Optimal Control for Wireless Software Defined Networks: Theory and Implementation

by

Quang Minh Nguyen

B.S., National University of Singapore (2020)

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

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2022

Author .....
Department of Electrical Engineering and Computer Science
August 26, 2022

Certified by .....
Eytan H. Modiano
Professor of Aeronautics and Astronautics
Thesis Supervisor

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

Optimal Control for Wireless Software Defined Networks: Theory and Implementation

by

Quang Minh Nguyen

Submitted to the Department of Electrical Engineering and Computer Science
on August 26, 2022, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

Wireless Software Defined Network (SDN) has emerged as a new programmable network paradigm that facilitates flexibility in robust control and management. Toward the production-level network deployment at scale, there has been a surge of interest in distributed architectures of wireless SDN. Despite the inherent importance, optimal network control for either centralized or distributed wireless SDN has remained an open problem, where previous works either fail to account for wireless interference constraints, or are only sub-optimal in throughput due to quasi-static shortest path routing. Though throughput-optimal and well-established in the literature, the Back-Pressure (BP) algorithm is not compatible with wireless SDN architecture. In contrast, the recently developed Universal Max-Weight (UMW) policy also achieves throughput-optimality, yet permits algorithmic structure more congruent with SDN's requirements. Unlike BP, UMW pre-computes a fixed route per-packet upon a packet arrival, which can be integrated with the flow installation phase of SDN, and uses novel easy-to-track virtual queues in place of physical queues (of backlogged packets), whose operations are not supported by SDN switches. In this thesis, we propose novel UMW-based optimal control frameworks for both centralized and distributed wireless SDN that achieve the full network capacity and support an arbitrary mix of multi-type traffic.

For centralized wireless SDN, we develop a Mininet-based implementation of the UMW framework to evaluate its performance. In order to improve robustness in dynamic wireless environments, we modify the UMW algorithm to enable re-routing around failed links. Compared against the conventional SDN shortest path routing, our algorithm improves throughput by over 100% and significantly reduces average per-packet delay in high-throughput regime. We further present the Randomized UMW (RUMW) algorithm that performs scheduling in linear time, yet still maintains the throughput-optimality under the setting of dynamic network.

For distributed wireless SDN, our proposed Distributed Universal Max-Weight (DUMW) algorithm is throughput-optimal and non-trivially extends the UMW policy to permit distributed control and optimal inter-domain scheduling under the setting of heterogeneously delayed network state information. Furthermore, we design controller

synchronization strategies that resolve the problem of multi-domain flow installation and are tailored to DUMW for maintaining throughput-optimality with negligible communication overhead. Extensive simulations validate the throughput-optimality and exhibit superior scalability of our framework.

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

Acknowledgments

First of all, I am deeply thankful to my advisor, Prof. Eytan Modiano, for his invaluable guidance and constant support in the completion of this thesis. His vision, expertise and attention have been crucial for my personal growth as a researcher.

I would like to acknowledge Shahir Rahman for our collaboration on the system implementation of centralized SDN. The Mininet-based experiments in Chapter 1 are based on the system developed and contributed by him.

Another big part of my graduate study was the wonderful labmates in CNRG. I very much appreciate the technical discussions with Bai Liu, Xinzhe Fu, and Vishrant Tripathi, which had helped me a lot in my transition to the new fields and problem formulations. Additionally, I am also thankful to other group members Nick Jones, Chirag Rao, Sathwik Chadaga, Jerrod Wigmore and Xinyu Wu for sharing not only the office but also the academic life with me; we have taken courses, had meals and hung out together. I am also grateful to my new and old friends, who have made my life and Ph.D. journey at MIT enjoyable.

Finally, I would like to express my deepest gratitude to my family, including my grandparents (Them Nguyen, Lam Nguyen, Van Nguyen, Loc Nguyen), parents (Hanh Nguyen, Luat Nguyen) and aunt (Luyen Nguyen). The unconditional love and support from them are irreplaceable throughout the ups and downs of this journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

1.1.1 Optimal Control for Wireless SDN

Software-Defined Networking (SDN) [35] emerged from the urgent need for a programmable networking framework with adaptive reconfiguration to meet modern networking requirements [30]. Its flexibility has facilitated the implementation of widespread network management functionalities such as routing [14] , load-balancing [62] and traffic engineering [44]. Unlike traditional network architectures, SDN decouples the data plane from the control plane, where the data forwarding devices, called switches, passively execute the instructions received from the programmable network controller. This unique architecture is often incompatible with many state-of-the-art network control algorithms, which is further exacerbated in wireless systems [12] with the requirements for dynamic control, distributed operations [59], and link scheduling. Nonetheless, there has been increasing interest in wireless SDN thanks to the surge in mobile communication [8] and wireless infrastructures [10].

Despite its inherent importance, the literature of scheduling in wireless SDN remains nascent. In terms of SDN scheduling, all the previous work fails to accommodate interference constraints, which are a critical element in any wireless networking system [19]. Moreover, the vast literature of routing in wireless SDN has only

considered quasi-static shortest path (SP) routing [12, 23], which is known to be sub-optimal in terms of throughput and delay. While the Back-Pressure (BP) algorithm is known to be throughput-optimal in wireless networks, it is incompatible with the SDN architecture, hindering its adoption. Moreover, though SDN routing that is robust to link failures via pre-computed backup paths has been well studied [18, 27, 28, 52], this naive approach is only effective against rare failures and is ineffective in highly dynamic wireless networks. The first goal of the thesis is thus to design a new framework for routing and scheduling in wireless SDN that is throughput-optimal and can handle generalized interference constraints.

1.1.2 Optimal Control for Distributed Wireless SDN

In addition to the challenges introduced by the wireless setting, the SDN architecture possesses some inherent limitations. Utilizing its global view of the network information, the logically centralized controller can be designed to make optimal decisions for application performance. However, the centralized nature of SDN incurs significant communication overhead to the control plane in large-scale networks [57] and suffers from the single point-of-failure problem [45]. Consequently, distributed SDN [2] has emerged to mitigate the scalability and reliability bottlenecks.

A distributed wireless SDN system decomposes the underlying network topology into inter-connected sub-networks, referred to as domains, and assigns each domain to a physically separate SDN controller. The controllers then synchronize with each other to partially or fully maintain their global view of the network state, which can be utilized to enhance decision making for inter-domain tasks. The coordination among controllers has attracted a lot of research [36]. While there have been several consistency models considered in the literature, the two most predominant classes are: *strong consistency* [3,20] and *eventual consistency* [1,20,48,51] . By requiring all the controllers to be synchronized at any time, strongly consistent protocols strive to maintain fresh global network information for optimizing application performance. However, its practicality is hindered by the unreliable nature of network communications [43] and the prohibitively high overhead incurred by frequent controller coordination [17]. On the

other hand, eventual consistency requires the controllers to be eventually synchronized, thereby allowing for temporarily inconsistent network view. The overhead reduction due to the relaxed synchronization requirement has been both a blessing and a curse: distributed SDN can scale pervasively and has been adopted at production-level [16, 20, 24, 47, 56] with wide applicability [13, 40, 54], yet the inconsistent network information significantly degrades the inter-domain application performance [26]. Improving the performance of ad-hoc inter-domain tasks via customized algorithms and synchronization strategies is an active area of research [37]. In particular, while there have been several applications considered in the literature, such as traffic engineering [29], load-balancing [15] or utility maximization [7], optimal network control comprised of routing and scheduling is the most prominent for being the backbone of network operations [53]. To this end, we focus on studying the optimal network control framework for distributed wireless SDN under the eventual consistency model.

To the best of our knowledge, optimal network control for distributed wireless SDN has remained an open problem. We attribute this to the nascent literature on network control for wireless SDN, and the challenge of making decisions with respect to inconsistent view of global network state information (NSI) and flow statistics. For inter-domain routing, the vast literature relies on quasi-static shortest path (SP) algorithms [21, 38], where much of the work is focused on optimizing the controller-synchronization rate [39, 60, 61]. Since SP routing is known to be sub-optimal in terms of throughput and not tailored to handling heterogeneous view of NSI, all of the proposed algorithms, even when predicated on sophisticated synchronization strategies, still operate below the throughput capacity of the network. In fact, to the best of our knowledge, no work has theoretically studied the throughput capacity of the inter-domain routing approaches. In terms of wireless scheduling for SDN, all the previous works fail to accommodate interference constraints, which are a critical element in any wireless networking system [19]. To fill this gap, we thus investigate the unsolved yet critical problem of optimal network control for distributed wireless SDN. Our second goal of this thesis is to develop a new algorithm that is theoretically

throughput-optimal and practically congruent with the distributed SDN architecture and the wireless environments.

