Analysis and Optimization of Networks in Overload

by

Xinyu Wu

B.E., Shanghai Jiao Tong University (2018)

M.S., Massachusetts Institute of Technology (2020)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
May 2024

Author . . . .
Department of Aeronautics and Astronautics
February 29, 2024

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

Certified by . . . .
Saurabh Amin
Professor of Civil and Environmental Engineering
Thesis Committee Member

Certified by . . . .
Gil Zussman
Professor of Electrical Engineering, Columbia University
Thesis Committee Member

Accepted by . . . .
Jonathan How
Richard C. Maclaurin Professor in Aeronautics and Astronautics
Chairman, Graduate Program Committee

Analysis and Optimization of Networks in Overload

by

Xinyu Wu

Submitted to the Department of Aeronautics and Astronautics
on February 29, 2024, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Network overload occurs when the demand of network users exceeds the network capacity. The increasing gap between the growth of network demand and capacity, resulting from the surge in data-intensive machine learning applications and the slowdown in Moore's law, led to more frequent and severe occurrences of network overload in recent years. Severe overload induces heavy congestion with high delay and packet loss, impairing network performance. In this thesis, we develop new models and methods to gain in-depth understanding of network overload in two aspects: (i) designing network control policies to optimize network performance; (ii) quantifying the capability of malicious routing attacks to induce network overload.

We first investigate network policy design to optimize multiple network performance metrics including *delay*, *fairness*, and *stability*. We leverage a deterministic fluid queueing model which regards packets as continuous flows to overcome the bottlenecks of the classic stochastic models with discrete packets in order to optimize the three metrics. (i) Delay: We establish the sets of transmission policies that can minimize the average and maximum queueing delay in both single-hop and multi-stage switching networks explicitly. We term the policies rate-proportional policies since they require an identical ratio between the ingress and egress rates of different nodes at the same layer of the network. We further generalize them to queue-proportional policies, which asymptotically minimizes queueing delay based on the real-time queue backlogs agnostic of packet arrival rates. (ii) Fairness: We identify that the existing policies that can balance the overload in networks with unbounded node buffers may not work given bounded buffer sizes. We propose a policy that combines Maxweight scheduling and Backpressure routing which can reach the most balanced queue overload in networks with bounded buffers. (iii) Stability: We demonstrate that the introduction of bounded node buffers affects the transmission policies that can guarantee queue stability and avoid queue overload. We derive an explicit set of transmission policies that can guarantee queue stability in both single-commodity and multi-commodity networks with bounded node buffers.

We then quantify the capability of network adversaries to induce network overload via *routing attacks*, where a subset of nodes is hijacked and the adversary can manipulate their packet forwarding. We consider two objectives of routing attacks: *no-loss throughput minimization* and *loss maximization*. The first objective attempts to minimize the network's throughput that is guaranteed to survive, and the second objective attempts to maximize the packet loss due to link overflow given the traffic demand. We start from networks with static routing. We propose exact algorithms in general multi-hop networks for the first objective, and two approximation algorithms with multiplicative and additive guarantees in single-hop networks for the second objective. We then extend our approach to networks with dynamic routing, where nodes that are not hijacked can optimize their routing policies to maximize network throughput in response to routing attacks on hijacked nodes. We show that the two objectives are equivalent in this case, and propose an algorithm based on dynamic programming with performance guarantee when the hijacked nodes are in a chain structure or parallel to each other. We demonstrate the near-optimality of the proposed algorithms through simulations over a wide range of network settings with either static or dynamic routing control, which demonstrates their ability to evaluate the potential throughput loss due to overload under malicious routing, and identify the critical nodes to be protected to reduce the impact of routing attack.

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

Thesis Committee Member: Saurabh Amin
Title: Professor of Civil and Environmental Engineering

Thesis Committee Member: Gil Zussman
Title: Professor of Electrical Engineering, Columbia University

Acknowledgement

First and foremost, I would like to express my sincere thanks to my advisor Professor Eytan Modiano for all the help and instructions on my academic career and life through the past five and a half years. I appreciate the precious experiences in exploring different research topics and become much clearer on how research should be done based on our discussion. I am also very thankful for Professor Modiano's suggestions when I met obstacles in pushing forward research progress and felt perplexed on what research topics we should choose. Moreover, I acknowledge much about the opportunities Professor Modiano gives to me to present our work and get to know more people in multiple conferences and seminars.

Next, I would like to give my thanks to the other members in my thesis committee: Professor Saurabh Amin and Gil Zussman. I really appreciate the valuable suggestions on my research directions and techniques during our meeting, which inspired me to make progress on the research in this thesis. I would also thank the two thesis readers: Jun Sun and Dan Wu, who went through my thesis thoroughly and proposed valuable comments on the thesis structure and technical writing.

I am also grateful to all the team members in our lab CNRG: Jianan Zhang, Qingkai Liang, Rajat Talak, Igor Kadota, Dan Wu, Vishrant Tripathi, Jerrod Wigmore, Chirag Rao, Nicholas Jones, Quang Minh Nguyen, Sathwik Chadaga, Rudrapatna Vallabh Ramakanth, Darin Jeff, and so on. I enjoy very much the times with you discussing problems, having group meetings, and going hiking and skating.

I am also thankful to all the other professors, friends, and staffs who helped me a lot in MIT. I remember discussing my first research on power systems with Professor Marija Ilic, and sharing my idea with Professor Moe Win, where I gained plenty of new insights and ideas. Also I would like to give thanks to all the professors that taught me courses. For my friends, I really appreciate the help from you and the times with you, going for meals, hiking, gym, and also sharing the challenges in finding jobs. I cannot imagine how I can go over the challenges and finally get here without your supports. Also I sincerely thank all the staffs in MIT AeroAstro Department and LIDS, for supporting our academic environment from different aspects.

I would also thank the funding sources of my research projects: CNS-1735463 and CNS-1907905 from National Science Foundation (NSF) grants, HDTRA1-13-1-0021 and HDTRA1-14-1-0058 from Defense Threat Reduction Agency (DTRA) grants.

Finally, I would like to thank my parents and family members. I really appreciate that all of you provide me with sufficient food and education, support me when I am down, and give me the chance to decide my own career path and develop my interests. Thanks very much!

Contents

1 Introduction **23**

1.1 Literature Review 25

1.1.1 Delay 25

1.1.2 Overload Balancing 26

1.1.3 Network Stability 27

1.1.4 Network Attacks 28

1.2 Contributions 30

1.2.1 Chapter 2: Queueing Delay Minimization in Overloaded Networks 30

1.2.2 Chapter 3: Overload Balancing in Networks with Bounded Buffers 31

1.2.3 Chapter 4: Queue Stability in Networks with Bounded Node Buffers 32

1.2.4 Chapter 5: Routing Attack on Network Overload with Static Routing 32

1.2.5 Chapter 6: Routing Attack on Network Overload with Dynamic Routing 33

2 Queueing Delay Minimization in Overloaded Networks **35**

2.1 Motivating Example 36

2.2 Models, Definitions and Problem Formulation 37

2.2.1 Network Models: Topology and Dynamics 38

2.2.2 Network Overload 41

2.2.3 Queueing Delay Characterization 42

2.2.4 Problem Formulation 44

2.3 Static Min-Delay Policy Design 46

2.3.1 $N \times 1$ Networks 46

2.3.2 General Single-Hop and Multi-Stage Networks 53

2.4 Queue-based Min-delay Policy Design 56

2.4.1 $N \times 1$ Networks 56

2.4.2 General Single-Hop and Multi-Stage Networks 58

2.5 Performance Evaluation 60

2.5.1 $N \times 1$ Networks 60

2.5.2 General Single-Hop Networks 61

2.5.3 General Multi-Stage Networks 62

2.6 Extensions and Discussions 64

2.6.1 Practical Extensions 65

2.6.2 Theoretical Extensions 68

2.7 Summary and Future Work 72

2.8 Chapter Appendix 72

2.8.1 Proof of Theorem 2.2 72

2.8.2 Proof of Theorem 2.3 75

2.8.3 Proof of Theorem 2.8 79

3 Overload Balancing in Networks with Bounded Buffers 81

3.1 Motivating Example 82

3.2 Model and Problem Formulation 83

3.2.1 Queue Dynamic Model 83

3.2.2 Problem Formulation: Overload Balancing 85

3.3 Quadratic Sum Minimization Leads to Lexicographic Minimum . . . 86

3.3.1 Quadratic Sum Minimization to Maximum Overload Rate Minimization 87

3.3.2 Quadratic Sum Minimization to Lexicographic Minimum . . . 88

3.4 Maxweight + Backpressure Leads to Most Balanced Overload 89

3.4.1 Methodology 90

3.4.2 Maxweight + Backpressure Policy in Differentiable Form . . . 91
3.4.3 MW+BP in Single-hop Networks with Sufficient Capacity . . 93
3.4.4 MW+BP in Single-hop Networks with Limited Capacity . . . 94
3.4.5 Practical Extensions 96
3.5 Performance Evaluation 98
3.5.1 Single-hop Networks 99
3.5.2 Clos Network Structure 102
3.6 Summary and Future Work 103
3.7 Chapter Appendix 105
3.7.1 Proof of Lemma 3.1 105
3.7.2 Proof of Theorem 3.2 107

4 Queue Stability in Networks with Bounded Node Buffers 111
4.1 Motivating Example 111
4.2 Single-Commodity System 112
4.2.1 Basic Setting 113
4.2.2 Stability Analysis 115
4.2.3 Existence of Equilibrium Point 119
4.3 Multi-Commodity System 121
4.3.1 Basic Setting 121
4.3.2 Stability Analysis 121
4.3.3 Existence of Equilibrium Point 124
4.3.4 Rule of Thumb 125
4.3.5 Shared buffer Case Study: Switched Networks 128
4.4 Summary and Future Work 131
4.5 Chapter Appendix 131
4.5.1 Proof of Theorem 4.1 131
4.5.2 Proof of Lemma 4.6 133
4.5.3 Example of Lemma 4.1 134
4.5.4 Necessity for Queue Length Stability 135

5 Routing Attack on Network Overload with Static Routing 137

5.1 Network Models and Problem Definition 138

5.1.1 Network Models 138

5.1.2 Problem Definition 141

5.2 No-Loss Throughput Minimization 143

5.2.1 Problem Formulation 143

5.2.2 Exact Algorithms 145

5.2.3 2-Approximation Algorithm with Partial Information 149

5.2.4 Practical Extensions 153

5.3 Loss Maximization 156

5.3.1 Problem Formulation and NP-Completeness 156

5.3.2 Approximation Algorithms 159

5.3.3 Practical Extensions 163

5.4 Optimal Node Selection 164

5.4.1 Problem Formulation and NP-Completeness 165

5.4.2 Algorithms Under Parallel $\mathcal{V}_{cand}$ 166

5.5 Performance Evaluation 167

5.5.1 No-Loss Throughput λ^* Minimization 169

5.5.2 Loss Maximization 171

5.5.3 Optimal Node Selection 175

5.6 Summary and Future Work 176

5.7 Chapter Appendix 177

5.7.1 Proof of Theorem 5.5 177

5.7.2 Proof of Theorem 5.6 178

5.7.3 Proof of Theorem 5.7 181

5.7.4 Proof of Theorem 5.8 183

6 Routing Attack on Network Overload with Dynamic Routing 185

6.1 Model, Problem and Basic Results 186

6.1.1 Network Model 186

6.1.2 Problem Formulation 187
6.1.3 Basic Results 189
6.2 Algorithms and Performance Guarantees 190
6.2.1 Chain Structure 190
6.2.2 Parallel Structure 194
6.3 Algorithm for General $\mathcal{V}_A$ 198
6.4 Performance Evaluation 201
6.5 Summary and Future Work 204
6.6 Chapter Appendix 205
6.6.1 Proof of Proposition 6.1 205
6.6.2 Proof of Theorem 6.1 207
6.6.3 Proof of Theorem 6.2 208
6.6.4 Proof of Theorem 6.3 209
6.6.5 Proof of Theorem 6.4 211
6.6.6 Proof of Theorem 6.5 213

7 Concluding Remarks **215**

List of Figures

2-1 An example of a 2×1 single-hop network: Under maxweight policy, link (s_1, d) is always activated while (s_2, d) is blocked since the queue in s_1 grows with rate $\lambda_1 - c_1 = 4$ while the queue in s_2 grows with rate at most $\lambda_2 = 3$. 37

2-2 (a) A single-hop network structure; (b) A switched network with ingress and egress ports; (c) A server farm with load balancers as ingress and servers as egress. 38

2-3 An example of a 4-layer multi-stage network 41

2-4 An example of queueing delay characterization of a packet passing two nodes 43

2-5 An example of an $N \times 1$ single-hop network 46

2-6 A 3x1 example of Theorem 2.1: (a) Setting $g_i/\lambda_i = 1/2, i = 1, 2, 3$ satisfies (2.10) and leads to minimum delay; (b) Setting $g_i \geq \lambda_i, i = 1, 2, 3$ satisfies (2.10) and leads to minimum delay, despite different queue growth rates compared with (a); (c) Setting $\mathbf{g} = \{3, 5, 4\}$ does not satisfy (2.10) and thus does not incur minimum $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$, although all the 3 links rates are greater than those in (a), primarily due to the higher congestion level at s_3. 50

2-7 The min-delay region in a 2×1 single-hop network: (a) unlimited capacity; (b) limited capacity ($c_i \leq \lambda_i,\ i = 1, 2$) 51

2-8 A 2x2 example of Theorem 2.2: (a) and (b) set $\mathbf{g}$ that satisfies $(g_{s_1d_1}+g_{s_1d_2})/(g_{s_2d_1}+g_{s_2d_2}) = \lambda_1/\lambda_2$ and $(g_{s_1d_1}+g_{s_2d_1})/(g_{s_1d_2}+g_{s_2d_2}) = \mu_1/\mu_2$ which lead to minimum $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$ simultaneously; Setting link rates as in (c) does not lead to minimum delay since $(g_{s_1d_1} + g_{s_2d_1})/(g_{s_1d_2} + g_{s_2d_2}) \neq \mu_1/\mu_2$, although the total link rates are higher than (a) and (b). 54

2-9 An example of link rate control in a 4-layer multi-stage network which minimizes $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$. The ingress and egress rates of nodes at layer 1 to 4 are 1, 6/5, 5/4, and 4/3 respectively. 56

2-10 CDFs of $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$ in 32×1 single-hop networks with sufficient capacity (OPT and MAX are overlapped) 61

2-11 CDFs of $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$ in 32×1 single-hop networks with limited capacity 62

2-12 CDFs of $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$ in 32×16 single-hop networks 63

2-13 CDFs of $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$ in $16 \times 12 \times 8 \times 6$ networks 65

2-14 CDFs of $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$ in $15 \times 12 \times 9 \times 12 \times 15$ networks 65

2-15 A 3-layer fan-in tree data center example under min-delay transmission policies derived in Theorem 2.8. 69

2-16 Validation of Conjecture 2.1 in 2×2 single-hop networks, where the notation $x(y)$ over a link (i, j) means $g_{ij} = x$ and its actual transmission rate $\bar{g}_{ij} = y$ calculated based on (2.23): (a) setting $\mathbf{g} = [4, 4, 5, 5]$ leads to $\tilde{\mathbf{g}} = [2, 2, 4, 4]$ which satisfies (2.18) thus min-delay; (b) setting $\mathbf{g} = [4, 4, 5, 15]$ leads to $\tilde{\mathbf{g}} = [2, 2, 2, 6]$ which does not satisfy (2.18) thus not min-delay. 71

2-17 An example of a 3-layer $2 \times 2 \times 2$ network 76

3-1 Suppose all ingress nodes have unbounded buffer. When node d has unbounded buffer, backpressure achieves most balanced overload rates where all 4 nodes grow with rate 3.75; When node d has finite buffer, then $\dot{q}_d = 0$ in the steady state due to buffer saturation, and backpressure achieves overload rates $(\dot{q}_{s_1}, \dot{q}_{s_2}, \dot{q}_{s_3}, \dot{q}_d) = (2, 5, 8, 0)$, deviating significantly from the most balanced one $(5, 5, 5, 0)$. 82

3-2 Geometric interpretation of Lemma 3.1 through a 2×1 single-hop network. The contour curves of (3.3) in red and (3.4) in green coincide at the same optimal point B on the boundary of $\mathcal{G}$ under different arrival rate vectors $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)$, denoted as point A where $\lambda_1 + \lambda_2 > \mu$. 89

3-3 Condition 1 (existence) and 2 (convergence) to verify that a queue-based policy achieves most balanced overload 91

3-4 Example that (3.9) does not achieve most balanced overload with limited capacity with $\gamma \to \infty$, where node d has bounded buffer. On the left, $(g_1, g_2, g_3) = (2.5, 0.5, 0)$ denotes the transmission rates under (3.9) in steady state, under which $\dot{\mathbf{q}} = [5.5, 5.5, 4]$. It is not the most balanced overload as presented on the right, where $\dot{\mathbf{q}}^* = [5, 3.5, 3.5]$ under $(g_1^*, g_2^*, g_3^*) = (3, 2.5, 0.5)$, the optimal solution to (3.3). Other γ values also suffer from the suboptimality. 94

3-5 Quadratic Sum Gap Comparison in Steady State 101

3-6 Max Overload Rate Gap Comparison in Steady State 101

3-7 Transient Process of Quadratic Sum Gap 102

3-8 Example of a 3-stage Clos structure from [1] 103

3-9 Quadratic Sum Gap Comparison (Clos) 104

3-10 Max Overload Rate Gap Comparison (Clos) 104

4-1 Finite buffer may affect stability result. On the right is an example of the queue dynamics in node K following the backpressure policy. The dashed frame denotes the packets to be served at the current time step. In the final state, the average number of commodity 2 packet in the buffer is 1.5, which arises from $1.5 = \frac{\mu_1}{c_{1K}} c_{2K}$ with details deferred to Section 4.3.3. Therefore the actual throughput of commodity 2 is 1.5, less than $\mu_2 = 3$, due to the finite buffer. 113

4-2 One-hop system with C commodities 130

5-1 (a) Switch network; (b) Server farm; (c) Bipartite graph (in the dashed box) with meta source s_0 and destination d_0 140

5-2 (a) A 6-node network with $\mathcal{V}_A = \{3\}$ and $\mathcal{V}_N = \{1, 2, 4, 5, 6\}$; (b) Optimal routing to minimize λ^* is $(x_{34}, x_{35}) = (0, 1)$: λ^*_{OPT} is 2 and $(5, 6)$ is the first saturated link; (c) Given $\lambda = 10$, the optimal routing to maximize loss is $(x_{34}, x_{35}) = (1, 0)$, with maximum loss of $\lambda - c_{46} = 10 - 3 = 7$. 142

5-3 Boundary Optimality: One of the 4 combinations must be optimal, where $\mathcal{V}_A = \{2, 3\}$ serve all traffic through the highlighted links. . . . 146

5-4 Example of Algorithm 5.1. Assume that $c_{12}, c_{13} \to \infty$ which means $(1, 2)$ and $(1, 3)$ will not be the first saturated links. We can calculate $MF[2] = x_{12}$, $MF[3] = 1$ since the adversarial node 2 can route all packets to 3, $MF[4] = x_{12} + x_{13}x_{34}$ since node 2 can route all packets to 4, and $MF[5] = 1$ since node 4 can route all packets to 5. Then we find the first saturated connected downstream link of each node in $\{2, 3, 4, 5\}$, where the calculation in (a) and (c) follows step 5 since $2, 4 \in \mathcal{V}_A$ with links highlighted in red, while that in (b) and (d) follows step 4 since $3, 5 \in \mathcal{V}_N$ highlighted in blue. 148

5-5 Example of running Algorithm 5.2 from (a) to (c) 150

5-6 Example of Algorithm 5.3. The original traffic flows are marked by the numbers over each link in (a), and we can measure the total throughput $F_{total} = 7$ at the destination node 6. We initialize $F[2]$ and $F[4]$ to be the total flow to node 2 and node 4, which are 3 and 5 respectively shown in (b). Since node 2 is the source node in $\mathcal{G}_A^{down}$, $F[2] = 3$ does not need to be updated since all the flows from node 2 to 3 and 4 only traverse node 2. Then in (c) we calculate the flows at node 4 that have traversed node 2, which contains paths $2 \to 4$ and $2 \to 3 \to 4$ with a total of $7/9 \times F[2]$. Then in (d) we substract these flow from $F[4]$ and we get the final $F[4] = 8/21$. 152

5-7 Example of $2^{|\mathcal{V}_A|} \times \lambda^*_{OPT}$ under Algorithm 5.4. Suppose $\mathcal{V}_A = \{2, 4, 6\}$; $c_{28} = 4 - 4\epsilon$, $c_{48} = 2 - 2\epsilon$, $c_{68} = 1 - \epsilon$ where $0 \leq \epsilon \leq 1$, $c_{78} = 1$, and other links have infinite capacity. Nodes 1, 3, and 5 route half of the traffic through each connected downstream link. Algorithm 5.1 routes all traffic through red links which leads to $\lambda^* = \lambda^*_{OPT} = 1$, while Algorithm 5.4 routes all traffic through blue links (topological order $6 \to 4 \to 2$) which leads to $\lambda^* = 8\lambda^*_{OPT} = 8$ when $\epsilon \to 0$. 154

5-8 Graph Construction from Set Cover 159

5-9 Tightness validation of Theorem 5.5 under $|\mathcal{V}_A| = 4$. Consider $\epsilon \to 0$, Approach 1 on network (a) and Approach 2 on network (b) leads to $1/|\mathcal{V}_A| = 1/4$ approximation ratio; Algorithm 5.5 gives worst-case approximation ratio $1/\sqrt{|\mathcal{V}_A|} = 1/2$ on network (c). 162

5-10 Tightness validation of Theorem 5.6 ($\lambda = 4$, $\mathcal{V}_A = \{s_2\}$): red route is optimal, blue route is the output of Algorithm 5.6. 163

5-11 CDFs of approximation ratio under density 0.8 and $|\mathcal{V}_A| = 20$ 168

5-12 CDFs of approximation ratio under density 0.3 and $\mu/\lambda = 2$ in 8×8 networks 173

5-13 CDFs of loss ratio under density 0.25 and $\mu/\lambda = 4$ in 16×16 networks 175

5-14 CDFs of approximation ratio of overload under density 0.3 and a (μ, λ)-ratio of 2 in 16×16 networks 176

5-15 Examples for Proof of Approach 2 179

5-16 Proof for Case 3 of Theorem 5.6 181

5-17 Proof for Case 4 of Theorem 5.6 182

5-18 NP-hardness of Node Selection Problem 183

6-1 Examples of reachability and the chain structure of $\mathcal{V}_A$. 191

6-2 Alg. 6.1 on Fig. 6-1(b): first on the left, we temporarily set the routing policy of node 2 to be $x_{23} = 1$ so that all the traffic at 2 is sent to 4, and determines the routing policy at 4 to be $x_{46} = 1$ to achieve minimum s-d cut value 1, under which no traffic goes through $(2, 5)$ and $(4, 5)$; then on the right, we fix $\mathbf{x}_4$ and determine the routing policy at 2 to be $x_{23} = 1$ which keeps the minimum s-d cut value to be 1. 192

6-3 Example where adversarial nodes at different layers (shaded in black) in a multi-layer network that are not in a parallel structure, although $i \not\to j$ for $\forall i, j \in \mathcal{V}_A$ and $i \neq j$. 195

6-4 Example of the first iteration of Alg. 6.2 for $\mathcal{V}_A$ in a parallel structure: first set $c_{ij} = 0$ for all the links (i, j) starting from an adversarial node i; then follow line 5 to 7 to calculate the minimum s-d cut value increment by activating each of the above links, and follow line 9 to 11 to calculate the minimum s-d cut value increment by activating all adversarial nodes reachable to each normal node, and obtain the mean minimum cut value increment by dividing the value by the number of adversarial node reachable to this normal node. 197

6-5 Example of Alg. 6.3 on $\mathcal{V}_A$ with a combination of chain and parallel structures. Given that $\mathcal{V}_A = \{2, 3, 4, 5\}$ in (a) where $2 \to 5$ and $3 \to 5$, the algorithm temporarily sets $x_{25} = 1$ and $x_{35} = 1$ (line 5) and determines the routing policies at 4 and 5 (line 6) as shown in (b); suppose that in line 6 the algorithm determines that $x_{57} = 1$ based on $\Delta_{57} = \min\{\{\Delta_{ij}\}_{i \in \mathcal{V}_A, (i,j) \in \mathcal{E}} \cup \{\Delta_k / |\mathcal{R}_k|\}_{k \in \mathcal{V}_N, \mathcal{R}_k \neq \emptyset}\}$, then it starts to consider adversarial nodes upstream to 5 in the chains, i.e., determines the routing policies at $\{2, 3, 4\}$. 199

6-6 The top-down dynamic programming mechanism in Alg. 6.3 on the example in Fig. 6-3. 200

6-7 CDFs of approximation ratios of Dynamic Programming (DP), Local Search (LS), and Alg. 6.3 to the optimal solution to (6.2). 203

6-8 CDFs of approximation ratios to the optimal solution to (6.2) when $\mathcal{V}_A$ is in a general structure ($|\mathcal{V}_A| = 10$). 204

6-9 Example of a constructed network based on an instance of Set Cover. 209

6-10 Example where COND holds: the optimal routing attack can be decomposed into 2 iterations, where in the first iteration it determines the routing policies of $\mathcal{R}_7$, and in the second iteration it determines those of $\mathcal{R}_5$ whose routing policies have not been determined in previous iterations. 212

List of Tables

2.1 Mean and maximum ratios between the two policies for comparison and the min-delay policy in terms of $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$ in 32×16 single-hop networks 63

2.2 Mean and maximum ratios between each of the two tested policies for comparison and the proposed min-delay policy in terms of $\bar{D}_{\text{avg}}$ (Column 1) and $\bar{D}_{\max}$ (Column 2), and the ratios $\bar{D}_{\max}/\bar{D}_{\text{avg}}$ of the tested policies themselves reflecting the delay imbalance of packets injected into different ingress nodes (Column 3), over 7 multi-stage topologies. 64

5.1 Approximation ratio statistics for Algorithm 5.2 (90% means 90-percentile) 168

5.2 Statistics of performance guarantee for (5.6) in 8×8 networks under different network settings (μ/λ means (μ, λ)-ratio; 10%, 90% mean 10-percentile, 90-percentile) 173

5.3 Mean loss ratio in 16×16 networks 174

6.1 Approximation ratio statistics of Dynamic Programming (DP), Local Search (LS), and Alg. 6.3 (90% means 90-percentile). 203

Chapter 1

Introduction

Network overload occurs when the total demand of network users exceeds the network capacity [2, 3]. Severe overload can impair network performance resulting in throughput reduction [4, 5], increased latency [6, 7], or unfairness where some sessions starve other sessions [3, 8]. Network overload becomes more frequent in data center networks due to the increase in network demand, notably driven by the upsurge of connected devices together with the exponentially increasing number and size of machine learning applications [9–11]. Meanwhile, the slowdown in Moore's law compared with traffic growth further increases the likelihood of overload [12, 13]. The increasing demand-capacity gap poses challenges for network service providers, that need to develop network policies that can utilize communication bandwidth more efficiently to improve network performance during overload [14–16]. Another source of network overload is the reduction of network capacity, during maintenance and upgrade of data center networks [1, 14], unexpected failures of network nodes and links [14, 17], and cyberattacks such as denial-of-service attack which occupies transmission resources [18, 19], and node hijacking which redirects the traffic to longer paths or blackholes [20, 21]. Maintaining the network performance under the temporary loss of network capacity requires high availability, reliability, and robustness of network systems under the potential overload.

Previous research investigated optimal policies to avoid or mitigate network overload and guarantee network performance. The seminal work from Tassiulas and

Ephremides showed that the backpressure routing policy can avoid queue overload as long as the packet arrival rates to the network are within the capacity region of the network [22]. Their result reveals the sufficient and necessary condition that can guarantee bounded queue backlogs in networks with unbounded node buffers, and serves as a basic policy design framework for a large body of works that capture utility maximization [23–26], delay minimization [25, 27, 28], and network fairness [3, 8, 24, 29]. When the packet arrival rates are beyond the capacity region, network overload cannot be avoided. In this case, an important measure to mitigate the consequences of severe overload is to balance the overload so as to ensure that all sessions are equally affected by the congestion. In [3], Georgiadis and Tassiulas demonstrated that the backpressure policy can achieve most balanced overload in a network with unbounded buffers. This work inspired studies on overload balancing considering different network constraints and topologies [8, 23, 29, 30]. A broader range of studies include reducing network latency when data center networks are overloaded [7, 31, 32], and quantification of the capability of cyberattacks to induce network overload [18, 19, 33, 34].

In this thesis, we build upon existing research to attain more in-depth understanding of network overload. We overcome the limitations of previous work and make progress by leveraging a deterministic fluid queue model to characterize the network dynamics under overload. We first develop link rate control policies to minimize the queueing delay of packets in overloaded multi-stage switching networks in data center infrastructures. We demonstrate that the policies that achieve close-to-minimum delay when the network is not overloaded may lead to bad delay performance in overloaded networks. We then develop transmission policies that balance the overload over different network nodes when packet arrival rates are beyond the capacity region, and guarantee queue stability to avoid overload otherwise, in networks with bounded node buffers. We point out that the bounded buffers can significantly affect the optimal policy for both overload balancing and network stability. We further quantify the capability of routing attacks to induce network overload. Routing attacks, such as BGP hijacking [20, 35, 36] and routing table poisoning [37, 38], are common in practice, their impact on network overload has not been studied as thoroughly as denial-of-service [18, 19] and node removal attacks [33, 34].

1.1 Literature Review

1.1.1 Delay

Reducing network delay is crucial to both network users and enterprises, given delay-sensitive applications such as short-form videos and quantitative trading [39]. Service level delay objectives are difficult to meet under overload in large-scale data centers [7]. Enterprise revenues are sensitive to delay: Google reports that advertisement revenues will decrease by 20% if web search delay increases from 0.4s to 0.9s, and Amazon reports that an extra 100ms response time decreases the sales by 1% [40].

The delay increase caused by network overload is mainly due to the *queueing delay* of packets, since the queue buffers are increasingly backlogged due to the overload. A common approach to reducing queueing delay is active queue management, which drops packets when overload is observed or the queueing delay exceeds a specific upper bound [31, 32, 41]. Alternative approaches include network calculus-based scheduling with worst-case latency guarantees [7, 42], traffic shaping and pacing [43, 44], and smart buffer design to absorb traffic spikes [45]. However, these approaches are heuristic with no performance guarantees. Understanding optimal policies that globally minimize queueing delay holds potential for further delay reduction, which remains a hard problem [46–48]. Extensive previous work unveils the power of load balancing approaches in achieving close-to-zero queueing delay over heavily-loaded parallel servers [40, 49–52]: The Join-the-shortest-queue policy is proven asymptotically delay-optimal [40], and a power-of-d-choices policy can reduce communication overhead [52]. Other methods to reduce queueing delay include packet replication [40] and network coding [53]. Techniques like Kleinrock

Independence Approximation have been applied to approximate the mean queueing delay of packets in general networks [54–58]. However, developing policies to minimize the queueing delay remains intractable.

Joint optimization of queueing delay and other metrics is another promising research direction which reveals the trade-off between minimizing delay and optimizing other metrics and guides policy design to balance the metrics for networks in practice. Georgiadis et al. revealed a fundamental $\{O(V), O(1/V)\}$ network utility-delay tradeoff under backpressure [59], and Huang et al. proved that under a LIFO-backpressure policy such tradeoff becomes near-optimal [25]. Zhao et al. pointed out energy consumption-delay tradeoff [60], and Talak and Modiano unveiled the trade off between age of information and delay [61]. Moreover, previous research has validated that the measured delay can facilitate policy design for the optimization of other metrics. Neely proposed a delay-based scheduling policy to achieve stability-utility joint optimization [47]. Ji et al. studied a delay-based backpressure policy to guarantee the stability of networks supporting multiple traffics with fixed routes [46]. Cardwell et al. utilized round-trip time to effectively reduce congestion [62].

1.1.2 Overload Balancing

The concept of overload balancing stems from load balancing, which aims to distribute traffic across multiple servers in large-scale network infrastructures to increase the network efficiency and avoid severe congestion [63, 64]. Extensive studies have been devoted to load balancing under server farms and cloud systems with elegant theory under stochastic queueing model [50, 65, 66] and widespread implementation in industry [67, 68]. However there is a lack of discussion and analytical results on overloaded networks where network demand surpasses capacity. As [3, 4] pointed out and we show in this work, overload balancing makes a difference compared with load balancing due to the buffer saturation.

A number of works considered the problem of overload balancing. Based on the network fairness literature in flow control [69], Georgiadis and Tassiulas demonstrated that the backpressure policy in [22] can achieve most balanced overloading in networks with unbounded buffers [3]. Their criterion of the most balanced state is that the queue overload rate vector achieves the lexicographic minimum, a concept related to min-max optimization [70]. More recent works study specific network structures. For parallel queues, [8] considered overload balancing by introducing explicit constraints on fairness level, and [23] studied packet dropping policies to control the flow. For server farms, [29] generalized different fairness notions through α-fair penalty functions, which allowed for a convex optimization formulation. For cloud systems, [30] studied detection and balancing the transient overload through distributed optimization.

1.1.3 Network Stability

In the seminal paper on network stability, Tassiulas and Ephremides showed that backpressure routing can stabilize networks whenever the packet arrival rates are within the network stability region [22]. Their result elegantly solves the network stability problem for systems with unbounded buffers, and serves as a basic policy design framework for a large body of works for utility maximization [23–26], delay minimization [25, 27, 28], and network fairness [3, 8, 24, 29].

Bounded Node Buffers

However, most of the related works rely on the assumption of unbounded buffers, which deviates from the fact that in reality buffers are finite [71]. In practice, internal nodes in a communication network often have limited buffers [72, 73]. For example, on-chip networks have very small internal buffers due to area and power limitation, and similarly, satellite networks have small buffers on-board the satellite. In constrast, buffers of the source nodes of the arriving packets have sufficient capacity to absorb bursty packet arrivals [2], e.g. in a satellite network the buffer at the ground terminal can be relatively large.

A plethora of works have tried to incorporate bounded buffer sizes in network analysis. Giaccone *et al.* studied the throughput region of network systems with bounded node buffers and discussed the relationship between buffer size and throughput [71]. Le *et al.* studied the relationship between buffer size and network utility under a modified backpressure mechanism [2], and Lien *et al.* designed a dynamic algorithm to stabilize any admissible traffic conditioned on a finite internal buffer with size larger than a certain bound [74]. All of these works proposed policies and analyze their performance on finite-buffer systems, under certain assumptions such as deterministic routing [71], separate buffers for different commodities [2, 71], equal buffer sizes [2, 71], and minimum buffer size requirements [74]. A systematic study of the policies that can achieve queue stability in buffered networks in general buffer size setting is still necessary.

Fluid Queue Model

We leverage the fluid queue model to characterize the queueing dynamics of a network. The fluid queue model in this thesis resembles the fluid model [75, 76] which was proposed as a flow-based approximation to the discrete network systems to obtain results for throughput [4], fairness [3] and delay [77]. However, that fluid model captures the scaled limit of the queue backlogs, which for nodes with bounded buffers is not very meaningful. A closely related framework is the ordinary differential equation (ODE) model used to study the Transmission Control Protocol (TCP) [78, 79]. Although sharing similar modeling of the queue dynamics, we point out that the fluid queue model in this thesis can capture more general policies.

1.1.4 Network Attacks

The prevalence and severity of network attacks exhibited a notable escalation in recent times, as evidenced by the growing number of reported incidents and their increasingly profound ramifications [20]. These attacks often result in substantial degradation in network performance, such as lower throughput and higher latency. The 2018 Pakistan Telecom hijacking incident caused considerable network disruption with extensive delays [80]. The 2016 Dyn DDoS attack led to a wide range of outages and significantly downgraded user experiences on Twitter and Netflix [81].

We focus on the degradation of network performance due to routing attacks, wherein adversaries hijack network servers to manipulate their routing decisions [82]. Routing attacks are a notable form of network node attacks that have broad impact, which can last for several hours or even longer than a day before resolved [20, 21]. Furthermore, their impact can be significant over the Internet. Incidents have been reported where routing attacks in 10 minutes polluted 90% of the network users [83]. Examples of routing attacks include BGP hijacking, where an attacker falsely claims ownership of an IP prefix to affect routing [35,36]; routing table poisoning, where false routing information is injected into a victim's routing table [37, 38]; OSPF attacks, which involve fabricating topology information to control routing [84]; and blackhole attacks, which divert the traffic to non-existent destinations [85]. Routing attacks have been detected in a wide range of networks, including data center networks [86], software-defined networks [87], wireless ad hoc networks [37], and robotics networks [88]. Over 40% of operators reported that their organizations have fallen victim to node hijacks [20].

The quantification of routing attacks' impact on network overload remains a relatively unexplored research area [20], unlike other attack types including denial-of-service [19], link removal [33], and node removal attacks [34]. We are motivated to study this problem since both routing attacks and network overload pose greater challenges as the network scale and user demand keep growing. Routing attacks are concerning due to their low implementation cost for adversaries [86]. They may lead to broader impact in data center networks that apply software defined networks, where a hijacked controller in the control plane may affect the routing policies of multiple nodes in the data plane [14,89]. However, current defense mechanisms against routing attacks are inadequate, and the detection of these attacks is challenging [20].

1.2 Contributions

In this thesis, we analyze network overload from two major perspectives. We first consider the side of network service providers to develop optimal network policies to guarantee network performance: minimizing queueing delay, balancing the overload across network nodes, and avoiding the overload so that queue stability is guaranteed. We then stand on the side of network adversaries to quantify their potential for causing network overload through routing attacks. We develop algorithms that can accurately estimate the most severe overload level by routing attacks on a subset of hijacked nodes, which can serve as the benchmarks for the evaluation of network vulnerability and guide the protection of critical nodes under different routing attacks. We summarize the contributions in each chapter of this thesis below.

1.2.1 Chapter 2: Queueing Delay Minimization in Overloaded Networks

We summarize the contributions on queueing delay minimization in overloaded networks as follows. (i) We derive explicit conditions on link rates that minimize both the average and maximum queueing delay of packets in general single-hop and multi-stage networks. The analytical results demonstrate that higher link rates are not guaranteed to reduce queueing delay. These conditions correspond to a *rate-proportional* policy which maintains an identical ratio between the ingress and egress rates of different nodes at the same layer, i.e., the ingress rates of all the nodes at a layer should be proportional to their egress rates. (ii) We generalize the rate-proportional policy to a *queue-proportional* policy, which can minimize queueing delay asymptotically based on real-time queue backlogs, and do not require knowledge of packet arrival rates [52, 90]. (iii) We validate that our proposed policies achieve minimum delay in various settings of single-hop and multi-stage networks, and demonstrate further delay reduction compared with benchmarks including the backpressure policy [3, 22] that maximizes network throughput and the max-link-rate policy that fully utilizes bandwidth. (iv) We demonstrate that the proposed explicit min-delay policies facilitate co-optimization with other metrics that are important in data centers: minimizing the total required bandwidth, balancing link utilization, and balancing overload rates at different node buffers. We also determine a set of more relaxed min-delay conditions for tree data center structures, and provide a conjecture on the sufficient and necessary condition minimizing queueing delay in general multi-stage networks.

1.2.2 Chapter 3: Overload Balancing in Networks with Bounded Buffers

We summarize the contributions on overload balancing in networks with bounded node buffers as follows. (i) We prove that minimizing the quadratic sum of queue overload rates leads to the minimum of the max queue overload rates among all nodes, and also lexicographic minimum of queue overload rates. The quadratic sum minimization offers an equivalent but more tractable way to analyze the most balanced overload, compared with lexicographic minimization in [3]. (ii) Agnostic of packet arrival rates and link capacities, we prove that a policy combining maxweight and backpressure (mw+bp) achieves the most balanced overload in single-hop networks, which only requires queue information. We show that our fluid queue formulation can embed queue-based policies under bounded buffers elegantly based on a novel characterization of the policy in a differentiable form. (iii) From a practical perspective, we propose a distributed version of the mw+bp policy which significantly reduces communication overhead. (iv) We verify our proposed policies by simulation in single-hop structures and their concatenations (Clos structure [1]), under randomly selected settings of packet arrival and departure rates, link capacities, and buffer settings. We show the mw+bp converges to the most balanced overload in all the test cases, while the distributed version sacrifices little optimality. Both policies work much better than pure backpressure proposed in [3] for unbounded-buffer systems.

1.2.3 Chapter 4: Queue Stability in Networks with Bounded Node Buffers

We summarize the contributions on queue stability in network with bounded node buffers as follows. (i) For single-commodity systems, we derive a sufficient condition for a set of local policies to stabilize the network based on ODE stability theory. We demonstrate that the condition has similar physical intuition to the backpressure policy; (ii) For multi-commodity systems, we similarly derive a sufficient condition for network stability, with an additional condition that captures the coupling level between different commodities. The core idea is that these conditions reduce the network stability problem to a problem of testing the existence of an equilibrium point, and thus facilitate stability analysis.; (iii) The derived sufficient condition for multi-commodity systems is not explicit enough for policy implementation in real networks. We extend the results to an explicit rule of thumb of policy design that facilitates network stability in multi-commodity systems.

1.2.4 Chapter 5: Routing Attack on Network Overload with Static Routing

We summarize the contributions on the impact of routing attacks on network overload when network nodes apply static routing policies as follows. (i) No-Loss Throughput Minimization: We develop an exact polynomial-time routing attack algorithm to minimize no-loss throughput in general multi-hop networks with arbitrary combinations of hijacked nodes. We further propose a 2-approximation algorithm for adversaries with partial information over the downstream of hijacked nodes. We further generalize the methodology to a wider range of situations including a heuristic algorithm that supports distributed attacks, constraints over the routing attacks, and extension to multi-commodity networks. (ii) Loss Maximization: We establish the NP-completeness of the loss maximization problem even in single-hop networks. We develop two approximation algorithms with multiplicative and additive performance guarantee respectively in single-hop networks. (iii) Optimal

Selection of Nodes to Hijack: We investigate the adversary's optimal selection of nodes to hijack over a set of candidate nodes to optimize the aforementioned two objectives via routing attacks. We prove the NP-completeness of this problem in general networks, propose heuristics and prove the performance guarantee for no-loss throughput minimization in special cases. (iv) Performance Evaluation: We evaluate the proposed algorithms and demonstrate their near-optimal performance under a wide range of network settings that cover different network scenarios, including different network densities, default routing policies, and number of hijacked nodes. Our results quantitatively confirm the significant threat posed by routing attacks, and demonstrate that our proposed algorithms can be used as benchmarks to quantify the overload risk given arbitrary sets of hijacked nodes and to identify the critical nodes that should be protected against routing attacks.

1.2.5 Chapter 6: Routing Attack on Network Overload with Dynamic Routing

We summarize the contributions on the impact of routing attacks on network overload when network nodes apply dynamic routing policies to maximize network throughput in response to the attack as follows. (i) We formulate a minimax optimization framework to calculate the maximum throughput loss due to adversarial routing at the attacked nodes, and demonstrate its equivalence to minimizing the minimum s-d cut value of the network. We show that this problem is NP-hard when the number of adversarial nodes scales linearly with network size, thus motivating approximation algorithm design. (ii) We develop algorithms for the adversary when the adversarial nodes are in a *chain* structure or a *parallel* structure, and prove that the algorithm for the chain structure can output a solution that maximizes throughput loss, and the algorithm for the parallel structure returns a solution with a logarithmic worst-case approximation ratio to the optimal solution. (iii) We generalize the above algorithms to an arbitrary subset of adversarial nodes, and empirically validate its near-optimal performance under a wide range of network settings with different topologies, capacity settings, and sets of adversarial nodes, through comparison with common heuristics.

Chapter 2

Queueing Delay Minimization in Overloaded Networks

In this chapter, we develop link rate control policies to minimize the queueing delay of packets in overloaded networks. We show that increasing link rates does not guarantee delay reduction during overload. We consider a fluid queueing model that facilitates explicit characterization of the queueing delay of packets, and establish explicit conditions on link rates that can minimize the average and maximum queueing delay in both single-hop and multi-stage switching networks. These min-delay conditions require maintaining an identical ratio between the ingress and egress rates of different nodes at the same layer of the network. We term the policies that follow these conditions *rate-proportional* policies. We further generalize the rate-proportional policies to *queue-proportional* policies, which minimize the queueing delay asymptotically based on the time-varying queue length while remaining agnostic of packet arrival rates. We validate that the proposed policies lead to minimum queueing delay under various network topologies and settings, compared with benchmarks including the backpressure policy that maximizes network throughput and the max-link-rate policy that fully utilizes bandwidth. We further remark that the explicit min-delay policy design in multi-stage networks facilitates co-optimization with other metrics, such as minimizing total bandwidth, balancing link utilization and node buffer usage. This demonstrates the wider utility of our main results in data center network optimization in practice.

2.1 Motivating Example

The motivation for redesigning transmission policies for queueing delay minimization in overloaded networks, is that network overload raises the technical challenge in characterizing queueing delay using stochastic models since Little's law [48], the foundation of queueing delay analysis in stationary systems, no longer holds in overloaded networks as the long-term expectation of delay is infinite. Hence, policies that achieve close-to-minimum delay when the network is not overloaded may no longer perform well in overloaded networks. We give an intuitive example in the 2×1 single-hop network of Fig. 2-1, where the link capacities are $c_1 = 4$, $c_2 = 2$, and external packets arrive at node s_i with rate λ_i $(i = 1, 2)$ and are transmitted to the shared buffer at node d whose service rate is $\mu = 2$. Suppose that at most one link can be activated at a time. In this case, the maxweight scheduling policy that activates the link that is connected to the source node with longer queue backlog [49] has been shown to have near-optimal delay performance when the network is not overloaded $(\lambda_1 + \lambda_2 < \mu)$. However, the maxweight scheduling policy fails in delay minimization for packets injected into node s_2 during overload when $(\lambda_1, \lambda_2) = (8, 3)$, as s_1 always has longer queue backlog than s_2, thus blocking packets in s_2 until the overload ends. For the case where the simultaneous activation of the two links is allowed, we show later in this chapter that neither the throughout-optimal backpressure policy [22] nor serving packets with maximum link rates can minimize delay under overload, while instead fixing the rate of link (s_1, d) to be 2, and link (s_2, d) to be 0.75, can minimize the delay in the example. These counter-intuitive observations reveal the necessity to redesign link rate control policies for queueing delay minimization in overloaded networks.

To this end, we develop a deterministic fluid queueing model that elegantly solves the technical challenges of queueing delay characterization in overloaded networks. The fluid model regards network traffic as continuous flows instead of

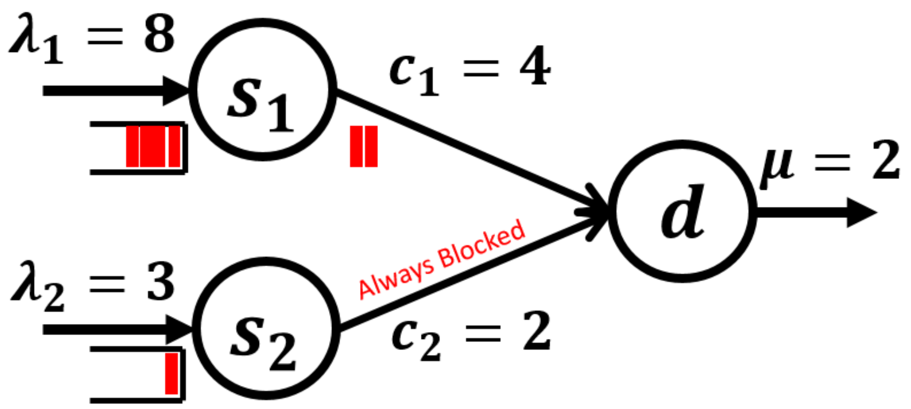


Figure 2-1: An example of a 2×1 single-hop network: Under maxweight policy, link (s_1, d) is always activated while (s_2, d) is blocked since the queue in s_1 grows with rate $\lambda_1 - c_1 = 4$ while the queue in s_2 grows with rate at most $\lambda_2 = 3$.

discrete packets. It well approximates the discrete packet transmission when the time unit is sufficiently small. Based on the model, we develop link rate control policies that minimize the queueing delay of the packets that arrive to the network within a bounded time interval, which corresponds to the duration of the overload. We demonstrate that our proposed policies minimize queueing delay in both single-hop and multi-stage switching networks, which serve as the basic structure of data center networks including Clos [1, 91, 92] and Tree [93, 94]. Hence, our results shed light on policy design to reduce delay in data centers under overload.

2.2 Models, Definitions and Problem Formulation

In this section, we introduce the single-hop and multi-stage network models and the fluid queueing model of packet flows, and define network overload. We then characterize the queueing delay of a packet based on the fluid model and formulate the queueing delay minimization problem based on the derived explicit forms of the average and maximum queueing delay of packets.

2.2.1 Network Models: Topology and Dynamics

Single-hop networks

A single-hop network contains a set of ingress nodes and egress nodes. We model an $N_S \times N_D$ single-hop network as a bipartite graph $(\mathcal{V}, \mathcal{E})$ with $\mathcal{V} := \{\mathcal{V}_S, \mathcal{V}_D\}$, where $\mathcal{V}_S$ denotes the set of ingress nodes with size $|\mathcal{V}_S| = N_S$, and $\mathcal{V}_D$ denotes the set of egress nodes with size $|\mathcal{V}_D| = N_D$, and $\mathcal{E}$ denotes the set of transmission links from $\mathcal{V}_S$ to $\mathcal{V}_D$. Fig. 2-2(a) visualizes the single-hop structure. Examples of single-hop networks include switched networks as Fig. 2-2(b) and server farms as Fig. 2-2(c). The single-hop structure is the basic network unit that constitutes many data center networks [1, 17].

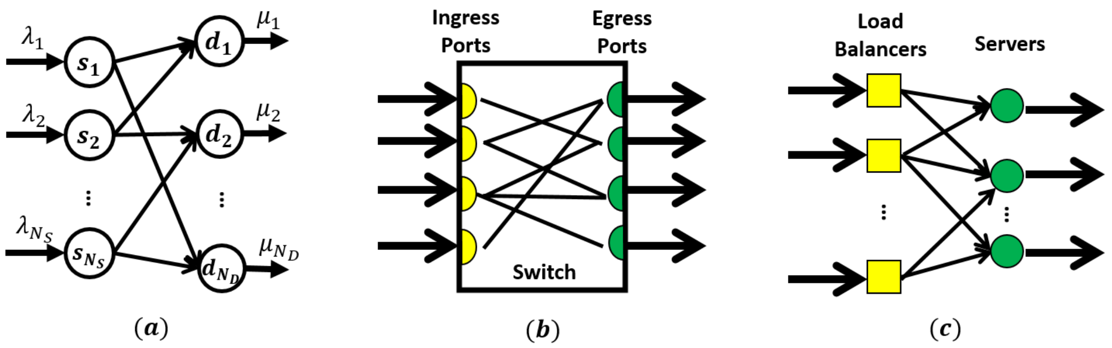


Figure 2-2: (a) A single-hop network structure; (b) A switched network with ingress and egress ports; (c) A server farm with load balancers as ingress and servers as egress.

We denote the ith ingress node by s_i and the jth egress node by d_j in single-hop networks. We consider that a packet injected into an ingress node can be dispatched to any connected egress node and depart. We denote the packet arrival rate at ingress node s_i by λ_i, which represents the average number of packets injected into node s_i in a time unit. We use $\boldsymbol{\lambda} := \{\lambda_i\}_{i=1}^{N}$ to represent the packet arrival rate vector which we assume static (i.e., time-invariant). We further assume to be that at each node, packets in the buffer follow the first-come-first-serve service which is common in real network infrastructures [48]. We denote the queue length in node k at time t by $q_k(t)$. We denote the packet transmission rate on link (s_i, d_j) at time t by $g_{s_i d_j}(t)$, which represents the number of packets transmitted over (s_i, d_j) at time t. Each link (s_i, d_j) is associated with a capacity value $c_{s_id_j}$, which is the maximum transmission rate, i.e., $0 \leq g_{s_id_j}(t) \leq c_{s_id_j}, \ \forall t, \ \forall (s_i, d_j) \in \mathcal{E}$. Note that trivially $g_{s_id_j}(t) \equiv 0$ for any $(s_i, d_j) \notin \mathcal{E}$, and $g_{s_id_j}(t) = 0$ when $q_{s_i}(t) = 0$ for any $(s_i, d_j) \in \mathcal{E}$, which means no packets will be transmitted through (s_i, d_j) when there is no queue backlog in node s_i. We use $\mathbf{g}(t) := \{g_{s_id_j}(t)\}_{(s_i,d_j)\in\mathcal{E}}$ to denote the transmission rate vector and $\mathbf{c} := \{c_{s_id_j}\}_{(s_i,d_j)\in\mathcal{E}}$ to denote the capacity vector. We consider that each egress node d_j serves packets in a work-conserving manner with its maximum service rate denoted by μ_j, whenever there exists queue backlog in the buffer. It is clear that work-conserving service at the egress nodes is a necessary condition for queueing delay minimization. Therefore we can merely focus on setting the transmission rate vector $\mathbf{g}(t)$ between the ingress and egress nodes to minimize queueing delay.

We apply a fluid queueing model to characterize the queueing dynamics: Packets are modeled as continuous traffic flows instead of discrete packet units, which means the queue length can be fractional. The fluid model is based on the flow conservation law, which states that the net increase of queue length equals to the difference between the number of new arrivals and departures at a node at any time, i.e.,

$$\begin{cases} \dot{q}_{s_i}(t) = \lambda_i - \sum_{d_j \in \mathcal{V}_D} g_{s_id_j}(t), \ \forall i = 1, \ldots, N_S \\ \dot{q}_{d_j}(t) = \sum_{s_i \in \mathcal{V}_S} g_{s_id_j}(t) - g_{d_j}(t), \ \forall j = 1, \ldots, N_D \end{cases} \tag{2.1}$$

where under the work-conserving mechanism at egress nodes, $g_{d_j}(t) := \mu_j$ if $q_{d_j}(t) > 0$. The dynamics (4.1) provide a simplified framework for flow control analysis compared with the discrete queueing model [3]. Note that it is different from the fluid model defined in some prior works which captures the scaled limit of the queue backlog [4, 76, 77], an indicator for queue stability but not suited to study queueing delay.

Multi-stage networks

We extend the definitions to multi-stage networks. A multi-stage network contains multiple layers of network nodes, and transmission links connecting nodes at adjacent layers. A multi-stage network with L layers of nodes is composed of an ingress layer where packets are injected into the network, an egress layer where packets depart from the network, and $L-2$ middle layers between them. We index the layers in order where the ingress layer is layer 1 and the egress layer is layer L. We can view an L-layer network as a cascade of $L-1$ single-hop networks, where a packet at any node at layer l can be dispatched to any of its connected node at layer $l+1$, and finally departs at some node at the egress layer. Each packet will traverse one node in each layer, and all the traversed nodes form the path of this packet. Different packets may take different paths. Fig. 2-3 gives an example of a multi-stage network with $L = 4$. Multi-stage networks are the common structures in data center infrastructures like Fat-tree [17, 95], Clos [91, 93], and the direct-connect topology with spine blocks removed [14, 96].

We use the following notations in multi-stage networks. Denote the set of nodes at layer l by $\mathcal{V}_l$ with size $|\mathcal{V}_l| = N_l$, the i-th node at layer l by n_i^l, and the transmission rate and capacity of link (n_i^l, n_j^{l+1}) between layer l and $l+1$ by $g_{n_i^l, n_j^{l+1}}$ and $c_{n_i^l, n_j^{l+1}}$ respectively. The packet arrival rate to the ingress node n_i^1 is λ_i, and the maximum service rate at the egress node n_j^L is μ_j. Similarly, all the egress nodes operate in a work-conserving manner, and $g_{n_i^{l-1}, n_j^l}(t) = 0$ if $q_{n_i^{l-1}}(t) = 0$ and $g_{n_i^L}(t) = 0$ if $q_{n_i^L}(t) = 0$. We define the queueing dynamics in multi-stage networks in (2.2), which is an extension of (4.1) from 2 layers to L layers.

$$\begin{cases} \dot{q}_{n_i^1}(t) = \lambda_i - \sum\limits_{n_j^2 \in \mathcal{V}_2} g_{n_i^1, n_j^2}(t), \ \forall i = 1, \dots, N_1 \\ \dot{q}_{n_i^l}(t) = \sum\limits_{n_k^{l-1} \in \mathcal{V}_{l-1}} g_{n_k^{l-1}, n_i^l}(t) - \sum\limits_{n_j^{l+1} \in \mathcal{V}_{l+1}} g_{n_i^l, n_j^{l+1}}(t), \\ \qquad\qquad\qquad\qquad \forall i = 1, \dots, N_l, \ \forall l = 2, \dots, L-1 \\ \dot{q}_{n_i^L}(t) = \sum\limits_{n_k^{L-1} \in \mathcal{V}_{L-1}} g_{n_k^{L-1}, n_i^L}(t) - g_{n_i^L}(t), \ \forall i = 1, \dots, N_L \end{cases} \tag{2.2}$$

In this work, we start from the special case of queueing dynamics (4.1) and (2.2) under *static* transmission policies where the transmission rates $g_{ij}(t)$ of each link (i, j) at different times t are a constant value g_{ij} when $q_i(t) > 0$. We demonstrate below

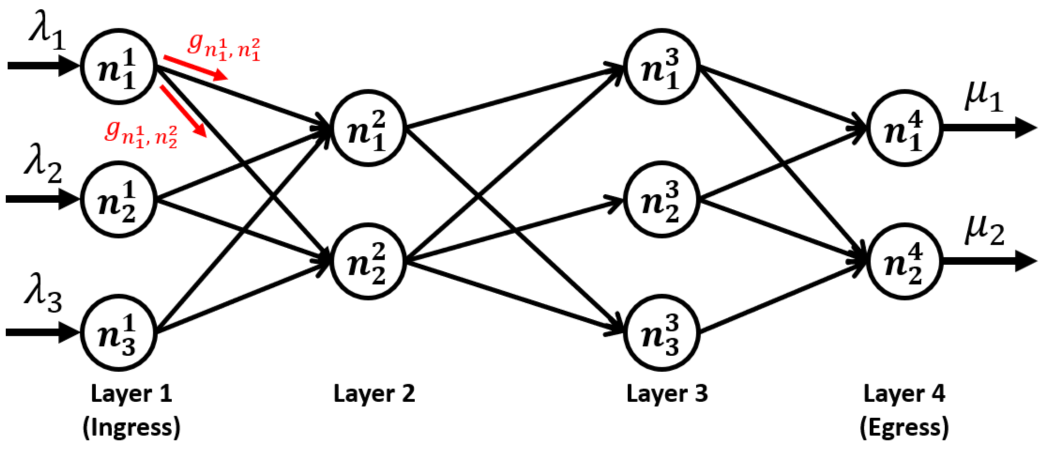


Figure 2-3: An example of a 4-layer multi-stage network

that the study over static policies facilitates the characterization of queueing delay of packets and the policy design for delay minimization, and the results inspire the *dynamic* policy design based on real-time queue backlog information instead of packet arrival rates at the ingress layer.

Remark: The equations in (4.1) and (2.2) give the general formulation of queueing dynamics without restrictions on packet routing, where the packets at a node can be transmitted to any of its connected nodes at the next layer. We can add routing constraints for example forcing $g_{ij}(t) \equiv 0$ which means packets at node i cannot be dispatched to j. We derive the policies that minimize queueing delay under the unrestricted dynamics and demonstrate that they still hold with routing restrictions, with details in Section 2.6.1.

2.2.2 Network Overload

We say that a network is overloaded if there is no transmission policy that guarantees bounded queueing backlog over all the node buffers in the network.

Definition 2.1. *A network is overloaded if there is no transmission policy* $\{\mathbf{g}(t)\}_{t\geq 0}$ *that can guarantee* $\lim_{t\to\infty} q_i(t) < \infty, \ \forall i \in \mathcal{V}$.

Definition 2.1 requires that no transmission policy can stabilize the network, which can be interpreted as the packet arrival rate vector $\boldsymbol{\lambda}$ beyond the network capacity region [97]. We can purely focus on overloaded networks since if $\boldsymbol{\lambda}$ is interior to the capacity region, there must exist a static policy $\mathbf{g}$ which guarantees that the total egress link rates of any node is greater than its total ingress traffic rate. Then it is trivial to apply this policy so that the queueing delay is zero under the deterministic fluid queueing model.

Under static transmission policies, we can derive more explicit conditions for a single-hop and a multi-stage network being overloaded in Definition 2.2 and 2.3 respectively.

Definition 2.2. *An $N_S \times N_D$ single-hop network under static policies is overloaded if there is no transmission rate vector $\mathbf{g}$ such that $g_{ij} \in [0, c_{ij}],\ \forall (i,j) \in \mathcal{E}$ and*

$$\begin{cases} \sum_{d_j:(s_i,d_j)\in\mathcal{E}} g_{s_i d_j} \geq \lambda_i,\ \forall i = 1,\ldots,N_S \\ \sum_{s_i:(s_i,d_j)\in\mathcal{E}} g_{s_i d_j} \leq \mu_j,\ \forall j = 1,\ldots,N_D \end{cases} \tag{2.3}$$

Definition 2.3. *An L-layer network under static policies is overloaded if there is no transmission rate vector $\mathbf{g}$ such that $g_{ij} \in [0, c_{ij}],\ \forall (i,j) \in \mathcal{E}$ and*

$$\begin{cases} \sum_{n_j^2\in\mathcal{V}_2} g_{n_i^1,n_j^2} \geq \lambda_i,\ \forall i = 1,\ldots,N_1 \\ \sum_{n_k^{l-1}\in\mathcal{V}_{l-1}} g_{n_k^{l-1},n_i^l} \leq \sum_{n_j^{l+1}\in\mathcal{V}_{l+1}} g_{n_i^l,n_j^{l+1}}, \quad \forall i = 1,\ldots,N_l,\ \forall l = 2,\ldots,L-1 \\ \sum_{n_k^{L-1}\in\mathcal{V}_{L-1}} g_{n_k^{L-1},n_i^L} \leq \mu_i,\ \forall i = 1,\ldots,N_L \end{cases} \tag{2.4}$$

2.2.3 Queueing Delay Characterization

We characterize the queueing delay of packets in overloaded networks under static transmission policies. The queueing delay dominates other network delays including preprocessing delay, transmission delay, and propagation delay in overloaded network, since the overload leads to severe increase of queue backlog in node buffers. Moreover, the other network delays are independent of the packet arrival rates and transmission policies. Therefore, we ignore the other delays in our analysis, and the delay only represents the queueing delay below.

We derive the explicit form of the total queueing delay of a packet that arrives at an ingress node. We first consider a 2-node network with a single link in Fig. 2-4 to explain the derivation. Consider the shaded packet at the tail of node 1. We assume it arrives at node 1 at time t. The queueing delay of this packet at node 1 is $q_1(t)/g_{12}$, as the shaded packet has to wait for all of the packets ahead of it to be served. The packet departs from node 1 and arrives at node 2 at time $t' := t + q_1(t)/g_{12}$, and thus its queueing delay at node 2 is $q_2(t')/\mu$. Therefore the total queueing delay for this packet is

$$\begin{aligned}\frac{q_1(t)}{g_{12}} + \frac{q_2(t')}{\mu} &= \frac{q_1(t)}{g_{12}} + \max\left\{\frac{q_2(t) + \frac{q_1(t)}{g_{12}}(g_{12} - \mu)}{\mu}, 0\right\} \\ &= \max\left\{\frac{q_1(t) + q_2(t)}{\mu}, \frac{q_1(t)}{g_{12}}\right\}\end{aligned} \tag{2.5}$$

where the queue growth rate at node 2 is $g_{12} - \mu$, and thus $q_2(t')$ is equal to $q_2(t)$ plus the total growth of packets in the buffer over time length $q_1(t)/g_{12}$. The max term in the second line is to take into account that $q_2(t')$ may reach 0 when $g_{12} < \mu$.

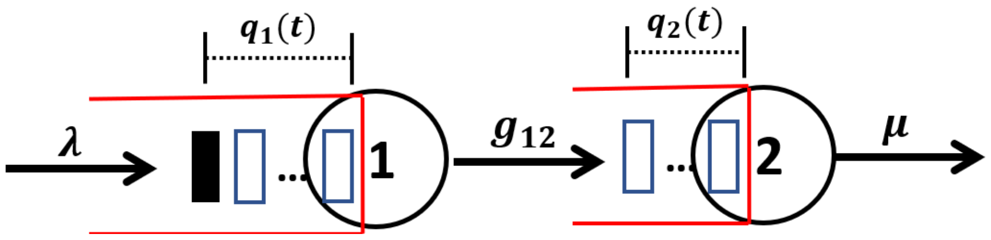


Figure 2-4: An example of queueing delay characterization of a packet passing two nodes

We extend the derivation in (2.5) to an $N_S \times N_D$ single-hop network. We denote by $D_{s_i d_j}(t)$ the queueing delay of a packet that arrives at the ingress node s_i at time t and departs at the egress node d_j. The queueing delay of this packet at s_i is $q_{s_i}(t)/\sum_{d_k:(s_i,d_k)\in\mathcal{E}} g_{s_i d_k}$ where $\sum_{d_k:(s_i,d_k)\in\mathcal{E}} g_{s_i d_k}$ is the sum of the egress link rates over all the links starting from s_i. Suppose that this packet is dispatched to d_j. The time it arrives at d_j is $t' = t + q_{s_i}(t)/\sum_{d_k:(s_i,d_k)\in\mathcal{E}} g_{s_i d_k}$, and its queueing delay at d_j is $q_{d_j}(t')/\mu_j$. We consider the case where the static transmission rate vector $\mathbf{g}$ guarantees $q_{d_j}(t) > 0$, $\forall t$, i.e., node d_j keeps serving with rate μ_j. In this case, we can express the total delay of this packet as

$$\begin{aligned} D_{s_i d_j}(t) &= \frac{q_{s_i}(t)}{\sum_{d_k} g_{s_i d_k}} + \frac{q_{d_j}(t')}{\mu_j} \\ &= \frac{q_{s_i}(t)}{\sum_{d_j} g_{s_i d_j}} + \frac{1}{\mu_j}\left(q_{d_j}(t) + \frac{q_{s_i}(t)}{\sum_{d_k} g_{s_i d_k}}\left(\sum\nolimits_{s_k} g_{s_k d_j} - \mu_j\right)\right) \\ &= \frac{1}{\mu_j}\left(q_{d_j}(t) + \frac{\sum_{s_k} g_{s_k d_j}}{\sum_{d_k} g_{s_i d_k}} q_{s_i}(t)\right) \end{aligned} \tag{2.6}$$

where the queueing delay of a single packet that arrives at the network at any time t can be expressed by a linear combination of the queue length at time t at the ingress and egress nodes that this packet traverses. We can generalize (2.6) to multi-stage networks with L layers to characterize the total queueing delay of packets taking any path p, denoted by $D_p(t)$, as a linear combination of the queue length at time t at all the nodes on the path. We show in later sections that the explicit forms like (2.5) and (2.6) facilitate the derivation of static policies $\mathbf{g}$ that minimize queueing delay.

2.2.4 Problem Formulation

We define two queueing delay metrics that we try to minimize in this work: (i) the average delay $\bar{D}_{\text{avg}}$ (ii) the maximum ingress delay $\bar{D}_{\text{max}}$, whose definitions are introduced later. At a high level, $\bar{D}_{\text{avg}}$ reflects the overall delay performance of all the arrived packets, which in practice is relevant to data centers where the overall performance is important [7, 14, 92]; $\bar{D}_{\text{max}}$ represents the largest delay of the packets that arrive at different ingress nodes, which in practice is related to the fairness and flow completion time of tasks parallelized to different ingress nodes [96, 98, 99]. We focus on minimizing both metrics for packets that arrive to the network in some *bounded* time interval $[t_0, t_0 + T]$ where t_0 is the initial timestamp and $T < \infty$ is the time duration, given that network overload is a temporary event in practice.

We give the formal definitions of $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$. We first consider an $N_S \times N_D$ single-hop network. Denote the average queueing delay of packets that arrive at the ingress node s_i within $[t_0, t_0 + T]$ by $\bar{D}_i$, which is

$$\bar{D}_i = \frac{1}{T} \int_{t_0}^{t_0+T} \sum_{j=1}^{N_D} \frac{g_{s_i d_j}}{\sum_{k=1}^{N_D} g_{s_i d_k}} D_{s_i d_j}(t) dt, \ \forall i = 1, \ldots, N_S, \tag{2.7}$$

where $D_{s_i d_j}(t)$ is as in (2.6). Note that (2.7) contains two layers of averaging: (i) averaging over different arrival times t within $[t_0, t_0 + T]$, which is an unweighted integral; (ii) averaging over packets sent to different egress nodes, which is weighted by $g_{s_i d_j} / \sum_{k=1}^{N_D} g_{s_i d_k}$, i.e., the portion of packets that arrive at s_i and will depart from d_j. With (2.7), we formulate the two delay metrics to be optimized as

$$\bar{D}_{\text{avg}} = \sum_{i=1}^{N} \frac{\lambda_i}{\sum_{j=1}^{N} \lambda_j} \bar{D}_i, \tag{2.8}$$

$$\bar{D}_{\max} = \max_{i=1,\ldots,N} \bar{D}_i. \tag{2.9}$$

The $\bar{D}_{\text{avg}}$ in (2.8) introduces an additional layer of averaging, weighted by the ratio $\lambda_i T / \left(\sum_{j=1}^{N} \lambda_j T \right) = \lambda_i / \left(\sum_{j=1}^{N} \lambda_j \right)$ that is the portion of the packets that arrive at the ingress node s_i within $[t_0, t_0 + T]$. The $\bar{D}_{\max}$ in (2.9) takes the maximum over all $\bar{D}_i$'s, which represents the highest average queueing delay of packets among all the ingress nodes.

We extend the definitions of both metrics to general multi-stage networks. We solely need to modify (2.7) into

$$\bar{D}_i = \frac{1}{T} \int_{t_0}^{t_0+T} \sum_{p:\ p[0]=n_i^1} w_p D_p(t) dt, \ \forall i = 1, \ldots, N_S$$

where $\{p : p[0] = n_i^1\}$ contains all the paths p that start from the ingress node n_i^1, and w_p represents the proportion of packets taking the path p among all packets starting from n_i^1. We show in the proof of the results in Section 2.3.2 that w_p can be expressed as a function of the transmission rate vector $\mathbf{g}$. The formulation of $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$ are the same as (2.8) and (2.9) respectively.

2.3 Static Min-Delay Policy Design

In this section, we develop the static policies that minimize $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$. We start from $N \times 1$ single-hop networks with $N_S = N$ ingress nodes and $N_D = 1$ egress node. Examples include a single server receiving requests from multiple sources, and packets that arrive from multiple upstream links sharing a single port of a downstream switch between two stages in a data center [1]. We prove a sufficient and necessary condition on link rates that minimize both the delay metrics. The conditions require that the link rates from all the ingress nodes to the egress node are in the same proportion to their corresponding packet arrival rates. We term any policy under which the condition holds as a *rate-proportional* policy. We unveil a counter-intuitive corollary that using larger link rates may increase delay. We then demonstrate that the rate-proportional policy can be extended to general single-hop networks and multi-stage networks, and guarantees minimum $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$.

2.3.1 $N \times 1$ Networks

Consider an $N \times 1$ network as shown in Fig. 2-5. We identify the transmission rate vector $\mathbf{g} := \{g_i\}_{i=1}^{N}$ that minimizes $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$, where we abbreviate link rate $g_{s_i,d}$ as g_i. We first derive the policies that minimize both delay metrics given unlimited link capacities, and then discuss how the capacities affect the result.

