results developed in earlier sections, we now provide numerical results from packet level simulations of all the policies. Throughout this section, we assume that the minislot is $9\mu s$ long (a typical value in IEEE 802.11 implementations [127]) and an update packet from a each source is roughly 600 kB.

Thus, at a data rate of 54 Mbps (which is the highest data rate for IEEE 802.11g

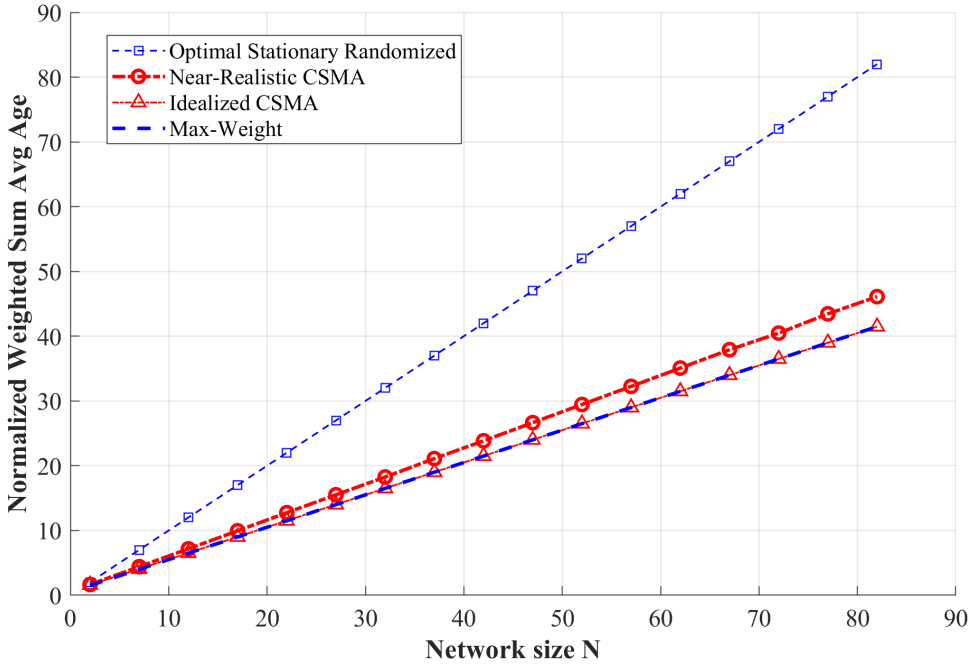


Figure 6-3: Normalized average AoI vs system size (N) for symmetric weights.

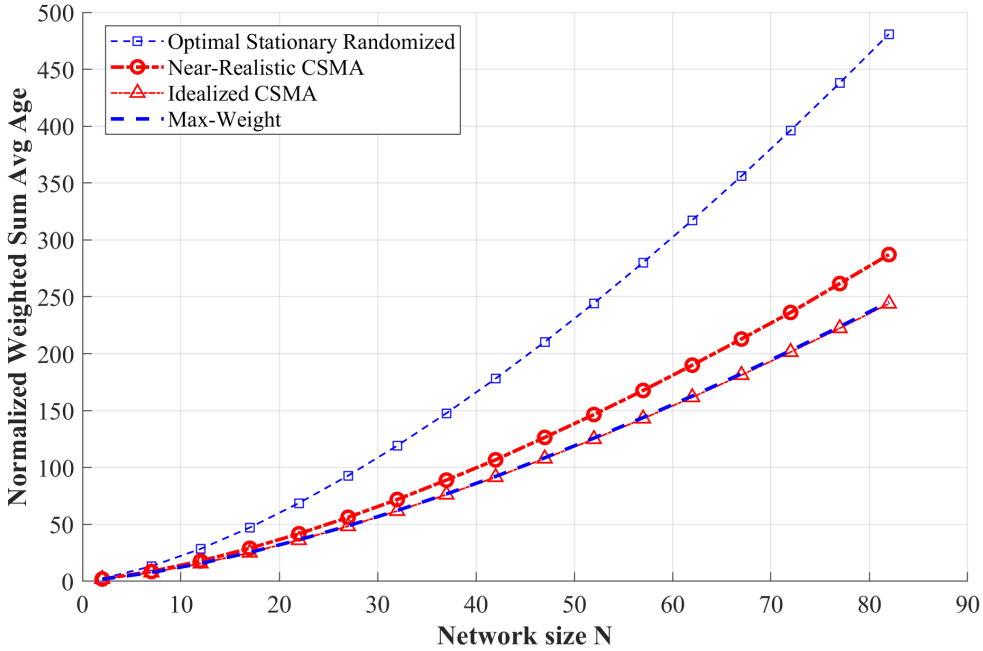


Figure 6-4: Normalized average AoI vs system size (N) for unequal weights.

in the 2.4 GHz band [127]), it takes 10000 minislots to finish sending an update. This, in turn, implies that we set $M = 10000$ for our simulations, i.e. each packet takes 10000 minislots to transmit.

Each experiment involves calculating the time-average of AoI, AoII, collision probabilities or backoff timer overheads. All of these time-averages are reported for experiments that involve 100000 application layer update packets being transmitted. Unless otherwise specified, we set $\alpha = 1+\frac{1}{\sum_i w_i}$, $\beta = 1.1+\max(\log(\log(N)), 0)$, and $B = 250 + N$ for the near-realistic CSMA implementation.

First, we consider the weighted sum average age minimization problem in the setting with equal weights, i.e. $w_i = 1, \forall i$. Fig. 6-3 plots the performance of two centralized policies - the optimal stationary randomized policy and the max-weight policy, as well as two distributed policies - the idealized Fresh-CSMA protocol and the near-realistic Fresh-CSMA protocol, as the the number of sources in the system N increases. We plot the normalized weighted sum average age, i.e. $\frac{1}{N}\sum_i w_i \bar{A}_i$ for each policy. We observe that the idealized Fresh-CSMA protocol matches the performance of the max-weight policy almost exactly, as expected from Theorems 15 and 16. Further, the near-realistic Fresh-CSMA protocol has only a small performance gap to the max-weight policy (due to collisions and backoff overheads) but still significantly outperforms the optimal stationary randomized policy.

Next, we consider the same setting but with asymmetric weights. We set the weight for source k to be $\sqrt{k}$, i.e. $w_k = \sqrt{k}$, and plot the normalized average age as the system size N increases in Fig. 6-4. We make the same observations regarding the performance of the policies as in the case of symmetric weights and further note that while our theoretical results needed weights w_i to be integers, that assumption is not required to get good performance in practice.

Next, we consider how the performance of our proposed protocols depends on the parameter α. Theorems 15 and 16 suggest that α needs to be very large

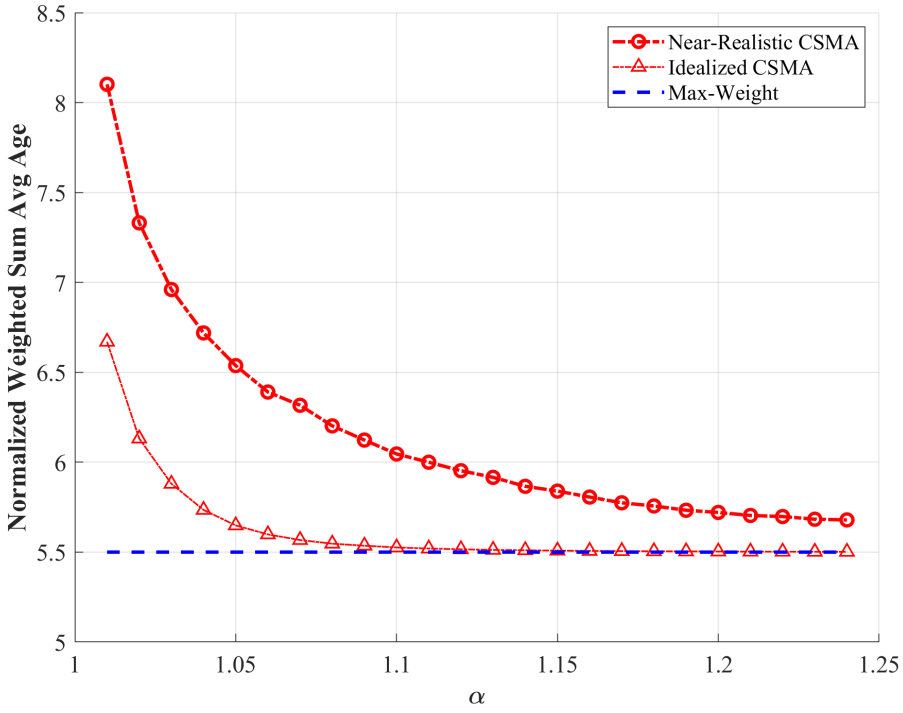


Figure 6-5: Normalized average AoI vs α.

to guarantee performance of Fresh-CSMA close to max-weight. However, we find that Fresh-CSMA starts performing very similarly to max-weight even at very small values of α, as show in Fig. 6-5 for a symmetric system with $N = 10$ sources. This is important in practice since large values of α could lead to integer overflows. As expected, the performance gap narrows as α increases.

Next, we look at the near-realistic Fresh-CSMA protocol in detail. In Fig. 6-6, we plot the collision probability for this protocol in a symmetric system with $N = 10$ sources as β is varied, while fixing all other parameters. We observe that for $\beta < 1.05$, the collision probability is 1 since all timers map to the first minislot. However, as we increase β beyond 1.05, we observe that the collision probability first drops to 0.01 and then gradually increases to 0.06. In Fig. 6-7, we plot the collision probability as B is varied, while fixing all other parameters. For small values of B, the collision probability is almost 1, but as B increases beyond a threshold, the collision probability stays roughly constant at around 0.015. These results are in line with what we expected from Theorem 17.

Next, we look at the average overhead of near-realistic Fresh-CSMA. Fig. 6-8 and Fig. 6-9 plot the average overhead (in number of minislots) as the parameters β and B are varied, respectively. As expected from Theorem 18, the overhead increases with both β and B. We note that our approximate expression for the overhead (6.53) is a good upper-bound that can be used in practice for system design. Also, we observe that the overhead remains relatively small (2-3%) compared to the update size, which takes 10000 minislots.

Finally, we consider the AoII metric discussed in Section 6.4. In this context, we look at the setting where each source is a symmetric two-state Markov chain, evolving independently over time (see Fig. 6-10). We set the transition probability q_i for each source to be 0.05 and implement the following three policies - centralized max-weight that uses AoI, distributed idealized Fresh-CSMA that uses AoII and distributed near-realistic Fresh-CSMA that uses AoII. For the CSMA imple-

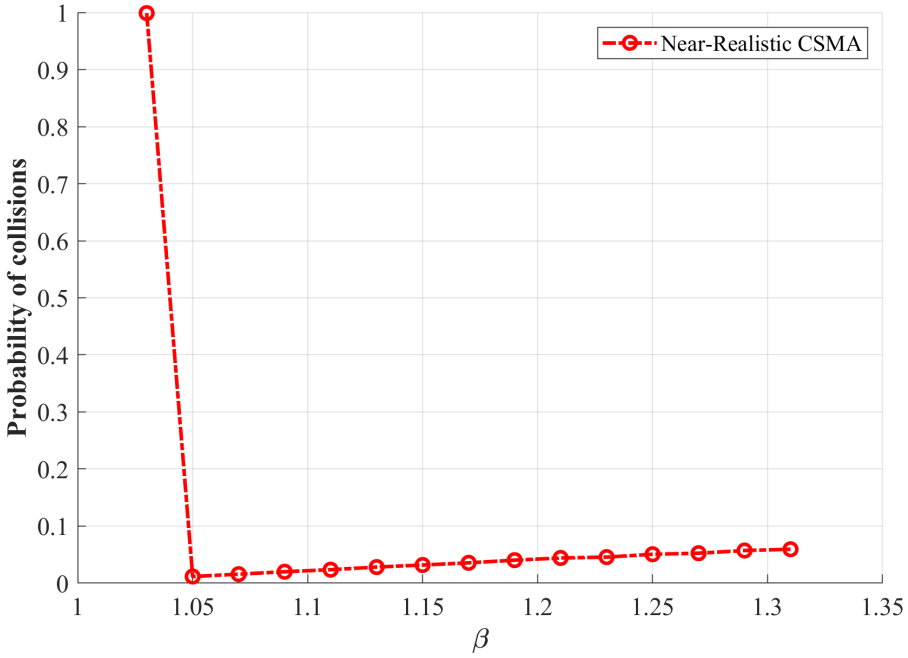


Figure 6-6: Probability of collisions vs β

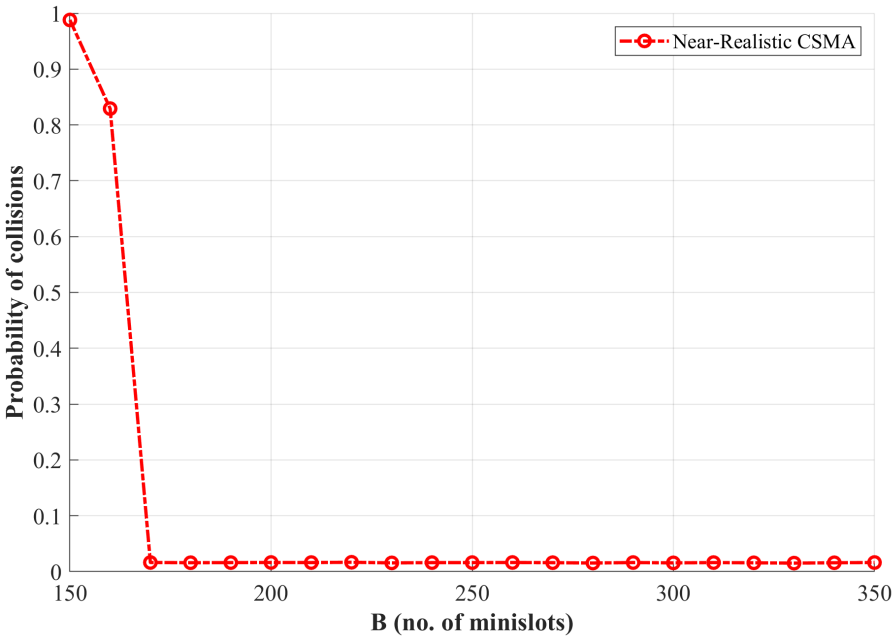


Figure 6-7: Probability of collisions vs B

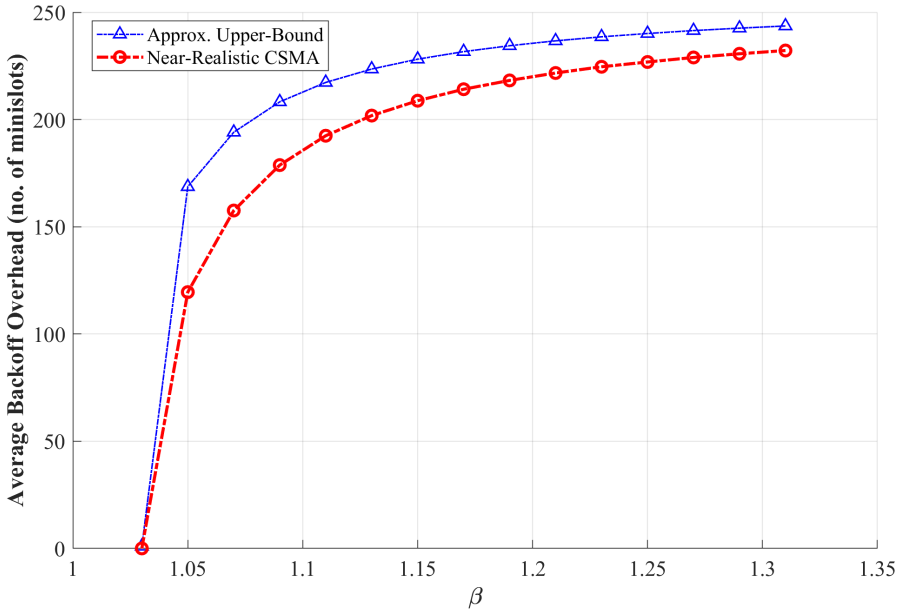


Figure 6-8: Average backoff overhead vs β

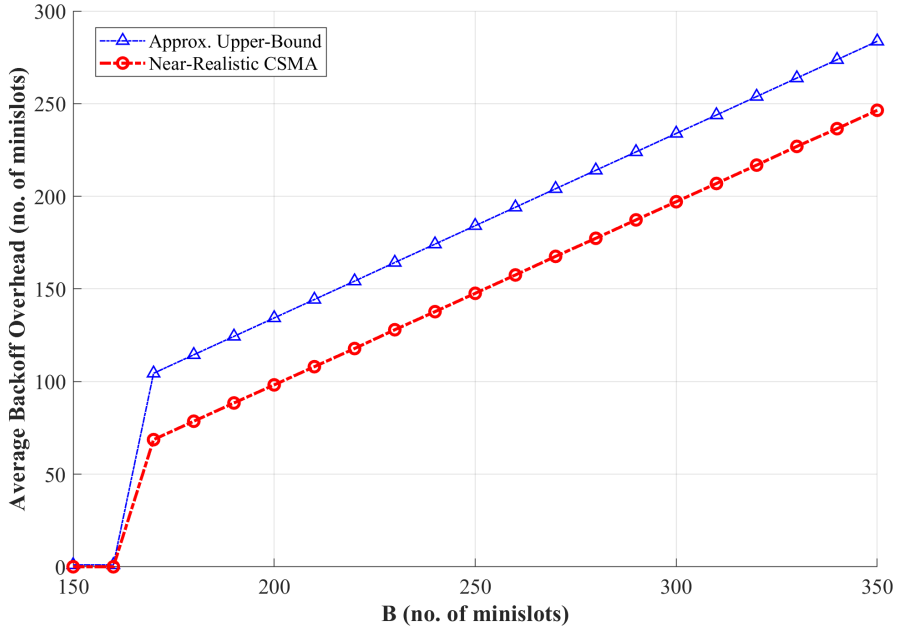


Figure 6-9: Average backoff overhead vs B

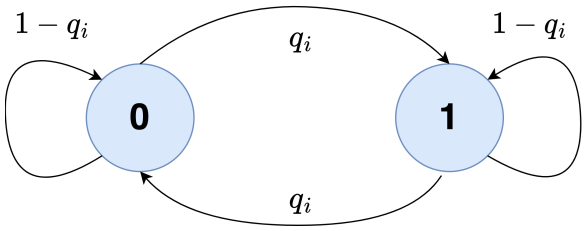


Figure 6-10: Symmetric two-state Markov chain representing the ith source.

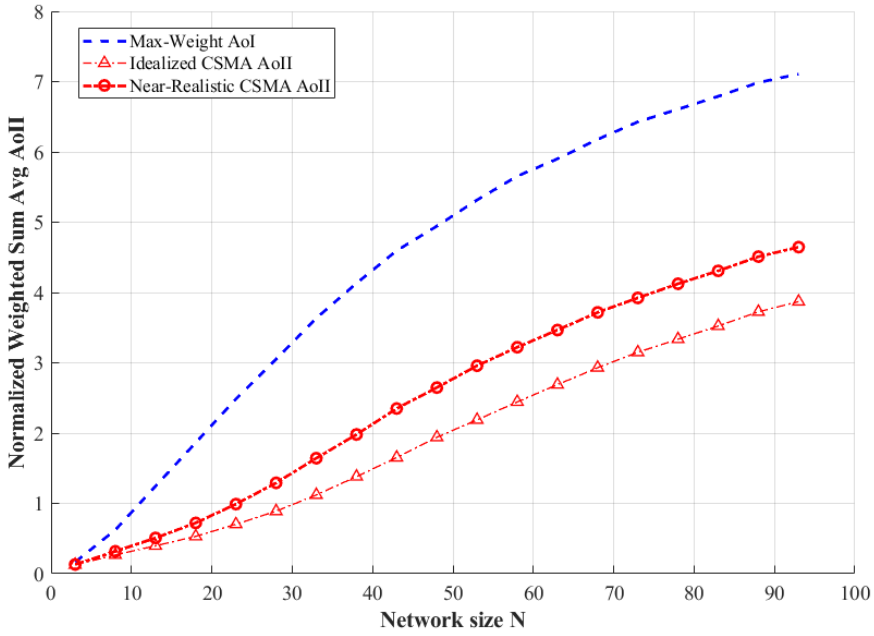


Figure 6-11: Normalized average AoII vs system size N when symmetric Markov sources

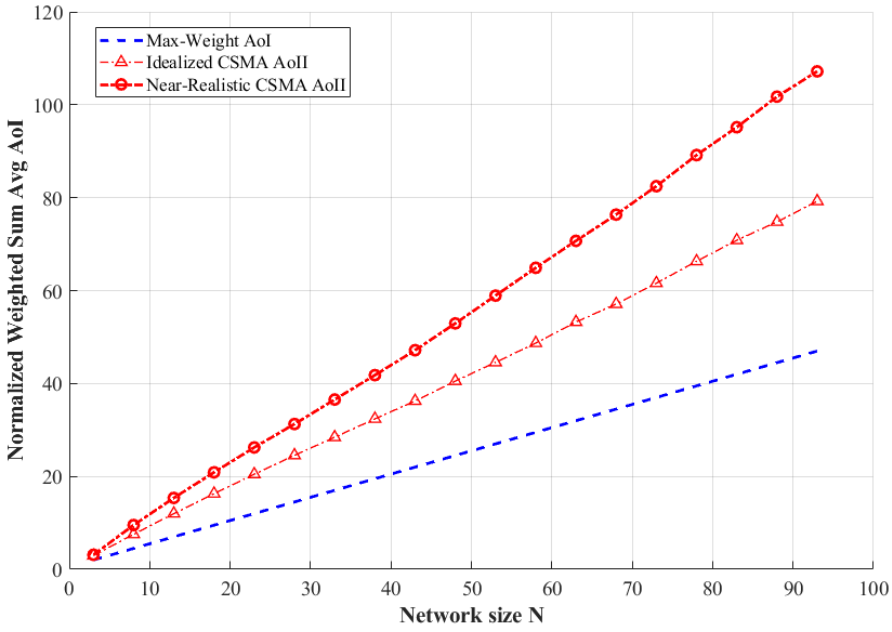


Figure 6-12: Normalized average AoI vs system size N when monitoring symmetric Markov sources

mentations, we set the parameters as follows $\alpha = 2.1, \beta = 1.05 + \log(\log(N))$ and $B = 250 + \lfloor N/4 \rfloor$.

Fig. 6-11 plots the time-average AoII performance of the three policies as we increase the number of sources in the system N. Here AoII is computed using (6.57). Clearly, the distributed CSMA versions deliver much better AoII performance than the centralized policy that is only able to utilize AoIs. Specifically, for 93 source, the idealized Fresh-CSMA with AoII performs about 45% better than AoI max-weight and the near-realistic version of Fresh-CSMA with AoII performs about 35% better than AoI max-weight. This suggests that our CSMA based design is general and can easily accommodate other kinds of information freshness and distortion metrics.

Interestingly, the gain in AoII performance comes at the cost of higher AoIs. In Fig. 6-12, we plot the long-term time-average AoIs for the same three policies while monitoring Markov sources as the number of source N is increased. We

observe that the distributed CSMA versions have higher AoIs than the centralized AoI max-weight. So, Fresh-CSMA based on AoII trades off better monitoring performance/error measured in terms of AoII with worse performance in terms of standard AoI.

6.6 Summary

In this chapter, we designed a distributed CSMA protocol to minimize weighted sum Age of Information in single-hop wireless networks. We showed that under idealized assumptions, our proposed protocol can closely replicate the behavior of centralized policies known to be nearly optimal. We also analyzed our protocol under a near-realistic medium access model and showed how system parameter choices affect packet collisions and overhead. Our simulation results confirm that our protocol works well in practice and that the performance gap between the idealized version and the near-realistic version of Fresh-CSMA is small. We have also extended some of our results to AoII, a more general information freshness metric. Two important directions of work involve further analysis of our protocols beyond AoI and AoII to metrics such as monitoring error or real-time control costs, and implementing the protocol in real systems to compare performance against standard WiFi, as done in Chapter 8.

Chapter 7

Optimizing Age of Information with Correlated Sources

While AoI is a proxy for measuring the cost of having out-of-date information, it may not properly reflect the impact of stale information on system performance. For example, a source might have a high AoI but the monitor might have a good estimate of its state because another source monitoring phenomena nearby sent updates very recently.

Many prior works have looked at optimizing information freshness metrics under different assumptions on the interference constraints [15, 17], arrival processes [132, 21], costs of AoI [16, 1], and update sizes [21, 3]. However, all of these works assume that the information across different sources is *decoupled* or *uncorrelated*, i.e. update deliveries from one source only influence the AoI evolution for that source. This is not strictly true in practice. Many monitoring and control applications involve observing information from correlated or coupled sources. Examples include: vehicular networks where vehicles communicate with their neighbors, multi-agent robotics tasks such as mapping where robots sense overlapping information, and wireless sensor networks where sensors collect spatially correlated updates or exchange information locally.

In such applications, updates from one source often contain information about the current state of other sources. The focus of this chapter is to understand to role of correlation in designing scheduling policies for information freshness in wireless networks. In Sec. 7.1, we formulate a simple model to analyze weighted-sum average AoI in the presence of correlated sources under wireless interference constraints. In Sec. 7.2, we use this model to design scheduling policies that can utilize the correlation structure between sources. We formulate a convex problem that solves for the optimal stationary randomized policy and show that it is factor-2 optimal in general. We then develop a Lyapunov drift-based max-weight policy that works well in practice and show that it is also constant factor optimal. In Sec. 7.3, we provide scaling results that allow us to understand how the degree of correlation affects information freshness. In Sec. 7.4, we discuss some alternate ways to model correlation and show that the average AoI for these models remains the same as our proposed model under randomized policies. This highlights the robustness of our results to the way in which correlation is modeled. In Sec. 7.5, we consider the setting where correlation parameters are unknown and possibly time-varying. Here, we propose a heuristic algorithm called *EMA-max-weight* based on exponential moving averages. This algorithm attempts to both keep track of the correlation parameters and adjust the scheduling decisions in an online manner so as to keep information fresh at the base station. Finally, in Sec. 7.6, we show numerically that our proposed policies outperform scheduling schemes that ignore the correlation structure inherent in the problem and verify our theoretical results.