1.2 Universal Max Weight (UMW)

We attribute the lack of an optimal control solution for wireless SDN in the literature to the unconventional routing requirement and limited control of physical queues of SDN, which will be discussed in details in Section 2.1.2 later. To this end, we propose the adoption of the recently developed Universal Max-Weight (UMW) algorithm [46] as a general network control framework for SDN that is throughput-optimal and compatible with the SDN architecture.

Nevertheless, in its original form, the UMW policy incurs significant computational cost for obtaining scheduling decisions, which can be practically prohibitive in certain modern wireless SDNs, especially in view of the emerging low-power devices in wireless systems. Moreover, toward optimal control for distributed wireless SDN, the analytical model of UMW lacks the generality to readily be extended to distributed control. In particular, the UMW policy is centralized in nature, which is incompatible with inter-domain operations and is not designed to deal with sophisticated dynamic networks.

1.3 Contributions

In this thesis, we propose novel UMW-based unified optimal control frameworks for both centralized and distributed wireless SDN, that fully address the aforementioned challenges. Our contributions can be summarized as follows:

- In chapter 2, we study the adaptation of UMW as an optimal control framework for wireless SDN. Next, we implement UMW on the real SDN system comprised of a Mininet backbone, OpenFlow switches and a Ryu controller. In terms of both throughput and latency, the results exhibit superior performance of UMW over the conventional SP routing. We also illustrate UMW's congruence with

SDN's requirements and natural extension to dynamic topologies that experience link failures. Our robust UMW version consistently improves the throughput of standard UMW under the setting of dynamic link failures, confirming the efficiency of our approach. Finally, in view of the emerging low-power devices in wireless systems, we propose the Randomized UMW (RUMW) algorithm with linear complexity scheduling which is throughput-optimal under the dynamic network setting. Finally, we propose the Randomized UMW (RUMW) algorithm for dynamic network setting that performs scheduling in linear time.

- In chapter 3, we present the Distributed Universal Max-Weight (DUMW) algorithm for distributed wireless SDN that is throughput-optimal and can handle generalized wireless interference constraints. In particular, we formulate the problem of optimal inter-domain routing and scheduling for distributed wireless SDN, whereby our analytical model is the first to capture the interplay between distributed control and SDN system idiosyncrasies. The fully wireless system studied in this work also accommodates wireless inter-domain communication, which was neglected by the previous works whereby wired inter-domain communication was assumed for simplicity [59]. Next, we propose a novel scheduling algorithm for DUMW, which is optimal under the considered setting of heterogeneously delayed NSI with hierarchy, and is of independent interest. For inter-domain routing, DUMW resolves the challenge of inter-domain flow installation by enforcing consensus among controller through periodic synchronization. Deviating from the vast literature on distributed SDN that relies on generic consensus protocols [37], we design wireless controller synchronization mechanisms specifically customized to DUMW so as to maintain throughput-optimality with negligible communication overhead.

Chapter 2

Optimal Control for Centralized Wireless SDN

2.1 Wireless SDN

2.1.1 Wireless SDN Architecture

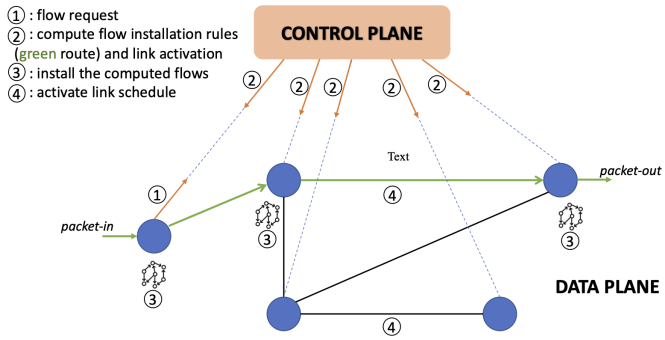


Figure 2-1: SDN architecture and packet life cycle.

The SDN architecture decouples the control plane from the data plane (Figure 2-1). This is in contrast to traditional network architecture where control logic is embedded to the forwarding hardware.

Data plane: The data plane is the SDN's infrastructure, comprised of forwarding devices, called switches, all connected to the control plane and interconnected by links to form the underlying network topology. While the switches are responsible for deploying the flows, i.e. packet forwarding rules, and activating their respective

links for packet transmissions, they rely on the control plane's logical computation of packet routes and link schedules.

Control plane: The control plane is in charge of all logical operations and essential functionalities of the network, such as traffic management, device configuration and network management. It gathers network state information from the data plane to compute routing and scheduling decisions, to be sent to the switches for packet routing and link activation. The control plane acts as a logically centralized controller, where multiple physically distributed controllers can be deployed as long as the collective behavior is consistent with the behavior of a single controller. In this paper, we interchangeably refer to the control plane as the controller.

Packet life cycle: Whenever a packet arrives at a switch in the network, a flow request is generated by the switch and sent to the control plane. The controller accumulates all the flow requests and network state information to consequently compute the flow installation rules and scheduling decisions. The link schedule is a subset of links that can be simultaneously activated, according the the interference constraints (e.g. primary or node-exclusive model [6]) of the network. The controller then deploys the packet forwarding rules, i.e. flows, onto the switches belonging to the computed routes and notifies the switches adjacent to activated links (in the link schedule) to transmit packets. The process is summarized in Figure 3-1.

Wireless model: The inter-switch communication corresponds to the network link activation for packet transmission, whereby wireless interference constraints [6] must be satisfied. We assume reliable and stable wireless controller-switch communication in order to ensure error-free flow installation and accumulation of network statistics. Reliable controller-switch communication is required by the SDN literature and, under the wireless setting, is facilitated via the controller placement in proximity to switches [41] or enhanced coding and retransmission at the physical and link layers.

2.1.2 Challenges of SDN Optimal Control

While facilitating flexibility and programmability, SDN architecture imposes new challenges for network flow control that hinder the adoption of prevalent routing and

scheduling algorithms.

Unconventional routing requirement: traditional multi-hop network routing schemes, including the throughput optimal BP policy, admit hop-by-hop routing decisions that are made along the way of packets' traversal. On the other hand, the installation of packet forwarding rules in SDN's workflow requires the route per packet to be established immediately upon a packet arrival and further deployed afterward.

Limited control of physical queues: Optimal control schemes such as BP leverage the physical queues of backlogged packets for both routing and scheduling. While traditional network architecture well supports operations on physical queues, SDN's switches lack logical capability for managing or distilling statistics from the physical queues. Although the SDN's controller can still request simple packet-in and packet-out information from the switches to maintain its own estimate of the physical queues, such process imposes significant burden on the communication bandwidth between the control plane and the data plane.

2.1.3 A New Optimal Control Framework for SDN

We attribute the lack of an optimal control solution for SDN in the literature to the two challenges described above. To bridge this gap, we propose the adoption of UMW as a general network control framework for SDN. UMW is a throughput-optimal routing and scheduling scheme, and is compatible with the SDN architecture, as its distinctive features directly address the SDN challenges discussed in Section 2.1.2, namely:

1. UMW computes a route per packet class immediately upon the arrival of any new packet and prescribes the route for the packet throughout its traversal in the network.

2. UMW maintains a system of virtual queues instead of physical queues to perform routing and scheduling. The update of the virtual queues does not require real-time packet-in and packet-out information from the switches.

2.2 Universal Max-Weight (UMW)

The UMW algorithm was introduced in [46] as a routing and scheduling scheme that maximizes network capacity and supports generalized flow problems including unicast, multicast and broadcast. The core idea of UMW is to relax the precedence constraint [25] to obtain a simple single-hop virtual queue system in place of the physical queues. The policy is then designed to stabilize the virtual queues, which is further proven to be sufficient for network throughput-optimality. To formally define the UMW policy, we hereby describe the system model in Section 2.2.1, establish the virtual queue dynamics in Section 2.2.2, and present the main algorithm in Section 2.2.3.

2.2.1 Network Model

We consider a wireless network with arbitrary topology represented by the directed graph $G(V, E)$, where V is the set of nodes, E is the set of directed point-to-point links. Time is slotted. At any time slot, only certain subsets of links can be activated, according to the interference constraints of the network. The set of all admissible link activations is denoted by $\mathcal{M} \in \{0, 1\}^{|E|}$, where 1 denotes that the link is activated and vice versa. An incoming packet belongs to some class c traffic, which is identified by its source node $s^{(c)} \in V$, the set of its required destination nodes $\mathcal{D}^{(c)} \subseteq V$ and the set $\mathcal{T}^{(c)}$ of all admissible routes from $s^{(c)}$ to $\mathcal{D}^{(c)}$. An admissible route $T^{(c)} \in \mathcal{T}^{(c)}$ is a tree rooted at the source node $s^{(c)}$ with the set of leaves formed by $\mathcal{D}^{(c)}$. We define the set of distinct classes of incoming traffic as $\mathcal{C}$. Packet arrivals are i.i.d. at every slot. At time slot t, $A^{(c)}(t)$ packets from class c arrive at source node $s^{(c)}$. The mean rate of arrival for class c is $\mathbb{E}[A^{(c)}(t)] = \lambda^{(c)}$.