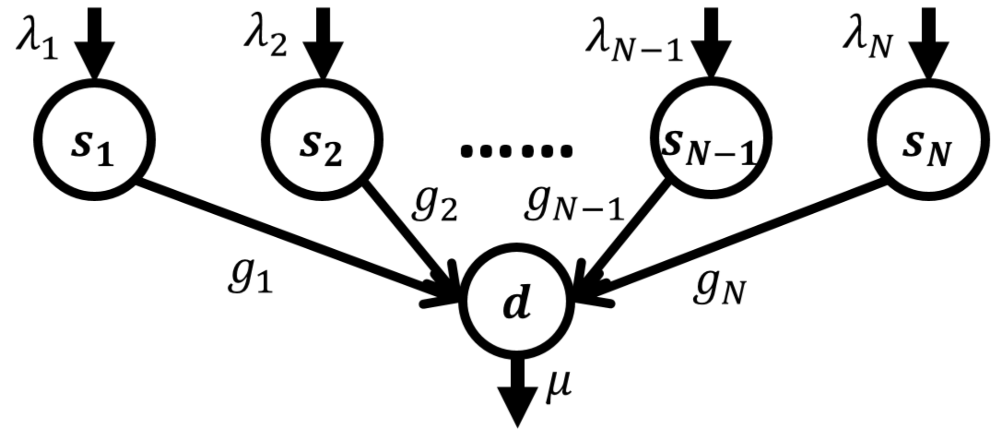


Figure 2-5: An example of an $N \times 1$ single-hop network

We identify a sufficient and necessary condition on $\mathbf{g}$ that minimizes $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$ in Theorem 2.1. We give the detailed proof under $N = 2$ for brevity, which can be extended to general N. We also derive the result under zero initial queue length for brevity.

Theorem 2.1. *Given an $N \times 1$ single-hop network with unlimited link capacity. For $\forall T > 0$, the set of $\mathbf{g} = \{g_i\}_{i=1}^N$ that minimizes $\bar{D}_{avg}$ and $\bar{D}_{max}$ of the packets that arrive within $[t_0, t_0 + T]$ where $\mathbf{q}(t_0) = \mathbf{0}$ is*

$$\left\{\left(\sum_{i=1}^{N} g_i \geq \mu\right) \cap \left(\frac{g_1}{\lambda_1} = \cdots = \frac{g_N}{\lambda_N}\right)\right\} \cup \{g_i \geq \lambda_i,\ \forall i = 1, \ldots, N\}, \tag{2.10}$$

under which $\bar{D}_{avg} = \bar{D}_{max} = \frac{T}{2\mu} \max\{\sum_{i=1}^N \lambda_i - \mu, 0\}$.

Proof. Consider $N = 2$. The main idea of the proof is that we divide the feasible link rate region of $\mathbf{g} = (g_1, g_2)$, which is $[0, \infty) \times [0, \infty)$, into 4 sub-regions:

$$\begin{cases} \mathcal{R}_1 := \{\mathbf{g} \mid g_1 \in [0, \lambda_1], g_2 \in [0, \lambda_2]\} \\ \mathcal{R}_2 := \{\mathbf{g} \mid g_1 \in [\lambda_1, \infty), g_2 \in [\lambda_2, \infty)\} \\ \mathcal{R}_3 := \{\mathbf{g} \mid g_1 \in [\lambda_1, \infty), g_2 \in [0, \lambda_2]\} \\ \mathcal{R}_4 := \{\mathbf{g} \mid g_1 \in [0, \lambda_1], g_2 \in [\lambda_2, \infty)\} \end{cases} \tag{2.11}$$

and we identify the optimal $\mathbf{g}$'s restricted in each of these sub-regions, denoted by $\mathbf{g}^*_{(1)}, \mathbf{g}^*_{(2)}, \mathbf{g}^*_{(3)}, \mathbf{g}^*_{(4)}$ respectively. We show that each $\mathbf{g}^*_{(i)}$ leads to the same average queueing delay $\bar{D}_{\text{avg}} = \frac{T}{2\mu} \max\{(\lambda_1 + \lambda_2 - \mu), 0\}$ and the same maximum ingress delay $\bar{D}_{\text{max}} = \frac{T}{2\mu} \max\{(\lambda_1 + \lambda_2 - \mu), 0\}$.

We define $D_i(t)$ as the total queueing delay of a packet injected into s_i at time t.

According to (2.6), for $i = 1, 2$,

$$
\begin{aligned}
D_i(t) &= \frac{q_{s_i}(t)}{g_i} + \max\left\{ \frac{q_d(t) + \frac{q_{s_i}(t)}{g_i}(g_1 + g_2 - \mu)}{\mu}, 0 \right\} \\
&= \begin{cases} \frac{1}{\mu}\left(q_d(t) + \frac{q_{s_i}(t)}{g_i}(g_1 + g_2) \right), & g_1 + g_2 \geq \mu \\ \frac{q_{s_i}(t)}{g_i}, & g_1 + g_2 < \mu \end{cases}
\end{aligned}
$$

due to $\mathbf{q}(t_0) = \mathbf{0}$ which guarantees that when $g_1 + g_2 < \mu$, $q_d(t)$ will keep zero and thus the only queueing delay is at the ingress nodes. The average delay for packets that arrive to ingress node s_i within $[t_0, t_0 + T]$ is $\bar{D}_i = \frac{1}{T}\int_{t_0}^{t_0+T} D_i(t)dt,\ i = 1, 2,$ based on which we can formulate the delay metrics $\bar{D}_{\text{avg}}$ given by (2.8) and $\bar{D}_{\max}$ given by (2.9) as functions of $\mathbf{g}$.

Case 1: $\mathcal{R}_1 := \{\mathbf{g} \mid g_1 \in [0, \lambda_1], g_2 \in [0, \lambda_2]\}$

In $\mathcal{R}_1$, we first consider the case when $g_1 + g_2 \geq \mu$.

$$
\begin{aligned}
\bar{D}_i &:= \frac{1}{T}\int_{t_0}^{t_0+T} D_i(t)dt = \frac{1}{T}\int_{t_0}^{t_0+T} \frac{q_d(t) + \frac{q_{s_i}(t)}{g_i}(g_1 + g_2)}{\mu} dt \\
&= \frac{1}{T\mu}\int_{t_0}^{t_0+T} (t - t_0)\max\{g_1 + g_2 - \mu, 0\} + \frac{g_1 + g_2}{g_i}(t - t_0)\max\{\lambda_i - g_i, 0\}dt \\
&= \frac{T}{2\mu}\left(\lambda_1 \frac{g_1 + g_2}{g_i} - \mu \right),\ i = 1, 2
\end{aligned}
$$

Then according to (2.8) and (2.9),

$$
\begin{aligned}
\bar{D}_{\text{avg}} &= \frac{\lambda_1}{\lambda_1 + \lambda_2}\bar{D}_1 + \frac{\lambda_2}{\lambda_1 + \lambda_2}\bar{D}_2 \\
&= \frac{T}{2\mu}\left(\frac{\lambda_1}{\lambda_1 + \lambda_2}\left(\lambda_1 \frac{g_1 + g_2}{g_1} - \mu \right) + \frac{\lambda_2}{\lambda_1 + \lambda_2}\left(\lambda_2 \frac{g_1 + g_2}{g_2} - \mu \right) \right)
\end{aligned}
$$

and

$$
\bar{D}_{\max} = \max\{\bar{D}_1, \bar{D}_2\} = \frac{T}{2\mu}\left\{ \left(\lambda_1 \frac{g_1 + g_2}{g_1} - \mu \right), \left(\lambda_2 \frac{g_1 + g_2}{g_2} - \mu \right) \right\}
$$

For $\bar{D}_{\text{avg}}$, we can obtain by Cauchy-Schwartz inequality that the optimal solutions are all $\mathbf{g}$ that satisfy $g_1 + g_2 \geq \mu$, $\frac{g_1}{g_2} = \frac{\lambda_1}{\lambda_2}$ under which the average delay is $\bar{D}_{\text{avg}} = \frac{T}{2\mu}(\lambda_1 + \lambda_2 - \mu)$. For $\bar{D}_{\max}$, we can obtain that the set of $\mathbf{g}$'s that satisfy (2.10) achieves the minimum $\bar{D}_{\max} = \frac{T}{2\mu}(\lambda_1 + \lambda_2 - \mu)$.

We then consider the case when $g_1 + g_2 \leq \mu$. In this case there will be no queue backlog in the egress node, and thus

$$\bar{D}_i = \frac{1}{T}\int_{t_0}^{t_0+T} \frac{q_{s_i}(t)}{g_i}dt = \frac{\max\{\lambda_i - g_i, 0\}}{g_i T}\int_{t_0}^{t_0+T}(t-t_0)dt = \frac{T}{2}\frac{\lambda_i - g_i}{g_i},\ i = 1, 2.$$

Therefore

$$\begin{cases} \bar{D}_{\text{avg}} = \frac{T}{2(\lambda_1+\lambda_2)}\left(\frac{\lambda_1^2}{g_1} + \frac{\lambda_2^2}{g_2} - \lambda_1 - \lambda_2\right) \\ \bar{D}_{\max} = \frac{T}{2}\max\left\{\frac{\lambda_1 - g_1}{g_1}, \frac{\lambda_2 - g_2}{g_2}\right\} \end{cases}$$

Then under $g_1 + g_2 \leq \mu$, the optimal metric values are $\bar{D}_{\text{avg}} = \bar{D}_{\max} = \frac{T}{2\mu}(\lambda_1 + \lambda_2 - \mu)$, achieved only at $g_1 = \frac{\lambda_1}{\lambda_1+\lambda_2}\mu$, $g_2 = \frac{\lambda_2}{\lambda_1+\lambda_2}\mu$, which is on the boundary $g_1 + g_2 = \mu$.

Case 2: $\mathcal{R}_2 := \{\mathbf{g} \mid g_1 \in [\lambda_1, \infty), g_2 \in [\lambda_2, \infty)\}$

When $g_1 \geq \lambda_1$, $D_1(t) = \frac{q_{s_1}(t)}{g_1} + \frac{q_d\left(t + \frac{q_{s_1}(t)}{g_1}\right)}{\mu} = \frac{q_d(t)}{\mu}$ as $q_{s_1}(t)$ keeps 0 since $q_{s_1}(t_0) = 0$, which means the queueing delay only occurs at node d. Similar for $D_2(t)$. Since $\lambda_1 + \lambda_2 > \mu$, packets will accumulate at node d and at time t, and $q_d(t) = (\lambda_1 + \lambda_2 - \mu)t$. Thus

$$D_1(t) = D_2(t) = \frac{q_d(t)}{\mu} = \frac{\lambda_1 + \lambda_2 - \mu}{\mu}t.$$

and $D_{s_1} = \frac{1}{T}\int_{t_0}^{t_0+T} D_1(t)dt = \frac{T}{2\mu}(\lambda_1 + \lambda_2 - \mu) = D_{s_2}$, and hence $\bar{D}_{\text{avg}} = \bar{D}_{\max} = \frac{T}{2\mu}(\lambda_1 + \lambda_2 - \mu)$ for $\forall \mathbf{g} \in \mathcal{R}_2$.

Case 3: $\mathcal{R}_3 := \{\mathbf{g} \mid g_1 \in [\lambda_1, \infty), g_2 \in [0, \lambda_2]\}$

Based on the derivation in case 1 and 2 respectively, we have $D_{s_1} = \frac{T}{2\mu}(\lambda_1 + g_2 - \mu)$ as packets that arrive at s_1 only suffer from delay at d, and $D_{s_2} = \frac{T}{2\mu}\left(\frac{\lambda_1 + g_2}{g_2}\lambda_2 - \mu\right)$ where packets that arrive at s_2 suffer from delay at s_2 and d. We can verify easily that any optimal $\mathbf{g} \in \mathcal{R}_3$ that achieves minimum $\bar{D}_{\text{avg}} = \bar{D}_{\max} = \frac{T}{2\mu}(\lambda_1 + \lambda_2 - \mu)$ should be guaranteed that $g_2 = \lambda_2$ holds.

Case 4: $\mathcal{R}_4 := \{\mathbf{g} \mid g_1 \in [0, \lambda_1], g_2 \in [\lambda_2, \infty)\}$ Similar to case 3, where any optimal $\mathbf{g} \in \mathcal{R}_4$ satisfies $g_1 = \lambda_1$. $\square$

We term (2.10) as the *min-delay region* of the transmission rate vector $\mathbf{g}$, which consists of a line segment connecting two points $\left\{\frac{\lambda_i}{\sum_{j=1}^{N}\lambda_j}\mu\right\}_{i=1}^{N}$ and $\{\lambda_i\}_{i=1}^{N}$ in an N-dimensional space, and a polytope of $\mathbf{g}$'s where $g_i \geq \lambda_i,\ \forall i = 1,\ldots,N$. Theorem 2.1 demonstrates that to achieve minimum delay, if there exists one link (s_i, d) with transmission rate g_i higher than the packet arrival rate λ_i at s_i, then all the other links should be as well; if on the contrary $g_i \leq \lambda_i$ for some i, then we need to guarantee that the link rates should be in the same proportion to the packet arrival rates among all the ingress nodes in order to achieve minimum delay, i.e. $g_i/\lambda_i,\ \forall i = 1,\ldots,N$ are the same. We term any policy that satisfies this condition as a *rate-proportional* policy. An important implication of this result is that setting link rates in the same proportion to the packet arrival rates can achieve minimum delay as done by setting them greater than the packet arrival rates but with less total bandwidth required. We give an example when $N = 3$ in Fig. 2-6: Both using the rate-proportional policy with $g_i \leq \lambda_i$ as Fig. 2-6(a) and setting $g_i \geq \lambda_i$, $i = 1,2,3$ as Fig. 2-6(b) lead to minimum $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$, while the transmission rate vector in Fig. 2-6(c) is not in the min-delay region. Moreover, the min-delay regions for both $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$ in an $N \times 1$ single-hop network are the same, which demonstrates that we can simultaneously achieve minimum average delay and in the meantime balance the delay of packets injected into different ingress nodes.

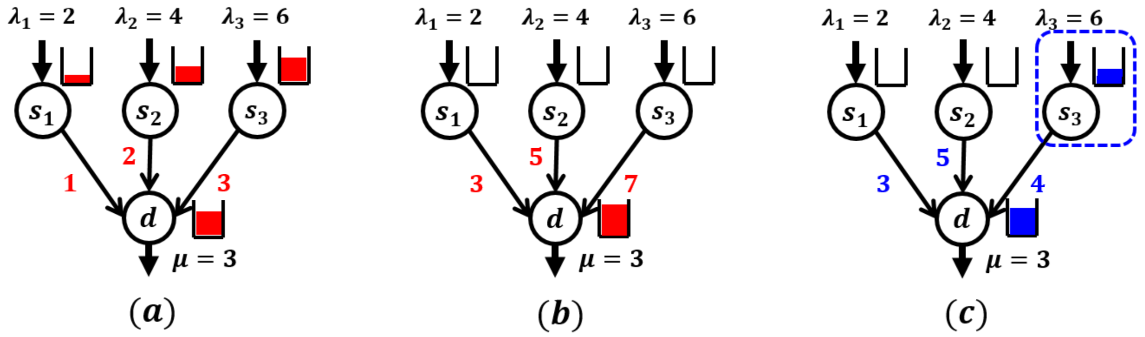
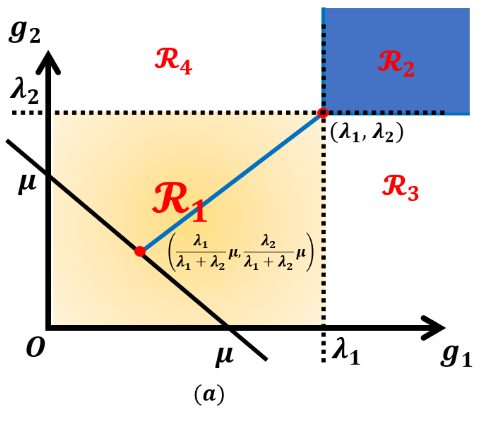


Figure 2-6: A 3x1 example of Theorem 2.1: (a) Setting $g_i/\lambda_i = 1/2, i = 1,2,3$ satisfies (2.10) and leads to minimum delay; (b) Setting $g_i \geq \lambda_i, i = 1,2,3$ satisfies (2.10) and leads to minimum delay, despite different queue growth rates compared with (a); (c) Setting $\mathbf{g} = \{3,5,4\}$ does not satisfy (2.10) and thus does not incur minimum $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$, although all the 3 links rates are greater than those in (a), primarily due to the higher congestion level at s_3.

We visualize the min-delay region when $N = 2$ for detailed explanation in Fig. 2-7(a), which is marked as the orange area: a line segment connecting the points $\left(\frac{\lambda_1}{\lambda_1+\lambda_2}\mu, \frac{\lambda_2}{\lambda_1+\lambda_2}\mu\right)$ and (λ_1, λ_2), and the polytope $\{\mathbf{g} \mid g_i \geq \lambda_i, i = 1, 2\}$. We mark the $\mathcal{R}_1$ to $\mathcal{R}_4$ in (2.11) in Fig. 2-7(a). We have the following insights: (i) Setting g_i no less than λ_i for both $i = 1, 2$ achieves minimum queueing delay, while further increasing g_1 and g_2 does not make a difference. This is because for any $\mathbf{g}$ that $g_i \geq \lambda_i$, the buffers of s_1 and s_2 are empty, hence all the queueing delay is at the egress node d bottlenecked by μ. (ii) We can achieve the minimum delay in $\mathcal{R}_1$ using lower transmission rates compared with those in $\mathcal{R}_2$ by consider the rate-proportional policy where $\mathbf{g} = \{g_1, g_2\}$ satisfies $g_1/g_2 = \lambda_1/\lambda_2$, and meanwhile maximum throughput is guaranteed, i.e., $g_1 + g_2 \geq \mu$. The minimum total link rate is μ at the intersection point $\left(\frac{\lambda_1}{\lambda_1+\lambda_2}\mu, \frac{\lambda_2}{\lambda_1+\lambda_2}\mu\right)$. (iii) It is not true that serving with higher rates leads to lower queueing delay. For example serving with transmission rates in $\mathcal{R}_3$ and $\mathcal{R}_4$ is inferior to controlling the transmission rates on the optimal line segment in $\mathcal{R}_1$. The counter-intuition is because packets from s_1 and s_2 share an egress node, where the imbalance between g_1 and g_2 leads to severe delay increase of packets that arrive to the ingress node with lower link rate downstream.



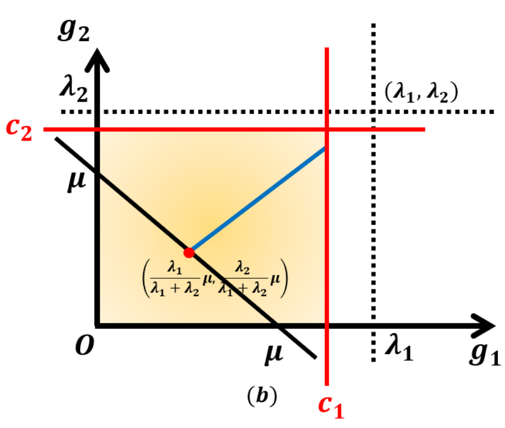

g_2
λ_2
(λ_1, λ_2)
c_2
μ
$\left(\frac{\lambda_1}{\lambda_1+\lambda_2}\mu, \frac{\lambda_2}{\lambda_1+\lambda_2}\mu\right)$
O
μ
λ_1
g_1
(b)
c_1

Figure 2-7: The min-delay region in a 2×1 single-hop network: (a) unlimited capacity; (b) limited capacity ($c_i \leq \lambda_i,\ i = 1, 2$)

We further extend Theorem 2.1 to the case of limited link capacities (i.e., capacity of link (s_i, d) is c_i). We can obtain directly that the min-delay region for limited capacity case is simply the intersection of (2.10) and $\{\mathbf{g} \mid g_i \leq c_i, i = 1, \ldots, N\}$ as limited capacity does not affect the proof, where c_i is the abbreviation of $c_{s_i d}$. We illustrate the min-delay region given that $\lambda_i > c_i,\ i = 1, 2$ in Fig. 2-7(b), which is a single solid orange line segment. We observe that serving both links with maximum rates equal to the link capacity does not lead to minimum delay in general cases, which validates the necessity of refined control of link rates based on (2.10). This result also gives insights on the demand-aware bandwidth allocation in data center networks [92,100,101], where allocating bandwidth in proportion to the demands from different ingress nodes leads to minimum delay when overload occurs. The minimum total bandwidth $c_1 + c_2$ required to achieve the global minimum delay as in the case with unlimited capacity is μ, where $c_i = \frac{\lambda_i}{\lambda_1+\lambda_2}\mu,\ i = 1, 2$.

Finally we discuss the impact of the initial queue length $\mathbf{q}(t_0)$ on the result. Consider $N = 2$ for example. We can follow the proof of Theorem 2.1 and obtain the min-delay region in $\mathcal{R}_1$ to be

$$\frac{g_1}{g_2} = \sqrt{\frac{\lambda_1(\lambda_1 + q_{s_1}(t_0)/T)}{\lambda_2(\lambda_2 + q_{s_2}(t_0)/T)}}, \tag{2.12}$$

under which the queue length at any ingress node will not reduce to zero. Note that (2.12) also follows the rate-proportional pattern with initial queue length and duration T included. For $\mathcal{R}_2$, $\mathcal{R}_3$ and $\mathcal{R}_4$, the derivation is of higher complexity as we need to analyze if the queue length at the ingress nodes will or will not change from non-zero to zero within $[t_0, t_0 + T]$, which involves at least two cases for each ingress node. In practice, the initial queue length $q_{s_1}(t_0)$ and $q_{s_2}(t_0)$ are generally very small before overload occurs, and we generally care about rate control for relatively long T instead of instantaneous overload. Therefore $q_{s_1}(t_0)/T$ is generally small and thus (2.12) is approximately $g_1/g_2 = \lambda_1/\lambda_2$, matching (2.10). For these reasons and the conciseness of proof, we neglect initial queue length and verify empirically in Section 6.4 that initial queue length does not affect the overall performance.

2.3.2 General Single-Hop and Multi-Stage Networks

We extend the rate-proportional policy shown in Theorem 2.1 to general single-hop networks and multi-stage networks and show that it is a sufficient condition for queueing delay minimization. The extended rate-proportional policy requires that all the nodes at the same layer share the same ratio between their total ingress rates and egress rates of packets.

$N_S \times N_D$ single-hop networks

We derive a sufficient condition on $\mathbf{g}$ to achieve minimum $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$ in Theorem 2.2 given unlimited capacity in $N_S \times N_D$ single-hop networks. We can extend the result to the case of limited capacity by adding the constraints $\mathbf{g}_{ij} \le c_{ij}, \forall (i,j) \in \mathcal{E}$ as discussed in $N \times 1$ networks.

Theorem 2.2. *Given an $N_S \times N_D$ single-hop network with unlimited link capacity. For $\forall T > 0$, a sufficient condition to globally minimize both $\bar{D}_{avg}$ and $\bar{D}_{max}$ of the packets that arrive within $[t_0, t_0 + T]$ where $\mathbf{q}(t_0) = \mathbf{0}$ is*

$$\begin{cases} \frac{\sum_{k=1}^{N_D} g_{s_i d_k}}{\sum_{k=1}^{N_D} g_{s_j d_k}} = \frac{\lambda_i}{\lambda_j}, \forall i,j = 1,\dots,N_S \\ \frac{\sum_{k=1}^{N_S} g_{s_k d_i}}{\sum_{k=1}^{N_S} g_{s_k d_j}} = \frac{\mu_i}{\mu_j}, \forall i,j = 1,\dots,N_D \\ \sum_{k=1}^{N_S} g_{s_k d_j} \ge \mu_j, \forall j = 1,\dots,N_D \end{cases} \tag{2.13}$$

under which $\bar{D}_{avg} = \bar{D}_{max} = \frac{T}{2\sum_{j=1}^{N_D}\mu_j} \max\{\sum_{i=1}^{N_S}\lambda_i - \sum_{j=1}^{N_D}\mu_j, 0\}$. Furthermore, (2.13) *is both sufficient and necessary for minimizing $\bar{D}_{avg}$ and $\bar{D}_{max}$ over the policies in $\{\mathbf{g} \mid \sum_{j=1}^{N_D} g_{s_i d_j} \le \lambda_i,\ \forall i = 1,\dots,N_S\}$.*

We defer the proof to Section 2.8. We explain the min-delay conditions (2.13): The first constraint requires that the total egress rates of different ingress nodes should be in the same proportion to their packet arrival rates $\{\lambda_i\}_{i=1}^{N_S}$; The second constraint requires that the total ingress rates of different egress nodes should be in the same proportion to their service rates $\{\mu_j\}_{j=1}^{N_D}$; The third constraint guarantees maximum throughput. Any transmission policy that satisfies these three conditions guarantees minimum $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$. Compared with (2.10) that requires the rate-proportional property at the ingress layer solely for $N \times 1$ networks, (2.13) requires the rate-proportional property at both the ingress and egress layers for general single-hop networks.

We further point out that (2.13) is also a necessary condition for delay minimization under limited transmission rates where $\mathbf{g} \in \{\mathbf{g} \mid \sum_{j=1}^{N_D} g_{s_i d_j} \leq \lambda_i,\ \forall i = 1, \ldots, N_S\}$, i.e., the egress rate of node s_i is less than the packet arrival rate λ_i. This result indicates that given limited link capacity, the rate-proportional policies are the only ones that minimize both delay metrics in overloaded single-hop networks.

We give a 2×2 example in Fig. 2-8, which demonstrate that multiple min-delay solutions can exist, as long as they satisfy (2.13) shown in Fig. 2-8(a) and (b), while higher link rates may even increase the delay shown in Fig. 2-8(c).

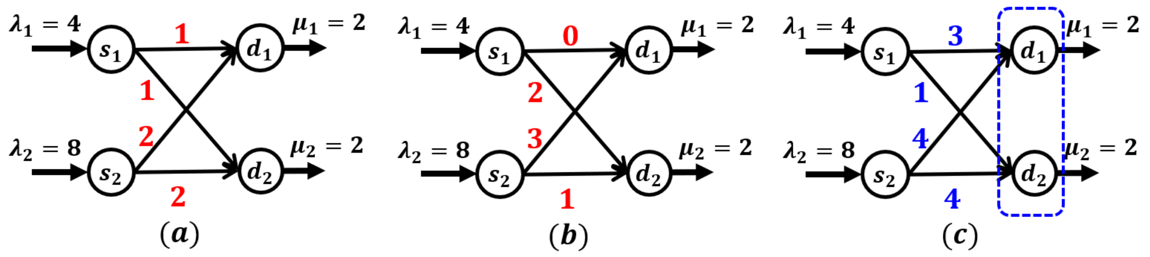


Figure 2-8: A 2x2 example of Theorem 2.2: (a) and (b) set $\mathbf{g}$ that satisfies $(g_{s_1d_1} + g_{s_1d_2})/(g_{s_2d_1} + g_{s_2d_2}) = \lambda_1/\lambda_2$ and $(g_{s_1d_1} + g_{s_2d_1})/(g_{s_1d_2} + g_{s_2d_2}) = \mu_1/\mu_2$ which lead to minimum $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$ simultaneously; Setting link rates as in (c) does not lead to minimum delay since $(g_{s_1d_1} + g_{s_2d_1})/(g_{s_1d_2} + g_{s_2d_2}) \neq \mu_1/\mu_2$, although the total link rates are higher than (a) and (b).

Multi-stage networks

We further extend the min-delay conditions for single-hop networks to multi-stage networks with L layers. We show in Theorem 2.3 that applying the rate-proportional policy design over each of the L layers leads to minimum $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$ as long as the maximum throughput is guaranteed.

Theorem 2.3. *Consider an L-layer network with unlimited link capacity. For $\forall T > 0$, a sufficient condition to globally minimize both $\bar{D}_{avg}$ and $\bar{D}_{max}$ of the packets that arrive within $[t_0, t_0 + T]$ when $\mathbf{q}(t_0) = \mathbf{0}$ is*

$$\begin{cases} \frac{\lambda_i}{\sum_{n_j^2 \in \mathcal{V}_2} g_{n_i^1, n_j^2}} = \gamma_1, \quad \forall n_i^1 \in \mathcal{V}_1 \\ \frac{\sum_{n_k^{l-1} \in \mathcal{V}_{l-1}} g_{n_k^{l-1}, n_i^l}}{\sum_{n_j^{l+1} \in \mathcal{V}_{l+1}} g_{n_i^l, n_j^{l+1}}} = \gamma_l, \quad \forall n_i^l \in \mathcal{V}_l, \ \forall l = 2, \ldots, L-1 \\ \frac{\sum_{n_k^{L-1} \in \mathcal{V}_{L-1}} g_{n_k^{L-1}, n_i^L}}{\mu_i} = \gamma_L, \quad \forall n_i^L \in \mathcal{V}_L \end{cases} \tag{2.14}$$

for some $\boldsymbol{\gamma} = \{\gamma_l\}_{l=1}^L \in \mathbb{R}_+^L$ and the maximum throughput is achieved, where $\bar{D}_{avg} = \bar{D}_{max} = \frac{T}{2} \max\left\{ \frac{\sum_{i=1}^{N_S} \lambda_i}{\sum_{j=1}^{N_D} \mu_j} - 1, 0 \right\}$.

We defer the proof to Section 2.8, whose primary idea is to apply $L - 1$ times the proof idea of Theorem 2.2 for single-hop networks. Theorem 2.3 shows that we guarantee minimum delay given that maximum throughput is achieved by setting link rates such that at each layer l, the ingress rates of all the nodes are in the same proportion to their egress rates. We denote the ratio between the ingress and egress rates at nodes at the l-th layer by γ_l. Note that we do not need to guarantee $\gamma_i = \gamma_j$ for different layers i and j. We give an example of $\mathbf{g}$ that satisfies (2.14) in Fig. 2-9. An important implication of the explicit sufficient condition is that they simplify the problem of link rate control for delay minimization to finding a feasible solution to (2.14) given a feasible $\boldsymbol{\gamma}$, which can be formulated as a linear programming (LP) problem. We defer the discussion of its feasibility analysis and its wider applications to Section 2.6.1. We point out that it is challenging to derive the necessary condition for multi-stage networks and we leave it to future work.

We leave a final remark to conclude this section. Note that the rate-proportional policy design is not equivalent to overload balancing in the buffers of nodes at the same layer [29, 90]. The former is to maintain identical ratios between ingress and egress rates, while the latter is to maintain identical differences between them. This means that the delay minimization and overload balancing among node buffers cannot be simultaneously achieved when the network is overloaded in general.

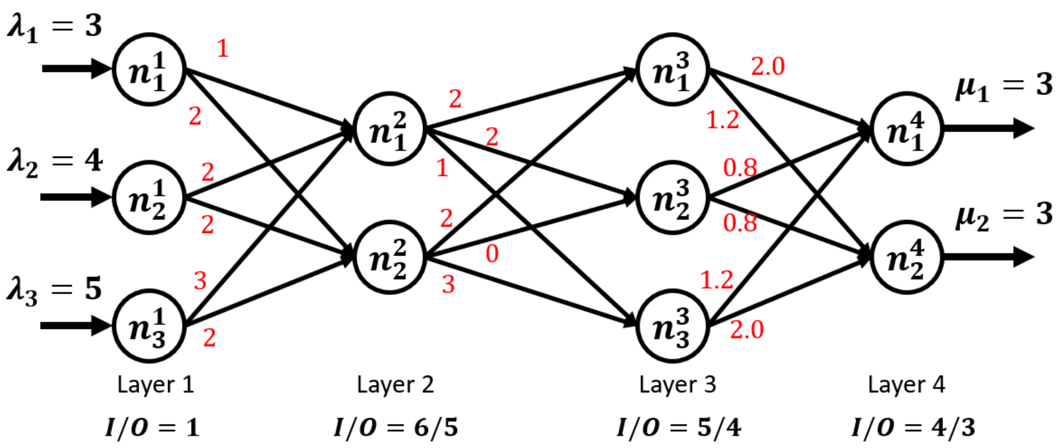


Figure 2-9: An example of link rate control in a 4-layer multi-stage network which minimizes $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$. The ingress and egress rates of nodes at layer 1 to 4 are 1, 6/5, 5/4, and 4/3 respectively.

2.4 Queue-based Min-delay Policy Design

In this section, we develop queue-based dynamic policies where link rates can be adjusted according to real-time queue backlog information in the network, based on the static min-delay policy design from Section 2.3. The motivation to study queue-based min-delay policies is that the static policies require the complete knowledge of network parameters, which in real networks may be difficult to estimate or unavailable, for example the packet arrival rate vector $\boldsymbol{\lambda}$ [102], while the real-time queue backlog $\mathbf{q}(t)$ is often accessible. We demonstrate that the *queue-proportional* policies can achieve minimum queueing delay asymptotically: setting the egress link rates of nodes at a layer in the same proportion to the queue backlog length in their buffers. We first introduce the min-delay queue-based policy design in $N \times 1$ networks, and then extend it to general single-hop and multi-stage networks.

2.4.1 $N \times 1$ Networks

We propose the queue-based min-delay policy for $N \times 1$ networks (as in Fig. 2-5) based on the static counterparts from Theorem 2.1. The min-delay condition (2.10) on static $\mathbf{g}$ which requires $g_i/g_j = \lambda_i/\lambda_j$, $\forall i \neq j$ implies that the dynamic rate control policy where $\mathbf{g}(t)$ satisfies $g_i(t)/g_j(t) = \dot{q}_{s_i}(t)/\dot{q}_{s_j}(t)$, $\forall i \neq j$ can minimize both $\bar{D}_{\text{avg}}$

and $\bar{D}_{\max}$, since

$$\frac{g_i(t)}{g_j(t)} = \frac{\dot{q}_{s_i}(t)}{\dot{q}_{s_j}(t)} = \frac{\lambda_i - g_i(t)}{\lambda_j - g_j(t)} \overset{(*)}{=} \frac{\lambda_i}{\lambda_j}$$

where $(*)$ holds since if for some $a, b, c, d \neq 0$ and $a + c, b + d \neq 0$, $a/b = c/d$, then $a/b = c/d = (a + c)/(b + d)$. This dynamic policy inspires the following queue-based policy without utilizing the information of $\boldsymbol{\lambda}$:

$$\frac{g_i(\mathbf{q}(t))}{g_j(\mathbf{q}(t))} = \frac{q_i(t)}{q_j(t)},\ \forall i \neq j, \quad \sum_{i=1}^{N} g_i(\mathbf{q}(t)) \geq \mu \tag{2.15}$$

where the link rates are set in the same proportion to the real-time queue length at the ingress nodes. We term any policy that follows (2.15) as a queue-proportional policy. We show in Theorem 2.4 that (2.15) achieves optimal $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$ with zero initial queue length at all nodes, and further in Theorem 2.5 that (2.15) asymptotically converges to the min-delay policy (2.10) given arbitrary initial queue vector $\mathbf{q}(t_0)$.

Theorem 2.4. *With $\mathbf{q}(t_0) = \mathbf{0}$, then the policy* (2.15) *achieves minimum $\bar{D}_{avg}$ and $\bar{D}_{max}$ as in* (2.10).

Proof. Initially at t_0, we take $\epsilon \to 0$ and have $\forall i \neq j$,

$$\begin{aligned}\frac{g_i(\mathbf{q}(t_0+\epsilon))}{g_j(\mathbf{q}(t_0+\epsilon))} &= \frac{q_{s_i}(t_0+\epsilon)}{q_{s_j}(t_0+\epsilon)} = \frac{\int_{t_0}^{t_0+\epsilon} \lambda_i - g_i(\mathbf{q}(s))ds}{\int_{t_0}^{t_0+\epsilon} \lambda_j - g_j(\mathbf{q}(s))ds} \\ &= \frac{\int_{t_0}^{t_0+\epsilon} \lambda_i ds - g_i(\mathbf{q}(t_0+\alpha_i\epsilon))}{\int_{t_0}^{t_0+\epsilon} \lambda_j ds - g_j(\mathbf{q}(t_0+\alpha_j\epsilon))} \to \frac{\int_{t_0}^{t_0+\epsilon} \lambda_i ds}{\int_{t_0}^{t_0+\epsilon} \lambda_j ds} = \frac{\lambda_i}{\lambda_j}\end{aligned}$$

where $\alpha_i, \alpha_j \in [0, 1]$. Then in the time interval $[t_0 + \epsilon, t_0 + 2\epsilon]$,

$$\begin{aligned}\frac{g_i(\mathbf{q}(t_0+2\epsilon))}{g_j(\mathbf{q}(t_0+2\epsilon))} &= \frac{q_{s_i}(t_0+2\epsilon)}{q_{s_j}(t_0+2\epsilon)} = \frac{q_{s_i}(t_0+\epsilon)+\epsilon\dot{q}_{s_i}}{q_{s_j}(t_0+\epsilon)+\epsilon\dot{q}_{s_j}} \\ &= \frac{q_{s_i}(t_0+\epsilon)+\epsilon(\lambda_i - g_i(\mathbf{q}(t_0+\epsilon)))}{q_{s_j}(t_0+\epsilon)+\epsilon(\lambda_j - g_j(\mathbf{q}(t_0+\epsilon)))} = \frac{\lambda_i}{\lambda_j}\end{aligned}.$$

Iteratively, we can obtain

$$\frac{g_i(\mathbf{q}(t))}{g_j(\mathbf{q}(t))} = \frac{q_{s_i}(t)}{q_{s_j}(t)} = \frac{\lambda_i}{\lambda_j},\ \forall t$$

which minimizes $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$ at any time t according to Theorem 2.1. $\square$

Theorem 2.5. *With arbitrary* $\mathbf{q}(t_0)$*,* (2.15) *converges to the state where* $\lim_{t\to\infty} g_i(\mathbf{q}(t))/g_j(\mathbf{q}(t)) = \lambda_i/\lambda_j$*,* $\forall i \neq j$ *which minimizes* $\bar{D}_{avg}$ *and* $\bar{D}_{max}$*.*

Proof. Under (2.15), for any $i \neq j$, when $t \to \infty$,

$$\left|\frac{q_{s_i}(t)}{q_{s_j}(t)} - \frac{\lambda_i}{\lambda_j}\right| = \left|\frac{q_{s_i}(t_0) + \int_{t_0}^{t} \lambda_i - g_i(\mathbf{q}(s))ds}{q_{s_j}(t_0) + \int_{t_0}^{t} \lambda_j - g_j(\mathbf{q}(s))ds} - \frac{\lambda_i}{\lambda_j}\right| \overset{\text{L'hos}}{=} \left|\frac{\lambda_i - g_i(\mathbf{q}(t))}{\lambda_j - g_j(\mathbf{q}(t))} - \frac{\lambda_i}{\lambda_j}\right|$$

and

$$\frac{g_i(\mathbf{q}(t))}{g_j(\mathbf{q}(t))} = \frac{q_{s_i}(t)}{q_{s_j}(t)} \overset{\text{L'hos}}{=} \frac{\dot{q}_{s_i}(t)}{\dot{q}_{s_j}(t)} = \frac{\lambda_i - g_i(\mathbf{q}(t))}{\lambda_j - g_j(\mathbf{q}(t))} \tag{2.16}$$

where L'hos means applying the L'hospital's rule[1]. We further derive based on (2.16) that $\lim_{t\to\infty} \frac{q_{s_i}(t)}{q_{s_j}(t)} = \lim_{t\to\infty} \frac{g_i(\mathbf{q}(t))}{g_j(\mathbf{q}(t))} = \frac{\lambda_i - \lim_{t\to\infty} g_i(\mathbf{q}(t))}{\lambda_j - \lim_{t\to\infty} g_j(\mathbf{q}(t))} = \frac{\lambda_i}{\lambda_j}$. This shows that the transmission rates asymptotically become proportional to the corresponding packet arrival rates at the ingress nodes, which reaches minimum $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$ based on Theorem 2.1. $\square$

Although not necessarily achieving minimum delay at any time given arbitrary $\mathbf{q}(t_0)$, the policy (2.15) keeps driving the queueing dynamics to the state under which delay is minimized. Intuitively, it drives $q_i(t)/q_j(t) \to \lambda_i/\lambda_j$: Suppose $q_i(t)/q_j(t) > \lambda_i/\lambda_j$, then $g_i(\mathbf{q}(t))/g_j(\mathbf{q}(t)) > \lambda_i/\lambda_j$ which drives down $q_i(t)/q_j(t)$, and when $q_i(t)/q_j(t) < \lambda_i/\lambda_j$, the policy increases $q_i(t)/q_j(t)$ closer to λ_i/λ_j.

2.4.2 General Single-Hop and Multi-Stage Networks

We extend the asymptotic min-delay policy (2.15) to general single-hop and multi-stage networks. We show that adjusting link rates so that the egress rates of nodes at the same layer in the same proportion to their queue backlogs at the current time leads to minimum delay asymptotically. Theorem 2.6 delivers a sufficient

[1]We solely need to consider the case where the queue backlogs in ingress nodes keep growing (i.e, $\lim_{t\to\infty} \int_{t_0}^{t} \lambda_i - g_i(\mathbf{q}(s))ds \to \infty,\ \forall i = 1, \ldots, N_S$), since otherwise the min-delay condition is trivial by serving with link rates higher than packet arrival rates at all the ingress nodes, as shown in Theorem 2.1.

condition that minimizes $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$ asymptotically in single-hop networks. Theorem 2.7 further generalizes the result to multi-stage networks.

Theorem 2.6. *Consider an $N_S \times N_D$ single-hop network. Any queue-based policy $\mathbf{g}(\mathbf{q}(t))$, $\forall t$ that satisfies*

$$\begin{cases} \frac{\sum_{k=1}^{N_D} g_{s_i d_k}(\mathbf{q}(t))}{\sum_{k=1}^{N_D} g_{s_j d_k}(\mathbf{q}(t))} = \frac{q_{s_i}(t)}{q_{s_j}(t)}, \ \forall i, j = 1, \ldots, N_S \\ \frac{\sum_{k=1}^{N_S} g_{s_k d_i}(\mathbf{q}(t))}{\sum_{k=1}^{N_S} g_{s_k d_j}(\mathbf{q}(t))} = \frac{\mu_i}{\mu_j}, \ \forall i, j = 1, \ldots, N_D \\ \sum_{k=1}^{N_S} g_{s_k d_j}(\mathbf{q}(t)) \geq \mu_j, \ \forall j = 1, \ldots, N_D \end{cases} \tag{2.17}$$

achieves asymptotically minimum $\bar{D}_{avg}$ and $\bar{D}_{max}$ as in (2.13) *with arbitrary initial queue backlog.*

Theorem 2.7. *Consider an L-layer multi-stage network. Any queue-based policy $\mathbf{g}(\mathbf{q}(t)), \forall t$ that satisfies*

$$\begin{cases} \frac{q_{n_i^l}(t)}{\sum_{n_j^{l+1} \in \mathcal{V}_{l+1}} g_{n_i^l, n_j^{l+1}}(\mathbf{q}(t))} = \gamma_l, \quad \forall n_i^l \in \mathcal{V}_l, \ \forall l = 1, \ldots, L-1 \\ \frac{\sum_{n_k^{L-1} \in \mathcal{V}_L} g_{n_k^{L-1}, n_i^L}(\mathbf{q}(t))}{\mu_i} = \gamma_L, \quad \forall n_i^L \in \mathcal{V}_L \end{cases} \tag{2.18}$$

for some $\boldsymbol{\gamma} = \{\gamma_l\}_{l=1}^L \in \mathbb{R}_+^L$ and guarantees maximum throughput can achieve asymptotic minimum $\bar{D}_{avg}$ and $\bar{D}_{max}$ with arbitrary initial queue backlog.

The proof idea of both theorems follows Theorem 2.5. Theorem 2.6 is a special case of Theorem 2.7 with $L = 2$. Both demonstrate the idea of min-delay link rate control by maintaining the egress rates of all the nodes in the same proportion to the current queue backlogs in these nodes at the same layer.

In summary, we determined that the format of the min-delay queue-based policy is similar to the static policies. The above analysis reduces the optimization problem in dynamical systems to finding feasible solutions to an explicit set of queue-proportional constraints. We show below in Section 2.6.1 that the explicit form facilitates the co-optimization of queueing delay together with other metrics.

2.5 Performance Evaluation

In this section, we evaluate the proposed min-delay policies in overloaded networks. We compare the performance of $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$ of our proposed methods with (i) the *Max-link-rate* policy where all links are activated with rates equal to their capacities, and (ii) the *Backpressure* policy that achieves optimal throughput and low latency [3], which serves packets over a link (i, j) with rate equal to its capacity if and only if node i has longer queue backlog than node j. We use the following abbreviations in the evaluation: OPT for the proposed min-delay policies, MAX for the max-link-rate policy, and BP for the backpressure policy.

We validate that our proposed methods achieve minimum $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$ in various network settings: (i) different topologies of both single-hop and multi-stage networks; (ii) different values of $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$; (iii) different capacities $\mathbf{c}$ (single-hop networks only). We consider multiple network instances with randomly sampled values of the above parameters, and measure the empirical cumulative distribution functions (CDFs) of $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$. We solely present the results for the queue-based policies, where the static policies result in similar performance. Moreover, packets are transmitted in discrete time intervals during simulation, and the results demonstrate that the min-delay property of our proposed policies based on the continuous fluid model holds under discrete transmission.

2.5.1 $N \times 1$ Networks

We evaluate the delay performance over 32×1 single-hop networks. We consider 500 different combinations of parameter settings sampled based on the following rules: (i) The arrival rate λ_i to each ingress node s_i is uniformly distributed in $[12, 20]$; (ii) The service rate of the shared egress node is $0.4 \times \sum_{i=1}^{32} \lambda_i$ so that the network is overloaded; (iii) Link capacities are uniformly distributed within $[20, 35]$ to represent the case of sufficient capacity where $\lambda_i \leq c_i$ for each node s_i, and $[5, 15]$ to represent the case of limited capacity where λ_i may exceed c_i. We round any rational number to an integer to characterize discrete packet transmission. We consider the initial queue length in each node to be a random integer within $[101, 300]$, and we consider the $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$ of packets that arrive within the first 200 time units that the network is overloaded.

Fig. 2-10 illustrates the CDF curves of both $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$ under sufficient capacity. The curves of the proposed min-delay policy and the max-link-rate policy highly overlap, which matches the result shown in Fig. 2-7(a). Their $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$ are lower than the backpressure policy: For $\bar{D}_{\text{avg}}$, the backpressure policy induces 5% higher delay on average and a maximum of 12% higher delay over the 500 tested samples; For $\bar{D}_{\text{max}}$, the backpressure induces 61% higher delay on average and a maximum of 159% higher delay over the 500 tested samples. Fig. 2-11 illustrates the results under limited capacity. A major contrast to the sufficient capacity case is the significantly poor delay performance of the max-link-rate policy, which echoes Theorem 2.1 and Fig. 2-7. We find that the $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$ of the max-link-rate policy are 18% and 123% higher than the proposed min-delay policy respectively on average.

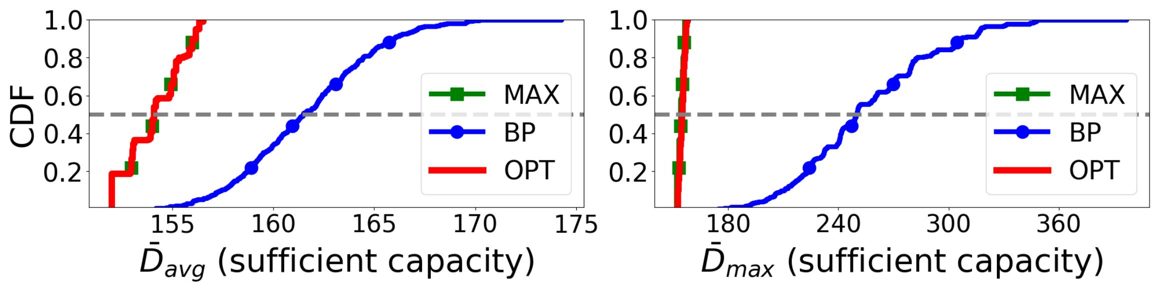


Figure 2-10: CDFs of $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$ in 32×1 single-hop networks with sufficient capacity (OPT and MAX are overlapped)

2.5.2 General Single-Hop Networks

We evaluate the delay performance over 32×16 single-hop networks. We consider 500 different combinations of parameter settings based on the following rules: (i) The arrival rate λ_i to each ingress node s_i is uniformly distributed in $[60, 100]$; (ii) The service rate μ_j of each egress node d_j is determined as $0.4 \times \alpha_j \times \sum_{i=1}^{32} \lambda_i$ where α_j is a randomly picked weight for d_j with $\sum_{j=1}^{16} \alpha_j = 1$. We consider sufficient link capacity

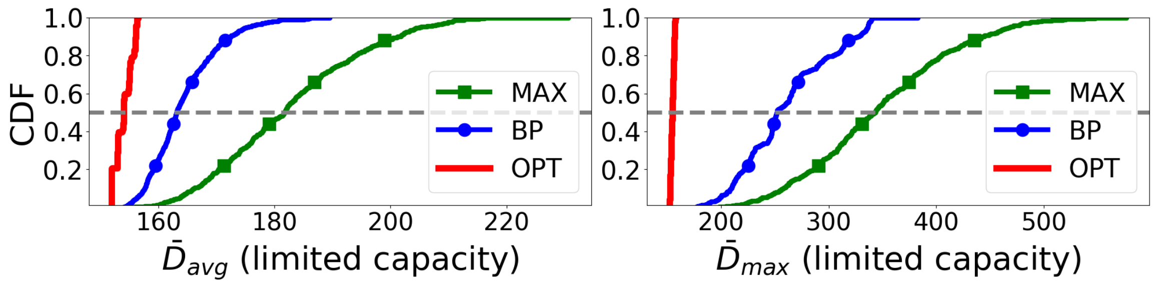


Figure 2-11: CDFs of $\bar{D}_{\mathrm{avg}}$ and $\bar{D}_{\mathrm{max}}$ in 32×1 single-hop networks with limited capacity

for each pair of ingress and egress nodes. We evaluate the $\bar{D}_{\mathrm{avg}}$ and $\bar{D}_{\mathrm{max}}$ of packets that arrive within the first 50 time units.

Fig. 2-12 illustrates the CDF curves of both $\bar{D}_{\mathrm{avg}}$ and $\bar{D}_{\mathrm{max}}$. Table 2.1 summarizes the mean and maximum ratio, in terms of both $\bar{D}_{\mathrm{avg}}$ and $\bar{D}_{\mathrm{max}}$, between the backpressure policy, or max-link-rate policy and the min-delay policy. Results show significant reduction of $\bar{D}_{\mathrm{avg}}$ and $\bar{D}_{\mathrm{max}}$ under policies that follow (2.17): On average the backpressure policy incurs 32% higher $\bar{D}_{\mathrm{avg}}$ and 111% higher $\bar{D}_{\mathrm{max}}$, while these metrics for the max-link-rate policy are 258% and 1166% higher than the min-delay policy. We also observe that the worst case for both the backpressure and the max-link-rate policy leads to more than 10× delay compared with the min-delay policies in terms of both $\bar{D}_{\mathrm{avg}}$ and $\bar{D}_{\mathrm{max}}$. We further find that unlike $N \times 1$ networks, the max-link-rate policy no longer minimizes the delay in general single-hop networks in spite of sufficient capacity. Moreover, we demonstrate that the $\bar{D}_{\mathrm{avg}}$ and $\bar{D}_{\mathrm{max}}$ of the proposed min-delay policy keep stable among all the tested cases. This matches Theorem 2.6 where the minimum $\bar{D}_{\mathrm{avg}}$ and $\bar{D}_{\mathrm{max}}$ depend only on the ratio $\sum_{i=1}^{N_S} \lambda_i / \sum_{j=1}^{N_D} \mu_j$ which are the same among all the tested samples[2].

2.5.3 General Multi-Stage Networks

We further evaluate the performance of the tested policies over general multi-stage networks with L layers. We consider 7 multi-stage network structures listed in Table

[2]There are mild fluctuations over different tested samples due to rounding the real numbers to integers.

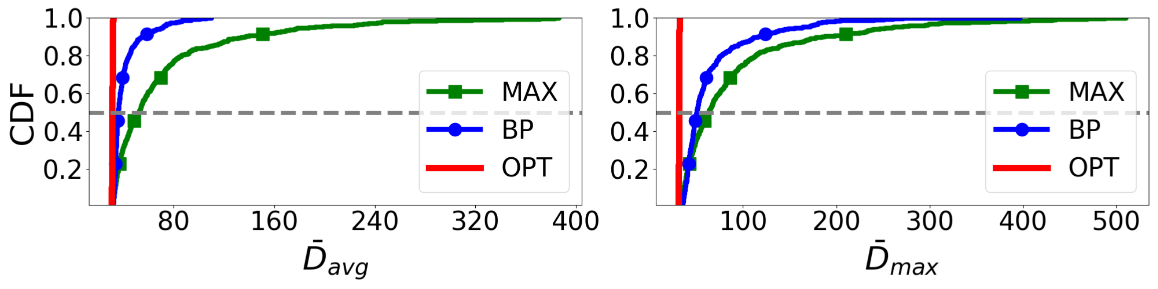


Figure 2-12: CDFs of $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$ in 32×16 single-hop networks

	$\bar{D}_{\text{avg}}$		$\bar{D}_{\text{max}}$	
	BP/OPT	MAX/OPT	BP/OPT	MAX/OPT
Mean	1.32	2.32	2.11	2.93
Max	3.58	12.38	12.66	16.29

Table 2.1: Mean and maximum ratios between the two policies for comparison and the min-delay policy in terms of $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$ in 32×16 single-hop networks

2.2, including different numbers of layers and fan-in-fan-out topologies. We consider full connection between adjacent layers. We consider 500 different combinations of parameter settings based on the following rules: (i) The arrival rate λ_i to each node n_i^1 at the ingress layer is uniformly distributed in $[30, 50]$; (ii) The service rate μ_j of each node n_j^L at the egress layer is determined as $0.4 \times \alpha_j \times \sum_{i=1}^{|\mathcal{V}_1|} \lambda_i$ where α_j is a randomly picked weight for n_j^L with $\sum_{j=1}^{|\mathcal{V}_L|} \alpha_j = 1$. We consider sufficient capacity over each pair of nodes at adjacent layers. We evaluate the $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$ of packets that arrive within the first 50 time units.

Table 2.2 lists the evaluation results over all the tested multi-stage networks. Columns 1 and 2 show the mean and maximum ratio between the backpressure policy, or the max-link-rate policy and the proposed min-delay policy with respect to $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$ respectively. Column 3 shows $\bar{D}_{\text{max}}/\bar{D}_{\text{avg}}$ under the backpressure policy and the max-link-rate policy, which reflects the level of delay fairness of packets injected into different ingress nodes. Note that $\bar{D}_{\text{max}}/\bar{D}_{\text{avg}} = 1$ under the min-delay policy as given in Theorem 2.3. The key takeaway is that both the backpressure and max-link-rate policies lead to at least $1.3\times$ $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$ of the min-delay policy on average in all 7 tested structures, showing the delay reduction of the proposed policy in general multi-stage networks. Moreover the mean imbalance ratios $\bar{D}_{\text{max}}/\bar{D}_{\text{avg}}$ for both the backpressure and max-link-rate policy are up to 20% higher than the optimal case over the tested instances. We visualize the CDFs of the $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$ of two multi-stage networks: $16 \times 12 \times 8 \times 6$ in Fig. 2-13 and $15 \times 12 \times 9 \times 12 \times 15$ in Fig. 2-14 for detailed characterization of their distributions. We can observe the backpressure and the max-link-rate policy have different performance in these two topologies, while our proposed policy achieves minimum delay in all the tested instances.

Topology	**Policy**	Ratio ($\bar{D}_{\text{avg}}$)		Ratio ($\bar{D}_{\text{max}}$)		$\bar{D}_{\text{max}}/\bar{D}_{\text{avg}}$	
		Mean	Max	Mean	Max	Mean	Max
16x12x16	BP	1.46	2.76	1.71	3.17	1.17	1.63
	MAX	1.46	2.16	1.51	2.23	1.03	1.11
12x16x12	BP	1.77	3.06	2.14	4.55	1.20	1.65
	MAX	1.47	2.29	1.52	2.43	1.03	1.08
16x12x8x6	BP	1.33	1.85	1.42	2.02	1.07	1.21
	MAX	1.50	3.53	1.53	3.62	1.02	1.07
6x8x12x16	BP	1.31	2.41	1.34	2.63	1.02	1.11
	MAX	1.35	2.13	1.38	2.16	1.02	1.06
15x12x9x12x15	BP	1.34	1.93	1.37	2.11	1.02	1.15
	MAX	1.49	2.28	1.52	2.34	1.02	1.07
9x12x15x12x9	BP	1.53	2.70	1.56	2.71	1.02	1.09
	MAX	1.45	2.74	1.47	2.76	1.01	1.07
12x12x12x12x12	BP	1.41	2.49	1.44	2.72	1.02	1.09
	MAX	1.51	2.64	1.54	2.70	1.02	1.07

Table 2.2: Mean and maximum ratios between each of the two tested policies for comparison and the proposed min-delay policy in terms of $\bar{D}_{\text{avg}}$ (Column 1) and $\bar{D}_{\text{max}}$ (Column 2), and the ratios $\bar{D}_{\text{max}}/\bar{D}_{\text{avg}}$ of the tested policies themselves reflecting the delay imbalance of packets injected into different ingress nodes (Column 3), over 7 multi-stage topologies.

2.6 Extensions and Discussions

In this section, we extend the main results on min-delay policy design to a wider range of scenarios both in practice and in theory. For practice, we demonstrate that the

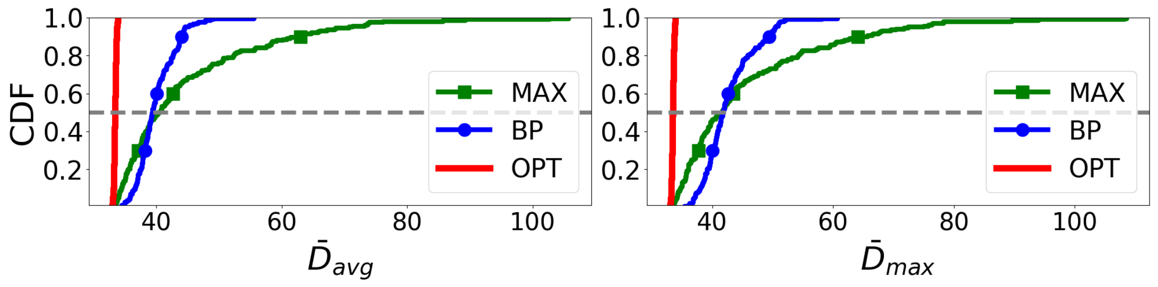


Figure 2-13: CDFs of $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$ in $16 \times 12 \times 8 \times 6$ networks

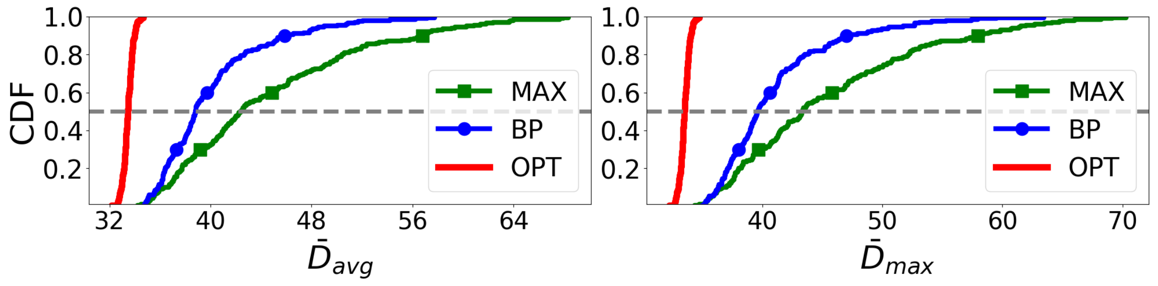


Figure 2-14: CDFs of $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$ in $15 \times 12 \times 9 \times 12 \times 15$ networks

explicit form of the min-delay condition (2.14) (i) facilitates co-optimization of delay and other metrics in data center networks with routing constraints, (ii) provides high flexibility in selecting $\boldsymbol{\lambda}$ based on different objectives in network control, and (iii) guarantees feasibility when adjacent layers are fully connected with sufficient capacity. For theory, we derive a more relaxed min-delay condition for tree data center structures than the general condition (2.14), and discuss our conjecture on the sufficient and necessary condition for delay minimization in multi-stage networks. We consider static policies in the following discussion for brevity, where the queue-based policies follow the same idea.

2.6.1 Practical Extensions

Multi-Objective Optimization with Routing Constraints

We have shown that a transmission rate vector $\mathbf{g}$ can achieve global minimum queueing delay if it is a feasible solution subject to the constraints (2.14). This result implies that we can co-optimize delay with some other metric $f(\mathbf{g})$ under the min-delay constraints (2.14) as follows.

$$\min_{\mathbf{g}} \quad f(\mathbf{g}) \qquad \text{s.t.} \quad \mathbf{g} \in (2.14). \tag{2.19}$$

As long as $f(\mathbf{g})$ is a convex function, (2.19) is a convex optimization problem that can be efficiently solved since the min-delay constraints (2.14) are linear given any $\boldsymbol{\gamma} \in \mathbb{R}_+$. Common examples of $f(\mathbf{g})$ include (i) minimizing total required bandwidth $f(\mathbf{g}) = \sum_{(i,j)\in\mathcal{E}} g_{ij}$ [103]; (ii) minimizing the maximum link rate $f(\mathbf{g}) = \max_{(i,j)\in\mathcal{E}} g_{ij}/c_{ij}$ and average link rate $f(\mathbf{g}) = |\mathcal{E}|^{-1} \sum_{(i,j)\in\mathcal{E}} (g_{ij}/c_{ij})$ [92]; (iii) minimizing the maximum queue overload rate $f(\mathbf{g}) = \max_{i\in\mathcal{V}} \sum_{k:(k,i)\in\mathcal{E}} g_{ki} - \sum_{j:(i,j)\in\mathcal{E}} g_{ij}$ [90]; (iv) minimizing the maximum total queue growth at a layer $f(\mathbf{g}) = \max_{l=1,\dots,L} \sum_{i\in\mathcal{V}_l} \left(\sum_{k:(k,i)\in\mathcal{E}} g_{ki} - \sum_{j:(i,j)\in\mathcal{E}} g_{ij}\right)$. The high generality of $f(\mathbf{g})$ demonstrates the wide application of (2.19) for multi-objective optimization together with delay minimization in data center networks.

We can further add constraints on the transmission rate vector $\mathbf{g}$ in (2.19) in order to capture the restrictions on link rate control. Note that introducing any convex constraint on $\mathbf{g}$ does not violate the convex property of (2.19). We give some common examples. (i) Packets cannot go through link (i,j): $g_{ij} = 0$; (ii) A maximum ratio β of packets can be transmitted from node i to j: $g_{ij} \le \beta \left(\sum_{k:(k,i)\in\mathcal{E}} g_{ki}\right)$; (iii) The maximum link utilization should not be higher than a threshold θ in order to avoid link overflow: $\max_{(i,j)\in\mathcal{E}} g_{ij}/c_{ij} \le \theta$.

Choices of $\boldsymbol{\gamma}$

Theorem 2.3 shows that both of the minimum $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$ are independent of $\boldsymbol{\gamma}$, however we show that $\boldsymbol{\gamma}$ affects the overload levels at different nodes and layers, where a lower value of γ_l means a higher proportion of packets ingress to layer l will be backlogged. We give an example on choosing $\boldsymbol{\gamma}$ to balance the total queue growth rates among all the L layers. Denote the total arrival rates at layer 1 by $\lambda_{\text{sum}} := \sum_{i=1}^{|\mathcal{V}_1|} \lambda_i$ and the total service rates at layer L by $\mu_{\text{sum}} := \sum_{j=1}^{|\mathcal{V}_L|} \mu_j$. Then the state we pursue is that the total queue growth rate at each layer is equal to

$(\lambda_{\text{sum}} - \mu_{\text{sum}})/L$. The corresponding $\boldsymbol{\gamma}$ that reaches this state is

$$\gamma_l = \frac{\lambda_{\text{sum}} - \frac{l-1}{L}(\lambda_{\text{sum}} - \mu_{\text{sum}})}{\lambda_{\text{sum}} - \frac{l}{L}(\lambda_{\text{sum}} - \mu_{\text{sum}})},\ l = 1, \ldots, L. \tag{2.20}$$

Note that we can easily derive a weighted version of (2.20) based on the node buffers that can balance the buffer utilization at each layer [32,104,105]. The main point we convey from this example is the high flexibility of choosing $\boldsymbol{\gamma}$ according to different objectives in network control.

Feasibility

We point out that it is difficult to characterize the conditions on the existence of a feasible transmission rate vector $\mathbf{g}$ subject to (2.14) in multi-stage networks under general network topology and link capacities. Instead, we show in Proposition 2.1 the existence of a solution under any given $\boldsymbol{\gamma} \in \mathbb{R}_+$, considering full connection between adjacent layers and sufficient link capacities so that the minimum network cut is $\sum_{i=1}^{|\mathcal{V}_L|} \mu_j$. We leave the discussion of general topologies and link capacities to future work.

Proposition 2.1. *Consider an L-layer multi-stage network with full connections and sufficient link capacities between adjacent layers. Given the arrival rate vector $\boldsymbol{\lambda}$ and the service rate vector $\boldsymbol{\mu}$, there exists a feasible $\mathbf{g}$ that satisfies the min-delay conditions* (2.14) *under any given $\boldsymbol{\gamma} \in \mathbb{R}_+$.*

Proof. Given a feasible $\boldsymbol{\gamma}$, we can construct a feasible solution to (2.14) as follows based on the full connection and sufficient capacity constraints. For the l-th layer ($l < L$), we first set its total egress rates as $\sum_{n_i^l \in \mathcal{V}_l} \sum_{n_j^{l+1}:(n_i^l, n_j^{l+1}) \in \mathcal{E}} g_{n_i^l, n_j^{l+1}} = \left(\sum_{i=1}^{|\mathcal{V}_1|} \lambda_i\right) \prod_{l'=1}^{l} \gamma_l$, and we randomly set a positive value for each $n_i^l \in \mathcal{V}_l$, denoted by $g_{n_i^l}^E$, which represents the egress rate of node n_i^l, subject to $\sum_{n_i^l \in \mathcal{V}_l} g_{n_i^l}^E = \sum_{n_i^l \in \mathcal{V}_l} \sum_{n_j^{l+1}:(n_i^l, n_j^{l+1}) \in \mathcal{E}} g_{n_i^l, n_j^{l+1}}$. We can now figure out a feasible solution that satisfies (2.14) by setting

$$g_{n_i^l, n_j^{l+1}} = \gamma_l \times g^I_{n_i^l} \times \left(g^E_{n_j^{l+1}} / \sum_{k=1}^{|\mathcal{V}_{l+1}|} g^E_{n_k^{l+1}} \right)$$

for $\forall n_i^l \in \mathcal{V}_l$ and $\forall n_j^{l+1} \in \mathcal{V}_{l+1}$ iteratively from layer 1 to $L-1$ to calculate $g^I_{n_i^l}$, the ingress to each node n_i^l, where we define $g^I_{n_i^1} := \lambda_i$ for each of the ingress nodes n_i^1 and $g^I_{n_i^l} := \sum_{n_k^{l-1}:(n_k^{l-1}, n_i^l)\in\mathcal{E}} g_{n_k^{l-1}, n_i^l}$. □

Remark: The existence of feasible solutions to (2.14) itself does not indicate minimum delay, which requires achieving maximum throughput in the meantime.

2.6.2 Theoretical Extensions

Tree Data Center Structure

A tree structure is a special case of the multi-stage network whose undirected topology is a tree. We visualize an example of a 3-layer tree in Fig. 2-15, which contains 6 ingress nodes and a single egress node. We focus on the fan-in structure where $|\mathcal{V}_{l+1}| \leq |\mathcal{V}_l|, \forall l = 1, \dots, L-1$ below, while the discussion of general fan-out structures follows in a similar manner. We derive a sufficient condition on the link rates to achieve minimum $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$ in Theorem 2.8 based on the *parent source set* of an node n_i^l at a layer l, denoted by $PSS[n_i^l]$, which is defined as all the nodes at the ingress layer that are the parents of n_i^l, and we set $PSS[n_i^1] = \{n_i^1\}$ for each ingress node n_i^1. In Fig. 2-15, we have $PSS[n_1^2] = \{n_1^1, n_2^1\}$, $PSS[n_2^2] = \{n_3^1, n_4^1\}$, $PSS[n_3^2] = \{n_5^1, n_6^1\}$, and $PSS[n_1^3] = \mathcal{V}_S$.

Theorem 2.8. *Consider an L-layer tree data center structure. Any transmission policy* $\mathbf{g}$ *that guarantees maximum throughput and in the meantime satisfies the following conditions achieves minimum $\bar{D}_{avg}$ and $\bar{D}_{max}$: For $\forall l \in 2, \dots, L$ and $\forall n_j^l \in \mathcal{V}_l$, we have $\frac{\sum_{k\in PSS[n_i^{l-1}]} \lambda_k}{g_{n_i^{l-1}, n_j^l}} = \gamma_{n_j^l}$ for some $\gamma_{n_j^l} > 0$ for $\forall n_i^{l-1} \in \mathcal{V}_{l-1}$ with $(n_i^{l-1}, n_j^l) \in \mathcal{E}$.*

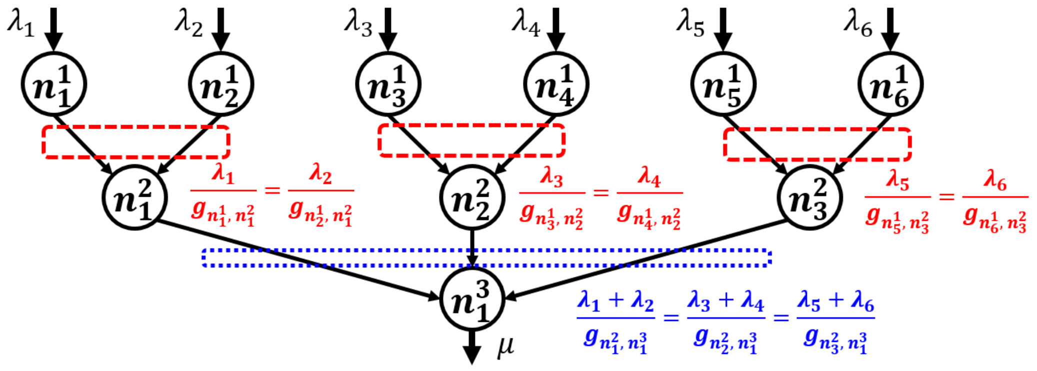

λ_1
λ_2
λ_3
λ_4
λ_5
λ_6
n_1^1
n_2^1
n_3^1
n_4^1
n_5^1
n_6^1
n_1^2
n_2^2
n_3^2
n_1^3
$\frac{\lambda_1}{g_{n_1^1,n_1^2}} = \frac{\lambda_2}{g_{n_2^1,n_1^2}}$
$\frac{\lambda_3}{g_{n_3^1,n_2^2}} = \frac{\lambda_4}{g_{n_4^1,n_2^2}}$
$\frac{\lambda_5}{g_{n_5^1,n_3^2}} = \frac{\lambda_6}{g_{n_6^1,n_3^2}}$
$\frac{\lambda_1+\lambda_2}{g_{n_1^2,n_1^3}} = \frac{\lambda_3+\lambda_4}{g_{n_2^2,n_1^3}} = \frac{\lambda_5+\lambda_6}{g_{n_3^2,n_1^3}}$
μ

Figure 2-15: A 3-layer fan-in tree data center example under min-delay transmission policies derived in Theorem 2.8.