To the best of our knowledge, our paper [7] on which this chapter is based, was the first one to consider coupled AoI evolution. In our model, the AoI of other sources can drop whenever a source that is correlated with them transmits, since a correlated update can reduce uncertainty about the state of other sources. This couples the AoI evolution of sources and leads to fundamentally new scheduling

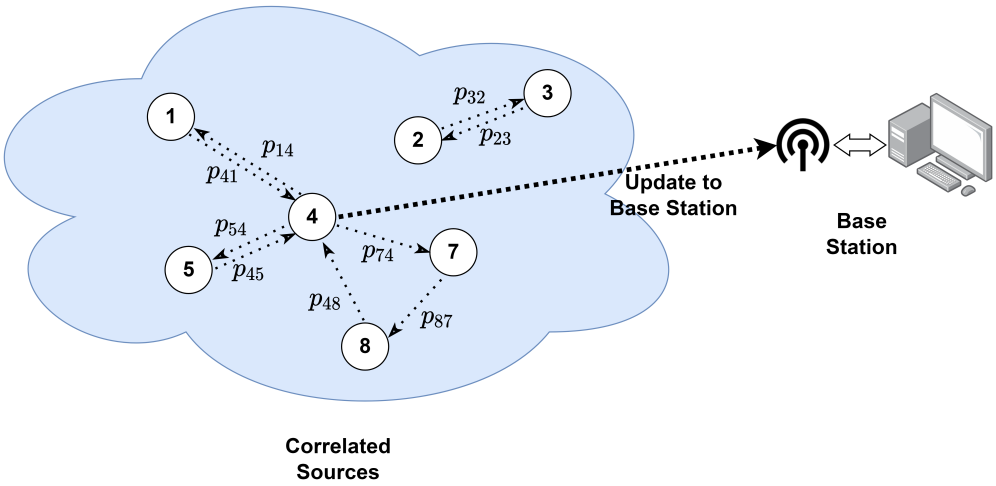


Figure 7-1: Sources share information locally and send updates to a base station.

design and scaling results.

7.1 System Model

Consider N sources monitoring phenomena of interest and sending updates to a base station. We assume discrete slotted time and assume that only one source can transmit to the base station successfully in any given time-slot.

We consider a simple model for the correlation structure between sources. At the beginning of every time-slot, each source i collects information about its own state. In addition, with probability p_{ij}, the update collected by source i also contains information about the current state of source j, for example due to overlapping fields of view between source i and j or due to spatial correlation between the processes being monitored. We assume that this information sharing or overlap happens *independently* across each *ordered pair* of sources and over time. Clearly, a value of $p_{ij} = 0$ suggests that there is never any information at i about j, while a value of $p_{ij} = 1$ suggests that i has complete information about j at all times. The overall correlation structure between the sources is described

by a matrix **P** which contains the pairwise correlation probabilities. Figure 7-1 depicts an example of such a system, where sources have access to some local information about their neighbors and send updates to a base station. Note that $p_{ii} = 1$ for all sources, since each source is assumed to have information about itself. However, our model can also capture situations where a source fails to obtain information about itself occasionally, by setting $p_{ii} < 1$.

When a particular source transmits to the base station, it sends information about its own state to the base station. However, this update will also contain shared information about some of its neighboring sources. Thus, updates sent to the base station are *correlated*: they can contain information about multiple sources at a time. For example, when source 4 transmits to the base station in the setup depicted in Figure 7-1, it transmission will contain information not just about itself but also about source 1 with probability p_{41}, source 5 with probability p_{45} and source 8 with probability p_{48}.

While we focus on the broadcast interference setting with reliable channels to develop our results and insights, our work can be easily extended to general interference constraints and unreliable channels.

7.1.A Correlated Age of Information

Let $u_i(t)$ be an indicator variable that denotes whether source i transmits to the base station in time-slot t. Further let $X_{ij}(t)$ be an indicator variable that denotes whether the current update at source i contains common information about source j in time-slot t. From our discussion above, we know that $X_{ij}(t) \sim Bern(p_{ij})$ independent across pairs (i, j) and also over time.

Given this correlation structure, the Age of Information for source i at the base

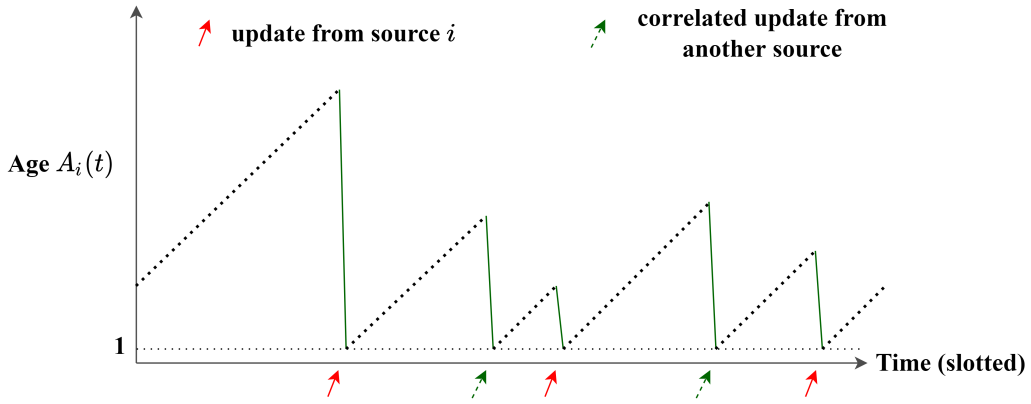


Figure 7-2: Correlated AoI evolution.

station evolves as follows:

$$A_i(t+1) = \begin{cases} 1, & \text{if } \sum_{j=1}^{N} u_j(t) X_{ji}(t) = 1 \\ A_i(t)+1, & \text{otherwise.} \end{cases} \tag{7.1}$$

The equation (7.1) implies that the AoI of source i at the base station drops to 1 whenever i itself transmits a new update or j transmits a new update containing information about i. Figure 7-2 depicts this AoI evolution. Note that (7.1) also assumes that sources only transmit their freshest update, i.e. an older update containing more information about neighbors might not be sent since it was replaced by a newer update with lesser information about neighbors in the next time-slot.

As in the previous chapter, our metric of interest will be average AoI, which is simply the long-term time-average of the AoI process. Specifically,

$$\bar{A}_i \triangleq \limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} A_i(t). \tag{7.2}$$

7.1.B Goal

Given the probabilistic correlation structure and the AoI evolution described above, we want to design a wireless scheduling policy that minimizes the weighted sum of average AoI across all sources:

$$\underset{\pi}{\operatorname{argmin}}\left(\limsup_{T\to\infty} \left[\frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{N} w_i A_i(t) \right] \right). \tag{7.3}$$

Here, the weights $\{w_1, w_2, ..., w_N\}$ are positive real numbers that denote the relative importance of each source to the overall monitoring or control application.

7.2 Scheduling Policies

To solve the optimization problem (7.3), we first study stationary randomized policies in Sec. 7.2.A. These policies are amenable to analysis and provide key structural insights. Then, in Sec. 7.2.B, we develop a Lyapunov drift-based max-weight policy and prove performance bounds for it using the structure of the optimal stationary randomized policy.

7.2.A Stationary Randomized Policies

A stationary randomized policy is described by a probability distribution π over the set of sources, where π_i denotes the probability of choosing source i. In every time-slot, the policy chooses which source gets to transmit by sampling from the distribution π and scheduling decisions are sampled independently across time-slots.

The following theorem relates the average AoI to the scheduling distribution π.

Theorem 20. *Consider any stationary randomized policy with scheduling probabilities $\boldsymbol{\pi}$. If $\sum_{j=1}^{N} \pi_j p_{ji} > 0$ for a source i, then the average AoI for this source is given by:*

$$\bar{A}_i \triangleq \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} A_i(t) = \frac{1}{\sum_{j=1}^{N} \pi_j p_{ji}}. \tag{7.4}$$

If $\sum_{j=1}^{N} \pi_j p_{ji} = 0$ for some source i, then the base station never receives any information regarding this source and its average AoI $\bar{A}_i$ is unbounded.

Proof. See Appendix 7.8.A. □

Using the above theorem, we can formulate an optimization problem to find optimal stationary randomized policies.

Lemma 11. *Consider the space of stationary randomized policies Π^{sr}. Finding a policy $\boldsymbol{\pi}^* \in \Pi^{sr}$ that minimizes the weighted sum of average AoI is equivalent to solving the following optimization problem:*

$$\begin{aligned} \underset{\boldsymbol{\pi}}{\operatorname{argmin}} & \sum_{i=1}^{N} \frac{w_i}{\sum_{j=1}^{N} \pi_j p_{ji}} \\ s.t. & \sum_{i=1}^{N} \pi_i \leq 1, \\ & \pi_i \geq 0, \forall i \in [N]. \end{aligned} \tag{7.5}$$

Proof. Using the definition of average AoI (7.2) and the expression derived in Theorem 20, we can simplify (7.3) to obtain (7.5). The constraints simply represent a valid stationary randomized policy given the assumption that only one source can transmit in any given time-slot. □

Next, we discuss how to solve the optimization problem (7.5).

Theorem 21. *The optimization problem* (7.5) *is convex in the probability distribution $\boldsymbol{\pi}$. Further, if the optimal solution $\boldsymbol{\pi}^*$ to* (7.5) *is such that $\pi_i^* > 0, \forall i \in S$*

where $S \subseteq [N]$ is a subset of sources, then $\boldsymbol{\pi}^*$ can be found by solving the following system of nonlinear equations:

$$\sum_{j=1}^{N} p_{ij} w_j \bar{A}_j^2 = \lambda, \forall i \in S, \text{ and} \tag{7.6}$$

$$\sum_{i \in S} \pi_i^* = 1. \tag{7.7}$$

Here, $\lambda > 0$ is a constant and $\bar{A}_i$ denotes the average AoI for source i under the policy $\boldsymbol{\pi}^*$ computed using (7.4).

Proof. See Appendix 7.8.B. □

Theorem 21 establishes two key results. First, since the optimization problem (7.5) is convex, it can be solved efficiently by using a standard solver such as cvx [133]. Second, if the optimal policy involves scheduling some subset of sources a positive fraction of the time, then the quantity $\sum_{j=1}^{N} p_{ij} w_j \bar{A}_j^2$ is constant across all sources in this subset. This can be contrasted with the equivalent result in the uncorrelated case, where the quantity $w_j \bar{A}_j^2$ is constant across all sources [15]. We also recover the well known result from the uncorrelated case, i.e. $\pi_i^* \propto \sqrt{w_i}$ if we set $\mathbf{P} = \mathbf{I}$.

Until now, we have only discussed optimization within the space of stationary randomized policies. The following theorem shows that this class is not too restrictive, i.e. the best stationary randomized policies are at-most a factor of two away from the best possible scheduling policy *in general.*

Theorem 22. *Consider an optimal stationary randomized policy $\boldsymbol{\pi}^*$ that is a solution to* (7.5) *and an optimal policy $\boldsymbol{\pi}^{opt}$ that solves the general problem* (7.3). *Let $\bar{\mathbf{A}}^*$ denote the average AoIs under $\boldsymbol{\pi}^*$ and $\bar{\mathbf{A}}^{opt}$ denote the average AoIs under $\boldsymbol{\pi}^{opt}$. Then,*

$$\frac{\sum_{i=1}^{N} w_i \bar{A}_i^*}{\sum_{i=1}^{N} w_i \bar{A}_i^{opt}} \leq 2. \tag{7.8}$$

Proof. See Appendix 7.8.C. □

7.2.B Max-Weight Policy

Motivated by the Lyapunov drift-based policies proposed in [15, 17, 28], we next look at an alternative way to design scheduling policies that take correlation into account. Consider the quadratic Lyapunov function given by:

$$L(t) \triangleq \sum_{i=1}^{N} w_i A_i^2(t). \tag{7.9}$$

Then, the *quadratic max-weight policy* chooses a scheduling decision that minimizes the one-slot Lyapuonv drift in every time-slot.

$$\pi^{qmw}(t) = \underset{i\in[N]}{\operatorname{argmin}} \ \mathbb{E}\Big[L(t+1) - L(t)\Big|\mathbf{A}(t)\Big]. \tag{7.10}$$

This simplifies to:

$$\pi^{qmw}(t) = \underset{i\in[N]}{\operatorname{argmin}} \ \sum_{j=1}^{N} w_j p_{ij} A_j(t)\Big(A_j(t)+2\Big). \tag{7.11}$$

The following theorem provides an upper-bound on the performance of the quadratic max-weight policy.

Theorem 23. *Consider the quadratic max-weight policy π^{qmw} and an optimal policy π^{opt} that solves the general problem* (7.3)*. Let $\bar{\mathbf{A}}^{qmw}$ denote the average AoIs under π^{qmw} and $\bar{\mathbf{A}}^{opt}$ denote the average AoIs under π^{opt}. Then,*

$$\frac{\sum_{i=1}^{N} w_i \bar{A}_i^{qmw}}{\sum_{i=1}^{N} w_i \bar{A}_i^{opt}} \leq 4. \tag{7.12}$$

Proof. See Appendix 7.8.D. □

Observe that computing the decisions in the equation above does not require us to explicitly solve for the optimal stationary randomized policy π^*. Next, we will develop a max-weight policy that utilizes the optimal stationary randomized policy π^* to get even better performance guarantees.

Consider an optimal stationary randomized policy π^* that solves the optimization problem (7.5). Using π^*, we define the quantities:

$$\alpha_i \triangleq \frac{w_i}{\sum_{j=1}^{N} \pi_j^* p_{ji}}, \forall i \in [N]. \tag{7.13}$$

We use the quantities α_i to construct the following linear Lyapunov function:

$$L(t) \triangleq \sum_{i=1}^{N} \alpha_i A_i(t). \tag{7.14}$$

Then, our new max-weight policy chooses a scheduling decision that minimizes the one-slot Lyapuonv drift in every time-slot.

$$\pi^{mw}(t) = \underset{i\in[N]}{\operatorname{argmin}} \ \mathbb{E}\Big[L(t+1) - L(t)\Big|\mathbf{A}(t)\Big]. \tag{7.15}$$

The structure of the max-weight policy can be obtained by simplifying the expression in (7.15) above:

$$\begin{aligned} \pi^{mw}(t) &= \underset{i\in[N]}{\operatorname{argmin}}\left(\sum_{j=1}^{N} p_{ij}\alpha_j A_j(t)\right) \\ &= \underset{i\in[N]}{\operatorname{argmin}}\left(\alpha_i A_i(t) + \sum_{j\neq i} p_{ij}\alpha_j A_j(t)\right). \end{aligned} \tag{7.16}$$

Note that the max-weight policy proposed in [28, Sec. 3.2.4] for the uncorrelated setting schedules the source i with the highest value of $\sqrt{w_i}A_i(t)$. On the other hand, our correlated max-weight policy (7.16) adds up the "value" of each possible AoI reduction including correlated updates, weighted by the of probability of

correlation.

The following theorem shows that the new max-weight policy defined in (7.16) enjoys a similar factor-2 optimality guarantee as the optimal stationary randomized policy which is better than the factor-4 guarantee for the quadratic max-weight policy described in (7.11).

Theorem 24. *Consider the max-weight policy π^{mw} and an optimal policy π^{opt} that solves the general problem* (7.3). *Let $\bar{\mathbf{A}}^{mw}$ denote the average AoIs under π^{mw} and $\bar{\mathbf{A}}^{opt}$ denote the average AoIs under π^{opt}. Then,*

$$\frac{\sum_{i=1}^{N} w_i \bar{A}_i^{mw}}{\sum_{i=1}^{N} w_i \bar{A}_i^{opt}} \leq 2. \tag{7.17}$$

Proof. See Appendix 7.8.E. □

While we show a factor of two optimality result above, we will show via simulations in Sec. 7.6 that the max-weight policy performs almost as well as the theoretical lower bound derived in Appendix 7.8.C and also outperforms the optimal stationary randomized policy in practice.

7.3 Scaling

In this section, we consider how correlation improves information freshness as network sizes scale. For deriving our scaling results, we will focus on the equal weights setting, i.e. $w_i = \frac{1}{N}, \forall i \in [N]$. However, similar scaling results will hold qualitatively for general weight configurations.

The lemma below characterizes the minimum average AoI of a network with N uncorrelated sources with equal weights. We will use this as a comparison baseline to see how the degree of correlation improves AoI.

Lemma 12. *Consider N uncorrelated sources sending updates to a base station. Let the scheduling weights for all sources be equal, i.e. $w_i = \frac{1}{N}, \forall i$. Then, the optimal weighted sum AoI satisfies the following:*

$$\sum_{i=1}^{N} w_i \bar{A}_i^{opt} = \frac{N+1}{2} \sim \Theta(N). \tag{7.18}$$

Proof. In the setting with symmetric weights, the greedy or the round-robin policy is known to be optimal [15]. It is easy to see that the average AoI under the round-robin policy is simply the average of the sequence $1, 2, ..., N$ which is $\frac{N+1}{2}$. □

7.3.A An Upper Bound

We derive a general upper bound for the weighted-sum average AoI by representing information about the correlation matrix **P** for N sources with a directed graph.

Consider a correlation threshold $p \in (0, 1)$. We are interested in the entries of the correlation matrix above this threshold, i.e the pairs of sources that are significantly correlated. To do so, we construct a directed graph $\mathcal{G}(V, E)$ on the set of sources. For every ordered pair of sources (i, j) such that $p_{ij} > p$, we add the edge (i, j) to the graph $\mathcal{G}$. Trivially, every node in $\mathcal{G}$ must have a self-loop, since $p_{ii} = 1, \forall i$. We will show that the average AoI of the network can be upper-bounded by analyzing the properties of these constructed graphs. To derive this upper bound, we first need to define the notion of a vertex cover for a directed graph.

Vertex Cover: Given a directed graph $\mathcal{G}(V, E)$, a vertex cover is defined to be a set of vertices $S \subseteq V$ if for every vertex $i \in V$, there exists a vertex $j \in S$ such that the edge (j, i) is in the set E.

The following theorem relates the average AoI to the size of the *minimum ver-*

tex cover of the graph $\mathcal{G}$ constructed using the correlation threshold p.

Theorem 25. *Consider N sources with the correlation matrix $\mathbf{P}$. Given a correlation threshold $p > 0$, construct a directed graph $\mathcal{G}$ that represents pairs of source with correlation higher than the threshold. Assuming equal weights $w_i = \frac{1}{N}, \forall i$, the optimal weighted sum average AoI satisfies:*

$$\sum_{i=1}^{N} w_i \bar{A}_i^{opt} \leq \frac{N_{cov}}{p}, \tag{7.19}$$

where N_{cov} is the size of a minimum vertex cover for the graph $\mathcal{G}$ and p is the correlation threshold.

Proof. See Appendix 7.8.F. □

This upper-bound allows us to relate the degree of correlation or information sharing between the sources to the average AoI. If a small subset of sources, of say size $O(\log N)$, are highly correlated with all other sources in the network, then average AoI with correlation is only $O(\log N)$ as well. In this case, correlation leads to a significant reduction in the average AoI compared to the uncorrelated case, which is $O(N)$ as shown in Lemma 12. If no such small vertex coverings exist or if correlation probabilities are very small, then a scheduler likely needs to communicate with a large fraction of all the sources and the average AoI would grow as $O(N)$, similar to the uncorrelated case.

Further, we show that it is possible to construct a correlation matrix such that the average AoI of the correlated max-weight given by (7.16) is $O(1)$ while the average AoI of the uncorrelated max-weight policy from [15] is $\Theta(N)$. This means that the performance gap between policies that consider and ignore correlation can grow linearly with the size of the network.

Theorem 26. *Given any number of sources N with equal weights $\frac{1}{N}$, we consider two policies - the correlated max-weight policy π^{mw} proposed in Sec. 7.2.B and the*

max-weight policy that does not take correlation into account π^u which was proposed in [15]. Let the average AoI of a source i under π^{mw} be $\bar{A}_i^{mw}$ and under π^u be $\bar{A}_i^u$. Then, there exists a correlation matrix $\mathbf{P} \in [0,1]^{N\times N}$ such that the following holds:

$$\frac{\sum_{i=1}^N w_i \bar{A}_i^u}{\sum_{i=1}^N w_i \bar{A}_i^{mw}} \sim \Omega(N). \tag{7.20}$$

Proof. Consider N sources with the following correlation matrix:

$$\mathbf{P} = \begin{bmatrix} 1 & p & p & \cdots & p \\ p & 1 & 0 & \cdots & 0 \\ p & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & 1 & \vdots \\ p & 0 & 0 & \cdots & 1 \end{bmatrix}. \tag{7.21}$$

The stationary randomized policy $\boldsymbol{\pi}^{(1)}$ that puts all scheduling weight on source 1 achieves a weighted-sum AoI of:

$$\sum_{i=1}^N w_i \bar{A}_i = \frac{1}{N} + \sum_{i=2}^N \frac{1}{N}\frac{1}{p} \leq \frac{1}{p}. \tag{7.22}$$

Thus, the optimal stationary randomized policy $\boldsymbol{\pi}^*$ also achieves weighted-sum average AoI that is at-least as good as that of $\boldsymbol{\pi}^{(1)}$. This implies:

$$\sum_{i=1}^N w_i \bar{A}_i^* \leq \frac{1}{p}, \tag{7.23}$$

where $\bar{A}_i^*$ is the average AoI of source i under the optimal stationary randomized policy $\boldsymbol{\pi}^*$. In Appendix 7.8.E, we show that the performance of the max-weight policy is upper-bounded by the policy of the optimal stationary randomized policy. Thus, we get:

$$\sum_{i=1}^N w_i \bar{A}_i^{mw} \leq \sum_{i=1}^N w_i \bar{A}_i^* \leq \frac{1}{p}, \tag{7.24}$$

where $\bar{A}_i^{mw}$ is the average AoI of source i under the max-weight policy π^{mw} described by (7.16).

Now consider the performance of the uncorrelated max-weight policy π^u. Since all the weights are symmetric, the optimal policy is greedy or max-AoI-first. In Appendix 7.8.G, we show that the max-AoI-first policy behaves similar to a round-robin policy on sources $2, \ldots, N$ while occasionally scheduling source 1. Specifically, we derive a lower bound on the weighted-sum AoI under policy π^u:

$$\sum_{i=1}^{N} w_i \bar{A}_i^u \geq \frac{(N-1)^2}{2N - (N+1)(1-p)^{N-1}}. \tag{7.25}$$

This bound holds under the assumtion that $p \geq \frac{1}{N-1}$. Combining (7.24) and (7.25), we get:

$$\frac{\sum_{i=1}^{N} w_i \bar{A}_i^u}{\sum_{i=1}^{N} w_i \bar{A}_i^{mw}} \geq \frac{p(N-1)^2}{2N - (N+1)(1-p)^{N-1}}. \tag{7.26}$$

Assuming p to be a fixed constant that does not depend on N completes the proof. Note that a similar result also holds for p scaling with N, for example if $p \sim \frac{1}{N^\beta}, \beta < 1$ the performance gap is $\Omega(N^{1-\beta})$. □

The intuition behind this result is straightforward: if there is one source that is correlated even by a small amount with all other sources then scheduling just that source all the time should be sufficient. The gap in performance is achieved by assuming that other sources are not correlated with each other.

Theorem 26 provides a key insight: *the gain in performance cannot be obtained by using policies that ignore correlation.* It is necessary to design scheduling policies that take the correlation structure into account, especially for correlation graphs where the degree distribution is highly skewed. Next, we consider scaling in the special case where the correlation matrix can be represented by random geometric graphs. These graphs are commonly used to model wireless sensors networks monitoring spatial phenomenon.

7.3.B Random Geometric Graphs

Consider N sources distributed uniformly at random on the unit square $[0,1]^2$. Each source has information about itself, by definition. Thus $p_{ii} = 1, \forall i$. Further, if the distance between sources i and j is less than a threshold r, then we set $p_{ij} = p$ and $p_{ji} = p$, otherwise we set $p_{ij} = 0$ and $p_{ji} = 0$. This leads to a symmetric correlation matrix $\mathbf{P}$. Constructing a graph that connects sources with correlation leads to the random geometric graph $\mathcal{G}(N,r)$.

The following theorem looks at the scaling of AoI under this geometric correlation structure.

Theorem 27. *Consider a symmetric correlation matrix generated by creating a random geometric graph $\mathcal{G}(N,r)$ on the two dimensional unit square and setting correlation probabilities for neighbors to be p. Assuming equal weights $w_i = \frac{1}{N}, \forall i$, the weighted sum average AoI satisfies:*

$$\sum_{i=1}^{N} w_i \bar{A}_i^{opt} \leq \frac{2}{pr^2}. \tag{7.27}$$

Proof. See Appendix 7.8.H. □

Note that the connectivity threshold of a random geometric graphs occurs at $r \sim \Theta\left(\sqrt{\frac{\log N}{N}}\right)$. For this choice of r, we can see that the overall AoI of the network is $O\left(\frac{N}{p \log N}\right)$. This leads to a factor of $\log N$ reduction over the uncorrelated case analyzed in Lemma 12, assuming p is a constant. Further, if $r \sim \Theta(1)$, then the AoI is also $\Theta(1)$ and there is an $O(N)$ reduction compared to the uncorrelated case.

7.4 Robustness

In this section, we consider a different way to model correlation between a set of sources. We will show that for stationary randomized policies, this model is

equivalent to our proposed model. Later, in Sec. 7.6, we will show via numerical results that the scheduling policies designed in Sec. 7.2 also perform well for this new correlation structure. This suggests that if the "amount of correlation" is the same on average, then the AoI performance tends to be similar, regardless of how correlation is modeled.

As before, consider a set of N sources and a correlation matrix $\mathbf{P}$ with entries $p_{ij} \in [0,1]$ that represent the amount of information shared at source i about source j. However, instead of a Bernoulli random variable indicating whether i either does or does not currently have information about j, we now assume that i always has a constant p_{ij} fraction of information regarding the state of j. In other words, an update from i can reduce the current uncertainty regarding j by a fraction of p_{ij}. This correlation structure is convenient for modeling settings where different sources are sensing information and have *partially* overlapping ranges of sensing, for example cameras with overlapping fields of view as studied in [71, 72]. Thus, p_{ij} can be viewed as the fraction of source j's range that source i also covers.