Precedence Constraints: In a multi-hop network, if a packet is routed along some path $P = e_1 - e_2 - ... - e_k$, where $e_i \in E$ is the i^{th} edge on the path, then it can be transmitted over the link e_j only after being transmitted by all the $j - 1$ *preceding* links $e_1, e_2, ..., e_{j-1}$.

2.2.2 The Virtual Queue Process $\{\mathbf{Q}(t)\}_{t\geq 1}$

The UMW policy controls the $|E|$-dimensional stochastic virtual queue process $\mathbf{Q}(t) = (Q_e(t), e \in E)$ imitating a fictitious queueing network without the precedence constraints. For all $A^{(c)}(t)$ packets of class c arriving at the source node $s^{(c)}$ during time slot t, the UMW policy prescribes a suitable route $T^{(c)}(t) \in \mathcal{T}^{(c)}$ to them. These incoming packets are routed along the prescribed physical path, while the corresponding virtual queues are immediately incremented. The total number of virtual packet arrivals to the virtual queue Q_e at time t is:

$$A_e(t) = \sum_{c\in\mathcal{C}} A^{(c)}(t)\mathbb{1}(e \in T^{(c)}(t)), \quad \forall e \in E. \tag{2.1}$$

Since we assume the relaxation of precedence constraints in the virtual network, any packet present in the virtual queues is eligible for service. Let $\{\boldsymbol{\mu}(t)\}_{t\geq 1}$ be the service process of the virtual queues controlled by the UMW policy. At time slot t, the service vector is selected with respect to the activation constraints, i.e. $\boldsymbol{\mu}(t) \in \mathcal{M}$, and further used for link activation in the physical network. Then the virtual queue process $\{\mathbf{Q}(t)\}_{t\geq 1}$ evolves as the following dynamics:

$$Q_e(t+1) = \big(Q_e(t) + A_e(t) - \mu_e(t)\big)^+, \quad e \in E. \tag{2.2}$$

2.2.3 UMW Algorithm

Utilizing the virtual queue process, the UMW scheme performs routing and dynamic scheduling on the physical network by solving weighted min-cost and max-weight problems.

Routing: For any class $c \in \mathcal{C}$ packet at time t, select route $T^{(c)}(t) \in \mathcal{T}^{(c)}$ that solves the weighted min-cost problem:

$$T^{(c)}(t) \in \underset{T^{(c)}\in\mathcal{T}^{(c)}}{\operatorname{argmin}} \left(\sum_{e\in E} Q_e(t)\mathbb{1}(e \in T^{(c)}) \right) \tag{2.3}$$

Upon the arrival of a new packet, its route is immediately computed according to (3.8) and fixed throughout execution. This unique algorithmic structure of UMW makes it compatible with the SDN architecture.

Scheduling: Select the link schedule $\boldsymbol{\mu}(t) \in \mathcal{M}$ that solves the max-weight problem:

$$\boldsymbol{\mu}(t) \in \underset{\boldsymbol{\mu} \in \mathcal{M}}{\operatorname{argmax}} \left(\sum_{e \in E} Q_e(t) \mu_e \right), \tag{2.4}$$

and forward physical packets from the physical queues over the activated links $\boldsymbol{\mu}(t)$.

Virtual queue update: Update the virtual queues assuming a precedence-relaxed system via (3.7).

While dynamically routing and scheduling packets via (3.8) and (3.9) to stabilize the virtual queues, UMW transmits real packets in the physical network and is guaranteed to achieve the full capacity region [46].

2.3 UMW Implementation in SDN

This Section presents details of our UMW implementation in SDN and related design perspectives. The high-level architecture is depicted in Figure 2-2, which has three hierarchical layers: the application plane, the control plane and the data plane. The northbound interface serves as the customized API passing application-oriented instructions between the application and the control plane. The southbound interface abstracts the communication between the control plane and the data plane. We also extend the three planes with additional features that support link failure simulation, network measurement and flow management.

2.3.1 Application Plane and Northbound Interface

The application plane contains high level programs that disseminate information to controllers via a northbound interface. An SDN thus may contain multiple applications for use cases such as load balancing, security, telemetry, etc. The controller then communicates with the switches to properly configure them for the applications.

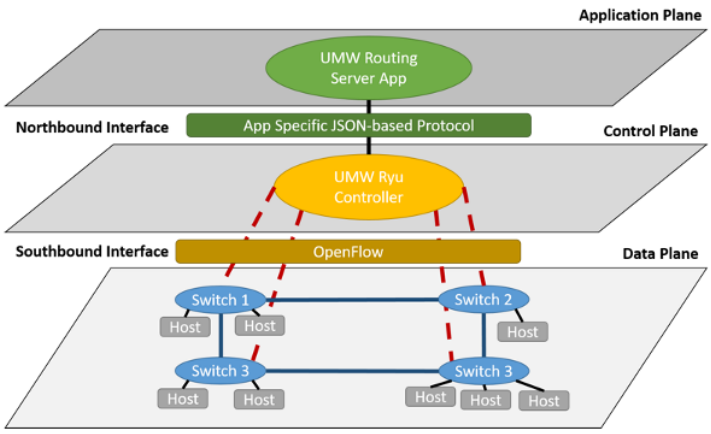


Figure 2-2: Architecture of SDN with UMW controller

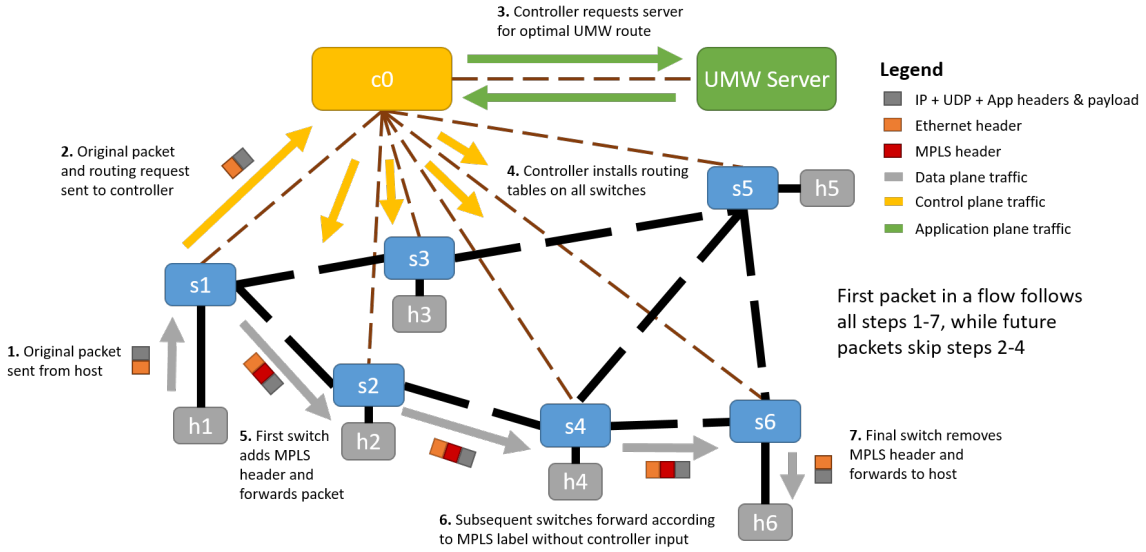


Figure 2-3: Workflow of flow installation and packet forwarding

UMW controller: We build the UMW algorithm (described in Section 2.2.3) at the application plane. The UMW server is responsible for keeping track of the virtual queues based on flow information from the controller, computing optimal paths, and decrementing virtual queues over time.

Northbound interface: The application layer communicates with the controller using TCP synchronously with a custom protocol using JSON packets. All of the routing and scheduling decisions are made in the application layer, which are then disseminated from the control plane to the data plane. The controller stores hashtables and other necessary data structures for keeping track of OpenFlow port numbers,

OpenFlow group numbers, MPLS labels, packet headers and link traffic statistics.

Handling link failures: We also add a robust mechanism to the original UMW algorithm that handles link failures and adapts to the dynamic nature of the SDN environment. When a link e fails, it is temporarily excluded from the topology and a copy of its latest virtual queue $Q_e(t)$ right before the failure is stored. Due to the exclusion of the failed link e from routing (3.8) and scheduling (3.9), any packet afterward would be routed around e and any schedule would not activate e. Once the failed link is restored, it is included in the topology with its old virtual queue value.