We prove Theorem 2.8 under a 3-layer $4\times 2\times 1$ tree structure in Section 2.8 which can be applied to general multi-layer tree structures. We explain Theorem 2.8 through the example in Fig. 2-15. Layer 2 contains three nodes n_1^2, n_2^2, n_3^2. The parents of these three nodes at layer 1 are $\{n_1^1, n_2^1\}$, $\{n_3^1, n_4^1\}$, and $\{n_5^1, n_6^1\}$ respectively. Then a sufficient condition to achieve minimum delay in Theorem 2.8 corresponding to these three nodes at layer 2 are

$$\begin{cases} \lambda_1/g_{n_1^1,n_1^2} = \lambda_2/g_{n_2^1,n_1^2} = \gamma_{n_1^2} \\ \lambda_3/g_{n_3^1,n_2^2} = \lambda_4/g_{n_4^1,n_2^2} = \gamma_{n_2^2} \\ \lambda_5/g_{n_5^1,n_3^2} = \lambda_6/g_{n_6^1,n_3^2} = \gamma_{n_3^2} \end{cases} \tag{2.21}$$

for some $\gamma_{n_1^2}, \gamma_{n_2^2}, \gamma_{n_3^2} > 0$. Similarly, layer 3 contains a single node n_1^3, whose parents are $\{n_1^2, n_2^2, n_3^2\}$. The corresponding condition for n_1^3 is that for some $\gamma_{n_1^3} > 0$,

$$\frac{\lambda_1+\lambda_2}{g_{n_1^2,n_1^3}} = \frac{\lambda_3+\lambda_4}{g_{n_2^2,n_1^3}} = \frac{\lambda_5+\lambda_6}{g_{n_3^2,n_1^3}} = \gamma_{n_1^3} \tag{2.22}$$

Combining (2.21) and (2.22) gives a sufficient condition on $\mathbf{g}$ to minimize $\bar{D}_{\text{avg}}$ and $\bar{D}_{\text{max}}$ for the 3-layer tree in Fig. 2-15 as long as maximum throughput is achieved.

An implication of Theorem 2.8 is that its min-delay condition also follows the rate-proportional property, where the transmission rates of the egress links from the same layer should be in the same proportion to the total external packet arrival rates among their parent source sets. Moreover, it is straightforward to verify that any policy that satisfies the general min-delay condition (2.14) in the tree structure must satisfy the condition in Theorem 2.8. This means Theorem 2.8 gives a more relaxed sufficient condition on link rate control to minimize delay than applying (2.14) directly to the tree.

Conjecture on Sufficient and Necessary Conditions

We have yet derived the necessary conditions to achieve minimum delay in general multi-stage networks. The challenge is to characterize all the min-delay policies due to the possible switching from non-zero to zero queue length at some nodes whose egress rates are greater than ingress rates in the time window $[t_0, t_0 + T]$. The $N \times 1$ network is the special case that we can fully characterize the complete set of min-delay policies, as shown in Theorem 2.1.

We conjecture that the crux lies in that the actual transmission rate of a link (i, j) starting from a node with zero queue length are not equal to the transmission rate we originally set. Consider the example where a node i at a layer l with links $(1, i)$, $(2, i)$, $(i, 3)$, and $(i, 4)$, where node 1 and 2 are at layer $l - 1$, node 3 and 4 are at layer $l + 1$. Suppose we set $g_{1i} = 1$, $g_{2i} = 2$, $g_{i3} = g_{i4} = 3$. Clearly $g_{1i} + g_{2i} < g_{i3} + g_{i4}$ thus $q_i(t) = 0$ starting from some time. Since the definition of link rate is the number of packets transmitted over this link in a time unit, there are 3 units of packets injected into node i at time t, and 1.5 units will be served to each of the egress links $(i, 3)$ and $(i, 4)$, and thus the actual transmission rates over $(i, 3)$ and $(i, 4)$ are both 1.5, less than $g_{i3} = g_{i4} = 3$. Following this idea, given the transmission rate vector $\mathbf{g}$ we set, we can obtain the actual transmission rates denoted by $\tilde{\mathbf{g}}$ based on (2.23) starting from the ingress layer in multi-stage networks: For $\forall l = 1, \dots, L-1,\ \forall i \in \mathcal{V}_l, j \in \mathcal{V}_{l+1}$,

$$\tilde{g}_{ij} = \begin{cases} g_{ij}, \text{if} \sum\limits_{k\in\mathcal{V}_{l-1}} \tilde{g}_{ki} \geq \sum\limits_{k\in\mathcal{V}_{l+1}} g_{ik} \\ g_{ij} \frac{\sum_{k\in\mathcal{V}_{l-1}} \tilde{g}_{ki}}{\sum_{k\in\mathcal{V}_{l+1}} g_{ik}}, \text{o.w.} \end{cases} = g_{ij} \min\left\{1, \frac{\sum_{k\in\mathcal{V}_{l-1}} \tilde{g}_{ki}}{\sum_{k\in\mathcal{V}_{l+1}} g_{ik}}\right\} \tag{2.23}$$

where we define $\sum_{k\in\mathcal{V}_0} \tilde{g}_{ki} := \lambda_i,\ \forall i \in \mathcal{V}_1$ to incorporate the ingress layer in (2.23).

We have the following conjecture of the sufficient and necessary condition to minimize $\bar{D}_{\text{avg}}$ and $\bar{D}_{\max}$ based on actual transmission rate vector $\tilde{\mathbf{g}}$.

Conjecture 2.1. *Consider an L-layer multi-stage network. The sufficient and necessary condition of a transmission policy $\mathbf{g}$ to minimize both $\bar{D}_{avg}$ and $\bar{D}_{max}$ is that its corresponding actual transmission rate vector $\tilde{\mathbf{g}}$ satisfies* (2.14) *and achieves maximum throughput.*

We can validate the conjecture in $N \times 1$ networks based on Fig. 2-5: Any $\mathbf{g}$ such that $g_i \geq \lambda_i$, $\forall i = 1, \ldots, N$ leads to minimum delay, and its corresponding $\tilde{g}_i = \lambda_i$ and thus $\{\tilde{g}_i\}_{i=1}^N$ are in proportion to $\{\lambda_i\}_{i=1}^N$. However, if there is a single $i_0 \in \mathcal{V}_S$ such that $g_{i_0} < \lambda_{i_0}$, which does not induce minimum delay, then clearly $\tilde{g}_{i_0} = g_{i_0}$ and thus $\tilde{g}_{i_0}/\lambda_{i_0} < \tilde{g}_i/\lambda_i = 1$, $\forall i \neq i_0$, i.e., (2.14) does not hold for $\tilde{\mathbf{g}}$. We further visualize an example of 2×2 networks in Fig. 2-16 to explain the conjecture. Furthermore, we point out that the conjecture can be generalized to queue-based policies, where we require that the corresponding actual transmission rate vector $\tilde{\mathbf{g}}(\mathbf{q}(t))$ meets the queue-proportional condition (2.18) and maximum throughput requirement in Theorem 2.7. We leave the proof of this conjecture to future work.

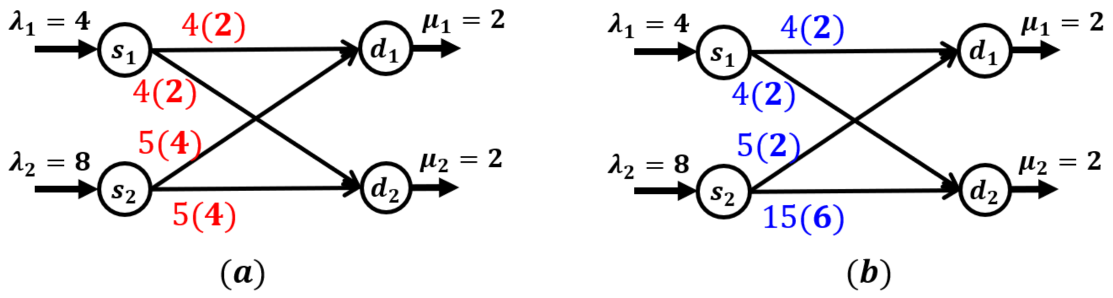


Figure 2-16: Validation of Conjecture 2.1 in 2×2 single-hop networks, where the notation $x(y)$ over a link (i, j) means $g_{ij} = x$ and its actual transmission rate $\bar{g}_{ij} = y$ calculated based on (2.23): (a) setting $\mathbf{g} = [4, 4, 5, 5]$ leads to $\tilde{\mathbf{g}} = [2, 2, 4, 4]$ which satisfies (2.18) thus min-delay; (b) setting $\mathbf{g} = [4, 4, 5, 15]$ leads to $\tilde{\mathbf{g}} = [2, 2, 2, 6]$ which does not satisfy (2.18) thus not min-delay.

2.7 Summary and Future Work

We study link rate control for queueing delay minimization in overloaded networks. Leveraging the fluid queueing model, we show that any static rate-proportional policy, which guarantees identical ratios between the ingress and egress rates of all the nodes at each layer, minimizes the average delay $\bar{D}_{\text{avg}}$ and the maximum ingress delay $\bar{D}_{\max}$ in general single-hop and multi-stage networks. We further extend the result to the queue-proportional policies which can achieve asymptotically minimum delay based on real-time queue information agnostic of packet arrival rates. We evaluate the performance of our proposed policies under different network settings, validate their min-delay property, and demonstrate their superiority in delay reduction compared with the backpressure policy and the max-link-rate policy. We finally discuss the extensions of the main results in practice and in theory. We envision multiple future directions including proving sufficient and necessary conditions on link rate control for delay minimization in general multi-stage networks, the extension of main results to multi-hop networks, and the implementation of the proposed policies in real data center networks.

2.8 Chapter Appendix

2.8.1 Proof of Theorem 2.2

We present the proof sketch here. Due to the space limit, we only prove for $\bar{D}_{\text{avg}}$, where $\bar{D}_{\max}$ is similar. For packets that arrive at the ingress node s_1 at time t and finally transmitted to the egress node d_1, the total queueing delay is

$$\begin{aligned}
D_{s_1 d_1}(t) &= \frac{q_{s_1}(t)}{g_{11}+g_{12}} + \frac{q_{d_1}\left(t+\frac{q_{s_1}(t)}{g_{11}+g_{12}}\right)}{\mu_1} \\
&= \frac{q_{s_1}(t)}{g_{11}+g_{12}} + \frac{1}{\mu_1}\max\left\{0, q_{d_1}(t) + \frac{q_{s_1}(t)}{g_{11}+g_{12}}(g_{11}+g_{21}-\mu)\right\} \\
&= \begin{cases} \frac{q_{d_1}(t)}{\mu_1} + \frac{q_{s_1}(t)}{\mu_1}\frac{g_{11}+g_{21}}{g_{11}+g_{12}}, & g_{11}+g_{21} \geq \mu_1 \\ \frac{q_{s_1}(t)}{g_{11}+g_{12}}, & g_{11}+g_{21} < \mu_1 \end{cases}
\end{aligned}$$

Since $\mathbf{q}(t_0) = \mathbf{0}$, then the average delay for packets from s_1 to d_1, denoted as $\bar{D}_{s_1 d_1}$, is

$$\begin{aligned}
\bar{D}_{s_1 d_1} &:= \frac{1}{T}\int_{t_0}^{t_0+T} D_{s_1 d_1}(t)dt \\
&= \begin{cases} \frac{T}{2\mu_1}(g_{11}+g_{21}-\mu_1) + \frac{T}{2\mu_1}\frac{g_{11}+g_{21}}{g_{11}+g_{12}}\max\{\lambda_1 - g_{11} - g_{12}, 0\}, & g_{11}+g_{21} \geq \mu_1 \\ \frac{T}{2(g_{11}+g_{21})}\max\{\lambda_1 - g_{11} - g_{12}, 0\}, & g_{11}+g_{21} \leq \mu_1 \end{cases}
\end{aligned}$$

We can verify that among all transmission rate vectors $\mathbf{g}$'s that $g_{11}+g_{21} \leq \mu_1$, the $\mathbf{g}$'s that satisfy $g_{11}+g_{21} = \mu_1$ achieve minimum delay[3]. Therefore the minimum delay achieved under $g_{11}+g_{21} \geq \mu_1$ is exactly the global optimum, under which

$$\bar{D}_{s_1 d_1} = \begin{cases} \frac{T}{2\mu_1}\left(\lambda_1 \frac{g_{11}+g_{21}}{g_{11}+g_{12}} - \mu_1\right), & g_{11}+g_{12} \leq \lambda_1 \\ \frac{T}{2\mu_1}\left(g_{11}+g_{21}-\mu_1\right), & g_{11}+g_{12} \geq \lambda_1 \end{cases}$$

Generally, we can obtain that for any $(i,j) \in \{(1,1),(1,2),(2,1),(2,2)\}$,

$$\bar{D}_{s_i d_j} = \begin{cases} \frac{T}{2\mu_j}\left(\lambda_i \frac{g_{1j}+g_{2j}}{g_{i1}+g_{i2}} - \mu_j\right), & g_{i1}+g_{i2} \leq \lambda_i \\ \frac{T}{2\mu_j}\left(g_{1j}+g_{2j}-\mu_2\right), & g_{i1}+g_{i2} \geq \lambda_i \end{cases}$$

[3]The intuition is clear that $g_{11}+g_{21} < \mu_1$ does not fully utilize the service capability of node d_1.

Therefore, we have

$$\begin{aligned}\bar{D}_{\text{avg}} = & \frac{\lambda_1}{\lambda_1+\lambda_2}\frac{g_{11}}{g_{11}+g_{12}}\bar{D}_{s_1d_1} + \frac{\lambda_1}{\lambda_1+\lambda_2}\frac{g_{12}}{g_{11}+g_{12}}\bar{D}_{s_1d_2} \\ & + \frac{\lambda_2}{\lambda_1+\lambda_2}\frac{g_{21}}{g_{21}+g_{22}}\bar{D}_{s_2d_1} + \frac{\lambda_2}{\lambda_1+\lambda_2}\frac{g_{22}}{g_{21}+g_{22}}\bar{D}_{s_2d_2}\end{aligned}$$

where $\frac{\lambda_i}{\lambda_1+\lambda_2}\frac{g_{ij}}{g_{i1}+g_{i2}}$ denotes the portion of packets from s_i to d_j that arrive within $[t_0, t_0+T]$. We again consider the four regions $\{\mathbf{g} \mid g_{11}+g_{12} \lesseqgtr \lambda_1, g_{21}+g_{22} \lesseqgtr \lambda_2\}$ and prove that the optimal solutions constrained in each region are all global optimum. Due to space limit we only show the details of the case $g_{11}+g_{12} \le \lambda_1,\ g_{21}+g_{22} \le \lambda_2$. In this case, the average delay of packets that arrive in $[t_0, t_0+T]$ among all ingress nodes is

$$\begin{aligned}\bar{D}_{\text{avg}} := & \frac{\lambda_1\frac{g_{11}}{g_{11}+g_{12}}D_{s_1d_1} + \lambda_1\frac{g_{12}}{g_{11}+g_{12}}D_{s_1d_2} + \lambda_2\frac{g_{21}}{g_{21}+g_{22}}D_{s_2d_1} + \lambda_2\frac{g_{22}}{g_{21}+g_{22}}D_{s_2d_2}}{\lambda_1+\lambda_2} \\ \sim & \frac{g_{11}+g_{21}}{\mu_1}\left(\frac{\lambda_1^2}{g_{11}}\left(\frac{g_{11}}{g_{11}+g_{12}}\right)^2 + \frac{\lambda_2^2}{g_{21}}\left(\frac{g_{21}}{g_{21}+g_{22}}\right)^2\right) \\ & + \frac{g_{12}+g_{22}}{\mu_2}\left(\frac{\lambda_1^2}{g_{12}}\left(\frac{g_{12}}{g_{11}+g_{12}}\right)^2 + \frac{\lambda_2^2}{g_{22}}\left(\frac{g_{22}}{g_{21}+g_{22}}\right)^2\right) \\ = & \frac{1}{\mu_1}\left(\lambda_1^2x^2 + \lambda_2^2y^2 + \lambda_1^2x^2\frac{g_{21}}{g_{11}} + \lambda_2^2y^2\frac{g_{11}}{g_{21}}\right) \\ & + \frac{1}{\mu_2}\left(\lambda_1^2(1-x)^2 + \lambda_2^2(1-y)^2 + \lambda_1^2(1-x)^2\frac{g_{22}}{g_{12}} + \lambda_2^2(1-y)^2\frac{g_{12}}{g_{22}}\right) \\ \overset{\text{(i)}}{\ge} & \frac{1}{\mu_1}(\lambda_1x+\lambda_2y)^2 + \frac{1}{\mu_2}(\lambda_1(1-x)+\lambda_2(1-y))^2 \\ \overset{\text{(ii)}}{\ge} & \frac{T}{2(\mu_1+\mu_2)}(\lambda_1+\lambda_2) - \frac{T}{2} = \frac{T}{2(\mu_1+\mu_2)}(\lambda_1+\lambda_2-\mu_1-\mu_2)\end{aligned}$$

where $\sim$ means removing constant terms, $x := \frac{g_{11}}{g_{11}+g_{12}}$, and $y := \frac{g_{21}}{g_{21}+g_{22}}$. The inequality (i) stems from Cauchy-Schwartz Inequality, which turns into equality when $\frac{g_{11}}{g_{21}} = \frac{\lambda_1x}{\lambda_2y}$, $\frac{g_{12}}{g_{22}} = \frac{\lambda_1(1-x)}{\lambda_2(1-y)}$ and equivalently,

$$\frac{g_{11}+g_{12}}{g_{21}+g_{22}} = \frac{\lambda_1}{\lambda_2}. \tag{2.24}$$

The inequality (ii) holds due to solving

$$\min_{x,y\in[0,1]} \quad \frac{1}{\mu_1}(\lambda_1 x+\lambda_2 y)^2+\frac{1}{\mu_2}(\lambda_1(1-x)+\lambda_2(1-y))^2$$

where the optimal (x,y) satisfies

$$\begin{cases}\lambda_1 x+\lambda_2 y=\frac{\lambda_1+\lambda_2}{\mu_2}\left(\frac{1}{\mu_1}+\frac{1}{\mu_2}\right)^{-1}\\ \lambda_1(1-x)+\lambda_2(1-y)=\frac{\lambda_1+\lambda_2}{\mu_1}\left(\frac{1}{\mu_1}+\frac{1}{\mu_2}\right)^{-1}\end{cases}$$

which, combined with (2.24), is equivalent to

$$\begin{cases}\frac{\lambda_2}{\lambda_1+\lambda_2}\frac{g_{11}+g_{21}}{g_{21}+g_{22}}=\frac{\mu_1}{\mu_1+\mu_2}\\ \frac{\lambda_2}{\lambda_1+\lambda_2}\frac{g_{12}+g_{22}}{g_{21}+g_{22}}=\frac{\mu_2}{\mu_1+\mu_2}\end{cases}$$

and thus

$$\frac{g_{11}+g_{21}}{g_{12}+g_{22}}=\frac{\mu_1}{\mu_2} \tag{2.25}$$

suffices to make the inequality (ii) achieve its lower bound. Therefore (2.24) and (2.25) with $g_{11}+g_{21}\geq\mu_1,\ g_{12}+g_{22}\geq\mu_2$ give us the sufficient and necessary condition on $\mathbf{g}$ to minimize $\bar{D}_{\text{avg}}$ under $g_{11}+g_{12}\leq\lambda_1$ and $g_{21}+g_{22}\leq\lambda_2$.

2.8.2 Proof of Theorem 2.3

We prove the min-delay conditions on static transmission policies under a 3-layer network with 2 nodes at each layer, and links connecting each pair of nodes at adjacent layers, as shown in Fig. 2-17. Note that we re-index each node from 1 to 6 to increase readability of the proof. The results can be smoothly extended to general L-layer networks with arbitrary numbers of nodes at each layer and links between adjacent layers. For simplicity, we consider zero initial queue length of each node.

Note that we can classify all the packets going through the 3-layer network by their paths. We characterize the total queueing delay of a packet taking the path $1\to 3\to 5$ as in (2.26), where the other possible paths are in similar form.

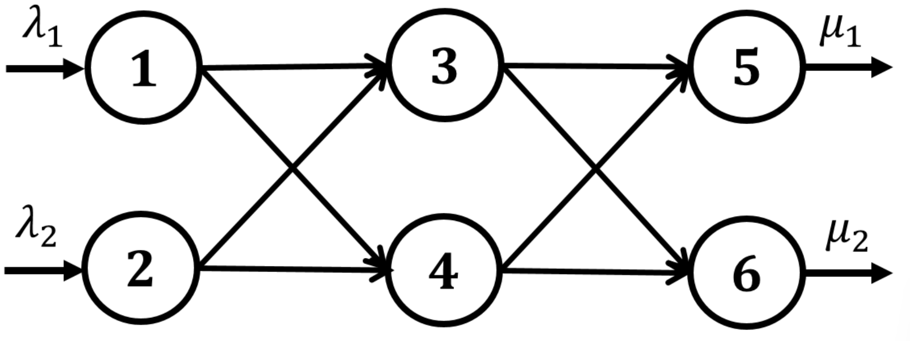


Figure 2-17: An example of a 3-layer $2 \times 2 \times 2$ network

$$\begin{aligned}
D_{135} &= \frac{q_1(t)}{g_{13}+g_{14}} + \frac{q_3\left(t+\frac{q_1(t)}{g_{13}+g_{14}}\right)}{g_{35}+g_{36}} + \frac{q_5\left(t+\frac{q_1(t)}{g_{13}+g_{14}}+\frac{q_3\left(t+\frac{q_1(t)}{g_{13}+g_{14}}\right)}{g_{35}+g_{36}}\right)}{\mu} \\
&\overset{(a)}{=} q_1(t) \times \frac{1}{g_{13}+g_{14}} \times \frac{g_{13}+g_{23}}{g_{35}+g_{36}} \times \frac{g_{35}+g_{45}}{\mu_1} + q_3(t) \times \frac{1}{g_{35}+g_{36}} \times \frac{g_{35}+g_{45}}{\mu_1} + q_5(t) \times \frac{1}{\mu_1} \\
&\overset{(b)}{=} \left(\frac{\lambda_1}{g_{13}+g_{14}} \times \frac{g_{13}+g_{23}}{g_{35}+g_{36}} \times \frac{g_{35}+g_{45}}{\mu_1} - 1\right) t
\end{aligned} \tag{2.26}$$

where (a) stems from the expansion of $q_3\left(t+\frac{q_1(t)}{g_{13}+g_{14}}\right)$ and $q_5\left(t+\frac{q_1(t)}{g_{13}+g_{14}}+\frac{q_3\left(t+\frac{q_1(t)}{g_{13}+g_{14}}\right)}{g_{35}+g_{36}}\right)$ into a linear combination of $q_1(t)$, $q_3(t)$, and $q_5(t)$, for example

$$q_3\left(t+\frac{q_1(t)}{g_{13}+g_{14}}\right) = q_3(t) + \frac{q_1(t)}{g_{13}+g_{14}} \times (g_{13}+g_{23}-g_{34}-g_{35})$$

and similarly for $q_5\left(t+\frac{q_1(t)}{g_{13}+g_{14}}+\frac{q_3\left(t+\frac{q_1(t)}{g_{13}+g_{14}}\right)}{g_{35}+g_{36}}\right)$; (b) is due to $q_1(t) = (\lambda_1 - g_{13} - g_{14})t$, $q_3(t) = (g_{13}+g_{23}-g_{34}-g_{35})t$, and $q_5(t) = (g_{35}+g_{45}-\mu_1)t$. We can similarly express the queueing delay of packets taking the other 7 paths.

We can then express the average queueing delay of packets that arrive to the network in overload within time window $[0,T]$ as $\bar{D}_{\text{avg}} := \sum_{p\in\mathcal{P}} w_p D_p$, where $\mathcal{P}$ denotes the set of possible paths of packets, and w_p denotes the proportion of the packets that take the path p. For example,

$$w_{135} = \lambda_1 \frac{g_{13}}{g_{13}+g_{14}} \frac{g_{35}}{g_{35}+g_{36}},$$

and similarly for the other paths. In the following, we derive the conditions on the transmission rate vector $\mathbf{g}$ that minimizes $\bar{D}_{\text{avg}}$. We first consider the sum of the weighted delay for two paths $1 \to 3 \to 5$ and $2 \to 3 \to 5$, which can be derived as

$$\begin{aligned}
w_{135}D_{135} + w_{235}D_{235} &= \frac{g_{13}+g_{23}}{\mu_1} \times \frac{g_{35}(g_{35}+g_{45})}{(g_{35}+g_{36})^2} \times \left(\lambda_1^2 \frac{g_{13}}{(g_{13}+g_{14})^2} + \lambda_2^2 \frac{g_{23}}{(g_{23}+g_{24})^2}\right) \\
&= \frac{1}{\mu_1} \times \frac{g_{35}(g_{35}+g_{45})}{(g_{35}+g_{36})^2} \times \left(\lambda_1^2 x^2 + \lambda_2^2 y^2 + \lambda_1^2 x^2 \frac{g_{23}}{g_{13}} + \lambda_2^2 y^2 \frac{g_{13}}{g_{23}}\right) \\
&\geq \frac{1}{\mu_1} \times \frac{g_{35}(g_{35}+g_{45})}{(g_{35}+g_{36})^2} \left(\lambda_1 x + \lambda_2 y\right)^2
\end{aligned}$$

where $x = \frac{g_{13}}{g_{13}+g_{14}}$ and $y = \frac{g_{23}}{g_{23}+g_{24}}$, and the last inequality is based on Cauchy-Schwartz inequality, where the condition of link rates to reach the lower bound is

$$\frac{\lambda_1 x}{\lambda_2 y} = \frac{g_{13}}{g_{23}}. \tag{2.27}$$

We can similarly derive the same condition (2.27) for the sum $w_{136}D_{136} + w_{236}D_{236}$ reaching its lower bound. For the other two sums $w_{145}D_{145} + w_{245}D_{245}$ and $w_{146}D_{146} + w_{246}D_{246}$, the condition becomes

$$\frac{\lambda_1(1-x)}{\lambda_2(1-y)} = \frac{g_{14}}{g_{24}}. \tag{2.28}$$

Note that with $x = \frac{g_{13}}{g_{13}+g_{14}}$ and $y = \frac{g_{23}}{g_{23}+g_{24}}$, both (2.27) and (2.28) lead to the condition

$$\frac{g_{13}+g_{14}}{g_{23}+g_{24}} = \frac{\lambda_1}{\lambda_2}. \tag{2.29}$$

Suppose that (2.29) is satisfied below. We can further extend the above idea to derive the min-delay transmission rates between layer 2 and 3. Specifically, let $z = \frac{g_{35}}{g_{35}+g_{36}}$, $w = \frac{g_{45}}{g_{45}+g_{46}}$, then we have

$$\begin{aligned}
&w_{135}D_{135} + w_{235}D_{235} + w_{145}D_{145} + w_{245}D_{245} \\
&= \frac{1}{\mu_1}\Big((\lambda_1 x + \lambda_2 y)^2 z^2 \big(1 + \frac{g_{45}}{g_{35}}\big) + (\lambda_1(1-x) + \lambda_2(1-y))^2 w^2 \big(1 + \frac{g_{35}}{g_{45}}\big)\Big) \\
&\geq \frac{1}{\mu_1}\left((\lambda_1 x + \lambda_2 y)z + (\lambda_1(1-x) + \lambda_2(1-y))w\right)^2
\end{aligned}$$

where the condition that achieves the lower bound is

$$\frac{z(\lambda_1 x+\lambda_2 y)}{w(\lambda_1(1-x)+\lambda_2(1-y))}=\frac{g_{35}}{g_{45}}=\frac{g_{35}+g_{36}}{g_{45}+g_{46}}=\frac{\lambda_1 x+\lambda_2 y}{\lambda_1(1-x)+\lambda_2(1-y)}. \tag{2.30}$$

The other half $w_{136}D_{136}+w_{236}D_{236}+w_{146}D_{146}+w_{246}D_{246}$ leads to the same condition as in (2.30).

Suppose that (2.30) is satisfied below. The problem to minimize the average queueing delay now becomes

$$\begin{aligned}\min_{x,y,z,w}\frac{1}{\mu_1}\left((\lambda_1 x+\lambda_2 y)z+(\lambda_1(1-x)+\lambda_2(1-y))w\right)^2\\ +\frac{1}{\mu_2}\left((\lambda_1 x+\lambda_2 y)(1-z)+(\lambda_1(1-x)+\lambda_2(1-y))(1-w)\right)^2\end{aligned}$$

Based on the Cauchy-Schwartz inequality, the optimal condition is that

$$\frac{(\lambda_1 x+\lambda_2 y)z+(\lambda_1(1-x)+\lambda_2(1-y))w}{(\lambda_1 x+\lambda_2 y)(1-z)+(\lambda_1(1-x)+\lambda_2(1-y))(1-w)}=\frac{\mu_1}{\mu_2} \tag{2.31}$$

We combine the three optimal conditions (2.29), (2.30), and (2.31) which altogether reach the lower bound of the average queuing delay of packets. We simplify them and obtain the min-delay constraints that reach the lower bound as follows.

$$\begin{cases}\frac{\lambda_1}{\lambda_2}=\frac{g_{13}+g_{14}}{g_{23}+g_{24}}\\ \frac{g_{13}+g_{23}}{g_{14}+g_{24}}=\frac{g_{35}+g_{36}}{g_{45}+g_{46}}\\ \frac{g_{35}+g_{45}}{g_{36}+g_{46}}=\frac{\mu_1}{\mu_2}\end{cases} \tag{2.32}$$

which means to guarantee an identical ratio between the ingress and egress rates of each node at a layer. Combining (2.32) and the condition to achieve maximum throughput in the network, we obtain a sufficient condition on a policy that minimizes the average queueing delay. We can derive similarly for $\bar{D}_{\max}$ minimization, which is the maximum average queueing delay of a packet in the network over different ingress nodes.

2.8.3 Proof of Theorem 2.8

We give the proof of a $4 \times 2 \times 1$ tree structure, where $\mathcal{V}_1 = \{n_1^1, n_2^1, n_3^1, n_4^1\}$, $\mathcal{V}_2 = \{n_1^2, n_2^2\}$, $\mathcal{V}_3 = \{n_1^3\}$. We denote the total queueing delay of a packet injected into node n_i^1 at the ingress layer at time t as $D_i(t)$, which is composed of the queueing delay at each of the 3 layers. For example,

$$D_1(t) = \frac{q_{n_1^1}(t)}{g_{n_1^1,n_1^2}} + \frac{q_{n_1^2}\left(t + \frac{q_{n_1^1}(t)}{g_{n_1^1,n_1^2}}\right)}{g_{n_1^2,n_1^3}} \quad + \frac{q_{n_1^3}\left(t + \frac{q_{n_1^1}(t)}{g_{n_1^1,n_1^2}} + \frac{q_{n_1^2}\left(t+\frac{q_{n_1^1}(t)}{g_{n_1^1,n_1^2}}\right)}{g_{n_1^2,n_1^3}}\right)}{\mu}$$

which is similar for $D_2(t)$, $D_3(t)$, and $D_4(t)$. We only show the derivation associated with packets that are injected into the network at node n_1^1. For simplicity, we only show the derivation of the case where all the nodes have positive length during overload, where the min-delay conditions also hold for the other cases. We can simplify $D_1(t)$ as

$$D_1(t) = q_{n_1^1}(t)\left(\frac{g_{n_1^2,n_1^3} + g_{n_2^2,n_1^3}}{\mu} \times \frac{g_{n_1^1,n_1^2} + g_{n_2^1,n_1^2}}{g_{n_1^2,n_1^3}} \times \frac{1}{g_{n_1^1,n_1^2}}\right)$$
$$+ q_{n_1^2}(t) \times \frac{g_{n_1^2,n_1^3} + g_{n_2^2,n_1^3}}{\mu} \times \frac{1}{g_{n_1^2,n_1^3}} + q_{n_1^3}\frac{1}{\mu}$$

and by expressing $q_i(t)$ as the multiplication of t (assuming $t_0 = 0$) and the queue growth rate, we have

$$D_1 := \frac{1}{T}\int_0^T D_1(t)dt = \frac{T}{2\mu}\left(\lambda_1 \times \frac{g_{n_1^2,n_1^3} + g_{n_2^2,n_1^3}}{g_{n_1^2,n_1^3}} \times \frac{g_{n_1^1,n_1^2} + g_{n_2^1,n_1^2}}{g_{n_1^1,n_1^2}} - \mu\right).$$

We can now express the average delay $\bar{D}_{\text{avg}}$ as

$$\begin{aligned}
\bar{D}_{\text{avg}} := \frac{\sum_{i=1}^{4}\lambda_i D_i}{\sum_{i=1}^{4}\lambda_i} &\overset{(a)}{\sim} \frac{g_{n_1^2,n_1^3}+g_{n_2^2,n_1^3}}{g_{n_1^2,n_1^3}}\left(\lambda_1^2\frac{g_{n_1^1,n_1^2}+g_{n_2^1,n_1^2}}{g_{n_1^1,n_1^2}}+\lambda_2^2\frac{g_{n_1^1,n_1^2}+g_{n_2^1,n_1^2}}{g_{n_2^1,n_1^2}}\right)\\
&+\frac{g_{n_1^2,n_1^3}+g_{n_2^2,n_1^3}}{g_{n_2^2,n_1^3}}\left(\lambda_3^2\frac{g_{n_3^1,n_2^2}+g_{n_4^1,n_2^2}}{g_{n_3^1,n_2^2}}+\lambda_4^2\frac{g_{n_3^1,n_2^2}+g_{n_4^1,n_2^2}}{g_{n_4^1,n_2^2}}\right)\\
&\overset{(b)}{\geq} \frac{g_{n_1^2,n_1^3}+g_{n_2^2,n_1^3}}{g_{n_1^2,n_1^3}}(\lambda_1+\lambda_2)^2+\frac{g_{n_1^2,n_1^3}+g_{n_2^2,n_1^3}}{g_{n_2^2,n_1^3}}(\lambda_3+\lambda_4)^2\\
&\overset{(c)}{\geq} (\lambda_1+\lambda_2+\lambda_3+\lambda_4)^2
\end{aligned} \tag{2.33}$$

where at (a) we remove the constant additive and multiplicative terms, and at (b) and (c) we apply the Cauchy-Schwarz inequality. The conditions to achieve equality at (b) are

$$\frac{g_{n_1^1,n_1^2}}{g_{n_2^1,n_1^2}}=\frac{\lambda_1}{\lambda_2},\qquad \frac{g_{n_3^1,n_2^2}}{g_{n_4^1,n_2^2}}=\frac{\lambda_3}{\lambda_4}, \tag{2.34}$$

and the condition to achieve equality at (c) is

$$\frac{g_{n_1^2,n_1^3}}{g_{n_2^2,n_1^3}}=\frac{\lambda_1+\lambda_2}{\lambda_3+\lambda_4}. \tag{2.35}$$

Therefore any policy that satisfies (2.34) and (2.35) leads to minimum $\bar{D}_{\text{avg}}$, which is $\frac{T}{2\mu}\max\left\{\left(\sum_{i=1}^{4}\lambda_i-\mu\right),0\right\}$. Similarly, we can show (2.34) and (2.35) together form a sufficient condition to minimize the maximum ingress delay $\bar{D}_{\max}$.

Chapter 3

Overload Balancing in Networks with Bounded Buffers

In this chapter, we consider overload balancing in single-hop networks with bounded buffers. We show that the backpressure policy, which is known to achieve the most balanced overload for networks with unbounded buffers, does not balance the overload for networks with bounded buffers. We formulate the problem of overload balancing in single-hop networks with bounded buffers by leveraging ordinary differential equations (ODE) to model the queue dynamics. We prove that choosing service rates on each transmission link that minimizes the quadratic sum of queue overload rates leads to the most balanced overload. Based on this result, we propose a queue-based policy combining maxweight scheduling with backpressure, which can asymptotically achieve the most balanced overload agnostic of packet arrival rate and capacity information. The proof technique is based on a novel characterization of the policy in a differentiable form, which is of independent interest. We further propose a distributed version of the policy, which reduces overhead by an order of magnitude. We evaluate our proposed policies under single-hop network and their concatenation into Clos structure, under randomly selected packet arrival rates, link capacities, and buffer sizes. Results demonstrate that our proposed policy converges to the most balanced overload in all cases, and the distributed version is nearly optimal.

3.1 Motivating Example

We explain the motivation of redesigning transmission policies for balancing the overload in networks with bounded buffers. Consider the 3×1 switched network in Fig. 3-1. According to [3], if the egress node d has unbounded buffer, the backpressure policy[1] guarantees that in the steady state, the queue overload rates in all 4 nodes are the same, which is most balanced. However, if node d has bounded buffer, which means its buffer size is finite, then backpressure will fill up the buffer in the steady state, and lead to overload imbalance as explained in the figure caption. Therefore, figuring out effective transmission policies to balance the queue overload in bounded-buffer systems is nontrivial and practically significant.

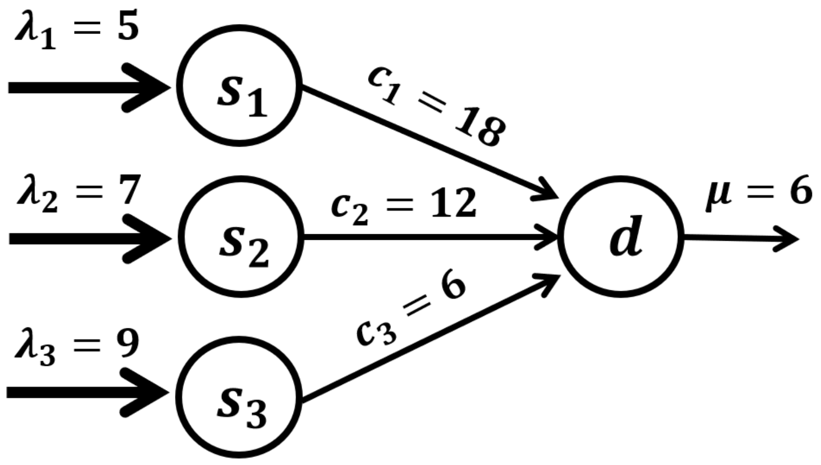


Figure 3-1: Suppose all ingress nodes have unbounded buffer. When node d has unbounded buffer, backpressure achieves most balanced overload rates where all 4 nodes grow with rate 3.75; When node d has finite buffer, then $\dot{q}_d = 0$ in the steady state due to buffer saturation, and backpressure achieves overload rates $(\dot{q}_{s_1}, \dot{q}_{s_2}, \dot{q}_{s_3}, \dot{q}_d) = (2, 5, 8, 0)$, deviating significantly from the most balanced one $(5, 5, 5, 0)$.

In this chapter, we propose a general analytical framework based on ordinary differential equations (ODE) to characterize queue dynamics, which we demonstrate can capture general buffer settings, and facilitate analytical results and policy design. We concentrate on single-hop networks modeled as a bipartite graph connecting ingress and egress nodes, each with a buffer to store incoming packets. This work can

[1]The definition of backpressure is deferred to Section 3.4 (eqn. (3.8)).

serve as the foundation for future work on multi-hop structures, such as datacenter network that are made up of multiple single-hop structures [17].

3.2 Model and Problem Formulation

3.2.1 Queue Dynamic Model

In this section, we introduce the ODE model for queue dynamics in single-hop network structure. We use a bipartite graph $(\mathcal{V}, \mathcal{E})$ to model the network, where $\mathcal{V} := \{\mathcal{V}_I, \mathcal{V}_E\}$ denotes the node set with $\mathcal{V}_I$ the set of ingress nodes, $\mathcal{V}_E$ the set of egress nodes, $\mathcal{E}$ the set of transmission links between $\mathcal{V}_I$ and $\mathcal{V}_E$, and $|\mathcal{V}_I| = N$ and $|\mathcal{V}_E| = M$. We term it as an $N \times M$ single-hop network. Denote the ith ingress node as s_i and the jth egress node as d_j. Each node k has a buffer that stores the packets, whose size is denoted by b_k. The packets in each node k form a queue, whose length at time t is denoted by $q_k(t)$. Therefore $q_k(t) \in [0, b_k], \forall k, \forall t$, which means that queue length cannot surpass the buffer size. We do not allow packet transmission to a saturated node. This is desirable in practice since it prevents packet dropping and significantly reduces the retransmission delay due to buffer overflow [2, 106], and can be implemented simply with a detection signal of the saturation level of downstream buffers. Packets will be backlogged until there exists spare buffer storage downstream.

Each packet arrives to one of the ingress nodes and departs from one of the egress nodes. The packet arrival rate at ingress node s_i, denoted by λ_i, represents the average number of packets that are injected into s_i in a time unit. We use $\boldsymbol{\lambda} := \{\lambda_i\}_{i=1}^N$ to denote the packet arrival rate vector. Packets in the buffer of s_i are transmitted to an adjacent egress node d_j through link $(s_i, d_j) \in \mathcal{E}$. The transmission rate on link (s_i, d_j) at time t, denoted by $g_{s_i d_j}(t)$, represents the number of packets transmitted over (s_i, d_j) in a time unit. Each link (s_i, d_j) is associated with a capacity value $c_{s_i d_j}$, which represents its maximum transmission rate. Specifically, $0 \leq g_{s_i d_j}(t) \leq c_{s_i d_j}$, $\forall (s_i, d_j) \in \mathcal{E}$. We use $\mathbf{c} := \{c_{s_i d_j}\}_{(s_i, d_j) \in \mathcal{E}}$ to denote the capacity vector. Finally, packets in an egress node d_j depart from the networks with departure rate denoted as $g_{d_j}(t)$, and the service rate of node d_j, which is the maximum departure rate for packets in d_j, is denoted by μ_j. Thus $g_{d_j}(t) \in [0, \mu_j]$, $\forall j = 1, \ldots, M$. Let $\boldsymbol{\mu} := \{\mu_j\}_{j=1}^{M}$.

Based on the above setting, we now formulate the queue dynamics according to the flow conservation law, which states that the net increase of queue length equals the difference between the number of new arrivals and departures at a node at any time. Specifically, for any ingress node s_i,

$$\dot{q}_{s_i}(t) = \lambda_i - \sum_{d_j : (s_i, d_j) \in \mathcal{E}} g_{s_i d_j}(t) \tag{3.1}$$

and for any egress node d_j,

$$\dot{q}_{d_j}(t) = \sum_{s_i : (s_i, d_j) \in \mathcal{E}} g_{s_i d_j}(t) - g_{d_j}(t). \tag{3.2}$$

$\dot{q}_k(t)$ denotes the *queue overload rate* of node k at time t, and we assume that $\dot{q}_k := \lim_{t \to \infty} \dot{q}_k(t)$ exists where $\dot{q}_k$ denotes node k's queue overload rate in steady state[2]. All the nodes thus have nonnegative queue overload rates in the steady state as the queue length is bounded below by 0. Furthermore, the queue length in nodes with bounded buffers will not grow with a positive rate in the steady state, therefore $\dot{q}_i = 0$ for any $i \in \mathcal{V}$ with bounded buffer.

In this work, we assume that internal (egress) buffers are bounded while ingress buffers are large enough to avoid saturation. In practice, internal nodes often have limited buffers [72,73]. For example, on-chip networks have very small internal buffers. Similarly, satellite networks have small buffers on-board the satellite. In contrast, ingress buffers have sufficient capacity to absorb bursty packet arrivals, e.g. in a satellite network the buffer at the ground terminal can be relatively large.

In reality, even ingress buffers have limited size, and packet loss will be inevitable when the packet arrival rate to an ingress node s_i is larger than the sum of the capacities of its downstream links. We can deal with bounded ingress buffers by

[2]The existence holds under most of the policies of interest [29, 102].

introducing a virtual queue for such s_i with unbounded buffer, whose length is q_{s_i} plus the number of dropped packets that are to be retransmitted, and thus the actual overload rate at s_i can be exactly characterized by the overload rate of a virtual queue with unbounded buffer. Therefore, we can assume without loss of generality that ingress buffers are unbounded.

We define $\dot{\mathbf{q}} := \{\dot{\mathbf{q}}_s, \dot{\mathbf{q}}_d\} \in \mathbb{R}^{N+M}$ as the *queue overload rate vector*, where $\dot{\mathbf{q}}_s = \{\dot{q}_{s_i}\}_{i=1}^{N} \in \mathbb{R}^N$ and $\dot{\mathbf{q}}_d = \{\dot{q}_{d_j}\}_{j=1}^{M} \in \mathbb{R}^M$ are the ingress and egress queue overload rate vector respectively[3]. Similarly, we define the *transmission rate vector* of the system as $\mathbf{g} := \big\{\{g_{s_i d_j}\}_{(s_i,d_j)\in\mathcal{E}}, \{g_{d_j}\}_{j=1}^{M}\big\}$. We further define the feasible flow region $\mathcal{G}$ as the set of transmission rate vectors $\mathbf{g}$ that satisfy the flow conservation laws (3.1) and (3.2) and capacity constraints. We then define the feasible queue overload rate region $\mathcal{R}$ as the set of queue overload rate vectors $\dot{\mathbf{q}}$ that can be achieved under some element in $\mathcal{G}$.

Remark: In (3.1) and (3.2), the queue length can be fractional. This is a fluid approximation to the real case where packets are discrete, which offers a simplified framework for studying flow control [3]. This fluid approximation is different from the fluid model defined in some prior works which captures the scaled limit of the queue backlog [4,76,77], an indicator for queue stability but not suited to study finite buffers and queue overload dynamics.

3.2.2 Problem Formulation: Overload Balancing

In this section we define the problem of overload balancing. The queue overload rate vector $\dot{\mathbf{q}}$ indicates the severity of queue overload. We need a metric to evaluate how balanced $\dot{\mathbf{q}}$ is. Multiple metrics related to network fairness have been investigated. The very first concept is *min-max fairness* which aims to identify a transmission rate vector such that any other vector that decreases the overload rate at some nodes must be at the expense of increasing the overload rate of some other nodes with a higher overload rate [70]. This concept stems from *max-min fairness* [69] which maximizes

[3]We neglect time t in the notations for brevity. We will clarify explicitly when notations without t represents steady state value to avoid ambiguity.

the minimum commodity flow to be transmitted, while in overload balancing the direction is reversed as reducing the max queue overload is desired.

Moreover, the min-max fairness solution is closely related to the lexicographic minimum solution [3] defined as follows.

Definition 3.1. *The queue overload vector* $\dot{\mathbf{q}}^*$ *is the lexicographic minimum in the feasible overload rate region* $\mathcal{R}$ *if and only if for* $\forall \dot{\mathbf{q}} \in \mathcal{R}$ *that* $\dot{\mathbf{q}} \neq \dot{\mathbf{q}}^*$, $\sum_{i=1}^{k} \dot{q}^*_{(i)} \leq \sum_{i=1}^{k} \dot{q}_{(i)}$, $\forall k$, *where* $\dot{q}^*_{(i)}$, $\dot{q}_{(i)}$ *denote the* i*th maximal element of* $\dot{\mathbf{q}}^*$, $\dot{\mathbf{q}}$ *respectively.*

The lexicographic minimum $\dot{\mathbf{q}}^*$ represents the most balanced overload vector since it guarantees that the top-k most severely overloaded nodes have been balanced under the metric of the sum of queue overload rates for every k. Therefore the overload balancing problem can be formally stated as: *determine the transmission rate vector* $\mathbf{g}$ *so that the resulting overload rate vector is the lexicographic minimum.*

Nevertheless, the lexicographical minimum is hard to formulate as it involves the ordering of a vector with cumulative sum comparisons. To overcome the challenge, we prove that minimizing the quadratic sum of queue overload rates serves as an equivalent criterion to lexicographic minimum under network flow constraints, which facilitates analysis of overload balancing, as shown in following sections.

3.3 Quadratic Sum Minimization Leads to Lexicographic Minimum

In this section, we prove that identifying $\mathbf{g} \in \mathcal{G}$ to minimize the quadratic sum of queue overload rates leads to lexicographic minimum overload rates. This result is derived under the prior information of $(\boldsymbol{\lambda}, \mathbf{c}, \boldsymbol{\mu})$, and it can capture the most balanced solution at any time shot when $(\boldsymbol{\lambda}, \mathbf{c}, \boldsymbol{\mu})$ is obtained. Analysis of policies without such prior information in following sections is based on it. This result can be directly proved by Cauchy-Schwarz inequality if there are no constraints on $\mathbf{g}$, however there is no general result (and it generally does not hold) when $\mathbf{g}$ is constrained. We, for the first time, prove the result under general single-hop network with bounded buffers.

We first introduce an intermediate result that the minimizer of the quadratic sum minimizes the maximum overload rate, and then show the main result.

3.3.1 Quadratic Sum Minimization to Maximum Overload Rate Minimization

The quadratic sum minimization framework of overload balancing under an $N \times M$ single-hop network can be formulated as

$$\begin{aligned} \min_{\mathbf{g}} \quad & \frac{1}{2}\sum_{i=1}^{N}\left(\lambda_i - \sum_{j=1}^{M} g_{s_i d_j}\right)^2 + \frac{1}{2}\sum_{j=1}^{M}\left(\sum_{i=1}^{N} g_{s_i d_j} - g_{d_j}\right)^2 \\ \text{s.t.} \quad & \sum_{i=1}^{N} g_{s_i d_j} = g_{d_j},\ \forall d_j \in \mathcal{B} \\ & 0 \le g_{s_i d_j} \le c_{s_i d_j},\ \forall (s_i, d_j) \in \mathcal{E} \\ & 0 \le g_{d_j} \le \mu_j,\ \forall j = 1, \ldots, M \end{aligned} \tag{3.3}$$

where the objective represents the quadratic sum of queue overload rates at all ingress and egress nodes according to (3.1) and (3.2), the variables $\mathbf{g}$ can represent the transmission rate vector at any specific time shot t, with $\mathcal{B}$ denoting the nodes whose buffer has been saturated at this time shot. The constraint $\sum_{i=1}^{N} g_{s_i d_j} = g_{d_j}$ means that $\dot{q}_{d_j} = 0$ for an egress node d_j saturated at this time shot. It is trivial to verify that the optimum can never be achieved when $\exists i \notin \mathcal{B}, \dot{q}_i < 0$, since the objective function is a quadratic sum of $\dot{\mathbf{q}}$. This property enables using (3.3) to study the steady state $(t \to \infty)$, since we require $\dot{q}_i \ge 0,\ \forall i$ in steady state mentioned in Section 3.2.1.

The minimization of maximum queue overload rate corresponds to the objective function $\min_{\mathbf{g} \in \mathcal{G}} \max_{i \in \mathcal{V}} \dot{q}_i$. This can be equivalently formulated as a linear programming problem

$$\begin{aligned}
\min_{\mathbf{g},v} \quad & v \\
\text{s.t.} \quad & \sum_{i=1}^{N} g_{s_i d_j} = g_{d_j},\ \forall d_j \in \mathcal{B} \\
& 0 \le g_{s_i d_j} \le c_{s_i d_j},\ \forall (s_i, d_j) \in \mathcal{E} \\
& 0 \le g_{d_j} \le \mu_j,\ \forall j = 1, \ldots, M \\
& \lambda_i - \sum_{j=1}^{M} g_{s_i d_j} \le v,\ \forall s_i \in \mathcal{V}_I \\
& \sum_{i=1}^{N} g_{s_i d_j} - g_{d_j} \le v,\ \forall d_j \in \mathcal{V}_E
\end{aligned} \tag{3.4}$$

by introducing an auxiliary variable v and additional constraints $\dot{q}_i \le v,\ \forall i \in \mathcal{V}$.

Now we state the intermediate result as Lemma 3.1.

Lemma 3.1. *Suppose that* $\mathbf{g}^* \in \mathcal{G}$ *is optimal in* (3.3)*, then* $\mathbf{g}^*$ *is optimal in* (3.4)*.*

The main proof idea is to take advantage of the Karush-Kuhn-Tucker (KKT) conditions of (3.3) and (3.4). Details are deferred to Section 3.7. We present a geometric interpretation of Lemma 3.1 in Fig. 3-2 through a 2×1 single-hop networks. Lemma 3.1 states that the the minimizer of (3.3) always overlaps with a minimizer of (3.4) in the feasible flow region $\mathcal{G}$ under the constraints.

3.3.2 Quadratic Sum Minimization to Lexicographic Minimum

We now demonstrate the main result in Theorem 3.1. The idea is that by Lemma 3.1, $\mathbf{g}^*$ minimizes the maximum queue overload rate, then we can show it must minimize the second maximum queue overload among all $\mathbf{g} \in \mathcal{G}$ that minimizes the maximum queue overload, otherwise it violates Lemma 3.1. Then iteratively we can obtain lexicographical minimum.

Theorem 3.1. *Suppose that* $\mathbf{g}^* \in \mathcal{G}$ *is optimal in* (3.3)*, then it is lexicographic minimum in* $\mathcal{G}$*.*

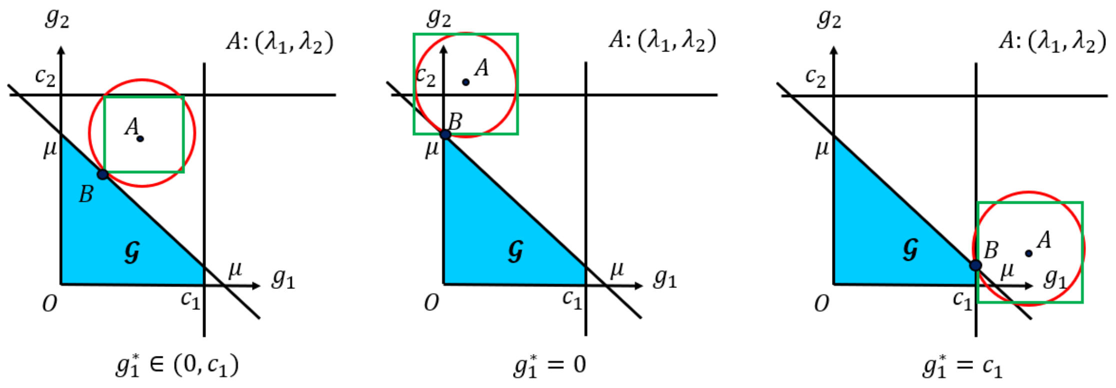


Figure 3-2: Geometric interpretation of Lemma 3.1 through a 2×1 single-hop network. The contour curves of (3.3) in red and (3.4) in green coincide at the same optimal point B on the boundary of $\mathcal{G}$ under different arrival rate vectors $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)$, denoted as point A where $\lambda_1 + \lambda_2 > \mu$.

Proof. (sketch) Note that $\mathbf{g}^*$, the minimizer of quadratic sum $\sum_{i=1}^{n} \dot{q}_i^2$ in (3.3) is the minimizer of max growth rate $\dot{q}_{(1)}$ in (3.4). Denote the minimum $\dot{q}_{(1)}$ as $\dot{q}_{(1)}^{\triangle}$, then the set $\mathcal{G}_1 = \{\mathbf{g} \in \mathcal{G} \mid \dot{q}_{(1)}^{(\mathbf{g})} = \dot{q}_{(1)}^{\triangle}\}$ contains $\mathbf{g}^*$, where $\dot{q}_{(1)}^{(\mathbf{g})}$ denotes the maximum queue overload rate under the transmission rate vector $\mathbf{g}$. Now we consider (3.3) and (3.4) with additional constraint that $\mathbf{g} \in \mathcal{G}_1$. Obviously $\mathcal{G}_1$ is a convex set, thus (3.4) with $\mathbf{g} \in \mathcal{G}_1$ is still convex. Meanwhile, with additional constraint that $\mathbf{g} \in \mathcal{G}_1$, (3.3) keeps in the form of a quadratic optimization problem. Therefore we can apply Lemma 3.1 to (3.3) and (3.4) in $\mathcal{G}_1$ similarly to obtain that the $\mathbf{g}^*$ minimizes $\dot{q}_{(2)}$, the second largest queue overload rate. Denote the minimum $\dot{q}_{(2)}$ as $\dot{q}_{(2)}^{\triangle}$, thus $\mathbf{g}^* \in \mathcal{G}_2 := \{\mathbf{g} \in \mathcal{G} \mid \dot{q}_{(1)}^{(\mathbf{g})} = \dot{q}_{(1)}^{\triangle}, \dot{q}_{(2)}^{(\mathbf{g})} = \dot{q}_{(2)}^{\triangle}\}$. Iteratively, $\mathbf{g}^* \in \mathcal{G}_{N+M}$ where any element in $\mathcal{G}_{N+M}$ induces the lexicographic minimum queue overload rate vector. $\square$

3.4 Maxweight + Backpressure Leads to Most Balanced Overload

Section 3.3 demonstrates that solving (3.3) can achieve most balanced overload. However, it requires the complete knowledge of network parameters $(\boldsymbol{\lambda}, \mathbf{c}, \boldsymbol{\mu})$, which in real networks may not be available [102]. In practice, the queue backlog $\mathbf{q}(t)$ is often accessible in real-time, thus we consider if there exists any *queue-based* transmission policy, which determines the transmission rate vector $\mathbf{g}(t)$ based on $\mathbf{q}(t)$, that can achieve most balanced overload as (3.3) does.

The ODE dynamical system (3.1) and (3.2) under queue-based policy form an autonomous system

$$\begin{cases} \dot{q}_{s_i}(t) = \lambda_i - \sum_{d_j:(s_i,d_j)\in\mathcal{E}} g_{s_i d_j}(\mathbf{q}(t)),\ \forall i = 1,\dots,N \\ \dot{q}_{d_j}(t) = \sum_{s_i:(s_i,d_j)\in\mathcal{E}} g_{s_i d_j}(\mathbf{q}(t)) - g_{d_j}(\mathbf{q}(t)),\ \forall j \in 1,\dots,M \end{cases} \tag{3.5}$$

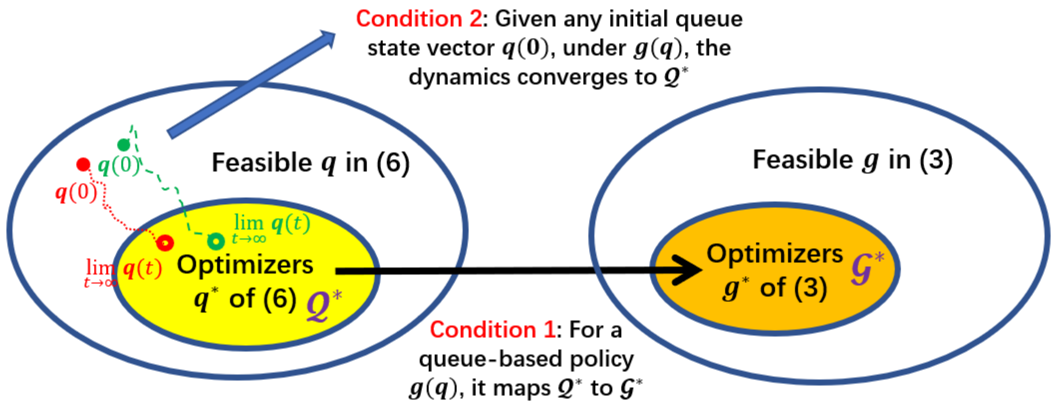
Due to the absence of prior knowledge of $(\boldsymbol{\lambda}, \mathbf{c}, \boldsymbol{\mu})$, we can no longer achieve most balanced overload in one stroke by optimizing (3.3). Instead, we aim to propose queue-based policies that can render the queue dynamics (3.5) to converge to the most balanced overload state. Formally, the problem of overload balancing under queue-based policy can be formulated as: *Is there* $\mathbf{g}(\mathbf{q})$ *that guarantees* $\lim_{t\to\infty} \dot{\mathbf{q}}(t) = \dot{\mathbf{q}}^*$ *in* (3.5)*, where* $\dot{\mathbf{q}}^*$ *is the overload rate vector induced by* $\mathbf{g}^*$*, the optimal solution to* (3.3) *with the oracle* $(\boldsymbol{\lambda}, \mathbf{c}, \boldsymbol{\mu})$*?* We prove that a *maxweight + backpressure (mw+bp)* queue-based policy yields a solution under $N \times M$ single-hop structure, and propose a distributed version of this policy that reduces communication overhead from $O(N)$ to $O(1)$, with performance close to optimum shown in Section 3.5.

3.4.1 Methodology

Our methodology to prove that a queue-based policy $\mathbf{g}(\mathbf{q})$ achieves the most balanced overload is to verify that it satisfies two conditions which together, as illustrated in Fig. 3-3, is a sufficient condition. For the first condition, we establish the **existence** of a queue vector $\mathbf{q}$ such that the transmission rate vector under the policy at $\mathbf{q}$ is an optimizer of (3.3) which leads to most balanced queue overload. Specifically, we consider the following optimization framework in which the only difference with (3.3) is that the queue vector $\mathbf{q}$ is the decision variable.

$$
\begin{aligned}
\min_{\mathbf{q}} \quad & \frac{1}{2}\sum_{i=1}^{N}\left(\lambda_i - \sum_{j=1}^{M} g_{s_id_j}(\mathbf{q})\right)^2 + \frac{1}{2}\sum_{j=1}^{M}\left(\sum_{i=1}^{N} g_{s_id_j}(\mathbf{q}) - g_{d_j}(\mathbf{q})\right)^2 \\
\text{s.t.} \quad & \sum_{i=1}^{N} g_{s_id_j}(\mathbf{q}) = g_{d_j}(\mathbf{q}), \ \forall d_j \in \mathcal{B} \\
& 0 \le g_{s_id_j}(\mathbf{q}) \le c_{s_id_j}, \ \forall (s_i, d_j) \in \mathcal{E} \\
& 0 \le g_{d_j}(\mathbf{q}) \le \mu_j, \ \forall j = 1, \ldots, M
\end{aligned}
\tag{3.6}
$$

Denote an optimizer of (3.6) as $\mathbf{q}^*$ and the set of all optimizers of (3.6) as $\mathcal{Q}^*$. The first condition is to verify that the policy $\mathbf{g}(\mathbf{q})$ satisfies $\forall \mathbf{q}^* \in \mathcal{Q}^*$, $\mathbf{g}(\mathbf{q}^*)$ equals some $\mathbf{g}^* \in \mathcal{G}^*$ where $\mathcal{G}^*$ denotes the set of optimizers of (3.3). For the second condition, we verify the **convergence** of the ODE dynamics (3.5) to the most balanced state under the policy $\mathbf{g}(\mathbf{q})$ given any initial queue vector. Namely, as $t \to \infty$, the second condition is to verify that the policy drives the queue vector to $\mathcal{Q}^*$. If the policy $\mathbf{g}(\mathbf{q})$ satisfies both conditions, it can achieve most balanced overload in the steady state.



Condition 2: Given any initial queue state vector $q(0)$, under $g(q)$, the dynamics converges to Q^*
Feasible q in (6)
$q(0)$
$q(0)$
$\lim_{t\to\infty} q(t)$
$\lim_{t\to\infty} q(t)$
Optimizers q^* of (6) Q^*
Feasible g in (3)
Optimizers g^* of (3) G^*
Condition 1: For a queue-based policy $g(q)$, it maps Q^* to G^*

Figure 3-3: Condition 1 (existence) and 2 (convergence) to verify that a queue-based policy achieves most balanced overload

3.4.2 Maxweight + Backpressure Policy in Differentiable Form

The ODE-based methodology in Section 3.4.1 requires the queue-based policy $\mathbf{g}(\mathbf{q})$ in a differentiable form. We now define the mw+bp policy accordingly. The policy contains two parts: maxweight scheduling and a backpressure mechanism. The idea of maxweight scheduling is to serve input nodes that have longer queue backlogs with higher priority [77]. The backpressure mechanism determines to transmit packets over a link (s_i, d_j) at t with rate $c_{s_i d_j}$ if $q_{s_i}(t) > q_{d_j}(t)$, and does not transmit otherwise [3]. To avoid buffer overflow, backpressure also ensures that packets are not served to a saturated node.

Both maxweight and backpressure were proposed in discrete forms originally. To embed them in the ODE framework, we propose the following differentiable characterization which can well approximate their original version.

Maxweight Scheduling: The maxweight scheduling in differentiable form is defined as

$$g_{s_i d_j}(\mathbf{q}) = c_{s_i d_j} \frac{e^{\gamma q_{s_i}}}{\sum_{k=1}^{N} e^{\gamma q_{s_k}}}, \ \forall (s_i, d_j) \in \mathcal{E} \tag{3.7}$$

where $\gamma > 0$ is a parameter. The larger γ is, the more we favor to serve ingress nodes with longer queue length. An extreme case is $\gamma \to \infty$, which matches to the serve-the-longest-queue policy [107]: only the ingress node with longest queue length will be served, and if there are K ingress nodes that have the same longest queue length, then (3.7) guarantees that each of these K nodes, say node s_i, will be served with rate $c_{s_i d}/K$. This corresponds to the result under serve-the-longest-queue policy in expectation, in which one of these K nodes is chosen uniformly at random to be served.

Backpressure Mechanism:

$$g_{s_i d_j}(\mathbf{q}) = c_{s_i d_j} \alpha_{s_i d_j} \beta_{d_j}, \ \forall (s_i, d_j) \in \mathcal{E} \tag{3.8}$$

where

$$\alpha_{s_i d_j} = \frac{1}{1 + e^{-a(q_{s_i} - q_{d_j})}}, \quad \beta_{d_j} := \frac{1}{1 + e^{-a(b_{d_j} - q_{d_j} - \epsilon)}}$$

and $a > 0$ and $\epsilon > 0$ are preset values. Note that if $a \to \infty$ and ϵ is close enough to 0, then the term $\alpha_{s_i d_j} = 1$ if $q_{s_i} > q_{d_j}$ and $\alpha_{s_i d_j} = 0$ if $q_{s_i} < q_{d_j}$; the term $\beta_{d_j} \to 1$ if $q_{d_j} < b_{d_j}$ and $\beta_{d_j} \to 0$ if $q_{d_j} \to b_{d_j}$. Therefore the policy (3.8) transmits the packets from an ingress node s_i to an egress node d_j with maximum service rate $c_{s_i d_j}$ if and only if the queue length in s_i is greater than in d_j, and meanwhile the buffer of node d is not saturated, which shows that (3.8) is an approximation to backpressure under sufficiently large a and small ϵ.

Maxweight + Backpressure (mw+bp): Combining (3.7) and (3.8), we can formulate the mw+bp policy as

$$g_{s_i d_j}(\mathbf{q}) = c_{s_i d_j} \alpha_{s_i d_j} \beta_{d_j} \frac{e^{\gamma q_{s_i}}}{\sum_{k=1}^{N} e^{\gamma q_{s_k}}}, \ \forall (s_i, d_j) \in \mathcal{E} \tag{3.9}$$

In (3.9), a transmission link is activated if and only if its corresponding ingress queue length satisfies the link activation requirements under both maxweight and backpressure.

In addition, egress nodes operate in a work-conserving manner, where each egress node d_j serves packets with rate μ_j whenever the buffer is nonempty. This guarantees that the egress nodes reduce the queue overload at the maximum rates. This work-conserving policy can also be formulated into a differentiable form as

$$g_{d_j}(\mathbf{q}) = \mu_j \frac{1}{1 + e^{-a(q_{d_j} - \epsilon)}}, \forall j \in 1, \dots, M \tag{3.10}$$

under sufficiently large a and $\epsilon \to 0$.

3.4.3 MW+BP in Single-hop Networks with Sufficient Capacity

In this part, we prove that the mw+bp policy (3.9) can achieve most balanced overload as by optimizing (3.3) if every transmission link has sufficient capacity, and all egress nodes run (3.10), stated in Theorem 3.2.

Theorem 3.2. *The queue dynamics under* (3.9) *and* (3.10) *converges to the most balanced overloading if* $c_{s_i d_j} > \mu_j, \ \forall (s_i, d_j) \in \mathcal{E}, \ j = 1, \dots, M$.

The proof idea is to verify the existence and convergence of most balanced state. More details are deferred to Section 3.7. Implementing (3.10) clearly makes for overload mitigation as all egress nodes do their best to send packets out. The intuition why (3.9) achieves most balanced overload is three-fold: (i) The maxweight (3.7) balances the ingress nodes as it favors serving queues with longer length; (ii) The backpressure (3.8) balances any connected ingress s_i and egress d_j as it sets the threshold for transmission decision at $q_{s_i} = q_{d_j}$; (iii) The condition $c_{s_i d_j} > \mu_j, \ \forall (s_i, d_j) \in \mathcal{E}$ guarantees that mw+bp can achieve maximum throughput since it makes any egress node d_j never be idle. The sufficient link capacity generally holds in data center networks and server farms which have sufficient transmission resources to guarantee high quality-of-service requirements [29].

3.4.4 MW+BP in Single-hop Networks with Limited Capacity

Next we consider limited capacity where $c_{s_i d_j} > \mu_j, \ \forall (s_i, d_j) \in \mathcal{E}$ does not hold. In this case, mw+bp policy (3.9) may not achieve most balanced overload since the maximum throughput may not be achieved, compared with sufficient capacity case in Section 3.4.3. We show an example in Fig. 3-4. Our solution is to consider a generalized version of mw+bp, where we run (3.9) and in the meantime additionally serve the ingress nodes with longest queues in order until maximum throughput is achieved. A special case is to consider $\gamma \to \infty$ in (3.9) so that the whole mechanism is exactly an extended version of serving-the-longest-queue policy.

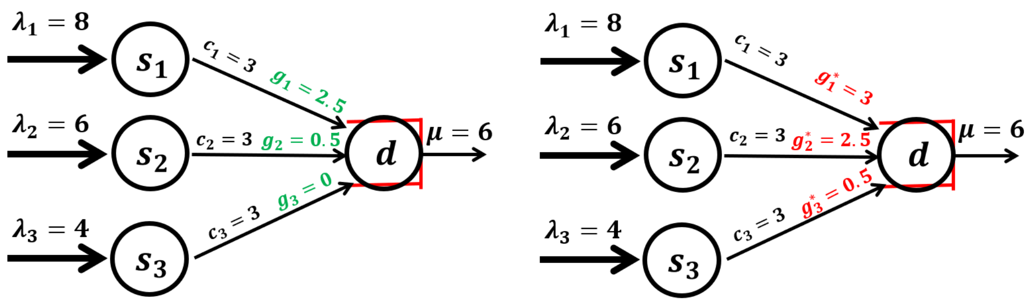


Figure 3-4: Example that (3.9) does not achieve most balanced overload with limited capacity with $\gamma \to \infty$, where node d has bounded buffer. On the left, $(g_1, g_2, g_3) = (2.5, 0.5, 0)$ denotes the transmission rates under (3.9) in steady state, under which $\dot{\mathbf{q}} = [5.5, 5.5, 4]$. It is not the most balanced overload as presented on the right, where $\dot{\mathbf{q}}^* = [5, 3.5, 3.5]$ under $(g_1^*, g_2^*, g_3^*) = (3, 2.5, 0.5)$, the optimal solution to (3.3). Other γ values also suffer from the suboptimality.

We summarize our solution in Algorithm 3.1. Algorithm 3.1 is a real-time algorithm that determines the service rates on every link given the current queue information $\mathbf{q}(t)$. We run (3.9) and calculate the remaining capacity on each link. Then we sort the ingress nodes in non-increasing order, and further follow the order to inject packets to egress nodes whose packet injection rate is lower than its service rate. Algorithm 3.1 presents our solution in a quantified way, which uses the information of $\mathbf{c}$ and $\boldsymbol{\mu}$ that may not be available. However, in practical implementation we do not require them: (i) We can sense if a link has been served with full capacity. If so, we do not inject more packets through this link. This replaces the need of calculation in line 3 and 8; (ii) Each egress node d_j can send the information of whether the queue length is increasing to ingress nodes through broadcasting or a controller, serving as an alternative indicator of $r_{d_j} < \mu_j$ in line 7 and replaces the need of calculation in line 4 and 9.