Consequently, the evolution of AoI for this model differs from the one studied in earlier sections. Let $u_i(t)$ be indicator variables that denote whether source i transmits in time-slot t or not. Then, AoI for source i evolves as follows:

$$A_i(t+1) = (1-p_{ji})A_i(t) + 1, \text{ if } u_j(t) = 1. \tag{7.28}$$

The equation above formalizes the notion that uncertainty regarding source i drops by a fraction p_{ji}, when j sends an update.

The following theorem establishes the equivalence between this new model and the one proposed in Sec. 7.1, under stationary randomized policies. In fact, we prove the equivalence result for a much more general class of correlation structures, where correlation is defined by i.i.d. random variables $X_{ij}(t) \in [0,1]$

such that $\mathbb{E}[X_{ij}(t)] = p_{ij}$. Our original model assumes $X_{ij}(t) \sim Bern(p_{ij})$, while the model proposed in this section assumes $X_{ij}(t) = p_{ij}$ and both of them belong to this class of correlation structures.

Theorem 28. *Consider N sources and a correlation matrix $\mathbf{P}$ with the AoIs for each source evolving according to:*

$$A_i(t+1) = A_i(t) + 1 - \left(\sum_{j=1}^{N} u_j(t) X_{ji}(t)\right) A_i(t), \forall i, t. \tag{7.29}$$

Here $X_{ji}(t)$ are i.i.d random variables such that $X_{ji}(t) \in [0,1]$ and $\mathbb{E}[X_{ji}(t)] = p_{ji}$. Given any stationary randomized policy with scheduling probabilities π, if $\sum_{j=1}^{N} \pi_j p_{ji} > 0$ for a source i, then the average AoI for this source is given by:

$$\bar{A}_i \triangleq \lim_{T\to\infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} A_i(t)\right] = \frac{1}{\sum_{j=1}^{N} \pi_j p_{ji}}. \tag{7.30}$$

If $\sum_{j=1}^{N} \pi_j p_{ji} = 0$ for some source i, then the base station never receives any information regarding this source and its average AoI $\bar{A}_i$ is unbounded.

Proof. See Appendix 7.8.I. □

Observe that the expression for the average AoI is the same as the one derived in Theorem 20. Thus the procedure to find the optimal stationary randomized policy and the policy itself also remain the same. We will show later via simulations that the performance of the max-weight policy is also similar for different distributions of $X_{ji}(t)$, which suggests that our analysis is fairly robust to the way in which correlation is modeled.

7.5 Learning The Correlation Matrix

Until now, we have focused on cases where the correlation structure is known in advance and fixed over time, and we use this information to analyze and optimize AoI. In this section, we consider the setting when the correlation matrix is unknown and possibly varying with time.

7.5.A Online Setting

Learning the correlation matrix in an online setting where the **P** changes over time, even slowly, is a challenging problem. This setting is of interest because correlation between source tends to be time-varying in practice, especially in settings involving mobility.

We want to implement a max-weight style policy that gradually updates its policy parameters to be able to track changes in the environment. However, note that the max-weight policy proposed in Sec. 7.2.B in (7.16) requires us to solve for the optimal stationary randomized policy. Thus, whenever the correlation matrix changes, we would have to recalculate $\boldsymbol{\pi}^*$ and the Lyapunov function weights α_i. To avoid this added computation, we will use the quadratic max-weight policy, given by (7.11). Recall that this policy is based on a quadratic Lyapunov function and does not require us to calculate the the optimal stationary randomized policy $\boldsymbol{\pi}^*$ repeatedly as $\boldsymbol{P}$ varies.

Algorithm 13 uses an exponential moving average to keep track of the correlation probabilities and then runs the *quadratic* max-weight scheduler from (7.11) using the estimated correlation matrix. We call this the *EMA-max-weight* policy.

Intuitively, if the probabilities change slowly over time, the exponential moving average estimate should be able to closely track the actual correlation matrix and the EMA-max-weight policy would perform similar to a max-weight policy that knows the entire sequence of correlation matrices in advance. We confirm

Algorithm 13: Exponential Moving Average Max-Weight

Input : parameter $\alpha > 0$

Start by assuming no correlation, i.e. set

$$\hat{\mathbf{P}}(1) = \mathbf{I}$$

while $t \in 1, ..., T$ **do**

Run quadratic max-weight using $\hat{\mathbf{P}}(t)$:

$$s = \underset{i \in [N]}{\operatorname{argmin}} \sum_{j=1}^{N} w_j \hat{p}_{ij}(t) A_j(t) \Big(A_j(t) + 2 \Big).$$

Schedule source s and receive correlated updates

Update AoIs for every source j:

$$A_j(t+1) = \begin{cases} 1, & \text{if } s \text{ sent an update about } j, \\ A_j(t) + 1, & \text{otherwise.} \end{cases}$$

Update the correlation matrix, for source s

$$\hat{p}_{sj}(t+1) = \begin{cases} (1-\alpha)\hat{p}_{sj}(t) + \alpha, & \text{if } s \text{ sent an update about } j, \\ (1-\alpha)\hat{p}_{sj}(t), & \text{otherwise.} \end{cases}$$

For sources other than s:

$$\hat{p}_{ij}(t+1) = \hat{p}_{ij}(t), \forall i \neq s.$$

end

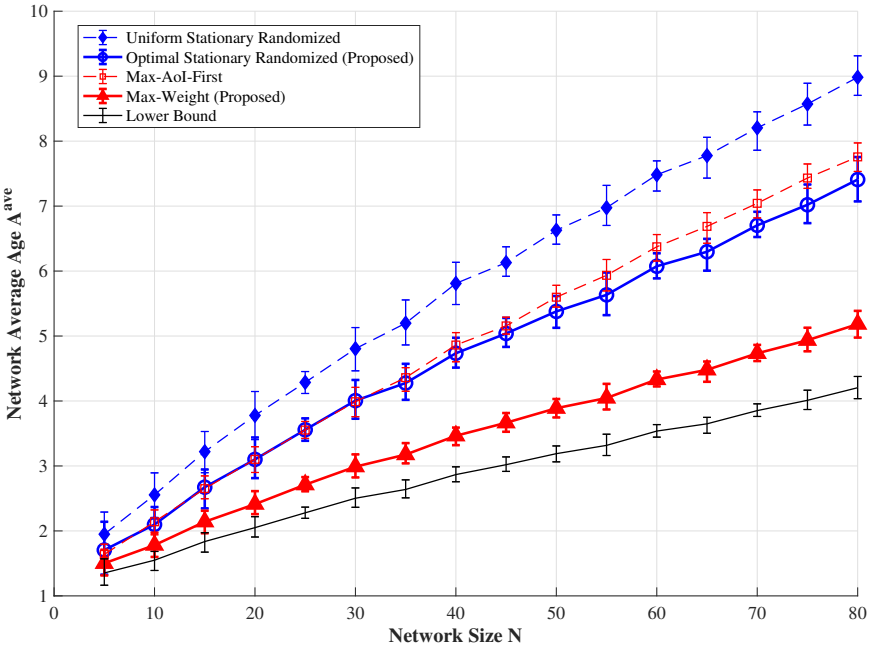


Figure 7-3: Average AoI vs network size N for random geometric graphs.

that this is indeed the case via simulations in Sec. 7.6.

7.6 Numerical Results

First, we consider random geometric graphs and see how different scheduling policies perform as the number of sources N increases. In particular, we simulate graphs $\mathcal{G}(N,r)$ on the unit square $[0,1]^2$ where $r = 1.1\sqrt{\frac{\log N}{N}}$ is slightly above the connection threshold. For each pair of nodes (i,j) in this graph, we set $P_{ij} = P_{ji} = 0.7$ if the nodes are closer than the distance r and set $P_{ij} = P_{ji} = 0$ otherwise. We fix all weights to be equal, i.e. $w_i = \frac{1}{N}$.

For each value of N, we compare the performance of four different policies - 1) the uniform stationary randomized policy which would have been the optimal stationary randomized policy if we ignore correlation, 2) our optimal stationary randomized policy which solves (7.5), 3) the max-Age-first or greedy policy which

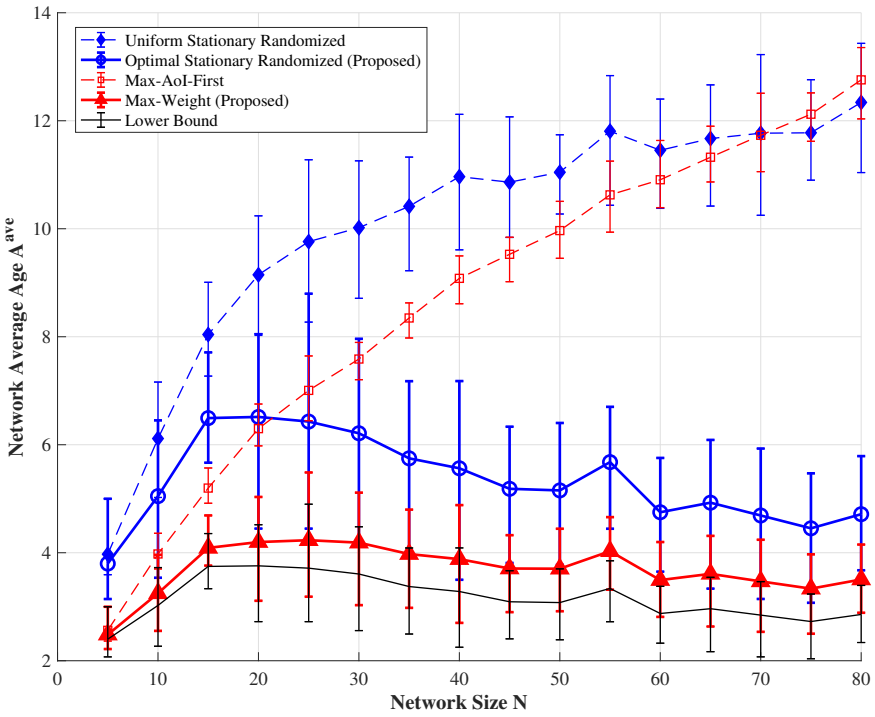


Figure 7-4: Average AoI vs network size N for hyperbolic geometric graphs.

would have been the optimal max-weight policy if we ignore correlation and 4) our proposed max-weight policy from Sec. 7.2.B.

Figure 7-3 plots the performance of these four policies as N increases. Each data point is computed by averaging over 50 random graph instances and running the policy for 15000 time-slots for each such instance. We also plot a lower bound for average AoI computed using (7.53) from Appendix 7.8.C. We observe that our proposed methods clearly outperform policies that ignore the correlation between sources for scheduling. In particular, our proposed max-weight policy outperforms max-AoI-first by almost 33%.

Next, we perform the same exercise for hyperbolic geometric graphs. Hyperbolic geometric graphs are generated by choosing points on a two-dimensional hyperbolic space and connecting vertices that are closer than the distance R where

distance is measured along hyperbolic geodesics. The most important feature of hyperbolic random graphs is that when their parameters are chosen appropriately, the degree distribution of nodes becomes *scale-free* [134, 135]. Importantly, for our simulation setup, we expect that scale-free degree distributions lead to correlation matrices with lower vertex covering numbers. According to Theorem 25, this should lead to larger performance gaps between policies that consider correlation and policies that ignore it. Figure 7-4 confirms this: as the number of sources increase, the gap in performance between our proposed policies and policies that ignore correlation also increases.

Next, we compare the performance of the optimal stationary randomized policy and our proposed max-weight policy for different correlation models. Specifically, we consider three correlation models: 1) Bernoulli correlation: $X_{ij}(t) \sim Bern(p_{ij})$, which is the main focus of this work, 2) Constant correlation: $X_{ij}(t) = p_{ij}$ as introduced in Sec. 7.4, and 3) Uniform correlation: $X_{ij}(t) = p_{ij} + Unif\big([-0.1, 0.1]\big)$, i.e. correlation is chosen uniformly at random from the interval $[p_{ij}-0.1, p_{ij}+0.1]$.

Figure 7-5 compares the performance of the two policies (optimal stationary randomized and max-weight) on random geometric graphs $\mathcal{G}(N = 90, r = 0.25)$ while varying the correlation parameter p from 0.1 to 0.9 for the three different correlation models discussed above.

We observe that for the stationary randomized policy, the average AoI values are the same across correlation models for each value of p. This is consistent with the result we derived in Theorem 28. For the max-weight policy, the average AoI values are close to one another for each value of p. Further, as the correlation increases, the gap between the average AoI for different correlation models decreases. Combined with our observation from Sec. 7.4, where we showed that average AoI under stationary randomized policies is the same for a large class of correlation models, this supports the claim that our results are fairly robust to the way in which correlation is modeled. Another important observation from

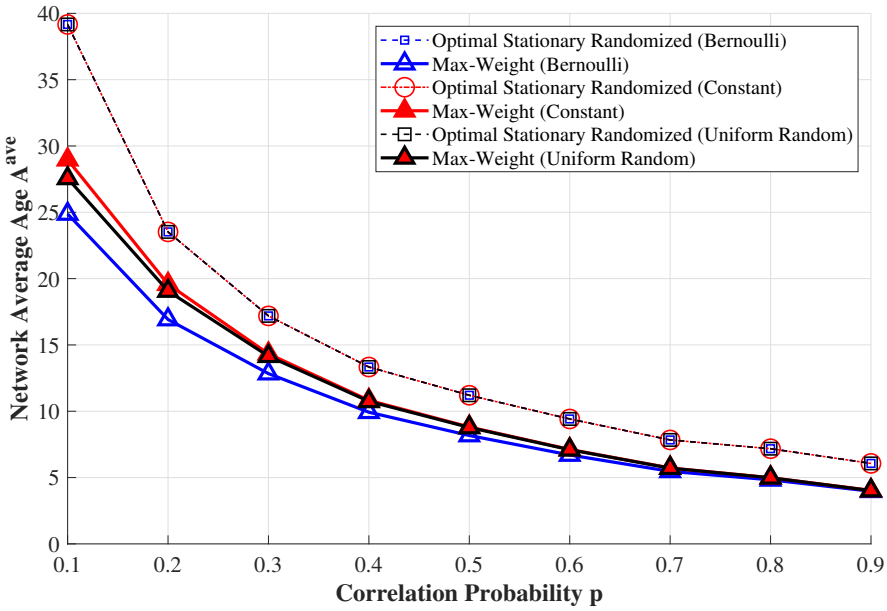


Figure 7-5: Average AoI vs correlation probability p.

Figure 7-5 is the inverse dependence of the average AoI on the correlation probability p, consistent with our upper-bound for random-geometric graphs from Theorem 27.

Finally, we consider a setting with time-varying correlation probabilities. Specifically, we consider a random geometric graph with $N = 90$ sources and a connectivity radius of 0.25 on the unit square $[0,1]^2$. However, unlike the previous simulations, we assume that these sources are *mobile* and move according to Brownian motion on $[0,1]^2$ with a maximum velocity of 0.01. As the sources move around randomly, the distances between them change and so does the corresponding correlation matrix $\mathbf{P}$. Figure 7-6 plots the performance of three different scheduling policies in this time-varying setting. To measure the performance, we consider a windowed time-average of the network AoI with a window size of 100. As the sources connectivity changes, we see the windowed time-average AoI change in response for each scheduling policy.

The three policies we consider are as follows. First, we consider the max-

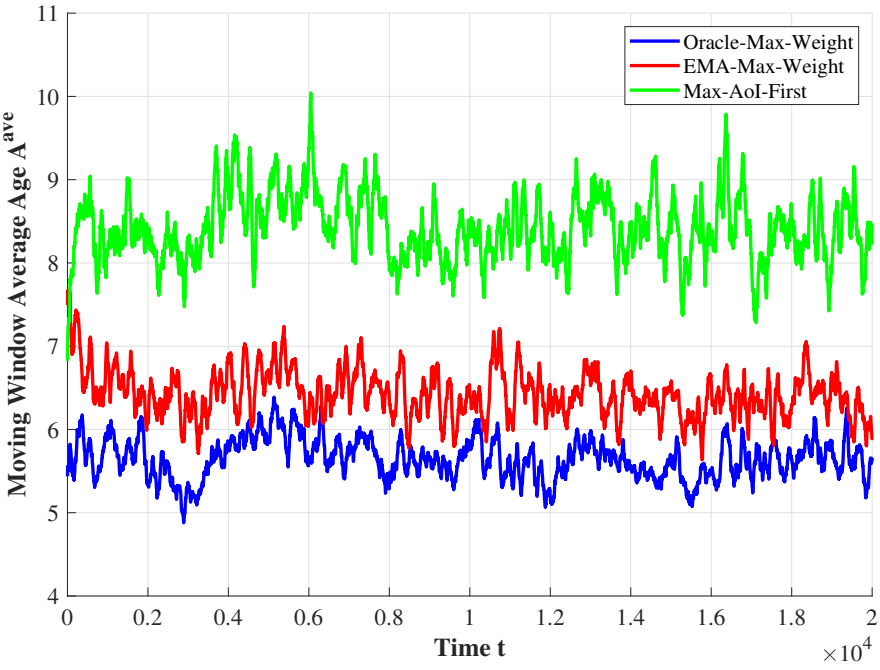


Figure 7-6: Moving window average AoI vs time.

AoI-first policy which is completely oblivious of the correlation while making scheduling decisions. Second, we consider the EMA-max-weight policy from Sec. 7.5.A which has no information about correlation in the beginning but gradually learns the matrix $\mathbf{P}$ and adapts to changes in it using an exponential moving average. Specifically, we set the learning rate $\alpha = 0.4$. Third, we consider the hypothetical *oracle-max-weight* policy. This is an omniscient policy that knows the current correlation matrix exactly in each time-slot and uses this information to run the quadratic max-weight scheduler (7.11).

We observe that EMA-max-weight is able to track the performance of the oracle-max-weight policy with only a small gap, indicating that it is able to learn the correlation structure and adapt in response to it. Further, max-AoI-first is not able to do so and has a larger gap in performance compared to the oracle-max-weight policy.

7.7 Summary

In this chapter, we formulated a simple model to study the timely monitoring of correlated sources over a wireless network. Using this model, we proposed new scheduling policies that optimize weighted-sum average Age of Information (AoI) in the presence of correlation. These policies have constant-factor optimality guarantees. We derived scaling results that illustrate how AoI improves in large networks in the presence of correlation and discussed how our model is relatively robust to correlation modeling assumptions. Lastly, We also developed a novel approach based on exponential moving averages that schedules correlated sources in a time-varying setting.

Important directions of future work involve proving performance bounds on EMA-max-weight under assumptions on how quickly the correlation matrices change, and incorporating more general AoI cost functions. A drawback that is worth mentioning is that we consider correlation to be a pairwise notion - information sharing happens only between pairs of sources. However, modeling more general notions of correlation and coupling between sources while still keeping analysis tractable is a challenging open problem.

7.8 Appendix

7.8.A Proof of Theorem 20

Consider a stationary randomized policy with the scheduling probabilities given by π. Then, the probability that the base station receives an update about source i in any time-slot t is given by

$$\sum_{j=1}^{N} \mathbb{P}(j \text{ has information about } i)\mathbb{P}(j \text{ transmits}) = \sum_{j=1}^{N} p_{ji}\pi_j. \tag{7.31}$$

For simplicity, we will refer to the quantity $\sum_{j=1}^{N} p_{ji}\pi_j$ as r_i. Clearly, if $r_i = 0$, then the base station never receives any information regarding source i and the average AoI $\bar{A}_i$ grows to be unbounded. This proves one part of Theorem 20.

Next, we focus on the case when $r_i > 0$. In each time-slot, the base station receives a new update about source i with probability r_i, *independent* of events in other time-slots. Thus, intervals between two consecutive update deliveries from source i to the base station are geometrically distributed i.i.d. random variables with the parameter r_i. Let $I_1, I_2, ...$, be i.i.d. geometric random variables that denote update inter-arrival periods for source i. Further, let K be the largest integer such that $\sum_{k=1}^{K} I_k \leq T$, i.e. there are K update deliveries over the first T time-slots. Note that K itself is a random variable. Then, the average AoI of source i is given by:

$$\begin{aligned}\bar{A}_i &= \limsup_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} A_i(t) = \limsup_{T\to\infty} \frac{1}{T}\sum_{k=1}^{K}\sum_{j=1}^{I_k} j \\ &= \limsup_{T\to\infty} \frac{1}{T}\sum_{k=1}^{K} \frac{I_k^2 + I_k}{2} = \frac{\mathbb{E}[I_1^2 + I_1]}{2\mathbb{E}[I_1]}.\end{aligned} \tag{7.32}$$

The last step above holds by applying the elementary renewal-reward theorem and the law of large numbers. I_1 is simply a geometric random variable with the parameter r_i. Thus, the equation above can be further simplified to:

$$\bar{A}_i = \frac{1}{2} + \frac{\mathbb{E}[I_1^2]}{2\mathbb{E}[I_1]} = \frac{1}{2} + \frac{(2 - r_i)/r_i^2}{2/r_i} = \frac{1}{r_i}. \tag{7.33}$$

We use the moments of geometric random variables to derive the expression above. Since $r_i = \sum_{j=1}^{N} p_{ji}\pi_j$, this completes the proof.

7.8.B Proof of Theorem 21

We will first show that the objective function of the optimization problem (7.5) is convex in the scheduling probabilities $\boldsymbol{\pi}$. To do so, we compute the Hessian of the average AoI of source i (given by $\bar{A}_i$) with respect to $\boldsymbol{\pi}$.

$$\begin{aligned}
\mathbf{H}(\bar{A}_i) &\triangleq \begin{bmatrix} \frac{\partial^2}{\partial \pi_1^2}\bar{A}_i & \frac{\partial^2}{\partial \pi_1 \partial \pi_2}\bar{A}_i & \cdots & \frac{\partial^2}{\partial \pi_1 \partial \pi_N}\bar{A}_i \\ \frac{\partial^2}{\partial \pi_2 \partial \pi_1}\bar{A}_i & \frac{\partial^2}{\partial \pi_2^2}\bar{A}_i & \cdots & \frac{\partial^2}{\partial \pi_2 \partial \pi_N}\bar{A}_i \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial \pi_N \partial \pi_1}\bar{A}_i & \frac{\partial^2}{\partial \pi_N \partial \pi_2}\bar{A}_i & \cdots & \frac{\partial^2}{\partial \pi_N^2}\bar{A}_i \end{bmatrix} \\
&= 2\bar{A}_i^3 \begin{bmatrix} p_{1i}^2 & p_{1i}p_{2i} & \cdots & p_{1i}p_{Ni} \\ p_{2i}p_{1i} & p_{2i}^2 & \cdots & p_{2i}p_{Ni} \\ \vdots & \vdots & \ddots & \vdots \\ p_{Ni}p_{1i} & p_{Ni}p_{2i} & \cdots & p_{Ni}^2 \end{bmatrix} \\
&= 2\bar{A}_i^3 \begin{bmatrix} p_{1i} \\ p_{2i} \\ \vdots \\ p_{Ni} \end{bmatrix} \begin{bmatrix} p_{1i} & p_{2i} & \cdots & p_{Ni} \end{bmatrix} \\
&= 2\bar{A}_i^3 \mathbf{P}_i \mathbf{P}_i^T .
\end{aligned} \tag{7.34}$$

Here, $\mathbf{P}_i$ is the i-th column of the correlation matrix $\mathbf{P}$. Note that any matrix of the form $\mathbf{v}\mathbf{v}^T$ where $\mathbf{v} \in \mathbf{R}^N$, is positive semi-definite since:

$$\mathbf{y}^T(\mathbf{v}\mathbf{v}^T)\mathbf{y} = (\mathbf{v}^T\mathbf{y})^2 \geq 0, \forall \mathbf{y} \in \mathbf{R}^N.$$

Thus, the hessian $\mathbf{H}(\bar{A}_i)$ with respect to the scheduling probabilities $\boldsymbol{\pi}$ is positive semi-definite, which implies that $\bar{A}_i$ is a convex function of $\boldsymbol{\pi}$. Since the objective function in (7.5) is simply a weighted sum of the average AoIs and the sum

of convex functions is convex, it is also a convex function of $\boldsymbol{\pi}$. Further, since the constraints of (7.5) are linear in $\boldsymbol{\pi}$, they are also trivially convex. Thus, the optimization problem (7.5) is convex as well. This proves the first part of Theorem 21.

Next, we apply KKT conditions to learn more about the structure of the optimal stationary randomized policy. To do so, we first formulate the Lagrangian function:

$$\mathcal{L}(\boldsymbol{\pi}, \lambda, \boldsymbol{\mu}) = \sum_{i=1}^{N} \frac{w_i}{\sum_{j=1}^{N} \pi_j p_{ji}} + \lambda \left(\sum_{i=1}^{N} \pi_i - 1 \right) - \sum_{i=1}^{N} \mu_i \pi_i.$$

Applying first-order KKT conditions, we see that:

$$\frac{\partial}{\partial \pi_i} \mathcal{L}(\boldsymbol{\pi}^*, \lambda^*, \boldsymbol{\mu}^*) = -\sum_{j=1}^{N} \frac{w_j p_{ij}}{(\sum_{k=1}^{N} \pi_k^* p_{kj})^2} + \lambda^* - \mu_i^* = 0, \forall i. \tag{7.35}$$

Now, assume that an optimal scheduling policy $\boldsymbol{\pi}^*$ is one that only schedules sources from the set S, where $S \subseteq [N]$ and $[N]$ is the set of all sources. Then, $\pi_i^* > 0, \forall i \in S$. Applying complementary slackness for this policy $\boldsymbol{\pi}^*$, we get that $\mu_i = 0, \forall i \in S$. Combining this with the first order condition above, we get:

$$\sum_{j=1}^{N} \frac{w_j p_{ij}}{(\sum_{k=1}^{N} \pi_k^* p_{kj})^2} = \lambda, \forall i \in S. \tag{7.36}$$

However, from Theorem 20, we know that the average AoI of source i under a policy $\boldsymbol{\pi}$ is given by $\frac{1}{\sum_{k=1}^{N} \pi_k p_{ki}}$. Thus, we can simplify the above equation to:

$$\sum_{j=1}^{N} p_{ij} w_j \bar{A}_j^2 = \lambda, \forall i \in S. \tag{7.37}$$

Here, $\bar{A}_i$ denotes the average AoI of source i under the policy $\boldsymbol{\pi}^*$. Since $\boldsymbol{\pi}^*$ is an optimal policy, the total utilization of the channel should be 100%. Thus,

$$\sum_{i \in S} \pi_i^* = 1. \tag{7.38}$$

Observe that equations (7.37) and (7.38) together form a system of $|S|+1$ equations in $|S|+1$ variables. Solving this system gives us the values of $\pi_i^*, \forall i \in S$. This completes our proof.