2.3.2 Data Plane

The data plane is implemented using Mininet [31], which simulates multiple switches, hosts, and controllers on the same machine. Hosts are emulated as a group of processes running in a network namespace. Yet each independent host has only its assigned processes visible, and is provided with its own network interfaces. Switches are emulated using Open vSwitch, which is compatible with OpenFlow and is responsible for routing with respect to table entries and switching packets.

Switches, hosts and links: Openflow switches forward packets via the packet delivery semantics installed by the controller. Each virtual host is connected to a switch. Switches are connected to each other via Ethernet links that have a configurable bounded capacity, and connected to the controller via dedicated links with unbounded capacity. These elements form the backbone for network simulation.

Network traffic: Traffic is generated in the data plane via Python applications running on the host nodes. Each Python sending process periodically sends UDP packets, which will be later received by Python receiving processes at the hosts.

Packet header: Packet header is critical for the execution of routing decisions and network traffic measurement. The standard IPv4 header section of each packet is comprised of the following static information: number of bytes per session, IP address, port number, OS process ID, timestamp of session, and 8 bytes of randomly generated salt value for hashing. This information is utilized to uniquely identify the session to which any packet belongs. We further augment the packet with a 24 byte dynamic

header that contains the session size, total bytes sent and the timestamp when the packet is generated. The information of session size and bytes sent is required by the UMW controller to correctly increment the virtual queues according to (3.7). The whole dynamic header is also used for post-simulation network traffic logging and analysis.

2.3.3 Control Plane and Southbound Interface

We design and implement the controller using the Ryu framework (under the Apache 2.0 license), and utilizing OpenFlow 1.5 [30] as the southbound interface.

Flow installation and packet forwarding: Packets arrive at the network in the form of sessions. A session contains multiple packets, identified by the port numbers and the same source and destination IPs. The UMW controller initially installs a routing table entry to forward all incoming packets to the control plane. Receiving the first packet in a new session, the controller parses the session size information from the packet, which is further communicated to the UMW server at the application layer to increment the virtual queues and compute the optimal route for the session's packets. Given the routing decisions, the controller augments every incoming packet with a Multiprotocol Label Switching (MPLS) label at only the first switch, i.e. the source node at which packets arrive, and installs the routing tables onto all the participating switches [1]. The routing tables allow the switches to forward packets to the next ports based on the MPLS labels without any further communication with the control plane. The whole workflow is illustrated in Figure 2-3.

2.3.4 Additional Features

We further extend the system with additional features to support network traffic measurement, built-in topology generation and simulation of link failures.

Network traffic measurement: As discussed in Section 2.3.2, Python receiving processes at the Mininet hosts receive UDP packets when they reach the destination

[1] i.e. switches that are on the routing path of the packet.

nodes. The UDP destination nodes extract the headers of each incoming UDP packet to identify its source node, timestamp, and packet size for later network traffic measurement. The Python receiving processes periodically log the output throughout the duration of the simulation. At the end of the simulation, the log files are parsed to compute the average delay and throughput. For packets that were dropped or did not reach the destination by the end of the simulation, we set their delay value to infinity.

Topology generation: Our framework wraps the Mininet library and allows for generation of a variety of network topologies. The generation of the topology is based on our own standardized JSON configuration protocol, which allows for the topology information to be passed to the controller and server in a consistent way, for making routing decisions.

Link failure simulation: We utilize the Mininet SDN system to simulate dynamic single link failures, where the interval between two failed links follows exponential distribution and every time a random link is chosen to be deactivated for a fixed period of time. Whenever a link is deactivated or reactivated, the switches attached to the link send OpenFlow port status messages to the controller to notify the link's status change.

2.4 Experiments

2.4.1 Experimental Design

Our topology library supports automatic generation of multiple toplogies including bidirectional grid of arbitrary size and the National Science Foundation (NSF) topology. In this work, we compare our proposed UMW scheme with SP routing, vastly used in SDN network control, through extensive testings on the 3×3 grid (Figure 2-4) and the NSF topology (Figure 2-5). Some complementary tests are also conducted on the 6×6 grid with 36 nodes and 120 links, thereby describing a moderately complex and large-scale network in practice. Each link in all our considered topologies is bidirectional, emulating full-duplex ethernet and effectively corresponding to two

unidirectional links.

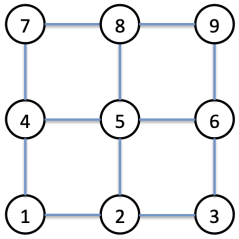


Figure 2-4: Bidirectional 3×3 grid

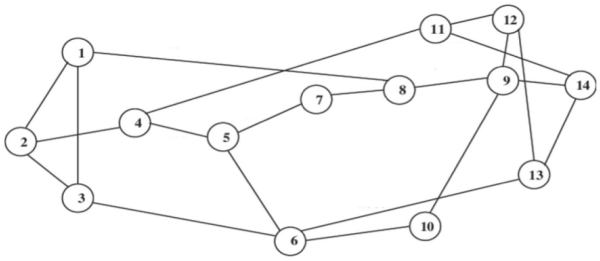


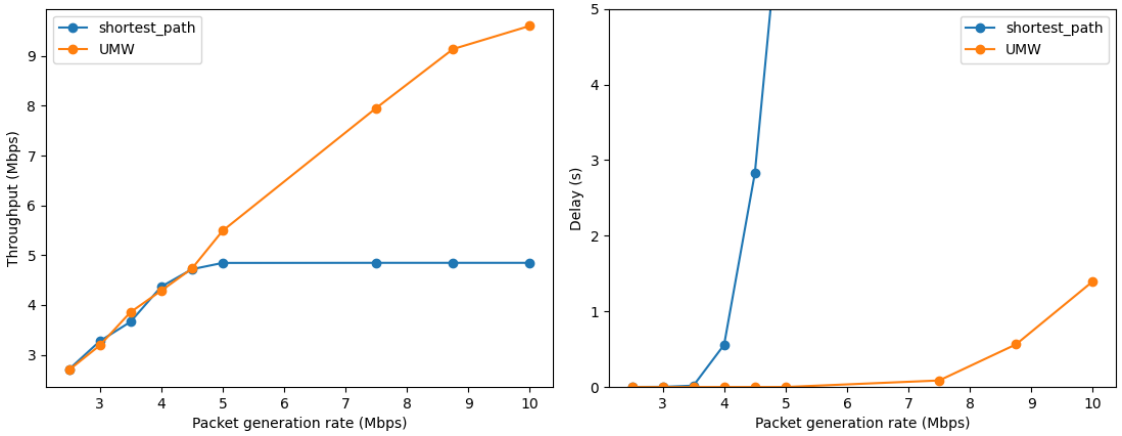
Figure 2-5: NSF topology

To replicate the realistic high-throughput regime, we set the bandwidth of each link to 5 Mbps. We limit the bandwidth based on Mininet simulation scalability tests we conducted, to ensure that the simulation is as realistic as possible and is not hampered by the limitations of CPU processing capabilities, since all hosts and switches are emulated on one machine. We experiment with unicast traffic, which corresponds to a source-destination (s-d) pair, and assume Poisson arrivals at the sources all with the same packet generation rate. To test the network capacity and saturation point, we gradually increase the packet generation rate at every source, and report the throughput (averaged over the s-d pairs) and packet delay (averaged over the total number of packets). Every simulation (i.e. one point in a plot) is run for 10 minutes.

2.4.2 Throughput Optimality of UMW via Dynamic Routing

We start with single s-d pair test, which is the simplest, yet already indicative of UMW's optimality and unique features.