Algorithm 3.1: Generalized maxweight + backpressure with limited capacity

```
1  Input: current queue vector $\mathbf{q} := \mathbf{q}(t)$;
2  Run (3.9) and (3.10), and obtain $\mathbf{g}(\mathbf{q})$;
3  For all $(s_i, d_j) \in \mathcal{E}$, calculate the remaining capacity $\tilde{c}_{s_i d_j} := c_{s_i d_j} - g_{s_i d_j}(\mathbf{q})$;
4  Calculate the packet injection rate to all $d_j \in \mathcal{V}_E$ as
     $r_{d_j} := \sum_{s_i:(s_i,d_j)\in\mathcal{E}} g_{s_i d_j}(\mathbf{q})$;
5  Sort the queue length of ingress nodes in non-increasing order
     $q_{s_{(1)}} \geq q_{s_{(2)}} \geq \cdots \geq q_{s_{(N)}}$, where $\{s_{(i)}\}_{i=1}^N$ is a permutation of $\{s_i\}_{i=1}^N$;
6  for $i = 1, \ldots, N$ do
7      for all $d_j$ that $r_{d_j} < \mu_j$ do
8          $g_{s_{(i)}d_j}(\mathbf{q}) \leftarrow g_{s_{(i)}d_j}(\mathbf{q}) + \min\{\tilde{c}_{s_{(i)}d_j}, \mu_j - r_{d_j}\}$;
9          $r_{d_j} \leftarrow r_{d_j} + \min\{\tilde{c}_{s_{(i)}d_j}, \mu_j - r_{d_j}\}$;
10 Return $\mathbf{g}(\mathbf{q})$ as the transmission policy;
```

We can show that the generalized maxweight scheduling leads to most balanced overloading.

Theorem 3.3. *The generalized mw+bp policy in Algorithm 3.1 achieves most balanced overloading.*

The proof idea is similar to the proof of Theorem 3.2 by expressing the generalized mw+bp policy into a differentiable form, and then verify the conditions 1 and 2 in Section 3.4.1. The intuition is that Algorithm 3.1 achieves maximum throughput, and is a combination of mw+bp (3.9) and the serving-the-longest-queue, both of which achieve most balanced overload once maximum throughput can be achieved. Due to space limitation, we omit the proof.

3.4.5 Practical Extensions

At the end, we discuss three more extensions of the results in this work: (i) distributed mw + bp; (ii) weighted overload balancing; (iii) systems with discrete packets. All of them extend the results to meet broader practical constraints and needs.

Distributed MW + BP

In mw+bp policy (3.9), collecting real-time queue information of all ingress nodes is required at each ingress node or through a centralized controller. This induces large communication overhead in large-scale networks. We consider a distributed version of (3.9) to reduce overhead. The idea is that each ingress node gets access to another r ingress nodes, and run (3.9) where the maxweight part only depends on the queue length of itself and these r ingress nodes. One extreme case is that $r = 1$, where each ingress node s_i has the information of s_{i+1} (s_N has the information of s_1). The ingress node s_i serves packets to egress nodes only if $q_{s_i} > q_{s_{i+1}}$ (for s_N the condition is $q_{s_N} > q_{s_1}$), thus no need of sharing queue information of ingress nodes other than s_{i+1} to s_i, which reduces the communication overhead at the ingress side from $N-1$ to 1 for each ingress node. The intuition this distributed mechanism works for overload balancing is that balancing the pairs (s_{i-1}, s_i) and (s_i, s_{i+1}) together indirectly balances the pair (s_{i-1}, s_{i+1}).

The above distributed mw + bp policy at any ingress s_i can be formalized into a differentiable form as

$$g_{s_i d_j}(\mathbf{q}) = c_{s_i d_j} \alpha_{s_i d_j} \beta_{d_j} \frac{e^{\gamma q_{s_i}}}{e^{\gamma q_{s_i}} + e^{\gamma q_{s_{i+1}}}} \tag{3.11}$$

where $\alpha_{s_i d_j}$ and β_{d_j} represents the backpressure terms as in (3.8). We show in Section 3.5 that this distributed variant, even under $r = 1$, is close to the optimum achieved by mw+bp (3.9) in a large portion of test cases, which serves as a promising alternative of (3.9) to reduce communication overhead with low performance sacrifice in practical implementation.

Weighted overload balancing

A more generalized problem is the weighted overload balancing problem, in which we require some nodes to have lower overload rates, for example bottlenecks and centroids, while other peripheral nodes whose overload does not affect majority of network transmission can have higher overload rates. A straightforward way to formulate the problem is to assign a weight w_i for each node according to practical need. The weighted queue overload rate of node i becomes $w_i \dot{q}_i$. The problem of weighted overload balancing can thus be formalized as figuring out transmission policies to achieve lexicographical minimum of $\mathbf{w} \odot \dot{\mathbf{q}} = \{w_i \dot{q}_i\}_{i \in \mathcal{V}}$.

It has been shown that in systems with unbounded buffers, a weighted backpressure policy can achieved the target result of weighted overload balancing [3]. In systems with bounded buffer, we can extend our result by changing the quadratic sum objective function corresponding to (3.3) to

$$\frac{1}{2} \sum_{i=1}^{N} w_{s_i} \left(\lambda_i - \sum_{j=1}^{M} g_{s_i d_j} \right)^2 + \frac{1}{2} \sum_{j=1}^{M} w_{d_j} \left(\sum_{i=1}^{N} g_{s_i d_j} - g_{d_j} \right)^2 \tag{3.12}$$

and show that figuring out optimal $\mathbf{g}$ that satisfies constraints in (3.3) achieves the weighted lexicographical minimum following the methodology of Lemma 3.1 and Theorem 3.1. We state the result as Lemma 3.2 and Theorem 3.4 without proof. Specifically, Lemma 3.2 can be explained geometrically through a 2×1 single-hop network similar as Fig. 3.1 with the circular and square contours replaced by ellipsoid and rectangular contours.

Lemma 3.2. *Suppose that* $\mathbf{g}^* \in \mathcal{G}$ *optimizes* (3.12)*, then* $\mathbf{g}^*$ *minimizes* $\max_{i \in \mathcal{V}} w_i \dot{q}_i$.

Theorem 3.4. *Suppose that* $\mathbf{g}^* \in \mathcal{G}$ *optimizes* (3.12)*, then* $\mathbf{g}^*$ *achieves weighted lexicographical minimum.*

For queue-based policy, following the idea of (3.9), we can similarly propose a weighted maxweight + backpressure policy. Consider a 2×1 single-hop system with egress node being bounded. Given a targeted weight vector $\mathbf{w} = [w_1, w_2]$ for $\dot{q}_{s_1}$ and $\dot{q}_{s_2}$, then the following policy achieves weighted most balanced overload under sufficient capacity.

$$\begin{cases} \mathbf{g}_{s_1 d} = c_{s_1 d} \beta_d \alpha_{s_1 d} \frac{e^{\gamma w_1 q_{s_1}}}{e^{\gamma w_1 q_{s_1}} + e^{\gamma w_2 q_{s_2}}} \\ \mathbf{g}_{s_2 d} = c_{s_2 d} \beta_d \alpha_{s_2 d} \frac{e^{\gamma w_2 q_{s_2}}}{e^{\gamma w_1 q_{s_1}} + e^{\gamma w_2 q_{s_2}}} \end{cases}$$

whose steady state is $w_1 \dot{q}_{s_1} = w_2 \dot{q}_{s_2}$. The case of $N \times M$ single-hop network can be similarly formulated.

Implementation in discrete systems

To harness the benefits of ODE stability analysis for overload balancing in systems with bounded node buffers, we set the queue length to be continuous. We point out that the idea of our result can be extended to the discrete packet transmission in real networks. The maxweight + backpressure policy can be implemented in a discrete manner naturally. Furthermore, the optimal solution $\mathbf{g}^*$ to (3.3) can serve as a baseline. For any $(i, j) \in \mathcal{E}$, an intuitive way to determine $g_{ij}(t)$, interpreted as the number of packets served through link (i, j) at t-th time unit in discrete implementation, is to compare the time-average $\hat{g}_{ij}(t) := \frac{1}{t} \sum_{s=1}^{t} g_{ij}(s)$ at any time t with g_{ij}^*. Suppose $\hat{g}_{ij}(t) < g_{ij}^*$, then activate (i, j); otherwise do not activate. In the long term, the solution will statistically converge to the optimal overload balancing.

3.5 Performance Evaluation

In this section, we verify our proposed policies and theories through experiments over (i) single-hop network: server farm, packet switch, etc.; (ii) tree-structured datacenter network, for example Clos structure [1]. Clos concatenates multiple stages of single-hop structures. Although not proved analytically, verification results over Clos structure below demonstrate the extendability of our proposed policies to multi-hop networks.

We evaluate three policies: (i) Pure backpressure (3.8) [3]; (ii) Centralized Maxweight + Backpressure ((3.9) and Algorithm 3.1); (iii) Distributed Maxweight + Backpressure under $r = 1$ (3.11). We evaluate their performance in overload balancing through measuring the gap between $\dot{\mathbf{q}}^*$, the optimal solution to (3.3) achieved with prior knowledge of $(\boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{c})$ in steady state, and $\dot{\mathbf{q}}^\pi$, the steady state queue overload rate vector under a particular queue-based policy π. Closer gap represents superior overload balancing performance. Specifically, we consider two gap ratio metrics: (i) Quadratic sum gap ratio: $||\dot{\mathbf{q}}^\pi||^2/||\dot{\mathbf{q}}^*||^2$ which is exactly the metric we postulate; (ii) Max overload rate gap ratio: $\max_{i\in\mathcal{V}} \dot{q}_i^\pi / \max_{i\in\mathcal{V}} \dot{q}_i^*$ which reflects particularly the balancing of the most severe overload. For both metrics, the closer to 1, the better π is. The first metric reflects more on overall balancing while the second fits into cases where maximum overload is more important.

To demonstrate the universality of our proposed policy, we evaluate it using (i) different $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, which represents different overload levels; (ii) different $\mathbf{c}$, which represents different service capacity; (iii) different buffer values $\mathbf{b}$, which represents different buffer settings, including the spatial distribution of sufficient and limited buffers. We consider multiple networks instances with randomly sampled values of the above parameters, and measure the empirical cumulative distribution function (CDF) of the two gap metrics.

As introduced at length below, we see that maxweight + backpressure achieves most balanced overload rates in steady state, far better than pure backpressure in both metrics, while the performance of distributed maxweight + backpressure approaches that of maxweight + backpressure.

3.5.1 Single-hop Networks

We evaluate on a 64×32 single-hop network with full connection between ingress and egress nodes, modeling real switched networks [1, 72]. We consider 200 randomly selected parameter settings: (i) The arrival rate to each ingress port is uniformly distributed in $[0, 4]$; (ii) The service rates of each egress port is uniformly distributed within $[0, 6]$; (iii) The capacity $c_{s_i d_j}$ for each ingress-egress pair (s_i, d_j) is uniformly distributed in $[0, 10]$; (iv) The buffer size for any ingress node is 10,000 so that it is never saturated during the simulation, and the buffer size for any egress node is 10,000 with probability 0.2 and uniformly distributed within $[30, 80]$ with probability 0.8. The initial queue length in each node is set to be uniformly distributed within $[0, \min(30, \text{buffer size})]$, so that no queues are overflow initially. The rationales behind the settings are: (i) and (ii) guarantee that the system is overloaded with high probability, as the expected sum of arrival rates is $2 \times 64 = 128$, 133% of the expected sum of egress service rates $3 \times 32 = 96$; (iii) considers both sufficient and limited capacity values; (iv) realizes different buffer settings, which follows the real case that buffers at ingress are large [2] while egress ports may have limited buffers [72, 73] generally.

We plot the CDFs of the quadratic sum gap ratio and max overload rate gap ratio of all 200 sampled settings in steady state in Fig. 3-5 and Fig. 3-6, where the x-axis is in logarithmic scale to make details clearer. Our proposed mw+bp (3.9) achieves quadratic sum gap close to 1 for nearly all test instances, and achieves max overload rate gap close to 1 for more than 70% instances. The optimality shown in the results is not 100% as in theory due to limited time span. The value difference between these two gaps is due to the balancing effect of quadratic sum[4]. The distributed version of mw + bp (3.11) loses some accuracy while still performs generally well, as in more than 85% instances, the quadratic sum overlaod gap is less than 1.15 and the max overload rate gap is less than 1.4, as pointed out in the figures. Comparatively, the backpressure policy incurs large gaps in the steady state and achieves the optimum in none of the cases. Typically, with more than 75% of instances the quadratic sum gap exceeds 1.4 and the max overload rate gap exceeds 1.5.

We further present the transient process of the quadratic sum gap ratio in Fig. 3-7

[4]Consider, for example, $\dot{\mathbf{q}}^* = [0.5, 0.5]$ and $\dot{\mathbf{q}}^\pi = [0.6, 0.4]$, then a gap of $0.6/0.5 = 1.2$ in the max overload rate only leads to a gap of $(0.6^2 + 0.4^2)/(0.5^2 + 0.5^2) = 1.04$ in quadratic sum, where the gap is smaller.

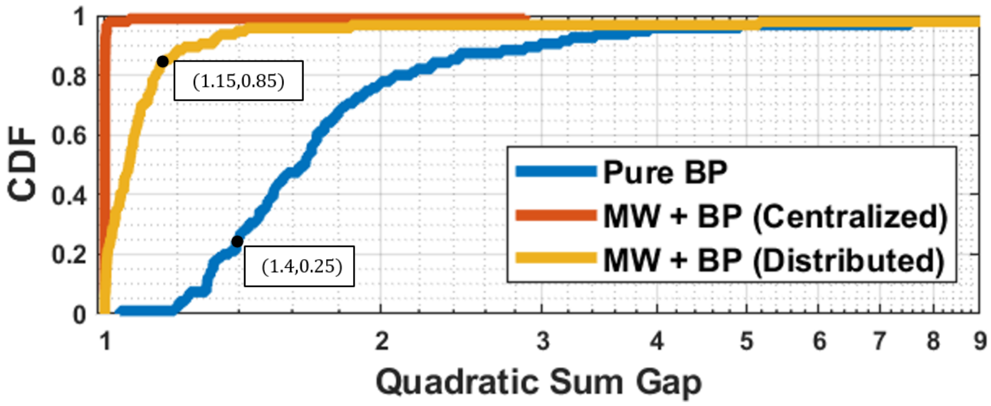


Figure 3-5: Quadratic Sum Gap Comparison in Steady State

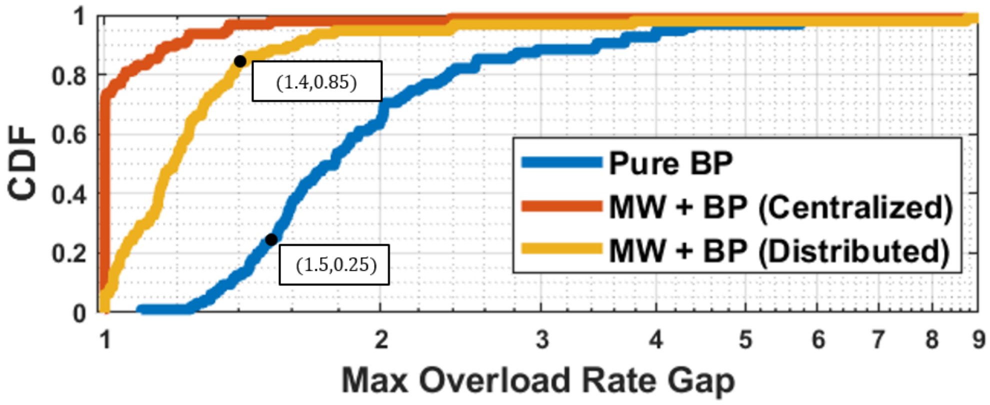


Figure 3-6: Max Overload Rate Gap Comparison in Steady State

under the ODE. Note that negative queue overload rate may exist in transient states, which is desirable for overload mitigation at a node. Therefore the quadratic sum gap ratio at any time in Fig. 3-7 only considers the sum of positive queue overload rates. At the beginning, the gap ratio is high for all three schemes, because the queue is far from the equilibrium point. For example, in pure backpressure, when all connected nodes satisfy backpressure constraints, then all intermediate links will be activated, thus all ingress nodes have negative overload rates while egress nodes have large positive rates due to the arriving packets from all upstream links. Approaching the steady state, shown in the zoomed-in subfigure, the gap of mw + bp, both centralized and distributed, converges close to 1, while the gap of pure backpressure converges a value greater than 1, not optimal solution for overload balancing.

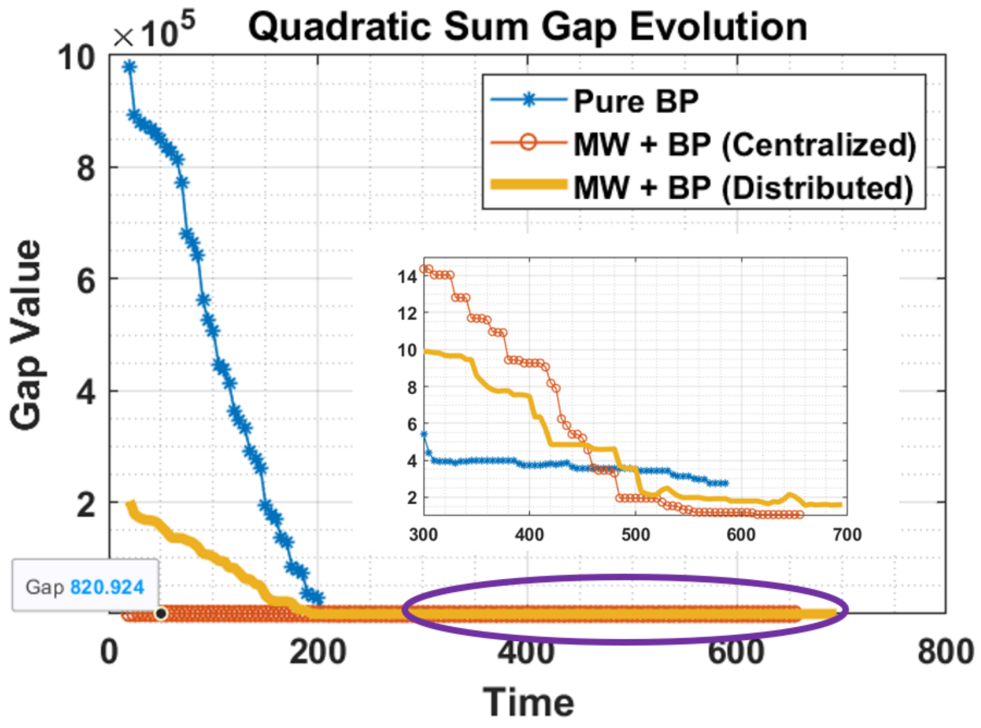


Figure 3-7: Transient Process of Quadratic Sum Gap

3.5.2 Clos Network Structure

We further test on a 3-stage Clos structure abstracted from Google's work on their Jupiter datacenter [1], shown in Fig. 3-8. It contains 24 ingress blocks at the top, fully connected to 12 aggregation blocks at the middle, and the aggregation blocks 1 to 4 are connected to 4 egress blocks, as are aggregation blocks 5 to 8. Packets depart from the network from the 8 egress blocks and aggregation blocks 9 to 12, which may have buffers with limited size. Similarly, we randomly take 200 different settings of parameters: Arrival rate is uniformly distributed within $[0, 12]$; Service rate of blocks where packet depart from the network is uniformly distributed within $[0, 12]$; Capacity values of different links, and buffer size setting at blocks from which packets depart are identical to the single-hop case in Section 3.5.1. The performance of the network policies are presented in Fig. 3-9 and Fig. 3-10. The results share a similar trend with the single-hop case.

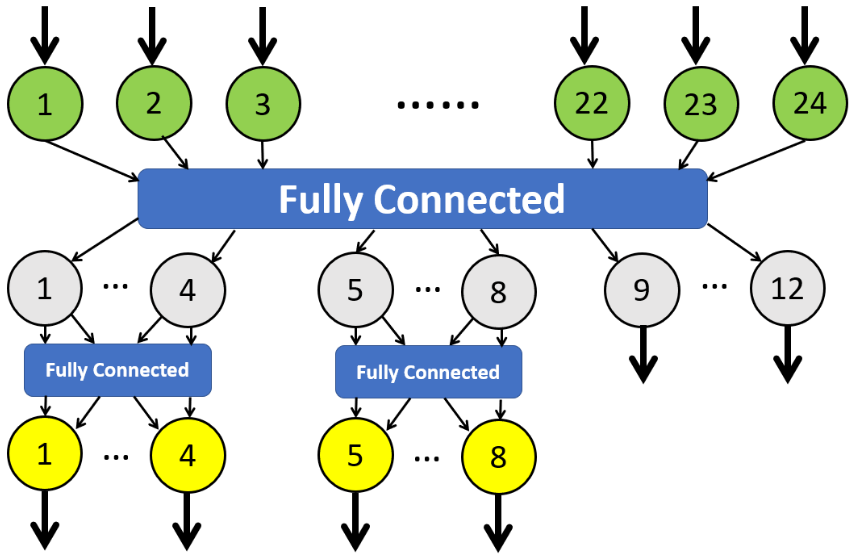


Figure 3-8: Example of a 3-stage Clos structure from [1]

3.6 Summary and Future Work

In this chapter, we study overload balancing in single-hop networks with bounded buffers. We show that bounded buffer affects the resulting policy to achieve most balanced overload. We leverage ordinary differential equations to model the queue dynamics in bounded buffer systems. We first prove that setting link service rates to minimize the quadratic sum of the queue overload rates leads to the lexicographic

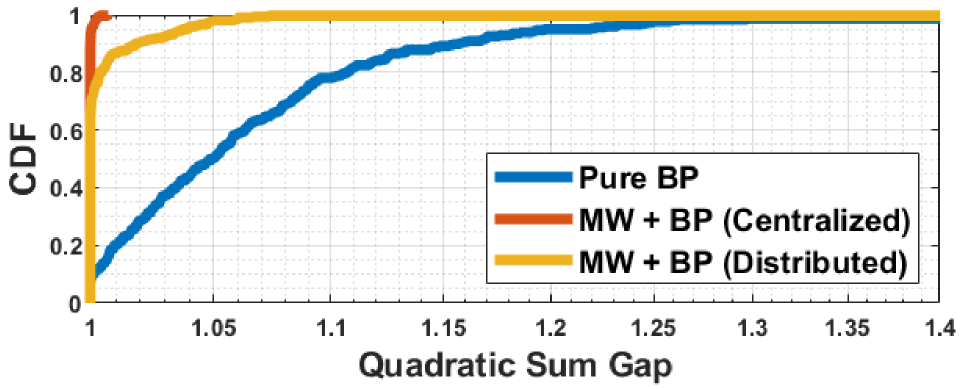


Figure 3-9: Quadratic Sum Gap Comparison (Clos)

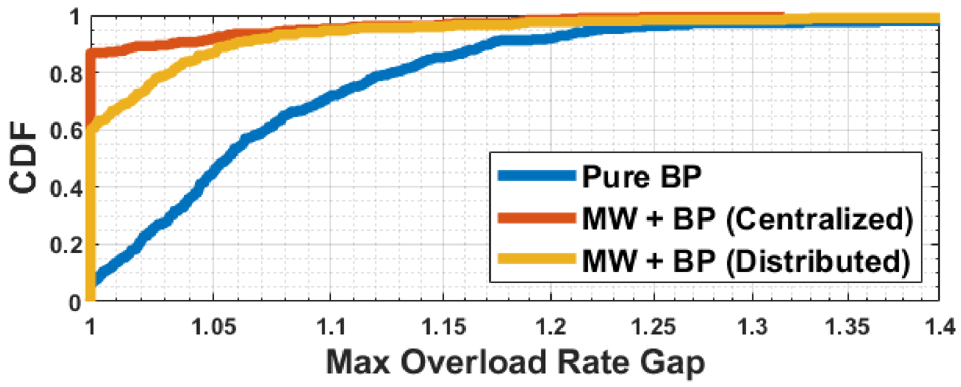


Figure 3-10: Max Overload Rate Gap Comparison (Clos)

minimum queue overload. Based on this result, we prove that a maxweight scheduling and backpressure policy asymptotically achieves most balanced overload, through a novel formulation of the policy in a differentiable form which may be of independent interest. We further propose a distributed maxweight + backpressure policy that can reduce communication overhead by one order of magnitude. We validate the performance of our proposed policies by simulation over single-hop structure and Clos networks under different packet arrival rates, link capacities, and buffer settings. Extension of the results in this work to multi-hop networks, and exploitation of the differential equation formulation for other network performance metrics, are promising future directions.

3.7 Chapter Appendix

3.7.1 Proof of Lemma 3.1

We term (3.3) as l_2 problem and (3.4) as l_∞ problem, since they respectively minimize the l_2 and l_∞ norm of $\dot{\mathbf{q}}$. The Lagrangian functions of (3.3) and (3.4) are[5]

- $\mathcal{L}^{(2)}(\mathbf{g}, \mathbf{a}, \mathbf{b}, h) = \frac{1}{2}\sum_{i\in\mathcal{V}}(\dot{q}_i)^2 + \sum_{(i,j)\in\mathcal{E}} a_{ij}(g_{ij} - c_{ij}) - \sum_{(i,j)\in\mathcal{E}} b_{ij}g_{ij} + \sum_{i\in\mathcal{B}} h_i\dot{q}_i$.
- $\mathcal{L}^{(\infty)}(\mathbf{g}, v, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \bar{h}) = v + \sum_{i\in\mathcal{V}} \gamma_i(\dot{q}_i - v) + \sum_{(i,j)\in\mathcal{E}} \alpha_{ij}(g_{ij} - c_{ij}) - \sum_{(i,j)\in\mathcal{E}} \beta_{ij}g_{ij} + \sum_{i\in\mathcal{B}} h_i\dot{q}_i$.

where $\dot{q}_i$ follows (3.1) and (3.2). Their KKT conditions are respectively

l_2 **problem:**

$$\begin{cases} -\dot{q}_i + \dot{q}_j + a_{ij} - b_{ij} - h_i + h_j = 0, \ \forall (i,j) \in \mathcal{E} \\ h_i\dot{q}_i = 0, \ \forall i \in \mathcal{B} \\ a_{ij}(g_{ij} - c_{ij}) = 0, \ a_{ij} \geq 0; \ b_{ij}g_{ij} = 0, \ b_{ij} \geq 0, \ \forall (i,j) \in \mathcal{E} \end{cases} \tag{3.13}$$

[5] $\{b_i\}_{i=1}^N$ in the proof are Lagrangian multipliers rather than buffer sizes.

l_∞ **problem:**

$$\begin{cases} 1 - \sum_{i \in \mathcal{V}} \gamma_i = 0 \\ -\gamma_i + \gamma_j + \alpha_{ij} - \beta_{ij} - \bar{h}_i + \bar{h}_j = 0, \ \forall (i,j) \in \mathcal{E} \\ \bar{h}_i \dot{q}_i = 0, \ \forall i \in \mathcal{B} \\ a_{ij}(g_{ij} - c_{ij}) = 0, \ a_{ij} \geq 0; \ b_{ij} g_{ij} = 0, \ b_{ij} \geq 0, \ \forall (i,j) \in \mathcal{E} \\ \gamma_i (\dot{q}_i - v) = 0, \ \gamma_i \geq 0, \ \forall i \in \mathcal{V} \end{cases} \tag{3.14}$$

Our proof idea is as follows. Denote an optimizer of the l_2 problem as $\mathbf{g}^{(2)}$, and it suffices to show that given $(\mathbf{g}^{(2)}, \mathbf{a}, \mathbf{b}, \mathbf{h})$ that satisfies (3.13), there exists $(\bar{\mathbf{g}}, v, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \bar{h})$ that satisfies (3.14), and $\mathbf{g}^{(2)} = \bar{\mathbf{g}}$. If this holds, then $\mathbf{g}^{(2)} = \bar{\mathbf{g}}$ minimizes (3.4) as the l_∞ problem is convex, thus satisfying KKT condition suffices to optimal solution.

To show this, suppose that under an optimizer of l_2 problem $\mathbf{g}^{(2)}$, inside the queue overload rate vector $\dot{\mathbf{q}}$ there are k maximum entries. Denote the set of these k elements as $\mathcal{K}$, then $\forall j \in \mathcal{V} \backslash \mathcal{K}$, we must have $\gamma_j = 0$ due to complementary slackness. Thus $\sum_{i \in \mathcal{K}} = 1$. We can set $v = \max_{\dot{q}_i}$ induced by $\mathbf{g}^{(2)}$. We set $\gamma_j = 1/|\mathcal{K}|, \ \forall i \in \mathcal{K}$ with the intuition that all these nodes share the same overload rate. Furthermore, note that the constraint $h_i \dot{q}_i = 0$ in l_2 problem and $\bar{h}_i \dot{q}_i = 0$ in l_∞ problem for $\forall i \in \mathcal{B}$, hence we can set $\bar{h}_i = h_i$. Now what remains is to figure out feasible $\alpha_{ij}, \beta_{ij}, \ \forall (i,j) \in \mathcal{E}$, and we show their existence under $3 \times 3 = 9$ combinations of i and j, as every node can be in one of the three subsets $\mathcal{K}$, $\mathcal{B}$, or $\mathcal{V} \backslash (\mathcal{K} \cup \mathcal{B})$, where $\mathcal{K} \cap \mathcal{B} = \emptyset$ in overloaded networks as $\dot{q}_j = 0, \ \forall i \in \mathcal{B}$. This can be validated with simple derivation.

3.7.2 Proof of Theorem 3.2

Condition 1: The Lagrangian function of (3.6) is

$$\mathcal{L}^{(q)}\left(\mathbf{q}, h^{(q)}, \mathbf{a}^{(q)}, \mathbf{b}^{(q)}\right) = \frac{1}{2}\sum_{i=1}^{N}\left(\lambda_i - g_{s_id}(\mathbf{q})\right)^2 + h^{(q)}\left(\sum_{i=1}^{N} g_{s_id}(\mathbf{q}) - \mu\right) + \sum_{i=1}^{N} a_i^{(q)}(g_{s_id}(\mathbf{q}) - c_{s_id}) - \sum_{i=1}^{N} b_i^{(q)} g_{s_id}(\mathbf{q}).$$

The KKT condition for (3.3) is (3.13) while the KKT condition for the queue-based framework (3.6) is that for $\forall i = 1, \ldots, N$,

$$\begin{cases} \frac{\partial \mathcal{L}^{(q)}}{\partial q_{s_i}} = -\sum_{j=1}^{N}(\lambda_j - g_{s_jd}(\mathbf{q}) - a_j^{(q)} + b_j^{(q)})\frac{\partial g_{s_jd}}{\partial q_{s_i}} = 0 \\ \frac{\partial \mathcal{L}^{(q)}}{\partial q_d} = -\sum_{j=1}^{N}(\lambda_j - g_{s_jd}(\mathbf{q}) - a_j^{(q)} + b_j^{(q)})\frac{\partial g_{s_jd}}{\partial q_d} = 0 \\ \frac{\partial \mathcal{L}^{(q)}}{\partial h^{(q)}} = \sum_{i=1}^{N} g_{s_id}(\mathbf{q}) - \mu = 0 \\ a_i^{(q)}(g_{s_id}(\mathbf{q}) - c_{s_id}) = 0, \ a_i^{(q)} \geq 0 \\ b_i^{(q)} g_{s_id}(\mathbf{q}) = 0, \ b_i^{(q)} \geq 0 \end{cases}$$

Under the mw+bp policy (3.9) and sufficient capacity, node d will keep saturated in the steady state, which means that $\sum_{i=1}^{N} g_{s_id}(\mathbf{q}) = \mu$ always holds in the steady state. Therefore we have $\sum_{j=1}^{N} \frac{\partial g_{s_jd}}{\partial q_{s_i}} = 0, \ \forall i = 1, \ldots, N$. We can then transform the first equation in the KKT conditions into

$$\frac{\partial \mathcal{L}^{(q)}}{\partial q_{s_i}} = -\sum_{j=1, j\neq i}^{N}(z_j - z_i)\frac{\partial g_{s_jd}}{\partial q_{s_i}} = 0, \ \forall i = 1, \ldots, N \tag{3.15}$$

where $z_i := \lambda_i - g_{s_id}(\mathbf{q}) - a_i^{(q)} + b_i^{(q)}$. Further, we note that under (3.9), for any $i \in \mathcal{V}_S$, $\frac{\partial g_{s_id}}{\partial q_{s_i}} > 0$, $\frac{\partial g_{s_jd}}{\partial q_{s_i}} < 0$, $\forall j \neq i$, which corresponds to the maxweight + backpressure policy that the service rate of a node will not be decreased whenever its queue gets longer. This indicates that $z_1 = z_2 = \cdots = z_N$, otherwise there must exist one of the

LHS's of (3.15) is strictly greater than 0. Then we have

$$\lambda_1 - g_{s_1 d}(\mathbf{q}) - a_1^{(q)} + b_1^{(q)} = \cdots = \lambda_N - g_{s_N d}(\mathbf{q}) - a_N^{(q)} + b_N^{(q)}. \tag{3.16}$$

Any optimizer $\mathbf{q}^*$ to (3.6) satisfies (3.16) under maxweight + backpressure policy denoted as $\mathbf{g}(\mathbf{q})$ in (3.9). Then we can assign $\tilde{\mathbf{g}} := \mathbf{g}(\mathbf{q})$, $\mathbf{a} := \mathbf{a}^{(q)}$, $\mathbf{b} := \mathbf{b}^{(q)}$, and $h := h^{(q)}$, such that $(\tilde{\mathbf{g}}, \mathbf{a}, \mathbf{b}, h)$ satisfies (3.13), the KKT condition of (3.3). Since (3.3) is a convex optimization problem, then $\tilde{\mathbf{g}} = \mathbf{g}(\mathbf{q}^*)$ is the optimizer of (3.3), which means $\mathbf{q}^*$ leads to most balanced overloading under maxweight + backpressure policy (3.9).

Condition 2: Now we prove that under the maxweight + backpressure policy $\mathbf{g}(\mathbf{q})$, $\mathbf{q}(t)$ converges into $\mathcal{Q}^*$ given any initial state $\mathbf{q}(0)$. Since $c_{s_i d} > \mu$, $\forall i = 1, \ldots, N$, we can categorize all N entries in $\mathbf{g}(\mathbf{q})$ at $\mathbf{q}^*$ into $\mathcal{A}^{(q)} = \{i \mid g_{s_i d}(\mathbf{q}^*) \in (0, c_{s_i d})\}, \mathcal{B}^{(q)} = \{i \mid g_{s_i d}(\mathbf{q}^*) = 0\}$, where we have shown that $\forall i, j \in \mathcal{A}, k \in \mathcal{B}$, $\dot{q}_{s_i} = \dot{q}_{s_j} \geq \dot{q}_{s_k} = \lambda_k$, where $\dot{q}_{s_p} = \lambda_p - g_{s_p d}(\mathbf{q}^*), p = i, j, k$ and we remove the trivial case that $\dot{q}_{s_i} = \dot{q}_{s_j} = \dot{q}_{s_k}$ which can be verified easily. Without loss of generality, we consider $\mathcal{A} := \{1, 2, \ldots, N_0\}$ and $\mathcal{B} = \{N_0 + 1, \ldots, N\}$. Then the most balanced overloading rates achieved by (3.3) satisfy that

$$\xi_1 \dot{q}_1 = \cdots = \xi_{N_0} \dot{q}_{N_0} = \xi_{N_0+1} \dot{q}_{N_0+1} = \cdots = \xi_N \dot{q}_N \tag{3.17}$$

for $\xi_1, \ldots, \xi_{N_0} = 1$ and some $\xi_{N_0+1}, \ldots, \xi_N > 1$.

Note that in overloaded networks, there exists $\dot{q}_{s_i}(t) \to \infty$, so there is no equilibrium point. To prove that the dynamics converge to the state (3.17), we introduce a series of auxiliary variables $\mathbf{x} := \{x_i\}_{i=1}^{N-1}$ where $x_i = \xi_i q_{s_i} - \xi_{i+1} q_{s_{i+1}}$, and transform the original $\mathbf{q}$-based ODE into $\mathbf{x}$-based ODE. Since $\mathbf{q} \in \mathbb{R}^N$ and $\mathbf{x} \in \mathbb{R}^{N-1}$, there will remain one $q_{s_{i_0}}$ in the $\mathbf{x}$-based ODE. As there exists a queue that grows to infinity, we let $q_{s_{i_0}}$ be this queue and thus let $q_{s_{i_0}} \to \infty$ in the $\mathbf{x}$-based ODE as we focus on the steady state. The reason of the transformation into $\mathbf{x}$-based ODE is that its equilibrium point $\mathbf{x}^*$, if there exists one, corresponds to the most balanced overloading. Namely, $\dot{\mathbf{x}} = 0$ leads to (3.17). Condition 1 guarantees the existence of

$\mathbf{x}^*$ under (3.9), so what remains is to prove any $\mathbf{x}^*$ is a stable equilibrium point.

For any input node s_j that $j \in \mathcal{B}$, it can be easily verified that the $\mathbf{x}$-based ODE under maxweight + backpressure policy (3.9) and $q_{s_{i_0}} \to \infty$ indicates $\dot{q}_{s_j} = \lambda_{s_j}$, which means that the queue dynamics of all input nodes corresponding to $\mathcal{B}$ converge to the most balanced rate values under (3.9). Thus $g_{s_j d}(\mathbf{q}) = 0$ in the steady state under (3.9) and all terms $x_{N_0}, \dots, x_{N-1}$ are eliminated from the $\mathbf{x}$-based ODE.

Therefore we only need to consider the set $\mathcal{A}$. Over $\mathcal{A}$, note that $x_i = q_{s_i} - q_{s_{i+1}}$, $\forall i = 1, \dots, N_0 - 1$. By the chain rule of partial derivatives, we have for any input node s_i,

$$\begin{cases} \frac{\partial g_{s_i d}}{\partial q_{s_1}} = -\frac{\partial g_{s_i d}}{\partial x_1}, \\ \frac{\partial g_{s_i d}}{\partial q_{s_k}} = -\frac{\partial g_{s_i d}}{\partial x_k} + \frac{\partial g_{s_i d}}{\partial x_{k-1}}, \ k = 2, \dots, N_0 - 1 \\ \frac{\partial g_{s_i d}}{\partial q_{s_{N_0}}} = \frac{\partial g_{s_i d}}{\partial x_{N_0 - 1}} \end{cases} \tag{3.18}$$

Equivalently, we can write it as $\frac{\partial g_{s_i d}}{\partial x_k} = -\sum_{j=1}^{k} \frac{\partial g_{s_i d}}{\partial q_{s_j}}$, $\forall k = 1, \dots, N_0 - 1$, and further express it as $\frac{\partial g_{s_i d}}{\partial \mathbf{x}} = -\mathbf{L}\frac{\partial g_{s_i d}}{\partial \mathbf{q}_{1:N_0-1}}$ where $\mathbf{L} \in \mathbb{R}^{(N_0-1)\times(N_0-1)}$ is a lower triangular matrix with $L_{ij} = 1$, $\forall i \geq j$, and $\frac{\partial g_{s_i d}}{\partial \mathbf{q}_{1:N_0-1}}$ denotes the derivative over q_{s_1} to $q_{s_{N_0-1}}$ where $q_{s_{N_0}}$ is neglected as it is redundant.

Now we connect the Jacobian $\mathbf{J}_x := \{J_{x,ij}\}_{i=1,\dots,N_0-1}^{j=1,\dots,N_0-1} \in \mathbb{R}^{(N_0-1)\times(N_0-1)}$ w.r.t. $\mathbf{x}$-based ODE with $\mathbf{J}_q := \{J_{q,ij}\}_{i=1,\dots,N_0}^{j=1,\dots,N_0} \in \mathbb{R}^{N_0 \times N_0}$ w.r.t. $\mathbf{q}$-based ODE. Note that $J_{x,ij} = \frac{\partial g_{s_i d}}{\partial x_j} - \frac{\partial g_{s_{i+1} d}}{\partial x_j}$ and $J_{q,ij} = -\frac{\partial g_{s_i d}}{\partial q_{s_j}}$. We can derive $\mathbf{J}_x = \mathbf{K}\mathbf{J}_{q,1:(N_0-1)}\mathbf{L}^T$, where $\mathbf{J}_{q,1:(N_0-1)}$ denotes the truncation of $\mathbf{J}_q$ by removing the N_0-th row and column, and

$$\mathbf{K} = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -1 \\ 1 & 1 & 1 & \cdots & 1 & 2 \end{bmatrix}.$$

The transformation uses $\sum_{i=1}^{N_0} \frac{\partial g_{s_i d}}{\partial q_{s_j}} = 0, \ \forall j = 1, \dots, N_0$.

Now our goal is to show the stability of $\mathbf{x}$-based ODE to guarantee the dynamics converge to its equilibrium point. It suffices to show all the eigenvalues of $\mathbf{J}_x$ have negative real parts at any equilibrium point. We need the following theorem.

Theorem 3.5. *[108] Let $\mathbf{A}, \mathbf{P} \in \mathbb{R}^{(N_0-1)\times(N_0-1)}$, where $\mathbf{P}$ is positive-definite. If $\mathbf{PA} + \mathbf{A}^T\mathbf{P}$ is negative definite, then all eigenvalues of $\mathbf{A}$ have negative real parts.*

This requires us to identify a $\mathbf{P}$ that is positive-definite. Interestingly, we identify that $\mathbf{P} = \mathbf{L}\mathbf{K}^{-1}$ serves as a solution, where we can calculate that $P_{ij} = j\frac{N_0-i}{N_0}$, $\forall i \geq j$ and $P_{ji} = P_{ij}$, $\forall i < j$. We can show $\mathbf{P}$ has the Cholesky decomposition: $\mathbf{P} := \mathbf{C}\mathbf{C}^T$, where $\mathbf{C}$ is lower triangular and

$$\begin{cases} C_{jj} = \sqrt{\frac{N_0-j}{N_0-(j-1)}},\ \forall j = 1,\ldots,N_0-1 \\ C_{ij} = \begin{cases} \frac{N_0-i}{N_0-(j-1)}\left(\sqrt{\frac{N_0-j}{N_0-(j-1)}}\right)^{-1} i > j \\ 0,\ i < j \end{cases} \end{cases}$$

Furthermore, since $\mathbf{P} = \mathbf{P}^T$, we have

$$\begin{aligned} \mathbf{P}\mathbf{J}_x + \mathbf{J}_x^T\mathbf{P} &= \mathbf{PKJ}_{q,1:(N_0-1)}\mathbf{L}^T + \left(\mathbf{KJ}_{q,1:(N_0-1)}\mathbf{L}^T\right)^T\mathbf{P}^T \\ &= \mathbf{LK}^{-1}\mathbf{KJ}_{q,1:(N_0-1)}\mathbf{L}^T + \mathbf{LJ}_{q,1:(N_0-1)}^T\mathbf{K}^T\left(\mathbf{K}^{-1}\right)^T\mathbf{L}^T \\ &= \mathbf{L}\left(\mathbf{J}_{q,1:(N_0-1)} + \mathbf{J}_{q,1:(N_0-1)}^T\right)\mathbf{L}^T \end{aligned}$$

Now it suffices to show that $\mathbf{J}_{q,1:(N_0-1)} + \mathbf{J}_{q,1:(N_0-1)}^T$ is negative definite, as once this holds, we have the Cholesky decomposition $\mathbf{J}_{q,1:(N_0-1)} + \mathbf{J}_{q,1:(N_0-1)}^T = -\mathbf{G}\mathbf{G}^T$ and thus $\mathbf{P}\mathbf{J}_x + \mathbf{J}_x^T\mathbf{P} = -\mathbf{L}\mathbf{G}\mathbf{G}^T\mathbf{L}^T$ which indicates that $\mathbf{P}\mathbf{J}_x + \mathbf{J}_x^T\mathbf{P}$ is negative definite, and thus $\mathbf{J}_{q,1:(N_0-1)} + \mathbf{J}_{q,1:(N_0-1)}^T$ is negative definite. Note that in the steady state the mw+bp policy satisfies that $\sum_{j=1}^{N_0}\frac{\partial g_{s_j d}}{\partial q_{s_i}} = 0,\ \forall i$, and $\frac{\partial g_{s_i d}}{\partial q_{s_i}} > 0$, $\frac{\partial g_{s_j d}}{\partial q_{s_i}} < 0,\ \forall j \neq i$ when $a \to \infty$. We have that $\mathbf{J}_{q,1:(N_0-1)}$ is diagonally dominant in any column. Now we show that $\left(\mathbf{J}_{q,1:(N_0-1)}\right)^T$ is diagonally dominant in any column as well. This stems from $\frac{\partial g_{s_i d}}{\partial q_{s_i}} > 0$, $\frac{\partial g_{s_j d}}{\partial q_{s_i}} < 0,\ \forall j \neq i$ and (3.18) which indicates $\sum_{j=1}^{N_0}\frac{\partial g_{s_i d}}{\partial q_{s_j}} = 0,\ \forall i$. Therefore $\mathbf{J}_{q,1:(N_0-1)} + \mathbf{J}_{q,1:(N_0-1)}^T$ is diagonally dominant in column and all its diagonal entries are negative, which guarantees that it is negative definite.

Chapter 4

Queue Stability in Networks with Bounded Node Buffers

In this chapter, we consider the problem of network stability in finite-buffer systems. We observe that finite buffer may affect stability even in simplest network structure. For single-commodity systems, we propose a sufficient condition, which follows the fundamental idea of backpressure, for local transmission policies to stabilize the networks based on ODE stability theory. We further extend the condition to multi-commodity systems, with an additional restriction on the coupling level between different commodities, which can model networks with per-commodity buffers and shared buffers. The framework characterizes a set of policies that can stabilize buffered networks, and is useful for analyzing the effect of finite buffers on network stability.

4.1 Motivating Example

We explain the motivation of reconsidering transmission policies to guarantee queue stability in networks with bounded buffers. The introduction of finite buffer size may affect stability even in simplest network structures. Consider the example in Fig. 4-1. The system transmits two commodities with arrival rate λ_1 and λ_2, and destination node T_1 and T_2 respectively. Both commodities share the buffer of node K on their paths. For the link (ℓ, K) where $\ell = 1, 2$, we implement the backpressure routing policy [22], where link (ℓ, K) transmits with rate equal to the capacity value $c_{\ell K}$ when the queue backlog in node ℓ is longer than in node K, while otherwise it does not transmit. Furthermore, we do not allow buffer overflow, i.e., when the buffer in node K is fully occupied then no new commodity will be transmitted to K. For packets departed from node K, the service rate of commodity ℓ is μ_ℓ, and we assume for each commodity it serves in a first-come-first-serve manner, where in every time unit the first μ_ℓ commodity ℓ packets in the buffer are sent to destination T_ℓ.

We consider the case where $\lambda_1 > 3$, i.e., commodity 1 is overloaded, and we figure out that the finiteness of buffer size of node K affects the stability result for commodity 2. When b_K is infinite, backpressure guarantees that commodity 2 will not be increasingly backlogged in the networks under $\lambda_2 \in [0, 3]$, which means commodity 2 is stabilized, as $\mu_2 = 3$ is the corresponding minimum cut value. However, this may not hold when b_k is finite. Consider the example that $b_K = 6$ and both commodities account for 3 units of packets initially in Fig. 4-1, then within 1 unit of time, all these packets are served due to $\mu_1 = \mu_2 = 3$, and $8/(4+8) \times 6 = 4$ units of commodity 1 and $4/(4+8) \times 6 = 2$ units of commodity 2 are injected into node K to fill in the buffer again. Note that queue backlog of commodity 1 takes up higher ratio in node K, and following this process, finally the number the actual throughput of commodity 2 will be 1.5, as we can calculate that there is averagely 1.5 units of commodity 2 packet in node K. Therefore only under $\lambda_2 \in [0, 1.5]$ can commodity 1 be stabilized, which means if $\lambda_2 \in (1.5, 3]$, node 2 will be overflowed, due to finite buffer at node K.

4.2 Single-Commodity System

In this section, we introduce the ODE model for single-commodity systems in buffered communication networks, and utilize ODE stability theory to study network stability. Specifically, we reveal a general sufficient condition for local policies to stabilize the systems. The results, with explicit guidance for transmission policy design in practice, serve as the basis for our analysis of multi-commodity systems and are the main

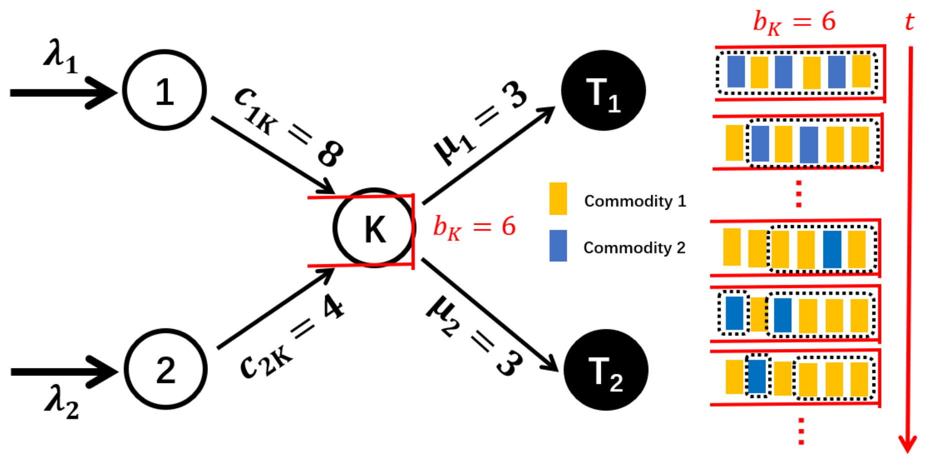


Figure 4-1: Finite buffer may affect stability result. On the right is an example of the queue dynamics in node K following the backpressure policy. The dashed frame denotes the packets to be served at the current time step. In the final state, the average number of commodity 2 packet in the buffer is 1.5, which arises from $1.5 = \frac{\mu_1}{c_{1K}} c_{2K}$ with details deferred to Section 4.3.3. Therefore the actual throughput of commodity 2 is 1.5, less than $\mu_2 = 3$, due to the finite buffer.

technical contribution of the work in this chapter.

4.2.1 Basic Setting

Given an acyclic directed network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = N$ nodes and $|\mathcal{E}| = M$ links. Each node i has a queue buffer, whose size is denoted by b_i. The queue length at node i at time t is denoted by $q_i(t)$, where $q_i(t) \in [0, b_i], \forall t$. We use an $N \times 1$ vector $\mathbf{q}(t)$ to denote the queue length vector of the system, and denote $\mathcal{Q} := \times_{i=1}^{N} [0, b_i]$ as the set of feasible queue length vectors, where $\times_{i=1}^{N}$ denotes the N-dimensional Cartesian product. Packets in the buffer of node i can be transmitted to an adjacent downstream node j through link $(i, j) \in \mathcal{E}$. The transmission rate on link (i, j) at time t, denoted by $g_{ij}(t)$, captures the number of packets transmitted over (i, j) in a time unit. Each link (i, j) is associated with a capacity value c_{ij}, which is the largest flow that link (i, j) can transmit at any time. Specifically, $0 \le g_{ij}(t) \le c_{ij}, \ \forall (i, j) \in \mathcal{E}$.

In communication systems, the transmission rate $g_{ij}(t)$ on each link (i, j) is generally determined by the controller at node i, according to the queue length vector $\mathbf{q}(t)$ and network configurations, including link capacity values $\{c_{ij}\}_{(i,j)\in\mathcal{E}}$ and node buffer size values $\{b_i\}_{i\in\mathcal{V}}$. Therefore $g_{ij}(t)$ is also referred to as the *transmission policy* over link (i,j). In this work, we consider a set of *local*, *stationary* policies that *do not allow buffer overflow.* (i) Locality: We say that a policy is local if $g_{ij}(t)$ depends only on $q_i(t)$ and $q_j(t)$. Local policies are attractive due to their simple implementation and light communication overhead for information exchange. This definition is to some extent extremely local as in reality, node i can have information on all $q_k(t)$'s for $(i,k) \in \mathcal{E}$. However, we show that even with this restricted queue information, a policy can stabilize the network under certain general conditions with clear physical intuition. (ii) Stationarity: A policy is stationary if it does not depend on time explicitly. Note that stationary policies can depend on the network state (e.g., queue size, channel conditions); i.e., a local policy $g_{ij}(t)$ can be denoted by $g_{ij}(q_i(t), q_j(t))$[1]. We neglect the notation t in the following unless specified. (iii) No buffer overflow: Any link (i,j) must stop packet transmission once the buffer of node j is saturated, i.e., $g_{ij} = 0$ when $q_j = b_j$. In addition to the above three constraints on the policy, we naturally have $g_{ij} = 0$ when $q_i = 0$, which means link (i,j) has nothing to transmit when the buffer of upstream node i is empty. For technical convenience, we assume $g_{ij}(q_i, q_j)$ is first-order differentiable with respect to q_i and q_j, and we show that it can well approximate discrete policies in Section 4.2.2.

Packets are injected into the networks at their source nodes. We denote the packet injection rate at node i as $\lambda_i(t)$ and assume that $\lambda_i(t)$ is independent from the queue length vector $\mathbf{q}(t)$. Packets may depart from the networks at any node, and we model this by introducing a meta destination node T to receive the departing packets. The transmission rate from node i to T is $g_{iT}(q_i)$, purely based on q_i under the local policy, and the capacity is denoted by $c_{iT} := \mu_i$, where μ_i denotes the maximum departure rate at node i.

We now formulate the ODE to characterize the queue dynamics. The general form is given by,

$$\dot{\mathbf{q}} = \frac{d\mathbf{q}}{dt} := \mathbf{f}(\mathbf{q}), \tag{4.1}$$

[1]A local policy $g_{ij}(t)$ should also depend on c_{ij}, b_i and b_j, but for brevity we neglect them in the notation as they do not vary with time.

where $\mathbf{f}(\mathbf{q}) := [f_i(\mathbf{q})]_{i=1}^N \in \mathbb{R}^N$ denotes the system dynamics. Due to flow conservation at each node, under the local policy we define earlier, we have for $\forall i \in \mathcal{V}$,

$$f_i(\mathbf{q}) = \lambda_i(t) + \sum_{k:(k,i)\in\mathcal{E}} g_{ki}(q_k, q_i) - \sum_{j:(i,j)\in\mathcal{E}} g_{ij}(q_i, q_j) - g_{iT}(q_i). \tag{4.2}$$

In the above dynamics, the term $\lambda_i(t) + \sum_{k:(k,i)\in\mathcal{E}} g_{ki}(q_k, q_i)$ denotes the total packet inflow rate to node i, while the term $\sum_{j:(i,j)\in\mathcal{E}} g_{ij}(q_i, q_j) + g_{iT}(q_i)$ denotes the total packet departure rate from node i.

4.2.2 Stability Analysis

Next, we study network stability of the dynamics (4.2) under local policies. We consider the queue length stability, as given by Definition 4.1.

Definition 4.1. *The system* (4.2) *is queue length stable if* $\forall i \in \mathcal{V}$, $\lim_{t\to\infty} \sup q_i(t) < \infty$.

Queue length stability ensures the boundedness of the queue backlog at each node. In a system with finite buffers, where overflow is not permitted, instability can only occur at the nodes with unbounded buffers (i.e., source nodes). We note that the assumption of "unbounded" buffers at the source node captures the reality that in many systems internal buffers are small, and buffers at source nodes are relatively large. It is also a useful modeling tool that captures the impact of finite buffers on congestion by "pushing" congestion to the source nodes. Finally, we note that it can be easily shown that queue stability at the source nodes implies rate stability, which is a more meaningful notion of stability in finite-buffer systems[2].

We study the queue length stability based on the stability analysis of the *equilibrium points* of ODE system.

Definition 4.2. $\mathbf{q}^*$ *is an equilibrium point of the system* (4.2) *if* $\mathbf{f}(\mathbf{q}^*) = \mathbf{0}$.

[2]Rate stability implies that the arrival rate is equal to the departure rate [97].

The stability of an equilibrium point $\mathbf{q}^*$ is defined based on either a Lyapunov function or the Jacobian matrix. In this work, we take the latter way to define stability.

Definition 4.3. *The equilibrium point* $\mathbf{q}^*$ *of an ODE system* (4.2) *is asymptotically stable if all the eigenvalues of the Jacobian matrix* $\mathbf{J}$ *at* $\mathbf{q}^*$, *i.e.,*

$$\mathbf{J}\Big|_{\mathbf{q}=\mathbf{q}^*} = \left[\frac{\partial f_i(\mathbf{q})}{\partial q_j}\right]\Big|_{\mathbf{q}=\mathbf{q}^*,\ i,j=1,2,\ldots,N}$$

have negative real parts.

An equilibrium point of an ODE being stable means that the dynamics will not move away from this equilibrium under small disturbances, which is different from queue length stability. In the following, our goal is to connect the notion of equilibrium point stability to queue length stability. The intuitive idea is that if the system in (4.1) has a unique asymptotically stable equilibrium point, then queue length stability follows as the dynamics will not diverge to infinity. Specifically, we first seek a sufficient condition for a policy $g_{ij}(q_i, q_j)$ such that any equilibrium point $\mathbf{q}^*$ is asymptotically stable, which ensures local queue length stability. We then extend it to a global condition which guarantees the global queue length stability.

Local Result

We first derive a sufficient condition of the local policy such that an equilibrium point of the system (4.2) is asymptotically stable.

Theorem 4.1. *For a local policy, if there exists an equilibrium point* $\mathbf{q}^*$, *such that at* $\mathbf{q} = \mathbf{q}^*$,

$$\begin{cases} \frac{\partial g_{ij}(q_i,q_j)}{\partial q_i} > 0,\ \frac{\partial g_{ij}(q_i,q_j)}{\partial q_j} < 0,\ \forall (i,j) \in \mathcal{E}, \\ \frac{\partial g_{iT}(q_i)}{\partial q_i} > 0,\ \exists i \in \mathcal{V} \end{cases} \tag{4.3}$$

then the ODE is asymptotically stable at $\mathbf{q}^*$.

Proof. (Sketch) According to (4.2), given any node $u \in \mathcal{V}$, and for $\forall i = 1, \ldots, N$,

$$\mathbf{J}_{iu} = \begin{cases} 0, \ (i,u) \notin \mathcal{E}, (u,i) \notin \mathcal{E}, i \neq u \\ -\frac{\partial g_{iu}(q_i,q_u)}{\partial q_u}, \ (i,u) \in \mathcal{E} \\ \frac{\partial g_{ui}(q_u,q_i)}{\partial q_u}, \ (u,i) \in \mathcal{E} \\ \sum_{k:(k,u)\in\mathcal{E}} \frac{\partial g_{ku}(q_k,q_u)}{\partial q_u} - \sum_{j:(u,j)\in\mathcal{E}} \frac{\partial g_{uj}(q_u,q_j)}{\partial q_u} - \frac{\partial g_{uT}(q_u)}{\partial q_u}, \ i = u \end{cases} \tag{4.4}$$

Under the condition (4.3), we can verify that all the diagonal entries of $\mathbf{J}$ are negative, and $|\mathbf{J}_{ii}| \geq \sum_{u:u\neq i} |\mathbf{J}_{ki}|, \ \forall i \in \mathcal{V}$. Hence $\mathbf{J}$ is a column diagonally dominant matrix. This indicates that all the eigenvalues have non-positive real parts by the Gershgorin circle theorem [109], and starting from this fact, we can further prove that all eigenvalues have negative real parts under (4.3), with details deferred in Section 4.5. □

Theorem 4.1 conveys that under any local policy so that the two conditions in (4.3) hold, the network will be queue length stable given the initial queue length lies in a sufficiently small neighborhood of the equilibrium point. The first condition is related to the intuition of the backpressure policy [22]: The fluid flows from high pressure nodes to low pressure nodes. The queue length represents the pressure, and once the pressure at the upstream node i increases, then pressure difference increases for i and j thus $g_{ij}(q_i, q_j)$, the flow over (i, j), should be larger; in contrast, once the pressure at the downstream node j increases, the difference decreases and thus $g_{ij}(q_i, q_j)$ should be smaller. The second condition means there exists at least one node (egress node) where packets can depart from the network , and the departure rate increases as more packets accumulate in the buffer. We show later in this section specific examples of network policies, including backpressure, that satisfy both conditions.

Different from prior works proposing and proving queue length stability under a *specific* policy, Theorem 4.1 presents an explicit and intuitive condition which *generalizes* a set of local policies that can guarantee an equilibrium point to be stable, with no limitations over the buffer setting embedded in the policy function

$g_{ij}(q_i, q_j),\ \forall (i,j) \in \mathcal{E}$.

Global Result

Theorem 4.1 can only guarantee local stability. In Theorem 4.2 we extend to a global stability result, by identifying a sufficient condition for the local policy to have at most one globally asymptotically stable equilibrium point, which is a crucial step towards queue length stability.

Theorem 4.2. *If for all queue length vector* $\mathbf{q} \in \mathcal{Q}$*, the policy* $g_{ij}(q_i, q_j)$ *for any* $(i,j) \in \mathcal{E}$ *satisfies* (4.3)*, then* (4.2) *either has a unique asymptotically stable equilibrium point or does not have any equilibrium points.*

Proof. Suppose there exists more than one equilibrium point, then the ODE system must have some equilibrium point $\tilde{\mathbf{q}}$ that is not asymptotically stable. However, by Theorem 4.1, since (4.3) holds at $\tilde{\mathbf{q}}$, then $\tilde{\mathbf{q}}$ should be asymptotically stable, which is a contradiction. $\square$

Theorem 4.2 ensures that any equilibrium point is unique and asymptotically stable when all feasible queue length vectors satisfy (4.3) under a local policy. This indicates that under such a policy, the problem to determine the queue length stability of the system can be reduced to determine whether there exists any feasible solution to the equation $\mathbf{f}(\mathbf{q}) = \mathbf{0}$. This reduces the network stability problem to a feasibility test problem of N-dim equations, which facilitates network stability analysis especially under large N.

To interpret Theorem 4.2 more concretely, we propose the following policy examples that globally satisfy (4.3).

Policy based on Backpressure: At time t, each node i compares its queue length to the queue length of each of its downstream nodes j. If $q_i(t) > q_j(t)$, link (i,j) transmits with its capacity value c_{ij}, and otherwise it does not transmit. The policy can be formulated using differentiable rate functions as follows.

$$\begin{cases} g_{ij}(q_i, q_j) = \frac{1}{1+e^{-a(q_i-q_j-\epsilon)}} \frac{1}{1+e^{-a(b_j-\epsilon-q_j)}} c_{ij}, \ \forall (i,j) \in \mathcal{E} \\ g_{iT}(q_i) = \frac{1}{1+e^{-a(q_i-\epsilon)}} \mu_i, \ \text{if } i \text{ is an egress node} \end{cases} \tag{4.5}$$

where $a > 0$ and $\epsilon > 0$ are preset values. It is easy to verify that (4.5) satisfies (4.3) globally for any feasible $\mathbf{q}$. The form of (4.5) matches backpressure under $a \to \infty$ and $\epsilon := 1/\sqrt{a} \to 0$, which transmits the packets from i to j with rate c_{ij} if and only if $q_i > q_j$ and $q_j < b_j$, i.e., the buffer of j is not saturated. This shows that (4.5) is an approximation to backpressure under sufficiently large a and small ϵ. Moreover, the form of $g_{iT}(q_i)$ guarantees work-conservation under $a \to \infty$ and $\epsilon := 1/\sqrt{a} \to 0$, i.e., maximum departure rate if there are packets in the buffer.

Policy based on Buffer Occupancy Level: Consider

$$\begin{cases} g_{ij}(q_i, q_j) = \frac{1}{1+e^{-a(q_i-\epsilon)}} \left(1 - \frac{q_j}{b_j}\right) c_{ij}, \ \forall (i,j) \in \mathcal{E} \\ g_{iT}(q_i) = \frac{1}{1+e^{-a(q_i-\epsilon)}} \mu_i, \ \text{if } i \text{ is an egress node} \end{cases}$$

We can similarly verify it globally satisfies (4.3). Taking a large enough and $\epsilon \to 0$ to guarantee $g_{ij} = 0$ when $q_i = 0$. The transmission rate of link (i, j) in this policy declines linearly with respect to q_j/b_j, the buffer occupancy level of node j. This policy does not transmit with rate equal to link capacity when the downstream node's buffer is not empty, but Theorem 4.2 reveals that it can also stabilize the network if there exists an equilibrium point for (4.2) under this policy.

4.2.3 Existence of Equilibrium Point

With Theorem 4.2, The remaining gap to proving queue length stability for a local policy that satisfies (4.3) globally is to show that there exists an equilibrium point for (4.2) under this policy. There is no well-known answer to this question except verifying the feasibility of (4.2) directly. We identify one sufficient condition for the existence of an equilibrium point in Lemma 4.1, proved by Poincare-Miranda

Theorem, a multi-dimensional version of the intermediate value theorem.

Poincare-Miranda Theorem [110]: Consider n continuous functions of n variables, $g_1, \ldots, g_n$. Assume that for each variable x_i, the function g_i is constantly negative when $x_i = -1$ and constantly positive when $x_i = 1$. Then there is a point in the n-dimensional cube $[-1, 1]^n$ such that $g_1, \ldots, g_n$ are simultaneously equal to 0.

Lemma 4.1. *Suppose that there exists finite values* $\{\bar{b}_j\}_{i=1}^N$ *and* $\{\underline{b}_j\}_{i=1}^N$ *such that for every node* i *and any* $q_j \in [\underline{b}_j, \bar{b}_j]$, $\forall j \neq i$: *(i) when* $q_i = \bar{b}_i$, *the policy leads to* $f_i(\mathbf{q}) \leq 0$; *(ii) when* $q_i = \underline{b}_i$, $f_i(\mathbf{q}) \geq 0$, *then system* (4.2) *has an equilibrium point in* $\bar{\mathcal{B}} := \times_{i=1}^N [\underline{b}_i, \bar{b}_i]$.

Proof. We can apply the Poincare-Miranda Theorem over the cube $\bar{\mathcal{B}} := \times_{i=1}^N [\underline{b}_i, \bar{b}_i]$ and taking $g_i := f_i$ in (4.1), which ensures at least one $\mathbf{q}$ such that $\mathbf{f}(\mathbf{q}) = \mathbf{0}$.[3] □

Lemma 4.1 is a general result for the existence of an equilibrium point in which $\bar{\mathcal{B}}$ requires specification for a particular policy. We consider the policy (4.5) as an example to show how we can obtain $\bar{\mathcal{B}}$ in Section 4.5. Combining Theorem 4.2 and Lemma 4.1, we have the following theorem for queue length stability.

Theorem 4.3. *For any local policy that satisfies the conditions in both Theorem 4.2 and Lemma 4.1, there exists a unique stable equilibrium point in* $\bar{\mathcal{B}}$ *defined in Lemma 4.1, and the system is queue length stable (and thus rate stable).*

Proof. Theorem 4.2 ensures there exists at most one stable equilibrium point, while Lemma 4.1 ensures there exists at least one equilibrium point. Therefore there exists a unique stable equilibrium point and thus starting at any queue length vector, the dynamics will converge to the equilibrium. □

Theorem 4.3 can be viewed as an extension to Theorem 4.2 which only adds a sufficient condition for equilibrium point existence. The above stability results differ from previous works in that (i) they capture a set of policies, and (ii) they can be applied in systems with arbitrary buffer settings.

[3]In the proof, we do not follow the condition *constantly negative/positive*, instead we consider *constantly non-positive/non-negative*. This does not affect our result as we consider the existence of solution in a closed cube.

4.3 Multi-Commodity System

We extend the above results to multi-commodity systems, where different commodities are coupled due to shared links or buffers. We identify a sufficient condition for queue length stability that is similar to the single-commodity case, but with an additional condition on the coupling among different commodities.

4.3.1 Basic Setting

Suppose that the system consists of C commodities. We use $q_i^{(\ell)}(t)$ to denote the queue length of commodity ℓ at node i and time t, and an $N_\ell \times 1$ vector $\mathbf{q}^{(\ell)}(t)$ to denote the queue length vector for commodity ℓ, where N_ℓ denotes the number of nodes on the available paths of commodity ℓ. We denote vector $\mathbf{q}(t)$ that concatenates $\{\mathbf{q}^{(\ell)}(t)\}_{\ell=1}^{C}$ as the queue length vector for the entire network. For commodity ℓ, we denote the arrival rate at node i as $\lambda_i^{(\ell)}(t)$, the transmission rate on link (i,j) as $g_{ij}^{(\ell)}(t)$, and the departure rate to outside the networks at node i as $g_{iT_\ell}^{(\ell)}(t)$, where T_ℓ denotes the meta destination connected for commodity ℓ.

We also consider local, stationary policies in the multi-commodity case with no overflow permitted, namely the same conditions as for the single-commodity systems. The queueing dynamics under a local policy for any commodity ℓ at any node i is given by

$$\begin{aligned}\dot{q}_i^{(\ell)} = \lambda_i^{(\ell)} &+ \sum_{k:(k,i)\in\mathcal{E}} g_{ki}^{(\ell)}(\{q_k^{(p)}\}_{p=1}^{C}, \{q_i^{(p)}\}_{p=1}^{C}) \\ &- \sum_{j:(i,j)\in\mathcal{E}} g_{ij}^{(\ell)}(\{q_i^{(p)}\}_{p=1}^{C}, \{q_j^{(p)}\}_{p=1}^{C}) - g_{iT}^{(\ell)}(\{q_i^{(p)}\}_{p=1}^{C}).\end{aligned} \tag{4.6}$$

4.3.2 Stability Analysis

We follow a similar pattern to the single-commodity case. We first derive conditions for local queue length stability, and extend to a global sufficient condition that ensures an equilibrium point $\mathbf{q}^*$ to be globally unique and stable, which captures a set of local policies that can achieve queue length stability for all commodities.

To study the stability at $\mathbf{q}^*$, we need to first introduce an important concept of block diagonally dominant matrix.

Definition 4.4. *A matrix* $\mathbf{J}$ *is called a* block diagonally dominant matrix *if* $\mathbf{J}$ *can be partitioned into the following* $C \times C$ *blocks*

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{1,1} & \mathbf{J}_{1,2} & \cdots & \mathbf{J}_{1,C} \\ \mathbf{J}_{2,1} & \mathbf{J}_{2,2} & \cdots & \mathbf{J}_{2,C} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{J}_{C,1} & \mathbf{J}_{C,2} & \cdots & \mathbf{J}_{C,C} \end{bmatrix} \tag{4.7}$$

where $\mathbf{J}_{i,j} \in \mathbb{R}^{N_i \times N_j}$*, the diagonal submatrices* $\{\mathbf{J}_{i,i}\}_{i=1}^{C}$ *are nonsingular, and for some operator norm* $||\cdot||$, $||\mathbf{J}_{j,j}^{-1}||^{-1} \geq \sum_{k=1,k\neq j}^{C} ||\mathbf{J}_{j,k}||$, $\forall j = 1, \ldots, C$. $\mathbf{J}$ *is called a block strictly diagonally dominant matrix if the last condition is a strict inequality.*

This definition is directly related to the system (4.6): Each block on the diagonal ($\mathbf{J}_{\ell,\ell}$) represents the derivative of the dynamics of commodity ℓ with respect to the $q_i^{(\ell)}$ at each node i, while each off-diagonal block ($\mathbf{J}_{\ell',\ell}$) denotes its derivative with respect to $q_i^{(\ell')}$ for commodity $\ell' \neq \ell$, which reflects the coupling between commodities ℓ and ℓ'. Intuitively, block diagonally dominance requires the total coupling effect of each commodity ℓ (i.e. $\sum_{k=1,k\neq \ell}^{C} ||\mathbf{J}_{\ell,k}||$) to be relatively small.