7.8.C Proof of Theorem 22

Consider a general scheduling policy π, that is not necessarily stationary randomized. We will restrict ourselves to policies that achieve finite average AoI for each source i, since we are interested in minimizing AoI and can safely ignore oplicies for which the average AoI is unbounded. Thus, we will assume that all expressions involving limits are well-defined throughout the proof.

Let $I_{k,i}$ denote the k-th inter-arrival time between update deliveries regarding source i to the base station. First, the average AoI of source i under this policy is given by:

$$\bar{A}_i \triangleq \lim_{T\to\infty}\left[\frac{\sum_{t=1}^{T} A_i(t)}{T}\right] = \lim_{K\to\infty}\left[\frac{\sum_{k=1}^{K} I_{k,i}^2 + I_{k,i}}{2(\sum_{k=1}^{K} I_{k,i})}\right]. \tag{7.39}$$

Next, define the following three three empirical average quantities:

$$\begin{aligned} \hat{I}_i &\triangleq \frac{\sum_{k=1}^{K} I_{k,i}}{K}, \\ \hat{I}_i^{(2)} &\triangleq \frac{\sum_{k=1}^{K} I_{k,i}^2}{K}, \\ \hat{Var}_i &\triangleq \hat{I}_i^{(2)} - (\hat{I}_i)^2. \end{aligned} \tag{7.40}$$

Then, using these definitions, we can simplify the expression for $\bar{A}_i$ as follows:

$$\bar{A}_i = \lim_{K\to\infty}\left[\frac{1}{2} + \frac{\hat{I}_i^{(2)}}{2\hat{I}_i}\right] = \lim_{K\to\infty}\left[\frac{1}{2} + \frac{(\hat{I}_i)^2 + \hat{Var}_i}{2\hat{I}_i}\right]. \tag{7.41}$$

Using the Cauchy-Schwarz inequality, it is easy to see that $\hat{Var}_i \geq 0$. Thus, we can

lower-bound the average AoI for source i by:

$$\bar{A}_i \geq \frac{1}{2} + \lim_{K\to\infty} \frac{\hat{I}_i}{2}. \tag{7.42}$$

Now, define f_j to be the fraction of time that the policy π schedules a source j on average. Further, define r_j to be the fraction of time that the base-station received a delivery about source j. Since there are correlated updates, $r_j \geq f_j$. Let $u_j(t)$ be an indicator variable that denotes whether policy π chooses source j in time-slot t or not and $X_{ji}(t)$ denote whether source j has a correlated update about source i at time t. Then,

$$r_i \triangleq \lim_{T\to\infty} \frac{\sum_{t=1}^{T}\sum_{j=1}^{N} u_j(t)X_{ji}(t)}{T}. \tag{7.43}$$

Let the set of time-slots in which source j is scheduled be $T_j = \{t : u_j(t) = 1\}$. Using this definition, we rewrite the equation above as:

$$r_i = \lim_{T\to\infty} \sum_{j=1}^{N} \frac{\sum_{t\in T_j} X_{ji}(t)}{T} = \lim_{T\to\infty} \sum_{j=1}^{N} \frac{|T_j|}{T} \frac{\sum_{t\in T_j} X_{ji}(t)}{|T_j|}. \tag{7.44}$$

Note that $\lim_{T\to\infty} \frac{|T_j|}{T}$ is simply f_j. Further, if $f_j > 0$, then as $T \to \infty$, $|T_j|$ must also go to infinity and we can apply the law of large numbers to get $\lim_{T\to\infty} \frac{\sum_{t\in T_j} X_{ji}(t)}{|T_j|} = p_{ji}$. If $f_j = 0$, the law of large numbers cannot be applied. However, it is easy to see that $\lim_{T\to\infty} \frac{\sum_{t\in T_j} X_{ji}(t)}{T} = 0 = f_j p_{ji}$. Thus, the expression for r_i simplifies to:

$$r_i = \sum_{j=1}^{N} f_j p_{ji}. \tag{7.45}$$

Another way to calculate r_i is to consider the following limit:

$$r_i = \lim_{K\to\infty} \frac{K}{\sum_{k=1}^{K} I_{k,i}}. \tag{7.46}$$

Thus, if $\lim_{K\to\infty} \hat{I}_i$ is well defined, then r_i is also given by:

$$r_i = \frac{1}{\lim_{K\to\infty} \hat{I}_i}. \tag{7.47}$$

Combining (7.45) and (7.47), we get:

$$\lim_{K\to\infty} \hat{I}_i = \frac{1}{\sum_{j=1}^{N} f_j p_{ji}}. \tag{7.48}$$

Combining (7.42) and (7.48), we get:

$$\sum_{i=1}^{N} w_i \bar{A}_i \geq \sum_{i=1}^{N} \frac{w_i}{2} + \frac{1}{2} \sum_{i=1}^{N} \frac{w_i}{\sum_{j=1}^{N} f_j p_{ji}}. \tag{7.49}$$

Note that the frequencies of transmission for each source f_j satisfy the following two constraints: $\sum_{j=1}^{N} f_j \leq 1$ and $f_j \geq 0, \forall j$. Thus, we can further lower-bound the weighted-sum average AoI by minimizing the RHS in (7.49) over all choices of transmission frequencies:

$$\sum_{i=1}^{N} w_i \bar{A}_i \geq \sum_{i=1}^{N} \frac{w_i}{2} + \frac{1}{2} \sum_{i=1}^{N} \frac{w_i}{\sum_{j=1}^{N} f_j^* p_{ji}}. \tag{7.50}$$

Here $\boldsymbol{f}^*$ is the solution to the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{f}}{\text{argmin}} & \sum_{i=1}^{N} \frac{w_i}{\sum_{j=1}^{N} f_j p_{ji}} \\ \text{s.t.} & \sum_{i=1}^{N} f_i \leq 1, f_i \geq 0, \forall i \in [N]. \end{aligned} \tag{7.51}$$

Observe that this optimization problem is identical to the (7.5) which solves for

the optimal stationary randomized policy π^*. Thus, $\boldsymbol{f}^* = \pi^*$ and we get:

$$\sum_{i=1}^{N} w_i \bar{A}_i \geq \sum_{i=1}^{N} \frac{w_i}{2} + \frac{1}{2} \sum_{i=1}^{N} \frac{w_i}{\sum_{j=1}^{N} \pi_j^* p_{ji}}. \tag{7.52}$$

Note that (7.52) is true for any scheduling policy π with finite average AoI for each source. So, it is also true for the policy π^{opt} that achieves minimum weighted-sum average AoI. This is because a simple round-robin policy achieves finite AoI for each source, so the optimal policy which performs at least as well as the round-robin policy, must also have well-defined and bounded average AoI for each source. Let's denote the average AoI for source i under the overall optimal policy π^{opt} by $\bar{A}_i^{opt}$ and the average AoI under the optimal stationary randomized policy π^* by $\bar{A}_i^*$. Then, we get:

$$\sum_{i=1}^{N} w_i \bar{A}_i^{opt} \geq \sum_{i=1}^{N} \frac{w_i}{2} + \frac{1}{2} \sum_{i=1}^{N} w_i \bar{A}_i^*. \tag{7.53}$$

Dividing, both sides in (7.53) by the LHS and multiplying by 2, we get:

$$\begin{aligned} 2 - \frac{\sum_{i=1}^{N} w_i}{\sum_{i=1}^{N} w_i \bar{A}_i^{opt}} &\geq \frac{\sum_{i=1}^{N} w_i \bar{A}_i^*}{\sum_{i=1}^{N} w_i \bar{A}_i^{opt}}, \\ \implies \frac{\sum_{i=1}^{N} w_i \bar{A}_i^*}{\sum_{i=1}^{N} w_i \bar{A}_i^{opt}} &\leq 2 \end{aligned} \tag{7.54}$$

This completes our proof.

7.8.D Proof of Theorem 23

Consider the one-slot Lyapunov drift for the quadratic Lyapunov function described by (7.9):

$$\begin{aligned}\mathbb{E}\Big[L(t+1)-L(t)\Big|\mathbf{A}(t)\Big] &= \mathbb{E}\Big[\sum_{i=1}^{N} w_i\big(A_i^2(t+1)-A_i^2(t+1)\big)\Big]\\ &= \sum_{i=1}^{N} w_i\Bigg(2A_i(t)+1-\sum_{j=1}^{N}\mathbb{E}\Big[u_j(t)X_{ji}(t)\Big|A_i(t)\Big]A_i(t)(A_i(t)+2)\Bigg)\\ &= \sum_{i=1}^{N} w_i\Bigg(2A_i(t)+1-\sum_{j=1}^{N} p_{ji}\mathbb{E}\Big[u_j(t)\Big|A_i(t)\Big]A_i(t)(A_i(t)+2)\Bigg).\end{aligned} \tag{7.55}$$

Since the quadratic max-weight policy minimizes the Lyapunov drift in every time-slot, we can upper-bound its one-slot drift by that of any other scheduling policy. In particular, we choose the optimal stationary randomized scheduling policy $\boldsymbol{\pi}^*$, for which we know that $\mathbb{E}[u_j(t)|A_i(t)] = \pi_j^*, \forall i,j,t$. Using this we upper-bound the one-slot Lyapunov drift as follows:

$$\begin{aligned}&\mathbb{E}\Big[L(t+1)-L(t)\Big|\mathbf{A}(t)\Big]\\ &\le \sum_{i=1}^{N} w_i\Bigg(2A_i(t)+1-\sum_{j=1}^{N} p_{ji}\pi_j^* A_i(t)(A_i(t)+2)\Bigg)\\ &= \sum_{i=1}^{N} w_i\Big(2A_i(t)+1-r_i A_i(t)(A_i(t)+2)\Big)\\ &= -\sum_{i=1}^{N} w_i r_i\Big(A_i(t)-\frac{1}{r_i}+1\Big)^2+\sum_{i=1}^{N} w_i\Bigg(r_i\Big(\frac{1}{r_1}-1\Big)^2+1\Bigg)\end{aligned} \tag{7.56}$$

Here, $r_i \triangleq \sum_{j=1}^{N} p_{ji}\pi_j^*$, ad before. Next, using the Cauchy Schwarz inequality we get:

$$\Bigg(\sum_{i=1}^{N} w_i r_i\Big(A_i(t)-\frac{1}{r_i}+1\Big)^2\Bigg)\Bigg(\sum_{i=1}^{N}\frac{w_i}{r_i}\Bigg) \ge \Bigg(\sum_{i=1}^{N} w_i\Big|A_i(t)-\frac{1}{r_i}+1\Big|\Bigg)^2. \tag{7.57}$$

Using the above inequality in (7.56), we get:

$$\mathbb{E}\Big[L(t+1)-L(t)\Big|\mathbf{A}(t)\Big] \leq -\left(\sum_{i=1}^{N} w_i\left|A_i(t)-\frac{1}{r_i}+1\right|\right)^2\left(\sum_{i=1}^{N}\frac{w_i}{r_i}\right)^{-1}+\sum_{i=1}^{N} w_i\left(r_i\left(\frac{1}{r_i}-1\right)^2+1\right). \tag{7.58}$$

Define $\Delta\big(\mathbf{A}(t)\big) \triangleq \mathbb{E}[L(t+1)-L(t)|\mathbf{A}(t)]$. Rearranging, we get:

$$\left(\sum_{i=1}^{N} w_i\left|A_i(t)-\frac{1}{r_i}+1\right|\right)^2 \leq -\left(\sum_{i=1}^{N}\frac{w_i}{r_i}\right)\Delta\big(\mathbf{A}(t)\big)+\left(\sum_{i=1}^{N}\frac{w_i}{r_i}\right)\sum_{i=1}^{N} w_i\left(r_i\left(\frac{1}{r_i}-1\right)^2+1\right). \tag{7.59}$$

Taking the expectation and summing over time, we get:

$$\begin{aligned}\sum_{t=1}^{T}\mathbb{E}\left[\left(\sum_{i=1}^{N} w_i\left|A_i(t)-\frac{1}{r_i}+1\right|\right)^2\right] \leq -\left(\sum_{i=1}^{N}\frac{w_i}{r_i}\right)\sum_{t=1}^{T}\mathbb{E}\Big[\Delta\big(\mathbf{A}(t)\big)\Big] \\ +T\left(\sum_{i=1}^{N}\frac{w_i}{r_i}\right)\sum_{i=1}^{N} w_i\left(r_i\left(\frac{1}{r_i}-1\right)^2+1\right).\end{aligned} \tag{7.60}$$

Dividing by T and applying Jensen's inequality, we get:

$$\begin{aligned}&\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left(\sum_{i=1}^{N} w_i\left|A_i(t)-\frac{1}{r_i}+1\right|\right)\right]^2 \leq \\ &-\left(\sum_{i=1}^{N}\frac{w_i}{r_i}\right)\sum_{t=1}^{T}\mathbb{E}\left[\frac{L(T+1)-L(1)}{T}\right]+\left(\sum_{i=1}^{N}\frac{w_i}{r_i}\right)\sum_{i=1}^{N} w_i\left(r_i\left(\frac{1}{r_i}-1\right)^2+1\right) \\ &\leq \left(\sum_{i=1}^{N}\frac{w_i}{r_i}\right)\sum_{t=1}^{T}\mathbb{E}\left[\frac{L(1)}{T}\right]+\left(\sum_{i=1}^{N}\frac{w_i}{r_i}\right)\sum_{i=1}^{N} w_i\left(r_i\left(\frac{1}{r_i}-1\right)^2+1\right).\end{aligned} \tag{7.61}$$

Applying Jensen's inequality again, and using the fact that $r_i \leq 1 \forall i$, we get:

$$\begin{aligned}
&\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\left(\sum_{i=1}^{N} w_i\left|A_i(t)-\frac{1}{r_i}+1\right|\right)\right]^2 \\
&\leq \left(\sum_{i=1}^{N}\frac{w_i}{r_i}\right)\sum_{t=1}^{T}\mathbb{E}\left[\frac{L(1)}{T}\right]+\left(\sum_{i=1}^{N}\frac{w_i}{r_i}\right)\sum_{i=1}^{N} w_i\left(r_i\left(\frac{1}{r_i}-1\right)^2+1\right) \\
&\leq \left(\sum_{i=1}^{N}\frac{w_i}{r_i}\right)\sum_{t=1}^{T}\mathbb{E}\left[\frac{L(1)}{T}\right]+\left(\sum_{i=1}^{N}\frac{w_i}{r_i}\right)\sum_{i=1}^{N} w_i\left(r_i+\frac{1}{r_i}-1\right) \\
&\leq \left(\sum_{i=1}^{N}\frac{w_i}{r_i}\right)\sum_{t=1}^{T}\mathbb{E}\left[\frac{L(1)}{T}\right]+\left(\sum_{i=1}^{N}\frac{w_i}{r_i}\right)\sum_{i=1}^{N}\frac{w_i}{r_i}.
\end{aligned} \tag{7.62}$$

Taking the square-root of the inequality above and using the fact $L(1)$ is a constant, we take the limit as T goes to infinity to get:

$$\lim_{T\to\infty}\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\left(\sum_{i=1}^{N} w_i\left|A_i(t)-\frac{1}{r_i}+1\right|\right)\right] \leq \left(\sum_{i=1}^{N}\frac{w_i}{r_i}\right). \tag{7.63}$$

This inequality can be further simplified to:

$$\lim_{T\to\infty}\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N} w_i A_i(t)\right] \leq 2\left(\sum_{i=1}^{N}\frac{w_i}{r_i}\right). \tag{7.64}$$

However, note that $r_i = \sum_{j=1}^{N} p_{ji}\pi_j^*$. Thus, we get:

$$\sum_{i=1}^{N} w_i \bar{A}_i^{qmw} \leq 2\sum_{i=1}^{N} w_i \bar{A}_i^*, \tag{7.65}$$

where $\bar{A}_i^*$ is the average AoI of source i under the optimal stationary randomized policy π^*. Since we have already shown factor-2 optimality of π^* in Appendix 7.8.C, this completes the proof.

7.8.E Proof of Theorem 24

We first rewrite the correlated AoI evolution (7.1) below:

$$A_i(t+1) = A_i(t) + 1 - A_i(t)\sum_{j=1}^{N} u_j(t)X_{ji}(t), \forall i,t. \tag{7.66}$$

Here, $u_j(t)$ indicates whether source j transmits in time-slot t or not and $X_{ji}(t)$ denotes whether source j receives information about source i at time-stop t or not. Using this evolution and the definition of the Lyapunov function (7.14), we calculate the one-slot Lyapunov drift:

$$\begin{aligned}\mathbb{E}\Big[L(t+1) - L(t)\Big|\mathbf{A}(t)\Big] &= \mathbb{E}\Big[\sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\alpha_i A_i(t)\sum_{j=1}^{N}u_j(t)X_{ji}(t)\Big|\mathbf{A}(t)\Big] \\ &= \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\alpha_i A_i(t)\sum_{j=1}^{N}\mathbb{E}\big[u_j(t)|\mathbf{A}(t)\big]p_{ji}.\end{aligned} \tag{7.67}$$

Since the max-weight policy attempts to minimize the one-slot Lyapunov drift in every time-slot, the corresponding drift of any other policy must be greater than or equal to that of the max-weight policy. So, we can upper-bound the drift of max-weight by the drift of the optimal stationary randomized policy. Note that for the optimal randomized policy π^*, the following holds:

$$\mathbb{E}\big[u_j(t)|\mathbf{A}(t)\big] = \pi_j^*, \forall j,t$$

since scheduling decisions are independent of the AoI values and across time-slots. Using this, we now upper-bound the one-slot drift of the max-weight policy as follows:

$$\mathbb{E}\Big[L(t+1) - L(t)\Big|\mathbf{A}(t)\Big] \leq \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\pi_j^* p_{ji}A_i(t). \tag{7.68}$$

Taking expectations and summing (7.68) from $t = 1$ to T, we get:

$$\mathbb{E}\Big[L(T+1) - L(1)\Big] \leq T\sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\left(\mathbb{E}\Big[\sum_{t=1}^{T}A_i(t)\Big]\sum_{j=1}^{N}\alpha_i\pi_j^* p_{ji}\right). \tag{7.69}$$

Dividing (7.69) by T, and re-arranging the terms, we get:

$$\frac{1}{T}\sum_{i=1}^{N}\left(\mathbb{E}\Big[\sum_{t=1}^{T}A_i(t)\Big]\sum_{j=1}^{N}\alpha_i\pi_j^* p_{ji}\right) \leq \sum_{i=1}^{N}\alpha_i + \frac{\mathbb{E}[L(1) - L(T+1)]}{T}. \tag{7.70}$$

Since $L(T+1) \geq 0$ and $\alpha_i = \frac{w_i}{\sum_{j=1}^{N}\pi_j^* p_{ji}}$, we get:

$$\frac{1}{T}\sum_{i=1}^{N}\mathbb{E}\Big[\sum_{t=1}^{T}w_i A_i(t)\Big] \leq \sum_{i=1}^{N}\frac{w_i}{\sum_{j=1}^{N}\pi_j^* p_{ji}} + \frac{\mathbb{E}[L(1)]}{T}. \tag{7.71}$$

Taking the limit as T goes to infinity, we get the following:

$$\sum_{i=1}^{N} w_i \bar{A}_i^{mw} \leq \sum_{i=1}^{N} w_i \bar{A}_i^{*}. \tag{7.72}$$

Here $\bar{A}_i^{mw}$ is the average AoI of source i under the max-weight policy while $\bar{A}_i^{*}$ is the average AoI of source i under the optimal stationary randomized policy $\boldsymbol{\pi}^*$.

Note that we had already proved the factor-2 optimality of $\boldsymbol{\pi}^*$. Thus, (7.72) is sufficient to obtain the same factor-2 optimality for the max-weight policy as well. This completes the proof.

7.8.F Proof of Theorem 25

Consider N sources with the correlation matrix $\mathbf{P}$. Given a correlation threshold $p > 0$, construct a directed graph $\mathcal{G}$ that represents pairs of source with correlation higher than the threshold.

Further, assume that the set $S \subseteq [N]$ is a minimum size vertex covering of the

graph $\mathcal{G}$. Recall that we defined the notion of vertex covering for directed graphs in Sec. 7.3. Let the size of this minimum vertex covering be denoted by N_{cov}.

We will show that under a specific scheduling policy π, the average AoI of every source is upper-bounded by $\frac{N_{cov}}{p}$. Consequently, the weighted sum of average AoIs under an optimal policy will also be upper-bounded by $\frac{N_{cov}}{p}$, since we assumed equal weights. The scheduling policy π we analyze is round-robin scheduling of sources in the covering set S. This is a cyclic policy of length N_{cov} time-slots where each source in the covering set S gets scheduled once, after which the scheduling pattern repeats every N_{cov} time-slots.

Before we analyze the performance of this policy, we present a lemma that discusses the monotonicity of AoI with the correlation parameters.

Lemma 13. *Consider N sources with two different correlation matrices $\mathbf{P}$ and $\mathbf{P'}$. If $p_{ij} \geq p'_{ij}, \forall i,j \in [N]$ then under a fixed scheduling policy π, the following holds:*

$$\bar{A}_i \leq \bar{A}'_i, \forall i \in [N]. \tag{7.73}$$

Here, $\bar{A}_i$ is the average AoI of source i under policy π with the correlation matrix $\mathbf{P}$, while $\bar{A}'_i$ is the average AoI of source i under policy π with the correlation matrix $\mathbf{P'}$.

Proof. The proof is easy to see via a stochastic dominance argument. Let $X_{ji}(t)$ be an indicator variable denoting whether j had information about i at time-slot t given the correlation matrix $\boldsymbol{P}$, and likewise $X'_{ji}(t)$ for the matrix $\boldsymbol{P'}$. Then, for all pairs (i,j) and for all time-slots t, $X_{ij}(t) \sim Bern(p_{ij})$ and $X'_{ij}(t) \sim Bern(p'_{ij})$, where $p_{ij} \geq p'_{ij}$. Thus, $X_{ij}(t) \geq_{st.} X'_{ij}(t)$.

Now, the AoI evolution (7.1) can be rewritten as:

$$\begin{aligned} A_i(t+1) &= A_i(t) + 1 - A_i(t)\sum_{j=1}^{N} u_j(t)X_{ji}(t), \text{ and} \\ A_i'(t+1) &= A_i'(t) + 1 - A_i'(t)\sum_{j=1}^{N} u_j'(t)X_{ji}'(t), \forall i, t. \end{aligned} \tag{7.74}$$

Since we have fixed the policy π, the scheduling decisions remain the same for both correlation matrices, i.e. $u_j(t) = u_j'(t), \forall t$. Setting $A_i(1) = A_i'(1)$, (7.74) implies that:

$$A_i(t) \leq_{st.} A_i'(t). \tag{7.75}$$

This, in turn, further implies that:

$$\frac{1}{T}\sum_{t=1}^{T} A_i(t) \leq_{st.} \frac{1}{T}\sum_{t=1}^{T} A_i'(t), \forall i, t. \tag{7.76}$$

Using the equation above and the definition of average AoI (7.2), we can conclude that:

$$\bar{A}_i \leq_{st.} \bar{A}_i', \forall i \in [N]. \tag{7.77}$$

If the average AoI limits exist, then $\bar{A}_i$ and $\bar{A}_i'$ are just constants and the stochastic dominance becomes a simple inequality. This completes the proof. □

The lemma above shows a rather simple result: AoI improves with correlation. To upper-bound the average AoI of sources under policy π, it is sufficient to analyze AoI for a new correlation matrix $\mathbf{P}'$, which is defined as:

$$p_{ij}' = \begin{cases} p, & \text{if } p_{ij} \geq p. \\ 0, & \text{otherwise.} \end{cases} \tag{7.78}$$

Clearly, $\mathbf{P}'$ is element-wise smaller than $\mathbf{P}$, so we can apply Lemma 13. Now, to

analyze AoI of a source i under the policy π with the correlation matrix $\mathbf{P}'$, we consider two cases.

Case 1: Source i is in the vertex covering set S. In this case, note that this source is guaranteed to be scheduled at least once in every N_{cov} time-slots due to way round-robin scheduling works. It is possible that the base station might receive a correlated update about i from some other source, even between two consecutive updates from i that are N_{cov} time-slots apart. However, since we are trying to upper-bound the AoI, we can safely ignore these updates. Thus, the average AoI of source i can be upper-bounded by:

$$\bar{A}_i \leq \frac{\sum_{k=1}^{N_{cov}} k}{N_{cov}} \leq \frac{N_{cov}+1}{2} \leq \frac{N_{cov}}{p}. \tag{7.79}$$

The last inequality follows since $p \leq 1$ and $N_{cov} \geq 1$.