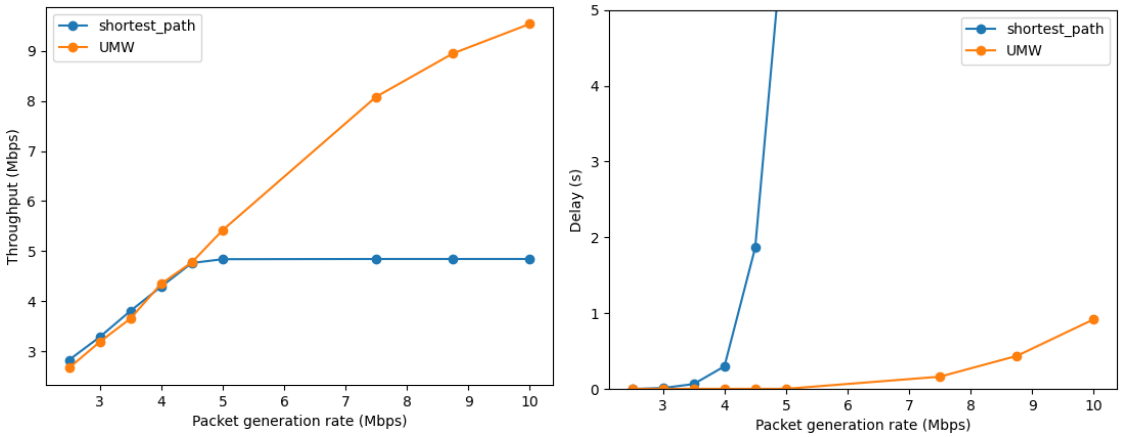
Benefit of dynamic routing: For the 3×3 grid, we consider the s-d pair (1, 9) and report the results in Figure 2-6. Since the bandwidth of every link is 5 Mbps and the destination node 9 has two incoming edges, the maximum throughput is 10 Mbps. Our throughput results illustrate that while UMW achieves close to the optimal rate 10 Mbps, SP routing loses half of the capacity, i.e. its maximum rate is only around 5



(a) Average throughput

(b) Average packet delay

Figure 2-6: Test on 3×3 grid with single s-d pair (1, 9)

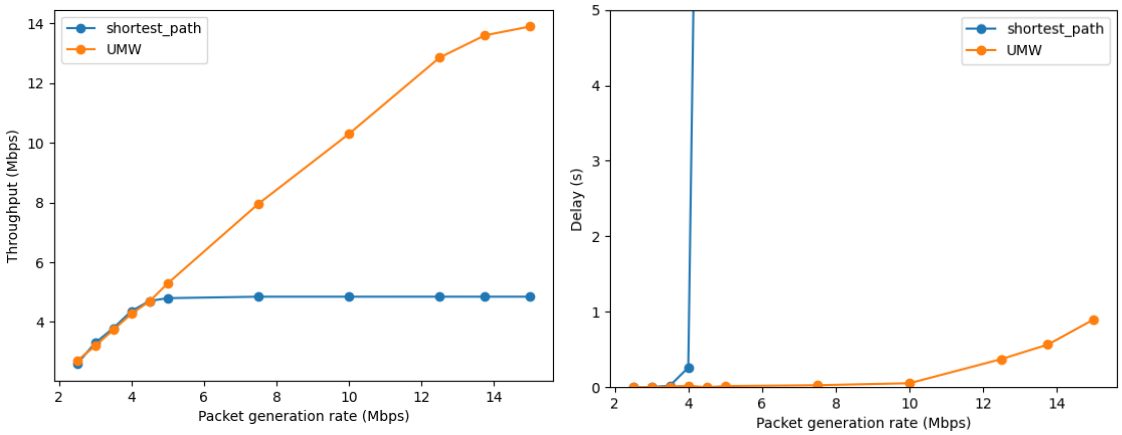


(a) Average throughput

(b) Average packet delay

Figure 2-7: Test on 6×6 grid with single s-d pair (1, 36)

Mbps. The limitation of SP routing is due to its deployment of a fixed path and thus inability to dynamically switch between paths to utilize both of the destination node's incoming edges. The delay results also show that SP routing leads to dropped packets at a packet generation rate rate of 5+ Mbps due to infinite delay. On the other hand, all packets of UMW are eventually served. The same effects on throughput and delay can be observed again in the test on 6×6 grid with the single s-d pair comprised of the bottom-left source node and top-right destination node (Figure 2-7). Though its structure is very similar to the 3×3 grid test, this 6×6 grid test requires UMW to operate on a considerably more complex network and thus incurs more overhead to the SDN controller for computationally intensive routing tasks. Nevertheless, UMW is not undermined in its efficiency and reproduces the favorable results in terms of



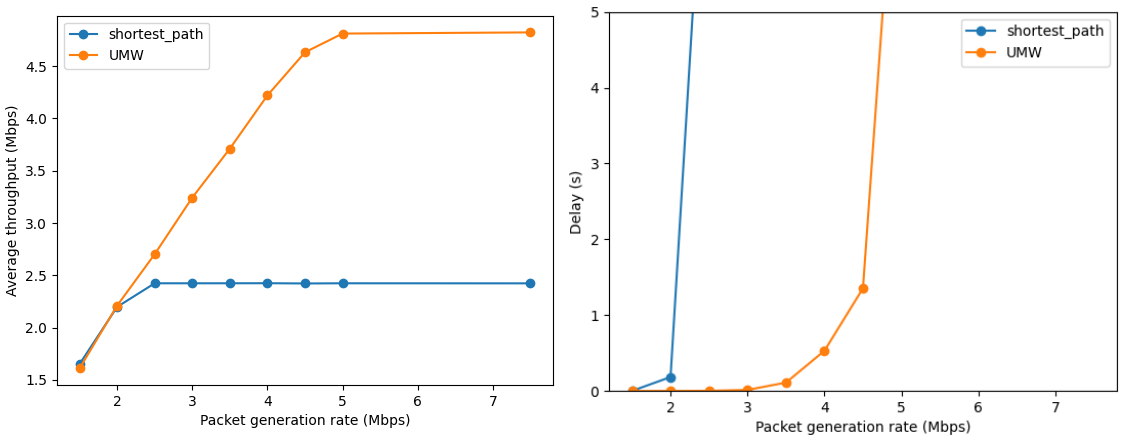
(a) Average throughput

(b) Average packet delay

Figure 2-8: Test on NSF topology with single s-d pair (1, 14)

both throughput and delay.

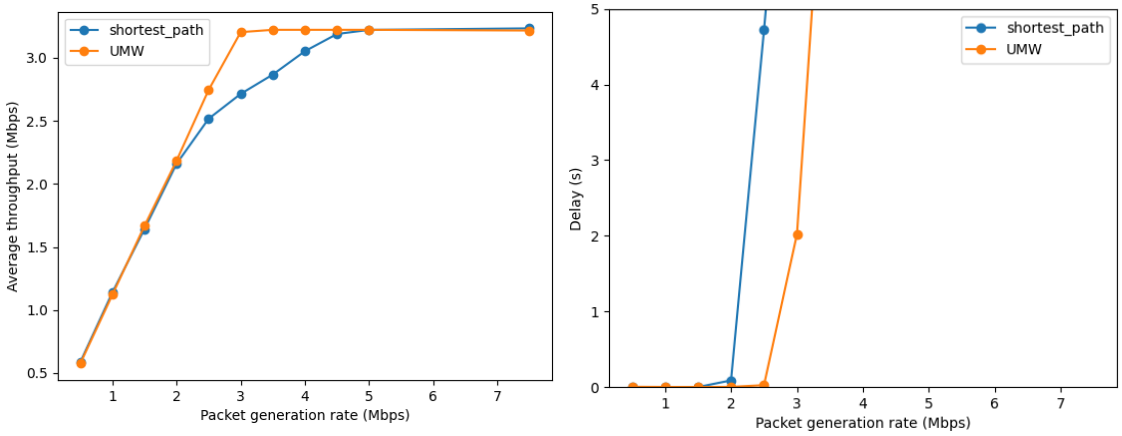
Routing non-shortest paths for optimal throughput: In the above 3×3 grid test (and similarly the 6×6 case), throughput optimality can be achieved for example, by routing along two separate paths $1 \to 4 \to 7 \to 8 \to 9$ and $1 \to 2 \to 3 \to 6 \to 9$, which themselves are shortest paths from 1 to 9. We now consider the test on the NSF topology with single s-d pair (1, 14). Since the bandwidth of every link is 5 Mbps and the destination node 14 has three incoming edges, the maximum throughput is 15 Mbps. However, in order to achieve the full throughput of 15 Mbps, three paths, some of which can be potentially be not shortest, must be deployed. One example is the combination of routes $1 \to 8 \to 9 \to 14$, $1 \to 3 \to 6 \to 13 \to 14$ and $1 \to 2 \to 4 \to 11 \to 14$, where only the first one is a shortest path. Despite the prominence of SP routing in SDN, this setting is meant to show its inherent inability to achieve throughput-optimality. To this end, in Figure 2-8 we illustrate UMW's superior performance achieving close to the optimal rate of 15 Mbps, while SP routing loses two thirds of the capacity, i.e. its maximum rate is only around 5 Mbps. The delay results also show that SP routing leads to dropped packets at 5 Mbps packet generation, in contrast to UMW's serving of all packets with bounded delay.