Before proving our results based on the block diagonally dominant matrix, we need to introduce the concepts of M-matrix and absolute norm, and a theorem [111] about matrix eigenvalues.

Definition 4.5. *A matrix is called an M-matrix if all of its eigenvalues have nonnegative real parts and all its off-diagonal entries are nonpositive.*

Definition 4.6. *A norm* $||\cdot||$ *is an absolute norm if* $||\mathbf{q}|| = ||\ |\mathbf{q}|\ ||$*, where* $|\mathbf{q}|$ *denotes the element-wise absolute value.*

Theorem: [111] Let matrix $\mathbf{J}$ be partitioned as in (4.7), and let $\mathbf{J}$ be block strictly diagonally dominant. Further, suppose that $-\mathbf{J}_{j,j}$ is an M-matrix, $\forall 1 \leq j \leq C$, and the norm is an absolute norm. The any eigenvalue of $\mathbf{J}$ has negative real part.

The theorem informs that we need to ensure three points to prove that an equilibrium point is asymptotically stable: (i) $\mathbf{J}$ is block strictly diagonally dominant; (ii) $-\mathbf{J}_{\ell,\ell}$ is an M-matrix, $\forall \ell = 1, \ldots, C$; (iii) the norm is an absolute norm. The last point obviously holds when we apply the l_2-norm. We have the following lemma for (i) and (ii).

Lemma 4.2. *Suppose that the Jacobian matrix* $\mathbf{J}$ *of* (4.6) *at an equilibrium point* $\mathbf{q}^*$ *under the local policy satisfies*

- *Block strictly diagonal dominance:*

$$\sqrt{\lambda_{\min}(\mathbf{J}_{\ell,\ell}\mathbf{J}_{\ell,\ell}^T)} > \sum_{p=1,p\neq\ell}^{C} \sqrt{\lambda_{\max}(\mathbf{J}_{p,\ell}^T\mathbf{J}_{p,\ell})}$$

  for $\forall \ell = 1, \ldots, C$.

- *M-matrix condition: For* $\forall (i,j) \in \mathcal{E}$ *and* $\forall \ell = 1, \ldots, C$, $\frac{\partial g_{ij}^{(\ell)}}{\partial q_i^{(\ell)}} > 0$, $\frac{\partial g_{ij}^{(\ell)}}{\partial q_j^{(\ell)}} < 0$, *and* $\frac{\partial g_{iT}^{(\ell)}}{\partial q_i^{(\ell)}} > 0$, $\exists i \in \mathcal{V}$.

then the equilibrium point $\mathbf{q}^*$ *is asymptotically stable.*

Proof. The block strictly diagonal dominance constraint is derived based on constraint $||\mathbf{J}_{s,s}^{-1}||^{-1} \geq \sum_{t=1,t\neq s}^{C} ||\mathbf{J}_{t,s}||$ under l_2-norm. Specifically,

$$||\mathbf{J}_{s,s}^{-1}|| = \sqrt{\lambda_{\max}((\mathbf{J}_{s,s}^T)^{-1}\mathbf{J}_{s,s}^{-1})} = \sqrt{\lambda_{\max}(\mathbf{J}_{s,s}\mathbf{J}_{s,s}^T)^{-1}} = \left(\sqrt{\lambda_{\min}(\mathbf{J}_{s,s}\mathbf{J}_{s,s}^T)}\right)^{-1}.$$

Therefore $||\mathbf{J}_{s,s}^{-1}||^{-1} = \sqrt{\lambda_{\min}(\mathbf{J}_{s,s}\mathbf{J}_{s,s}^T)}$ while the RHS $\sum_{t=1,t\neq s}^{C} ||\mathbf{J}_{t,s}||$ is equal to $\sum_{t=1,t\neq s}^{C} \sqrt{\lambda_{\max}(\mathbf{J}_{t,s}^T\mathbf{J}_{t,s})}$ based on the definition of l_2-norm. The M-matrix condition can be similarly proved as Theorem 4.1 to show that under the condition, $-\mathbf{J}_{\ell,\ell}$ is an M-matrix. Then all the eigenvalues of $\mathbf{J}$ have negative real parts based on [111], and thus $\mathbf{q}^*$ is asymptotically stable. □

Compared with Theorem 4.1 for single-commodity systems, the *block strictly diagonal dominance* in Lemma 4.2 is an additional condition for the restriction of coupling level among different commodities, while for each commodity, the *M-matrix* *condition* coincides with the conditions (4.3). In fact, in the case that different commodities do not affect each other, the block strictly diagonal dominance holds naturally as all the off-diagonal blocks are zero matrices, and thus Lemma 4.2 is reduced to a C-fold version of Theorem 4.1.

Similar to the single-commodity case, we can obtain a sufficient condition of a local policy such that (4.6) has a unique stable equilibrium point, reducing the stability problem to testing the existence of an equilibrium point.

Theorem 4.4. *Suppose that the conditions in Lemma 4.2 hold for any feasible* $\mathbf{q}$*, then there either exists a unique asymptotically stable equilibrium point for* (4.6) *or there does not exist any equilibrium point.*

4.3.3 Existence of Equilibrium Point

In terms of the existence of an equilibrium point, we similarly apply the Poincare-Miranda Theorem. However since it can only capture cube form regions of $\mathbf{q}$, we can only obtain results for systems with per-commodity buffers, where node i allocates a portion of its buffer to each commodity ℓ, with length denoted as $b_i^{(\ell)}$, which satisfies $q_i^{(\ell)} \leq b_i^{(\ell)}$ and $\sum_{p=1}^{C} b_i^{(p)} \leq b_i$. For systems with shared buffers, the constraint is $\sum_{p=1}^{C} q_i^{(p)} \leq b_i$, not a cube, hence this theorem is not applicable. Systems with shared buffers will be discussed in Section 4.3.5.

Lemma 4.3. *For every commodity* ℓ*, under the condition that there exists finite values* $\{\bar{b}_j^{(\ell)}\}_{i=1}^N$ *and* $\{\underline{b}_j^{(\ell)}\}_{i=1}^N$ *such that for every node* i*, and* $q_j^{(\ell)} \in [\underline{b}_j^{(\ell)}, \bar{b}_j^{(\ell)}]$, $\forall j \neq i$*: (i) when* $q_i^{(\ell)} = \bar{b}_i^{(\ell)}$, $f_i^{(\ell)}(\mathbf{q}) \leq 0$*; (ii) when* $q_i^{(\ell)} = \underline{b}_i^{(\ell)}$, $f_i^{(\ell)}(\mathbf{q}) \geq 0$*. There exists a feasible equilibrium point* $\mathbf{q} \in \bar{\mathcal{B}} := \times_{i=1}^N \times_{p=1}^C [\underline{b}_i^{(p)}, \bar{b}_i^{(p)}]$ *for system* (4.6).

Proof. The proof is a simple extension of Lemma 4.1 by applying its idea for each commodity ℓ. □

Note that Lemma 4.3 only decouples different commodities at node buffers, which means it still applies when different commodities affect each other's transmission rates on shared links. Combining Lemma 4.3 with Theorem 4.4, we obtain the following result regarding queue length stability for all commodities. The proof is similar to Theorem 4.3.

Theorem 4.5. *For policies under the conditions of Theorem 4.4 and Lemma 4.3, there exists a globally unique stable equilibrium point for* (4.6)*, which guarantees queue length stability for all commodities.*

4.3.4 Rule of Thumb

Theorem 4.4 gives a sufficient condition to guarantee queue stability. However, the coupling constraint is not an explicit form of the queue policy as in Theorem 4.1, which impedes its practical usage. Technical difficulty blocks derivation of explicit coupling conditions. In this section, we propose a rule of thumb in an explicit form that benefits the queue stability in two folds: (i) For a system that is not stable, adding coupling between different commodities following this rule drives the system to approach or reach being stable; (ii) For a system that is stable, the introduction of such coupling accelerates the convergence to the steady state. The result reveals an easy-to-implement way to introduce coupling among the queueing dynamics of different commodities that drives the network system to queue stability.

The idea is to quantify the effect of the coupling on the eigenvalues. Recall the block matrix (4.7). Introducing the coupling changes the form of off-diagonal blocks, which does not affect the form of the trace of the matrix, as it is only associated with diagonal elements. Since the sum of the eigenvalues equals to the trace, then if the eigenvalues of each diagonal block matrix $\mathbf{J}_{i,i}$ have negative real part, the eigenvalues sum is negative. Given this fact, our rule of thumb is to *balance* all the eigenvalues, which minimizes the maximum real part of all the eigenvalues since it determines the convergence rate. Balancing the eigenvalues of the Jacobian J can be formulated as an optimization problem that minimizes the quadratic sum of the real part of the eigenvalues.

We first present our idea through a toy example: two nodes connected by a single link shared by two commodities, and all the eigenvalues of $\mathbf{J}$ are real numbers. The corresponding queueing dynamics is

$$\begin{cases} \dot{q}_1^{(1)} = \lambda_1^{(1)} + g_{12}^{(1)}(\{q_1^{(p)}\}_{p=1}^2, \{q_2^{(p)}\}_{p=1}^2) \\ \dot{q}_2^{(1)} = g_{12}^{(1)}(\{q_1^{(p)}\}_{p=1}^2, \{q_2^{(p)}\}_{p=1}^2) - g_{2T}^{(1)}(q_2^{(1)}) \\ \dot{q}_1^{(2)} = \lambda_1^{(2)} + g_{12}^{(2)}(\{q_1^{(p)}\}_{p=1}^2, \{q_2^{(p)}\}_{p=1}^2) \\ \dot{q}_2^{(2)} = g_{12}^{(2)}(\{q_1^{(p)}\}_{p=1}^2, \{q_2^{(p)}\}_{p=1}^2) - g_{2T}^{(2)}(q_2^{(2)}) \end{cases} \tag{4.8}$$

Under the assumption that the eigenvalues of $\mathbf{J}$ are all real, our objective is to reduce $\sum_{i=1}^2 \lambda_i^2$. Note that $\sum_{i=1}^2 \lambda_i^2 = \mathbf{trace}(\mathbf{J}^2)$, then the objective can be expressed as

$$\sum_i \lambda_i^2 = \mathbf{trace}\left(\begin{bmatrix} J_{1,1} & J_{1,2} \\ J_{2,1} & J_{2,2} \end{bmatrix}^2\right) = \mathbf{trace}(J_{1,1}^2 + J_{1,2}J_{2,1} + J_{2,1}J_{1,2} + J_{2,2}^2)$$

Since coupling only lies in the off-diagonal matrices, then the objective function with respect to the coupling satisfies $\mathbf{trace}(J_{1,2}J_{2,1} + J_{2,1}J_{1,2}) = 2\mathbf{trace}(J_{1,2}J_{2,1})$ due to the fact that commutation does not affect trace value. The objective is an explicit function of the off-diagonal Jacobian matrix, which is

$$J_{1,2} = \begin{bmatrix} -\frac{\partial g_{12}^{(1)}}{\partial q_1^{(2)}} & -\frac{\partial g_{12}^{(1)}}{\partial q_2^{(2)}} \\ \frac{\partial g_{12}^{(1)}}{\partial q_1^{(2)}} & \frac{\partial g_{12}^{(1)}}{\partial q_2^{(2)}} \end{bmatrix}, \; J_{2,1} = \begin{bmatrix} -\frac{\partial g_{12}^{(2)}}{\partial q_1^{(1)}} & -\frac{\partial g_{12}^{(2)}}{\partial q_2^{(1)}} \\ \frac{\partial g_{12}^{(2)}}{\partial q_1^{(1)}} & \frac{\partial g_{12}^{(2)}}{\partial q_2^{(1)}} \end{bmatrix}$$

and thus

$$\mathbf{trace}(J_{1,2}J_{2,1}) = \left(\frac{\partial g_{12}^{(1)}}{\partial q_1^{(2)}}\frac{\partial g_{12}^{(2)}}{\partial q_1^{(1)}}\right) + \left(-\frac{\partial g_{12}^{(1)}}{\partial q_2^{(2)}}\frac{\partial g_{12}^{(2)}}{\partial q_1^{(1)}}\right) + \left(-\frac{\partial g_{12}^{(1)}}{\partial q_1^{(2)}}\frac{\partial g_{12}^{(2)}}{\partial q_2^{(1)}}\right) + \left(\frac{\partial g_{12}^{(1)}}{\partial q_2^{(2)}}\frac{\partial g_{12}^{(2)}}{\partial q_2^{(1)}}\right) \tag{4.9}$$

Our goal is to reduce the objective function, therefore one rule of thumb of the policy is to satisfy

$$\frac{\partial g_{12}^{(p)}}{\partial q_1^{(l)}}\frac{\partial g_{12}^{(l)}}{\partial q_1^{(p)}} < 0, \; \frac{\partial g_{12}^{(p)}}{\partial q_1^{(l)}}\frac{\partial g_{12}^{(p)}}{\partial q_2^{(l)}} < 0, \; p, l \in \{1, 2\}, \; p \neq l \tag{4.10}$$

under which each of the four bracketed terms in (4.9) to negative. Therefore for this link $(1,2)$ and the two commodities sharing this link, with this condition, it achieves more balanced eigenvalues compared with non-coupling case.

The intuition of (4.10) is as follows. (i) For $\frac{\partial g_{12}^{(p)}}{\partial q_1^{(l)}}\frac{\partial g_{12}^{(l)}}{\partial q_1^{(p)}} < 0$, suppose that when $q_1^{(1)}$ increases, $g_{12}^{(2)}$ increases, i.e., $\frac{\partial g_{12}^{(2)}}{\partial q_1^{(1)}} > 0$, then an increasing $g_{12}^{(2)}$ tends to decrease $q_1^{(2)}$, and an intuitive idea to balance the two commodities is to increase $g_{12}^{(1)}$ to prevent the gap $g_{12}^{(2)} - g_{12}^{(1)}$ from being too large. This means $\frac{\partial g_{12}^{(1)}}{\partial q_1^{(2)}} < 0$. (ii) For $\frac{\partial g_{12}^{(p)}}{\partial q_1^{(l)}}\frac{\partial g_{12}^{(p)}}{\partial q_2^{(l)}} < 0$, it shares the intuition with the condition in Theorem 4.2, which means the flow of commodity p on link (i,j) follows different trends when $q_i^{(l)}$ and $q_j^{(l)}$ shares the same changing direction. This balances the $g_{ij}^{(p)}$ to fluctuate around an intermediate value and impedes it from reaching extreme values, either occupying the resource of or fully sacrificing itself for other commodities.

This result can be extended to multiple commodities sharing a link, as Lemma 4.4 indicates.

Lemma 4.4. *Consider a system with a single link* $(1,2)$ *shared by* C *commodities, then* $\sum_{p=1}^{C}\sum_{l\neq p} \boldsymbol{trace}(J_{p,l}J_{l,p}) \leq 0$ *if*

$$\frac{\partial g_{12}^{(p)}}{\partial q_1^{(l)}}\frac{\partial g_{12}^{(l)}}{\partial q_1^{(p)}} \leq 0,\ \ \frac{\partial g_{12}^{(p)}}{\partial q_1^{(l)}}\frac{\partial g_{12}^{(p)}}{\partial q_2^{(l)}} \leq 0,\ \ p,l \in \{1,2,\ldots,C\},\ \ p\neq l \tag{4.11}$$

Proof. Based on extending (4.8) to C commodities, we can derive that

$$\begin{aligned}&\sum_{p=1}^{C}\sum_{l\neq p}\mathbf{trace}(J_{p,l}J_{l,p})\\ &=\sum_{p=1}^{C}\sum_{l\neq p}\left(\frac{\partial g_{12}^{(l)}}{\partial q_1^{(p)}}\frac{\partial g_{12}^{(p)}}{\partial q_1^{(l)}}\right)+\left(-\frac{\partial g_{12}^{(l)}}{\partial q_2^{(p)}}\frac{\partial g_{12}^{(p)}}{\partial q_1^{(l)}}\right)+\left(-\frac{\partial g_{12}^{(l)}}{\partial q_1^{(p)}}\frac{\partial g_{12}^{(p)}}{\partial q_2^{(l)}}\right)+\left(\frac{\partial g_{12}^{(l)}}{\partial q_2^{(p)}}\frac{\partial g_{12}^{(p)}}{\partial q_2^{(l)}}\right)\end{aligned}$$

and the condition (4.11) guarantees that each bracked term is non-positive. □

Lemma 4.4 characterizes a rule of thumb to introduce and utilize the coupling to balance the eigenvalues on a single link. Following the idea, we can extend the result to multi-hop networks, where each link satisifies the condition (4.11) with an extra condition regulating the coupling of all upstream and downstream links at each node, stated in Theorem 4.6.

Theorem 4.6. *Consider a multi-hop network shared by C commodities, then* $\sum_{p=1}^{C}\sum_{l\neq p} \boldsymbol{trace}(J_{p,l}J_{l,p}) \leq 0$ *if* (4.11) *holds for each link* (i,j) *and at each node* i, $\forall p = 1,\ldots,C$,

$$\begin{cases} \frac{\partial g_{ij_1}^{(p)}}{\partial q_i^{(l)}}\frac{\partial g_{ij_2}^{(p)}}{\partial q_i^{(l)}} \geq 0, \forall (i,j_1),(i,j_2)\in\mathcal{E} \\ \frac{\partial g_{k_1 i}^{(p)}}{\partial q_i^{(l)}}\frac{\partial g_{k_2 i}^{(p)}}{\partial q_i^{(l)}} \geq 0, \forall (k_1,i),(k_2,i)\in\mathcal{E} \\ \frac{\partial g_{ki}^{(p)}}{\partial q_i^{(l)}}\frac{\partial g_{ij}^{(p)}}{\partial q_i^{(l)}} \leq 0, \forall (k,i),(i,j)\in\mathcal{E} \end{cases} \tag{4.12}$$

The extra condition (4.12) has following intuition. Consider an arbitrary node i. (i) $\frac{\partial g_{ij_1}^{(p)}}{\partial q_i^{(l)}}\frac{\partial g_{ij_2}^{(p)}}{\partial q_i^{(l)}} \geq 0$ means the coupling between commodity l and p should be in same direction for all downstream links of node i; (ii) $\frac{\partial g_{k_1 i}^{(p)}}{\partial q_i^{(l)}}\frac{\partial g_{k_2 i}^{(p)}}{\partial q_i^{(l)}} \geq 0$ means the same direction also holds for all upstream links of node i; (iii) $\frac{\partial g_{ki}^{(p)}}{\partial q_i^{(l)}}\frac{\partial g_{ij}^{(p)}}{\partial q_i^{(l)}} \leq 0$ means that the dependence of any upstream-downstream link pair of node i are opposite. The condition (4.12) in fact echoes the underlying idea of Lemma 4.1 (also the M-matrix constraint in Lemma 4.2), under which all upstream (downstream) links of a node i share the same trend, while an upstream link and a downstream link share different trend, taking partial derivative with respect to q_i.

4.3.5 Shared buffer Case Study: Switched Networks

In networks with shared buffers, one commodity may fully occupy a shared buffer and thus squeeze out other commodities, which may induce the instability of these commodities. While we have not been able to obtain results for general networks under this setting, we can obtain explicit results over single-hop network structure as Fig. 4-1, which serves as the basic structure for server farms [29] and switches in data center networks [17].

Consider the policy based on backpressure (4.5) for the system in Fig. 4-1. Let

$$\begin{cases} \alpha_{ij}^{(\ell)} = \frac{1}{1+e^{-a(q_i^{(\ell)}-q_j^{(\ell)}-\epsilon)}}, \ \forall (i,j) \in \mathcal{E}, \ \ell = 1,2 \\ \beta_K = \frac{1}{1+e^{-a(b_K-q_K^{(1)}-q_K^{(2)}-\epsilon)}}, \end{cases}$$

where $a \to \infty$ and $\epsilon := 1/\sqrt{a} \to 0$, and we can then write the queueing dynamics as $^{(1)}_{1} = \lambda_1 - c_{1K}\alpha_{1K}^{(1)}\beta_K, \ \dot{q}_K^{(1)} = c_{1K}\alpha_{1K}^{(1)}\beta_K - \mu_1\alpha_{KT}^{(1)}$
$\dot{q}_2^{(2)} = \lambda_2 - c_{2K}\alpha_{2K}^{(2)}\beta_K, \ \dot{q}_K^{(2)} = c_{2K}\alpha_{2K}^{(2)}\beta_K - \mu_2\alpha_{KT}^{(2)}$ We assume $c_{1K} > \mu_1$ and $c_{2K} > \mu_2$, then the admissible region for (λ_1, λ_2) is $[0, \mu_1] \times [0, \mu_2]$. If both commodities have arrival rates interior to the admissible region, then the network is queue length stable for both commodities under (4.5). However, suppose that commodity 1 is overloaded ($\lambda_1 > \mu_1$), Lemma 4.5 shows that under $c_{1K}/\mu_1 > c_{2K}/\mu_2$, $\lambda_2 < \mu_2$ does not guarantee the queue length stability of commodity 2, which explains the instability example given in the introduction (Fig. 4-1) due to the non-existence of equilibrium point for the subsystem (4.3.5).

Lemma 4.5. *For the 2-commodity toy system in Fig. 4-1, suppose that $\lambda_1 > \mu_1$ and $c_{1K}/\mu_1 > c_{2K}/\mu_2$, then under the* (4.3.5) *and* (4.3.5)*, the subsystem* (4.3.5) *has an equilibrium point if and only if $\lambda_2 \in [0, \frac{\mu_1}{c_{1K}}c_{2K}] \subset [0, \mu_2)$.*

Proof. To guarantee that the subsystem (4.3.5) has an equilibrium point, we need have $\dot{q}_2^{(2)} = 0$ and $\dot{q}_K^{(2)} = 0$ i.e., $\lambda_2 = c_{2K}\alpha_{2K}^{(2)}\beta_K = \mu_2\alpha_{KT}^{(2)}$. To capture this, we first identify the value of β_K. Since $c_{1K}/\mu_1 > c_{2K}/\mu_2$, then when node K saturates, commodity 1 will squeeze out commodity 2 and dominates in node K, hence $\alpha_{1K}^{(1)} \to 1$ as queue backlog at node 1 increases to infinity and $\alpha_{KT}^{(1)} \to 1$ since there exists positive queue backlog length of commodity 1 in node K. Since node K has finite buffer, then when $t \to \infty$ we should have

$$\dot{q}_K^{(1)} = c_{1K}\alpha_{1K}^{(1)}\beta_K - \mu_1\alpha_{KT}^{(1)} = c_{1K}\beta_K - \mu_1 = 0,$$

and thus $\beta_K = \mu_1/c_{1K}$. Then we require

$$\lambda_2 = c_{2K}\alpha_{2K}^{(2)}\beta_K = \frac{\mu_1 c_{2K}}{c_{1K}}\alpha_{2K}^{(2)} \in \left[0, \frac{\mu_1 c_{2K}}{c_{1K}}\right]$$

to ensure the existence of an equilibrium point, and $\left[0, \frac{\mu_1 c_{2K}}{c_{1K}}\right] \subset [0, \mu_2)$ because $c_{1K}/\mu_1 > c_{2K}/\mu_2$. □

Lemma 4.5 quantifies the shrinkage of the range of λ_2 under which backpressure can stabilize commodity 2. The result can be extended to Theorem 4.7, where C commodities shares a single buffer in the one-hop system as shown in Fig. 4-2. Theorem 4.7 identifies the maximum arrival rate of each commodity to guarantee the existence of an equilibrium point for the subsystem of each commodity under the policy based on backpressure, in the form of (4.3.5), when overloading occurs on one commodity. Proof of Theorem 4.7 is similar to that of Lemma 4.5 above.

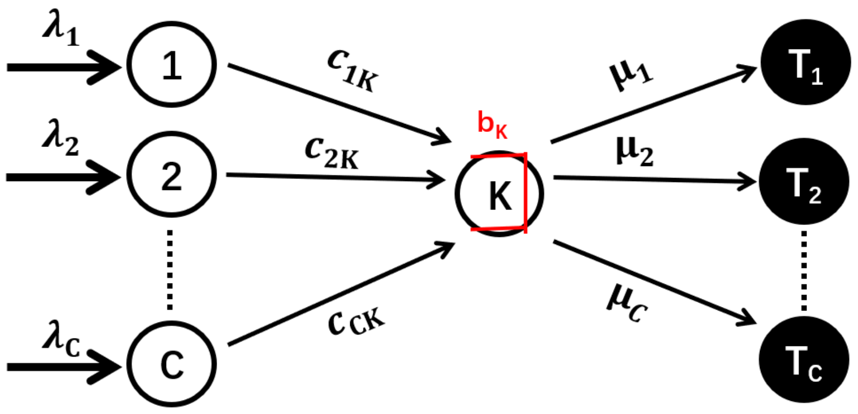


Figure 4-2: One-hop system with C commodities

Theorem 4.7. *For the system in Fig. 4-2, assume that $c_{iK} > \mu_i$, $\forall i = 1, 2, \ldots, C$, and w.l.o.g $c_{iK}/\mu_i > c_{jK}/\mu_j$ for $\forall i, j$, $1 \le i < j \le C$. Suppose that commodity ℓ is the* only *overloaded commodity ($\lambda_\ell > \mu_\ell$), then there exists an equilibrium point for the subsystem of commodity p ($p \neq \ell$) with $\lambda_p \in [0, \mu_p)$ for $p = 1, 2, \ldots, \ell - 1$ and with $\lambda_p \in [0, \mu_\ell c_{pK}/c_{\ell K}] \subset [0, \mu_p)$ for $p = \ell + 1, \ldots, C$, under the policy in the form of* (4.3.5) *for every commodity.*

4.4 Summary and Future Work

In this chapter, we propose an ODE model that can capture the dynamics of buffered communication systems to study network stability. For single-commodity systems, we propose a sufficient condition for a local policy to stabilize the network. The result characterizes a set of policies, and captures systems with arbitrary buffers. For such policies the network stability problem is reduced to an problem testing the existence of an equilibrium point for the ODE system. For multi-commodity systems, we extend the condition by incorporating an additional condition on the coupling level between different commodities, and explain the existence of an equilibrium point in different buffer settings. Future work includes obtaining necessary conditions for network stability, and the analysis of other network performance metrics such as throughput, delay, and fairness using this framework.

4.5 Chapter Appendix

4.5.1 Proof of Theorem 4.1

The basic idea is to use Theorem 4.8 to show all the eigenvalues of $\mathbf{J}$ at the equilibrium point under (4.3).

Theorem 4.8. *[108] All the eigenvalues of a matrix $\mathbf{J} \in \mathbb{R}^{N\times N}$ have strictly negative real parts if and only if there exists a symmetric positive-definite matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ that satisfies $\mathbf{A}\mathbf{J}^T + \mathbf{J}\mathbf{A}$ is negative definite.*

We prove the existence of a *diagonal* matrix $\mathbf{A} := \text{diag}\{a_i\}_{i=1}^N$ with $a_i > 0, \forall i = 1, \dots, N$ so that $\mathbf{A}\mathbf{J} + \mathbf{J}^T\mathbf{A}$ is negative definite. The proof relies on the notation $\mathbf{J}_0 = \mathbf{J} - \mathbf{\Lambda}$, where $\mathbf{\Lambda} := \text{diag}\left\{-\frac{\partial g_{iT}}{\partial q_i}\right\}_{i=1}^N$, and the following Lemma 4.6, with proof in Section 4.5.2.

Lemma 4.6. $\exists \boldsymbol{\delta} \in \mathbb{R}_+^N$ *such that* $\mathbf{J}_0\boldsymbol{\delta} = \mathbf{0}$.

Denote $\mathbf{Q} := \mathbf{A}\mathbf{J} + \mathbf{J}^T\mathbf{A}$, where we can obtain

$$Q_{ij} = \begin{cases} -a_i \frac{\partial g_{ij}}{\partial q_j} + a_j \frac{\partial g_{ij}}{\partial q_i},\ i \neq j \\ 2a_i \left(\sum_{k:(k,i)\in\mathcal{E}} \frac{\partial g_{ki}}{\partial q_i} - \sum_{l:(i,l)\in\mathcal{E}} \frac{\partial g_{il}}{\partial q_i} - \frac{\partial g_{iT}}{\partial q_i} \right),\ i = j \end{cases} \tag{4.13}$$

We show that there exists a negative vector $\boldsymbol{\alpha} := \{\alpha_{ij}\}_{(i,j)\in\mathcal{E}}$ so that

$$\mathbf{z}^T\mathbf{Q}\mathbf{z} = \sum_{i:(i,j)\in\mathcal{E}} \alpha_{ij} \left(a_i z_i - a_j z_j\right)^2 - 2\sum_{i=1}^N a_i \frac{\partial g_{iT}}{\partial q_i} z_i^2 \tag{4.14}$$

where α_{ij} is a function of the entries in $\mathbf{J}$. Once (4.14) is proved, then $\forall \mathbf{z} \in \mathbb{R}^N \backslash \mathbf{0}$, $\mathbf{z}^T\mathbf{Q}\mathbf{z} < 0$ due to the existence of i such that $\frac{\partial g_{iT}}{\partial q_i}$ is negative.

Based on Lemma 4.6, there exists $\boldsymbol{\delta} \in \mathbb{R}^N_+$ so that $\mathbf{J}_0\boldsymbol{\delta} = \mathbf{0}$. Take $\mathbf{A} := \mathrm{diag}\{\delta_i^{-1}\}_{i=1}^N$, i.e. $a_i = \delta_i^{-1} > 0,\ \forall i = 1,\ldots,N$, and take

$$\alpha_{ij} := -\delta_i \frac{\partial g_{ij}}{\partial q_i} + \delta_j \frac{\partial g_{ij}}{\partial q_j},$$

which is negative under the condition $\frac{\partial g_{ij}}{\partial q_i} > 0,\ \frac{\partial g_{ij}}{\partial q_j} < 0$. Then based on (4.13), we can show the LHS of (4.14) is

$$\begin{aligned} \mathbf{z}^T\mathbf{Q}\mathbf{z} &= \sum_{i=1}^N Q_{ii} z_i^2 + 2\sum_{1\le i<j\le N} Q_{ij} z_i z_j \\ &= \sum_{i=1}^N \frac{2}{\delta_i} \left(\sum_{k:(k,i)\in\mathcal{E}} \frac{\partial g_{ki}}{\partial q_i} - \sum_{j:(i,j)\in\mathcal{E}} \frac{\partial g_{ij}}{\partial q_i} \right) z_i^2 \\ &\quad + 2\sum_{1\le i<j\le N} \left(\frac{1}{\delta_i}\frac{\partial g_{ij}}{\partial q_j} - \frac{1}{\delta_j}\frac{\partial g_{ij}}{\partial q_i} \right) z_i z_j - 2\sum_{i=1}^N a_i \frac{\partial g_{iT}}{\partial q_i} z_i^2 \\ &= \sum_{i=1}^N \frac{1}{\delta_i^2} \left(\sum_{k:(k,i)\in\mathcal{E}} \alpha_{ki} + \sum_{j:(i,j)\mathcal{E}} \alpha_{ij} \right) z_i^2 + 2\sum_{1\le i<j\le N} \alpha_{ij} z_i z_j - 2\sum_{i=1}^N a_i \frac{\partial g_{iT}}{\partial q_i} z_i^2 \end{aligned} \tag{4.15}$$

where the equivalence of the coefficients of z_i^2 for any i in the last equation is due to $\mathbf{J}_0\boldsymbol{\delta} = \mathbf{0}$. It is trivial to verify the coefficients of $\{z_i^2\}_{i=1}^N$ and $\{z_{ij}\}_{1\le i<j\le N}$ in (4.15)

match to the RHS of (4.14)[4]. Thus we obtain the proof.

4.5.2 Proof of Lemma 4.6

Proof. Note that $\mathbf{J}_0$ is not full rank as we can verify $\mathbf{1}^T\mathbf{J}_0 = \mathbf{0}$. Therefore there exists an eigenvalue 0. Denote the eigenvalues of $\mathbf{J}_0$ as $\{\lambda_i\}_{i=1}^N$, without loss of generality that they are sorted with non-increasing real part: $\mathrm{Re}(\lambda_1) \geq \mathrm{Re}(\lambda_2) \cdots \geq \mathrm{Re}(\lambda_N)$. Since $\mathbf{J}_0$ is diagonally dominant and all diagonal entries are negative, thus $\mathrm{Re}(\lambda_i) \leq 0$ by the Gershgorin circle theorem [109]. Therefore $\lambda_1 = 0$. Suppose $\lambda_i := r_i + \mathbf{j}u_i,\ \forall i \neq 1$ where $r_i \leq 0$.

Denote $\mathbf{J}_\theta = \mathbf{J}_0 + \theta\mathbf{I}$. Since all diagonal entries of $\mathbf{J}_0$ are negative while all the off-diagonal entries are non-negative. Then $\exists\theta > 0$ so that $\mathbf{J}_\theta$ is a matrix with all entries non-negative. Since the network is acyclic[5], it does not contain any strongly connected component. Therefore $\mathbf{J}_\theta$ is a irreducible non-negative matrix [112], to which the Perron-Frobenius theorem [112] can be applied: For such δ, there exists a real eigenvalue $r > 0$ such that (i) $\exists\mathbf{v} \in \mathbb{R}_+^N$ so that $\mathbf{J}_\delta\mathbf{v} = r\mathbf{v}$, and (ii) any other eigenvalue of $\mathbf{J}_\delta$ has smaller magnitude than r.

The i-th eigenvalue of $\mathbf{J}_\theta$, denoted as $\tilde{\lambda}_i$, equals to $\lambda_i + \theta$. Note that $\{\tilde{\lambda}_i\}_{i=1}^N$ are not necessarily sorted in non-decreasing real part, where $\tilde{\lambda}_1 = \theta$. Take θ so that the matrix is non-negative.

Step 1: Consider the case where $\mathbf{J}_0$ has no pure imaginary eigenvalues. In this case, $r_i < 0,\ \forall i \neq 1$. The eigenvalues of $\mathbf{J}_\theta$ is $\tilde{\lambda}_1 = \delta,\ \tilde{\lambda}_i = r_i + \theta + \mathbf{j}u_i$. It is easy to verify that by taking $\delta > \max_{i:i\neq 1}\left\{-2\frac{|\lambda_i|}{\mathrm{Re}(\lambda_i)}\right\}$, we can guarantee that $\theta = \tilde{\lambda}_1 > |\tilde{\lambda}_i|,\ \forall i \neq 1$, which means $\tilde{\lambda}_1 = \theta = r$. Thus the eigenvector, denoted as $\mathbf{v}$, of $\mathbf{J}_\theta$ associated with eigenvalue r, is positive. Therefore

$$\mathbf{J}_\theta\mathbf{v} = (\mathbf{J}_0 + \theta\mathbf{I})\,\mathbf{v} = r\mathbf{v} = \theta\mathbf{v}$$

Thus $\mathbf{J}_0\mathbf{v} = 0$. This $\mathbf{v}$ is the $\boldsymbol{\delta}$ that meet our condition.

[4]For $(i,j) \notin \mathcal{E}$, $g_{ij} \equiv 0$ and the coefficient of z_iz_j is 0.

[5]In fact it only requires the available paths of the commodity is acyclic.

Step 2: Prove that $\mathbf{J}_0$ does not contain pure imaginary eigenvalues. It is to show there is no $i \neq 1$ such that $r_i = 0$ and $u_i \neq 0$. We prove by contradiction. Suppose that there exists a pure imaginary eigenvalue. Then it is λ_2, and λ_3 is also pure imaginary which is the conjugate of λ_2. Then there exists θ such that $\mathbf{J}_\theta$ is non-negative, and $|\tilde{\lambda}_3| = |\tilde{\lambda}_2| > |\tilde{\lambda}_1| > |\tilde{\lambda}_i|,\ i \geq 4$. Then by Perron-Frobenius theorem, $\tilde{\lambda}_3 = \lambda_3 + \theta$ should be a real positive eigenvalue of $\mathbf{J}_\theta$, which contradicts that λ_3 is pure imaginary. Therefore we get the proof. $\square$

4.5.3 Example of Lemma 4.1

For the policy (4.5) based on backpressure, if we suppose that

$$\lambda_i + \sum_{k:(k,i)\in E} c_{ki} \leq \sum_{j:(i,j)\in E} c_{ij} + c_{iT},$$

then we can set $\underline{b}_i = 0,\ \forall i \in V$. We can then set the values of $\{\bar{b}_i\}_{i=1}^n$ such that (i) $\bar{b}_j > \bar{b}_k$ for any link (j,k), and (ii) $\bar{b}_i \leq b_i - \delta_i$ for any node i with finite buffer, where δ_i is a positive constant close to 0. We can verify that for $\dot{q}_i = f_i(\mathbf{q})$, when $q_i = \underline{b}_i = 0$, then

$$\begin{aligned} f_i(\mathbf{q}) &= \lambda_i + \sum_{k:(k,i)\in E} g_{ki}(q_k, 0) - \sum_{j:(i,j)\in E} g_{ij}(0, q_j) - g_{iT}(0) \\ &= \lambda_i + \sum_{k:(k,i)\in E} g_{ki}(q_k, 0) \geq 0 \end{aligned}$$

and when $q_i = \bar{b}_i$, note that for any node j such that $(i,j) \in E$, $g_{ij}(\bar{b}_i, q_j) = c_{ij}$ since $q_j \in [0, \bar{b}_j] \in [0, b_j) \cap [0, \bar{b}_i)$, therefore

$$\begin{aligned} f_i(\mathbf{q}) &= \lambda_i + \sum_{k:(k,i)\in E} g_{ki}(q_k, \bar{b}_i) - \sum_{j:(i,j)\in E} g_{ij}(\bar{b}_i, q_j) - g_{iT}(\bar{b}_i) \\ &\leq \lambda_i + \sum_{k:(k,i)\in E} c_{ki} - \sum_{j:(i,j)\in E} c_{ij} - c_{iT} \leq 0 \end{aligned}$$

Therefore the conditions in Lemma 4.1 holds, which implies the policy (4.5) can ensure a unique stable equilibrium point, and thus render the networks to be queue length stable.

4.5.4 Necessity for Queue Length Stability

We have proposed the sufficient condition for the policies to stabilize the networks. Here we add more discussion over the necessary condition. We first pose a toy example that *the sufficient condition is not necessary.* Consider a two node system where the queue dynamics[6] are

$$\begin{cases} \dot{q}_1 = -g_{12}(q_1, q_2) = -(q_1^2 + q_2^2 - 5)(q_1^2 + q_2^2 - 1) \\ \dot{q}_2 = g_{12}(g_1, g_2) = (q_1^2 + q_2^2 - 5)(q_1^2 + q_2^2 - 1) \end{cases}.$$

The set of equilibrium points is $\{(q_1, q_2) : q_1^2 + q_2^2 = 1 \cup q_1^2 + q_2^2 = 5\}$. Inside, the stable points can be obtained by the two eigenvalues of the Jacobian matrix J, where $\lambda_1 = 0$ and $\lambda_2 = 4(q_2 - q_1)(q_1^2 + q_2^2 - 3)$. The set of stable points, denoted by S, contains all the queue length vector (q_1, q_2) such that that $\lambda_2 \leq 0$. The set of stable points obtained from our proposed sufficient condition in Lemma 4.1, denoted by S', is

$$\{\frac{\partial g_{12}}{\partial q_1} = q_1(q_1^2 + q_2^2 - 3) \geq 0,\ \frac{\partial g_{12}}{\partial q_2} = q_2(q_1^2 + q_2^2 - 3) \leq 0\}.$$

It can be easily verified that S' is a proper subset of S, where point $(q_1, q_2) = (1/2, \sqrt{3}/2)$ belongs to S but not to S'.

Although not acquiring explicit necessary conditions, here we propose a potential idea to solve this problem based on the concept of *unstable manifold* for an equilibrium point [113], which can be interpreted as the set of points in the space, once given as the initial point, that achieve the equilibrium point when taking $t \to -\infty$. Similarly the *stable manifold* is defined with only difference at taking $t \to \infty$. For example, in the 2-dim system $\dot{q}_1 = -q_1,\ \dot{q}_2 = q_2 + q_1^2$, the unique equilibrium point is $(0, 0)$ which

[6]We can translate the origin such that the queue length are nonnegative.

is a saddle point. The stable manifold of it is $\{(q_1, q_2)|q_2 = -q_1^2/3\}$ while the unstable manifold is $\{(q_1, q_2)|q_1 = 0\}$. In this system, we can observe that $q_2 \to \infty$ primarily result from the unstable manifold $\{(q_1, q_2)|q_1 = 0\}$ being unbounded, therefore we may naturally conjecture that once the policy guarantees the unstable manifold of any equilibrium point to be bounded, we can ensure network stability as the queue length is prevented from going to infinity. We formalize this as a conjecture that may enlighten the future exploration over this problem.

Conjecture 4.1. *A necessary condition for the policy to achieve queue length stability is that it can ensure that for every equilibrium point, its unstable manifold is bounded.*

Chapter 5

Routing Attack on Network Overload with Static Routing

In this chapter, we consider a network where a subset of the nodes have been hijacked and is subject to a routing attack, and quantify the ability of network adversaries to induce network overload through a routing attack. We develop routing attack strategies for two objectives related to overload: no-loss throughput minimization and loss maximization. The first objective attempts to identify a routing attack strategy that minimizes the network's throughput that is guaranteed to survive, and the second objective simply attempts to maximize the throughput loss. We propose a polynomial-time routing attack algorithm that minimizes no-loss throughput for general multi-hop networks. In contrast, we demonstrate that finding the optimal routing attack strategy for loss maximization is NP-complete even in single-hop networks. We develop two approximation algorithms with multiplicative and additive guarantees respectively for single-hop networks, and validate the near-optimal performance of the proposed algorithms over a wide range of network settings. Our results quantitatively confirm the significant threat posed by routing attacks, and demonstrate that our proposed algorithms can be used as benchmarks to quantify the overload risk given arbitrary sets of hijacked nodes and to identify the critical nodes that should be shielded against routing attacks.

5.1 Network Models and Problem Definition

In this section, we introduce the multi-hop and single-hop network models, along with basic definitions including routing and attack policies. We then define the problems of finding the optimal routing attack to minimize no-loss throughput and maximize loss.

5.1.1 Network Models

Multi-hop network

We define a multi-hop network as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of nodes and $\mathcal{E}$ denotes the set of edges, with $|\mathcal{V}| = N$ and $|\mathcal{E}| = M$. Node indices are $1, 2, \cdots, N$, and $(i, j) \in \mathcal{E}$ if there is a link from node i to j. Denote the traffic flow over link (i, j) by f_{ij}. Each link has capacity c_{ij} which is the maximum transmission rate over (i, j), i.e., $f_{ij} \in [0, c_{ij}]$. We treat (i, j) and (j, i) as different links. We start by considering a single commodity, with node 1 as the source and N as the destination.

The network traffic may have multiple available paths from node 1 to N, where each network node dispatches the traffic to its connected nodes. We define the dispatch ratio vector at a non-destination node i as a vector $\mathbf{x}_i \in \mathbb{R}^N$, where each element x_{ij} denotes the fraction (ratio) of traffic sent from node i to its connected node j among the total traffic at node i, and $x_{ij} = 0$ for $(i, j) \notin \mathcal{E}$, and $\sum_{j=1}^{N} x_{ij} = 1$. We call $\mathbf{x}_i$ the *routing policy* at node i, and $\mathbf{X} = [\mathbf{x}_i]_{i=1,\cdots,N}$ denotes the *routing matrix* of the network[1]. We validate this definition of routing policy through Proposition 5.1, which shows the equivalence between the above dispatch ratio characterization and the common multi-path characterization [19]. Henceforth, we will solely use the dispatch ratios to model routing. We give an example of a routing policy in Fig. 5-2(a) where at source node 1, $(x_{12}, x_{13}) = (0.5, 0.5)$.

Proposition 5.1. *Suppose there are P paths $\{Path_p\}_{p=1}^{P}$, where a fraction θ_p*

[1] $\mathbf{x}_N = \mathbf{0}$. We put node N in to keep the dimension equal to network size.

of the traffic takes $Path_p$. The corresponding routing matrix $\mathbf{X}$ satisfies $x_{ij} = \beta_{ij}/\sum_{k:(i,k)\in\mathcal{E}}\beta_{ik},\ \forall(i,j)\in\mathcal{E}$, where $\beta_{ij} = \sum_{p:(i,j)\in Path_p}\theta_p$, and $x_{ij} = 0,\ \forall(i,j)\notin\mathcal{E}$, where $(i,j)\in Path_p$ if link (i,j) is on $Path_p$.

We can characterize the *overload* at a link (i,j) using the definition of routing policy. If the traffic to be sent through (i,j) exceeds c_{ij}, i.e., $\left(\sum_{k:(k,i)\in\mathcal{E}} f_{ki}\right) x_{ij} > c_{ij}$, then link (i,j) will be *saturated*, i.e., $f_{ij} = c_{ij}$, and we say that overload occurs at (i,j), where the excess traffic $\left(\sum_{k:(k,i)\in\mathcal{E}} f_{ki}\right) x_{ij} - c_{ij}$ will be dropped, and lost.

We define the *upstream* and *downstream* node set used in algorithm design. The upstream node set of a node i, denoted by $\mathcal{V}_i^{up}$, contains all the nodes except i that have a path in the available path set to transmit traffic to i under the routing matrix $\mathbf{X}$. The downstream node set of a node i, denoted by $\mathcal{V}_i^{down}$, contains all the nodes except i to which there exists a path in the available path set starting from i. We define a node $j\in\mathcal{V}_i^{down}$ to be a connected downstream node of i if further $(i,j)\in\mathcal{E}$. We define a link (j,k) to be a downstream link of node i if $j,k\in\mathcal{V}_i^{down}\cup\{i\}$, and a connected downstream link of node i if further $j=i$. Note that $\mathcal{V}_i^{up}\cap\mathcal{V}_i^{down}=\emptyset$ holds in directed acyclic networks, while there may exist nodes that are simultaneously the upstream and downstream node of a node i in general networks. The framework and analysis in this work hold for general networks. However, for ease of understanding we use examples with $\mathcal{V}_i^{up}\cap\mathcal{V}_i^{down}=\emptyset$ for explanation, as in Fig. 5-2(a) we have $\mathcal{V}_5^{up}=\{1,3\}$ and $\mathcal{V}_2^{down}=\{4,6\}$.

Single-hop network

A single-hop network contains a set of source (ingress) nodes and destination (egress) nodes. Examples include server farms, switched networks, and the basic structure of data center networks like Fat-Tree [17] and Clos [1], as shown in Fig. 5-1(a) and (b). We show in Section 5.3 that we can obtain theoretical performance guarantees for our proposed attack algorithms under single-hop networks.

A $N_s\times N_D$ single-hop network is a bipartite graph $\mathcal{G}=(\mathcal{V},\mathcal{E})$ with $\mathcal{V}:=\{\mathcal{V}_S,\mathcal{V}_D\}$, where $\mathcal{V}_S$ and $\mathcal{V}_D$ represent the set of source and destination nodes respectively, and $\mathcal{E}$ denotes the set of links between $\mathcal{V}_S$ and $\mathcal{V}_D$, and $|\mathcal{V}_S|=N_S$ and $|\mathcal{V}_D|=N_D$. Denote

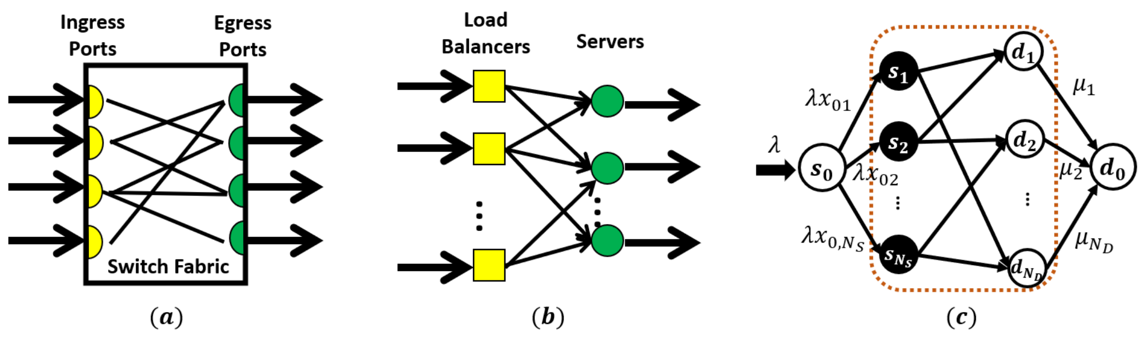


Figure 5-1: (a) Switch network; (b) Server farm; (c) Bipartite graph (in the dashed box) with meta source s_0 and destination d_0

the ith source node by s_i and jth destination node by d_j. Each traffic flow is injected into one of the source nodes and departs from one of the destination nodes. We use slightly different notations and objectives in single-hop networks as follows. Denote the traffic arrival rate at source s_i by λ_i, the service rate of destination d_j by μ_j, and the transmission rate over link (s_i, d_j) by f_{ij}, with their corresponding vector forms being $\boldsymbol{\lambda} := \{\lambda_i\}_{i=1}^{N_S}$, $\boldsymbol{\mu} := \{\mu_j\}_{j=1}^{N_D}$, and $\mathbf{f} = \{f_{ij}\}_{(i,j)\in\mathcal{E}}$, respectively. Denote the routing policy at s_i by $\mathbf{x}_i = \{x_{ij}\}_{d_j\in\mathcal{V}_D}$. We consider sufficient capacity for links between $\mathcal{V}_S$ and $\mathcal{V}_D$ so that they will not be saturated, with a primary focus on how a routing attack at $\mathcal{V}_S$ affects overload at $\mathcal{V}_D$ [18, 29]. Overload occurs at $d_j \in \mathcal{V}_D$ if $\sum_{i=1}^{N_S} \lambda_i x_{ij} > \mu_j$ under routing matrix $\mathbf{X}$.

We can transform the single-hop structure to a single commodity starting from a meta-source s_0, connected to each source in $\mathcal{V}_S$, to a meta-destination d_0 that receives traffic from each destination in $\mathcal{V}_D$, by setting the traffic arrival rate to s_0 to be $\lambda = \sum_{i=1}^{N_S} \lambda_i$, the dispatch ratio from s_0 to s_i to be $x_{0i} = \lambda_i/\lambda$, and the capacity of link (d_j, d_0) to be μ_j. It is straightforward to see that the transformed graph is equivalent to the bipartite graph, shown in Fig. 5-1(c).

Remark: We focus on static parameters in this work. We anticipate that the methodology we propose will inspire addressing time-varying network parameters in future work.

5.1.2 Problem Definition

We consider a network adversary who hijacks and gains control over the routing decisions of certain network nodes, $\mathcal{V}_A \subseteq \mathcal{V}$. We call $\mathcal{V}_A$ the set of *adversarial nodes*, and $\mathcal{V}_N := \mathcal{V} \backslash \mathcal{V}_A$ the set of *normal nodes.* The adversary aims to find the routing policies at adversarial nodes, denoted by $\mathbf{x}_A := \{\mathbf{x}_i\}_{i \in \mathcal{V}_A}$, to maximize overload. We consider *fixed* routing policies at normal nodes $\mathcal{V}_N$ (*default routing*) that do not change after the adversary alters the routing of nodes in $\mathcal{V}_A$. We assume that the adversary has access to the default routing at normal nodes[2]. In the example in Fig. 5-2(a), $\mathcal{V}_A = \{3\}$ and $\mathcal{V}_N = \{1, 2, 4, 5, 6\}$. The whole numbers denote link capacities and the highlighted fraction over a link denotes the dispatch ratio. We use shaded nodes to denote adversarial nodes and unshaded nodes to denote normal nodes.

We optimize routing attacks for two objectives related to overload given $\mathcal{V}_A$. (i) **Minimize No-Loss Throughput**: We define *no-loss throughput*, denoted by λ^*, as the maximum traffic arrival rate λ at the source node that will not lead to link overload given the routing matrix $\mathbf{X}$. Equivalent interpretations of λ^* include the maximum traffic arrival rate that can be successfully transmitted [114], the max arrival rate that guarantees maximum link utilization below 100% [92], and the max arrival rate that ensures queue stability [18]. Essentially, λ^* captures the network robustness to routing attacks. We investigate the capability of routing attack over $\mathcal{V}_A$ to minimize λ^*, i.e., minimizing network robustness to overload. The minimum λ^*, denoted by λ^*_{OPT}, can be interpreted as the upper limit of traffic arrival rate that guarantees no traffic loss under any adversarial routing attack on the given $\mathcal{V}_A$. (ii) **Maximize Loss**: Given the default routing policies at normal nodes $\mathcal{V}_N$ and the arrival rate λ, the adversary manipulates routing in $\mathcal{V}_A$ to maximize traffic loss caused by link overload, equivalent to minimize total traffic transmitted to the destination. Loss maximization reflects the most severe overload that can be caused via a routing attack. These two objectives reflect different facets of network overload, and we show in Fig. 5-2(b)

[2]As traffic flow is considered static, the dispatch ratio from node i to j can be estimated via $f_{ij} / \sum_{k:(k,i) \in \mathcal{E}} f_{ki}$ given that there is no overload over any link starting from i, which is common when no attack is conducted.

and 5-2(c) that their optimal routing attack policies are different, motivating our algorithm designs in Section 5.2 and 5.3. In summary, we give the formal statement of the core problem of this work: *Given* $\mathcal{G}$*, capacity* $\{c_{ij}\}_{(i,j)\in\mathcal{E}}$*, default routing at normal nodes* $\{\mathbf{x}_i\}_{i\in\mathcal{V}_N}$*, and adversarial nodes* $\mathcal{V}_A$*, what is the optimal routing attack policy over* $\mathcal{V}_A$ *to minimize no-loss throughput and maximize loss?*

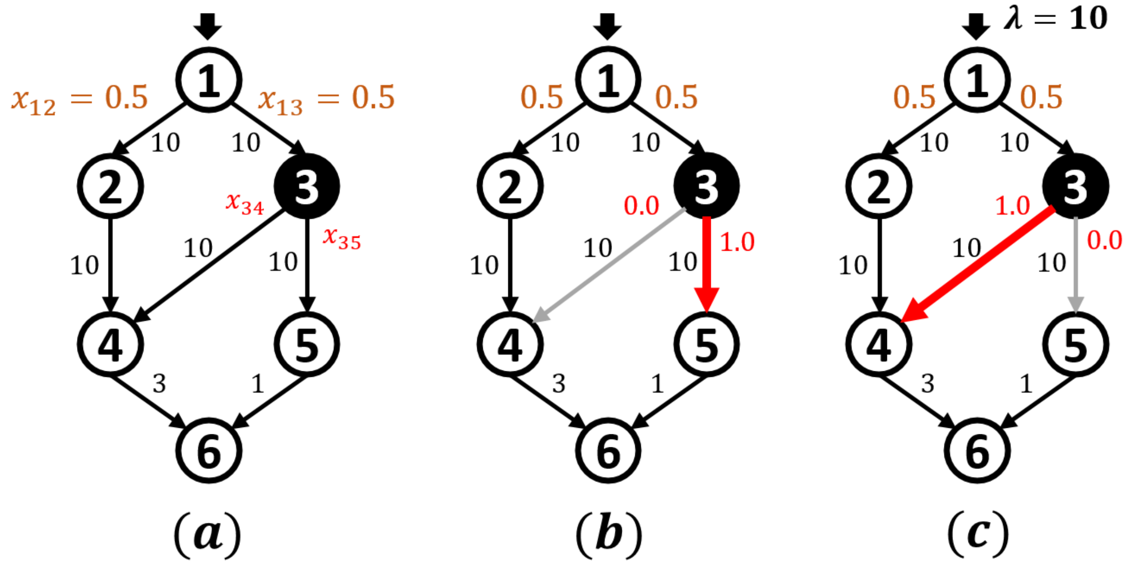


Figure 5-2: (a) A 6-node network with $\mathcal{V}_A = \{3\}$ and $\mathcal{V}_N = \{1, 2, 4, 5, 6\}$; (b) Optimal routing to minimize λ^* is $(x_{34}, x_{35}) = (0, 1)$: λ^*_{OPT} is 2 and $(5, 6)$ is the first saturated link; (c) Given $\lambda = 10$, the optimal routing to maximize loss is $(x_{34}, x_{35}) = (1, 0)$, with maximum loss of $\lambda - c_{46} = 10 - 3 = 7$.

Furthermore, in Section 5.4 we investigate the *optimal node selection* problem: The adversary needs to identify the optimal nodes to hijack from a set of candidate nodes to conduct routing attack so that it can minimize no-loss throughput or maximize loss. This problem fits into the practice where the adversary can only hijack and manipulate the routing polices over a limited number of nodes due to the cost or the risk of being exposed. The answers to this problem inform network service providers of the critical nodes that should be protected against routing attacks whose control by the adversary can lead to high risk of overload and traffic loss.

5.2 No-Loss Throughput Minimization

In this section, we investigate the optimal routing attack to minimize no-loss throughput λ^*. We develop an exact polynomial-time algorithm that returns an optimal routing attack based on linear programming (LP), given the global information of the network topology, link capacities, and default routing at $\mathcal{V}_N$. We tehn study the case where the adversary only has access to the routing information downstream to the adversarial nodes $\mathcal{V}_A$, and propose a routing attack algorithm with an approximation ratio of at most 2. We further demonstrate that the proposed algorithms can be extended to more practical settings, including routing attack constraints, multiple commodities, and distributed heuristics. The key takeaway is that the adversary can execute routing attacks efficiently to achieve optimal performance in diminishing the network's robustness against overload in general network settings given arbitrary sets of $\mathcal{V}_A$, which quantitatively unveils the threat of routing attacks in increasing the risk of overload.

5.2.1 Problem Formulation

We formulate the λ^*-minimization problem as follows.

$$\begin{aligned} \lambda^*_{OPT} := \min_{\mathbf{f}} \quad & \max_{\lambda} \quad \lambda \\ \text{s.t.} \quad & \mathbf{f} \in \Lambda, \quad \lambda = f_{01}, \quad f_{ij} \in [0, c_{ij}], \forall (i,j) \in \mathcal{E} \end{aligned} \tag{5.1}$$

where the constraints $\Lambda := \Lambda_N \cap \Lambda_A$ and

$$\begin{cases} \Lambda_N : f_{ij} = \left(\sum_{k:(k,i)\in\mathcal{E}} f_{ki}\right) x_{ij}, \forall i \in \mathcal{V}_N,\ \forall (i,j) \in \mathcal{E} \\ \Lambda_A : \sum_{j:(i,j)\in\mathcal{E}} f_{ij} = \sum_{k:(k,i)\in\mathcal{E}} f_{ki}, \forall i \in \mathcal{V}_A \end{cases} . \tag{5.2}$$

In (5.1), we introduce a meta source node 0 that serves traffic flows only to the original source node 1 with $f_{01} = \lambda$ and $c_{01} \geq \lambda$, so that node 1 can be either a normal or adversarial node. Note that λ is a variable instead of a given parameter. The decision variables of (5.1) are the flow variables $\mathbf{f} := \{f_{ij}\}_{(i,j)\in\mathcal{E}}$. In constraints

(5.2), Λ_N represents the flow conservation at any normal node $i \in \mathcal{V}_N$, which has fixed default routing x_{ij} through any $(i,j) \in \mathcal{E}$. The flow on (i,j) is equal to the total traffic injection to node i multiplied by the dispatch ratio x_{ij} from i to j. Λ_A represents the flow conservation at any adversarial node $i \in \mathcal{V}_A$, where the adversary can manipulate routing arbitrarily subject to flow conservation. The optimal solution to (5.1), denoted by $\mathbf{f}^*$, leads to the optimal routing attack, denoted by $\mathbf{x}_A^* := \{\mathbf{x}_i^*\}_{i \in \mathcal{V}_A}$, where $x_{ij}^* = f_{ij}^* / \sum_{k:(k,i)\in\mathcal{E}} f_{ki}^*$ for $(i,j) \in \mathcal{E}$. In the above formulation, we use $\mathbf{f}$ instead of $\mathbf{x}_A$ as decision variables in order to formulate the constraints in linear form.

However, note that the minimax problem (5.1) cannot be formulated as a standard LP problem. We thus transform (5.1) into the following equivalent optimization framework (5.3) that paves the way for polynomial-time algorithm design.

$$\begin{aligned} \max_{\mathbf{f}} \quad & \max_{(i,j)\in\mathcal{E}} \quad f_{ij}/c_{ij} \\ \text{s.t.} \quad & \mathbf{f} \in \Lambda, \quad f_{01} = 1, \quad f_{ij} \geq 0, \ \forall (i,j) \in \mathcal{E}. \end{aligned} \tag{5.3}$$

Proposition 5.2. *The optimal solutions* $\mathbf{f}^*$ *to* (5.1) *and* (5.3) *are equivalent. The minimum no-loss throughput* λ_{OPT}^* *is given by* $(\max_{(i,j)\in\mathcal{E}} f_{ij}^*/c_{ij})^{-1}$ *from* (5.3).

Proof. In (5.1), define new variables $\tilde{f}_{ij} := f_{ij}/\lambda, \forall (i,j) \in \mathcal{E}$. Then the optimization in (5.1) is equivalent to

$$\begin{aligned} \min_{\tilde{\mathbf{f}}} \quad & \max_{\lambda} \quad \lambda \\ \text{s.t.} \quad & \tilde{\mathbf{f}} \in \Lambda, \quad \tilde{f}_{01} = 1, \quad \tilde{f}_{ij} \in [0, c_{ij}/\lambda], \ \forall (i,j) \in \mathcal{E}. \end{aligned} \tag{5.4}$$

We have $\lambda \leq c_{ij}/\tilde{f}_{ij}$ for each link, and thus $\lambda \leq \min_{(i,j)\in\mathcal{E}} c_{ij}/\tilde{f}_{ij}$. Note that this is the only constraint for λ, and λ is an independent decision variable after the transformation. Therefore the maximum λ that guarantees no loss is equal to $\min_{(i,j)\in\mathcal{E}} c_{ij}/\tilde{f}_{ij}$, and the objective function becomes $\min_{\tilde{\mathbf{f}}} \min_{(i,j)\in\mathcal{E}} c_{ij}/\tilde{f}_{ij} = \max_{\tilde{\mathbf{f}}} \max_{(i,j)\in\mathcal{E}} \tilde{f}_{ij}/c_{ij}$ which is exactly (5.3) by changing notation $\tilde{f}_{ij}$ to f_{ij}. $\square$

The formulation (5.3) lets $f_{01} = 1$, i.e., one unit of traffic arrival rate into the networks[3], and the goal is to find the optimal routing policy that maximizes the maximum link utilization, which is $\max_{(i,j)\in\mathcal{E}} f_{ij}/c_{ij}$. Then the minimum possible λ^* under an optimal routing attack is $(\max_{(i,j)\in\mathcal{E}} f^*_{ij}/c_{ij})^{-1}$ where $\mathbf{f}^*$ is the optimal solution to (5.3). The intuition is that the link with the highest utilization is the first saturated link when the arrival rate increases gradually from 0. The critical point of Proposition 5.2 is that the new link-wise maximization formulation (5.3) inspires the following design of polynomial-time algorithms.

5.2.2 Exact Algorithms

We propose exact algorithms to minimize no-loss throughput λ^* based on (5.3). We derive a simple brute-force algorithm which is polynomial under $|\mathcal{V}_A| = O(1)$, and a polynomial-time algorithm under general $\mathcal{V}_A$ based on solving an LP for maximum flow.

Brute-Force under $|\mathcal{V}_A| = O(1)$

Based on (5.3), we prove Theorem 6.1 below, which states that there must exist an optimal routing policy that each adversarial node $i \in \mathcal{V}_A$ dispatches all traffic flows to a single downstream connected node.

Theorem 5.1 (Boundary Optimality). *There exists an optimal solution* $\mathbf{f}^*$ *to* (5.3) *such that for* $\forall i \in \mathcal{V}_A$, $\exists j$ *that* $(i,j) \in \mathcal{E}$ *and* $x^*_{ij} = \frac{f^*_{ij}}{\sum_{k:(i,k)\in\mathcal{E}} f^*_{ik}} = 1$, *while* $x^*_{ik} = 0$ *for* $k \neq j$.

Proof. The objective function of problem (5.3) is convex with respect to the flow variables $\mathbf{f} = \{f_{ij}\}_{(i,j)\in\mathcal{E}}$ due to the convexity of the elementise maximum function. Meanwhile the constraints are all linear which form a polytope. Therefore (5.3) is a convex maximization problem, where at least one of the optimal solutions is at one of the vertices of the polytope. There is a total of $M+1$ decision variables (including f_{01}). Denote the number of links starting from $\mathcal{V}_N \cup \{0\}$ as M_N, and links starting

[3] We can choose arbitrary constant for f_{01} as we evaluate the saturation level and thus no constraint on $f_{ij} \in [0, c_{ij}]$.

from $\mathcal{V}_A$ as M_A. Then the total number of equality constraints in (5.3) is $M_N + |\mathcal{V}_A|$, and thus at any vertex there should have at least $M + 1 - M_N - |\mathcal{V}_A| = M_A - |\mathcal{V}_A|$ flow variables f_{ij} over link (i,j) to be zero. Each $i \in \mathcal{V}_A$ routing traffic to a single downstream adjacent node guarantees the above condition, i.e., $\exists j$ that $(i,j) \in \mathcal{E}$ and $x^*_{ij} = \frac{f^*_{ij}}{\sum_{k:(i,k)\in\mathcal{E}} f^*_{ik}} = 1$, while $x^*_{ik} = 0$ for $k \neq j$. $\square$

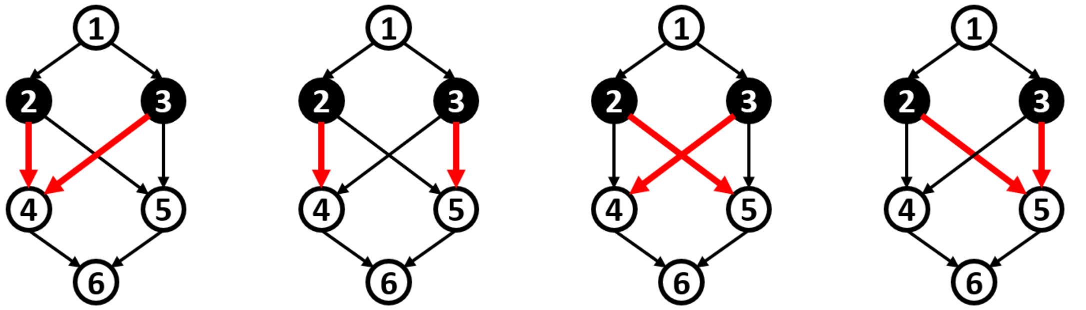


Figure 5-3: Boundary Optimality: One of the 4 combinations must be optimal, where $\mathcal{V}_A = \{2,3\}$ serve all traffic through the highlighted links.

The intuition of Theorem 6.1 is that (5.3) maximizes a convex function over a polytope, hence one of the vertices of the polytope must be the optimal solution. The implication of Theorem 6.1 is that the optimal attack to minimize λ^* can be identified in a brute force manner by exhausting all the combinations where each adversarial node sends all the traffic to one of its connected downstream nodes. The upper bound on the number of combinations is $d_{\max}^{|V_A|}$, where $d_{\max}$ denotes the maximum number of connected downstream links of a node. Note that $d_{\max} = O(N)$ and thus $d_{\max}^{|V_A|}$ is a polynomial function of network size N when $|V_A| = O(1)$. We give an example with $|\mathcal{V}_A| = 2$ in Fig. 5-3.

For each of the above combinations of routing attacks over $\mathcal{V}_A$, we can calculate λ^* given the routing matrix $\mathbf{X}$ by a two-step process: First, for each $(i,j) \in \mathcal{E}$, calculate the proportion of the traffic starting from the source node that will go through this link, under the assumption that all link capacities are infinite. The flow variable f_{ij} given $\lambda = 1$ under this setting is exactly the proportion of traffic that flows over link $(i,j) \in \mathcal{E}$. In general, this can be done by solving the linear equations built via the first constraint in (5.2) given the routing policy at each node i, with worst time complexity $O(M^3)$. For directed acyclic networks, the time complexity can be reduced to $O(M)$ by assuming $\lambda = 1$ and following $f_{ij} = \left(\sum_{k:(k,i)\in\mathcal{E}} f_{ki}\right) x_{ij}$ over all the M links. Second, resume the original link capacities, and find the link $(i^*, j^*) = \arg\max_{(i,j)\in\mathcal{E}} f_{ij}/c_{ij}$ in $O(M)$ time. Then $\lambda^* = (f_{i^*j^*}/c_{i^*j^*})^{-1}$. The limitation of the brute-force algorithm is the exponential time complexity under general $\mathcal{V}_A$ with $|\mathcal{V}_A| = O(N)$.

MaxFlow-based Solution for General $\mathcal{V}_A$

We demonstrate that there exists a polynomial algorithm that outputs the optimal routing attack to (5.3) under general $\mathcal{V}_A$, presented in Algorithm 5.1. First, we figure out the routing policy at $\mathcal{V}_A \cap \mathcal{V}_i^{up}$ that maximizes the total flow to each node i (except destination), assuming arrival rate $\lambda = 1$ at the source and infinite capacity over all the links. Denote the corresponding maximum possible flow to node i by $MF[i]$. $MF[1] = 1$ trivially, while for $i > 1$, $MF[i]$ can be obtained by solving an LP that outputs the maximum flow solution. Second, for each non-destination node i, we identify the first link that can be saturated under routing attack among all the connected downstream links of node i as follows. If $i \in \mathcal{V}_N$, the first saturated link starting from i must be $(i, j_i^*) = \arg\max_{(i,j)\in\mathcal{E}} x_{ij}/c_{ij}$; If $i \in \mathcal{V}_A$, the adversary can arbitrarily alter the routing at i, thus the first-saturated link starting from i is $(i, j_i^*) = \arg\min_{(i,j)\in\mathcal{E}} c_{ij}$. It is straightforward to see that the first link that will be saturated under the optimal routing attack given by (5.3) must be among the $N-1$ first-saturated links $\{(i, j_i^*)\}_{i=1}^{N-1}$. Combining the two steps as done in Algorithm 5.1 establishes its correctness to solving (5.3), which outputs the routing attack solution with minimum traffic arrival rate required to saturate one of the $N-1$ first-saturated links, and the resulting minimum arrival rate is λ^*_{OPT} in (5.3). We state this result in Theorem 5.2. We give a toy example running Algorithm 5.1 in Fig. 5-4.

Theorem 5.2. *Algorithm 5.1 outputs the optimal solution to* (5.3).