Case 2: Source i is not in the vertex covering set S. By the definition of a vertex covering, there must be at least one source $j \in S$ such that the edge (j,i) is in the graph $\mathcal{G}$ or alternatively, that $p'_{ji} = p$. This means that whenever the scheduling policy schedules source j, the base station also receives an update regarding source i with probability p. Note that the scheduling policy π schedules source j once every N_{cov} time-slots. Then, the time intervals between two successful correlated update deliveries regarding source i by source j are $N_{cov}I_1, N_{cov}I_2, ..., N_{cov}I_K$ where $I_1, I_2, ..., I_K$ are i.i.d. geometrically distributed random variables with parameter p. It is possible that there are other sources also correlated with source i that deliver correlated updates to the base-station between two consecutive correlated updates from source j. For the sake of upper-bounding the AoI we can safely ignore any such updates. Then, the average AoI of source i can be upper-

bounded by:

$$\begin{aligned}\bar{A}_i &\leq \lim_{K\to\infty} \frac{\sum_{k=1}^{K}(1+2+\ldots+N_{cov}I_k)}{\sum_{k=1}^{K} N_{cov}I_k} \\ &\leq \lim_{K\to\infty} \frac{\sum_{k=1}^{K} N_{cov}^2 I_k^2 + N_{cov}I_k}{2\sum_{k=1}^{K} N_{cov}I_k} \\ &\leq \frac{1}{2} + N_{cov} \lim_{K\to\infty} \frac{\sum_{k=1}^{K} I_k^2}{2\sum_{k=1}^{K} I_k}.\end{aligned} \tag{7.80}$$

Applying the law of large numbers and using the moments of a geometric random variable, we get:

$$\bar{A}_i \leq \frac{1}{2} + N_{cov}\frac{(2-p)/p^2}{2/p} \leq \frac{N_{cov}}{p} - \frac{N_{cov}-1}{2} \leq \frac{N_{cov}}{p}. \tag{7.81}$$

The last inequality follows since a vertex covering must have at least one vertex, i.e. $N_{cov} \geq 1$.

Together, (7.79) and (7.81) imply that the average AoI $\bar{A}_i$ for every source i under the vertex cover round-robin policy π is upper-bounded by $\frac{N_{cov}}{p}$. Clearly, the performance of the policy that achieves minimum weighted-sum average AoI must be better than that of π. This implies:

$$\sum_{i=1}^{N} w_i \bar{A}_i^{opt} \leq \sum_{i=1}^{N} w_i \bar{A}_i \leq \frac{N_{cov}}{p}. \tag{7.82}$$

The last inequality follows since we have set all weights w_i to be equal to $\frac{1}{N}$. This completes the proof.

7.8.G Proof of Theorem 26

We want to prove a lower bound on the performance of the max-AoI-first policy π^u. To do so, we start with $\boldsymbol{A}(1)$ such that $A_i(1) = i$. Clearly, the policy does not schedule source 1 at time $t = 1$ since it does not have the maximum AoI.

In fact, since sources $2, ..., N$ are uncorrelated, source 1 only has a chance to get scheduled once its AoI reaches N. This is because max-AoI-first over the set of uncorrelated sources $\{2, ...N\}$ is simply the round-robin policy and the round-robin policy over $N-1$ sources always has at least one source with AoI $N-1$ in any time-slot. Suppose at some time-slot t the AoI of source 1 hits N and so it gets scheduled, breaking the round-robin phase. Then, in the next time-slot, the AoI of source 1 will go down to 1 while the AoI of all other sources will be at least 1 and possibly greater (if 1 failed to send a correlated update). Assuming a tie-breaking rule that always prefers sources with higher indices, max-AoI-first again starts scheduling sources $2, ..., N$ since source 1 has the smallest AoI. Again, source 1 will only be scheduled once its AoI reaches N and during this period π^u will simply be round-robin.

Since all the sources $2, ..., N$ are correlated with source 1 with probability p, the AoI of source 1 during each round-robin phase grows as a geometric random variable with parameter p. Define f_i to be the fraction of time that the policy π^u schedules a source i on average. Then, we can upper-bound f_1 by looking at how often a geometric random variable with the parameter p is smaller than N.

$$f_1 \leq 1 - \mathbb{P}(Geo(p) \leq N-1) = (1-p)^{N-1}. \tag{7.83}$$

Note that the policy π^u is equivalent to round-robin over $2, ..., N$ the rest of the time, when 1 is not being scheduled. Thus, $f_i = f, \forall i \neq 1$. Further, since the policy π^u never idles, we know that:

$$f_1 + (N-1)f = 1 \tag{7.84}$$

Together, (7.83) and (7.84) imply the following:

$$f \geq \frac{1-(1-p)^{N-1}}{N-1} \triangleq f_{min}. \tag{7.85}$$

Next, define r_i to be the fraction of time that the base-station received a delivery about source i. In Appendix 7.8.C, we show the following for every policy with well-defined average AoI:

$$r_i = \sum_{j=1}^{N} f_j p_{ji}, \forall i \in [N]. \tag{7.86}$$

Putting in $f_i = f, \forall i \neq 1$ and using the correlation matrix $\mathbf{P}$, we get:

$$r_i = \begin{cases} 1 - f(N-1)(1-p), & \text{if } i = 1, \\ f\big(1-(N-1)p\big) + p, & \text{otherwise .} \end{cases} \tag{7.87}$$

We know from analysis in Appendix 7.8.C that for any scheduling policy π the following holds:

$$\sum_{i=1}^{N} w_i \bar{A}_i \geq \frac{1}{2} \sum_{i=1}^{N} \frac{w_i}{r_i}. \tag{7.88}$$

Applying the inequality (7.88) to the uncorrelated max-weight policy π^u and putting in values of r_i from (7.87), we get:

$$\begin{aligned} \sum_{i=1}^{N} w_i \bar{A}_i^u &\geq \frac{1}{2N} \frac{1}{1-f(N-1)(1-p)} + \frac{N-1}{2N} \frac{1}{f(1-(N-1)p)+p} \\ &\geq \frac{N-1}{2N} \frac{1}{f_{min}(1-(N-1)p)+p}. \end{aligned} \tag{7.89}$$

The last inequality above follows by assuming $p \geq \frac{1}{N-1}$ and using the fact that $\frac{1}{r_i}$ become monotone *increasing* functions of f so we can lower-bound the RHS by choosing the smallest possible value of f. Putting in the expression for f_{min} from (7.85), and simplifying we get:

$$\sum_{i=1}^{N} w_i \bar{A}_i^u \geq \frac{(N-1)^2}{2N - (N+1)(1-p)^{N-1}}. \tag{7.90}$$

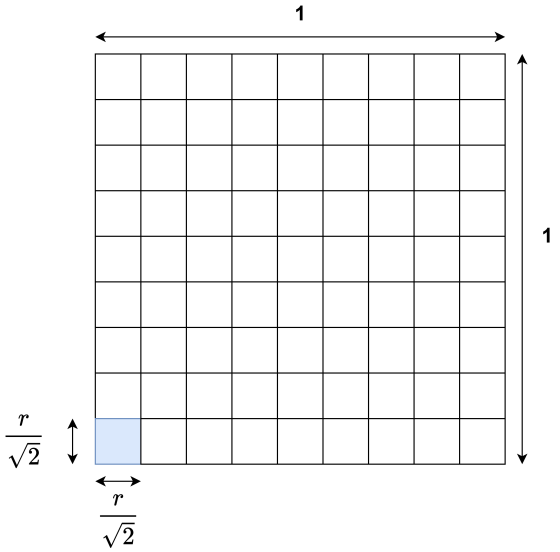


Figure 7-7: Grid of size $\frac{r}{\sqrt{2}}$ on the unit square.

This lower bound combined with the upper bound for the correlated max-weight policy completes the proof.

7.8.H Proof of Theorem 27

Consider a symmetric correlation matrix generated by creating a random geometric graph $\mathcal{G}(N,r)$ on the two dimensional unit square and setting correlation probabilities for neighbors to be p. Assume equal weights $w_i = \frac{1}{N}, \forall i$.

We will apply the vertex covering result from Theorem 25 to this geometric graph setting. Divide the unit square into square cells of size $r/\sqrt{2} \times r/\sqrt{2}$ (see Figure 7-7). For each cell on this grid, choose one source within the cell to be a member of the vertex covering set S. If there are no sources, ignore the cell and if there are more than one sources, pick one at random. Note that every source within one such cell is at most a distance r away from any other source within the same cell. Thus, all source pairs (i,j) within the same cell are connected on $\mathcal{G}(N,r)$ and must have correlation probabilities $p_{ij} = p_{ji} = p$.

It is easy to see that the set S, which consists of at most one unique source from

each cell, is a vertex covering on $\mathcal{G}(N,r)$. The size of the set S is upper-bounded by the total number of cells. Since the area of each cell is $\frac{r^2}{2}$, the total number of cells required to cover the unit square is $\frac{2}{r^2}$. Further, since all correlation probabilities are equal to p, we can use Theorem 25 on the graph $\mathcal{G}(N,r)$ with the correlation threshold p. This gives us:

$$\sum_{i=1}^{N} w_i \bar{A}_i^{opt} \leq \frac{N_{cov}}{p} \leq \frac{2}{pr^2}. \tag{7.91}$$

The first part of the inequality is a direct application of Theorem 25 while the second part holds since we have a found a vertex covering of size at-most $\frac{2}{r^2}$ which is an upper-bound on the size of the minimum vertex covering N_{cov}. This completes our proof.

7.8.I Proof of Theorem 28

Consider the AoIs evolution:

$$A_i(t+1) = A_i(t) + 1 - \left(\sum_{j=1}^{N} u_j(t) X_{ji}(t)\right) A_i(t), \forall i, t. \tag{7.92}$$

Taking conditional expectation with respect to $A_i(t)$ we get:

$$\mathbb{E}[A_i(t+1)|A_i(t)] = A_i(t) + 1 - \left(\sum_{j=1}^{N} \mathbb{E}[u_j(t)|A_i(t)] p_{ji}\right) A_i(t), \forall i, t. \tag{7.93}$$

Now, for a stationary randomized policy π we know that

$$\mathbb{E}[u_j(t)|A_i(t)] = \pi_j, \forall j, i, t.$$

Thus, we get:

$$\mathbb{E}[A_i(t+1)|A_i(t)] = A_i(t) + 1 - \left(\sum_{j=1}^{N} \pi_j p_{ji}\right) A_i(t), \forall i, t. \tag{7.94}$$

Denoting $\sum_{j=1}^{N} \pi_j p_{ji}$ by r_i, we get:

$$\mathbb{E}[A_i(t+1)|A_i(t)] = (1 - r_i) A_i(t) + 1, \forall t \in \mathbb{N}. \tag{7.95}$$

Observe that if $r_i = 0$ then $A_i(t)$ simply increases linearly with time and the average AoI is unbounded. This completes one part of the proof. For the remaining part, assume $r_i > 0$.

Taking expectation again, we get:

$$\mathbb{E}\big[\mathbb{E}[A_i(t+1)|A_i(t)]\big] = (1 - r_i)\mathbb{E}\big[\mathbb{E}[A_i(t)|A_i(t-1)]\big] + 1, \forall t \in \mathbb{N}. \tag{7.96}$$

Solving the recursion in the equation above we get:

$$\mathbb{E}[A_i(t+1)] = (1 - r_i)^t A_i(1) + \sum_{\tau=0}^{t-1} (1 - r_i)^\tau. \tag{7.97}$$

Summing up the above for $t = 0, \ldots, T-1$ we get:

$$\mathbb{E}\left[\sum_{t=1}^{T} A_i(t)\right] = \frac{1 - (1 - r_i)^T}{r_i} A_i(1) + \frac{T}{r_i} - \frac{1 - (1 - r_i)^T}{r_i^2}. \tag{7.98}$$

Dividing the equation above by T and taking the limit as T goes to infinity:

$$\lim_{T \to \infty} \mathbb{E}\left[\frac{\sum_{t=1}^{T} A_i(t)}{T}\right] = \frac{1}{r_i}. \tag{7.99}$$

If we consider a sequence of random variables $\frac{\sum_{t=1}^{T} A_i(t)}{T}$, then (7.99) shows that this sequence converges in expectation to $\frac{1}{r_i}$. However, convergence in expectation

implies convergence in probability. Thus, we get:

$$\lim_{T\to\infty} \frac{\sum_{t=1}^{T} A_i(t)}{T} \overset{p}{=} \frac{1}{r_i}. \tag{7.100}$$

Observe that the quantity on the LHS is the average AoI, so we have shown that:

$$\bar{A}_i = \frac{1}{\sum_{j=1}^{N} \pi_j p_{ji}}. \tag{7.101}$$

This completes our proof.

Chapter 8

WiSwarm: A System for AoI-based Scheduling

Emerging time-sensitive applications increasingly rely on collaborative multi-agent systems. Examples include: search and rescue missions using a team of unmanned aerial vehicles (UAVs), smart factories with connected automated machinery, and smart city intersections with connected self-driving cars. In such application domains, it is essential that agents communicate in a timely manner about changes in the environment and adapt their behavior accordingly. A major roadblock in deploying these applications in the real-world is that traditional communication networks were not designed to support large-scale multi-agent system that need to share time-sensitive information to collaborate effectively.

Our contributions: (1) Middleware design. We develop a networking middleware that makes WiFi networks customizable, allowing system designers to easily tailor WiFi to the needs of specific time-sensitive applications. Our networking middleware drives the underlying distributed WiFi network to behave as a network with centralized resource allocation and with custom queues at the sources. By controlling the storage and flow of information in the WiFi network, the middleware can: (i) prevent packet collisions; (ii) dynamically prioritize the

transmissions that are most valuable to the application; and (iii) discard stale packets that are no longer useful to the application before they are ever transmitted, thus alleviating congestion.

The networking middleware has two distinct features. First, it is *implemented at the application layer*, without modifications to lower layers of the networking protocol stack. The middleware runs over UDP/IP and standard 802.11 WiFi, making it easy to customize and integrate to existing time-sensitive applications that are already implemented using WiFi, such as [76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 88, 89, 90, 136]. Second, the middleware is *designed around the idea of information freshness*, specifically the **Age-of-Information (AoI) metric**. As we have seen, AoI captures the freshness of the information *from the perspective of the destination*, in contrast to the long-established packet delay that represents the latency of a *particular packet*. The resource allocation mechanism can leverage AoI to prioritize transmissions to destinations with stale information. Keeping information fresh is critical for time-sensitive applications, especially those that rely on cooperative multi-agent systems. **To the best of our knowledge, [8] was the first work to experimentally evaluate AoI-based resource allocation in a real-world time-sensitive application.**

Our contribution: (2) WiSwarm implementation. To demonstrate the performance improvement that can be achieved by customizing the WiFi network, we implement WiSwarm: an instantiation of the networking middleware for a mobility tracking application that relies on a collaborative UAV swarm. Following the recent growing interest in computational offloading to enhance the scale of multi-agent robotics applications [137], we implement a mobility tracking application composed of several small and inexpensive UAVs and one leader node with high compute power. Each UAV senses the environment (e.g., collects video) and transmits this contextual information to the leader node. The leader node consolidates the information from the UAVs and transmits trajectory updates

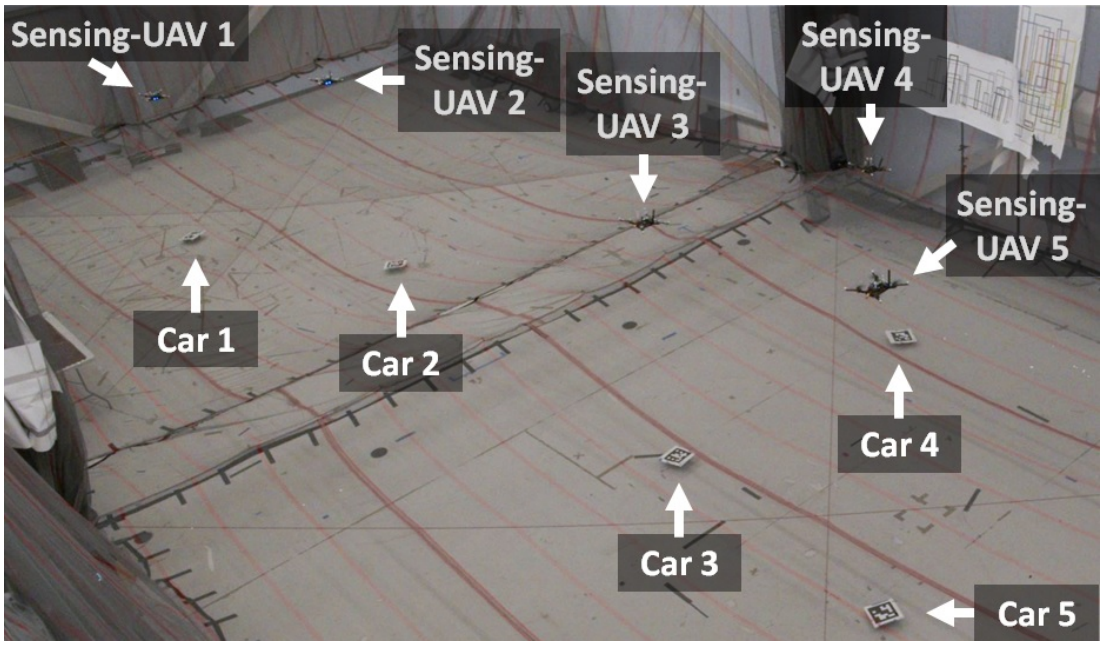


Figure 8-1: Setup for flight experiments with 5 UAVs.

that allow the UAVs to track the moving objects. Clearly, *it is essential to keep the contextual information at the leader node and the trajectory updates at every UAV as fresh as possible*, since outdated information loses its value and can lead to system failures (e.g., UAV losing track of an object) and safety risks (e.g., UAV collisions).

We evaluate WiSwarm in **flight experiments** [138] with up to five sensing-UAVs (see Fig. 8-1) and in **stationary experiments** with up to fourteen Raspberry Pis (RasPis) emulating UAVs. We collect data from nearly 4 hours of flight tests and 400 hours of stationary tests. Our experimental results show that WiSwarm significantly outperforms WiFi in terms of throughput, information freshness, tracking performance, and scalability. The stationary experiments with fourteen sources shows that WiSwarm improves information freshness by a factor of 50, and tracking error by a factor of 4. The flight tests show that mobility tracking with WiFi can support *at most two* sensing-UAVs while WiSwarm can support *at least five* sensing-UAVs under similar conditions.

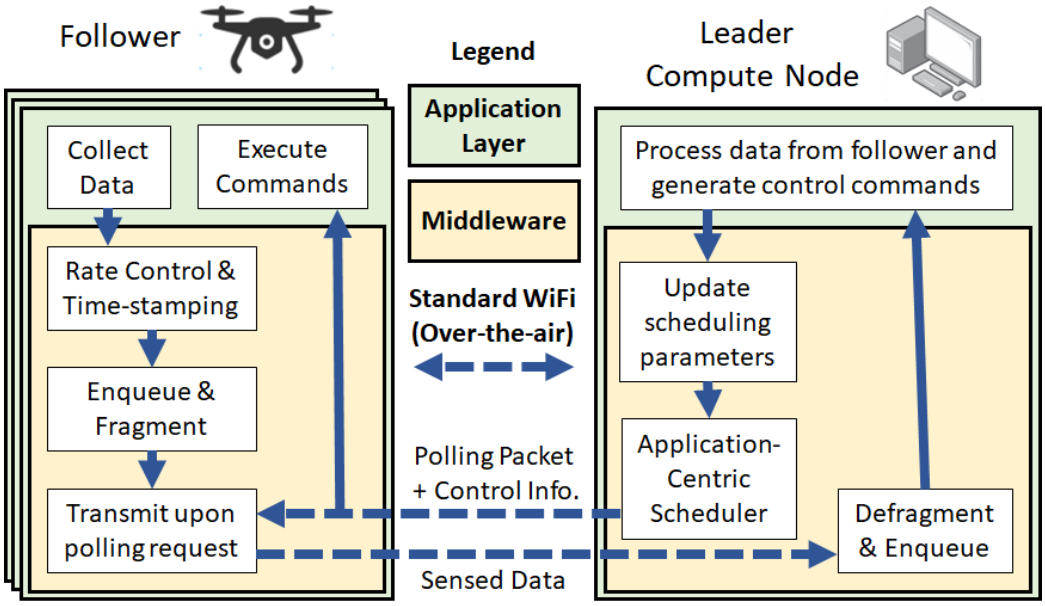


Figure 8-2: Overview of the Networking Middleware that provides information freshness for applications with a team of followers and a single leader.

8.1 Design

In this section, we describe the design of our networking middleware (illustrated in Fig. 8-2) that customizes WiFi to the needs of the important and broad class of time-sensitive applications that rely on multi-agent systems. In these applications, agents (also called followers) collect and transmit time-sensitive information to a central compute node (also called leader).

8.1.A Rate Control and Queuing

The *follower middleware* architecture illustrated on the left in Fig. 8-2 receives updates at rates that are determined by sensors/applications, then it time-stamps and enqueues these updates. The time-stamping is necessary to calculate AoIs at the leader from received updates. Upon receiving a polling request from the leader, the follower middleware releases a *single* update via UDP/IP to lower layers of the network protocol stack. The system designer can employ different

queueing disciplines to store/drop/prioritize updates aiming to satisfy the performance requirements of the application.

First-In First-Out (FIFO) queues are widely deployed in communication networks. Consider a middleware implementation utilizing FIFO queues to store information updates at the followers before transmitting them to the leader. If updates are generated at a low rate, then the leader would receive information updates too infrequently, resulting in outdated information and, thus, high AoI. On the other hand, if updates are generated at a very high rate, then the FIFO queue will often be backlogged and fresh updates will have to face large queueing delays before reaching the leader. This queueing delay leads to outdated information and, thus, high AoI. When FIFO queues are employed, it is imperative that the generation rate is carefully controlled to minimize AoI.

To adjust the update generation rate, the follower middleware employs a rate control mechanism that only updates its queue at fixed intervals of time, dropping any updates generated in between. This mechanism ensures that the middleware only accepts new updates at the desired rate and is crucial for controlling AoI in applications that use FIFO queues. Note that finding the optimal generation rate for a given network setup is a nontrivial task, as the optimal rate depends on the network's topology, traffic load, link reliability, and Medium Access Control (MAC) mechanism.

Last-In First-Out (LIFO) queues transmit the most recently generated update first, making them ideal for applications that rely on the knowledge of the *current state* of the system, such as mobility tracking. Consider a middleware implementation utilizing a LIFO queue. When an update is generated, the LIFO queue simply replaces the old head-of-line update with the fresh update. A higher update generation rate at the followers can only lead to fresher updates at the leader and, hence, a lower AoI. For this reason, LIFO queues eliminate the need for adjusting the update generation rate. LIFO queues have been shown to be *optimal* for min-

imizing AoI in a wide variety of network settings [14, 139]. However, LIFO queues are rarely implemented at the transport, MAC, or physical layers in practice.

In general, delivering *older* information updates to the leader after a *fresher* update was successfully received does not improve information freshness. Hence, discarding older updates when a fresh update arrives at the follower middleware could save network resources, alleviating congestion. On the other hand, older information may still be useful to the application. For example, in a mobility tracking application, older position information can be useful in predicting future movement. This trade-off should be considered by the system designer when deciding whether or not to discard older updates at the follower middleware.

The follower middleware receives updates from different sensors/applications and enqueues them according to the rules set forth by the system designer. Specifically, the system designer can choose between different queueing disciplines, e.g., LIFO, FIFO, Priority Queueing, and Fair Queueing, and set the rate at which this queue accepts new updates from the application. For example, encoded video that needs frames to be delivered sequentially can be stored in a FIFO queue, while raw video frames can be stored in a LIFO queue that keeps only the freshest update.

8.1.B Scheduling

Consider a multi-agent system with several followers transmitting time-sensitive information to the leader, while the leader coordinates the followers' behavior by transmitting control updates in a timely manner. The multiple access mechanism controls the method by which followers and leader share information using the limited communication resources. WiFi employs a distributed random access mechanism that works well for small-scale underloaded networks, but not for large-scale congested networks due to excess packet collisions that lead to

lower throughput and higher latency, which can result in degraded information freshness.

The design goals of the *leader middleware* are to: (i) prevent packet collisions; (ii) enable dynamic prioritization of the transmissions that are most valuable to the application; and (iii) facilitate integration with existing multi-agent systems that use WiFi. To achieve these goals, we propose the *leader middleware* illustrated on the right in Fig. 8-2. The middleware drives the underlying distributed WiFi network to behave as a centralized network with a polling multiple access mechanism. Specifically, the leader middleware coordinates the flow of information in the network by sending polling packets to the followers selected for transmission. At every decision time t, the leader selects the next follower to poll based on an application-centric *transmission scheduling policy* π, which can be a function of the current AoI of the followers $A_i(t)$, the reliability of the WiFi links $p_i(t)$, where $p_i(t) \in (0,1]$ represents the probability of a successful transmission from follower i to the leader, and the application-defined priority weights $w_i(t) \geq 0$, which represent the relative importance of each follower's information to the overall application goal. For example, in a mobility tracking application, the estimated velocities of the moving objects can be assigned as application weights $w_i(t)$, since faster objects may require more updates than slower objects in order to achieve the same tracking performance.

To capture application priorities and information freshness, we define the expected time-average of the weighted sum of AoIs across the entire network as

$$\frac{1}{T}\mathbb{E}\left[\sum_{i=1}^{N}\left(\int_{t=0}^{T} w_i(t)A_i(t)dt\right)\right], \tag{8.1}$$

where N is the number of followers, T is the time-horizon, and the expectation is with respect to the randomness in the link's reliability $p_i(t)$ and the policy π. To minimize (8.1), the scheduling policy π should attempt to improve information

freshness, i.e. reduce $A_i(t)$, where the application needs it the most, i.e. where $w_i(t)$ is higher.

In the previous chapters, we have studied the structure of scheduling policies that attempt to minimize objective functions of the form (8.1). A key take away from there is that, given the knowledge of the application weights $w_i(t)$, link reliabilities $p_i(t)$, and information freshness $A_i(t)$ of every follower i, *the Whittle's Index Policy is a near-optimal solution to the problem of minimizing* (8.1). The **Whittle's Index Policy** selects, at every decision time t, the follower i^* that satisfies

$$i^* \in \text{argmax}_i \left\{ w_i(t) p_i(t) A_i^2(t) \right\} , \tag{8.2}$$

with ties being broken arbitrarily. Intuitively, the Whittle's Index Policy is polling the followers associated with high application weights, reliable WiFi links, and outdated information at the leader.