(a) Average throughput

(b) Average packet delay

Figure 2-9: Test on 3×3 grid with two s-d pairs (1, 3) and (2, 3)



(a) Average throughput

(b) Average packet delay

Figure 2-10: Test on 3×3 grid with three s-d pairs (1, 9), (3, 9) and (7, 9)

2.4.3 Congestion Control by UMW

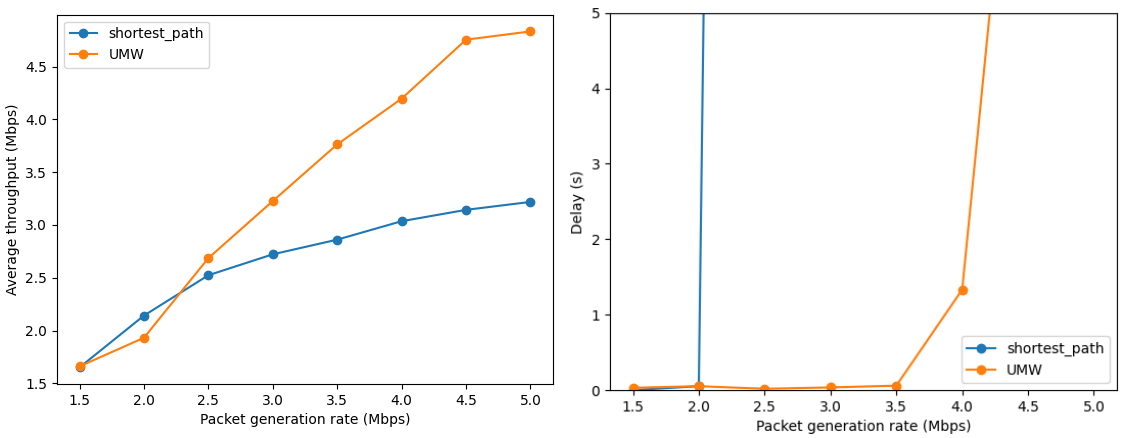
We now move to multiple s-d pairs tests that better capture the congestion phenomenon where different flows compete for the same link. The throughput reported in this Section is averaged among the s-d pairs considered.

Routing around congested links: We consider the 3×3 grid with two s-d pairs (1, 3) and (2, 3). In this scenario, if SP routing is deployed, the link $2 \to 3$ is used by both of the flows $1 \to 2 \to 3$ and $2 \to 3$. This would expectedly drive the incoming rate received at destination node 3 to just 5 Mbps, i.e. exactly the capacity of the congested link $2 \to 3$, and saturate the network at the packet generation rate of $5/2 = 2.5$ Mbps. On the other hand, the maximum supportable rate at node 3 is 10 Mbps, achieved by, for example, prescribing the two s-d pairs with any two separate

paths. The result of this test is given in Figure 2-9, which confirms UMW's mitigation of link congestion and illustrates our aforementioned phenomena. In particular, while SP routing's average throughput is only 2.5 Mbps (total 5 Mbps averaged by 2 s-d pairs), UMW achieves close to optimal average throughput 5 Mbps (total 10 Mbps averaged by 2 s-d pairs). In terms of delay, UMW starts to drop packets at packet generation rate 5 Mbps, while SP routing saturates the network at packet generation rate 2.5 Mbps.

Dynamic routing around congested links: We now consider a more intricate test on the 3×3 grid, yet with three s-d pairs (1, 9), (3, 9) and (7, 9), and report the results in Figure 2-10. Unlike our previous test where static routing could avoid congestion, dynamic routing is required for alleviating congestion in this case. Indeed, since there are three routes (1, 9), (3, 9) and (7, 9) and only two incoming edges of destination node 9, if static routing is deployed, there must be two routes that compete for the same node 9's incoming edge. Given that the capacity of any link is 5 Mbps, any type of static routing would saturate the network at packet generation rate $5/2 = 2.5$ Mbps. On the other hand, an optimal routing scheme should saturate the network at packet generation rate $10/3 \approx 3.33$ Mbps, i.e. when total arrivals at three sources match exactly the receiving capacity of the destination node 9. One example of such optimal routing is that (3, 9) and (7, 9) are routed respectively via $3 \to 6 \to 9$ and $7 \to 8 \to 9$, while (1, 9) is routed by switching with equal probability between $1 \to 4 \to 5 \to 8 \to 9$ and $1 \to 2 \to 5 \to 6 \to 9$. The delay plot also reasserts the above analytical interpretation where SP routing and UMW start to drop packets at packet generation rate 2.5 Mbps and ≈ 3.3 Mbps respectively. In terms of throughput, UMW can also achieve close to average optimal throughput of 3.33 Mbps (10 Mbps averaged by 3 s-d pairs) at its saturation point. Note that though SP routing could also reach the average throughput close to 3.33 Mbps at packet generation rate of 5 Mbps, it is because of network overloading, i.e. links are overloaded and drop packets.

Complex network setting: The next test on NSF topology with three s-d pairs (1, 14), (3, 14) and (7, 14) illustrates a more realistic setting with multiple possibilities for routing. For any s-d pair, there always exists a path that reaches the destination



(a) Average throughput

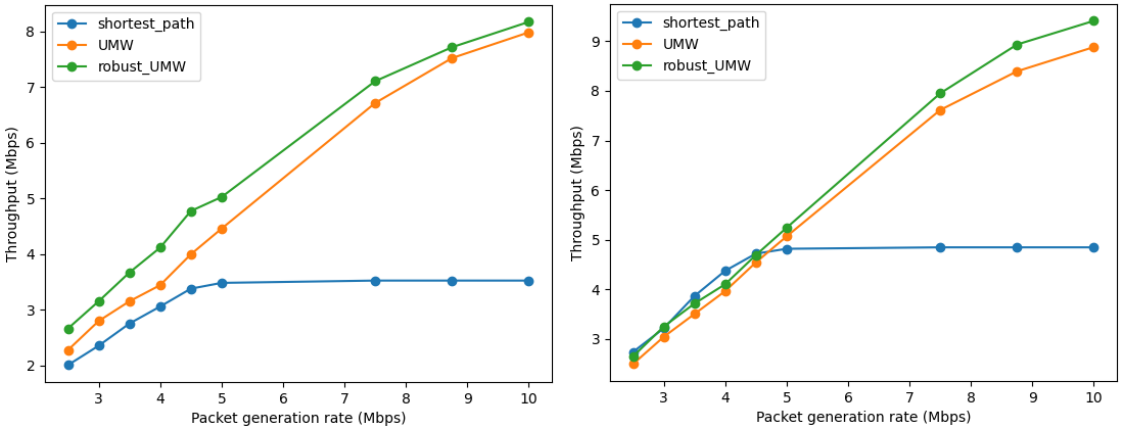
(b) Average packet delay

Figure 2-11: Test on NSF topology with three s-d pairs (1, 14), (3, 14) and (7, 14)

node via any of its three incoming edges. While the 15 Mbps optimal throughput (corresponding to average throughput 5 Mbps) can be achieved at a 5 Mbps packet generation rate, SP routing would congest two links $8 \to 9$ and $9 \to 14$ due to the competing shortest paths of (1, 14) and (7, 14) and thus saturate the network at a packet generation rate of $5/2 = 2.5$ Mbps. The result in Figure 2-11 shows that UMW and SP routing start to drop packets at packet generation rates of 4.5 and 2.5 Mbps respectively. Though UMW loses packets before our expected 5 Mbps point, it still achieves close to optimal 5 Mbps average throughput. This test exhibits UMW's consistency even in challenging network settings.

2.4.4 Link Failure Tolerance

We hereby test the efficiency of our robust mechanism integrated with UMW (details in Section 2.3), termed robust UMW, on a variety of both single s-d pair and multiple s-d pair tests. Since in the presence of failed links packet drops are inevitable, which corresponds to infinite packet delay, we only report the average throughput in the failure tests. On a side note, in these failure tests, the average packet delay among packets successfully served in the system is upperbounded by the average packet delay in non-failure tests, in which UMW has already shown its superior performance.



(a) Average throughput of 3×3 grid test with single s-d pair (1, 9)

(b) Average throughput of 6×6 grid test with single s-d pair (1, 36)

Figure 2-12: Failure tests on grid topologies with single s-d pair

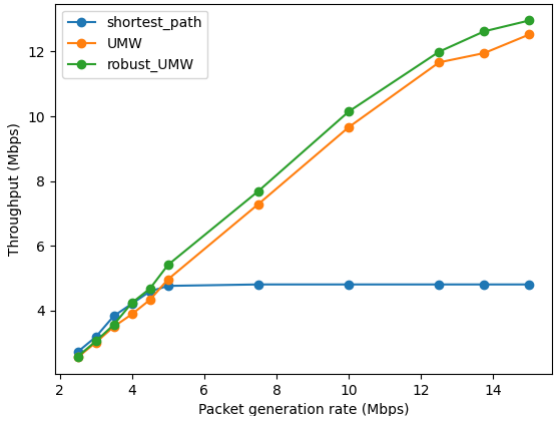


Figure 2-13: Failure test on NSF topology with single s-d pair (1, 14)

To simulate dynamic single link failures, we set the interval between two failed links to follow an exponential distribution with a mean of 30s, and at every time choose a random link to be shut down for 20s.