The computation cost of Algorithm 5.1 mainly lies in solving LPs to obtain $MF[i]$ for each non-destination node i. We can apply interior point methods to solve LPs

Algorithm 5.1: Exact Algorithm to Minimize λ^* for General $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\mathcal{V}_A$

```
1 Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V}_A$, default routing $\{\mathbf{x}_i\}_{i \in \mathcal{V}_N}$;
2 for $\forall i \in \mathcal{V} \backslash \{N\}$ do
3     Given one unit of arrival while assuming unlimited capacity, calculate
        $MF[i]$, the max flow to node $i$;
4     if $i \in \mathcal{V}_N$ then $V[i] = MF[i] \times x_{ij^*}/c_{ij^*}$ with
        $(i, j^*) = \arg\max_{(i,j) \in \mathcal{E}} x_{ij}/c_{ij}$;
5     else $V[i] = MF[i]/c_{ij^*}$ with $(i, j^*) = \arg\min_{(i,j) \in \mathcal{E}} c_{ij}$;
6 Find $i^* = \arg\max_{i \in \mathcal{V}} V[i]$;
7 Construct the routing attack at $\forall i \in \mathcal{V}_A \cap (\mathcal{V}_{i^*}^{up} \cup \{i^*\})$, corresponding to
    node $i^*$, denoted by $\mathbf{x}_i^{(i^*)}$;
8 For all $i \in \mathcal{V}_A$, let $\mathbf{x}_i = \mathbf{x}_i^{(i^*)}$ if $i \in \mathcal{V}_{i^*}^{up} \cup \{i^*\}$ else arbitrarily set the routing
    policy $\mathbf{x}_i$;
9 Return $\mathbf{x}_A = \{\mathbf{x}_i\}_{i \in \mathcal{V}_A}$;
```

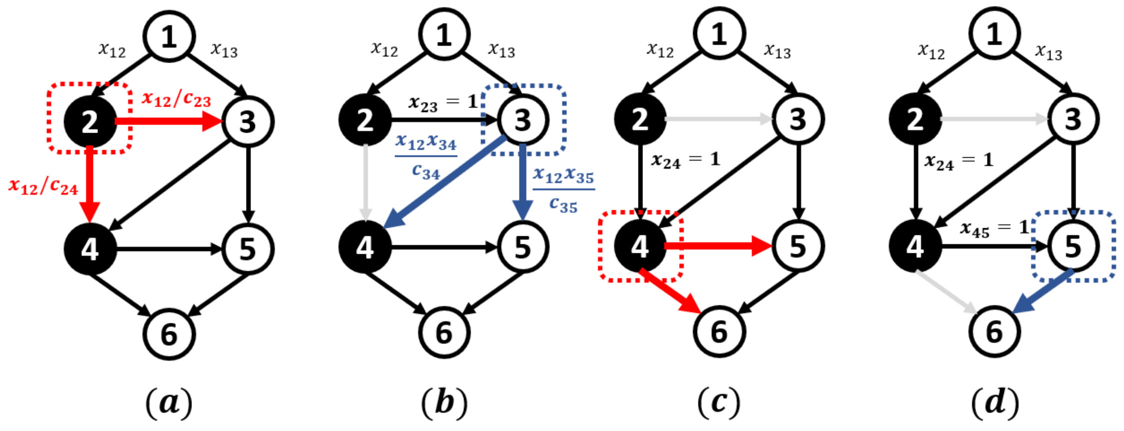


Figure 5-4: Example of Algorithm 5.1. Assume that $c_{12}, c_{13} \to \infty$ which means $(1, 2)$ and $(1, 3)$ will not be the first saturated links. We can calculate $MF[2] = x_{12}$, $MF[3] = 1$ since the adversarial node 2 can route all packets to 3, $MF[4] = x_{12} + x_{13}x_{34}$ since node 2 can route all packets to 4, and $MF[5] = 1$ since node 4 can route all packets to 5. Then we find the first saturated connected downstream link of each node in $\{2, 3, 4, 5\}$, where the calculation in (a) and (c) follows step 5 since $2, 4 \in \mathcal{V}_A$ with links highlighted in red, while that in (b) and (d) follows step 4 since $3, 5 \in \mathcal{V}_N$ highlighted in blue.

in polynomial time.

5.2.3 2-Approximation Algorithm with Partial Information

We extend to the situation where each adversarial node $i \in \mathcal{V}_A$ only has the access to the routing information of its downstream nodes in $\mathcal{V}_i^{down}$. This is realistic as routers may only store the topology and routing information downstream. We demonstrate an interesting result that even with such partial information, there exists a 2-approximation algorithm for minimizing λ^*, which means λ^*_{ALG}, the no-loss throughput under the routing attack given by the algorithm, satisfies $\lambda^*_{ALG} \leq 2\lambda^*_{OPT}$, where λ^*_{OPT} is the no-loss throughput under the optimal routing attack with complete routing information.

We propose the approximation algorithm in Algorithm 5.2. The core idea is to decompose the traffic flows into two sets: flows that pass some $i \in \mathcal{V}_A$, and those not passing $\forall i \in \mathcal{V}_A$. Then it constructs the routing attack based on the first set of flows, which we show can be derived purely based on the topology and routing information downstream to $\mathcal{V}_A$ by Algorithm 5.3, explained soon below. In detail, Algorithm 5.2 contains three steps: (i) Construct the downstream subgraph starting from $\mathcal{V}_A$, denoted by $\mathcal{G}_A^{down} = (\mathcal{V}_A^{down}, \mathcal{E}_A^{down})$, where $\mathcal{V}_A^{down} = \mathcal{V}_A \cup \{\mathcal{V}_i^{down}\}_{i \in \mathcal{V}_A}$ and $\mathcal{E}_A^{down} = \{(i,j) \in \mathcal{E} \mid i,j \in \mathcal{V}_A^{down}\}$; (ii) For each $i \in \mathcal{V}_A$, calculate the ratio of traffic flows that are transmitted from the source node to node i without passing any other adversarial node $j \in \mathcal{V}_A \backslash \{i\}$, denoted by $R[i]$, based on Algorithm 5.3; (iii) Find the optimal routing of $\mathcal{V}_A$ to minimize λ^* in $\mathcal{G}_A^{down}$ with $R[i]$ units of arrival to $i \in \mathcal{V}_A$.

Algorithm 5.2: 2-Approximation algorithm to minimize λ^* based on $\mathcal{G}_A^{down}$

```
1 Input: $\mathcal{V}_A$, $\mathcal{G}_A^{down}$, normal routing $\{\mathbf{x}_i\}_{i \in \mathcal{V}_N \cap \mathcal{V}_A^{down}}$;
2 Determine $R[i], \forall i \in \mathcal{V}_A$ by Algorithm 5.3;
3 Add $R[i]$ units of arrival rate to node $i \in \mathcal{V}_A$ and solve the optimal solution
    $\{\mathbf{x}_i^*\}_{i \in \mathcal{V}_A}$ by (5.3) on $\mathcal{G}_A^{down}$;
4 Return $\mathbf{x}_A = \{\mathbf{x}_i^*\}_{i \in \mathcal{V}_A}$;
```

We introduce Algorithm 5.2 by a toy example in Fig. 5-5, where $\mathcal{V}_A = \{2, 4\}$. In this case, $\mathcal{V}_A^{down} = \{2, 3, 4, 5, 6\}$ and adversarial nodes only can access the routing information in the red dashed frame in Fig. 5-5(a). We then calculate $R[i], \forall i \in \mathcal{V}_A$ based on $\mathcal{G}_A^{down}$ by Algorithm 5.3, while here we explain the meaning of $R[i]$ using the upstream information to i as in Fig. 5-5(b) for simplicity. We observe that x_{12} of the traffic will arrive to node 2 without passing the other adversarial node 4, therefore $R[2] = x_{12}$. For node 4, the only traffic flow that does not pass node 2 and arrive to 4 take the path $1 \to 3 \to 4$, thus $R[4] = x_{13}x_{34}$. A special case is that $R[i] = 0$ if all the traffic arrived at adversarial node i needs to pass some other adversarial nodes. Finally, we add an arrival rate of $R[i]$ unit to adversarial node i in the downstream subgraph $\mathcal{G}_A^{down}$ as in Fig. 5-5(c), and we solve the optimal routing attack over $\mathcal{G}_A^{down}$ by Algorithm 5.1.

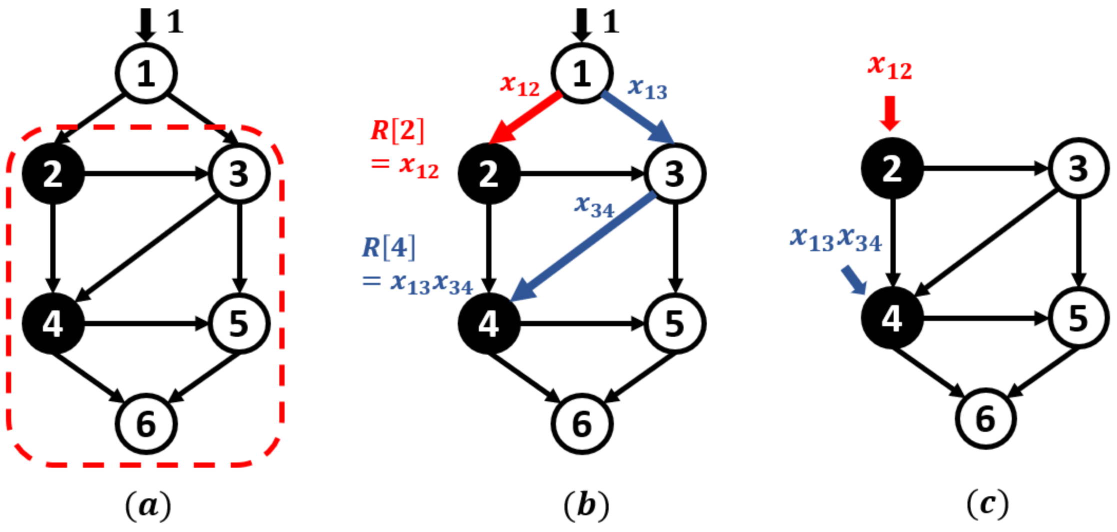


Figure 5-5: Example of running Algorithm 5.2 from (a) to (c)

Now we introduce Algorithm 5.3, which presents the mechanism to calculate $R[i]$ for $\forall i \in \mathcal{V}_A$ purely based on $\mathcal{G}_A^{down}$. The adversary first does not conduct routing attack on $\mathcal{V}_A = \{2, 4\}$, which means nodes in $\mathcal{V}_A$ take their default routing policies before being hijacked[4]. It chooses a timestamp where the network is not overloaded[5] to measure the traffic flow f_{ij} for $\forall (i, j) \in \mathcal{E}_A^{down}$ under the default routing policies of both $\mathcal{V}_N$ and $\mathcal{V}_A$, and the total throughput F_{total} at the destination node. Then

[4]We visualize $\mathcal{V}_A$ in Fig. 5-6 by shaded instead black nodes for this reason.

[5]We assume that the network before the routing attack does not contain saturated links, which is common in real networks with sufficient provisioned capacity [92].

we can calculate $R[i]$ by traversing the downstream nodes $\mathcal{V}_A^{down}$ in the order of topological sorting over $\mathcal{G}_A^{down}$: Iteratively determine $R[i]$ whenever traversing at $i \in \mathcal{V}_A$, and remove the adversarial node i and its adjacent downstream links with updated $\mathcal{V}_A^{down} = \mathcal{V}_A^{down} \backslash \{i\}$ and $\mathcal{E}_A^{down} = \mathcal{E}_A^{down} \backslash \{(i,j) \in \mathcal{E}, \forall j\}$, and then determine the next adversarial node. Fig. 5-6 gives an example of applying Algorithm 5.3 over the networks in Fig. 5-5, and it is easy to verify the correctness.

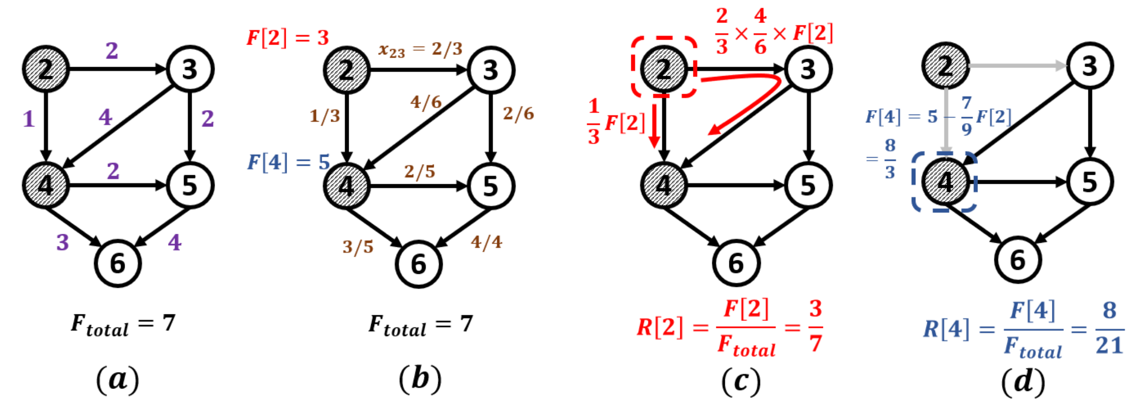
Algorithm 5.3: Calculate $R[i], i \in \mathcal{V}_A$ based on $\mathcal{G}_A^{down}$

```
1  **Input:** $\mathcal{V}_A$, $\mathcal{G}_A^{down}$, normal routing $\{\mathbf{x}_i\}_{i \in \mathcal{V}_N \cap \mathcal{V}_A^{down}}$;
2  Measure the traffic flow $f_{ij}, \forall (i,j) \in \mathcal{E}_A^{down}$ without routing attack on $\mathcal{V}_A$, and
     total flow $F_{total}$ at destination node;
3  Measure the total traffic arrival to each source node $i \in \mathcal{V}_A^{down}$ as $F[i]$;
4  Initialize the total flow to each node $j \in \mathcal{V}_A^{down}$ from within $\mathcal{G}_A^{down}$ as
     $F[j] = \sum_{i:(i,j) \in \mathcal{E}_A^{down}} f_{ij}$;
5  Topological sort $\mathcal{V}_A^{down}$;
6  **for** $i \in \mathcal{V}_A^{down}$ **do**
7      **If** $i \in \mathcal{V}_N$, **continue**;
8      $R[i] = F[i]/F_{total}$;
9      Remove $i$ and $\{(i,j) \mid (i,j) \in \mathcal{E}_A^{down}\}$ from $\mathcal{V}_A^{down}$ and $\mathcal{E}_A^{down}$ respectively;
10     Update $F[j]$ based on $(\mathcal{V}_A^{down}, \mathcal{E}_A^{down})$ by removing flow starting from $i$ to
         $j$, $\forall j \in \mathcal{V}_A^{down}$;
11 **Return** $\{R[i]\}_{i \in \mathcal{V}_A}$;
```

We prove Theorem 5.3 that λ_{ALG}^* under the routing attack $\mathbf{x}_A$ output by Algorithm 5.2 satisfies $\lambda_{ALG}^* \leq 2\lambda_{OPT}^*$.

Theorem 5.3. *Algorithm 5.2 is a 2-approximation algorithm.*

Proof. Recall the optimal traffic flow vector to (5.3) is denoted by $\mathbf{f}^*$. We denote $\mathbf{f}^*$ by $\mathbf{f}^{OPT}$ in the proof, and the traffic flow vector under the routing output by Algorithm 5.2 by $\mathbf{f}^{ALG}$. Therefore $(\lambda_{OPT}^*)^{-1} = \max_{(i,j)\in\mathcal{E}} f_{ij}^{OPT}/c_{ij}$ and $(\lambda_{ALG}^*)^{-1} = \max_{(i,j)\in\mathcal{E}} f_{ij}^{ALG}/c_{ij}$. Further notice that the flow f_{ij} is the sum of flow that are in the constructed $\mathcal{G}_A^{down}$ and those are not. Denote them as g_{ij} and h_{ij} respectively, and



2
3
4
5
6
$F_{total} = 7$
(a)
$F[2] = 3$
$x_{23} = 2/3$
1/3
4/6
2/6
$F[4] = 5$
2/5
3/5
4/4
$F_{total} = 7$
(b)
$\frac{2}{3} \times \frac{4}{6} \times F[2]$
$\frac{1}{3} F[2]$
$R[2] = \frac{F[2]}{F_{total}} = \frac{3}{7}$
(c)
$F[4] = 5 - \frac{7}{9} F[2] = \frac{8}{3}$
$R[4] = \frac{F[4]}{F_{total}} = \frac{8}{21}$
(d)

Figure 5-6: Example of Algorithm 5.3. The original traffic flows are marked by the numbers over each link in (a), and we can measure the total throughput $F_{total} = 7$ at the destination node 6. We initialize $F[2]$ and $F[4]$ to be the total flow to node 2 and node 4, which are 3 and 5 respectively shown in (b). Since node 2 is the source node in $\mathcal{G}_A^{down}$, $F[2] = 3$ does not need to be updated since all the flows from node 2 to 3 and 4 only traverse node 2. Then in (c) we calculate the flows at node 4 that have traversed node 2, which contains paths $2 \to 4$ and $2 \to 3 \to 4$ with a total of $7/9 \times F[2]$. Then in (d) we substract these flow from $F[4]$ and we get the final $F[4] = 8/21$.

thus $f_{ij} = g_{ij} + h_{ij}$. Therefore the approximation ratio

$$\begin{aligned}
\left(\frac{\lambda^*_{ALG}}{\lambda^*_{OPT}}\right)^{-1} &= \frac{\max_{(i,j)\in\mathcal{E}}(g_{ij}^{ALG} + h_{ij}^{ALG})/c_{ij}}{\max_{(i,j)\in\mathcal{E}}(g_{ij}^{OPT} + h_{ij}^{OPT})/c_{ij}} \\
&\overset{(a)}{=} \frac{\max_{(i,j)\in\mathcal{E}}(g_{ij}^{ALG} + h_{ij}^{ALG})/c_{ij}}{\max_{(i,j)\in\mathcal{E}}(g_{ij}^{OPT} + h_{ij}^{ALG})/c_{ij}} \\
&\overset{(b)}{\geq} \frac{\max\left\{\max_{(i,j)\in\mathcal{E}} g_{ij}^{ALG}/c_{ij}, \max_{(i,j)\in\mathcal{E}} h_{ij}^{ALG}/c_{ij}\right\}}{\max_{(i,j)\in\mathcal{E}}(g_{ij}^{OPT} + h_{ij}^{ALG})/c_{ij}} \\
&\overset{(c)}{\geq} \frac{\max\left\{\max_{(i,j)\in\mathcal{E}} g_{ij}^{OPT}/c_{ij}, \max_{(i,j)\in\mathcal{E}} h_{ij}^{ALG}/c_{ij}\right\}}{\max_{(i,j)\in\mathcal{E}} g_{ij}^{OPT}/c_{ij} + \max_{(i,j)\in\mathcal{E}} h_{ij}^{ALG}/c_{ij}} \\
&\overset{(d)}{\geq} 1/2.
\end{aligned}$$

In the derivation, (a) holds because h_{ij} represents the flow that does not pass any adversarial node any under default routing, thus not related to the difference between adversarial routing solutions and $h_{ij}^{ALG} = h_{ij}^{OPT}$; (b) holds based on $\max_i\{a_i + b_i\} \geq \max\{\max_i\{a_i\}, \max_i\{b_i\}\}$; (c) holds since Algorithm 5.2 outputs the solution that minimizes the no-loss throughput of the constructed $\mathcal{G}_A^{down}$, thus $\max_{(i,j)\in\mathcal{E}} g_{ij}^{ALG}/c_{ij} \geq$

$\max_{(i,j)\in\mathcal{E}} g_{ij}^{OPT}/c_{ij}$, and the denominator is due to $\max_i\{a_i + b_i\} \leq \max_i\{a_i\} + \max_i\{b_i\}$; (d) holds since $\max\{a,b\}/(a+b) \geq 1/2$. Therefore $\lambda^*_{ALG}/\lambda^*_{OPT} \leq 2$. □

We then can obtain a straightforward corollary, which states that as long as $\mathcal{V}_A$ contains a *node cut* of $\mathcal{G}$, the removal of which disconnects the source and destination, then Algorithm 5.2 returns the optimal routing attack that minimizes λ^* solely with downstream information.

Corollary 5.1. *With $\mathcal{G}_A^{down}$, Algorithm 5.2 outputs the optimal solution to* (5.3) *if $\mathcal{V}_A$ contains a node cut to $\mathcal{G}$.*

Proof. If $\mathcal{V}_A$ contains a node cut of the network, then no traffic flows will be sent to destination node without passing any adversarial node. Therefore $h_{ij}^{ALG} = h_{ij}^{OPT} = 0, \forall(i,j) \in \mathcal{E}_A^{down}$, and thus $\frac{\lambda^*_{ALG}}{\lambda^*_{OPT}} = 1$. □

5.2.4 Practical Extensions

Heuristic for Distributed Attack

Algorithm 5.1 and 5.2 require a centralized adversary who can manipulate the routing at different adversarial nodes simultaneously, while it does not apply to distributed routing attack where each adversarial node determines the routing separately. We introduce a heuristic in Algorithm 5.4 that works for distributed attack in directed acyclic networks with lower time complexity $O(d_{max}|\mathcal{V}_A|M)$. Each $i \in \mathcal{V}_A$ decides its routing that minimizes the no-loss throughput λ^* of the induced subgraph formed by $\{i\} \cup \mathcal{V}_i^{down}$ where node i itself serves as the source. We can traverse the nodes from the destination, and whenever we encounter an adversarial node, we decide its routing as above, fix it and continue the traversal to the source. The routing of the predecessor adversarial nodes depends on the successor ones.

We point out that Algorithm 5.4 can lead to an approximation ratio of $2^{|\mathcal{V}_A|}$ under the topology in Fig. 5-7. However, we show empirically in Section 5.5.1 that it is an effective heuristic that balances the attack performance and time efficiency.

Algorithm 5.4: Heuristics for Distributed Attack

```
1 Input: directed acyclic $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V}_A$, $\{\mathbf{x}_i\}_{i \in \mathcal{V}_N}$;
2 Topology sort $\mathcal{V}_A$ from sink to source;
3 for $i \in \mathcal{V}_A$ in topological sorted order do
4     Determine $\mathbf{x}_i^*$ by running Algorithm 5.1 on the induced graph of $\mathcal{V}_i^{down}$,
      and fix $\mathbf{x}_i^*$;
5 Return $\mathbf{x}_A^*$;
```

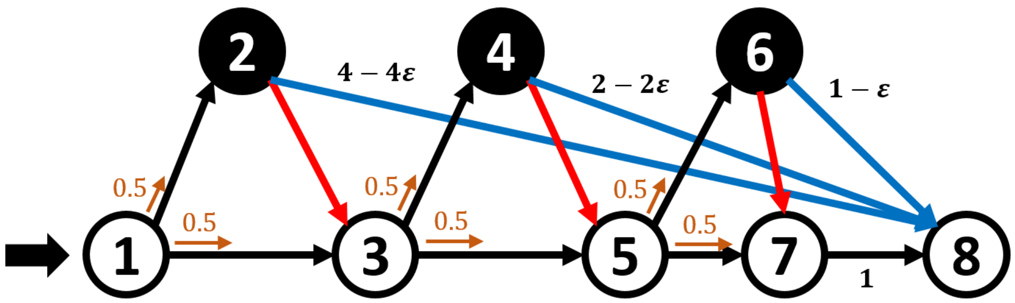


Figure 5-7: Example of $2^{|\mathcal{V}_A|} \times \lambda^*_{OPT}$ under Algorithm 5.4. Suppose $\mathcal{V}_A = \{2, 4, 6\}$; $c_{28} = 4 - 4\epsilon$, $c_{48} = 2 - 2\epsilon$, $c_{68} = 1 - \epsilon$ where $0 \leq \epsilon \leq 1$, $c_{78} = 1$, and other links have infinite capacity. Nodes 1, 3, and 5 route half of the traffic through each connected downstream link. Algorithm 5.1 routes all traffic through red links which leads to $\lambda^* = \lambda^*_{OPT} = 1$, while Algorithm 5.4 routes all traffic through blue links (topological order $6 \rightarrow 4 \rightarrow 2$) which leads to $\lambda^* = 8\lambda^*_{OPT} = 8$ when $\epsilon \rightarrow 0$.

Constraints on Routing Attacks

There may exist constraints over routing attacks in practice for the adversary. For example, the dispatch ratio may need to be within a normal range, i.e. $x_{ij} \in [x_{ij}^{\min}, x_{ij}^{\max}]$ for some $(i,j) \in \mathcal{E}$. Routing in an extreme manner outside such range, like routing all traffic to a single link based on Algorithm 5.1, may have a high risk of being detected as anomalous traffic and thus being exposed [115]. By incorporating such upper and lower bounds on dispatch ratio, the brute-force approach based on Theorem 6.1 is no longer guaranteed to be solved in polynomial time even under $|\mathcal{V}_A| = O(1)$ in that the polytope with constraints over each dimension may contain exponential number of vertices[6], while Algorithm 5.1 can still output the optimal solution in polynomial time since $x_{ij} = f_{ij}/\sum_{k:(k,i)\in\mathcal{E}} f_{ki} \in [x_{ij}^{\min}, x_{ij}^{\max}]$ is a linear constraint given $x_{ij}^{\min}$ and $x_{ij}^{\max}$.

Multi-Commodity Networks

The λ^*-minimization formulation and results can be extended to multi-commodity networks as follows. Assume L commodities, each commodity l has its source s_l, destination d_l, and default routing $x_{ij}^{(l)}$ for each link (i,j) that it passes through. Suppose the adversary knows the ratio of different commodities in the network, denoted by $\{\alpha_l\}_{l=1}^{L}$ where $\sum_{l=1}^{L} \alpha_l = 1$, and it can manipulate the routing of any commodity that pass the nodes in $\mathcal{V}_A$, then the following optimization identifies the optimal routing attack that minimizes no-loss *total* throughput λ^*, under which each commodity l has an arrival rate of $\lambda^*\alpha_l$, whose correctness can be erected similar as

[6]Suppose $\mathbf{A} \in \mathbb{R}^{m\times n}$ $(m < n)$ in the constraints $\mathbf{A}\mathbf{x} = \mathbf{b}$ of an LP in standard form, with linearly independent rows, and columns in any subset of m columns of $\mathbf{A}$, then there are at most $\binom{n}{m}$ basic feasible solutions (vertices), each of which corresponds to a basis of the LP. In our problem, $n = O(M)$. Without additional routing constraints, there are only $m = |\mathcal{V}_A| = O(1)$ flow conservation constraints, hence the brute-force algorithm has polynomial-time complexity. However, we may have $m = O(n)$ that leads to $\binom{n}{m} = O(2^M)$ vertices by adding routing constraints.

Algorithm 5.1.

$$\begin{aligned}
&\max_{\mathbf{f}=\{\mathbf{f}^{(l)}\}_{l=1}^{L}} \quad \max_{(i,j)\in\mathcal{E}} \quad \sum_{l=1}^{L} f_{ij}^{(l)}/c_{ij} \\
&\text{s.t.} \quad \mathbf{f}^{(l)} \in \Lambda^{(l)}, \quad f_{0s_l}^{(l)} = \alpha_l, \quad f_{ij}^{(l)} \geq 0, \ \forall (i,j) \in \mathcal{E}
\end{aligned} \tag{5.5}$$

In (5.5), the $\mathbf{f}^{(l)}$ and $\Lambda^{(l)}$ denote the flow variables and conservation laws as in (5.2) for commodity l. This problem can be solved in polynomial time in a way similar to (5.3), where for each link, we calculate the traffic arrival rate that saturates it assuming that the other links have infinite capacity, based on interior point methods to solve LP problems, and then take the routing attack which requires the minimum arrival rate to guarantee a saturated link. The 2-approximation ratio in Theorem 5.2 holds for (5.5) when the adversary only has access to the downstream routing information of each commodity, with similar proof. One future extension is to study the optimal attack when $\boldsymbol{\alpha}$ fluctuates within a certain range, which is practical in data center networks [92, 104].

5.3 Loss Maximization

In this section, we investigate the optimal routing attack $\mathbf{x}_A = \{\mathbf{x}_i\}_{i\in\mathcal{V}_A}$ to maximize the total traffic loss given the arrival rate λ. We establish that the problem is NP-complete. We propose two approximation algorithms with performance guarantee for single-hop networks, and discuss the extension to more general constraints and multi-hop networks.

5.3.1 Problem Formulation and NP-Completeness

We formulate the loss maximization problem in a general network as in (5.6).

$$
\begin{aligned}
\min_{\mathbf{f},\mathbf{x}_A} \quad & \sum_{i:(i,d)\in\mathcal{E}} f_{id} \\
\text{s.t.} \quad & f_{ij} = \min\left\{\left(\sum_{k:(k,i)\in\mathcal{E}} f_{ki}\right)x_{ij}, c_{ij}\right\}, \forall(i,j)\in\mathcal{E} \\
& \lambda = f_{01}, \qquad f_{ij} \geq 0,\ \forall(i,j)\in\mathcal{E}
\end{aligned}
\tag{5.6}
$$

The objective of (5.6) is to minimize the total traffic arrived at the destination node, which is equivalent to maximizing the loss $\lambda - \sum_{i:(i,d)\in\mathcal{E}} f_{id}$. The decision variables contain both the flow variables $\mathbf{f}$ and routing policies $\mathbf{x}_A$. The constraints require that the flow over a link (i,j) is equal to the minimum between the total flow injected from node i to node j and the link capacity c_{ij}. If the flow routed to (i,j) from i exceeds c_{ij}, then $f_{ij} = c_{ij}$ and the excess $f_{ij} - c_{ij}$ traffic will be dropped. We point out the loss maximization problem cannot be formulated with linear constraints as (5.1). A more compact but less intuitive equivalent way is to remove $\mathbf{x}_A$ from decision variables by expressing $\mathbf{x}_i$ via $\mathbf{f}$ as done in (5.1).

We prove Proposition 5.3 that the problem (5.6) also has the boundary optimality property as (5.3). This indicates a brute-force approach following the idea of Theorem 6.1 with the objective replaced by loss maximization can output the optimal solution to (5.6), with time complexity $O(N^{|\mathcal{V}_A|}M)$ which is polynomial under $|\mathcal{V}_A| = O(1)$.

Proposition 5.3 (Boundary Optimality). *There exists an optimal solution to* (5.6) *such that for $\forall i \in \mathcal{V}_A$, $\exists j$ that $(i,j)\in\mathcal{E}$ and $x^*_{ij} = 1$, while $x^*_{ik} = 0$ for $k \neq j$.*

Proof. Note that the constraint in (5.6) implies that the flow f_{ij} can be expressed by a k-layer nested min function by the flows over links k hops predecessor to link (i,j). Consider the example in Fig. 5-2 w.l.o.g and of higher clarity, the objective function can be written as

$$
\begin{aligned}
f_{46} + f_{56} &= \min\{f_{24} + f_{34}, c_{46}\} + \min\{f_{35}, c_{56}\} \\
&= \min\{\min\{f_{12}, c_{24}\} + \min\{f_{13}x_{34}, c_{34}\}\} \\
&\quad + \min\{\min\{f_{13}x_{35}, c_{35}\}, c_{56}\} = ...
\end{aligned}
$$

We first consider the case with a single adversarial node $\mathcal{V}_A = \{i\}$. Then the objective in (5.6) is a nested combination of minimum and sum of affine functions of $\mathbf{x}_i$, which is concave since the min function is concave. Furthermore, by expressing (5.6) with decision variables of $\mathbf{x}_i$, the constraints are only $\sum_{j:(i,j)\in\mathcal{E}} x_{ij} = 1, x_{ij} \geq 0$, which are linear. Therefore (5.6) is to minimize a concave function in an affine feasible region, indicating that one of the vertices must be optimal, i.e. $\exists j$ that $(i,j) \in \mathcal{E}$ and $x^*_{ij} = 1$, while $x^*_{ik} = 0$ for $k \neq j$.

Now we extend to general $\mathcal{V}_A$. Note that for $\forall i \in \mathcal{V}_A$, when the routing policies of nodes in $\mathcal{V}_A \backslash \{i\}$ are fixed, the objective is a concave function of the routing policies $\mathbf{x}_i$. Given any fixed routing of nodes in $\mathcal{V}_A \backslash \{i\}$, one of the optimal routing polices at node i must be $x_{ij} = 1$ over some link (i,j). This means that given an adversarial routing solution $\{\mathbf{x}_i\}_{i\in\mathcal{V}_A}$, if there exists any node i_0 that satisfies $x_{i_0 j} < 1, \forall (i,j) \in \mathcal{E}$, there must exist j_0 where letting $x_{i_0 j_0} = 1$ will not cause less loss. Therefore via proof by contradiction, there must exist a vertex that maximizes the loss. $\square$

However, we show that unlike no-loss throughput minimization, loss maximization (5.6) is an NP-complete problem under general $\mathcal{V}_A$.

Theorem 5.4. *Problem* (5.6) *is NP-complete under* $|\mathcal{V}_A| = O(N)$.

Proof. We reduce Set Cover to Problem 5.6. Given an instance of Set Cover: m elements $\{e_i\}_{i=1}^m$, n sets $\{S_j\}_{j=1}^n$ where each set covers several elements, what is the minimum number of sets that can fully covers all the elements? We construct the graph in Fig. 5-8, where element e_i as a node s_i, and set S_j as a node d_j, and there is a directed link from (s_i, d_j) if and only if the set S_j contains e_i, with capacity $c_{s_i,d_j} = \infty$. We further introduce a meta source node s_0, and a link from s_0 to each of the nodes in $\{s_i\}_{i=1}^m$ with link capacity $c_{s_0,s_i} = \infty$. The routing policy at s_0 is $x_{s_0,s_i} = 1/m$, $\forall i = 1, \cdots, m$, and an external arrival rate of m. We then introduce a meta destination node d_0, and introduce a directed link (d_j, d_0) with capacity $c_{d_j,d_0} = 1$. The adversarial node set $\mathcal{V}_A = \{s_i\}_{i=1}^m$, and they can dispatch traffic to any of its connected downstream nodes. Clearly, the optimal routing attack of $\mathcal{V}_A$ that maximizes the loss is to route traffic to minimum number of nodes in

$\{d_j\}_{j=1}^n$, where the maximum loss is $m - p$ suppose that the minimum number is p. Therefore if there exists a polynomial algorithm for Problem (5.6), then it outputs the minimum number of nodes in $\{d_j\}_{j=1}^n$ that receives traffic under the optimal attack, which corresponds to the minimum set cover over elements $\{e_i\}_{i=1}^m$, thus contradicting to the NP-hardness of Set Cover. □

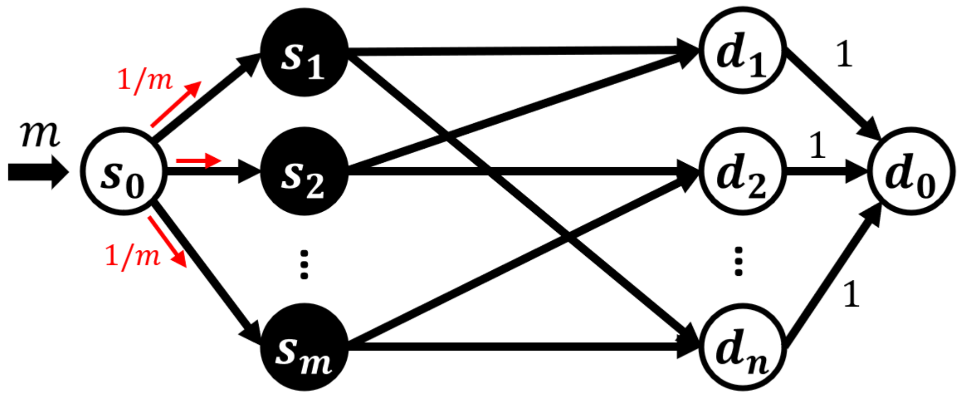


Figure 5-8: Graph Construction from Set Cover

5.3.2 Approximation Algorithms

We propose two approximation algorithms with multiplicative and additive performance guarantees respectively in single-hop networks, where routing attacks are conducted on source nodes $\mathcal{V}_S$. Under static routing, we can assume w.l.o.g. $\mathcal{V}_S = \mathcal{V}_A$, i.e., all source nodes are hijacked. This is because if a source node is normal, then based on its default routing policy we can calculate the traffic flow sent from this node to each of its connected downstream destination nodes. We can then correspondingly reduce the service rate μ_j at each destination node d_j by the total amount of flow arriving from all normal source nodes to d_j until the remaining service rate at d_j reaches zero. Therefore, the single-hop network with normal source nodes can be equivalently transformed into the one with all sources being adversarial, and updated remaining service rates at $\mathcal{V}_D$. We consider $\mathcal{V}_S = \mathcal{V}_A$ below.

Algorithm with Multiplicative Guarantee

We propose a greedy-based algorithm that has a worst-case approximation ratio of $1/\sqrt{|\mathcal{V}_A|}$. We summarize the details in Algorithm 5.5, which executes two greedy approaches and outputs the solution that leads to higher loss. Both approaches share the idea of iteratively routing traffic to the destination node that leads to maximum overload. Approach 1 maximizes the overload at a destination node *without* normalization. In each iteration, the adversary takes aim at the destination node $d_j^* = \arg\max_{d_j \in \mathcal{V}_D} \left(\sum_{s_i \in UD:(s_i,d_j)\in\mathcal{E}} \lambda_i\right) - \mu_j$, where UD denotes the set of adversarial nodes whose routing is undecided by the current iteration, and then sets the routing policy at all source nodes in $UD \cap \{s_i \in \mathcal{V}_S \mid (s_i, d_j^*) \in \mathcal{E}\}$ to be dispatching all their traffic to d_j^*. Approach 2 maximizes overload at a destination node *with* normalization, where at each $d_j \in \mathcal{V}_D$, the adversary finds the routing that maximizes the per-source overload at d_j, defined as

$$PSO[d_j] := \max_{\mathcal{S}_{d_j} \subseteq UD} \frac{1}{|\mathcal{S}_{d_j}|} \left(\sum_{s_i \in \mathcal{S}_{d_j}:(s_i,d_j)\in\mathcal{E}} \lambda_i - \mu_j \right) \tag{5.7}$$

which means to find the subset of source nodes, denoted by $\mathcal{S}_{d_j}$, that can send traffic to and maximizes the overload at d_j normalized by the size of $\mathcal{S}_{d_j}$. Denote the optimal $\mathcal{S}_{d_j}$ with respect to (5.7) for node d_j by $\mathcal{S}^*_{d_j}$. The adversary routes all the traffic at source nodes in $\mathcal{S}^*_{d_j^*}$ to $d_j^* = \arg\max_{d_j \in \mathcal{V}_D} PSO[d_j]$. We iterate the above process over the source nodes whose routing policies are yet to be determined. Note that the determination of $\mathcal{S}^*_{d_j}$ for each $d_j \in \mathcal{V}_D$ can be done in polynomial time: Initialize $\mathcal{S}^*_{d_j} = \emptyset$, sort all source nodes in $UD \cap \{s_i \in \mathcal{V}_S \mid (s_i, d_j^*) \in \mathcal{E}\}$ non-increasingly with respect to the traffic arrival rates to them, and add these source nodes to $\mathcal{S}^*_{d_j}$ in sequel, until the $PSO[d_j]$ reaches the peak value and starts decreasing. This guarantees that Approach 2 is a polynomial-time algorithm.

We demonstrate in Theorem 5.5 that taking the better routing attack solution between Approach 1 and 2 leads to $1/\sqrt{|\mathcal{V}_A|}$ worst-case approximation ratio. A single application of either of the two approaches, however, has a worst-case approximation

Algorithm 5.5: Approx. with Multiplicative Guarantee

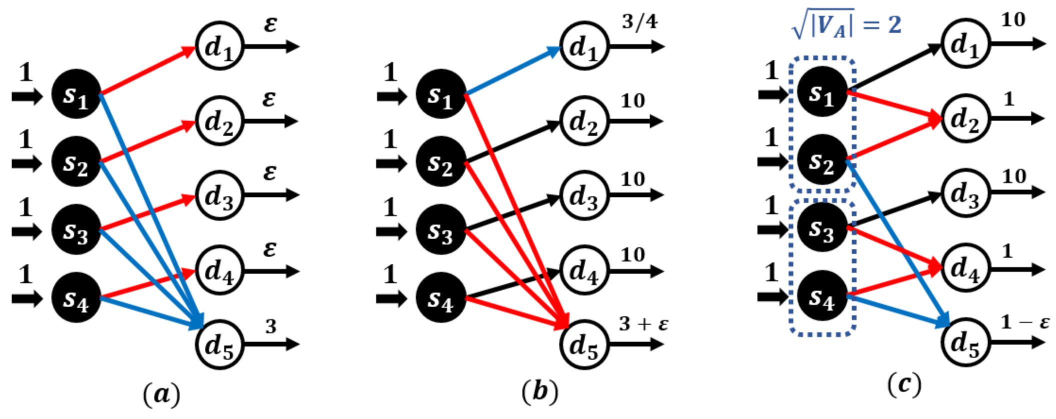
```
1  Input: Single-hop network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V}_A$, $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$;
2  Function GreedyAlg($\mathcal{G}$, $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, GreedyType):
3      Initialize $\mathbf{x}_i = \mathbf{0}$, $\forall s_i \in \mathcal{V}_A$; $UD = \mathcal{V}_A$;
4      while $\exists s_i \in UD$ do
5          if GreedyType is Approach1 then
6              $j^* \leftarrow \arg\max_j \left( \sum_{s_i \in UD, (s_i, d_j) \in \mathcal{E}} \lambda_i \right) - \mu_j$;
7          if GreedyType is Approach2 then
8              $j^* \leftarrow \arg\max_j PSO[d_j]$ as (5.7);
9          Let $\mathbf{x}_{ij^*} = 1$, $UD = UD \backslash s_i$, $\forall s_i, (s_i, d_j^*) \in \mathcal{E}$;
10     Calculate loss based on $sol \leftarrow \{\mathbf{x}_i\}_{i \in \mathcal{V}_A}$;
11     return loss, sol
12 Update remaining service rates at $\mathcal{V}_D$;
13 Remove $\mathcal{V}_N$ from $\mathcal{V}_S$, and associated links from $\mathcal{E}$;
14 $loss_1$, $sol_1 \leftarrow$ GreedyAlg ($\mathcal{G}$, $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, Approach1);
15 $loss_2$, $sol_2 \leftarrow$ GreedyAlg ($\mathcal{G}$, $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, Approach2);
16 Return $sol_1$ if $loss_1 > loss_2$ else $sol_2$;
```

ratio of $O(|\mathcal{V}_A|^{-1})$. We give toy examples in Fig. 5-9 to validate the tightness of these performance guarantees. The red routing is the optimal routing attack, while the blue one is the routing attack output by Approach 1 in Fig. 5-9(a), Approach 2 in Fig. 5-9(b), and Algorithm 5.5 in Fig. 5-9(c), respectively. In the table in Fig. 5-9, Δ_{OPT}, Δ_{A1}, Δ_{A2} and Δ_{MUL} denote respectively the loss under optimal routing attack, applying Approach 1, applying Approach 2, and applying Algorithm 5.5 which has multiplicative guarantee.

Theorem 5.5. *Algorithm 5.5 outputs a solution with loss Δ_{MUL} that satisfies $\frac{\Delta_{MUL}}{\Delta_{OPT}} \geq \frac{1}{\sqrt{|\mathcal{V}_A|}}$.*

Algorithm with Additive Guarantee

We further propose Algorithm 5.6 with additive performance guarantee. The main idea is to iteratively solve the λ^*-minimization problem (5.3) until either all the adversarial nodes have determined their routing policies or the λ^* exceeds the given arrival rate λ. The intuition behind Algorithm 5.6 is to greedily find the routing attack



Network	Δ_{OPT}	Δ_{A1}	Δ_{A2}	Δ_{MUL}
(a)	$\mathbf{4(1-\varepsilon)}$	$\mathbf{1}$	$4(1-\varepsilon)$	$4(1-\varepsilon)$
(b)	$\mathbf{1-\varepsilon}$	$1-\varepsilon$	$\mathbf{1/4}$	$1-\varepsilon$
(c)	$\mathbf{2}$	$1+\varepsilon$	$1+\varepsilon$	$\mathbf{1+\varepsilon}$

Figure 5-9: Tightness validation of Theorem 5.5 under $|\mathcal{V}_A| = 4$. Consider $\epsilon \to 0$, Approach 1 on network (a) and Approach 2 on network (b) leads to $1/|\mathcal{V}_A| = 1/4$ approximation ratio; Algorithm 5.5 gives worst-case approximation ratio $1/\sqrt{|\mathcal{V}_A|} = 1/2$ on network (c).

to cause the next possible saturated link with minimum arrival rate increment, which can be done by iteratively fixing the routing decided in previous iterations (line 7), and updating the capacity of previous saturated links to be infinite (line 8).

Algorithm 5.6: Approx. with Additive Guarantee

```
1  Input: Single-hop network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V}_A$, $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$;
2  Introduce meta-source normal node $s_0$ to $\mathcal{G}$ with dispatch ratio
     $x_{s_0,s_i} = \lambda_i / \sum_{k=1}^{|\mathcal{V}_S|} \lambda_k, \forall s_i$;
3  Introduce meta-destination $d_0$ with $c_{d_j,d_0} = \mu_j$, $\forall d_j$;
4  Initialize $\mathbf{x}_i = \mathbf{0}, \forall s_i \in \mathcal{V}_A$; $UD = \mathcal{V}_A$;
5  while $\exists s_i \in UD$ do
6      Run Algorithm 5.1 on the updated $\mathcal{G}, \mathcal{V}_A, \mathbf{x}_N$ and get the first saturated
         link $(d_{j^*}, d_0)$;
7      Let $x_{ij^*} = 1$, $UD = UD \backslash \{s_i\}$, and fix the routing at $s_i$, for
         $\forall s_i \in UD, (s_i, d_j^*) \in \mathcal{E}$;
8      Let $c_{d_{j^*},d_0} = \infty$;
9  Calculate $loss$ based on $sol \leftarrow \{\mathbf{x}_i\}_{s_i \in \mathcal{V}_A}$;
10 Return $sol$, $loss$;
```

We demonstrate in Theorem 5.6 that the gap between the output of Algorithm 5.6 and the maximum loss given by (5.6) is bounded above by $\lambda/4$ in the worst case for any single-hop network. This means, for example, if the optimal attack causes traffic loss $50\% \times \lambda$, then Algorithm 5.6 at least causes $25\% \times \lambda$.

Theorem 5.6. *Algorithm 5.6 outputs the solution with loss Δ_{ADD} that satisfies $\frac{\Delta_{OPT}-\Delta_{ADD}}{\lambda} \leq \frac{1}{4}$.*

We point out that the 1/4 bound is tight, given by the example shown in Fig. 5-10. The optimal routing at s_2 is to route traffic to d_1, causing loss of 2, while the solution by Algorithm 5.6 will route traffic to d_2, causing loss of $1+\epsilon$ where $\epsilon > 0$. Thus the gap $(2-(1+\epsilon))/\lambda \to 1/\lambda = 1/4$ when $\epsilon \to 0$. However, the condition to achieve the worst-case gap 1/4 is very strong, which requires the arrival rates to all sources and the service rates of all destinations to meet some ratio equality, where in Fig. 5-10 they are $x_{s_0,s_2} = \mu_2/\mu_1$ and $\mu_2/\mu_1 \to 1/2$. We show in Section 6.4 that empirically Algorithm 5.6 leads to the gap less than $5\% \times \lambda$ in more than 95% tested cases.

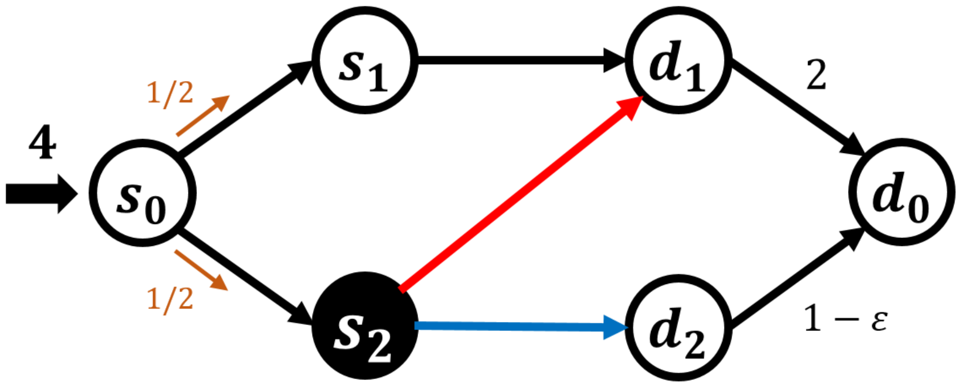


Figure 5-10: Tightness validation of Theorem 5.6 ($\lambda = 4$, $\mathcal{V}_A = \{s_2\}$): red route is optimal, blue route is the output of Algorithm 5.6.

5.3.3 Practical Extensions

Constraints on Routing Attack

We follow the constraint form, $x_{ij} \in [x_{ij}^{\min}, x_{ij}^{\max}]$ for some $(s_i, d_j) \in \mathcal{E}$, in Section 5.2.4. We show that both Algorithm 5.5 and 5.6 can incorporate this constraint in single-hop networks. For the lower bound, we can require that each adversarial source s_i initially sends $\lambda_i \times x_{ij}^{\min}$ traffic through each link (s_i, d_j). For the upper bound, Algorithm 5.6, which is based on no-loss throughput minimization, can keep unchanged as discussed in Section 5.2.4. In Algorithm 5.5, an adversarial node s_i will still be considered in future iterations if s_i has not determined the destination nodes for all its traffic, unlike the unconstrained case where the routing of an adversarial node will be determined in one of the iterations. For example, suppose $x_{ij}^{\min} = 0$ and consider that s_i routes $x_{s_i d_j}^{\max}$ of all the traffic to d_j, then there remains $\lambda_i(1 - x_{s_i d_j}^{\max})$ units of traffic rates at s_i whose routing is undecided, and we will consider s_i in future iterations to route the remaining traffic to destinations other than d_j until there no longer exists traffic with routing undecided at s_i. With such adjustment, the factor $1/|\mathcal{S}_{d_j}|$ in (5.7) should be the reciprocal of the sum of the remaining proportions of traffic with undecided routing over $\mathcal{S}_{d_j}$.

Multi-hop Networks

We briefly discuss the extension of the algorithms to multi-hop networks. In Algorithm 5.5, unlike in single-hop networks we maximize the inflow to each node in a greedy manner, in multi-hop networks we greedily find the node i that achieves maximum possible loss over its connected downstream links, under the adversarial routing at $\mathcal{V}_A \cap \mathcal{V}_i^{up}$ that maximizes the total inflow to node i. This is similar to lines 2-5 in Algorithm 5.1 with the objective changed to traffic loss. For Algorithm 5.6, we can naturally extend it to multi-hop networks polynomially based on Algorithm 5.1: iteratively saturate the next link by minimizing the no-loss throughput λ^*, until $\lambda^* \geq \lambda$ or all nodes in $\mathcal{V}_A$ have determined their routing policies.

5.4 Optimal Node Selection

In this section, we investigate the situation where the adversary needs to select a limited number of nodes to hijack before conducting routing attack. The discussion serves as an extension to the capability of routing attack in practice, and gives a benchmark to evaluate the most critical nodes that should be protected from routing attack.

5.4.1 Problem Formulation and NP-Completeness

We consider a set of candidate nodes that can be hijacked by the adversary, $\mathcal{V}_{cand}$. The adversary selects at most $K < |\mathcal{V}_{cand}|$ nodes to hijack, and the objective is to conduct routing attack on them to minimize no-loss throughput or maximize loss, i.e., solving (5.3) or (5.6). This setting maps to the practice: $\mathcal{V}_{cand}$ is a subset of nodes with insecure firewalls so that the adversary can hijack into, and the adversary can attack a limited number of nodes due to the attack cost, and the risk to be detected. The problem can extended naturally to a weighted version with a bounded total cost, which shares the idea below but is deferred to future work.

We prove that for both no-loss throughput minimization and loss maximization, the problem is NP-complete for general $K = O(N)$, while polynomially solvable for $K = O(1)$.

Theorem 5.7. *Finding the optimal selection of K nodes from $\mathcal{V}_{cand}$ to conduct routing attack for no-loss throughput minimization and loss maximization is NP-complete under $K = O(N)$ while P under $K = O(1)$.*

Although polynomial under $K = O(1)$, the brute-force solution is impractical when K is a large constant[7]. Below we propose efficient heuristic algorithms which can minimize the no-loss throughput exactly, and achieve close-to-optimal traffic loss verified empirically in Section 5.5.3, when $\mathcal{V}_{cand}$ are in *parallel* structure. Parallel $\mathcal{V}_{cand}$ means any node $i_1 \in \mathcal{V}_{cand}$ is not upstream or downstream to another node $i_2 \in \mathcal{V}_{cand}$. The single-hop network is a special case where $\mathcal{V}_{cand} \subseteq \mathcal{V}_S$ are parallel to each other.

[7]Time complexity is at least $O(N^5)$ if $|\mathcal{V}_{cand}| = O(N)$ and $K = 5$, which is high in practice.

5.4.2 Algorithms Under Parallel $\mathcal{V}_{cand}$

No-Loss Throughput Minimization

We show that the optimal node selection problem is polynomially solvable with parallel $\mathcal{V}_{cand}$ for no-loss throughput minimization by Algorithm 5.7. Denote the candidate nodes upstream to node i as $\mathcal{V}_{cand}^{up,i}$ (assume $i \in \mathcal{V}_{cand}^{up,i}$ if $i \in \mathcal{V}_{cand}$). The main idea is to traverse each link (i,j) in the downstream graph to $\mathcal{V}_{cand}$. For each candidate node $v \in \mathcal{V}_{cand}^{up,i}$, the adversary can evaluate the maximum reduction of the required traffic arrival rate at the source node to saturate (i,j) by choosing to attack v. The adversary then chooses $\min\{K, |\mathcal{V}_{cand}^{up,i}|\}$ nodes from $\mathcal{V}_{cand}^{up,i}$ non-decreasingly in terms of the maximum reduction. Denote the choice for $(i,j) \in \mathcal{E}$ as $\mathcal{V}_A^{(i,j)}$. We can traverse all links (i,j) downstream to $\mathcal{V}_{cand}$ to build $\{\mathcal{V}_A^{(i,j)}\}_{(i,j)\in\mathcal{E}}$, and pick the $\mathcal{V}_A^{(i^*,j^*)}$ under which the no-loss throughput is minimized by solving the routing attack for (5.3). We show that it gives the exact optimal node selection in Theorem 5.8.

Algorithm 5.7: Optimal node selection for no-loss throughput minimization under parallel $\mathcal{V}_{cand}$

```
1 Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, parallel $\mathcal{V}_{cand}$, number of nodes to attack $K$, default
    routing $\mathbf{X}$;
2 Build $\mathcal{V}_{cand}^{up,i}$ for each node $i \in \mathcal{V}_{cand}$ and downstream to $\mathcal{V}_{cand}$;
3 for $(i,j) \in \mathcal{E}$ downstream to $\mathcal{V}_{cand}$ do
4     Calculate $\Delta_v^{(i,j)}$, $\forall v \in \mathcal{V}_{cand}^{up,i}$, the difference of minimum arrival rate that
        saturates a $(i,j)$ with and without attacking $v$, not attacking other
        candidate nodes;
5     Sort non-increasingly $\{\Delta_v^{(i,j)}\}_{v\in\mathcal{V}_{cand}^{up,i}}$;
6     Determine $\mathcal{V}_A^{(i,j)}$, the optimal node choice to saturate link $(i,j)$, by finding
        the top-$\min\{K, |\mathcal{V}_{cand}^{up,i}|\}$ candidate nodes in $\mathcal{V}_{cand}^{up,i}$ with maximum $\Delta_v^{(i,j)}$;
7 Calculate the minimum no-loss throughput under routing attack over $\mathcal{V}_A^{(i,j)}$
    for all $(i,j) \in \mathcal{E}$ downstream to $\mathcal{V}_{cand}$, and find the minimum one
    corresponding to link $(i^*,j^*)$;
8 Return $\mathcal{V}_A^{(i^*,j^*)}$ as the selected nodes to attack;
```

Theorem 5.8. *Algorithm 5.7 outputs the optimal node selection to conduct routing attack that leads to minimum no-loss throughput given parallel $\mathcal{V}_{cand}$.*

Loss Maximization

Unlike no-loss throughput, it is challenging to derive heuristics with performance guarantee even under parallel $\mathcal{V}_{cand}$ for loss maximization. The main bottleneck lies in the performance verification of a specific node selection, i.e., finding the optimal routing attack to achieve maximum loss among a given set of the selected nodes, is NP-complete. We leave the theoretical challenge in future work, while we propose a heuristic in Algorithm 5.8 over single-hop networks with $\mathcal{V}_{cand} \subseteq \mathcal{V}_S$ to explain the basic idea by a greedy approach over each destination node, where the extension to parallel $\mathcal{V}_{cand}$ is natural by a greedy approach over each link. Algorithm 5.8 iteratively picks nodes to attack until K nodes have been chosen. In each iteration, for each destination node d_j, the algorithm evaluates the *value* for each of its connected source node $s_i \in UD$, i.e. whose routing has not been decided, to route all its traffic to d_j. The value is defined as $v_{ij} = \lambda_i(1 - x_{ij})$, the increment of inflow to d_j after routing attack where x_{ij} is the default dispatch ratio from s_i to d_j. Then for each destination node d_j, we have a corresponding choice of the connected source nodes, which have not been selected in previous iterations, according to the non-increasing order of $\{v_{ij}\}_{s_i:(s_i,d_j)\in\mathcal{E}}$ until either the per-source contribution to the loss at d_j, defined as $PSC[d_j]$ in Algorithm 5.8, reaches the peak value denoted by $PSC^*[d_j]$[8], or the algorithm has chosen K nodes in total till this iteration. With all $PSC^*[d_j]$ calculated and the corresponding selected source nodes to attack, denoted by $\mathcal{S}_j$, we then determine the new nodes to attack in this iteration to be $\mathcal{S}_{j^*}$, with $d_{j^*} = \arg\max_{d_j\in\mathcal{V}_D} PSC^*[d_j]$, and determine their routing to be dispatching all the traffic to d_{j^*}. We then fix the routing of the selected sources and update the remaining service rate of the d_{j^*}, and do iteratively.

5.5 Performance Evaluation

In this section, we evaluate the proposed routing attack and optimal node selection algorithms to showcase their near-optimal performance over no-loss throughput

[8]The process is similar to the (5.7) in Algorithm 5.5

Algorithm 5.8: Heuristic node selection for loss maximization in single-hop networks

```
1  Input: Single-hop network $\mathcal{G}$, $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, $\mathcal{V}_{cand}$, number of nodes to attack $K$,
      default routing $\mathbf{X}$;
2  while $|Selected| < K$ do
3      for $d_j \in \mathcal{V}_D$ do
4          Evaluate value for each $s_i \in \mathcal{V}_S \backslash Selected$ as $v_{ij} = \lambda_i(1 - x_{ij})$;
5          Sort nodes in $\mathcal{V}_S \backslash Selected$ non-increasingly with respect to value;
6          Initialize $S_{tmp} = \emptyset$;
7          Add each source node to $d_j$ in the above sorted order to $S_{tmp}$ and
             update $PSC[d_j] := \frac{1}{|S_{tmp}|}\left(\sum_{s_i \in S_{tmp}} v_{ij} - \mu_j\right)$ before this value starts
             to decrease;
8          Find the peak value $PSC^*[d_j]$ in line 7, and the selected source nodes
             as $\mathcal{S}_j$;
9      Determine $j^* = \arg\max_{d_j \in \mathcal{V}_D} PSC^*[d_j]$, and add $\mathcal{S}_{j^*}$ to Selected.
         Updated the routing of nodes in $\mathcal{S}_{j^*}$ in $\mathbf{X}$, and the remaining service rate
         $\mu_{j^*}$;
10 Return Selected as the selected nodes, $\mathbf{X}$ as the updated routing of
      Selected;
```

minimization and loss maximization over a wide range of network settings, including different network densities, sizes of adversarial node set, and default routing policies.

		Uniform			Proportional			ECMP			MaxFlow		
Den	$\|\mathcal{V}_A\|$	Mean	90%	Max	Mean	90%	Max	Mean	90%	Max	Mean	90%	Max
0.4	10	1.00	1.00	1.09	1.01	1.00	1.18	1.00	1.00	1.15	1.01	1.00	2.00
0.8	10	1.01	1.07	1.24	1.02	1.10	1.27	1.01	1.08	1.15	1.04	1.17	1.50
0.4	20	1.00	1.00	1.03	1.00	1.00	1.04	1.00	1.00	1.01	1.00	1.00	1.25
0.8	20	1.00	1.00	1.03	1.00	1.00	1.04	1.00	1.00	1.02	1.01	1.07	1.33

Table 5.1: Approximation ratio statistics for Algorithm 5.2 (90% means 90-percentile)

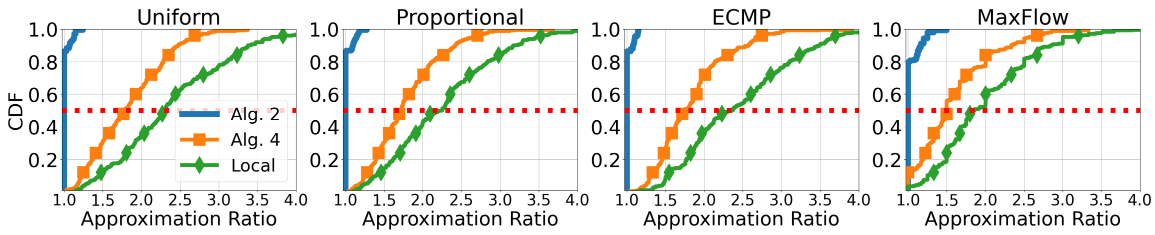


Figure 5-11: CDFs of approximation ratio under density 0.8 and $|\mathcal{V}_A| = 20$

5.5.1 No-Loss Throughput λ^* Minimization

For no-loss throughput minimization, we have shown that Algorithm 5.1 yields the exact optimal solution in polynomial time. This exact algorithm serves as our baseline, and we test the performance of 3 algorithms by measuring their gaps to the baseline: (i) the 2-approximation Algorithm 5.2, in order to validate the correctness of Theorem 5.3 and show its closeness to the baseline; (ii) the distributed attack heuristic Algorithm 5.4, in order to evaluate the performance gap between distributed and centralized routing attack; (iii) a *local heuristic* where each adversarial node routes all the traffic through the link with minimum capacity among all its connected downstream links, in order to demonstrate the performance enhancement that adversaries can achieve via Algorithm 5.1 compared with a naive attack.

We simulate multi-hop networks with size $|\mathcal{V}| = 50$. Our evaluation has the following dimensions. (I) Network Density: We generate random network topologies[9] given different link existence probabilities p, and present representative results under $p = 0.4$ and $p = 0.8$; (II) Number of Adversarial Nodes $|\mathcal{V}_A|$: We showcase results for $\mathcal{V}_A = 10$ and 20, with other values exhibiting similar outcomes; (III) Default Routing Policy: We consider 4 routing policies at normal nodes, which have been applied in different scenarios: (i) Uniform routing: Each normal node i dispatches an equal portion of traffic to each of its downstream connected nodes, oblivious to link capacity, i.e., $x_{ij} = 1/\text{outdegree}[i], (i, j) \in \mathcal{E}$. (ii) Capacity Proportional routing: The dispatch ratio of traffic to a downstream connected node is proportional to the capacity of this link, i.e., $x_{ij} = c_{ij}/\sum_{k:(i,k)\in\mathcal{E}} c_{ik}$. (iii) Equal-Cost Multi-Path (ECMP) routing [116]: Traffic at the source is dispatched through the paths with minimum cost. If there are $L > 1$ min-cost paths, then $1/L$ of the traffic takes each of the paths. We only present the results where each link has the same cost, as the other costs we have tested share similar results. ECMP is widely used in load balancing and commonly applied in industrial data center and cloud networks [1]. (iv) MaxFlow routing [117]: This default routing achieves maximum throughput under no routing attacks, which is widely studied in theory and serves as a paradigm for network design

[9]We ensure that there exists at least a path from source to destination.

[22]. We simulate 10000 network instances for each combination of the above network settings: We construct 20 different topologies, and under each topology, we consider 20 different adversarial node sets $\mathcal{V}_A$, and further under each $\mathcal{V}_A$, we set 25 different link capacity allocations.

We evaluate the cumulative distribution function (CDF) of the approximation ratio $\lambda^*_{ALG}/\lambda^*_{OPT}$, where λ^*_{ALG} is the no-loss throughput output by the tested algorithms and λ^*_{OPT} is the optimal routing attack by Algorithm 5.1. We present the approximation ratio statistics of the proposed Algorithm 5.2 in Table 5.1, including the mean, 90-percentile, and maximum approximation ratio under various network settings. For instance, the value 1.09 in the first row signifies the maximum approximation ratio of Algorithm 5.2 under density 0.4, $|\mathcal{V}_A| = 10$, and uniform default routing, over all 10000 tested instances under this setting. We have the following observations: (i) In any tested setting, Algorithm 5.2 exhibits near-optimal performance in most instances. The mean approximation ratio is less than 1.05, and over 90% of instances have an approximation ratio of less than 1.20 under all network settings. This result demonstrates that the routing attack by Algorithm 5.2 achieves near-optimal overload increase in most cases, utilizing downstream information only. (ii) Algorithm 5.2 performs worst on MaxFlow routing, with mean, p90, and maximum approximation ratios larger than those of the other three policies. Notably, under density 0.4 and $|\mathcal{V}_A| = 10$, an instance exists where the approximation ratio reaches 2, the bound established in Theorem 5.3. This outcome can be attributed to MaxFlow routing distributing traffic to optimally exploit network capacity, resulting in a considerable portion of traffic being dispersed through paths without adversarial nodes. (iii) The routing attack performs best on density 0.4 and $|\mathcal{V}_A| = 20$ and worst on density 0.8 and $|\mathcal{V}_A| = 10$ in general. This observation aligns with the intuition that higher density combined with fewer adversarial nodes allows network traffic to access more available paths without adversarial nodes, thus reducing the possibility of forming a node cut so that Algorithm 5.2 is able to output the optimal attack as Algorithm 5.1, which echoes Corollary 5.1.

We proceed to compare the performance gaps to the baseline over the 3 algorithms:

Algorithm 5.2, Algorithm 5.4, and the local heuristic. We illustrate the CDFs of the approximation ratios of the three algorithms under four default routing policies with density 0.8 and $|\mathcal{V}_A| = 20$ in Fig. 5-11. Despite Algorithm 5.2 exhibits the worst approximation performance under this setting compared to the other settings in Table 5.1, it still significantly outperforms Algorithm 5.4 and the local heuristic under all the given default routing policies. The CDF curves reveal that Algorithm 5.2 identifies the optimal solution in over 80% of the instances. Moreover, Algorithm 5.4 strikes a balance between attack performance and time efficiency, compared with the optimal attack performance of Algorithm 5.1 with higher time cost in solving LPs, and the local heuristics with lowest time complexity but without any performance guarantee. We can observe that Algorithm 5.4 leads to approximation ratio below 2.0 in more than 50% of test cases under all these four default routing policies.

5.5.2 Loss Maximization

We evaluate the performance of loss maximization under Algorithm 5.5 and 5.6 in single-hop networks. We simulate various settings in 8×8 and 16×16 single-hop networks, considering all source nodes as adversarial, which does not compromise generality as explained in Section 5.3.2. We compare our proposed algorithms with two heuristics: (i) **Min**μ: Each adversarial node routes all the traffic to its connected destination node with the minimum service rate. (ii) **Rand**: Each adversarial node randomly selects a connected destination node and routes all the traffic to it.

We evaluate algorithms across various network settings in three dimensions: (i) Network Density: as in Section 5.5.1. (ii) (μ, λ)-ratio, which represents $\sum_{j=1}^{N_D} \mu_j / \sum_{i=1}^{N_S} \lambda_i$, the ratio between the total service rate and traffic arrival rates. A higher ratio implies lighter traffic loads in the network, thus more challenging for applying routing attacks to cause overload. (iii) Uniformity of service rates $\boldsymbol{\mu}$ among all destination nodes. We consider two scenarios: *heterogeneous* service rates, where the service rates are randomly generated given the (μ, λ)-ratio, and *homogeneous* service rates, where the maximum difference of service rates between any pair of destination nodes is within 10%, i.e., $\max_{j_1 \neq j_2} |\mu_{j_1} - \mu_{j_2}| / \max\{\mu_{j_1}, \mu_{j_2}\} \leq 10\%$, the idea of which is widely adopted in real-world data center networks to avoid speed mismatch [1, 92]. For each network setting, we evaluate 30 random topologies between $\mathcal{V}_S$ and $\mathcal{V}_D$, and 50 different values of $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ per topology, subject to the given constraints of (μ, λ)-ratio and uniformity of $\boldsymbol{\mu}$.

We first present the results on 8×8 networks. We validate the proved performance guarantees of Algorithm 5.5 and 5.6 , since the brute force mechanism in Proposition 5.3 is not prohibitive to simulate under $|\mathcal{V}_A| = 8$. We present the results in Table 5.2. For Algorithm 5.5, we assess the approximation ratio $\Delta_{MUL}/\Delta_{OPT} \in [0, 1]$, which achieves optimum in more than 90% of the tested instances, and the worst approximation ratio is above 2/3 under all tested settings, significantly surpassing the bound $1/\sqrt{|\mathcal{V}_A|} = 1/\sqrt{8} = 0.35$ in Theorem 5.5. For Algorithm 5.6, we evaluate $(\Delta_{OPT} - \Delta_{ADD})/\lambda$, which is less than 0.05 under heterogeneous service rates and close to 0 under homogeneous service rates in 90% of the instances, with the largest gap being less than 0.19, below the bound of 1/4 in Theorem 5.6. These results demonstrate the near-optimal performance in most of the tested cases by both Algorithm 5.5 and 5.6.

We then compare Algorithm 5.5 and 5.6 with the heuristic algorithms Minμ and Rand in Fig. 5-12. Here we only present the CDFs of approximation ratios $\Delta_{ALG}/\Delta_{OPT}$ given density 0.3 and a (μ, λ)-ratio of 2, where the ranking of the tested algorithms remains the same under the metric of performance gap $(\Delta_{ALG} - \Delta_{OPT})/\lambda$. We have the following observations: (i) Algorithm 5.5 and 5.6 significantly outperform the other heuristics, achieving approximation ratios close to 1 under both heterogeneous and homogeneous $\boldsymbol{\mu}$ in most instances. (ii) The advantage of Algorithm 5.5 and 5.6 over Minμ diminishes under heterogeneous $\boldsymbol{\mu}$ compared to homogeneous $\boldsymbol{\mu}$. This is because under heterogeneous service rates, there are instances where a destination node has very low service rates but is connected to a large number of source nodes. Minμ is well-suited to these instances by overloading this destination node, thus closely approximating the optimal attack.

We further evaluate over larger 16×16 networks. The brute-force approach becomes prohibitive to simulate under $|\mathcal{V}_A| = 16$. Therefore we measure the

Den	μ/λ	Alg. 5.5 (Hetero) 10%	Alg. 5.5 (Hetero) Min	Alg. 5.5 (Homo) 10%	Alg. 5.5 (Homo) Min	Alg. 5.6 (Hetero) 90%	Alg. 5.6 (Hetero) Max	Alg. 5.6 (Homo) 90%	Alg. 5.6 (Homo) Max
0.3	2.0	1.00	0.69	1.00	0.75	0.04	0.17	0.00	0.17
0.5	2.0	1.00	0.68	1.00	0.74	0.04	0.14	0.00	0.14
0.3	3.0	1.00	0.73	1.00	0.69	0.01	0.13	0.00	0.06
0.5	3.0	1.00	0.76	1.00	0.71	0.04	0.18	0.00	0.07

Table 5.2: Statistics of performance guarantee for (5.6) in 8×8 networks under different network settings (μ/λ means (μ, λ)-ratio; 10%, 90% mean 10-percentile, 90-percentile)

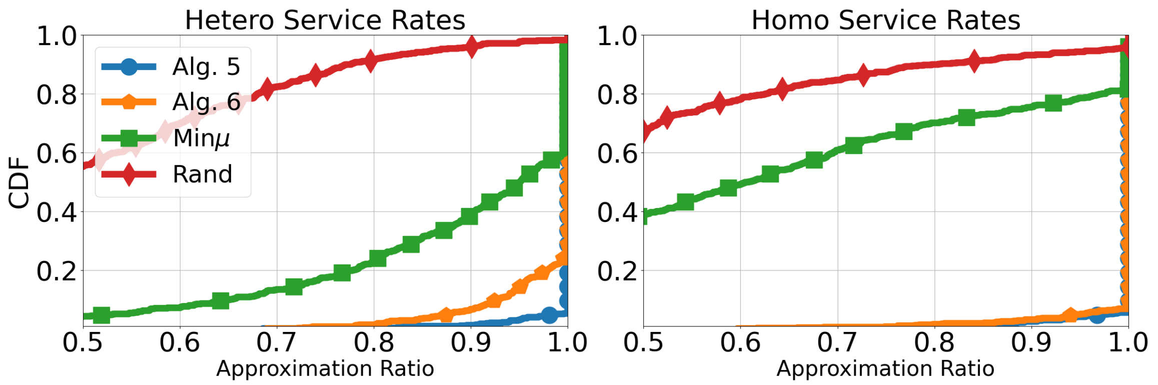


Figure 5-12: CDFs of approximation ratio under density 0.3 and $\mu/\lambda = 2$ in 8×8 networks

alternative metric *loss ratio*, which is the ratio between the loss and total traffic arrival rate. We present the mean loss ratio results among all the tested instances under different settings in Table 5.3 with the following observations. (i) Minμ achieves similar performance to our Algorithm 5.5 and 5.6 under heterogeneous service rates, while far inferior under homogeneous service rates, matching the results in Fig. 5-12. (ii) Under a (μ, λ)-ratio of 8 with homogeneous $\boldsymbol{\mu}$, with high probability there is no way to induce overload by routing attack. However, our proposed Algorithm 5.5 and 5.6 can still cause overload whenever such possibility exists, while Minμ and Rand are prone to miss. We further visualize the CDFs of loss ratio under density 0.25 and a (μ, λ)-ratio of 4 in Fig. 5-13, which echoes the results in Table 5.3. The evaluation reveals that the proposed Algorithm 5.5 and 5.6 can effectively overload the networks with near-optimal performance.