In Chapter 3, we considered network settings with time-varying, unknown and even adversarial application weights $w_i(t)$ and showed that an online version of the Whittle's Index policy is near optimal. This suggests that Whittle's Index Policies are remarkably robust and can be applied to a wide variety of applications. Moreover, the Whittle's Index Policy has low computational complexity: it only requires solving the maximization in (8.2) and computing estimates of $w_i(t)$, $p_i(t)$, and $A_i(t)$.

8.1.C Middleware

We describe the networking middleware illustrated in Fig. 8-2, which incorporates both the application-centric queueing at the followers and transmission scheduling at the leader.

Followers collect information updates about their immediate environment (e.g., video, pictures, laser scans, and temperature) and about their own plat-

forms (e.g., position, attitude, velocity, and battery level). These updates are sent to the *follower middleware* to be prepared for transmission.

The rate control mechanism decides whether each update is discarded or enqueued. The follower middleware time-stamps each update that is not discarded at the time of collection and enqueues them. These time-stamps are used to compute $A_i(t)$ at the leader upon delivery. The queuing discipline, rate control mechanism, and queue buffer size can be specified by the system designer to satisfy the requirements of the application.

When the follower receives a polling packet, it releases a single information update from its queue. Assuming that the update does not exceed the maximum length of the UDP payload (or any threshold set by the system designer), upon being released from the queue, the update can be simply forwarded via UDP/IP to lower layers of the networking protocol stack. However, if the update is too large, then the middleware divides the update into *fragments*.

Fragments are stored in a separate FIFO queue and then transmitted one-by-one to the leader. Each fragment is transmitted via UDP/IP over standard WiFi. Since the maximum WiFi frame length can be smaller than the UDP payload size, it is possible that WiFi will require multiple successful over-the-air transmissions to deliver a single fragment to the leader. If WiFi fails to deliver a fragment, the middleware attempts to re-transmit the fragment using an error-control mechanism based on acknowledgements at the fragment level.

Leader's responsibilities include coordinating both the flow of information in the WiFi network and the followers' behavior. To do so, the leader manages the generation and transmission of *polling packets* and *control information*. Since follower's updates are transmitted only upon reception of a polling packet, the leader has almost full control of the flow of information in the WiFi network, irrespective of the number of followers and the amount of data they generate. This control allows the leader to alleviate congestion and prevent excessive packet col-

lisions in the WiFi network.

The leader uses the Whittle's Index Policy (8.2) to decide the next follower to poll. After transmitting a polling packet, the leader waits for the reception of a fragment. If this waiting period exceeds a timeout interval (e.g., 300 milliseconds), the attempt is assumed to have failed. Upon receiving a fragment or after a timeout, the leader starts preparing for the transmission of the next polling packet.

Prior to transmitting the next polling packet, the leader takes a series of steps that depend on whether the received fragment was the final fragment of an information update or not. If the received fragment from follower i was not the final, then the leader middleware simply updates $p_i(t)$. On the other hand, if the received fragment was the final, then the leader: (i) updates $p_i(t)$; (ii) combines fragments to obtain the original information update; (iii) extracts the associated time-stamp and updates $A_i(t)$; (iv) sends the information update to the application for processing; and (v) updates both $w_i(t)$ and the *control information* based on the results of this processing.

To estimate $p_i(t)$, the leader computes $\hat{p}_i(t) = D_i(t)/W$, where $D_i(t)$ is the number of polling packets which received a successful response from follower i out of the last W polling packets sent to it. To accurately compute $A_i(t) = t - \tau_i(t)$, where t is the current time measured by the leader and $\tau_i(t)$ is the largest time-stamp received from follower i, the clock at follower i should be synchronized with the leader's clock. The middleware performs periodic clock synchronization across all followers and the leader, at every 120 seconds using NTP [140]. Note that $A_i(t)$ is typically on the order of tens or hundreds of milliseconds. Thus the synchronization accuracy of NTP, which is around 1 millisecond for local area networks, is sufficient for our experiments.

After performing the necessary updates, the leader middleware transmits a polling packet to the selected follower. The latest control information is broadcast to all followers along with every polling packet.

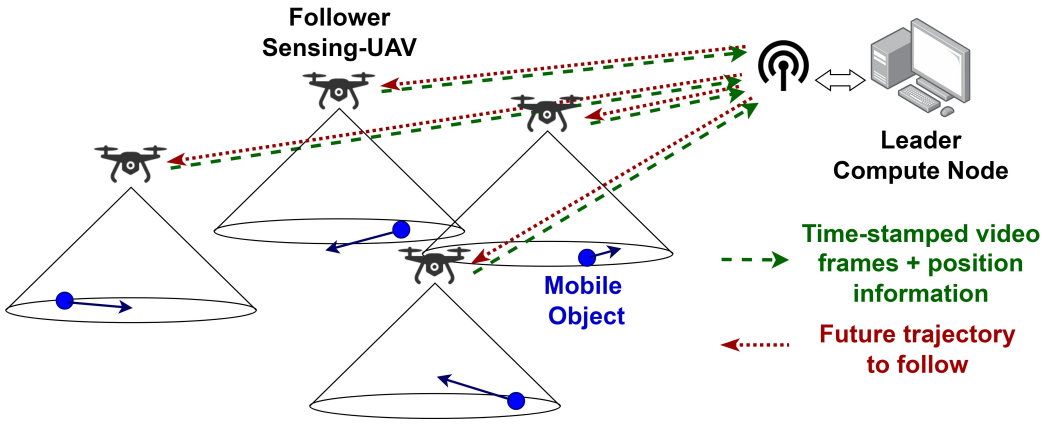


Figure 8-3: Mobility tracking using a swarm of sensor-UAVs and a leader compute node.

8.2 Implementation

In this section, we describe the design and implementation of WiSwarm which is an instantiation of the networking middleware for information freshness discussed in Sec. 8.1 tailored to a mobility tracking application.

8.2.A Mobility Tracking Application

Consider a setting where multiple UAVs are tracking moving objects on the ground. Clearly, outdated information about the position of the objects has a direct impact on the tracking capability of the UAVs. Ideally, the UAVs would like to receive fresh information about the objects continuously. One simple system design that achieves this goal consists of UAVs with high on-board computational power that are able to process video frames acquired from their cameras to detect and track objects. The continuous stream of images is processed locally, adding almost no delay, which keeps the UAVs updated about the position of the objects. A critical drawback of this approach is the prohibitively high cost of deploying numerous UAVs with high on-board computational power.

The separation of computing and sensing allows for more scalable system de-

sign - with one *leader-node* that has plenty of on-board computational power, and numerous low-cost *sensing-UAVs* that have little computational power but can effectively collect sensor data and communicate over a wireless network. Figure 8-3 illustrates an example of this system design approach. In general, the leader node could be a drone with a powerful on-board computer such as a Jetson TX2, a compute node located at the wireless edge, or even a cloud server performing high-speed inference and sending back control commands.

In our specific implementation of the mobility tracking application, the sensing-UAVs capture video of the immediate environment below them and send the captured video frames (without any pre-processing) to the leader compute node. The leader processes the received frames, infers the position of the objects, and sends trajectory updates to the sensing-UAVs via WiFi. *The main challenge of this design approach is to manage the limited wireless resources efficiently in order to keep information at the UAVs as fresh as possible.* WiSwarm, an instantiation of our networking middleware, ensures information freshness and scalable tracking performance by carefully controlling the flow of information over the network.

Next, we describe the different individual components involved in our application - the mobile objects to be tracked, the sensing-UAVs, the leader compute node. We also discuss how WiSwarm is implemented at the sensing-UAVs and the leader compute node.

8.2.B Mobile Objects

We use small autonomous cars equipped with RasPis (3B) as the moving objects whose mobility is tracked by the UAVs. Figure 8-4(b) shows one such car, with the ArUco marker tag on top, which is used for uniquely identifying and tracking the position of the cars by the leader compute node.

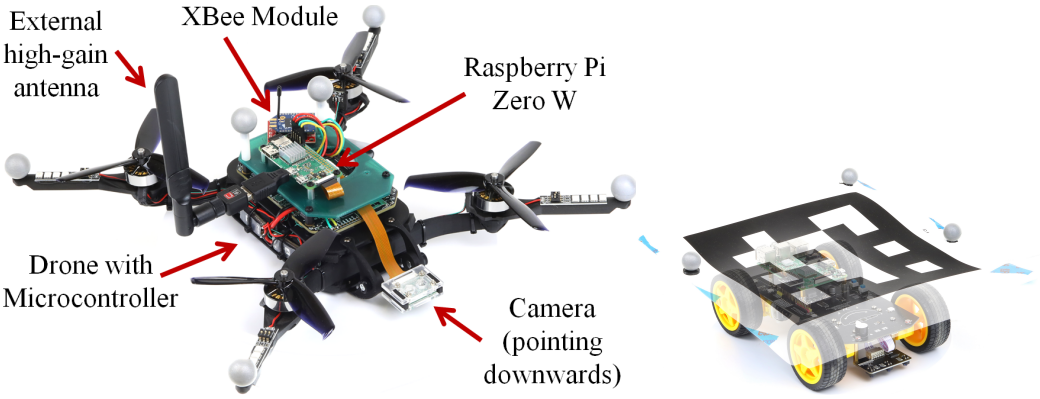


Figure 8-4: (a) Sensing-UAV. (b) Autonomous car with an identifying ArUco marker on top.

8.2.C Follower Sensing-UAVs

The sensing-UAV consists of two subsystems: (i) a quadcopter drone; and (ii) a RasPi (Zero W). Figure 8-4(a) shows a sensing-UAV with a RasPi on board the quadcopter drone, along with the sensing and communication peripherals we use for our application.

RasPi (Zero Ws) have very little computation capability (1 GHz single-core CPU and 512 MB RAM), but can effectively interact with multiple sensors and also communicate over WiFi. They are also extremely cost-efficient ($10). This makes them ideal for use in the sensing-UAVs. Each UAV is also equipped with a micro-controller unit (MCU) that runs state estimation and control algorithms. The state estimator combines measurements from an on-board inertial measurement unit (IMU) with global position and orientation measurements. These global measurements are obtained from a motion capture system and received by an Xbee WiFi module mounted on the vehicle. When motion capture data is not available, the Xbee module can be replaced by an alternative data source, such as a global navigation satellite system (GNSS) receiver.

The RasPi is connected to a camera that captures video of the area below the drone. Along with each frame, the RasPi also collects the position and orientation

at which the frame was collected by asking for this information from the MCU using an asynchronous serial connection. Following the discussion in Sec. 8.1, we know that fresh frames are the most useful for tracking, so we set the queuing discipline at the sensing-UAVs to be LIFO and the buffer size to be such that it can accommodate only one frame at a time.

The RasPi is connected to the leader compute node over 2.4 GHz WiFi using a high gain (8 dBi) antenna. Whenever WiSwarm at the RasPi receives a polling packet, it transmits the most recent update in its LIFO queue to the compute node. WiSwarm at the RasPi also collects the control information transmitted by the compute node. The control information contains the times and locations (in global coordinates) where the drone should be in the future in order to track the moving object. WiSwarm at the RasPi sends these waypoints over the serial connection to the drone MCU. The drone MCU then plans and executes a trajectory that reaches the specified waypoints at the specified future time instants. It does this by interpolating the waypoints to obtain a continuous trajectory that is followed using the flight control algorithm described in [141]. This completes the control loop.

8.2.D Leader Compute Node

WiSwarm at the compute node collects video-frames received from sensing-UAVs in response to polling requests. These video-frames are stored in separate LIFO queues - one for each sensing-UAV. At the application layer, the leader compute node runs an image processing thread. This processing thread goes over the queues maintained by WiSwarm in a round-robin manner and processes the received video-frames whenever it finds a non-empty queue.

For each sensing UAV, the thread attempts to locate the car that the UAV was assigned to track. If the car is found, it uses the relative location of the tag in the frame and the absolute position and orientation at which the frame was captured

to compute the global coordinates of the car. The processing thread also keeps a record of the last known locations of the car. Using the current location and previous known location of the car, the application computes: (i) the relative velocity between the car and the sensing UAV; and (ii) a list of future waypoints and the time-stamps at which it expects the car to reach these coordinates. In our implementation, we use a simple linear extrapolation scheme to predict future waypoints.

The application layer sends the control waypoints and time-stamps to WiSwarm along with information about the relative velocity between the car and the sensing-UAV. WiSwarm uses the relative velocity information to update its application-defined priority weights

$$w_i(t) \leftarrow \alpha w_i(t^-) + (1-\alpha)\hat{v}_i(t), \tag{8.3}$$

where $\hat{v}_i(t)$ is the estimate of relative velocity between the car and the associated sensing UAV, and $\alpha = 0.8$. Since velocity estimates are noisy and car velocities are time-varying, we use an exponential moving average motivated by the adaptive AoI-based scheduling algorithms proposed in Chapter 3. WiSwarm updates link reliabilities $p_i(t)$ by using the number of successful fragment deliveries, as described in Sec. 8.1.

With updated application weights $w_i(t)$ and link reliabilities $p_i(t)$, WiSwarm uses Whittle's Index Policy (8.2) to select the sensing-UAVs that need to be scheduled for transmission most urgently. Together with the unicast transmission of a polling packet, WiSwarm broadcasts the *most recent* list of future waypoints and time-stamps for every sensing-UAV. This repeated broadcast ensures redundancy in the delivery of control information.

8.3 Evaluation

In this section, we evaluate the performance of both WiFi and WiSwarm for the mobility tracking application. We perform our experiments in a *dynamic indoor campus space with multiple external sources of interference* such as WiFi base stations, mobile phones, and laptops. Throughout this section when we refer to WiFi, we mean 2.4 GHz WiFi.

In our evaluation, we consider two experimental setups: (i) **Stationary experiments**, which involve up to fourteen RasPis running an emulated version of the mobility tracking application and sending video-frames to a central Compute Node. These experiments involved hardware-in-the-loop and allowed us to test a variety of network sizes, update generation rates, scheduling policies, frame resolutions, packet sizes and interference conditions. (ii) **Flight experiments**, which involve interfacing the RasPis with UAVs and conducting real mobility tracking experiments. These allowed us to test how WiSwarm performs with mobile agents, at longer distances, and in the presence of significant interference. They also illustrate the drawbacks of using WiFi more clearly.

Baseline. To demonstrate the performance improvement of WiSwarm, we compare it with two baseline WiFi systems, namely WiFi-TCP and WiFi-UDP. Both systems collect video frames from the application layer at a fixed rate, packetize them, store them in FIFO queues, and send these packets over standard WiFi to the Compute Node. TCP uses its congestion control mechanism to adjust the number of packets in flight, while UDP simply forwards packets. In all of our stationary experiments, we found that accommodating the entire video frame within a single UDP packet (i.e., with no fragmentation) was the best choice in terms of tracking error.

For flight experiments, we consider an *optimized version of WiFi-UDP* as the baseline. Our flight tests showed that mobility tracking with WiFi-UDP and WiFi-

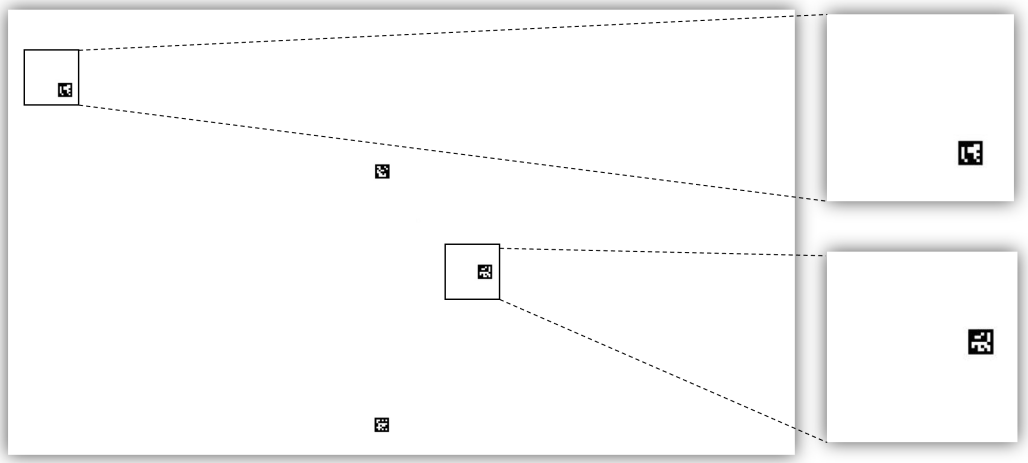
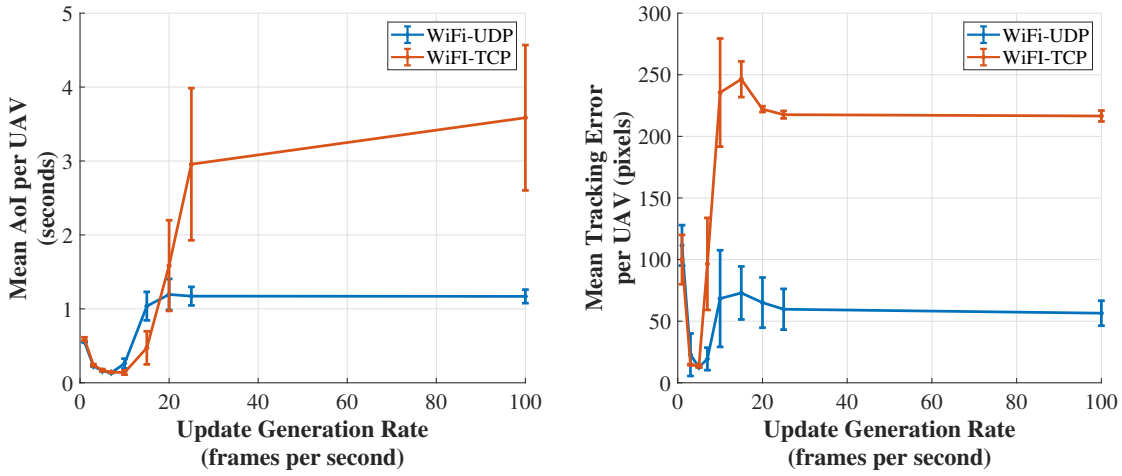


Figure 8-5: Screenshot from the videos used to simulate car movement during stationary experiments. The tags are programmed to perform random walks with time-varying velocities. The virtual UAVs need to keep track of the tags. On the right, two examples of 224x224 frames sent to the Compute Node by the RasPis based on their current virtual UAV locations.

TCP with fixed video frame rate (e.g., 50 fps) was not possible for more than a single sensing-UAV. To get mobility tracking to work with two sensing-UAVs, we had to carefully tune the frame generation rate (to 5 fps) and the UDP packet size (to 6 kB per fragment). This is due to the high congestion and unreliability caused by high generation rates and large packets, which caused tracking failures. Further, we also had to tune RTS/CTS thresholds. Despite all of this optimization, WiFi-UDP was only able to enable tracking for *at most two UAVs* at a time, as we show in the discussion on flight experiments.

8.3.A Stationary Experiments

In this section, we discuss the performance improvements of WiSwarm over WiFi for three different metrics: (i) AoI, (ii) throughput, and, most importantly, (iii) tracking error. Each data-point in the following discussion represents 16 minutes worth of experiments, split into 4 batches of 4 minutes each. We calculate the time-average of the performance metric over the entire 4 minutes of each batch



(a) (b)

Figure 8-6: (a) AoI and (b) tracking error of baseline WiFi-TCP and WiFi-UDP plotted against the update generation rate of each of the $N = 6$ emulated UAVs.

and then the mean and standard deviation across batches.

Experimental Setup. The experiments involve multiple RasPis running an emulated virtual UAV application. This application does two things. First, each RasPi has a video simulating the movement of cars stored on it. Using this video, the RasPis create cropped frames of size 224x224, based on the current location of the virtual UAV, which capture the local Field-of-View (FoV). These video frames are generated at a specified rate that can be set using the rate control mechanism, and are forwarded to WiFi or WiSwarm for delivery. The frames are stored as unencoded grayscale yuv images (1 byte per pixel), so each video frame is 49 kB in size. Second, the application decodes the control packets received from the Compute Node and updates the virtual UAV's location by moving between control waypoints at a specified speed. Figure 8-5 shows a frame from the video used for simulating movement of the car tags, along with two examples of 224x224 frames that the RasPis send to the Compute Node for processing.

Figure 8-6 plots the mean AoI and tracking error per UAV for both WiFi-TCP and WiFi-UDP as the frame generation rate at the RasPis increases. This plot is for a system with 6 transmitting RasPis. Note that lower AoI and lower tracking

error are preferred in terms of performance.

We make two important observations from Fig. 8-6. First, the performance of both WiFi-TCP and WiFi-UDP degrades when the generation rate is high, since the network becomes congested. Second, *WiFi needs optimization of the generation rate at the application layer* to be anywhere close to working in practice. This optimization is challenging since it needs to be at the application layer and also adjust quickly to changes in the traffic load and link reliability, which can vary due to external interference. This is true for both TCP and UDP, i.e. *TCP congestion control was unable to adjust to the optimal rate on its own.*

Next, we compare the performance of WiSwarm with both fixed-rate versions of WiFi and rate-optimized versions of WiFi. We choose the frame generation rates from the set $\{1, 3, 5, 7, 10, 15, 20, 25, 50, 100\}$ fps and the number of RasPis from the set $N \in \{2, 4, 6, 8, 10, 12, 14\}$. We find the best performing rates for each value of N from the rate set (based on tracking error).

To the best of our knowledge, there are no general purpose systems that can do application layer rate control for a wide variety of real-time applications, so the **rate-optimized WiFi systems are overly optimistic baselines**. Despite this, WiSwarm achieves significant performance gains over both fixed-rate and optimized rate versions of WiFi-TCP and WiFi-UDP.

AoI. Figure 8-7 plots the mean AoI per UAV as the system size N increases. More sources in the system means more congestion, more packet collisions (in WiFi) and hence poor performance and scalability. We see this clearly in Fig. 8-7(a), where we compare the baseline versions of WiFi-UDP and WiFi-TCP to WiSwarm. The baseline versions of WiFi have fixed update generation rate of 50 fps at each source while WiSwarm uses the maximum generation rate of 100 fps. Mean AoI improves by 16x for $N = 8$ and by almost 50x for $N = 14$ compared to fixed-rate WiFi. A major cause of the poor performance of WiFi is buildup of FIFO queues once the network becomes congested. Fixed-rate TCP eventually starts outper-

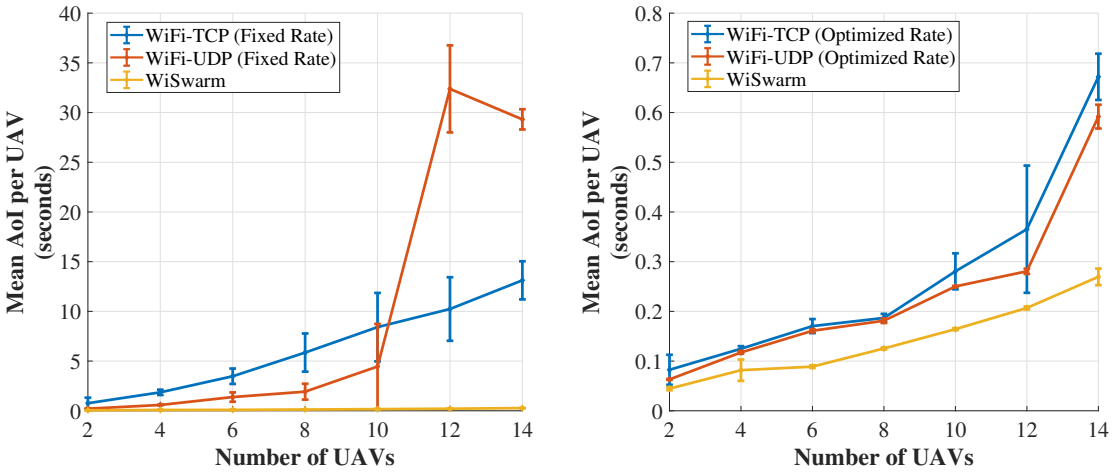


Figure 8-7: Mean AoI per UAV plotted for (a) fixed-rate (50 fps) and (b) optimized rate WiFi, as well as WiSwarm, as the number of UAVs increases.

forming fixed-rate UDP for larger N, due to its congestion control mechanism. WiSwarm does not suffer from the congestion problem due to the LIFO queues.

Figure 8-7(b) compares rate-optimized versions of WiFi-TCP and WiFi-UDP with WiSwarm. We observe that mean AoI still improves by 1.5x for $N = 8$ and 2.2x for $N = 14$. While the FIFO queues in WiFi are no longer congested due to careful tuning of the frame generation rates, there are still packet collisions due to the distributed nature of the CSMA protocol and external interference sources. WiSwarm avoids these collisions by centralizing medium access scheduling decisions and prioritizing sources with higher AoI.