UMW's utilization of all possible routes: Under the setting of dynamic link failures, we rerun all the single s-d pair tests on 3×3 grid, 6×6 grid and NSF topology as in Section 2.4.2. In particular, the three tests for 3×3 grid, 6×6 grid and NSF topology respectively use the single s-d pairs (1, 9), (1, 36)[2] and (1, 14), and are depicted in Figure 2-12a, 2-12b and 2-13. The robust UMW algorithm consistently improves the throughput of standard UMW in all three tests. The less noticeable improvement of robust UMW in 6×6 grid and NSF topology also exhibits UMW's

[2]Source node 1 is the bottom-left corner and destination node 36 is the top-right corner in the 6×6 grid.

tendency to diversify its routing options. In particular, since these two topologies are more complex than simple 3×3 grid and offer many more routing options, the marginal loss in performance of UMW compared to robust UMW shows that UMW well utilizes all the possible routes and evenly distributes the traffic onto them. Thus, even if a link fails, thereby invalidating a route, UMW only loses a small fraction of the throughput. Such comprehensive route utilization makes UMW, even without any robustness augmentation, inherently good against link failures.

Failure test with multiple s-d pairs: We run the failure tests on 3×3 grid and NSF topologies with multiple s-d pairs. The 3×3 grid test with three s-d pairs (1, 9), (3, 9) and (7, 9) is plotted in Figure 2-14. The NSF topology test with three s-d pairs (1, 14), (3, 14) and (7, 14) is plotted in Figure 2-15. Robust UMW again demonstrates throughput improvement over UMW in both tests except in the case of packet generation rate of 5 Mbps in Figure 2-14. However, we note that at such point the network is highly overloaded ($5 \times 3 = 15$ Mbps arrival rate and only 10 Mbps receiving capacity at node 9), which may cause some minor fluctuation in the results.

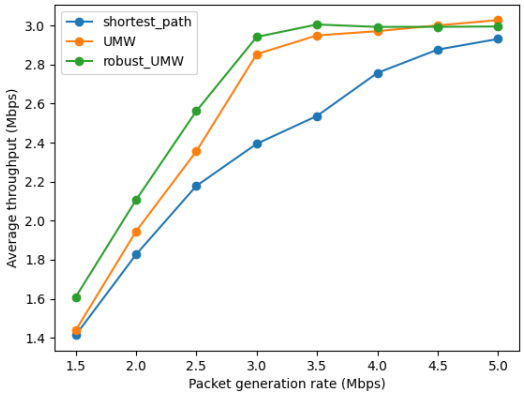


Figure 2-14: Failure test on 3×3 grid with three s-d pairs (1, 9), (3, 9) and (7, 9)

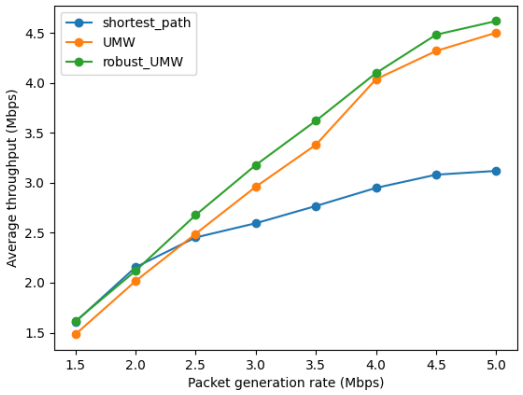


Figure 2-15: Failure test on NSF topology with three s-d pairs (1, 14), (3, 14) and (7, 14)

2.5 UMW with Linear Complexity Scheduling

In the previous Section, we have empirically experimented the basic UMW algorithm for static network on a real SDN system to validate its potential of being the solution for SDN optimal control. Nevertheless, in its original form, the UMW policy incurs significant computational cost for scheduling, which cannot meet the stringent

requirement of computation resources required by many emerging wireless systems of low-power devices. To this end, we propose to integrate the well-known pick-and-compare algorithm [11, 49] with UMW for the practical setting of dynamic network. The core idea is to perform sampling of incrementally "better" scheduling decisions instead of always finding the optimal one, which is computationally prohibitive. Consequently, we present the novel Randomized UMW (RUMW) algorithm for wireless SDN with dynamic topology that is throughput-optimal and incurs linear complexity for scheduling. Since the original pick-and-compare algorithm [11, 49] has considered only the static network setting, RUMW is derived by non-trivially generalizing the algorithmic development and analysis to the setting of dynamic network.

2.5.1 Setting of Dynamic Network

To model time-variation, we consider the ON-OFF model where a link can be in one of the two states, ON or OFF[3]. We denote by $C_e[t]$ the state of link $e \in E$ at time slot t:

$$C_e[t] = \begin{cases} 1, \text{ if } e \text{ is ON at time } t \\ 0, \text{ if } e \text{ is OFF at time } t \end{cases}.$$

For any link subset $E' \subseteq E$, we define $C_{E'}[t] = \{C_e[t]\}_{e \in E'}$ as the vector of links' states of E'. At a given time, the network can be in any configuration $C_E[t] = \alpha$, out of the set of all possible network configurations Ξ. Each element $\alpha \in \Xi$ corresponds to a sub-graph $G(V, E_\alpha) \subseteq G(V, E)$, with $E_\alpha \subseteq E$, denoting the set of links that are ON. The set of all admissible link activations of $G(V, E_\alpha)$ is denoted by $\mathcal{M}_\alpha \in \{0, 1\}^{|E|}$, where 1 denotes that the link is activated and vice versa. The network-configuration process $\{C_E[t]\}_{t \geq 0}$ evolves in discrete-time according to a stationary ergodic process

[3]We use this ON-OFF model for simplicity of presentation. Generalization to more sophisticated models is straightforward (albeit cumbersome).

with the stationary distribution $\{p(\alpha)\}_{\alpha\in\Xi}$, where:

$$\sum_{\alpha\in\Xi} p(\alpha) = 1, \text{ where } p(\alpha) > 0, \forall \alpha \in \Xi.$$

Definition 1 (Capacity Region)**.** *We define the set $\bar{\mathbf{\Lambda}}$ to be the set of all arrival vectors $\boldsymbol{\lambda} \in \mathbb{R}_+^{|\mathcal{C}|}$, for which there exists non-negative scalars $\{\lambda_i^{(c)}\}$, indexed by admissible routes $T_i^{(c)} \in \mathcal{T}^{(c)}$, and $\forall \alpha \in \Xi$ there exists a convex combination of the link activation vectors $\boldsymbol{\mu}^\alpha \in conv(\mathcal{M}_\alpha)$ such that,*

$$\lambda^{(c)} = \sum_{T_i^{(c)}\in\mathcal{T}^{(c)}} \lambda_i^{(c)}, \quad \forall c \in \mathcal{C} \tag{2.5}$$

$$\lambda_e \overset{(def.)}{=} \sum_{(i,c):e\in T_i^{(c)}, T_i^{(c)}\in\mathcal{T}^{(c)}} \lambda_i^{(c)} \leq \sum_{\alpha\in\Xi} p(\alpha)\mu_e^\alpha, \quad \forall e \in E \tag{2.6}$$

Theorem 1. *The network-layer capacity region is characterized by the set $\bar{\mathbf{\Lambda}}$, up to its boundary.*

Proof. Proof of Theorem consists of converse and achievability. The proof of achievability follows from the construction of the policy with linear complexity scheduling in Section 2.5.2 that achieves any arrival rate in the interior of the set $\bar{\mathbf{\Lambda}}$. For the converse, consider any admissible arrival rate vector $\boldsymbol{\lambda}$ supported by some policy π. WLOG, we may assume the policy π to be stationary and the associated DTMC to be ergodic. Let $A_i^{(c)}(t)$ and $A^{(c)}(t)$ be respectively the total number of packets from class c up to time t that have finished their routing along the route $T_i^{(c)}$ and have arrived at the source $s^{(c)}$. We first note that the number of serviced packets of class c is upperbounded by the total arrival of that class, i.e.

$$A^{(c)}(t) \geq \sum_{T_i^{(c)}\in\mathcal{T}^{(c)}} A_i^{(c)}(t) = R^{(c)}(t). \tag{2.7}$$