		$\mu/\lambda = 4$ Den$= 0.5$	$\mu/\lambda = 4$ Den$= 0.25$	$\mu/\lambda = 8$ Den$= 0.5$	$\mu/\lambda = 8$ Den$= 0.25$
Hetero	Alg. 5.5	0.79	0.53	0.67	0.37
Hetero	Alg. 5.6	0.78	0.53	0.66	0.37
Hetero	Minμ	0.76	0.49	0.64	0.34
Hetero	Rand	0.14	0.14	0.07	0.07
Homo	Alg. 5.5	0.53	0.29	0.25	0.02
Homo	Alg. 5.6	0.53	0.28	0.25	0.02
Homo	Minμ	0.34	0.11	0.05	0.00
Homo	Rand	0.01	0.01	0.00	0.00

Table 5.3: Mean loss ratio in 16×16 networks

The theoretical optimality of Algorithm 5.1 and the empirical near-optimality of Algorithm 5.5 and 5.6 demonstrate that our proposed routing attack strategies can be used as benchmarks to accurately quantify the network vulnerability to a routing attack with respect to both no-loss throughput minimization and loss maximization.

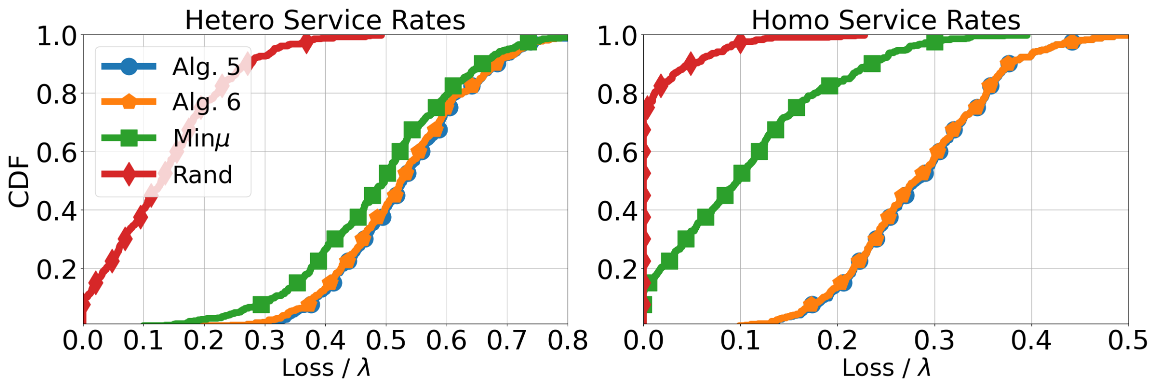
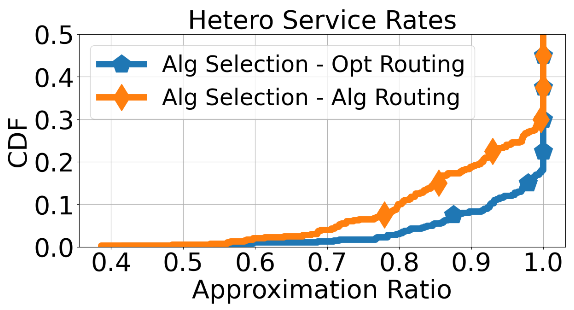


Figure 5-13: CDFs of loss ratio under density 0.25 and $\mu/\lambda = 4$ in 16×16 networks

5.5.3 Optimal Node Selection

We evaluate the performance of Algorithm 5.8 for loss maximization. We consider 400 randomly generated 12×12 single-hop networks, with the link existence probability being 0.3 and (μ, λ)-ratio being 2. The candidate node set $\mathcal{V}_{rand}$ includes all 12 source nodes. We set to choose at most $K = 4$ nodes to attack. We consider heterogeneous and homogeneous service rates at the destination nodes as in Section 5.5.2. The total loss depends on the performance of both the node selection, and the routing attack algorithm based on the selected nodes to hijack. We evaluate the following 3 methods: (i) Optimal node selection + Optimal routing attack, where the node selection is to brute-force all $\binom{12}{4}$ choices, and the optimal routing attack is done based on Proposition 5.3; (ii) Algorithm 5.8 for node selection + Optimal routing attack; (iii) Both node selection and routing attack output by Algorithm 5.8. We calculate the approximation ratio of the total loss under method (ii) and (iii) over the maximum total loss under method (i), and present the CDFs in Fig. 5-14. Results demonstrate that with Algorithm 5.8 for node selection, the adversary can achieve maximum loss in almost 80% of the test cases, and at least 80% of the maximum loss in more than 95% of the test cases, under optimal routing attack, which demonstrates the consistent performance of Algorithm 5.8 in node selection. Furthermore, directly applying the output routing attack solution of Algorithm 5.8, which is of polynomial time complexity for general $\mathcal{V}_{rand}$ and K, has performance close to applying the optimal routing attack: the adversary can achieve maximum loss in almost 70% of the test cases, and at least 80% of the maximum loss in more than 90% of the test cases. The evaluation result demonstrates the good routing attack performance of Algorithm 5.8 in most network settings, showing the high capability of routing attack with node selection to maximize loss.



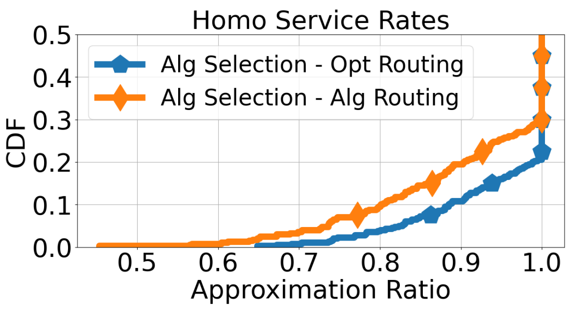


Figure 5-14: CDFs of approximation ratio of overload under density 0.3 and a (μ, λ)-ratio of 2 in 16×16 networks

5.6 Summary and Future Work

In this chapter, we quantify the threat of routing attacks on causing network overload. We investigate the optimal routing attacks for no-loss throughput minimization and loss maximization. We demonstrate that the no-loss throughput can be minimized in polynomial time in general multi-hop networks. We further develop a 2-approximation algorithm by only leveraging the downstream information of the adversarial nodes. We establish that loss maximization is NP-complete and propose two approximation algorithms with guaranteed performance in single-hop networks. Moreover, we address the adversary's optimal selection of nodes to conduct routing attacks and propose heuristic algorithms for this NP-complete problem. Our performance evaluation showcases the near-optimal performance of the proposed algorithms across a wide range of network settings. Future directions include deriving the performance guarantee of loss maximization in multi-hop networks, investigating the case where normal nodes can adjust their routing in response to routing attacks, and designing network control algorithms under routing attacks.

5.7 Chapter Appendix

5.7.1 Proof of Theorem 5.5

Proof. Based on Proposition 5.3, there exists one optimal routing attack where each adversarial node in $\mathcal{V}_A$ sends all its traffic to a destination node. Consider an optimal routing attack w.l.o.g. where source node s_i sends all traffic to destination node d_{j_i}, i.e., $x_{s_i d_{j_i}} = 1$. Denote the total overload of the optimal attack as Δ_{OPT}. Multiple source nodes may choose the same destination node (i.e., $d_{j_{i_1}}$ and $d_{j_{i_2}}$ may be the same for some $i_1 \neq i_2$), hence we remove the repeated elements in $\{d_{j_i}\}_{i=1}^{m}$ into $\{d_j\}_{j=1}^{L'}$, where L' denotes the number of destination nodes that receive traffic from at least one source node. Furthermore, among L' destination nodes, there may exist nodes with no overload where total ingress is no greater than total egress. We further remove these nodes from $\{d_j\}_{j=1}^{L'}$ into $\{d_j\}_{j=1}^{L}$ w.l.o.g., i.e. L is the number of overloaded destination nodes under this optimal routing attack. Note that $L \leq |\mathcal{V}_A|$. In the following, we prove that the approximation ratio under Approach 1 in Algorithm 5.5 is at least $1/L$, and that under Approach 2 is at least $L/|\mathcal{V}_A|$. With these proved, taking the solution that causes higher loss between these two methods causes approximation ratio $\max\{1/L, L/|\mathcal{V}_A|\} \geq 1/\sqrt{|\mathcal{V}_A|}$.

Approach 1: Denote the total overload under the output of Approach 1 as Δ_1. The first step in Approach 1 is the find $j^* \leftarrow \arg\max_j \sum_{(s_i,d_j)\in\mathcal{E}} \lambda_i - \mu_j$. If there is no overload at destination node d_{j^*}, then $\Delta_1 = \Delta_{OPT} = 0$, i.e., no overload can be caused under any routing attack. When there exists overload, then $\Delta_1 \geq \sum_{(s_i,d_{j^*})\in\mathcal{E}} \lambda_i - \mu_{j^*} > 0$, while for any overloaded destination node in $\{d_j\}_{j=1}^{L}$, the overload is less than at d_{j^*} under Approach 1. Therefore

$$\frac{\Delta_1}{\Delta_{OPT}} \geq \frac{\sum_{(s_i,d_{j^*})\in\mathcal{E}} \lambda_i - \mu_{j^*}}{L\left(\sum_{(s_i,d_{j^*})\in\mathcal{E}} \lambda_i - \mu_{j^*}\right)} = \frac{1}{L}.$$

Approach 2: We first prove the case when Approach 2 only runs one iteration to overload a single destination node $d_{j^*} \in \mathcal{V}_D$ according to the definition of PSO in (5.7), and we explain the idea based on the example in Fig. 5-15. Denote the set of connected source nodes that send all their traffic to d_{j^*} in this iteration as $\mathcal{S}_{j^*} = \arg\max_{\mathcal{S}} PSO[d_{j^*}]$. In Fig. 5-15, $j^* = 3$ and $\mathcal{S}_3 = \{s_4, s_5, s_6\}$, where the routing is highlighted in blue[10]. We then denote as $\mathcal{D}_{j^*}$ the set of destination nodes to which the optimal routing attack routes all the traffic in $\mathcal{S}_{j^*}$. In Fig. 5-15, $\mathcal{D}_{j^*} = \mathcal{D}_3 = \{d_2, d_4\}$, with the optimal routing attack highlighted in red. Then we evaluate the ratio between the loss at d_{j^*} under Approach 2, denoted by $\delta^{(2)}_{d_{j^*}}$, and the total loss over $\mathcal{D}_{j^*}$ under optimal solution $\{\mathbf{x}_i^*\}_{s_i \in \mathcal{V}_A}$, denoted by $\sum_{j \in \mathcal{D}_{j^*}} \delta^{OPT}_{d_j}$, which can be lower bounded as follows

$$\frac{\delta^{(2)}_{d_{j^*}}}{\sum_{j \in \mathcal{D}_{j^*}} \delta^{OPT}_{d_j}} = \frac{\left(\sum_{s_i \in \mathcal{S}_{j^*}} \lambda_i\right) - \mu_{j^*}}{\sum_{d_j \in \mathcal{D}_{j^*}} \left(\left(\sum_{s_i : x^*_{s_i d_j} = 1} \lambda_i\right) - \mu_j\right)} = \frac{|\mathcal{S}_{j^*}| \frac{\left(\sum_{s_i \in \mathcal{S}_{j^*}} \lambda_i\right) - \mu_{j^*}}{|\mathcal{S}_{j^*}|}}{\sum_{d_j \in \mathcal{D}_{j^*}} |\mathcal{S}_j^{OPT}| \frac{\left(\sum_{s_i \in |\mathcal{S}_j^{OPT}|} \lambda_i\right) - \mu_j}{|\mathcal{S}_j^{OPT}|}}$$

$$\overset{(a)}{\geq} \frac{|\mathcal{S}_{j^*}| \frac{\left(\sum_{s_i \in \mathcal{S}_{j^*}} \lambda_i\right) - \mu_{j^*}}{|\mathcal{S}_{j^*}|}}{\sum_{d_j \in \mathcal{D}_{j^*}} |\mathcal{S}_j^{OPT}| \frac{\left(\sum_{s_i \in \mathcal{S}_{j^*}} \lambda_i\right) - \mu_{j^*}}{|\mathcal{S}_{j^*}|}} = \frac{|\mathcal{S}_{j^*}|}{\sum_{d_j \in \mathcal{D}_{j^*}} |\mathcal{S}_j^{OPT}|} \overset{(b)}{\geq} \frac{|\mathcal{S}_{j^*}|}{|\mathcal{V}_A|} \overset{(c)}{\geq} \frac{L}{|\mathcal{V}_A|} \tag{5.8}$$

where inequality (a) holds due the optimality of d_{j^*} among all destination nodes w.r.t. (5.7), while inequality (b) holds due to $\cup_{d_j \in \mathcal{D}_{j^*}} \mathcal{S}_j^{OPT} \subseteq \mathcal{V}_A$ (in the example of Fig. 5-15 $\cup_{d_j \in \mathcal{D}_{j^*}} \mathcal{S}_j^{OPT} = \mathcal{V}_A$), and (c) holds since each source node sends all its traffic to a single destination, thus the number of overloaded destination nodes L under optimal attack is at most $|\mathcal{S}_{j^*}|$. In Fig. 5-15, $|\mathcal{S}_{j^*}| = |\mathcal{S}_3| = 3$, $L = 2$, and $|\mathcal{V}_A| = 7$.

□

5.7.2 Proof of Theorem 5.6

Proof. We conduct the proof through three steps: (i) 2×2 single-hop networks; (ii) general single-hop networks while the optimal routing attack only overloads a single destination node; (iii) extension of results in previous step to general single-hop networks.

Case 1: 2×2 *networks.* Consider the 2×2 example in Fig. 5-10 with $\mathcal{V}_S = \{s_1, s_2\}$

[10] Algorithm will choose the blue routing when λ_3 is small so that it will lower down the overload-per-source.

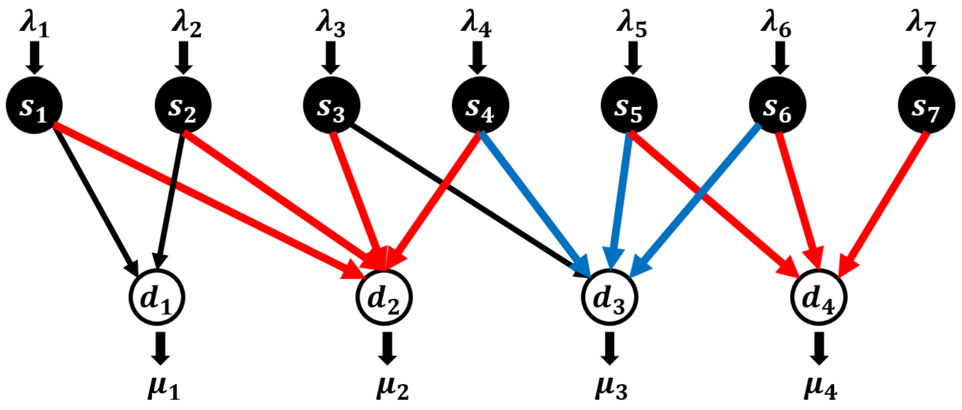


Figure 5-15: Examples for Proof of Approach 2

and $\mathcal{V}_D = \{d_1, d_2\}$, and general $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)$ and $\boldsymbol{\mu} = (\mu_1, \mu_2)$, where we introduce a meta source and destination node equivalently. s_1 can only send traffic to d_1. The routing of s_2 under Algorithm 5.6 is determined by the comparison between $\mu_1/(x_1 + x_2) = \mu_1$ and μ_2/x_2: Sending all traffic to d_1 if $\mu_1 < \mu_2/x_2$ else d_2. If $\mu_1 \leq \mu_2/x_2$, then Δ_{ALG}, the loss by Algorithm 5.6 is $[\lambda - \mu_1]^+$, and in this case the optimal solution has the same loss $\Delta_{OPT} = \Delta_{ALG}$. If $\mu_1 > \mu_2/x_2$, then the $\Delta_{ALG} = [\lambda x_1 - \mu_1]^+ + (\lambda x_2 - \mu_2)$ where $\lambda x_2 > \mu_2$ otherwise node s_2 will not route to d_2. The only possible case that $\Delta_{OPT} > \Delta_{ALG}$ is that the optimal solution is $x_{s_2 d_1} = 1$. In this case, we have the following gap

$$\begin{aligned}\frac{\Delta_{OPT} - \Delta_{ALG}}{\lambda} &= \frac{(\lambda - \mu_1) - [\lambda x_1 - \mu_1]^+ - (\lambda x_2 - \mu_2)}{\lambda} \\ &= \frac{\mu_2 + \lambda x_1 - \mu_1 - [\lambda x_1 - \mu_1]^+}{\lambda} \overset{(i)}{\leq} \frac{\mu_2}{\mu_1} x_1 \overset{(ii)}{\leq} x_1 x_2 \overset{(iii)}{\leq} \frac{1}{4}\end{aligned}$$

where (i) holds as $\lambda = \frac{\mu_1}{x_1}$ maximizes the gap, which is $x_1 - \frac{\mu_1 - \mu_2}{\lambda}$ when $\lambda < \frac{\mu}{x_1}$ and $\frac{\mu_2}{\lambda}$ and $\lambda \geq \frac{\mu}{x_1}$, and note that $\mu_1 > \mu_2/x_2 \geq \mu_2$. (ii) holds due to $\mu_1 > \mu_2/x_2$, and (iii) holds due to $x_1 + x_2 = 1$, thus $x_1 x_2 \leq 1/4$.

Case 2: $M \times N$ networks with the optimal routing overloading one destination node. We extend the example in Fig. 5-10 to general $M = K$ source nodes and $N = K$ destination nodes w.l.o.g., where all sources nodes are connected to d_1, while s_i is at least connected to d_i. We consider the extreme case that will cause biggest gap between the optimal solution and the output from Algorithm 5.6, where the optimal routing is $x^*_{s_i d_1} = 1$, while the algorithm outputs the solution $x_{s_i d_i} = 1$, $\forall i = 1, \cdots, K$, under the condition $\frac{\mu_K}{x_K} \leq \frac{\mu_{K-1}}{x_{K-1}} \cdots \leq \frac{\mu_2}{x_2} \leq \frac{\mu_1}{\sum_{i=1}^K x_i} = \mu_1$. This can be derived by induction based on Case 1. Then we have the bound

$$\begin{aligned}
\frac{\Delta_{OPT} - \Delta_{ALG}}{\lambda} &= \frac{\left(\lambda \sum_{i=1}^K x_i - \mu_1\right) - \sum_{j=1}^K (\lambda x_j - \mu_j)}{\lambda} \\
&= \frac{\lambda x_1 - \mu_1 - [\lambda x_1 - \mu_1]^+ + \sum_{j=2}^K \mu_j}{\lambda} \\
&\leq \frac{\sum_{j=2}^K \mu_j}{\mu_i} x_1 \leq x_1 (\sum_{i=2}^K x_i) = x_1(1 - x_1) \leq \frac{1}{4}
\end{aligned}$$

Case 3: Special case of $M \times N$ networks with the optimal routing attack overloading multiple destination nodes. We now extend to multiple overloaded destination nodes under optimal routing attack. Here we first consider a special case in Fig. 5-16, where the optimal routing attack overloads two destination nodes d_1 and d_4 highlighted in red, while Algorithm 5.6 outputs the routing attack as highlighted in blue, under the conditions that $\mu_2/x_2, \mu_3/x_3 \leq \mu_1/(x_1 + x_2 + x_3)$ and $\mu_5/x_5 \leq \mu_4/(x_4 + x_5)$. Based on the results in Case 2, we can derive

$$\begin{aligned}
\frac{\Delta_{OPT} - \Delta_{ALG}}{\lambda} &= \frac{1}{\lambda}\left(\lambda x_1 - \mu_1 - [\lambda x_1 - \mu_1]^+ + \mu_2 + \mu_3\right) \\
&\quad + \frac{1}{\lambda}\left(\lambda x_4 - \mu_4 - [\lambda x_4 - \mu_4]^+ + \mu_5\right) \\
&\leq \frac{x_1 + x_2 + x_3}{\lambda(x_1 + x_2 + x_3)}\left(\lambda x_1 - \mu_1 - [\lambda x_1 - \mu_1]^+ + \mu_2 + \mu_3\right) \\
&\quad + \frac{x_4 + x_5}{\lambda(x_4 + x_5)}\left(\lambda x_4 - \mu_4 - [\lambda x_4 - \mu_4]^+ + \mu_5\right) \\
&\overset{(a)}{\leq} \frac{1}{4}(x_1 + x_2 + x_3) + \frac{1}{4}(x_4 + x_5) = \frac{1}{4}
\end{aligned}$$

where (a) is based on the result from Case 2. We will use this result in the following discussion over general single-hop networks in Case 4.

Case 4: General single-hop networks. The optimal attack may overload multiple destination nodes. In this case, we can transform the original graph so that results of Case 2 above can be applied. We give an example in Fig. 5-17. Suppose that the optimal routing attack as highlighted in red, while the routing attack from Algorithm

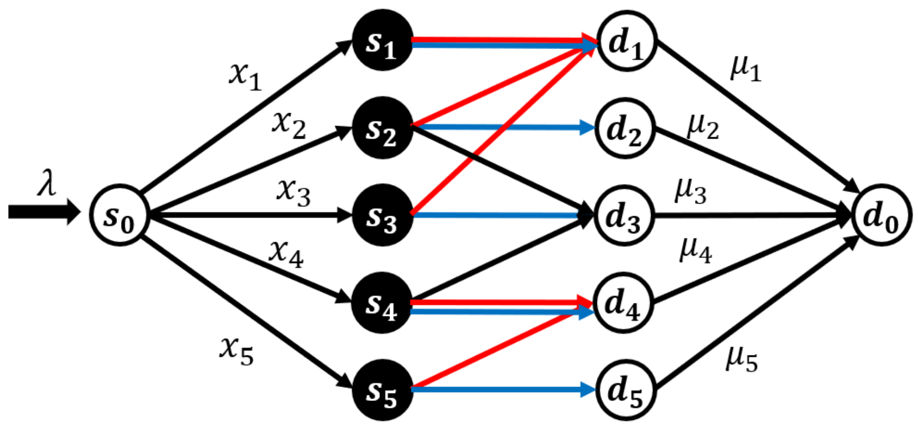


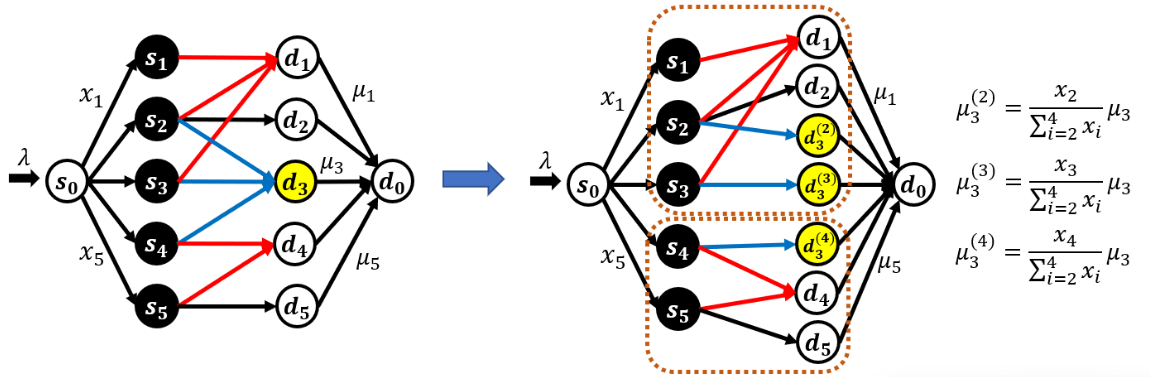
Figure 5-16: Proof for Case 3 of Theorem 5.6

5.6 that minimizes no loss throughput λ^* is to route traffic from s_2, s_3 and s_4 to d_3, highlighted in blue. Given the routing attack output by Algorithm 5.6, denoted by $\mathbf{x}_{\mathcal{A}}^{ALG}$ we show the following transformation of network topology will keep the same overload as $\mathbf{x}_{\mathcal{A}}^{ALG}$: For each overloaded destination node d_j under $\mathbf{x}_{\mathcal{A}}^{ALG}$, suppose that it receives traffic from $\mathcal{S}_j = \{s_i\}_{i:x_{ij}^{ALG}=1}$, then we decompose node d_j into $|\mathcal{S}_j|$ nodes denoted by $\{d_j^{(k)}\}_{k=1}^{|\mathcal{S}_j|}$, each connect to the k-th source node in $\mathcal{S}_j$ denoted by $s^{(k)}$. The new service rate for node $d_j^{(k)}$ is $\mu_j^{(k)} = \frac{x_{s^{(k)}}}{\sum_{k=1}^{|\mathcal{S}_j|} x_{s^{(k)}}}\mu_j$. It is easy to verify that applying Algorithm 5.6 to the transformed network outputs a routing attack solution leads to same overload, as the minimum λ^* that saturates $d_j^{(k)}$, $\forall k$ is the same, equal to the λ^* that saturates d_j in the original graph. For example in Fig. 5-17, node d_3 is decomposed into 3 new nodes with new service rates denoted by $\mu_3^{(2)}, \mu_3^{(3)}$ and $\mu_3^{(4)}$. Note that this transformation is only for proof, where in Algorithm 5.6 we use the original graph.

With the above transformation into multiple basic units in Case 1 (the dashed boxes in Fig. 5-17), we can directly apply the analysis for Case 2, and obtain the upper bound of 1/4. □

5.7.3 Proof of Theorem 5.7

Proof. Under $K = O(1)$, the following brute-force algorithm outputs the optimal solution: Enumerate all $\binom{|\mathcal{V}_{cand}|}{K}$ combinations of nodes to attack, and under each



λ
x_1
x_5
μ_1
μ_3
μ_5
s_0
s_1
s_2
s_3
s_4
s_5
d_0
d_1
d_2
d_3
d_4
d_5
$d_3^{(2)}$
$d_3^{(3)}$
$d_3^{(4)}$
$\mu_3^{(2)} = \frac{x_2}{\sum_{i=2}^4 x_i}\mu_3$
$\mu_3^{(3)} = \frac{x_3}{\sum_{i=2}^4 x_i}\mu_3$
$\mu_3^{(4)} = \frac{x_4}{\sum_{i=2}^4 x_i}\mu_3$

Figure 5-17: Proof for Case 4 of Theorem 5.6

combination apply brute force method for λ^* minimization given by Theorem 6.1, and apply Proposition 5.3 for loss maximization. The time complexity are both $O\left(\binom{|\mathcal{V}_{cand}|}{K} \times NK^{d_{\max}}\right) = O(|\mathcal{V}_{cand}|^K K^{d_{\max}} N)$ where $d_{\max}$ is the maximum degree among the K selected nodes to attack.

Under $K = O(N)$, we reduce Set Cover to the problem of node selection for λ^*-minimization, and the extension to loss maximization is trivial. Given an instance of Set Cover as done in the proof of Theorem 5.4, we construct the graph as in Fig. 5-18, which adds a node T compared with Fig. 5-8 and the added links are highlighted. The candidate node set is $\{s_i\}_{i=1}^m \cup \{d_j\}_{j=1}^n$, and their default routing policies are all dispatching all the traffic to the intermediate node T. All links have unbounded capacity except link (d_0, T) with $c_{d_0T} = 1$. Without routing attack, the no-loss throughput $\lambda^* = \infty$. We show that if we can solve the decision problem of node selection: *Can we reduce* λ^* *from* ∞ *to* 1 *by attacking* $m+k$ *nodes?*, then there exists a polynomial algorithm to decide if there exists k sets in $\{d_i\}_{i=1}^n$ that can cover all elements corresponding to $\{s_i\}_{i=1}^m$. To reduce λ^* to 1, all traffic flows need to go through link (d_0, T) to node T, thus all m source nodes must be attacked to adjust their routing to one of destination nodes instead of directly to T. Then if attacking k more nodes in $\{d_j\}_{j=1}^n$ can reduce λ^* to 1, then it guarantees that all the source nodes route traffic to one of the k attacked destination nodes, and these k nodes routing all the traffic to d_0, which is equivalent to being able to using K sets to cover all elements corresponding to $\{s_i\}_{i=1}^m$.

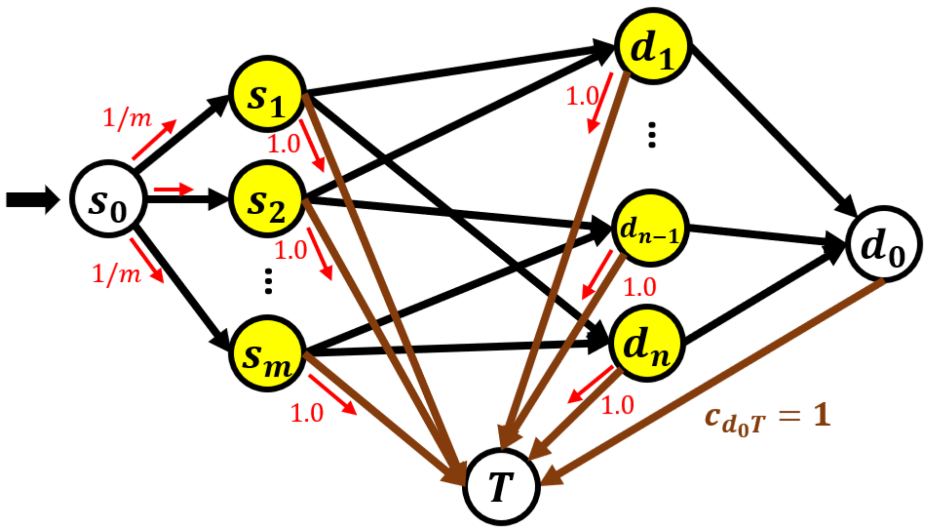


Figure 5-18: NP-hardness of Node Selection Problem

□

5.7.4 Proof of Theorem 5.8

Proof. Algorithm 5.7 goes through each downstream link (i, j) to $\mathcal{V}_{cand}$ (including $\mathcal{V}_{cand}$ themselves), and find the optimal choice of K nodes to attack to minimize the arrival rate that saturates one of them (i, j). Since all candidate nodes are parallel, we can separately evaluate the contribution of attacking a node $v \in \mathcal{V}_{cand}$ over (i, j), denoted by $\Delta_v^{(i,j)}$, which is the difference between the arrival rate that saturates link (i, j) with and without attack. We calculate $\Delta_v^{(i,j)}, \forall v \in \mathcal{V}_A^{up,i}$, and then sort the value over all $\mathcal{V}_A^{up,i}$ non-decreasingly. Denote $\Delta_v^i = \max_{(i,j)\in\mathcal{E}} \Delta_v^{(i,j)}$. Then we can get the optimal choice to each node i as follows, where the choices to each link starting from node i are identical: If $|\mathcal{V}_A^{up,i}| > K$, we choose the top-$K$ nodes in $\mathcal{V}_A^{up,i}$ with highest Δ_v^i values; If $|\mathcal{V}_A^{up,i}| \leq K$, we choose all $|\mathcal{V}_A^{up,i}|$ candidate nodes to attack, where choosing $K - |\mathcal{V}_A^{up,i}|$ candidate nodes does not affect the saturation level of link (i, j) since they are not upstream to node i. Denote the no-throughput loss under the choice $\mathcal{V}_A^{(i)}$ as $\lambda_{(i)}^*$, then the optimal node selection to attack is $\mathcal{V}_A^{(i^*)} = \arg\min_{i^* \in \mathcal{V}_A^{down}} \lambda_{(i)}^*$. □

Chapter 6

Routing Attack on Network Overload with Dynamic Routing

In this chapter, we quantify the maximum throughput loss that can be induced when network adversaries maliciously control the routing policies at a subset of network nodes, given that the other nodes can adjust their routing policies to minimize the loss caused by the adversaries. We unveil the equivalence between maximizing the loss and minimizing the minimum s-d cut value of the network, and prove its NP-hardness under general adversarial node sets. We develop polynomial-time algorithms for adversarial nodes that in a *chain* structure can output the optimal solution that maximizes the loss, and in a *parallel* structure guarantee a logarithmic worst-case approximation ratio to the optimal solution. We further generalize the algorithms for the above two basic structures to an arbitrary structure of adversarial nodes. We validate the near-optimality of the proposed algorithms under a wide range of network settings, which demonstrates their ability to evaluate the potential throughput loss due to overload under malicious routing, and identify the critical nodes to be protected to reduce the impact of routing attack.

6.1 Model, Problem and Basic Results

We start by introducing the network model, the problem formulation of network overload maximization, and basic results including the boundary optimality and the NP-hardness of this problem.

6.1.1 Network Model

We define a network by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of nodes and $\mathcal{E}$ denotes the set of links, with $|\mathcal{V}| = N$ and $|\mathcal{E}| = M$. We denote the node indices by $1, 2, \cdots, N$, and $(i,j) \in \mathcal{E}$ if there is a link from node i to j. We consider a single commodity with node 1 as the source and node N as the destination. We denote the traffic arrival rate to the network at the source by λ. Each node i, representing a switch or a router, can forward the traffic to the nodes adjacent to it, denoted by $\mathcal{V}_i := \{i \in \mathcal{V} | (i,j) \in \mathcal{E}\}$. We denote the transmission rate over link (i,j) by f_{ij}, defined as the amount of traffic transmitted over (i,j) in a time unit. We denote the capacity of a link (i,j) by c_{ij} which is the maximum transmission rate over (i,j), i.e., $f_{ij} \in [0, c_{ij}]$. Traffic departs from the network when it arrives at the destination.

We define the *routing policy* at a non-destination node i by a forwarding ratio vector $\mathbf{x}_i \in \mathbb{R}^N$, where each element x_{ij} denotes the fraction of traffic at node i that will be forwarded to its adjacent node j. The routing policy at a non-destination node i should be in the feasible set $\mathcal{X}_i = \{\mathbf{x}_i \in \mathbb{R}^N \mid \sum_{j=1}^{N} x_{ij} = 1, x_{ij} \geq 0 \text{ for } (i,j) \in \mathcal{E}, x_{ij} = 0 \text{ for } (i,j) \notin \mathcal{E}\}$, where the sum of forwarding ratios of node i to all its adjacent nodes should be 1. Examples include random packet spraying [118], where a node dispatches traffic uniformly to its adjacent nodes, i.e., $x_{ij} = \frac{1}{|\mathcal{V}_i|}$, $\forall (i,j) \in \mathcal{E}$, and weighted spraying based on link capacity where $x_{ij} = \frac{c_{ij}}{\sum_{k:(i,k)\in\mathcal{E}} c_{ik}}$, $\forall (i,j) \in \mathcal{E}$. As we explained in Proposition 5.1, our definition of routing policy based on forwarding ratio vectors is equivalent to the multipath characterization for a commodity [19].

6.1.2 Problem Formulation

We investigate the maximum level of overload that can be induced by malicious routing over a subset of network nodes, denoted by $\mathcal{V}_A$. We term $\mathcal{V}_A$ the set of *adversarial nodes*, and $\mathcal{V}_N := \mathcal{V}\backslash\mathcal{V}_A$ the set of *normal nodes* which are not controlled by the adversary. The traffic transmitted from a node i to an adjacent node j is $\left(\sum_{k:(k,i)\in\mathcal{E}} f_{ki}\right) x_{ij}$. Overload occurs at (i,j) if the traffic exceeds c_{ij}, where $f_{ij} = c_{ij}$ and the excess traffic $\left(\sum_{k:(k,i)\in\mathcal{E}} f_{ki}\right) x_{ij} - c_{ij}$ will be dropped.

We consider that the routing policies at normal nodes $\mathcal{V}_N$ can be reconfigured by the network controller, which adjusts their routing policies $\mathbf{x}_N := \{\mathbf{x}_i\}_{i\in\mathcal{V}_N}$ in order to minimize the throughput loss given the adversary's routing policies $\mathbf{x}_A := \{\mathbf{x}_i\}_{i\in\mathcal{V}_A}$. Such dynamic routing control in response to the change of network states has been widely enabled in modern network infrastructures [89]. In practice this requires knowledge of the attack. However, here the dynamic re-routing allows us to evaluate the fundamental limits on the attack, by considering the worst-case scenario for the adversary. We denote the set of feasible routing policies at $\mathcal{V}_A$ and $\mathcal{V}_N$ by $\mathcal{X}_A := \times_{i\in\mathcal{V}_A}\mathcal{X}_i$ and $\mathcal{X}_N := \times_{i\in\mathcal{V}_N}\mathcal{X}_i$ respectively, where $\times$ represents the Cartesian product, i.e., we require $\mathbf{x}_A \in \mathcal{X}_A$ and $\mathbf{x}_N \in \mathcal{X}_N$. We term $\mathbf{x}_A$ a *routing attack*.

We give the formal definition of our problem: Given the network $\mathcal{G} = (\mathcal{V},\mathcal{E})$ and the adversarial node set $\mathcal{V}_A \subseteq \mathcal{V}$, find the optimal routing attack $\mathbf{x}_A \in \mathcal{X}_A$ that maximizes throughput loss, given that the network controller can optimize $\mathbf{x}_N \in \mathcal{X}_N$ to minimize such loss in response to $\mathbf{x}_A$. We formulate the problem by the following minimax optimization framework (6.1).

$$\begin{aligned} \max_{\mathbf{x}_A\in\mathcal{X}_A} \quad \min_{\mathbf{x}_N\in\mathcal{X}_N} \quad & \lambda - \sum_{i:(i,N)\in\mathcal{E}} f_{iN} \\ \text{s.t.} \quad & f_{ij} = \min\left\{\left(\sum_{k:(k,i)\in\mathcal{E}} f_{ki}\right) x_{ij}, c_{ij}\right\},\ \forall i \neq 1,\ (i,j)\in\mathcal{E}, \\ & f_{1j} = \min\{\lambda x_{1j}, c_{1j}\},\ \forall (1,j)\in\mathcal{E}. \end{aligned} \tag{6.1}$$

The objective function $\lambda - \sum_{i:(i,N)\in\mathcal{E}} f_{iN}$ of (6.1) represents the total amount of overload during the transmission: the gap between the traffic arrival rate λ at the source and the total traffic $\sum_{i:(i,N)\in\mathcal{E}} f_{iN}$ that arrives at the destination. The constraints reflect the flow conservation law over each link: the total traffic forwarded to a link (i,j) should be all transmitted through the link if it is no greater than c_{ij}, otherwise $f_{ij} = c_{ij}$ is transmitted and the excess traffic is dropped. The adversary aims to find the $\mathbf{x}_A \in \mathcal{X}_A$ that maximizes the minimum loss achievable by some $\mathbf{x}_N \in \mathcal{X}_N$.

Analyzing (6.1) directly is challenging due to the constraints with the element-wise minimum function. We show in Proposition 6.1 that the original formulation (6.1) is equivalent to the following formulation (6.2).

$$\begin{aligned}
\min_{\mathbf{x}_A\in\mathcal{X}_A} \quad & \max_{\mathbf{f}} \quad \sum_{i:(i,N)\in\mathcal{E}} f_{iN} \\
\text{s.t.} \quad & f_{ij} = \left(\sum_{k:(k,i)\in\mathcal{E}} f_{ki}\right) x_{ij}, \forall i \in \mathcal{V}_A, (i,j)\in\mathcal{E}, i \neq 1, \\
& \sum_{j:(i,j)\in\mathcal{E}} f_{ij} = \sum_{k:(k,i)\in\mathcal{E}} f_{ki}, \forall i \in \mathcal{V}_N, i \neq 1, \\
& f_{ij} \in [0, c_{ij}],\ \forall (i,j)\in\mathcal{E}.
\end{aligned} \tag{6.2}$$

Proposition 6.1. *The sets of optimal solutions of* (6.1) *and* (6.2) *are the same.*

The transformed problem (6.2) is to find the routing attack that minimizes the maximum total network throughput that can be achieved without inducing loss. The constraints of (6.2) are different from the original problem (6.1), where the flow conservation law at both adversarial and normal nodes holds without loss. The first constraint guarantees that at each $i \in \mathcal{V}_A$, the transmission rate over link (i,j) is equal to the total transmission rates injected into i multiplied by the forwarding ratio x_{ij}; The second constraint states that each $i \in \mathcal{V}_N$ can adjust the routing policy arbitrarily to avoid loss, where the routing policy at $i \in \mathcal{V}_N$ can be obtained from the decision variables $\{f_{ij}\}_{(i,j)\in\mathcal{E}}$ by $x_{ij} = f_{ij}/\sum_{k:(k,i)\in\mathcal{E}} f_{ki}$.

Proposition 6.1 reveals an important property of (6.2) that the optimal routing attack is independent of the traffic arrival rate λ. We point out that this is a unique property for the dynamic routing control over $\mathbf{x}_N$, which does not hold if normal nodes have static routing policies. The transformation to (6.2) facilitates the analysis by removing the dimension of λ and simplifying the constraints. We develop basic analytical results based on (6.2) below that inspire routing attack design to optimize (6.2).

6.1.3 Basic Results

We demonstrate that there exists a polynomial algorithm to solve (6.2) when the number of adversarial nodes $|\mathcal{V}_A|$ is a constant, i.e., $|\mathcal{V}_A| = O(1)$, while (6.2) is NP-hard for general $\mathcal{V}_A$ with $\mathcal{V}_A = O(N)$. We first prove that given any network $\mathcal{G}$ and adversarial node set $\mathcal{V}_A$, there exists an optimal solution to (6.2) by routing all the traffic at each adversarial node to one of its adjacent nodes.

Theorem 6.1. *There exists an optimal solution* $\mathbf{x}_A^* \in \mathcal{X}_A$ *to* (6.2) *which satisfies that for* $\forall i \in \mathcal{V}_A$, $\exists j \in \mathcal{V}_i$ *so that* $x_{ij}^* = 1$.

Theorem 6.1 states that there must exist a vertex of the polytope $\mathcal{X}_A$ that optimizes (6.2). We have $\prod_{i\in\mathcal{V}_A} |\mathcal{V}_i| = O(N^{|\mathcal{V}_A|})$ vertices in $\mathcal{X}_A$, one of which is an optimal solution to (6.2). We can enumerate all these vertices and find the one that minimizes the objective in (6.2), i.e., the maximum throughput without inducing loss. This objective can be calculated in polynomial time when $\mathbf{x}_A$ is a vertex of $\mathcal{X}_A$ by evaluating the *minimum s-d cut* of a network, defined as the set of links with minimum total capacity whose removal can disconnect the source node 1 and destination node N. Specifically, at a vertex $\mathbf{x}_A$ of $\mathcal{X}_A$, each $i \in \mathcal{V}_A$ forwards all the traffic to a single adjacent node $j \in \mathcal{V}_i$, and thus the original network $\mathcal{G}$ under $\mathbf{x}_A$ is equivalent to a network $\mathcal{G}'$ where each $i \in \mathcal{V}_A$ has only one egress link (i, j) with capacity $c'_{ij} = c_{ij}$ and the other links (i, k) with capacity $c'_{ik} = 0$ for $k \neq j$. Therefore the maximum throughput without loss of $\mathcal{G}$ is equal to the minimum s-d cut value of $\mathcal{G}'$, which can be solved in polynomial time by the Edmonds-Karp algorithm [119] or Karger's algorithm [120]. We remark that the equivalence between (6.2) under $\mathcal{G}$ and the minimum s-d cut of $\mathcal{G}'$ only holds when $\mathbf{x}_A$ is a vertex of $\mathcal{X}_A$, as Fig. 5-2 shows.

The above discussion demonstrates that the original problem (6.1) can be formulated as a combinatorial optimization problem: Each $i \in \mathcal{V}_A$ needs to choose a single adjacent node $j \in \mathcal{V}_i$ and activate the link (i, j) with the goal of minimizing the minimum s-d cut value of the network. The brute-force algorithm above is only efficient in practice when the size of $\mathcal{V}_A$ is a very small constant. The time complexity becomes exponential when $|\mathcal{V}_A|$ scales linear with network size N. This is typical in SDN where a routing controller is responsible for the routing policies of a fixed proportion of data plane nodes, for example each of the Google's SDN controller taking over 25% of the networks [89]. We show in Theorem 6.2 that the problem (6.2) is NP-hard given general size of $|\mathcal{V}_A|$ by reducing the Set Cover problem to (6.2). This result demonstrates the need to develop approximation algorithms with acceptable performance guarantees and polynomial time complexity.

Theorem 6.2. *Problem* (6.2) *is NP-hard under* $|\mathcal{V}_A| = O(N)$.

6.2 Algorithms and Performance Guarantees

We develop polynomial-time routing attack algorithms with performance guarantees for the problem (6.2), given that $\mathcal{V}_A$ follows either a *chain* and a *parallel* structure. We prove that the algorithm for $\mathcal{V}_A$ in a chain structure can output the exact optimal solution to (6.2), and the algorithm for $\mathcal{V}_A$ in a parallel structure leads to an approximation ratio at most $O(\log |\mathcal{V}_A|)$, over arbitrary network instance $\mathcal{G}$.

6.2.1 Chain Structure

We first study the case where $\mathcal{V}_A$ is in a chain structure. The definition of the chain structure is based on the *reachable* nodes of adversarial nodes.

Definition 6.1. *Given the network $\mathcal{G}$ and the adversarial node set $\mathcal{V}_A$, a node j is reachable from $i \in \mathcal{V}_A$, denoted by $i \to j$, if there exists a routing attack over $\mathcal{V}_A$ such that all the traffic at i will arrive at node j, given arbitrary routing policies of $\mathcal{V}_N$.*

We explain the definition of reachable nodes in Fig. 6-1, where in both (a) and (b), $2 \to 4$, since taking $x_{23} = 1$ for (b) and further $x_{34} = 1$ for (a) guarantees all the traffic at node 2 are sent to 4, while in (c) $2 \not\to 4$ since node 3 can forward traffic to node 6. The reachability is important in the routing attack algorithm design below, which characterizes the ability of an adversarial node i to forward traffic to overload the links starting from another node j. If node j is not reachable from i, the dynamic routing at $\mathcal{V}_N$ can prevent this attempt by diverting traffic away from node j.

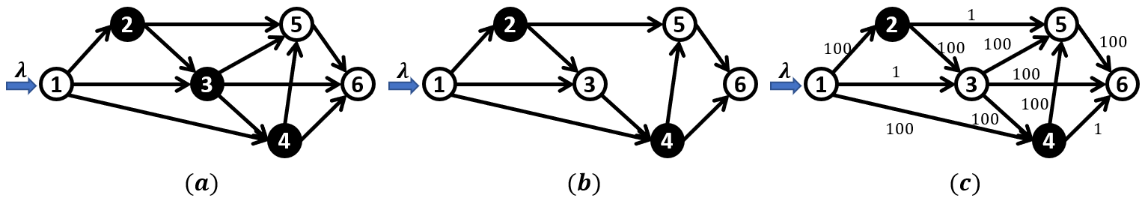


Figure 6-1: Examples of reachability and the chain structure of $\mathcal{V}_A$.

We define an adversarial node set $\mathcal{V}_A$ to be in a chain structure based on the reachability.

Definition 6.2. *The adversarial node set $\mathcal{V}_A$ is in a chain structure if there exists a permutation of nodes in $\mathcal{V}_A$, denoted by $\{i_1, \ldots, i_{|\mathcal{V}_A|}\}$, such that $i_j \to i_{j+1}$, $\forall j = 1, \ldots, |\mathcal{V}_A| - 1$.*

Intuitively, the chain structure $\{i_1, \ldots, i_{|\mathcal{V}_A|}\}$ facilitates overloading the network in that an adversarial node i_j can target overloading links starting from node i' with the guarantee that all the traffic at i_j can be routed to i' as long as node i' is in the union of reachable nodes of $\{i_k\}_{k=j}^{|\mathcal{V}_A|}$. In Fig. 6-1, the $\mathcal{V}_A$'s in (a) and (b) are in a chain structure, while that in (c) is not. Notice that if $i \to j$ and $j \to k$, then $i \to k$. Moreover, the chain only requires the existence of a permutation, where there may exist loops where $i_{j+1} \to i_j$ for some j. $\mathcal{V}_A$ in a chain structure is plausible in real networks, where the adversary hijacks the nodes over a physical path of the traffic in a network, or a controller in the control plane that is responsible for a physically local mesh network where nodes are connected to each other.

We develop Alg. 6.1 with polynomial time complexity for $\mathcal{V}_A$ in a chain structure, and prove in Theorem 6.3 that it can output the exact optimal routing attack to (6.2).

Algorithm 6.1: Exact Algorithm to (6.2) for $\mathcal{V}_A$ in Chain

```
1 Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V}_A$ in a chain structure;
2 Find a permutation of $\mathcal{V}_A$, $\{i_1, \ldots, i_{|\mathcal{V}_A|}\}$, such that
    $i_j \to i_{j+1}$, $\forall j = 1, \ldots, |\mathcal{V}_A| - 1$;
3 Route the traffic from each $i_j \in \mathcal{V}_A$ to an adjacent node so that all the traffic
    at $i_j$ can be transmitted to $i_{j+1}$, $\forall j = 1, \ldots, |\mathcal{V}_A| - 1$;
4 for $j = |\mathcal{V}_A|$ to 1 do
5     for $k$ in $\mathcal{V}_j$ do
6         Set $x_{i_j k} = 1$ and $x_{i_j k'} = 0$, $\forall k' \neq k$;
7         Calculate the minimum s-d cut value of $\mathcal{G}$ by assuming
            $c_{i_j k'} = 0, \forall k' \neq k$, denoted by $C_k$;
8     Let $k^* = \arg\min_{k \in \mathcal{V}_j} C_k$, and determine the routing policy of node $i_j$ as
        $x_{i_j k^*} = 1$ and $x_{i_j k'} = 0, k' \neq k^*$;
9 Return $\mathbf{x}_A = \{\mathbf{x}_i\}_{i \in \mathcal{V}_A}$;
```

Theorem 6.3. *Alg. 6.1 outputs an optimal routing attack to* (6.2) *if* $\mathcal{V}_A$ *is in a chain structure.*

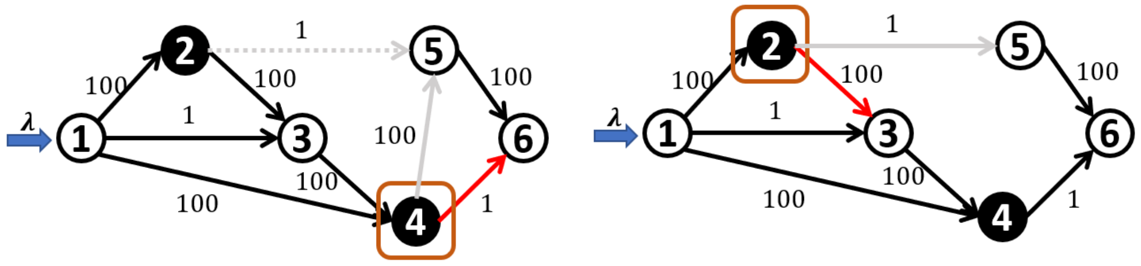


Figure 6-2: Alg. 6.1 on Fig. 6-1(b): first on the left, we temporarily set the routing policy of node 2 to be $x_{23} = 1$ so that all the traffic at 2 is sent to 4, and determines the routing policy at 4 to be $x_{46} = 1$ to achieve minimum s-d cut value 1, under which no traffic goes through $(2, 5)$ and $(4, 5)$; then on the right, we fix $\mathbf{x}_4$ and determine the routing policy at 2 to be $x_{23} = 1$ which keeps the minimum s-d cut value to be 1.

We explain how Alg. 6.1 works by going through the first iteration which determines the routing policy of $i_{|\mathcal{V}_A|}$: Alg. 6.1 temporarily sets the routing policies at adversarial nodes $\{i_1, \ldots, i_{|\mathcal{V}_A|-1}\}$ following the chain $i_j \to i_{j+1}$, $\forall j = 1, \ldots, |\mathcal{V}_A| - 1$, which guarantees that the traffic sent from i_j will all arrive at i_{j+1}. It then determines the optimal routing policy of $i_{|\mathcal{V}_A|}$ based on Theorem 6.1 by sending all the traffic from $i_{|\mathcal{V}_A|}$ to each of its adjacent nodes and computing their corresponding minimum s-d cut values. The optimal routing policy at $i_{|\mathcal{V}_A|}$ is the one with the lowest minimum s-d cut value. In the k-th iteration of Alg. 6.1, we determine the routing policy of node $i_{|\mathcal{V}_A|+1-k}$ as we do in the first iteration, given that the routing policies of $\{i_j\}_{j=|\mathcal{V}_A|+2-k}^{|\mathcal{V}_A|}$ have been determined in previous iterations, and the routing policies of $\{i_j\}_{j=1}^{|\mathcal{V}_A|-k}$ follow the chain temporarily. Fig. 6-2 shows how Alg. 6.1 works over the example in Fig. 6-1(b) where the numbers over links represent the link capacities.

The worst-case time complexity of Alg. 6.1 is $O(|\mathcal{V}_A| \times |\mathcal{V}| \times T_{\text{mincut}})$: In each iteration, the algorithm determines the routing policy of an adversarial node $i \in \mathcal{V}_A$, which involves traversing $|\mathcal{V}_i| = O(N)$ adjacent nodes of i, with worst-case time complexity of calculating minimum cut $O(T_{\text{mincut}}) = O(|\mathcal{V}||\mathcal{E}|^2) = O(NM^2)$ using Edmonds-Karp algorithm or $O(T_{\text{mincut}}) = O(|\mathcal{V}|^2) = O(N^2)$ using Karger's algorithm. Alg. 6.1 can be accelerated in real implementation. First, traversing the $\mathcal{V}_i$ for each $i \in \mathcal{V}_A$ can be implemented in parallel, where parallelization over K machines reduces the complexity to $O(|\mathcal{V}_A| \times \frac{|\mathcal{V}|}{K} \times T_{\text{mincut}})$. Second, it is not always necessary to traverse $\mathcal{V}_i$ at each $i \in \mathcal{V}_A$. The idea is that at $i \in \mathcal{V}_A$, we can first obtain the set of links that constitute the minimum s-d cut under the assumption that $c_{ik} = 0$, $\forall k \in \mathcal{V}_i$. If the set of links does not contain a link (i, k) for some $k \in \mathcal{V}_i$, then it means that adjusting the routing policy at node i does not affect the minimum s-d cut value, and thus we can randomly find $j \in \mathcal{V}_i$ and forward all the traffic at i to node j, which does not affect the optimality of the output. This optimized approach does not lower the worst-case complexity but can remove the unnecessary computation of minimum s-d cut to reduce computation time in general networks.

Remarks: (i) Alg. 6.1 is optimal as long as there *exists* a permutation $\{i_1, \ldots, i_{|\mathcal{V}_A|}\}$ of $\mathcal{V}_A$ that forms a chain. The existence of loops where nodes $i_{j_1}, i_{j_2} \in \mathcal{V}_A$ such that $i_{j_1} \to i_{j_2}$ and $i_{j_2} \to i_{j_1}$ does not affect the optimality. Consider the example where $i_{|\mathcal{V}_A|} \to i_1$ in the chain, then an optimal routing attack is to route all the traffic at i_j to a node in $\mathcal{V}_{i_j}$ that they can all arrive at i_{j+1}, and route the traffic at $i_{|\mathcal{V}_A|}$ back to i_1. This routing attack guarantees that any traffic forwarded to an adversarial node will be trapped in the loop and thus cannot reach the destination. Alg. 6.1 will output this solution, since in the first iteration the routing policy of $i_{|\mathcal{V}_A|}$ will be determined to be routing following $i_{|\mathcal{V}_A|} \to i_1$ so that all the traffic is sent back to i_1, which does not increase the minimum s-d cut value. (ii) The verification and identification of chain structure are straightforward. We can validate if $i \to j$ by the following method: Suppose that each node uniformly forwards the traffic to its adjacent nodes, and there is a close-to-zero amount of traffic ϵ starting at i that will not cause any overload, then $i \to j$ if and only if all the traffic can reach node j. We can identify all the chain structures in the network by applying the above validation method for each pair of network nodes. (iii) The optimality of Alg. 6.1 is based on the prerequisite that $\mathcal{V}_A$ is in a chain structure. Consider the example in Fig. 6-1(c) where $2 \not\to 4$. Suppose that to determine the routing policy at node 4, we set the routing at node 2 to route all the traffic to node 3, then the output routing attack will be $x_{45} = 1$ and $x_{25} = 1$, since in the first iteration setting $x_{45} = 1$ gives a minimum cut value of 200 while $x_{46} = 1$ gives 201. This attack leads to a minimum s-d cut value of 102. However, the optimal routing attack is to set $x_{46} = 1$ and $x_{25} = 1$, under which the minimum s-d cut value is 3. This fact motivates the need to develop alternative algorithms for $\mathcal{V}_A$ not in a chain structure.

6.2.2 Parallel Structure

We further investigate the case where $\mathcal{V}_A$ is in a parallel structure, as defined below.

Definition 6.3. *The adversarial node set $\mathcal{V}_A$ is in a parallel structure if for $\forall i \in \mathcal{V}_A$ and any routing policy $\mathbf{x}_i$, there exists routing policies over normal nodes $\mathcal{V}_N$ so that the traffic can be delivered from the source to the destination through i without passing through other adversarial nodes.*

The definition requires that no matter what routing attack is taken over $\mathcal{V}_A$, for each adversarial node i, there exists a path from the source to the destination where the only adversarial node on it is i under some routing policy over $\mathcal{V}_N$. The intuition is that for each pair of adversarial nodes i and j, normal nodes can adjust the routing so that the traffic at node i will not be delivered to j. Real-world examples of $\mathcal{V}_A$ in a parallel structure include the load balancers in a server farm, and a middle layer of nodes in a multi-layer network that serves a commodity from the ingress to the egress layer.

We have the following remarks: (i) $\mathcal{V}_A$ in a parallel structure implies that each pair of adversarial nodes are not reachable from one to another. However, the reverse does not hold, i.e., $i \not\rightarrow j$ for $\forall i, j \in \mathcal{V}_A$, $i \neq j$ does not imply $\mathcal{V}_A$ in a parallel structure. We give an example in Fig. 6-3 and leave the discussion to Section 6.3. (ii) Definition 6.3 does not mean that there is no path between a pair of adversarial nodes. Fig. 6-1(c) is an example where adversarial nodes $\{2, 4\}$ are parallel since $2 \not\rightarrow 4$, although there exists a path 2 to 3 to 4.

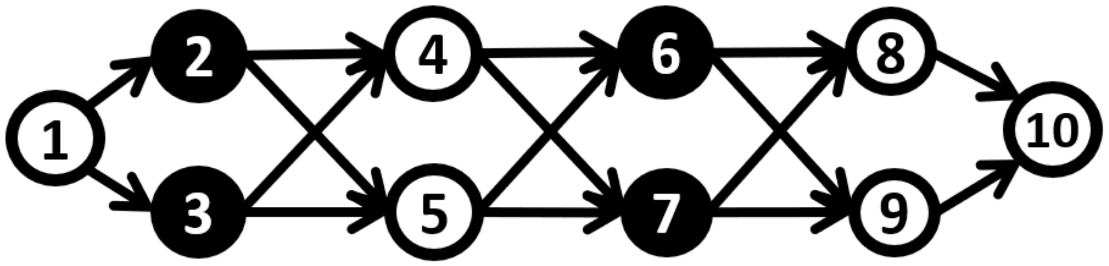


Figure 6-3: Example where adversarial nodes at different layers (shaded in black) in a multi-layer network that are not in a parallel structure, although $i \not\rightarrow j$ for $\forall i, j \in \mathcal{V}_A$ and $i \neq j$.

We develop Alg. 6.2 that can output a routing attack in polynomial time given that $\mathcal{V}_A$ is in a parallel structure. The idea behind Alg. 6.2 is to determine the routing policies at adversarial nodes so that the average contribution of an adversarial node on increasing the minimum s-d cut value is minimized in each iteration. Fig. 6-4 gives an example to explain how Alg. 6.2 works by going through the first iteration. Alg. 6.2 investigates the minimum increment of the minimum s-d cut value by two steps. The first step (line 5 to 8) is to go over each link (i, j) where $i \in \mathcal{V}_A$, and calculate the increment of the minimum s-d cut value, denoted by Δ_{ij}, given that the adversarial node i forwards all the traffic to node j. The second step (line 9 to 12) is to investigate the normal node set $\mathcal{V}_N$. For each normal node k, we figure out the set of adversarial nodes from which node k is reachable, denoted by $\mathcal{R}_k$ where $\mathcal{R}_k \subseteq \mathcal{V}_A$ and $\forall i \in \mathcal{R}_k,\ i \rightarrow k$. The algorithm skips k if $\mathcal{R}_k = \emptyset$, otherwise it routes the traffic at each node $i \in \mathcal{R}_k$ so that all the traffic is transmitted to node k. The algorithm calculates the corresponding increment of the minimum s-d cut value based on the above routing policies at $\mathcal{R}_k$, denoted by Δ_k, and evaluate the average increment from each adversarial node in $\mathcal{R}_k$, which is $\Delta_k/|\mathcal{R}_k|$. Finally, the algorithm chooses the minimum value from the union of $\{\Delta_{ij}\}_{i\in\mathcal{V}_A,(i,j)\in\mathcal{E}}$ and $\{\Delta_k/|\mathcal{R}_k|\}_{k\in\mathcal{V}_N,\mathcal{R}_k\neq\emptyset}$, and determines the routing policies of the adversarial nodes corresponding to the choice as done in line 14 or line 17. Alg. 6.2 iterates the above process over the adversarial nodes whose routing policies have not been determined, and terminates when the routing policies of all adversarial nodes are determined.

Algorithm 6.2: $O(\log|\mathcal{V}_A|)$-Approximation Algorithm to (6.2) for $\mathcal{V}_A$ in a Parallel Structure

```
1  Input: $\mathcal{G}=(\mathcal{V},\mathcal{E})$, $\mathcal{V}_A$ in a parallel structure;
2  Initialize the undetermined adversarial node set $UD \leftarrow \mathcal{V}_A$, and
     $\mathbf{x}_i=\mathbf{0},\ \forall i\in\mathcal{V}_A$;
3  For each $i\in\mathcal{V}_N$, calculate $\mathcal{R}_i$, the set of adversarial nodes from which $i$ is
     reachable;
4  while $UD\neq\emptyset$ do
5      for link $(i,j)$ where $i\in UD$ do
6          Set $x_{ij}=1$ and $x_{ij'}=0,\ j'\neq j$;
7          Calculate $\Delta_{ij}$, the increment of minimum s-d cut value;
8      Find $(i^*,j^*)\leftarrow \arg\min_{i\in\mathcal{V}_A,(i,j)\in\mathcal{E}}\Delta_{ij}$;
9      for $k\in\mathcal{V}_N$ and $\mathcal{R}_k\cap UD\neq\emptyset$ do
10         Each adversarial node $i\in\mathcal{R}_k\cap UD$ routes all traffic to a node in $\mathcal{V}_i$
             s.t. all the traffic is guaranteed to reach node $k$;
11         Calculate $\Delta_k$, the increment of minimum s-d cut value under the
             above routing;
12     Find $k^*\leftarrow \arg\min_{k\in\mathcal{V}_N}\Delta_k/|\mathcal{R}_k\cap UD|$;
13     if $\Delta_{i^*j^*}\leq\Delta_{k^*}/|\mathcal{R}_{k^*}\cap UD|$ then
14         Determine the routing policy at $i^*$ to be $x_{i^*j^*}=1$ and
             $x_{i^*j}=0,\ \forall j\neq j^*$;
15         $UD\leftarrow UD\backslash\{i^*\}$; $\mathcal{R}_k\leftarrow\mathcal{R}_k\backslash\{i^*\}$, $\forall k\in\mathcal{V}_N$;
16     else
17         Determine the routing policy of each node in $\mathcal{R}_{k^*}\cap UD$ so that all
             traffic at this node is guaranteed to be sent to $k^*$;
18         $UD\leftarrow UD\backslash\mathcal{R}_{k^*}$; $\mathcal{R}_k\leftarrow\mathcal{R}_k\backslash\mathcal{R}_{k^*}$, $\forall k\in\mathcal{V}_N$;
19 Return $\mathbf{x}_A=\{\mathbf{x}_i\}_{i\in\mathcal{V}_A}$;
```

The worst-case time complexity of Alg. 6.2 is $O(|\mathcal{V}_A|\times(|\mathcal{E}|+|\mathcal{V}_N|)\times T_{\text{mincut}})$: Alg. 6.2 has at most $|\mathcal{V}_A|$ iterations where each iteration only determines the routing

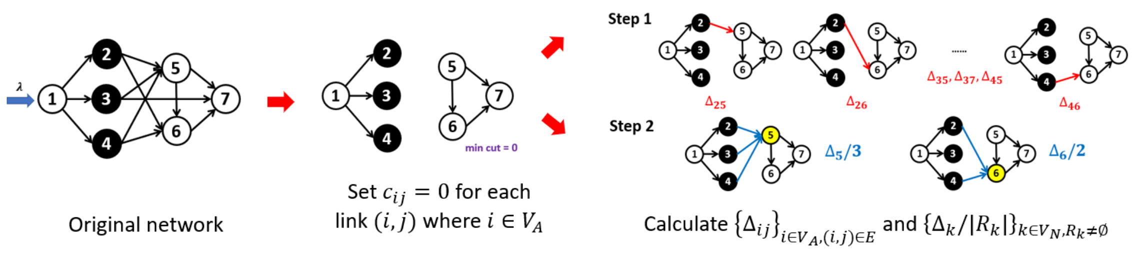

λ
Original network
min cut = 0
Set $c_{ij} = 0$ for each link (i,j) where $i \in V_A$
Step 1
Δ_{25}
Δ_{26}
$\Delta_{35}, \Delta_{37}, \Delta_{45}$
Δ_{46}
Step 2
$\Delta_5/3$
$\Delta_6/2$
Calculate $\{\Delta_{ij}\}_{i \in V_A,(i,j)\in E}$ and $\{\Delta_k/|R_k|\}_{k\in V_N, R_k \neq \emptyset}$

Figure 6-4: Example of the first iteration of Alg. 6.2 for $\mathcal{V}_A$ in a parallel structure: first set $c_{ij} = 0$ for all the links (i,j) starting from an adversarial node i; then follow line 5 to 7 to calculate the minimum s-d cut value increment by activating each of the above links, and follow line 9 to 11 to calculate the minimum s-d cut value increment by activating all adversarial nodes reachable to each normal node, and obtain the mean minimum cut value increment by dividing the value by the number of adversarial node reachable to this normal node.

policy of a single adversarial node in the worst case, and in each iteration calculates $|\{\Delta_{ij}\}_{i\in\mathcal{V}_A,(i,j)\in\mathcal{E}}|+|\{\Delta_k/|\mathcal{R}_k|\}_{k\in\mathcal{V}_N,\mathcal{R}_k\in\emptyset}| = O(|\mathcal{E}|+|\mathcal{V}_N|)$ times of the minimum s-d cut value. We can accelerate the process in real implementation by pre-computing $\mathcal{R}_i$ for $\forall i \in \mathcal{V}_N$, parallel computing $\{\Delta_{ij}\}_{i\in\mathcal{V}_A,(i,j)\in\mathcal{E}}$ and $\{\Delta_k/|\mathcal{R}_k|\}_{k\in\mathcal{V}_N}$ in each iteration, and calculating the minimum s-d cut value only if the newly introduced routing policies at $\mathcal{V}_A$ in this iteration increases the minimum s-d cut value, similar as discussed in Alg. 6.1.

We prove in Theorem 6.4 that Alg. 6.2 outputs a routing attack which leads to a worst-case approximation ratio of $O(\log|\mathcal{V}_A|)$ to the optimal solution to (6.2) given an arbitrary network instance $\mathcal{G}$ and adversarial node set $\mathcal{V}_A$ in a parallel structure. This logarithmic gap demonstrates the slow performance degradation of Alg. 6.2 when $|\mathcal{V}_A|$ scales linearly with the network size $|\mathcal{V}|$. The approximation ratio is asymptotic given that $|\mathcal{V}_A| \to \infty$ with $|\mathcal{V}| \to \infty$. For a constant $|\mathcal{V}_A|$, the worst-case approximation ratio is $\sum_{i=1}^{|\mathcal{V}_A|} i^{-1}$.

Theorem 6.4. *Alg. 6.2 is an $O(\log|\mathcal{V}_A|)$-approximation algorithm to the problem* (6.2) *given $\mathcal{V}_A$ in a parallel structure.*

We further demonstrate in Theorem 6.5 that $O(\log|\mathcal{V}_A|)$ is the minimum worst-case approximation ratio that can be achieved by any polynomial-time algorithm for the general problem (6.2). The proof idea is based on the inapproximability of the Set Cover problem. For $\mathcal{V}_A$ in a parallel structure, it means that there does not exist another polynomial-time algorithm that can perform better than Alg. 6.2 in terms of the worst-case approximation ratio.

Theorem 6.5. *There is no polynomial-time algorithm with a worst-case approximation ratio lower than* $O(\log|\mathcal{V}_A|)$ *to* (6.2).

6.3 Algorithm for General $\mathcal{V}_A$

In this section, we develop a polynomial-time algorithm given that the adversarial node set $\mathcal{V}_A$ is in a *general* structure, which means $\mathcal{V}_A$ can be an arbitrary subset of nodes. We provide the pseudo-code in Alg. 6.3 which is a recursive algorithm. We explain the two major challenges when generalizing the algorithms designed in Section 6.2 to $\mathcal{V}_A$ in a general structure below.