Since AoI combines the idea of service regularity with latency, we are also interested in the tail of information freshness. Figure 8-8 plots the performance of baseline WiFi systems and WiSwarm for the $95th$ percentile of AoI, i.e. the value of AoI which is only exceeded 5% of the time during an entire experiment. We observe very similar gains as mean AoI. For fixed rate, we observe an 18x reduction at $N = 8$ and 36x reduction at $N = 14$. For rate-optimized, we observe a 1.2x reduction at $N = 8$ and a $1.7x$ reduction for $N = 14$. Note that the tail AoI is important for our tracking application in addition to mean AoI, since a worse

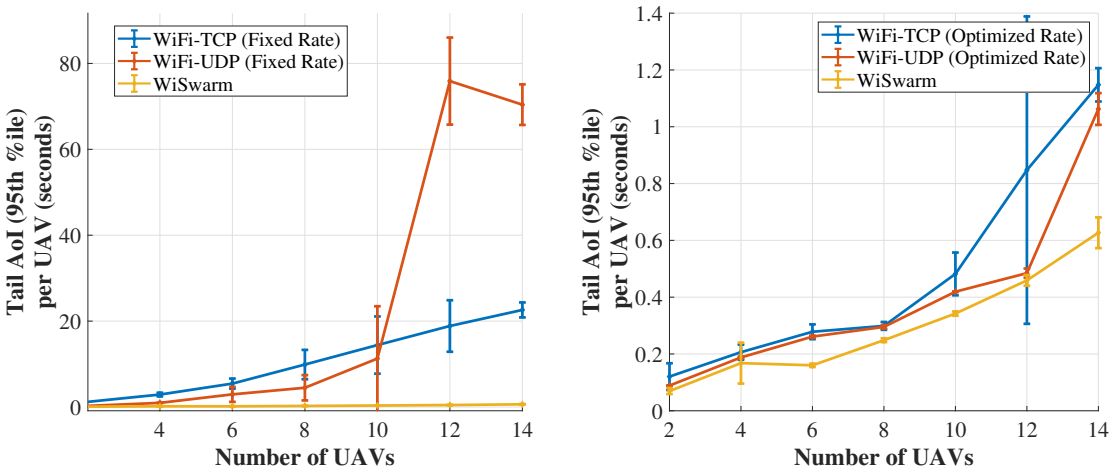
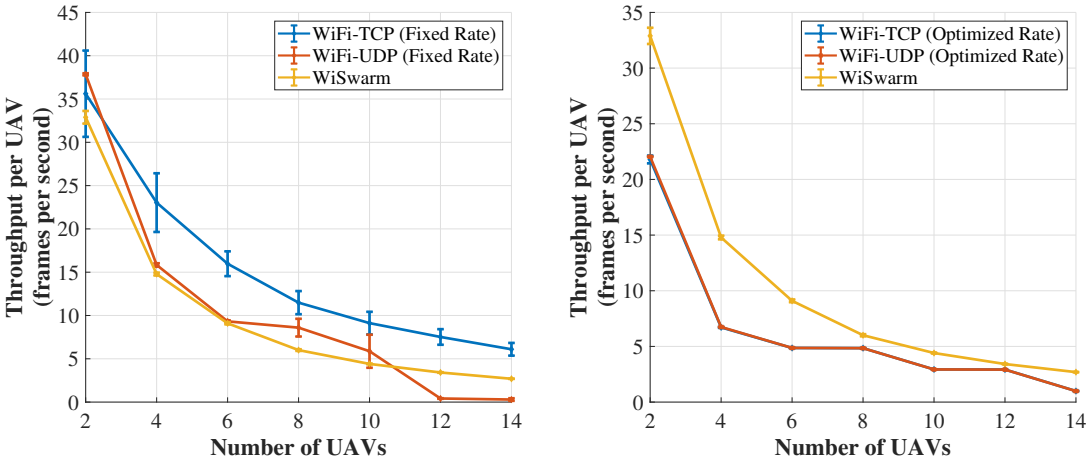


Figure 8-8: Tail (95^{th} percentile) AoI per UAV plotted for (a) fixed-rate (50 fps) and (b) optimized rate WiFi, as well as WiSwarm, as the number of UAVs increases.

tail suggests a higher probability of the car going out of the UAV's Field-of-View leading to lost tracking.

Throughput. Figure 8-9 plots the mean throughput per UAV for each of the considered systems as the number of UAVs increases. From Fig. 8-9(a), we observe that both fixed-rate WiFi-TCP and WiFi-UDP have higher per UAV throughput than WiSwarm. However, this doesn't help in getting better AoI (as we saw earlier) or tracking performance (as we will see later). **This supports the idea that high throughput alone is not sufficient and AoI is the right metric to optimize for in such real-time applications**. For the rate-optimized versions of WiFi, we see a performance improvement in mean throughput per-UAV since WiSwarm can avoid packet collisions and deliver higher rates than the distributed CSMA mechanism while also ensuring lower AoIs. For $N = 8$, WiSwarm achieves 1.2x higher throughput and for $N = 14$, it achieves 2.7x higher throughput.

Tracking Error. This is where we see how all the pieces of our system design come together to deliver better application performance. Figure 8-10 plots the mean tracking error (in pixels) per UAV as the number of UAVs increases,



(a) (b)

Figure 8-9: Mean Throughput per UAV plotted for (a) fixed-rate (50 fps) and (b) optimized rate WiFi, as well as WiSwarm, as the number of UAVs increases.

for WiSwarm and WiFi implementations. From Fig. 8-10(a), which shows the fixed-rate baselines, we observe that tracking performance improves by 12x for $N = 8$ and 4x for $N = 14$. From Fig. 8-10(b), with rate-optimized WiFi versions, we observe that tracking error is reduced by 2x at $N = 10$ and 4x at $N = 14$ with WiSwarm. We also note that the gap in performance between WiSwarm and the WiFi baselines *increases* with the system size. In other words, the performance of WiFi-TCP and WiFi-UDP degrades much more quickly with N leading to poor scalability.

8.3.B Flight Experiments

While the stationary experiments allowed us to test our system in great detail and provide extensive comparisons, they did not involve implementing the application on real UAVs tracking actual mobile targets in a dynamic environment. Our flight experiments address exactly this setting. Broadly, we will observe that the mobility of UAVs and higher degree of interference leads to worse wireless connectivity and, in turn, more congestion and packet collisions for WiFi. This allows

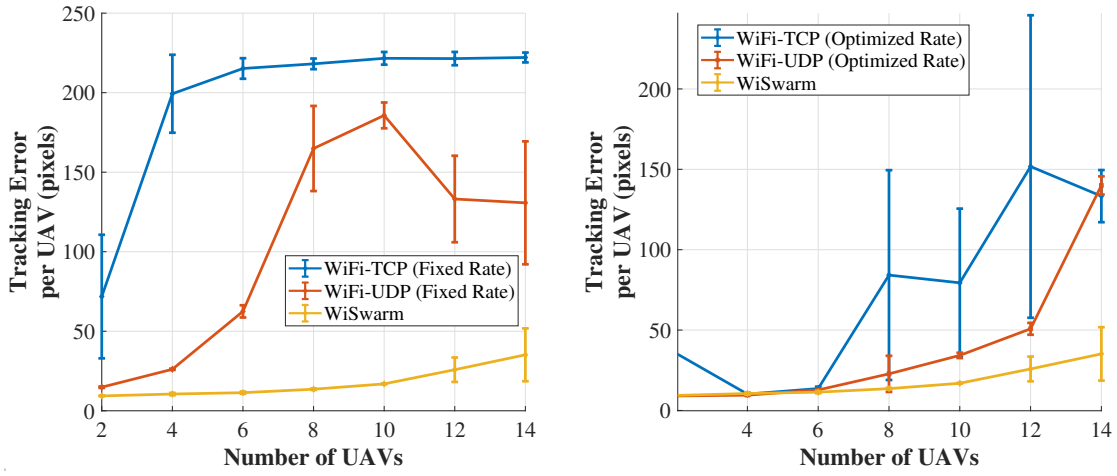


Figure 8-10: Mean Tracking Error per UAV plotted for (a) fixed-rate (50 fps) and (b) optimized rate WiFi, as well as WiSwarm, as the number of UAVs increases.

us to bring the robustness of WiSwarm into focus. We provide a video describing the setup and results from the flight experiments at [138].

Experimental Setup. In the flight tests, we replace the internal antenna of the RasPis with an external high-gain (8 dBi) antenna to improve range and reliability when the UAVs fly. We fly up to 5 UAVs at a time in our experiment space which is roughly 20 meters x 10 meters in size. The mobile objects are autonomous cars with RasPi 3Bs shown in Fig. 8-4(b). We program these cars to move in different polygonal trajectories over time and also stop occasionally at random for a few seconds. These trajectories are unknown to the UAVs and the Compute Node, and the job of the UAVs is to track the cars as closely as possible. Figure 8-1 depicts the setup for an experiment involving 5 UAVs tracking the corresponding cars.

We configure the Pi-Cameras at the UAVs to generate video frames at the maximum possible rate, which is 90 frames per second. For WiSwarm, we utilize this full rate, while for WiFi, we choose the optimized rate by using rate control. The video frames are 160x160 unencoded grayscale images in the yuv format (1 byte

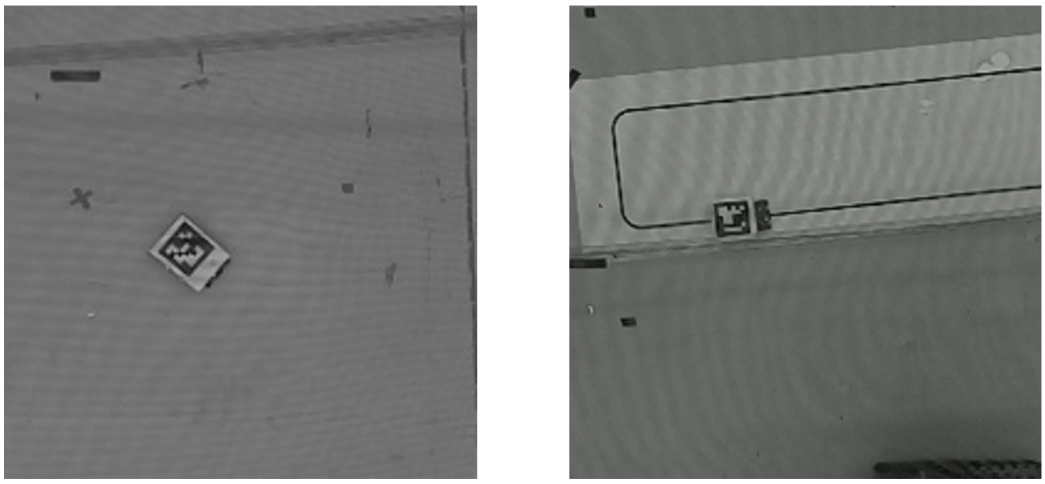


Figure 8-11: Two Examples of 160x160 grayscale video frames sent by the RasPis during flight experiments.

per pixel), with a total size of 25 kB per frame. Figure 8-11 shows two examples of frames sent to the Compute Node by RasPis from the flying UAVs during different experiments.

The sensing-UAVs implement a controller that requires knowledge of their own global position and orientation to be able to plan desired trajectories. A Motion Capture (MoCap) system provides this information to the UAVs (also via 2.4 GHz WiFi). These MoCap messages are sent to the UAVs in UDP messages at 30 messages/second and each message contains timestamp, position, and orientation of a single vehicle in 45 bytes. So the MoCap network usage is approximately 1.3 kB/s (or 11 kb/s) per UAV. Importantly, the MoCap system runs completely independently from the WiSwarm and WiFi systems and causes a low level of persistent interference in the channel. Thus, results from our flight experiments are a good measure of robustness of WiSwarm and WiFi to external interference.

Results. Figure 8-12 plots the coordinates of the sensing-UAVs and the target cars over time, for a two drone WiSwarm experiment, in both 2-D and 3-D. Similarly, Fig. 8-13 plots the coordinates of the sensing-UAVs and the target cars over time, for a two drone WiFi experiment. It is easy to see that WiSwarm allows for

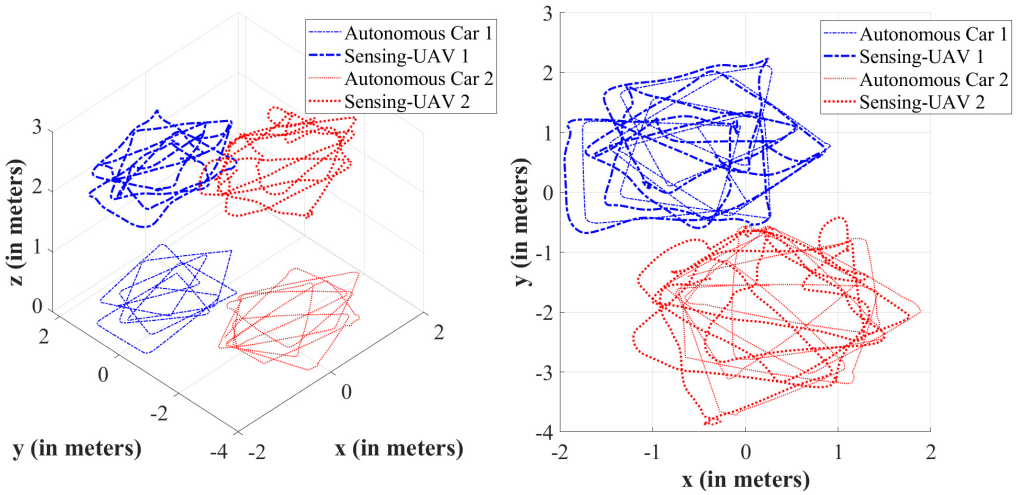


Figure 8-12: Coordinates of sensing-UAVs and target cars in 2-D and 3-D, for a two drone flight experiment running WiSwarm.

far better tracking than WiFi even for just two UAVs. This is further supported by the histograms of AoI and tracking error plotted in Fig. 8-14. The lower tracking error for WiSwarm is *due to the fact that it can achieve lower AoI*, and hence deliver fresher information.

We summarize the results of all of our flight experiments in Tables 8.1 and 8.2. We average over 4 minutes of flight data for each experiment. Our main observation is as follows: **while WiFi allows tracking for up to two UAVs at a time, WiSwarm can easily allow tracking for up to five UAVs at a time**. In fact, when there are more than two sources in the system, WiFi is unable to deliver more than a handful of packets and essentially no UAV control is possible. The main reason for this is the high level of packet collisions for WiFi. WiSwarm is relatively robust to the unreliable wireless channels, interference and mobility issues encountered in flight experiments, due to our scheduler design that avoids packet collisions and prioritizes AoI.

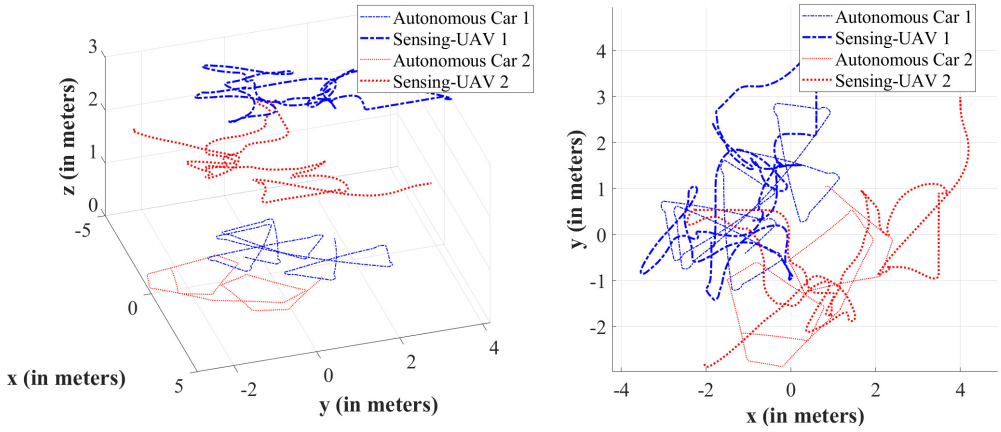


Figure 8-13: Coordinates of sensing-UAVs and target cars in 2-D and 3-D, for a two drone flight experiment running optimized WiFi-UDP.

Number of Drones	1	2	3	4	5
WiFi-UDP (Optimized)	0.43	1.85	-	-	-
WiSwarm	0.39	0.30	0.39	0.35	0.36

Table 8.1: Average tracking error per sensing-UAV (in meters).

Number of Drones	1	2	3	4	5
WiFi-UDP (Optimized)	0.10	0.19	-	-	-
WiSwarm	0.08	0.09	0.11	0.12	0.16

Table 8.2: Average AoI per sensing-UAV (in seconds).

8.4 Summary

In this chapter, we propose an AoI-based networking middleware that enables the customization of WiFi networks to the needs of time-sensitive applications that rely on multi-agent systems. By controlling the storage and flow of information in the underlying WiFi network, the middleware can prevent packet collisions and dynamically prioritize transmissions aiming to optimize information freshness. The middleware is implemented at the application layer, facilitating customization and integration to existing systems To demonstrate the benefits of our middleware, we implement a mobility tracking application using a swarm of

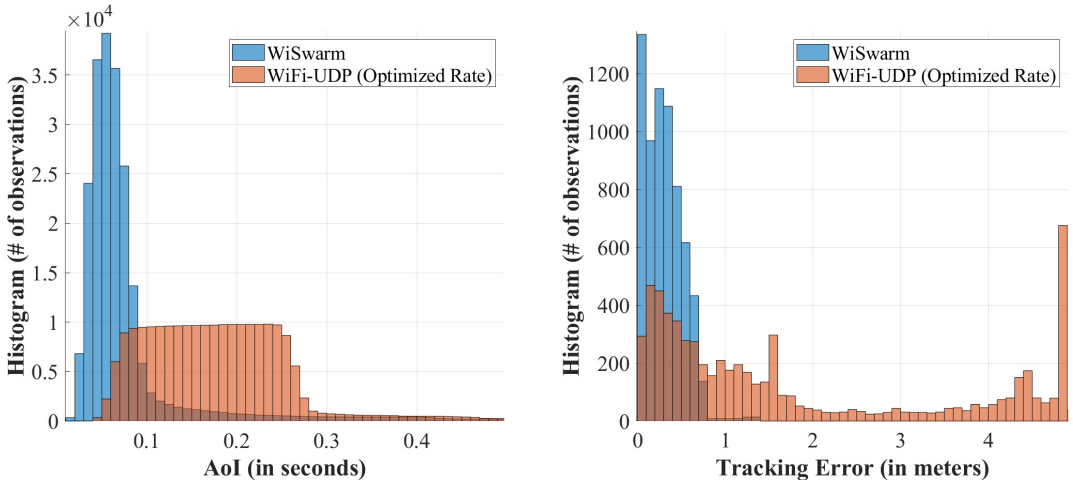


Figure 8-14: Histograms of (a) AoI and (b) tracking error for flight experiments with two UAVs, comparing WiSwarm with WiFi.

sensing-UAVs communicating with a central controller via WiFi. Our experimental results show that our middleware can improve information freshness and, as a result, tracking accuracy by *more than one order of magnitude* when compared to an equivalent system that uses plain WiFi. Our flight tests also show that the middleware improves scalability of the mobility tracking application. Interesting extensions of our work include consideration of a distributed middleware architecture. We provide the theoretical underpinnings of a distributed protocol for minimizing AoI in Chapter 6.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 9

Concluding Remarks

Our work in this thesis looked at the optimization of general Age of Information functions over single hop and multi-hop wireless networks. We further addressed open questions in AoI optimization literature on how to handle correlation among sources, and how to design distributed scheduling policies. Finally, we applied the insights gained from theoretical analysis to problems in robotics both via simulations of multi-agent mapping and implementation of a real multi-UAV wireless system (WiSwarm).

As we discussed in our literature review earlier, significant progress has been made in AoI optimization over the past decade, with a growing community of researchers around the world contributing to the topic. However, the task of applying theoretical insights from AoI literature to real-world problems and designing communication systems with AoI as a priority are still in their preliminary stages. Below, we list some open problems, challenges and future directions of work that might interest the reader working on these topics. This is by no means a complete list, and we invite suggestions on further topics of interest and collaboration.

9.1 Open Questions

9.1.A Complexity of Age Optimal Scheduling

The problem of minimizing weighted-sum AoI and functions of AoI in single-hop wireless networks has been well studied. We know of scheduling policies that provide constant factor performance guarantees for different kinds of settings. However, the only known solution providing exact optimality is dynamic programming. Unfortunately, dynamic programming suffers from the curse of dimensionality and is not a scalable method for networks with even a moderate number of sources.

The following question remains open: **Is there a computationally efficient class of scheduling policies that guarantees *exact optimality* for single-hop AoI optimization problems?** This question is closely related to the existence of delay optimal scheduling policies as well. Prior work on the complexity of delay optimal scheduling already exists [142]. However, this only looks at order optimality of delay rather than exact optimality.

If the answer to the previous question is in the negative, a natural question that is raised is the following - **What is the best approximation factor achievable for a computationally efficient scheduling policy?** We know of approximation factors of 2 [15], 4 [17], and more recently 1.62 [143]. From simulations, we know that max-weight and Whittle style policies are *almost* optimal. What is the closest that such methods can go to optimality?

A third and related question (which was motivated by discussion with Prof. Yury Polyanskiy) is the following - **What kinds of performance can neural network based approaches achieve for AoI optimization and can they lead to more efficient/scalable design?** Traditionally, deep learning and reinforcement learning based approaches have not worked as well as classical methods to optimize performance in queuing networks. They require a lot of data and don't general-

ize well to different kinds of network conditions, cost functions, topologies and channel settings. However, there is a possibility that these methods might become more efficient at solving network optimization problems in the near future.

9.1.B Multi-Hop AoI Optimization

All of the questions in the section above were related to AoI optimization in single-hop wireless networks. However, **the problem of making scheduling and routing decisions to optimize AoI for unicast, multicast and broadcast flows in general multi-hop wireless networks remains unsolved.** Our Age Debt approach provides one possible answer to this problem in Chapter 5. However, our approach is heuristic and does not provide performance guarantees. Solving the multi-hop setting in a computationally efficient manner, while also providing performance guarantees, is *one of the main open questions in AoI literature at present.*

9.1.C Correlated or Coupled Sources

In Chapter 7, we describe a simple way to model correlated sources and analyze how AoI optimization changes in the presence of such correlation. In recent ongoing work, we have realized that the way in which correlation influences monitoring and AoI is crucially controlled by how it is modeled. Changing the correlation model can lead to very different conclusions regarding how to do scheduling design and the benefits of correlation. A natural question to ask is - **What's the best way to model correlation or coupling between sources in a remote monitoring or networked control setting? How does this influence AoI optimization?**

9.1.D From Freshness to Semantics

AoI is one particular way to measure the quality of information at a destination compared to the quality at the source. More recent works have started looking at newer metrics such as Age of Incorrect Information [128, 129, 130, 131] and version Age of Information [144, 145]. There has been a growing push to move beyond AoI to metrics that consider the semantics of information being transmitted, as well as to consider contextual aspects. For example, in the mobility tracking application, if we knew that the object to be tracked moves very slowly in certain areas of the environment, then we would reduce its weight when it enters those areas. *Incorporating these kinds of notions in information freshness optimization is an interesting area of future research.*

This section was motivated by discussions with Prof. John Tsitsiklis, Prof. Nikolaos Pappas and Prof. Roy Yates.

9.1.E Applications

The promise of AoI in the networking community has always been to impact real-time applications, especially monitoring and control. Our work realizes some of this promise for robotics applications. However, plenty of work remains. We identify three specific areas where information freshness optimization might lead to better systems design in practice. Again, this is not a complete list, but the starting point for a reader looking for applications.

1) **Multi Agent Robotics** - This application domain has been the primary focus of our applied work. We believe that AoI-based methods can lead to significantly better ways to co-ordinate teams of robots. Offloading computing to the edge, and using wireless networks that utilize AoI, can lead to more scalable and efficient autonomous robots.

2) **Federated Learning** - Using the freshness of gradients to be collected might

improve the speed of convergence or the quality of the aggregated model in federated learning contexts. It might also provide a way to figure out which subset of users to poll at any given time for gradient information to ensure fairness, address resource availability and time-varying datasets.

3) **Software Defined Networking (SDN)** - The typical way in which SDN controllers work is that they require *timely* information about the entire network state to be able to make network control decisions. A monitoring strategy that utilizes information freshness can lead to more efficient and scalable monitoring for very large networks.

9.1.F Hardware Implementations

Apart from different applications, an important direction for future research is **implementing AoI-based scheduling policies at the MAC layer** in networking hardware. Our WiSwarm implementation in Chapter 8 was at the application layer, making it inefficient. Implementing protocols such as Fresh-CSMA at the MAC layer and then comparing them to standard networking solutions would further demonstrate the utility of AoI-based design.

THIS PAGE INTENTIONALLY LEFT BLANK

Bibliography

[1] Vishrant Tripathi and Eytan Modiano. A whittle index approach to minimizing functions of age of information. In *Proc. 57th Allerton Conf. Commun. Control Comput.*, pages 1160–1167. IEEE, 2019.

[2] Vishrant Tripathi and Eytan Modiano. An online learning approach to optimizing time-varying costs of aoi. In *Proceedings of the Twenty-second International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, pages 241–250, 2021.

[3] Vishrant Tripathi, Luca Ballotta, Luca Carlone, and Eytan Modiano. Computation and communication co-design for real-time monitoring and control in multi-agent systems. In *2021 19th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*, pages 1–8. IEEE, 2021.

[4] Vishrant Tripathi and Eytan Modiano. Age debt: A general framework for minimizing age of information. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE, 2021.

[5] Vishrant Tripathi, Rajat Talak, and Eytan Modiano. Information freshness in multihop wireless networks. *IEEE/ACM Transactions on Networking*, 2022.

[6] Vishrant Tripathi, Nicholas Jones, and Eytan Modiano. Fresh-csma: A distributed protocol for minimizing age of information. *arXiv preprint arXiv:2212.03087*, 2022.

[7] Vishrant Tripathi and Eytan Modiano. Optimizing age of information with correlated sources. In *Proceedings of the Twenty-Third International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, pages 41–50, 2022.

[8] Vishrant Tripathi, Igor Kadota, Ezra Tal, Muhammad Shahir Rahman, Alexander Warren, Sertac Karaman, and Eytan Modiano. Wiswarm: Age-

of-information-based wireless networking for collaborative teams of uavs. *arXiv preprint arXiv:2212.03298*, 2022.

[9] Sanjit Kaul, Roy Yates, and Marco Gruteser. Real-time status: How often should one update? In *Proc. IEEE INFOCOM*, pages 2731–2735, 2012.

[10] Clement Kam, Sastry Kompella, and Anthony Ephremides. Age of information under random updates. In *Proc. IEEE Int. Symp. Information Theory (ISIT)*, pages 66–70, 2013.

[11] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff. Update or wait: How to keep your data fresh. *IEEE Trans. Information Theory*, 63(11):7492–7508, Nov. 2017.

[12] Longbo Huang and Eytan Modiano. Optimizing age-of-information in a multi-class queueing system. In *Proc. IEEE Int. Symp. Information Theory (ISIT)*, pages 1681–1685, 2015.

[13] Yoshiaki Inoue, Hiroyuki Masuyama, Tetsuya Takine, and Toshiyuki Tanaka. A general formula for the stationary distribution of the age of information and its application to single-server queues. *arXiv preprint arXiv:1804.06139*, 2018.

[14] Ahmed M Bedewy, Yin Sun, and Ness B Shroff. Minimizing the age of information through queues. *IEEE Trans. Information Theory*, 65(8):5215–5232, 2019.

[15] Igor Kadota, Abhishek Sinha, Elif Uysal-Biyikoglu, Rahul Singh, and Eytan Modiano. Scheduling policies for minimizing age of information in broadcast wireless networks. *IEEE/ACM Trans. Netw.*, 26(6):2637–2650, 2018.

[16] Igor Kadota, Abhishek Sinha, and Eytan Modiano. Scheduling algorithms for optimizing age of information in wireless networks with throughput constraints. *IEEE/ACM Trans. Netw.*, 27(4):1359–1372, 2019.

[17] Rajat Talak, Sertac Karaman, and Eytan Modiano. Optimizing information freshness in wireless networks under general interference constraints. In *Proc. ACM Int. Symp. Mobile Ad Hoc Netw. Comput. (MobiHoc)*, pages 61–70, 2018.

[18] Vishrant Tripathi and Sharayu Moharir. Age of information in multi-source systems. In *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, pages 1–6, 2017.

[19] Shahab Farazi, Andrew G Klein, John A McNeill, and D Richard Brown. On the age of information in multi-source multi-hop wireless status update networks. In *Proc. IEEE Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, pages 1–5, 2018.

[20] Yu-Pin Hsu, Eytan Modiano, and Lingjie Duan. Age of information: Design and analysis of optimal scheduling algorithms. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 561–565. IEEE, 2017.

[21] Chengzhang Li, Shaoran Li, and Y Thomas Hou. A general model for minimizing age of information at network edge. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 118–126. IEEE, 2019.

[22] Yin Sun, Yury Polyanskiy, and Elif Uysal-Biyikoglu. Remote estimation of the wiener process over a channel with random delay. In *Proc. IEEE Int. Symp. Information Theory (ISIT)*, pages 321–325, 2017.

[23] Yin Sun and Benjamin Cyr. Sampling for data freshness optimization: Nonlinear age functions. *IEEE Journal Commun. Netw.*, 21(3):204–219, 2019.

[24] Tasmeen Zaman Ornee and Yin Sun. Sampling for remote estimation through queues: Age of information and beyond. *IEEE Int. Symp. Model. Optim. Mobile, Ad Hoc Wireless Netw. (WiOpt)*, 2019.

[25] Jaya Prakash Champati, Mohammad H Mamduhi, Karl H Johansson, and James Gross. Performance characterization using aoi in a single-loop networked control system. In *Proc. IEEE INFOCOM AoI Workshop*, pages 197–203, 2019.

[26] Markus Klügel, Mohammad H Mamduhi, Sandra Hirche, and Wolfgang Kellerer. Aoi-penalty minimization for networked control systems with packet loss. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 189–196. IEEE, 2019.

[27] Antzela Kosta, Nikolaos Pappas, Vangelis Angelakis, et al. Age of information: A new concept, metric, and tool. *Foundations and Trends in Networking*, 12(3):162–259, 2017.

[28] Yin Sun, Igor Kadota, Rajat Talak, and Eytan Modiano. Age of information: A new metric for information freshness. *Synthesis Lectures on Communication Networks*, 12(2):1–224, 2019.

[29] Antzela Kosta, Nikolaos Pappas, Anthony Ephremides, and Vangelis Angelakis. Age and value of information: Non-linear age case. In *Proc. IEEE Int. Symp. Information Theory (ISIT)*, pages 326–330, 2017.

[30] Prakirt Raj Jhunjhunwala and Sharayu Moharir. Age-of-information aware scheduling. In *Proc. IEEE SPCOM*, 2018.

[31] Igor Kadota, Elif Uysal-Biyikoglu, Rahul Singh, and Eytan Modiano. Minimizing the age of information in broadcast wireless networks. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 844–851. IEEE, 2016.

[32] Antzela Kosta, Nikolaos Pappas, Anthony Ephremides, and Vangelis Angelakis. The cost of delay in status updates and their value: Non-linear ageing. *IEEE Transactions on Communications*, 68(8):4905–4918, 2020.

[33] Onur Ayan, Mikhail Vilgelm, Markus Klügel, Sandra Hirche, and Wolfgang Kellerer. Age-of-information vs. value-of-information scheduling for cellular networked control systems. In *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, pages 109–117, 2019.

[34] Ali Maatouk, Saad Kriouile, Mohamad Assaad, and Anthony Ephremides. On the optimality of the whittle's index policy for minimizing the age of information. *IEEE Trans. Wireless Commun.*, 2020.

[35] Xi Zheng, Sheng Zhou, and Zhisheng Niu. Context-aware information lapse for timely status updates in remote control systems. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2019.

[36] Clement Kam, Sastry Kompella, and Anthony Ephremides. Learning to sample a signal through an unknown system for minimum aoi. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 177–182. IEEE, 2019.

[37] Kavya Bhandari, Santosh Fatale, Urvidh Narula, Sharayu Moharir, and Manjesh Kumar Hanawal. Age-of-information bandits. *arXiv preprint arXiv:2001.09317*, 2020.

[38] Subhankar Banerjee, Rajarshi Bhattacharjee, and Abhishek Sinha. Fundamental limits of age-of-information in stationary and non-stationary environments. *arXiv preprint arXiv:2001.05471*, 2020.

[39] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *IPSN 2005. Fourth Int. Symp. Inf. Proc. Sensor Netw., 2005*, pages 63–70, 2005.

[40] Ruggero Carli, Alessandro Chiuso, Luca Schenato, and Sandro Zampieri. Distributed kalman filtering based on consensus strategies. *IEEE Journal on Selected Areas in communications*, 26(4):622–633, 2008.

[41] Reza Olfati-Saber and Jeff S Shamma. Consensus filters for sensor networks and distributed sensor fusion. In *Proc. 44th IEEE CDC*, pages 6698–6703, 2005.

[42] M. Amir and T. Givargis. Priority neuron: A resource-aware neural network for cyber-physical systems. *IEEE Trans. Comp.-Aided Design of Integrated Circ. and Sys.*, 37(11):2732–2742, Nov 2018.

[43] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proc. IEEE CVPR*, pages 4510–4520, 2018.

[44] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. *arXiv e-prints*, page arXiv:1804.02767, Apr 2018.

[45] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proc. IEEE/CVF Int. Conf. Comp. Vision*, pages 1314–1324, 2019.

[46] Daniel Crankshaw, Xin Wang, Guilio Zhou, Michael J Franklin, Joseph E Gonzalez, and Ion Stoica. Clipper: A low-latency online prediction serving system. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 613–627, 2017.

[47] Sandeep Chinchali, Apoorva Sharma, James Harrison, Amine Elhafsi, Daniel Kang, Evgenya Pergament, Eyal Cidon, Sachin Katti, and Marco Pavone. Network offloading policies for cloud robotics: A learning-based approach. In *Proceedings of Robotics: Science and Systems*, Freiburgim-Breisgau, Germany, June 2019.

[48] Q. He, D. Yuan, and A. Ephremides. Optimizing freshness of information: On minimum age link scheduling in wireless systems. In *Proc. WiOpt*, pages 1–8, May 2016.

[49] A. M. Bedewy, Y. Sun, and N. B. Shroff. Age-optimal information updates in multihop networks. In *Proc. ISIT*, pages 576–580, Jun. 2017.

[50] Shahab Farazi, Andrew G Klein, John A McNeill, and D Richard Brown. On the age of information in multi-source multi-hop wireless status update networks. In *Proc. SPAWC*, pages 1–5. IEEE, 2018.

[51] Shahab Farazi, Andrew G Klein, and D Richard Brown. Fundamental bounds on the age of information in general multi-hop interference networks. In *Proc. INFOCOM Wkshps.*, pages 96–101, 2019.

[52] Baturalp Buyukates, Alkan Soysal, and Sennur Ulukus. Age of information in multihop multicast networks. *Journal of Communications and Networks*, 21(3):256–267, 2019.

[53] Rajat Talak, Sertac Karaman, and Eytan Modiano. Distributed scheduling algorithms for optimizing information freshness in wireless networks. In *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5. IEEE, 2018.

[54] Zhiyuan Jiang, Bhaskar Krishnamachari, Sheng Zhou, and Zhisheng Niu. Can decentralized status update achieve universally near-optimal age-of-information in wireless multiaccess channels? In *2018 30th International Teletraffic Congress (ITC 30)*, volume 1, pages 144–152. IEEE, 2018.

[55] Xingran Chen, Konstantinos Gatsis, Hamed Hassani, and Shirin Saeedi Bidokhti. Age of information in random access channels. *IEEE Trans. Inf. Theory*, 2022.

[56] Xingran Chen, Xinyu Liao, and Shirin Saeedi Bidokhti. Real-time sampling and estimation on random access channels: Age of information and beyond. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2021.

[57] Orhan Tahir Yavascan and Elif Uysal. Analysis of age-aware slotted aloha. In *2020 IEEE Globecom Workshops (GC Wkshps*, pages 1–6. IEEE, 2020.

[58] Orhan Tahir Yavascan and Elif Uysal. Analysis of slotted aloha with an age threshold. *IEEE Journal on Selected Areas in Communications*, 39(5):1456–1470, 2021.

[59] Ali Maatouk, Mohamad Assaad, and Anthony Ephremides. On the age of information in a csma environment. *IEEE/ACM Trans. Netw.*, 28(2):818–831, 2020.

[60] Andrea Baiocchi and Ion Turcanu. Age of information of one-hop broadcast communications in a csma network. *IEEE Commun. Lett.*, 25(1):294–298, 2020.

[61] Mei Wang and Yunquan Dong. Broadcast age of information in csma/ca based wireless networks. In *Proc. IEEE Int. Wireless Commun. Mob. Comp. Conf. (IWCMC)*, pages 1102–1107, 2019.

[62] Ahmed M Bedewy, Yin Sun, Rahul Singh, and Ness B Shroff. Optimizing information freshness using low-power status updates via sleep-wake scheduling. In *Proc. ACM Int. Symp. Mobile Ad Hoc Netw. Comput. (MobiHoc)*, pages 51–60, 2020.

[63] Ahmed M Bedewy, Yin Sun, Rahul Singh, and Ness B Shroff. Low-power status updates via sleep-wake scheduling. *IEEE/ACM Trans. Netw.*, 29(5):2129–2141, 2021.

[64] Leonard Kleinrock and Fouad Tobagi. Packet switching in radio channels: Part i-carrier sense multiple-access modes and their throughput-delay characteristics. *IEEE Trans. Commun.*, 23(12):1400–1416, 1975.

[65] Giuseppe Bianchi. Performance analysis of the ieee 802.11 distributed coordination function. *IEEE J. Sel. Areas Commun.*, 18(3):535–547, 2000.

[66] Libin Jiang and Jean Walrand. A distributed csma algorithm for throughput and utility maximization in wireless networks. *IEEE/ACM Transactions on Networking*, 18(3):960–972, 2009.

[67] Libin Jiang, Mathieu Leconte, Jian Ni, R Srikant, and Jean Walrand. Fast mixing of parallel glauber dynamics and low-delay csma scheduling. *IEEE Trans. Inf. Theory*, 58(10):6541–6555, 2012.

[68] Jian Ni, Bo Tan, and Rayadurgam Srikant. Q-csma: Queue-length-based csma/ca algorithms for achieving maximum throughput and low delay in wireless networks. *IEEE/ACM Transactions on Networking*, 20(3):825–836, 2011.

[69] Bin Li and Atilla Eryilmaz. Optimal distributed scheduling under time-varying conditions: A fast-csma algorithm with applications. *IEEE Trans. Wireless Commun.*, 12(7):3278–3288, 2013.

[70] Sudheer Poojary, Sanidhay Bhambay, and Parimal Parag. Real-time status updates for correlated source. In *2017 IEEE Information Theory Workshop (ITW)*, pages 274–278. IEEE, 2017.

[71] Qing He, Gyorgy Dan, and Viktoria Fodor. Minimizing age of correlated information for wireless camera networks. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 547–552. IEEE, 2018.

[72] Qing He, György Dán, and Viktoria Fodor. Joint assignment and scheduling for minimizing age of correlated information. *IEEE/ACM Transactions on Networking*, 27(5):1887–1900, 2019.

[73] Bo Zhou and Walid Saad. On the age of information in internet of things systems with correlated devices. In *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pages 1–6. IEEE, 2020.

[74] Zhiyuan Jiang and Sheng Zhou. Status from a random field: How densely should one update? In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 1037–1041. IEEE, 2019.

[75] Anders E Kalør and Petar Popovski. Minimizing the age of information from sensors with common observations. *IEEE Wireless Communications Letters*, 8(5):1390–1393, 2019.

[76] Pablo Valerio. Amazon robotics: IoT in the warehouse. online: https://www.informationweek.com/strategic-cio/amazon-robotics-iot-in-the-warehouse/d/d-id/1322366, 2015.

[77] Javier Gozalvez, Miguel Sepulcre, and Ramon Bauza. IEEE 802.11p vehicle to infrastructure communications in urban environments. *IEEE Communications Magazine*, 50(5):176–183, 2012.

[78] Martin Klapez, Carlo Augusto Grazia, and Maurizio Casoni. Application-level performance of IEEE 802.11p in safety-related V2X field trials. *IEEE Internet of Things Journal*, 7(5):3850–3860, 2020.

[79] New York City Department of Transportation. Nyc connected vehicle project: For safer transportation. online: https://cvp.nyc/, 2022.

[80] Marco Tranzatto, Frank Mascarich, Lukas Bernreiter, Carolina Godinho, Marco Camurri, Shehryar Khattak, Tung Dang, Victor Reijgwart, Johannes Loje, David Wisth, Samuel Zimmermann, Huan Nguyen, Marius Fehr, Lukas Solanka, Russell Buchanan, Marko Bjelonic, Nikhil Khedekar, Mathieu Valceschini, Fabian Jenelten, Mihir Dharmadhikari, Timon Homberger, Paolo De Petris, Lorenz Wellhausen, Mihir Kulkarni, Takahiro Miki, Satchel Hirsch, Markus Montenegro, Christos Papachristos, Fabian Tresoldi, Jan Carius, Giorgio Valsecchi, Joonho Lee, Konrad Meyer, Xiangyu Wu, Juan Nieto, Andy Smith, Marco Hutter, Roland Siegwart, Mark Mueller, Maurice Fallon, and Kostas Alexis. CERBERUS: Autonomous legged and aerial robotic exploration in the tunnel and urban circuits of the DARPA subterranean challenge. *Field Robotics*, 2021. [accepted for publication].

[81] Jonathan Cacace, Alberto Finzi, Vincenzo Lippiello, Michele Furci, Nicola Mimmo, and Lorenzo Marconi. A control architecture for multiple drones operated via multimodal interaction in search amp; rescue mission. In *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 233–239, 2016.

[82] Javier Alonso-Mora, Eduardo Montijano, Tobias Nägeli, Otmar Hilliges, Mac Schwager, and Daniela Rus. Distributed multi-robot formation control in dynamic environments. *Autonomous Robots*, 43:1079–1100, 2019.

[83] Nesrine Mahdoui, Vincent Frémont, and Enrico Natalizio. Communicating multi-UAV system for cooperative SLAM-based exploration. *Journal of Intelligent & Robotic Systems*, 98:325–343, 2020.

[84] Riccardo Petrolo, Yingyan Lin, and Edward Knightly. ASTRO: Autonomous, sensing, and tetherless networked drones. In *Proc. of ACM MobiSys – DroNet workshop*, 2018.

[85] Tobias Nägeli, Lukas Meier, Alexander Domahidi, Javier Alonso-Mora, and Otmar Hilliges. Real-time planning for automated multi-view drone cinematography. *ACM Transactions on Graphics*, 36(4), 2017.

[86] Pierre-Yves Lajoie, Benjamin Ramtoula, Yun Chang, Luca Carlone, and Giovanni Beltrame. DOOR-SLAM: distributed, online, and outlier resilient slam for robotic teams. *IEEE Robotics and Automation Letters*, 5(2):1656–1663, 2020.

[87] Songtao He, Favyen Bastani, Arjun Balasingam, Karthik Gopalakrishna, Ziwen Jiang, Mohammad Alizadeh, Hari Balakrishnan, Michael Cafarella, Tim Kraska, and Sam Madden. Beecluster: Drone orchestration via predictive optimization. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, page 299–311, 2020.

[88] Pablo Urcola, María T. Lázaro, José A. Castellanos, and Luis Montano. Cooperative minimum expected length planning for robot formations in stochastic maps. *Robotics and Autonomous Systems*, 87:38–50, 2017.

[89] María T. Lázaro, Lina M. Paz, Pedro Pinies, José A. Castellanos, and Giorgio Grisetti. Multi-robot SLAM using condensed measurements. In *Proc. of IEEE/RSJ IROS*, pages 1069–1076, 2013.

[90] Laetitia Matignon and Olivier Simonin. Multi-robot simultaneous coverage and mapping of complex scene - comparison of different strategies. In *Proc. of ACM AAMAS*, page 559–567, 2018.

[91] 3GPP. Radio access network; e-utra and e-utran overall description. Tech. spec. 3GPP TS 36.300, 2012.

[92] O-RAN ALLIANCE. O-ran: Towards an open and smart ran. White paper, 2018.

[93] O-RAN ALLIANCE. O-ran use cases and deployment scenarios. White paper, 2020.

[94] VMware. O-ran: Defining the path for innovating the ran. White paper, 2021.

[95] John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 1989.

[96] Jihyeon Yun, Changhee Joo, and Atilla Eryilmaz. Optimal real-time monitoring of an information source under communication costs. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 4767–4772. IEEE, 2018.

[97] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

[98] Elad Hazan. Introduction to online convex optimization. *arXiv preprint arXiv:1909.05207*, 2019.

[99] Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

[100] James Hannan. Approximation to bayes risk in repeated play. *Contributions to the Theory of Games*, 3:97–139, 1957.

[101] Alon Cohen and Tamir Hazan. Following the perturbed leader for online structured learning. In *International Conference on Machine Learning*, pages 1034–1042. PMLR, 2015.

[102] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

[103] Omar Besbes, Yonatan Gur, and Assaf Zeevi. Non-stationary stochastic optimization. *Operations research*, 63(5):1227–1244, 2015.

[104] Ali Jadbabaie, Alexander Rakhlin, Shahin Shahrampour, and Karthik Sridharan. Online optimization: Competing with dynamic comparators. In *Proc. Int. Conf. Artificial Intell. Stats. (AISTATS)*, pages 398–406, 2015.

[105] Omar Besbes, Yonatan Gur, and Assaf Zeevi. Optimal exploration–exploitation in a multi-armed bandit problem with non-stationary rewards. *Stochastic Systems*, 9(4):319–337, 2019.

[106] Wang Chi Cheung, David Simchi-Levi, and Ruihao Zhu. Learning to optimize under non-stationarity. In *Proc. Int. Conf. Artificial Intell. Stats. (AISTATS)*, pages 1079–1087, 2019.

[107] Injong Rhee, Minsu Shin, Seongik Hong, Kyunghan Lee, Seong Joon Kim, and Song Chong. On the levy-walk nature of human mobility. *IEEE/ACM Trans. Netw.*, 19(3):630–643, 2011.

[108] L. Ballotta, L. Schenato, and L. Carlone. Computation-communication trade-offs and sensor selection in real-time estimation for processing networks. *IEEE Trans. Net. Sci. Eng.*, 7(4), 2020.

[109] Peter Whittle. Restless bandits: Activity allocation in a changing world. *Journal of applied probability*, 25(A):287–298, 1988.

[110] Richard R Weber and Gideon Weiss. On an index policy for restless bandits. *Journal of applied probability*, pages 637–648, 1990.

[111] Vishrant Tripathi, Luca Ballotta, Luca Carlone, and Eytan Modiano. Computation and communication co-design for real-time monitoring and control in multi-agent systems. https://www.youtube.com/watch?v=gs5lb3YE-ec, 2021.

[112] Daniel Meyer-Delius, Maximilian Beinhofer, and Wolfram Burgard. Occupancy grid models for robot mapping in changing environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, 2012.

[113] Jari Saarinen, Henrik Andreasson, and Achim J Lilienthal. Independent markov chain occupancy grid maps for representation of dynamic environment. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pages 3489–3495, 2012.

[114] Frederic Bourgault, Alexei A Makarenko, Stefan B Williams, Ben Grocholsky, and Hugh F Durrant-Whyte. Information based adaptive robotic exploration. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, volume 1, pages 540–545, 2002.

[115] Henry Carrillo, Philip Dames, Vijay Kumar, and José A Castellanos. Autonomous robotic exploration using occupancy grid maps and graph slam based on shannon and rényi entropy. In *IEEE ICRA*, pages 487–494, 2015.

[116] A. Y. S. Lam, Y. Leung, and X. Chu. Autonomous-vehicle public transportation system: Scheduling and admission control. *IEEE Trans. Intell. Transp. Syst.*, 17(5):1210–1226, 2016.

[117] A. O. Al-Abbasi, A. Ghosh, and V. Aggarwal. Deeppool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning. *IEEE Trans. Intell. Transp. Syst.*, 20(12):4714–4727, 2019.

[118] Santiago Muelas, Antonio LaTorre, and Jose-Maria Pena. A distributed vns algorithm for optimizing dial-a-ride problems in large-scale scenarios. *Transportation Research Part C: Emerging Technologies*, 54:110–130, 2015.

[119] Uber. How does uber pool expand access?, 2018.

[120] Ahmed M. Bedewy, Yin Sun, and Ness B. Shroff. Minimizing the age of the information through queues. *IEEE Trans. Inf. Theory*, Aug. 2019.

[121] IH Hou, V Borkar, and PR Kumar. A theory of qos for wireless. In *Proc. INFOCOM*, pages 486–494, 2009.

[122] Michael J Neely. Stability and capacity regions or discrete time queueing networks. *arXiv preprint arXiv:1003.3396*, 2010.

[123] Leonidas Georgiadis, Michael J Neely, and Leandros Tassiulas. *Resource allocation and cross-layer control in wireless networks*. Now Publishers Inc, 2006.

[124] I. Kadota, A. Sinha, E. Uysal-Biyikoglu, R. Singh, and E. Modiano. Scheduling policies for minimizing age of information in broadcast wireless networks. *IEEE/ACM Trans. Netw.*, 26(6):2637–2650, Dec. 2018.

[125] Vishrant Tripathi and Eytan Modiano. A whittle index approach to minimizing functions of age of information. In *Proc. Allerton*, pages 1160–1167, Oct. 2019.

[126] Yin Sun, Igor Kadota, Rajat Talak, and Eytan Modiano. Age of information: A new metric for information freshness. *Synthesis Lectures on Communication Networks*, 12(2):1–224, 2019.

[127] IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pages 1–4379, 2020.

[128] Ali Maatouk, Saad Kriouile, Mohamad Assaad, and Anthony Ephremides. The age of incorrect information: A new performance metric for status updates. *IEEE/ACM Transactions on Networking*, 28(5):2215–2228, 2020.

[129] Clement Kam, Sastry Kompella, and Anthony Ephremides. Age of incorrect information for remote estimation of a binary markov source. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE, 2020.

[130] Saad Kriouile and Mohamad Assaad. Minimizing the age of incorrect information for real-time tracking of markov remote sources. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 2978–2983. IEEE, 2021.

[131] Ali Maatouk, Mohamad Assaad, and Anthony Ephremides. The age of incorrect information: An enabler of semantics-empowered communication. *IEEE Transactions on Wireless Communications*, 2022.

[132] Igor Kadota and Eytan Modiano. Minimizing the age of information in wireless networks with stochastic arrivals. *IEEE Transactions on Mobile Computing*, 20(3):1173–1185, 2019.

[133] Michael Grant and Stephen Boyd. Cvx: Matlab software for disciplined convex programming, version 2.1, 2014.

[134] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010.

[135] Fragkiskos Papadopoulos, Dmitri Krioukov, Marián Boguná, and Amin Vahdat. Greedy forwarding in dynamic scale-free networks embedded in hyperbolic metric spaces. In *2010 Proceedings IEEE Infocom*, pages 1–9. IEEE, 2010.

[136] Justin Hu, Ariana Bruno, Brian Ritchken, Brendon Jackson, Mateo Espinosa, Aditya Shah, and Christina Delimitrou. Hivemind: A scalable and serverless coordination control platform for uav swarms. *arXiv preprint arXiv:2002.01419*, 2020.

[137] Sandeep Chinchali, Apoorva Sharma, James Harrison, Amine Elhafsi, Daniel Kang, Evgenya Pergament, Eyal Cidon, Sachin Katti, and Marco Pavone. Network offloading policies for cloud robotics: a learning-based approach. *Autonomous Robots*, 45(7):997–1012, 2021.

[138] Wiswarm: Time-sensitive wireless networking for a collaborative team of uavs. https://www.youtube.com/watch?v=zGjGxK5AXFk, 2022. Video Attachment.

[139] Maice Costa, Marian Codreanu, and Anthony Ephremides. On the age of information in status update systems with packet management. *IEEE Transactions on Information Theory*, 62(4):1897–1910, 2016.

[140] David Mills, Jim Martin, Jack Burbank, and William Kasch. Network time protocol version 4: Protocol and algorithms specification. RFC 5905, 2010.

[141] Ezra Tal and Sertac Karaman. Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness. *IEEE Transactions on Control Systems Technology*, 29(3):1203–1218, 2021.

[142] Devavrat Shah, NC David, and John N Tsitsiklis. Hardness of low delay network scheduling. *IEEE Transactions on Information Theory*, 57(12):7810–7817, 2011.

[143] Chengzhang Li, Shaoran Li, Qingyu Liu, Y Thomas Hou, Wenjing Lou, Sastry Kompella, and Virginia Tech. Eywa: A general approach for scheduler design in aoi optimization.

[144] Jing Zhong, Roy D Yates, and Emina Soljanin. Two freshness metrics for local cache refresh. In *Proc. IEEE Int. Symp. Information Theory (ISIT)*, pages 1924–1928, 2018.

[145] Baturalp Buyukates, Melih Bastopcu, and Sennur Ulukus. Version age of information in clustered gossip networks. *IEEE Journal on Selected Areas in Information Theory*, 3(1):85–97, 2022.