Algorithm 6.3: Algorithm to (6.2) for General $\mathcal{V}_A$

```
1 Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V}_A$ in a general structure;
2 Initialize $\mathbf{x}_i = \mathbf{0}$, $\forall i \in \mathcal{V}_A$;
3 Find adversarial nodes $\mathcal{V}_A^p \subseteq \mathcal{V}_A$ to which the source node has a path that
    solely consists of normal nodes;
4 Identify all the chain structures in $\mathcal{V}_A^p$, and temporarily set the routing
    policies of adversarial nodes in these chains by following line 3 in Alg. 6.1;
5 Apply line 5 to 18 of Alg. 6.2 over the subset of nodes in $\mathcal{V}_A^p$ whose routing
    policies are not temporarily set. If there exists any adversarial node
    downstream whose routing policy is undetermined, recursively call Alg. 6.3
    in the downstream subnetwork;
6 Return $\mathbf{x}_A = \{\mathbf{x}_i\}_{i \in \mathcal{V}_A}$;
```

The first challenge is that $\mathcal{V}_A$ can be a combination of chain and parallel structures. We show an example in Fig. 6-5(a). Alg. 6.3 addresses this challenge by combining the ideas in Alg. 6.1 and 6.2. Alg. 6.3 contains two steps in each recursion. The first step (line 4) is to identify the chain structures among the adversarial nodes whose routing policies have not been decided, and temporarily sets their routing policies following the chain structure. This step follows the idea in Alg. 6.1 where for two adversarial nodes i, j and $i \to j$, we first determine the routing policy at node j

given that node i adopts the routing so that all the traffic at i can arrive at j. In Fig. 6-5(b), the algorithm first assumes that $x_{25} = 1$ and $x_{35} = 1$ since $2 \to 5$ and $3 \to 5$. The second step (line 5) is to determine the routing policies of the adversarial nodes that are at the end of the above chains which are guaranteed to be in a parallel structure. We then adopt the idea in Alg. 6.2 to determine the routing policies of a subset of these parallel adversarial nodes. In Fig. 6-5(b), Alg. 6.3 runs its line 5 over $\{4, 5\}$, and suppose that it determines that node 5 is to route all the traffic to node 7 in a single hop, i.e., $x_{57} = 1$. Alg. 6.3 iteratively follows the above process until the routing policies of all adversarial nodes have been determined, where in Fig. 6-5(c) the second iteration is to consider the routing policies of adversarial nodes $\{2, 3, 4\}$ given that the routing policy at node 5 has been determined. In summary, the main idea of Alg. 6.3 is to synthesize the optimality of chain structures and the good performance guarantee of parallel structures to $\mathcal{V}_A$ in a general structure.

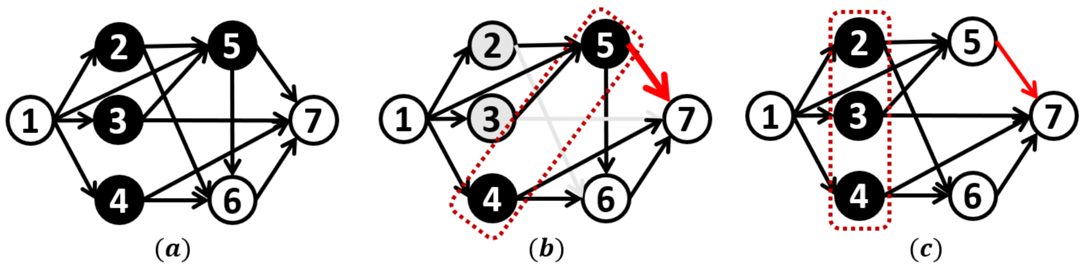


Figure 6-5: Example of Alg. 6.3 on $\mathcal{V}_A$ with a combination of chain and parallel structures. Given that $\mathcal{V}_A = \{2, 3, 4, 5\}$ in (a) where $2 \to 5$ and $3 \to 5$, the algorithm temporarily sets $x_{25} = 1$ and $x_{35} = 1$ (line 5) and determines the routing policies at 4 and 5 (line 6) as shown in (b); suppose that in line 6 the algorithm determines that $x_{57} = 1$ based on $\Delta_{57} = \min\{\{\Delta_{ij}\}_{i \in \mathcal{V}_A, (i,j) \in \mathcal{E}} \cup \{\Delta_k / |\mathcal{R}_k|\}_{k \in \mathcal{V}_N, \mathcal{R}_k \neq \emptyset}\}$, then it starts to consider adversarial nodes upstream to 5 in the chains, i.e., determines the routing policies at $\{2, 3, 4\}$.

The second challenge is that $\mathcal{V}_A$ may be in multiple layers of parallel structures, where we cannot either utilize the optimality of chain structures or directly apply Alg. 6.2 to solve the problem. Consider the example in Fig. 6-3, where the adversary controls two layers of network nodes, and nodes 6, 7 are not reachable from nodes 2, 3. We cannot solely determine the routing policies at a single layer of adversarial nodes ($\{2, 3\}$ or $\{6, 7\}$), since any routing policy at a single layer will result in zero increment of minimum s-d cut value and thus may lead to unbounded performance guarantee. Therefore we introduce recursion in Alg. 6.3, which is a top-down dynamic programming mechanism, to resolve this challenge. We explain the method by the example in Fig. 6-6. In line 3, Alg. 6.3 figures out that $\{2, 3\}$ are the adversarial nodes which are on a path built by normal nodes starting from the source node. It applies the idea in Alg. 6.2 to determine the routing policies of the parallel adversarial nodes $\{2, 3\}$. Consider the calculation of Δ_4, i.e., the increment of the minimum s-d cut value by $x_{24} = 1$ and $x_{34} = 1$. Since $\{7, 8\}$ are adversarial nodes downstream to node 4 whose routing policies are undetermined, Alg. 6.3 will recursively call itself to determine the routing attack of the subnetwork where 4 is viewed as the source node, and the adversarial nodes are $\{7, 8\}$, denoted by the highlighted links in Fig. 6-6. The routing attack solution starting from node 4 can be solved without further recursion since $\{7, 8\}$ are in a parallel structure without more adversarial nodes downstream to them. Calculating Δ_5 can be done similarly. The solutions starting from 4 and 5 are then memorized for further usage in the top-down dynamic programming.

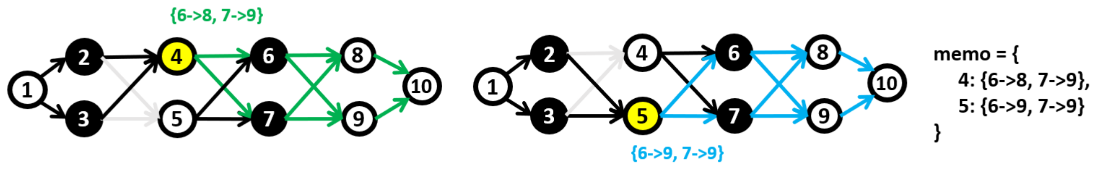


Figure 6-6: The top-down dynamic programming mechanism in Alg. 6.3 on the example in Fig. 6-3.

Remarks: (i) Alg. 6.3 returns the same results and performance guarantees as Alg. 6.1 when $\mathcal{V}_A$ is in a chain structure, and Alg. 6.2 when $\mathcal{V}_A$ is in a parallel structure. (ii) Time complexity: Alg. 6.3 is guaranteed to complete in time polynomial to the network size. The memorization ensures that the routing attack solution of the subnetwork downstream to each node is calculated only once. The calculation start from different nodes can be implemented in parallel. An alternative method is the bottom-up dynamic programming which calculates the routing attack of subnetworks starting from nodes in the order from $N - 1$ to 1. The bottom-up approach has no difference on the worst-case time complexity compared with top-down approach, however it introduces unnecessary computation starting from nodes that will not be used in later iterations. (iii) Performance Guarantee: We find that it is challenging to prove the performance guarantee as done in Section 6.2, when $\mathcal{V}_A$ is a general structure. We observe in the simulation in Section 6.4 that if $\mathcal{V}_A$ is in the form of K chains in parallel[1], the worst-case approximation ratio is at most $O(\log K)$. We leave the proof of this conjecture to future work.

6.4 Performance Evaluation

We evaluate the performance of the proposed algorithms over a wide range of network settings including different topologies, link capacities, and sets of adversarial nodes. We show that the proposed algorithm is superior to other heuristics in terms of approximating the optimal solution to (6.2).

Algorithms for Comparison: We compare two heuristic algorithms with Alg. 6.3. (i) Dynamic Programming (DP): this algorithm traverses the adversarial nodes from the destination to the source, where at each adversarial node the algorithm determines its routing policy to minimize the minimum s-d cut value in the subnetwork downstream to this node. The main differences to Alg. 6.3 are that DP only utilizes the downstream information to determine the routing policy of an adversarial node, and treats each adversarial node separately. (ii) Local Search (LS): this algorithm initializes a routing attack where each adversarial node forwards all the traffic to one of its adjacent nodes, and iteratively check if there exists a better routing attack that can reduce the minimum s-d cut value by adjusting the routing policy of a single adversarial node. It adopts the better routing attack if there exists one, otherwise outputs the current best routing attack.

Network Settings: We test the proposed algorithms over networks with a fixed size of $|\mathcal{V}| = 50$. We first consider $|\mathcal{V}_A| = 6$ for $\mathcal{V}_A$ in a parallel structure, and we can add links between different pairs of adversarial nodes to introduce chains in $\mathcal{V}_A$, where our focus is to validate the analytical results developed in previous sections. We

[1] $K = 1$ is equivalent to the chain structure, $K = |\mathcal{V}_A|$ is equivalent to the parallel structure.

further consider $|\mathcal{V}_A| = 10$ for $\mathcal{V}_A$ in a general structure to showcase the superiority of Alg. 6.3 than DP and LS in terms of approximating the risk of overload in general networks under malicious routing. For any given structure of $\mathcal{V}_A$, we simulate $10,000$ network instances which comprises of 20 different network topologies, 25 different link capacity settings under a given topology, and 20 randomly selected adversarial node sets $\mathcal{V}_A$ that follows the structure, in order to evaluate the algorithms under different network settings.

Evaluation Results: We evaluate the performance of the proposed algorithms in terms of approximating the optimal solution to (6.2). Based on the equivalence between (6.1) and (6.2), an algorithm with a lower approximation ratio to (6.2) indicates higher potential for inducing network overload.

We first evaluate the performance when $\mathcal{V}_A$ is in a parallel structure. Fig. 6-7(a) presents the cumulative distribution functions (CDF) of the approximation ratios of the three algorithms to the optimal solution to (6.2). The statistics of mean, 90-percentile (p90), and maximum approximation ratios are listed in Table 6.1. Results show that Alg. 6.3 leads to a lower mean, p90, and maximum approximation ratio than DP and LS. The maximum approximation ratio of Alg. 6.3 among the tested instances is 1.49, which is below the theoretical upper bound $\sum_{i=1}^{6} i^{-1} = 2.45$. Moreover, the mean approximation ratio 1.04 shows that on average Alg. 6.3 leads to an approximation gap of at most 5% to the optimal routing attack for (6.2), and the p90 approximation ratio 1.11 means that over 90% of the instances Alg. 6.3 leads to a gap of at most 12%. These results demonstrate the high consistency of inducing routing attack with good approximation performance on minimizing the minimum s-d cut values via Alg. 6.3 compared with the other algorithms.

Next we evaluate the performance when there exists chain structures in $\mathcal{V}_A$. We introduce links among the 6 adversarial nodes, denoted by $\{i_1, i_2, i_3, i_4, i_5, i_6\}$, in three ways: (i) introducing (i_1, i_2), (i_3, i_4), (i_5, i_6) so that there are 3 chains $i_1 \rightarrow i_2$, $i_3 \rightarrow i_4$, $i_5 \rightarrow i_6$ in parallel; (ii) introducing (i_1, i_2), (i_2, i_3), (i_4, i_5), (i_5, i_6) so that there are 2 chains $i_1 \rightarrow i_2 \rightarrow i_3$, $i_4 \rightarrow i_5 \rightarrow i_6$ in parallel; (iii) introducing $(i_j, i_{j+1}), j = 1, \ldots, 5$ so that $\mathcal{V}_A$ is in a chain structure. We present the CDFs of the approximation ratios

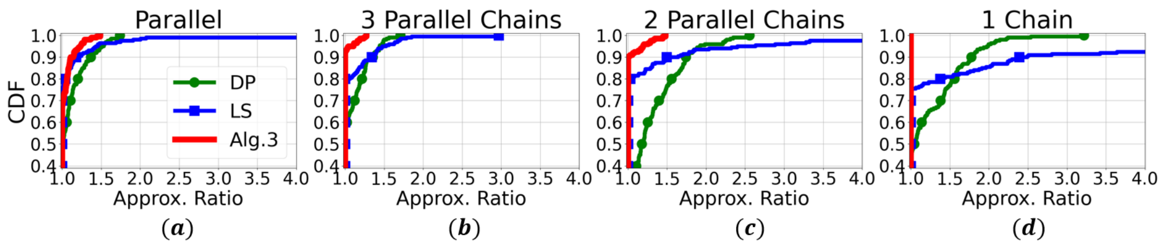


Figure 6-7: CDFs of approximation ratios of Dynamic Programming (DP), Local Search (LS), and Alg. 6.3 to the optimal solution to (6.2).

in Fig. 6-7(b) to 6-7(d) for these three ways respectively, and present their statistics in Table 6.1. We find that Alg. 6.3 outperforms the other two algorithms in both the average and worst-case performance. Moreover, Alg. 6.3 leads to better approximation performance when the number of parallel chains is smaller, and outputs the optimal solution when $\mathcal{V}_A$ is in a chain structure. This observation echoes our conjecture in Section 6.3 that the worst-case approximation ratio is logarithmic to the number of chains in parallel.

	Parallel			3 Parallel Chains			2 Parallel Chains			1 Chain		
	Mean	90%	Max	Mean	90%	Max	Mean	90%	Max	Mean	90%	Max
DP	1.11	1.37	1.74	1.30	1.74	2.56	1.10	1.35	1.71	1.25	1.78	3.22
LS	1.15	1.18	13.35	1.31	1.51	13.17	1.08	1.34	2.97	4.85	2.40	25.14
Alg.6.3	**1.04**	**1.11**	**1.49**	**1.02**	**1.00**	**1.48**	**1.01**	**1.00**	**1.27**	**1.00**	**1.00**	**1.00**

Table 6.1: Approximation ratio statistics of Dynamic Programming (DP), Local Search (LS), and Alg. 6.3 (90% means 90-percentile).

We finally investigate $\mathcal{V}_A$ in a general structure. Fig. 6-8 shows the CDFs of the approximation ratios, where the performance of Alg. 6.3 in approximating the optimal solution to (6.2) is better than the other heuristics: The worst approximation ratio of Alg. 6.3 is 1.26, less than 2.0 for DP and 4.0 for LS. Moreover, Alg. 6.3 outputs the optimal routing attack in around 90% of the 10,000 tested network instances.

Discussion: We learn from the simulation results that our proposed algorithms lead to close-to-optimal performance in terms of maximizing overload in a network with dynamic routing control over a wide range of network topologies, link capacities, and sets of adversarial nodes. The near-optimality of Alg. 6.3 under different network settings demonstrates that it can be used to approximate the highest overload that

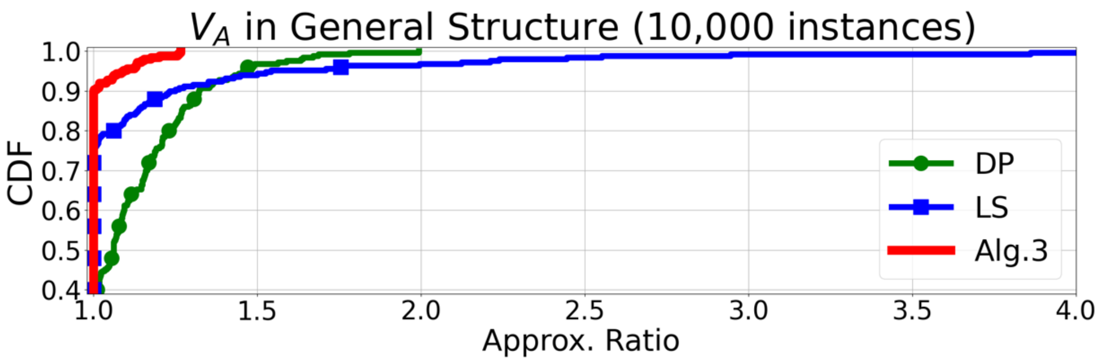


Figure 6-8: CDFs of approximation ratios to the optimal solution to (6.2) when $\mathcal{V}_A$ is in a general structure ($|\mathcal{V}_A| = 10$).

malicious routing can induce given an arbitrary set of adversarial nodes, and identify the critical nodes to protect from adversaries.

6.5 Summary and Future Work

We investigate the ability of malicious routing to induce overload, given that the network can dynamically adjust the routing policies at normal nodes. We prove the equivalence of maximizing the throughput loss and minimizing the minimum s-d cut value of the network, and prove the NP-hardness of the problem. We develop polynomial-time algorithms with performance guarantee when the adversarial nodes are in a chain or a parallel structure. We further extend the proposed algorithms to adversarial nodes that are in a general structure. We validate that the proposed algorithms are near-optimal in most network instances under a wide range of network settings that cover different network topologies, link capacities, and adversarial node sets. Future directions include investigating the cases where normal nodes do not always implement the optimal routing policies in response to the routing attack, and the network adversaries may not have knowledge of the routing patterns at the normal nodes.

6.6 Chapter Appendix

6.6.1 Proof of Proposition 6.1

We prove Proposition 6.1 in two steps. First, we show that there exists an optimal solution $\mathbf{x}_A^*$ to (6.1) where $\forall i \in \mathcal{V}_A$, $\mathbf{x}_i^*$ satisfies that $x_{ij}^* = 1$ for some j and $x_{ik}^* = 0$ for $k \neq j$. Then we show that the problem (6.1) and (6.2) are equivalent if we add the constraint that each adversarial node forwards all the traffic to a single adjacent node. With these two results, together with Theorem 6.1 proved in the following section, we show that the optimal solutions to (6.1) and (6.2) are equivalent.

Step 1: We prove the first result starting from a single adversarial node, and extend to a general set of adversarial nodes. We first prove the following lemma.

Lemma 6.1. *Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with arrival rate λ at node 1 and $\mathcal{V}_A = \{1\}$, then one of the routing attacks such that $\exists j \in \mathcal{V}$, $x_{1j} = 1$, and $\forall k \neq j$, $x_{1k} = 0$ is optimal to* (6.1)*.*

Proof. Consider a specific routing policy $\mathbf{x}_1$ at the adversarial node 1. Denote the set of nodes adjacent to node 1 by $\mathcal{V}_1$. We show that this is a special case where (6.1) is equivalent to the following problem.

$$\begin{aligned} \max_{\mathbf{f}} \quad & \sum_{(i,N)\in\mathcal{E}} f_{iN} \\ \text{s.t.} \quad & \sum_{j:(i,j)\in\mathcal{E}} f_{ij} = \sum_{k:(k,i)\in\mathcal{E}} f_{ki}, \forall i \in \mathcal{V}\backslash\{1\}, \\ & \sum_{j:(i,j)\in\mathcal{E}} f_{ij} = \min\{\lambda x_{1i}, c_{1i}\}, \forall i \in \mathcal{V}_1, \\ & f_{ij} \in [0, c_{ij}],\ \forall (i,j) \in \mathcal{E}. \end{aligned} \tag{6.3}$$

The equivalence stems from the fact that (6.3) aims to optimize the routing policies at normal nodes to maximize the throughput to the destination given that the adversarial node 1 sets its routing policy. Instead of allowing loss at any node, (6.3) requires that there is no loss at nodes downstream to node 1 and thus the traffic that will cause loss at these nodes will be dropped at the source. This means maximizing the throughput is equivalent to minimizing the loss.

We then investigate the dual problem of (6.3) below.

$$\begin{aligned}
\max_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}} \quad & \sum_{i\in\mathcal{V}_1} \beta_i \min\{\lambda x_{1i}, c_{1i}\} + \sum_{(i,j)\in\mathcal{E}} c_{ij}\gamma_{ij} \\
\text{s.t.} \quad & \gamma_{ij} + \beta_i - \alpha_j \le -\mathbf{1}_{j=N},\ \forall i \in \mathcal{V}_1,\ (i,j)\in\mathcal{E}, \\
& \gamma_{ij} + \alpha_i - \alpha_j \le -\mathbf{1}_{j=N},\ \forall i \notin \mathcal{V}_1 \cup \{1\}, (i,j)\in\mathcal{E}, \\
& \gamma_{ij} \le 0,\ \forall (i,j)\in\mathcal{E}.
\end{aligned} \tag{6.4}$$

We point out that the optimal solution to (6.4) guarantees $\beta_i^* \ge 0$, $\forall i \in \mathcal{V}_1$. Therefore, the optimal objective value to (6.4) is a concave function with respect to $\mathbf{x}_1$, and thus one of the optimal routing policies $\mathbf{x}_1^*$ that minimizes the optimal objective value of (6.4) is at a vertex of the set of feasible routing policies of node 1, i.e., $x_{1j}^* = 1$ for some j. Meanwhile, (6.3) is a linear programming, and thus the optimal objective values to (6.3) and (6.4) are the same. Therefore the optimal routing policy at node 1 to (6.3) satisfies that $\exists j \in \mathcal{V}$, $x_{1j} = 1$, and $\forall k \ne j$, $x_{1k} = 0$ is optimal to (6.1). $\square$

Based on Lemma 6.1, we can generalize the result from $\mathcal{V}_A = \{1\}$ to $\mathcal{V}_A = \{i\}$ for some i by considering the subnetwork downstream to node i, as the result holds for arbitrary incoming traffic of node i, represented by λ in (6.3). We further extend the results to general $\mathcal{V}_A$: Given an arbitrary routing attack over $\mathcal{V}_A$, suppose that $\exists i \in \mathcal{V}_A$ whose routing policy $\mathbf{x}_i$ is not a 0-1 vector, then fixing the routing policy of the other adversarial nodes, there must exist j so that changing the routing policy of node i such that $x_{ij} = 1$ is not worse than the current routing attack. This fact demonstrates that there always exists a routing attack $\mathbf{x}_A$ that optimizes (6.1) where at each adversarial node $i \in \mathcal{V}_A$, $x_{ij}^* = 1$ for some j and $x_{ik}^* = 0$ for $k \ne j$.

Step 2: We prove that given the constraints that the routing attack $\mathbf{x}_A$ must be restricted to the boundary of $\mathcal{X}_A$, the optimal solutions to (6.1) and (6.2) are the same. With this constraint, each adversarial node can only forward the traffic to a single adjacent node. As discussed in Section 6.1.3, the optimal routing attack $\mathbf{x}_A^*$ to (6.2) for $\mathcal{G}$ is the one that minimizes the minimum s-d cut of the network $\mathcal{G}'$. There are only two network states given any routing attack restricted to the boundary of $\mathcal{X}_A$: When λ is not greater than the minimum s-d cut of $\mathcal{G}'$, the loss is 0, otherwise the excess traffic will be lost. We further show that the optimal solutions of (6.1) are the same to (6.2): Since (6.2) outputs the routing attack that minimizes the minimum s-d cut value, then if λ is no greater than this value, there is no chance to induce any loss, otherwise the loss increases with the same rate as the arrival rate λ, where the growth of loss is maximized compared with the cases under other routing attacks, and thus the maximum loss is achieved for any specific given λ under the optimal solutions of (6.2).

6.6.2 Proof of Theorem 6.1

For (6.2), we consider an arbitrarily given $\mathbf{x}_A \in \mathcal{X}_A$, and investigate the dual formulation of the problem with the same constraints as (6.2) and the objective is $\max_{\mathbf{f}} \sum_{i:(i,N)\in\mathcal{E}} f_{iN}$. Define the problem as $\mathbf{P}$. The dual problem of $\mathbf{P}$ is

$$\begin{aligned}
\min_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}} \quad & \sum_{(i,j)\in\mathcal{E}} c_{ij}\alpha_{ij} \\
\text{s.t.} \quad & \alpha_{ij} - \gamma_i + \gamma_j \geq \mathbf{1}_{j=N}, \forall i \notin \mathcal{V}_A, j \notin \mathcal{V}_A, (i,j)\in\mathcal{E}, \\
& \alpha_{ij} - \beta_{ij} + \gamma_j \geq \mathbf{1}_{j=N}, \forall i \in \mathcal{V}_A, j \notin \mathcal{V}_A, (i,j)\in\mathcal{E}, \\
& \alpha_{ij} - \gamma_i + \sum_{l:(i,l)\in\mathcal{E}} \beta_{jl}x_{jl} \geq 0, \forall i \notin \mathcal{V}_A, j \in \mathcal{V}_A, (i,j)\in\mathcal{E}, \\
& \alpha_{ij} - \beta_{ij} + \sum_{l:(i,l)\in\mathcal{E}} \beta_{jl}x_{jl} \geq 0, \forall i \in \mathcal{V}_A, j \in \mathcal{V}_A, (i,j)\in\mathcal{E}, \\
& \alpha_{ij} \geq 0, \ \forall (i,j)\in\mathcal{E}
\end{aligned} \tag{6.5}$$

where the decision variables $\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}$ correspond to the constraints $f_{ij} \leq c_{ij}$, $f_{ij} = \left(\sum_{k:(k,i)\in\mathcal{E}} f_{ki}\right) x_{ij}$, and $\sum_{j:(i,j)\in\mathcal{E}} f_{ij} = \sum_{k:(k,i)\in\mathcal{E}} f_{ki}$ in (6.2). We can transform the objective function by replacing the variables $\boldsymbol{\alpha}$, which becomes

$$\begin{aligned} &\sum_{i\notin\mathcal{V}_A, j\notin\mathcal{V}_A,(i,j)\in\mathcal{E}} c_{ij}\max\{\gamma_i-\gamma_j+\mathbf{1}_{j=N},0\} \\ &+\sum_{i\in\mathcal{V}_A, j\notin\mathcal{V}_A,(i,j)\in\mathcal{E}} c_{ij}\max\{\beta_{ij}-\gamma_j+\mathbf{1}_{j=N},0\} \\ &+\sum_{i\notin\mathcal{V}_A, j\in\mathcal{V}_A,(i,j)\in\mathcal{E}} c_{ij}\max\{\gamma_i-\sum_{l:(i,l)\in\mathcal{E}}\beta_{jl}x_{jl},0\} \\ &+\sum_{i\in\mathcal{V}_A, j\in\mathcal{V}_A,(i,j)\in\mathcal{E}} c_{ij}\max\{\beta_{ij}-\sum_{l:(i,l)\in\mathcal{E}}\beta_{jl}x_{jl},0\}. \end{aligned} \tag{6.6}$$

Notice that the problem $\mathbf{P}$ is a linear programming problem, therefore the optimal values of the primal and dual functions are the same, and thus our goal is equivalent to find a routing attack to minimize the dual problem. Consider any given $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, the objective (6.6) is a function of the routing attack $\mathbf{x}_A$ which only exists in the last two terms of (6.6), and therefore for $\forall j\in\mathcal{V}_A$, setting $x_{jl'}=1$ where $l'=\arg\min_{l:(j,l)\in\mathcal{E}}\beta_{jl}$ can minimize (6.6). Due to the arbitrariness of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ in the above analysis, we have that there must exist a routing attack $\mathbf{x}_A^*\in\mathcal{X}_A$ that can minimize the primal problem, which satisfies that for $\forall i\in\mathcal{V}_A$, $\exists j\in\mathcal{V}_i$ so that $x_{ij}^*=1$.

6.6.3 Proof of Theorem 6.2

We reduce the Set Cover problem to problem (6.2). Consider an instance of Set Cover with N elements, denoted by $\{s_1,\ldots,s_N\}$, and M collections $\{d_1,\ldots,d_M\}$. We introduce a link from s_i to d_j with infinite capacity if the collection d_j covers s_i. We introduce a meta source node s_0 and a meta destination node d_0, and links (s_0,s_i) with infinite capacity for $\forall i=1,\ldots,N$, and links (d_j,d_0) with capacity 1 for $\forall j=1,\ldots,M$. Suppose that $\mathcal{V}_A=\{s_i\}_{i=1}^N$ and $\mathcal{V}_N=\{s_0,d_0\}\cup\{d_j\}_{j=1}^M$. Fig. 6-9 presents an example of the construction based on an instance of Set Cover.

Suppose that there is a polynomial-time algorithm that can solve (6.2). We apply it over the constructed graph from the Set Cover instance, and denote the optimal solution by $\mathbf{x}_p^*$ at each $p\in\mathcal{V}_A$. Denote the number of nodes in $\{d_j\}_{j=1}^M$ that there exists p such that $x_{pd_j}>0$ by L. We can show that by contradiction L is equal to the minimum number of collections that can cover $\{s_1, \dots, s_N\}$, denoted by L'. First, it is obvious that $L \geq L'$ since each adversarial node needs to forward the traffic at it to at least one of the nodes in $\{d_j\}_{j=1}^M$. Second, $L \leq L'$, because L is the maximum traffic arrival rate at the meta source node s_0 that does not cause any loss under the optimal routing attack while L' is only one of such maximum arrival rates without inducing loss that is feasible via routing all the traffic at s_i to d_j that is assigned to cover d_j in the minimum set cover solution. Therefore $L = L'$. This means that the Set Cover can be solved in polynomial time if (6.2) can be solved in polynomial time, which contradicts that the Set Cover is NP-hard.

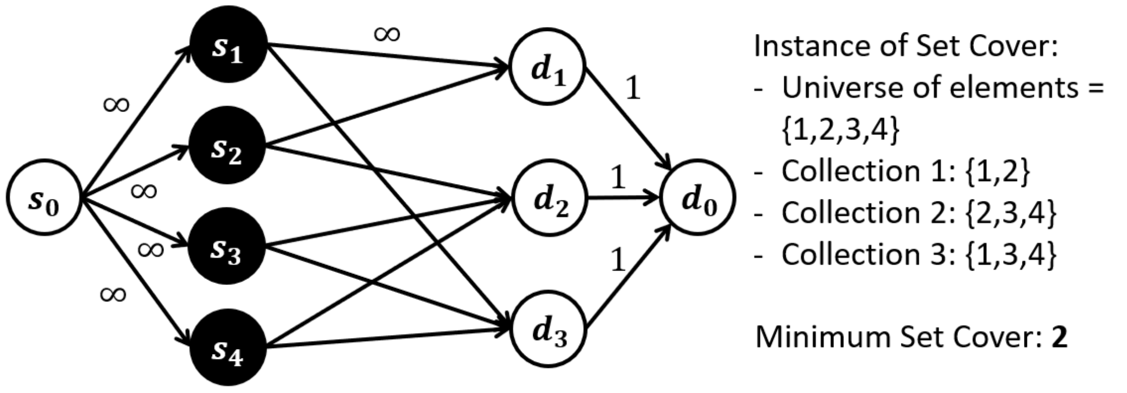


Figure 6-9: Example of a constructed network based on an instance of Set Cover.

6.6.4 Proof of Theorem 6.3

Consider an optimal solution $\mathbf{x}_A^*$ to the problem (6.2) where $\mathbf{x}_i^*$ being a 0-1 vector for $\forall i \in \mathcal{V}_A$. As explained in Section 6.1.3, the network $\mathcal{G}$ under such $\mathbf{x}_A^*$ can be transformed to another network $\mathcal{G}'$ where $\forall i \in \mathcal{V}_A$ there is only one link (i, j_i) with capacity $c'_{ij_i} := c_{ij_i}$ where $x_{ij_i} = 1$, and $c'_{ik} = 0$, $\forall k \neq j_i$. Therefore the minimum s-d cut of $\mathcal{G}'$ is equal to the maximum throughput without inducing loss.

We consider the minimum s-d cut $(\mathcal{S}, \mathcal{V} \backslash \mathcal{S})$ of $\mathcal{G}'$ under the optimal solution $\mathbf{x}_A^*$. The set of links that build the cut is

$$\mathcal{E}_c := \{(i,j) \in \mathcal{E} \mid i \in \mathcal{S}, j \in \mathcal{V} \backslash \mathcal{S}\}.$$

We classify the adversarial nodes into three categories:

- $\mathcal{V}_A^1 := \{i \in \mathcal{V}_A \mid c'_{ij} = 0, \forall j \in \mathcal{V} \text{ and } (i,j) \in \mathcal{E}_c\}$;
- $\mathcal{V}_A^2 := \{i \in \mathcal{V}_A \mid c'_{ij} = c_{ij}, \exists j \in \mathcal{V} \text{ and } (i,j) \in \mathcal{E}_c\}$;
- $\mathcal{V}_A^3 := \{i \in \mathcal{V}_A \mid \nexists j \in \mathcal{V} \text{ s.t. } (i,j) \in \mathcal{E}_c\}$.

Note that $\mathcal{V}_A = \mathcal{V}_A^1 \cup \mathcal{V}_A^2 \cup \mathcal{V}_A^3$, and adversarial nodes in $\mathcal{V}_A^3$ are not in $\mathcal{S}$. Therefore taking any routing policy at nodes in $\mathcal{V}_A^3$ does not affect the result if the routing attack in $\mathcal{V}_A^1 \cup \mathcal{V}_A^2$ is optimal.

We first consider the base case: two adversarial nodes i_1, i_2 where $i_1 \to i_2$. We show the optimality of Alg. 6.1 given that i_1 and i_2 are in each of the following four cases under the minimum cut $\mathcal{S}$: (i) If $i_1, i_2 \in \mathcal{V}_A^1$, then the cut $\mathcal{S}$ is the minimum cut at the initial state in Alg. 6.1 where $\mathbf{x}_{i_1} = \mathbf{0}$ and $\mathbf{x}_{i_2} = \mathbf{0}$, and Alg. 6.1 will decide the routing policy at i_2 so that $x_{i_2 j} = 1$ and $(i_2, j) \notin \mathcal{E}_c$ for some j in the first iteration, and similar for i_1 in the second iteration, since the increment of minimum s-d cut value is zero. (ii) If $i_1 \in \mathcal{V}_A^1$ and $i_2 \in \mathcal{V}_A^2$, then since $i_1 \to i_2$, temporarily setting i_1 to route all the traffic to i_2 does not affect the decision at i_2 in the first iteration, and i_2 will route the traffic to minimize the increment of minimum s-d cut value as the optimal solution does. (iii) If $i_1 \in \mathcal{V}_A^2$ and $i_2 \in \mathcal{V}_A^1$, then i_1 must route the traffic to a node that bypasses i_2 in the optimal routing attack (otherwise this case is impossible). Therefore, the minimum s-d cut is the same as $\mathcal{S}$ at the initial state, and Alg. 6.1 will route all the traffic at i_1 to a node j that causes zero increment of minimum s-d cut value, and route i_2 as the optimal solution does to achieve minimum increment of the min-cut value. (iv) If $i_1 \in \mathcal{V}_A^2$ and $i_2 \in \mathcal{V}_A^2$, then in the optimal routing attack i_1 will not route to i_2, therefore as discussed in the third case, routing i_1 to i_2 does not affect the optimality, and in each iteration Alg. 6.1 routes the traffic at an adversarial node so that the minimum s-d cut is increased with minimum amount (in this case the increment must be positive). Based on the case of two adversarial nodes, we can generalize to larger size of $\mathcal{V}_A$ by the mathematical reduction, following the similar process for i_k and i_{k+1} in the chain $\{i_1, \cdots, i_{|\mathcal{V}_A|}\}$ in the $|\mathcal{V}_A| - k$-th iteration. In summary, the idea is to show that at the k-th iteration we do not need to modify the routing policies determined in previous $k-1$ iterations to guarantee the optimality.

6.6.5 Proof of Theorem 6.4

Recall that Alg. 6.2 selects the minimum value among $\{\Delta_{ij}\}_{i\in\mathcal{V}_A,(i,j)\in\mathcal{E}}$ and $\{\Delta_k/|\mathcal{R}_k|\}_{k\in\mathcal{V}_N,\mathcal{R}_k\neq\emptyset}$ and determines the routing policies of a subset of adversarial nodes in each iteration. Notice that we only need to develop the proof considering that Alg. 6.2 selects among $\{\Delta_k/|\mathcal{R}_k|\}_{k\in\mathcal{V}_N,\mathcal{R}_k\neq\emptyset}$ in each iteration, since $\{\Delta_{ij}\}_{i\in\mathcal{V}_A,(i,j)\in\mathcal{E}}$ can be equivalently transformed to this form: for each $(i,j)\in\mathcal{E}$ where $i\in\mathcal{V}_A$ when $\mathcal{V}_A$ is in a parallel structure, we can introduce an intermediate node k and replace the link (i,j) by two links (i,v) with $c_{iv}=\infty$ and (v,j) with c_{vj} equal to the original capacity of the original link (i,j), and then the original evaluation over Δ_{ij} is equivalently replaced by evaluating Δ_v in the transformed network. Therefore we solely develop the proof below selecting over $\{\Delta_k/|\mathcal{R}_k|\}_{k\in\mathcal{V}_N,\mathcal{R}_k\neq\emptyset}$ in each iteration.

We first prove the following Lemma 6.2, which states in every iteration, there exists a choice $k\in\mathcal{V}_N$ and $\mathcal{R}_k\neq\emptyset$ such that the corresponding routing policies over $\mathcal{R}_k$ determined by Alg. 6.2 leads to a bounded increment of minimum s-d cut value. We give relevant definitions used below: Δ^* denotes the minimum increment of the minimum s-d cut value under the optimal routing attack; $\mathcal{J}_k$ denotes the set of adversarial nodes whose routing policies have not been determined in the first k iterations; $\Delta_{i,k}$ denotes the increment of minimum s-d cut value when targeting $i\in\mathcal{V}_N$ and routing the traffic of all adversarial nodes in $\mathcal{R}_i\cap\mathcal{J}_{k-1}$ in the k-th iteration of Alg. 6.2.

Lemma 6.2. *Given a input instance $(\mathcal{G},\mathcal{V}_A)$ to Alg. 6.2, then at the k-th iteration, there exists a node i such that $\Delta_{i,k}/|\mathcal{R}_i\cap\mathcal{J}_{k-1}|\leq\Delta^*/|\mathcal{J}_{k-1}|$.*

Proof. We first show that the optimal routing attack can be decomposed into multiple iterations such that the k-th iteration targets at a normal node i and determines the routing policies of adversarial nodes in $\mathcal{R}_i\cap\mathcal{J}^*_{k-1}$, where $\mathcal{J}^*_{k-1}$ denotes the set of adversarial nodes whose routing policies have not been determined, and in the meantime the following condition **COND** holds: the increment of the minimum s-d cut value when targeting at node i at this iteration is equal to the increment when targeting at node i at the first iteration over $\mathcal{R}_i \backslash |\mathcal{J}_{k-1}^*|$. Recall that we define $\mathcal{S}$ as the minimum s-d cut under the optimal routing attack. In the transformed network with intermediate nodes introduced in each link starting from an adversarial node, the link set $\mathcal{E}_c$ in $\mathcal{S}$ only contains normal nodes. It is straightforward to verify that an arbitrary order of traversal of the nodes that start from some link in $\mathcal{E}_c$ guarantees that **COND** holds. Denote the set of these normal nodes by $\mathcal{V}_N^*$, an order of traversal over $\mathcal{V}_N^*$ by $\{i_1, i_2, \ldots, i_{|\mathcal{V}_N^*|}\}$. Note that $\Delta_{i,k} \geq 0$. Then we have $\Delta^* = \sum_k \Delta_{i_k,k}$. Fig. 6-10 shows an example.

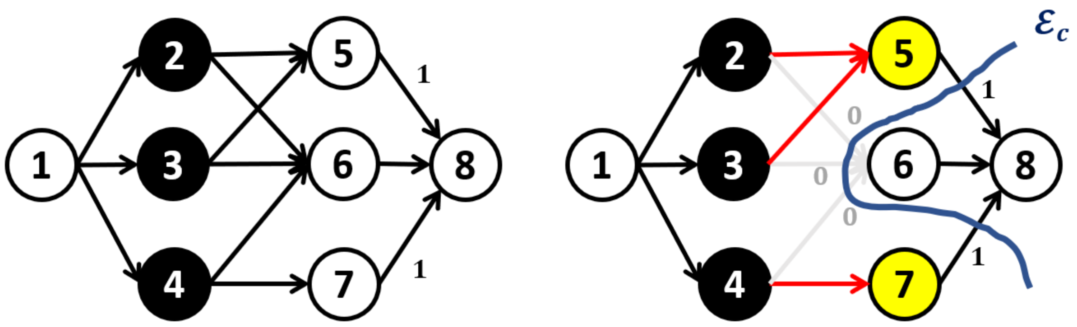


Figure 6-10: Example where COND holds: the optimal routing attack can be decomposed into 2 iterations, where in the first iteration it determines the routing policies of $\mathcal{R}_7$, and in the second iteration it determines those of $\mathcal{R}_5$ whose routing policies have not been determined in previous iterations.

We then prove the lemma by contradiction in the transformed network. Suppose that at the k-th iteration of Alg. 6.2, $\Delta_{i,k}/|\mathcal{R}_i \cap \mathcal{J}_{k-1}| > \Delta^*/|\mathcal{J}_{k-1}|$ holds for $\forall i \in \mathcal{V}_N, \mathcal{R}_i \cap \mathcal{J}_{k-1}$. Then we have

$$\begin{aligned}\Delta^* &= \sum_k \Delta_{i_k,k} = \sum_k \frac{\Delta_{i_k,k}}{|\mathcal{R}_{i_k} \cap \mathcal{J}_{t-1}|}|\mathcal{R}_{i_k} \cap \mathcal{J}_{t-1}| \\ &> \sum_k \frac{\Delta^*}{|\mathcal{J}_{k-1}|}|\mathcal{R}_{i_k} \cap \mathcal{J}_{k-1}| \geq \frac{\Delta^*}{|\mathcal{J}_{k-1}|}|\mathcal{J}_{k-1}| = \Delta^*\end{aligned}$$

which leads to contradiction $\Delta^* > \Delta^*$. □

Based on Lemma 6.2, we can bound the total increment of minimum s-d cut value Δ induced by the output routing attack of Alg. 6.2. Suppose that Alg. 6.2 has $T \leq |\mathcal{V}_A|$ iterations over a problem instance. Then we have

$$\begin{aligned}\Delta &\leq \sum_{k=1}^{T} \Delta_{i_k,k} \overset{(a)}{=} \sum_{k=1}^{T} \frac{\Delta_{i_k,k}}{|\mathcal{R}_{i_k} \cap \mathcal{J}_{k-1}|} |\mathcal{R}_{i_k} \cap \mathcal{J}_{k-1}| \\ &\overset{(b)}{\leq} \Delta^* \sum_{k=1}^{T} \frac{|\mathcal{R}_{i_k} \cap \mathcal{J}_{k-1}|}{|\mathcal{J}_{k-1}|} = \Delta^* \sum_{k=1}^{T} \frac{|\mathcal{R}_{i_k} \cap \mathcal{J}_{k-1}|}{|\mathcal{V}_A| - \sum_{l=1}^{k-1} |\mathcal{R}_{i_k} \cap \mathcal{J}_l|} \\ &\overset{(c)}{\leq} \Delta^* \sum_{k=1}^{|\mathcal{V}_A|} k^{-1} = \Delta^* \times \log |\mathcal{V}_A|\end{aligned}$$

where (a) holds in the transformed network, and (b) holds since we find the normal node i with minimum $\Delta_{i,k}/|\mathcal{R}_{i_k} \cap \mathcal{J}_{k-1}|$, and (c) can be verified easily.

6.6.6 Proof of Theorem 6.5

We prove by contradiction. Suppose there exists an algorithm with worst-case approximation ratio lower than $O(\log |\mathcal{V}_A|)$, then applying this algorithm to the networks that can be viewed as instances of the Set Cover problem (for example Fig. 6-9) guarantees that the approximation ratio is lower than $O(\log |\mathcal{V}_A|)$ where $|\mathcal{V}_A|$ in the Set Cover instance is the number of elements. This contradicts to the well-known inapproximability result of the Set Cover problem [121], which states that there is no polynomial-time algorithm with a worst-case approximation ratio lower than $O(\log |\mathcal{V}_A|)$.

Chapter 7

Concluding Remarks

In this thesis, we build upon the previous research on network overload and obtain novel results that deepen the understanding in this research topic, in terms of (i) proposing optimal network policies to minimize queueing delay, balance the queue overload over network nodes, and guarantee queue stability so that network overload is avoided, and (ii) quantifying the capability of network adversaries inducing network overload via routing attacks given static or dynamic routing policies. Our work extends the models and methods to more diverse scenarios in real-world networks including multi-stage structures for delay minimization, bounded node buffers when investigating overload balancing and network stabilization, and the impact of routing attacks on inducing network overload.

In Chapter 2, we study link rate control for queueing delay minimization in overloaded networks. Leveraging the fluid queueing model, we show that any static rate-proportional policy, which guarantees identical ratios between the ingress and egress rates of all the nodes at each layer, minimizes the average delay $\bar{D}_{\text{avg}}$ and the maximum ingress delay $\bar{D}_{\text{max}}$ in general single-hop and multi-stage networks. We further extend the result to the queue-proportional policies which can achieve asymptotically minimum delay based on real-time queue information agnostic of packet arrival rates. We evaluate the performance of our proposed policies under different network settings, validate their min-delay property, and demonstrate their superiority in delay reduction compared with the backpressure policy and the max-link-rate policy. We finally discuss the extensions of the main results in practice and in theory.

In Chapter 3, we study overload balancing in single-hop networks with bounded buffers. We show that bounded buffer affects the resulting policy to achieve most balanced overload. We leverage ordinary differential equations to model the queue dynamics in bounded buffer systems. We first prove that setting link service rates to minimize the quadratic sum of the queue overload rates leads to the lexicographic minimum queue overload. Based on this result, we prove that a maxweight scheduling and backpressure policy asymptotically achieves most balanced overload, through a novel formulation of the policy in a differentiable form which may be of independent interest. We further propose a distributed maxweight + backpressure policy that can reduce communication overhead by one order of magnitude. We validate the performance of our proposed policies by simulation over single-hop structure and Clos networks under different packet arrival rates, link capacities, and buffer settings.

In Chapter 4, we leverage the ODE fluid-queue model to capture the dynamics of buffered communication systems to study network stability. For single-commodity systems, we propose a sufficient condition for a local policy to stabilize the network. The result characterizes a set of policies, and captures systems with arbitrary buffers. For such policies the network stability problem is reduced to an problem testing the existence of an equilibrium point for the ODE system. For multi-commodity systems, we extend the condition by incorporating an additional condition on the coupling level between different commodities, and explain the existence of an equilibrium point in different buffer settings. We finally extend the results in multi-commodity networks to a more explicit rule of thumb of policy design that facilitates network stability in real-world networks.

In Chapter 5, we quantify the threat of routing attacks on causing network overload. We investigate the optimal routing attacks for no-loss throughput minimization and loss maximization. We demonstrate that the no-loss throughput can be minimized in polynomial time in general multi-hop networks. We further develop a 2-approximation algorithm by only leveraging the downstream information of the adversarial nodes. We establish that loss maximization is NP-complete and propose two approximation algorithms with guaranteed performance in single-hop networks. Moreover, we address the adversary's optimal selection of nodes to conduct routing attacks and propose heuristic algorithms for this NP-complete problem. Our performance evaluation showcases the near-optimal performance of the proposed algorithms across a wide range of network settings.

In Chapter 6, we investigate the ability of malicious routing to induce overload, given that the network can dynamically adjust the routing policies at normal nodes. We prove the equivalence of maximizing the throughput loss and minimizing the minimum s-d cut value of the network, and prove the NP-hardness of the problem. We develop polynomial-time algorithms with performance guarantee when the adversarial nodes are in a chain or a parallel structure. We further extend the proposed algorithms to adversarial nodes that are in a general structure. We validate that the proposed algorithms are near-optimal in most network instances under a wide range of network settings that cover different network topologies, link capacities, and adversarial node sets. Future directions include investigating the cases where normal nodes do not always implement the optimal routing policies in response to the routing attack, and the network adversaries may not have knowledge of the routing patterns at the normal nodes.

We list multiple future directions that may further broaden and deepen the understanding of network overload in real-world network infrastructures. In terms of queueing delay minimization, we envision promising directions including proving sufficient and necessary conditions on rate control in general multi-stage networks, extending the min-delay policies to multi-hop networks, and implementing the proposed min-delay policies in real data center networks. In terms of overload balancing, one typical future direction is to extend the algorithms and theoretical analysis to general multi-hop networks with arbitrary settings of node buffers. In terms of network stability, the proof of necessary conditions of network policies to stabilize the networks with bounded node buffers is an unsolved question. In terms of cyberattacks on network overload, future directions include deriving the performance guarantee of loss maximization in multi-hop networks given the static routing policies, and quantifying the risk of routing attacks and other cyberattacks on inducing network overload when the adversaries do not have access to the complete knowledge of routing policies in the networks, for example inferring the routing policies of network nodes based on their historical behaviors.

Bibliography

[1] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, et al. Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network. *ACM SIGCOMM computer communication review*, 45(4):183–197, 2015.

[2] Long Bao Le, Eytan Modiano, and Ness B Shroff. Optimal control of wireless networks with finite buffers. In *2010 Proceedings IEEE INFOCOM*, pages 1–9. IEEE, 2010.

[3] Leonidas Georgiadis and Leandros Tassiulas. Optimal overload response in sensor networks. *IEEE Transactions on Information Theory*, 52(6):2684–2696, 2006.

[4] Devavrat Shah and Damon Wischik. Fluid models of congestion collapse in overloaded switched networks. *Queueing Systems*, 69(2):121, 2011.

[5] Ohad Perry and Ward Whitt. Chattering and congestion collapse in an overload switching control. *Stochastic Systems*, 6(1):132–210, 2016.

[6] VJ Venkataramanan and Xiaojun Lin. On the queue-overflow probability of wireless systems: A new approach combining large deviations with lyapunov functions. *IEEE transactions on information theory*, 59(10):6367–6392, 2013.

[7] Yiwen Zhang, Gautam Kumar, Nandita Dukkipati, Xian Wu, Priyaranjan Jha, Mosharaf Chowdhury, and Amin Vahdat. Aequitas: admission control for performance-critical rpcs in datacenters. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 1–18, 2022.

[8] Carri W Chan, Mor Armony, and Nicholas Bambos. Fairness in overloaded parallel queues. *arXiv preprint arXiv:1011.1237*, 2010.

[9] openai. Ai and compute, 2018.

[10] Anny Xijia Zheng, Jianan Zhang, Rui Wang, and Leon Poutievski. How traffic analytics shapes traffic engineering, topology engineering, and capacity planning of jupiter. In *2023 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3. IEEE, 2023.

[11] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15, 2021.

[12] Hitesh Ballani, Paolo Costa, Istvan Haller, Krzysztof Jozwik, Kai Shi, Benn Thomsen, and Hugh Williams. Bridging the last mile for optical switching in data centers. In *Optical Fiber Communication Conference*, pages W1C–3. Optica Publishing Group, 2018.

[13] Qizhe Cai, Shubham Chaudhary, Midhul Vuppalapati, Jaehyun Hwang, and Rachit Agarwal. Understanding host network stack overheads. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 65–77, 2021.

[14] Leon Poutievski, Omid Mashayekhi, Joon Ong, Arjun Singh, Mukarram Tariq, Rui Wang, Jianan Zhang, Virginia Beauregard, Patrick Conner, Steve Gribble, et al. Jupiter evolving: Transforming google's datacenter network via optical circuit switches and software-defined networking. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 66–85, 2022.

[15] William M Mellette, Rob McGuinness, Arjun Roy, Alex Forencich, George Papen, Alex C Snoeren, and George Porter. Rotornet: A scalable, low-complexity, optical datacenter network. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 267–280, 2017.

[16] Hitesh Ballani, Paolo Costa, Raphael Behrendt, Daniel Cletheroe, Istvan Haller, Krzysztof Jozwik, Fotini Karinou, Sophie Lange, Kai Shi, Benn Thomsen, et al. Sirius: A flat datacenter network with nanosecond optical switching. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 782–797, 2020.

[17] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. *ACM SIGCOMM computer communication review*, 38(4):63–74, 2008.

[18] Xinzhe Fu and Eytan Modiano. Fundamental limits of volume-based network dos attacks. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(3):1–36, 2019.

[19] Xinzhe Fu and Eytan Modiano. Network interdiction using adversarial traffic flows. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1765–1773. IEEE, 2019.

[20] Pavlos Sermpezis, Vasileios Kotronis, Alberto Dainotti, and Xenofontas Dimitropoulos. A survey among network operators on bgp prefix hijacking. *ACM SIGCOMM Computer Communication Review*, 48(1):64–69, 2018.

[21] Bahaa Al-Musawi, Philip Branch, and Grenville Armitage. Bgp anomaly detection techniques: A survey. *IEEE Communications Surveys & Tutorials*, 19(1):377–396, 2016.

[22] Leandros Tassiulas and Anthony Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. In *29th IEEE Conference on Decision and Control*, pages 2130–2132. IEEE, 1990.

[23] Chih-ping Li and Eytan Modiano. Receiver-based flow control for networks in overload. *IEEE/ACM Transactions on Networking*, 23(2):616–630, 2014.

[24] Michael J Neely, Eytan Modiano, and Chih-Ping Li. Fairness and optimal stochastic control for heterogeneous networks. *IEEE/ACM Transactions On Networking*, 16(2):396–409, 2008.

[25] Longbo Huang, Scott Moeller, Michael J Neely, and Bhaskar Krishnamachari. Lifo-backpressure achieves near-optimal utility-delay tradeoff. *IEEE/ACM Transactions On Networking*, 21(3):831–844, 2012.

[26] Hao Yu and Michael J Neely. A new backpressure algorithm for joint rate control and routing with vanishing utility optimality gaps and finite queue lengths. *IEEE/ACM Transactions on Networking*, 26(4):1605–1618, 2018.

[27] Ying Cui, Edmund M Yeh, and Ran Liu. Enhancing the delay performance of dynamic backpressure algorithms. *IEEE/ACM Transactions on Networking*, 24(2):954–967, 2015.

[28] Majed Alresaini, Kwame-Lante Wright, Bhaskar Krishnamachari, and Michael J Neely. Backpressure delay enhancement for encounter-based mobile networks while sustaining throughput optimality. *IEEE/ACM Transactions on Networking*, 24(2):1196–1208, 2015.

[29] Chih-ping Li, Georgios S Paschos, Leandros Tassiulas, and Eytan Modiano. Dynamic overload balancing in server farms. In *2014 IFIP Networking Conference*, pages 1–9. IEEE, 2014.

[30] Chenhao Qu, Rodrigo Neves Calheiros, and Rajkumar Buyya. Mitigating impact of short-term overload on multi-cloud web applications through geographical load balancing. *concurrency and computation: practice and experience*, 29(12):e4126, 2017.

[31] Inho Cho, Ahmed Saeed, Joshua Fried, Seo Jin Park, Mohammad Alizadeh, and Adam Belay. Overload control for μs-scale rpcs with breakwater. In

Proceedings of the 14th USENIX Conference on Operating Systems Design and Implementation, pages 299–314, 2020.

[32] Vamsi Addanki, Maria Apostolaki, Manya Ghobadi, Stefan Schmid, and Laurent Vanbever. Abm: Active buffer management in datacenters. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 36–52, 2022.

[33] Giacomo Como, Ketan Savla, Daron Acemoglu, Munther A Dahleh, and Emilio Frazzoli. Robust distributed routing in dynamical networks—part i: Locally responsive policies and weak resilience. *IEEE Transactions on Automatic Control*, 58(2):317–332, 2012.

[34] Liang Tian, Amir Bashan, Da-Ning Shi, and Yang-Yu Liu. Articulation points in complex networks. *Nature communications*, 8(1):1–9, 2017.

[35] Johann Schlamp, Ralph Holz, Quentin Jacquemart, Georg Carle, and Ernst W Biersack. Heap: reliable assessment of bgp hijacking attacks. *IEEE Journal on Selected Areas in Communications*, 34(6):1849–1861, 2016.

[36] Shinyoung Cho, Romain Fontugne, Kenjiro Cho, Alberto Dainotti, and Phillipa Gill. Bgp hijacking classification. In *2019 Network Traffic Measurement and Analysis Conference (TMA)*, pages 25–32. IEEE, 2019.

[37] Seyyit Alper Sert, Adnan Yazıcı, and Ahmet Cosar. Impacts of routing attacks on surveillance wireless sensor networks. In *2015 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 910–915. IEEE, 2015.

[38] Yubo Song, Shang Gao, Aiqun Hu, and Bin Xiao. Novel attacks in ospf networks to poison routing table. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2017.

[39] Xinyu Wu, Dan Wu, and Eytan Modiano. Queueing delay minimization in overloaded networks via rate control. In *2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1–8. IEEE, 2022.

[40] Yin Sun, C Emre Koksal, and Ness B Shroff. On delay-optimal scheduling in queueing systems with replications. *arXiv preprint arXiv:1603.07322*, 2016.

[41] Sanjeewa Athuraliya, Victor H Li, Steven H Low, and Qinghe Yin. Rem: Active queue management. In *Teletraffic Science and Engineering*, volume 4, pages 817–828. Elsevier, 2001.

[42] Rene L Cruz. A calculus for network delay. i. network elements in isolation. *IEEE Transactions on information theory*, 37(1):114–131, 1991.

[43] Achieving data center networking efficiency. `https://network.nvidia.com/related-docs/whitepapers/WT-PPR-DC-network-efficiency-WEB.pdf`.

[44] Gautam Kumar, Nandita Dukkipati, Keon Jang, Hassan Wassel, Xian Wu, Behnam Montazeri, Yaogong Wang, Kevin Springborn, Christopher Alfeld, Mike Ryan, David J. Wetherall, and Amin Vahdat. Swift: Delay is simple and effective for congestion control in the datacenter. 2020.

[45] Broadcom smart-buffer technology in data center switches for cost-effective performance scaling of cloud applications. `https://docs.broadcom.com/doc/12358325`.

[46] Bo Ji, Changhee Joo, and Ness B Shroff. Delay-based back-pressure scheduling in multihop wireless networks. *IEEE/ACM Transactions on Networking*, 21(5):1539–1552, 2012.

[47] Michael J Neely. Delay-based network utility maximization. *IEEE/ACM Transactions on Networking*, 21(1):41–54, 2012.

[48] Dimitri Bertsekas and Robert Gallager. *Data networks*. Athena Scientific, 2021.

[49] Ping-Chun Hsieh, I Hou, Xi Liu, et al. Delay-optimal scheduling for queueing systems with switching overhead. *arXiv preprint arXiv:1701.03831*, 2017.

[50] Xin Liu and Lei Ying. On achieving zero delay with power-of-d-choices load balancing. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 297–305. IEEE, 2018.

[51] Weina Wang, Mor Harchol-Balter, Haotian Jiang, Alan Scheller-Wolf, and Rayadurgam Srikant. Delay asymptotics and bounds for multi-task parallel jobs. *ACM SIGMETRICS Performance Evaluation Review*, 46(3):2–7, 2019.

[52] Wentao Weng and Weina Wang. Achieving zero asymptotic queueing delay for parallel jobs. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(3):1–36, 2020.

[53] Atilla Eryilmaz, Asuman Ozdaglar, Muriel Médard, and Ebad Ahmed. On the delay and throughput gains of coding in unreliable networks. *IEEE Transactions on Information Theory*, 54(12):5511–5524, 2008.

[54] Leonard Kleinrock. *Communication nets: Stochastic message flow and delay*. Courier Corporation, 2007.

[55] Joseph Pang and R Donaldson. Approximate delay analysis and results for asymmetric token-passing and polling networks. *IEEE journal on selected areas in communications*, 4(6):783–793, 1986.

[56] Hideaki Takagi. Queuing analysis of polling models. *ACM Computing Surveys (CSUR)*, 20(1):5–28, 1988.

[57] Eytan Modiano, Jeffrey E Wieselthier, and Anthony Ephremides. A simple analysis of average queueing delay in tree networks. *IEEE Transactions on Information Theory*, 42(2):660–664, 1996.

[58] Li Xia and Basem Shihada. A jackson network model and threshold policy for joint optimization of energy and delay in multi-hop wireless networks. *European Journal of Operational Research*, 242(3):778–787, 2015.

[59] Leonidas Georgiadis, Michael J Neely, and Leandros Tassiulas. *Resource allocation and cross-layer control in wireless networks.* Now Publishers Inc, 2006.

[60] Xiaoyu Zhao, Wei Chen, Joohyun Lee, and Ness B Shroff. Delay-optimal and energy-efficient communications with markovian arrivals. *IEEE Transactions on Communications*, 68(3):1508–1523, 2019.

[61] Rajat Talak and Eytan H Modiano. Age-delay tradeoffs in queueing systems. *IEEE Transactions on Information Theory*, 67(3):1743–1758, 2020.

[62] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. Bbr: congestion-based congestion control. *Communications of the ACM*, 60(2):58–66, 2017.

[63] Valeria Cardellini, Michele Colajanni, and Philip S Yu. Dynamic load balancing on web-server systems. *IEEE Internet computing*, 3(3):28–39, 1999.

[64] Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma. Performance analysis of load balancing algorithms. *World academy of science, engineering and technology*, 38(3):269–272, 2008.

[65] Patrick Eschenfeldt and David Gamarnik. Join the shortest queue with many servers. the heavy-traffic asymptotics. *Mathematics of Operations Research*, 43(3):867–886, 2018.

[66] Xin Liu and Lei Ying. Steady-state analysis of load-balancing algorithms in the sub-halfin–whitt regime. *Journal of Applied Probability*, 57(2):578–596, 2020.

[67] Rajeev Kumar and Tanya Prashar. Performance analysis of load balancing algorithms in cloud computing. *International journal of computer Applications*, 120(7), 2015.

[68] Daniel E Eisenbud, Cheng Yi, Carlo Contavalli, Cody Smith, Roman Kononov, Eric Mann-Hielscher, Ardas Cilingiroglu, Bin Cheyney, Wentao Shang, and Jinnah Dylan Hosein. Maglev: A fast and reliable software network load balancer. In *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, pages 523–535, 2016.

[69] Dimitri P Bertsekas, Robert G Gallager, and Pierre Humblet. *Data networks*, volume 2. Prentice-Hall International New Jersey, 1992.

[70] Bozidar Radunovic and Jean-Yves Le Boudec. A unified framework for max-min and min-max fairness with applications. *IEEE/ACM Transactions on networking*, 15(5):1073–1083, 2007.

[71] Paolo Giaccone, Emilio Leonardi, and Devavrat Shah. Throughput region of finite-buffered networks. *IEEE Transactions on Parallel and Distributed Systems*, 18(2):251–263, 2007.

[72] Juniper qfx5210 switch. `https://www.juniper.net/us/en/products/switches/qfx-series/qfx5210-switch-datasheet.html`.

[73] Asaf Baron, Ran Ginosar, and Isaac Keslassy. The capacity allocation paradox. In *IEEE INFOCOM 2009*, pages 1359–1367. IEEE, 2009.

[74] Ching-Min Lien, Cheng-Shang Chang, Jay Cheng, and Duan-Shin Lee. Maximizing throughput in wireless networks with finite internal buffers. In *2011 Proceedings IEEE INFOCOM*, pages 2345–2353. IEEE, 2011.

[75] Jim G Dai. On positive harris recurrence of multiclass queueing networks: a unified approach via fluid limit models. *The Annals of Applied Probability*, pages 49–77, 1995.

[76] Jim G Dai and Wuqin Lin. Maximum pressure policies in stochastic processing networks. *Operations Research*, 53(2):197–218, 2005.

[77] Mihalis G Markakis, Eytan Modiano, and John N Tsitsiklis. Delay analysis of the max-weight policy under heavy-tailed traffic via fluid approximations. *Mathematics of Operations Research*, 43(2):460–493, 2018.

[78] Vishal Misra, Wei-Bo Gong, and Don Towsley. Stochastic differential equation modeling and analysis of tcp-windowsize behavior. In *Proceedings of PERFORMANCE*, volume 99, 1999.

[79] Yu Gu, Yong Liu, and Don Towsley. On integrating fluid models with packet simulation. In *IEEE INFOCOM 2004*, volume 4, pages 2856–2866. IEEE, 2004.

[80] Yixin Sun, Maria Apostolaki, Henry Birge-Lee, Laurent Vanbever, Jennifer Rexford, Mung Chiang, and Prateek Mittal. Securing internet applications from routing attacks. *Communications of the ACM*, 64(6):86–96, 2021.

[81] Steve Mansfield-Devine. Ddos goes mainstream: how headline-grabbing attacks could make this threat an organisation's biggest nightmare. *Network Security*, 2016(11):7–13, 2016.

[82] Securing. Securing internet applications from routing attacks, 2021.

[83] Xingang Shi, Yang Xiang, Zhiliang Wang, Xia Yin, and Jianping Wu. Detecting prefix hijackings in the internet with argus. In *Proceedings of the 2012 Internet Measurement Conference*, pages 15–28, 2012.

[84] Gabi Nakibly, Alex Kirshon, Dima Gonikman, and Dan Boneh. Persistent ospf attacks. In *NDSS*, 2012.

[85] Ankur O Bang, Udai Pratap Rao, Pallavi Kaliyar, and Mauro Conti. Assessment of routing attacks and mitigation techniques with rpl control messages: A survey. *ACM Computing Surveys (CSUR)*, 55(2):1–36, 2022.

[86] Anja Feldmann, Philipp Heyder, Michael Kreutzer, Stefan Schmid, Jean-Pierre Seifert, Haya Shulman, Kashyap Thimmaraju, Michael Waidner, and Jens Sieberg. Netco: Reliable routing with unreliable routers. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*, pages 128–135. IEEE, 2016.

[87] Chengjun Wang, Baokang Zhao, Wanrong Yu, Chunqing Wu, and Zhenghu Gong. Routing algorithm based on nash equilibrium against malicious attacks for dtn congestion control. In *International Conference on Availability, Reliability, and Security*, pages 488–500. Springer, 2012.

[88] Hyongju Park and Seth Hutchinson. Worst-case performance of rendezvous networks in the presence of adversarial nodes. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5579–5585. IEEE, 2013.

[89] Andrew D Ferguson, Steve D Gribble, Chi-Yao Hong, Charles Edwin Killian, Waqar Mohsin, Henrik Muehe, Joon Ong, Leon Poutievski, Arjun Singh, Lorenzo Vicisano, et al. Orion: Google's software-defined networking control plane. In *NSDI*, pages 83–98, 2021.

[90] Xinyu Wu, Dan Wu, and Eytan Modiano. Overload balancing in single-hop networks with bounded buffers. In *2022 IFIP Networking Conference (IFIP Networking)*, pages 1–9. IEEE, 2022.

[91] Shizhen Zhao, Rui Wang, Junlan Zhou, Joon Ong, Jeffrey C Mogul, and Amin Vahdat. Minimal rewiring: Efficient live expansion for clos data center networks. In *NSDI*, pages 221–234, 2019.

[92] Mingyang Zhang, Jianan Zhang, Rui Wang, Ramesh Govindan, Jeffrey C Mogul, and Amin Vahdat. Gemini: Practical reconfigurable datacenter networks with topology and traffic engineering. *arXiv preprint arXiv:2110.08374*, 2021.

[93] Brandon Heller, Srinivasan Seetharaman, Priya Mahadevan, Yiannis Yiakoumis, Puneet Sharma, Sujata Banerjee, and Nick McKeown. Elastictree: Saving energy in data center networks. In *Nsdi*, volume 10, pages 249–264, 2010.

[94] Charles E Leiserson. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE transactions on Computers*, 100(10):892–901, 1985.

[95] Brian Lebiednik, Aman Mangal, and Niharika Tiwari. A survey and evaluation of data center network topologies. *arXiv preprint arXiv:1605.01701*, 2016.

[96] Weiyang Wang, Moein Khazraee, Zhizhen Zhong, Manya Ghobadi, Zhihao Jia, Dheevatsa Mudigere, Ying Zhang, and Anthony Kewitsch. {TopoOpt}: Co-optimizing network topology and parallelization strategy for distributed training jobs. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 739–767, 2023.

[97] Michael J Neely. Stability and capacity regions or discrete time queueing networks. *arXiv preprint arXiv:1003.3396*, 2010.

[98] David Zats, Tathagata Das, Prashanth Mohan, Dhruba Borthakur, and Randy Katz. Detail: Reducing the flow completion time tail in datacenter networks. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 139–150, 2012.

[99] NM Mosharaf Kabir Chowdhury. *Coflow: A networking abstraction for distributed data-parallel applications.* University of California, Berkeley, 2015.

[100] Kanthi Nagaraj, Dinesh Bharadia, Hongzi Mao, Sandeep Chinchali, Mohammad Alizadeh, and Sachin Katti. Numfabric: Fast and flexible bandwidth allocation in datacenters. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 188–201, 2016.

[101] Alok Kumar, Sushant Jain, Uday Naik, Anand Raghuraman, Nikhil Kasinadhuni, Enrique Cauich Zermeno, C Stephen Gunn, Jing Ai, Björn Carlin, Mihai Amarandei-Stavila, et al. Bwe: Flexible, hierarchical bandwidth allocation for wan distributed computing. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 1–14, 2015.

[102] Michael J Neely. Stochastic network optimization with application to communication and queueing systems. *Synthesis Lectures on Communication Networks*, 3(1):1–211, 2010.

[103] Rui Zhang-Shen and Nick McKeown. Designing a predictable internet backbone with valiant load-balancing. In *Quality of Service–IWQoS 2005: 13th International Workshop, IWQoS 2005, Passau, Germany, June 21-23, 2005. Proceedings 13*, pages 178–192. Springer, 2005.

[104] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C Snoeren. Inside the social network's (datacenter) network. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 123–137, 2015.

[105] Qiao Zhang, Vincent Liu, Hongyi Zeng, and Arvind Krishnamurthy. High-resolution measurement of data center microbursts. In *Proceedings of the 2017 Internet Measurement Conference*, pages 78–85, 2017.

[106] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al. B4: Experience with a globally-deployed software defined wan. *ACM SIGCOMM Computer Communication Review*, 43(4):3–14, 2013.

[107] Paul Dupuis, Kevin Leder, and Hui Wang. Importance sampling for weighted-serve-the-longest-queue. *Mathematics of Operations Research*, 34(3):642–660, 2009.

[108] Joao P Hespanha. *Linear systems theory.* Princeton university press, 2018.

[109] Chi-Kwong Li and Fuzhen Zhang. Eigenvalue continuity and ger\v {s} gorin's theorem. *arXiv preprint arXiv:1912.05001*, 2019.

[110] Wladyslaw Kulpa. The poincaré-miranda theorem. *The American Mathematical Monthly*, 104(6):545–550, 1997.

[111] David G Feingold, Richard S Varga, et al. Block diagonally dominant matrices and generalizations of the gerschgorin circle theorem. *Pacific Journal of Mathematics*, 12(4):1241–1250, 1962.

[112] Carl D Meyer. *Matrix analysis and applied linear algebra, Chapter 8*, volume 71. Siam, 2000.

[113] Michael Charles Irwin. *Smooth dynamical systems*, volume 17. World Scientific, 2001.

[114] Giacomo Como, Ketan Savla, Daron Acemoglu, Munther A Dahleh, and Emilio Frazzoli. Robust distributed routing in dynamical networks–part ii: Strong resilience, equilibrium selection and cascaded failures. *IEEE Transactions on Automatic Control*, 58(2):333–348, 2012.

[115] Anderson Santos da Silva, Juliano Araujo Wickboldt, Lisandro Zambenedetti Granville, and Alberto Schaeffer-Filho. Atlantic: A framework for anomaly traffic detection, classification, and mitigation in sdn. In *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, pages 27–35. IEEE, 2016.

[116] Junlan Zhou, Malveeka Tewari, Min Zhu, Abdul Kabbani, Leon Poutievski, Arjun Singh, and Amin Vahdat. Wcmp: Weighted cost multipathing for improved fairness in data centers. In *Proceedings of the Ninth European Conference on Computer Systems*, pages 1–14, 2014.

[117] Douglas S Altner, Özlem Ergun, and Nelson A Uhan. The maximum flow network interdiction problem: valid inequalities, integrality gaps, and approximability. *Operations Research Letters*, 38(1):33–38, 2010.

[118] Advait Dixit, Pawan Prakash, Y Charlie Hu, and Ramana Rao Kompella. On the impact of packet spraying in data center networks. In *2013 Proceedings IEEE INFOCOM*, pages 2130–2138. IEEE, 2013.

[119] Jack Edmonds and Richard M Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264, 1972.

[120] David R Karger. Global min-cuts in rnc, and other ramifications of a simple min-cut algorithm. In *Soda*, volume 93, pages 21–30. Citeseer, 1993.

[121] Erika Melder. A chronology of set cover inapproximability results. *arXiv preprint arXiv:2111.08100*, 2021.