Dividing both sides of (3.35) by t and taking the limit $t \to \infty$, we obtain that w.p.1:

$$\lambda^{(c)} \overset{(a)}{=} \lim_{t\to\infty} \frac{A^{(c)}(t)}{t} \geq \lim_{t\to\infty} \frac{\sum_{T_i^{(c)}\in\mathcal{T}^{(c)}} A_i^{(c)}(t)}{t} = \lim_{t\to\infty} \frac{R^{(c)}(t)}{t} \overset{(b)}{=} \lambda^{(c)}, \tag{2.8}$$

where (a) is by SLLN, and (d) follows the definition of supportable arrival rate vector $\boldsymbol{\lambda}$. We thus conclude that w.p.1:

$$\lim_{t\to\infty} \frac{\sum_{T_i^{(c)}\in\mathcal{T}^{(c)}} A_i^{(c)}(t)}{t} = \lambda^{(c)}$$

Since π is stationary and the associated DTMC is ergodic, the time-average limits $\lim_{t\to\infty} \frac{1}{t} A_i^{(c)}(t)$ exist a.s.. We consequently define:

$$\lambda_i^{(c)} := \lim_{t\to\infty} \frac{1}{t} A_i^{(c)}(t). \tag{2.9}$$

By (3.37), we have:

$$\lambda^{(c)} = \sum_{T_i^{(c)}\in\mathcal{T}^{(c)}} \lambda_i^{(c)}$$

Now for any edge $e \in E$, the total number of packets that have have finished their routing along the the routes $T_i^{(c)}$ such that $e \in T_i^{(c)}$ is upperbounded by the total service of the edge e, i.e.

$$\sum_{(i,c):e\in T_i^{(c)}, T_i^{(c)}\in\mathcal{T}^{(c)}} A_i^{(c)}(t) \leq \sum_{\tau=1}^{t} \mu_e^{\pi}(\tau). \tag{2.10}$$

We have:

$$\frac{1}{t}\sum_{\tau=1}^{t} \mu_e^{\pi}(\tau) = \sum_{\alpha\in\Xi} \frac{|\{\tau : C[\tau] = \alpha\}|}{t} \cdot \left(\sum_{\tau:\tau\leq t, C[\tau]=\alpha} \frac{1}{|\{\tau : C[\tau] = \alpha\}|} \mu_e^{\pi}(\tau) \right)$$

Let $\mu_e^{\alpha} = \lim_{t\to\infty} \sum_{\tau:\tau\leq t, C[\tau]=\alpha} \frac{1}{|\{\tau:C[\tau]=\alpha\}|} \mu_e^{\pi}(\tau)$. Since $\boldsymbol{\mu}^{\pi}(\tau) \in \mathcal{M}_{\alpha}$ for all $\tau : C[\tau] = \alpha$ and the set $conv(\mathcal{M}_{\alpha})$ is closed, we conclude that $\boldsymbol{\mu}^{\alpha} \in conv(\mathcal{M}_{\alpha})$. Furthermore, we

have $\lim_{t\to\infty} \frac{|\{\tau:C[\tau]=\alpha\}|}{t} = p(\alpha)$, and thus obtain that:

$$\lim_{t\to\infty} \frac{1}{t} \sum_{\tau=1}^{t} \mu_e^{\pi}(\tau) = \sum_{\alpha\in\Xi} p(\alpha)\boldsymbol{\mu}_e^{\alpha}$$

Dividing both sides of (2.10) by t and taking the limit $t \to \infty$ and noting that the left hand side limit exists w.p.1, we obtain that:

$$\sum_{(i,c):e\in T_i^{(c)}, T_i^{(c)}\in\mathcal{T}^{(c)}} \lambda_i^{(c)} \leq \sum_{\alpha\in\Xi} p(\alpha)\mu^{\alpha}$$

□

2.5.2 Randomized UMW (RUMW)

Algorithm 1: Randomized UMW (RUMW) for Dynamic Network

Input: Network topology $G = (V, E)$

for $t = 1, ..., T$ **do**

- Maintain the set of link activation vectors $\{D_E^{\alpha}(m)\}_{\alpha\in\Xi}$.
- Solve the min-cost route selection for every class c packet:

$$T^{(c)}(t) \in \underset{T^{(c)}\in\mathcal{T}^{(c)}}{\operatorname{argmin}} \left(\sum_{e\in E} Q_e(t)\mathbb{1}(e \in T^{(c)}) \right)$$

- Observe the current global NSI $C_E[t] = \alpha$ and obtain a scheduling decision vector $R_E^{\alpha}(m) \in \mathcal{M}_{\alpha}$ uniformly at random.
- **if** $\sum_{e\in E} Q_e(m)R_e^{\alpha}(m) > \sum_{e\in E} Q_e(m)D_e^{\alpha}(m-1)$ **then**
  - Set $D_E^{\alpha}(m) = R_E^{\alpha}(m)$.
- **end**
- **else**
  - Set $D_E^{\alpha}(m) = D_E^{\alpha}(m-1)$.
- **end**
- Set $D_E^{\alpha'}(m) = D_E^{\alpha'}(m-1), \forall \alpha' \neq \alpha$.
- Activate the link schedule $\mu(t) = D_E^{\alpha}(t)$
- Compute the arrivals to links: $A_e(t) = \sum_{c\in\mathcal{C}} A^{(c)}(t)\mathbb{1}(e \in T^{(c)}(t)), \forall e \in E$
- Update the virtual queues: $Q_e(t+1) = \big(Q_e(t) + A_e(t) - \mu_e(t)\big)^+, \forall e \in E$

end

We first establish Theorem 2 that shows the strong stability of the virtual queue

process under the RUMW policy, which is crucial for later proving the rate-stability of the physical queues and thus the throughput-optimality.

Theorem 2. *The RUMW policy (Algorithm 1) strongly stabilizes the virtual queue process, i.e.*

$$\limsup_{K\to\infty} \frac{1}{K} \sum_{t=0}^{K-1} \sum_{e\in E} \mathbb{E}[Q_e(t)] < \infty,$$

Proof. Consider any arrival rate vector $\boldsymbol{\lambda} \in int(\boldsymbol{\Lambda})$. Then there exists some scalar $\epsilon > 0$ and vectors $\boldsymbol{\mu}^\alpha \in conv(\mathcal{M}_\alpha)$ for all $\alpha \in \Xi$, such that we can decompose the total arrival for each class $c \in \mathcal{C}$ into a finite number of routes in the sense of (3.33), and such that:

$$\lambda_e \overset{(def.)}{=} \sum_{(i,c):e\in T_i^{(c)}, T_i^{(c)}\in\mathcal{T}^{(c)}} \lambda_i^{(c)} \leq \sum_{\alpha\in\Xi} p(\alpha)\mu_e^\alpha - \epsilon, \quad \forall e \in E \tag{2.11}$$

From the proof of Theorem 8, there exists some randomized policy **RAND** such that $\mathbb{E}[A_e^{RAND}(t)] = \lambda_e$ and $\mathbb{E}[\mu_e^{RAND}(t)\big|C_E[t-\tau] = \alpha] = \mu_e^\alpha$.

Consider the quadratic Lyapunov function in terms of the virtual queue lengths:

$$L(Q(m)) = Q(m)^T Q(m) = \sum_{e\in E} Q_e(m)^2.$$

From the queue evolution

$$Q_e(m+1) = (Q_e(m) + A_e(m) - \mu_e(m))^+,$$

we obtain that:

$$L(\mathbf{Q}(m+1)) - L(\mathbf{Q}(m)) \leq B + 2\mathbf{Q}(m)^T\mathbf{A}(m) - 2\mathbf{Q}(m)^T\boldsymbol{\mu}(m)$$

where B is a constant bounded by $A_{max}^2 + |E|$. For some fixed T, telescoping gives:

$$L(\mathbf{Q}(m+T)) - L(\mathbf{Q}(m)) \leq BT + 2\sum_{k=0}^{T-1} \mathbf{Q}(m+k)^T \mathbf{A}(m+k) - 2\sum_{k=0}^{T-1} \mathbf{Q}(m+k)^T \boldsymbol{\mu}(m+k)$$

Given some fixed T to be decided later, we define $m_j = j \cdot T$ and consider the T-step Lyapunov drift of $L(.)$ conditioned on the virtual queue lengths as follows:

$$\begin{aligned}
\Delta(m_j) &= \mathbb{E}\big[L(\mathbf{Q}(m_{j+1})) - L(\mathbf{Q}(m_j))\big|\mathbf{Q}(m_j)\big] \\
&\leq BT + 2\mathbb{E}\big[\sum_{k=0}^{T-1} \mathbf{Q}(m_j+k)^T \mathbf{A}(m+k)\big|\mathbf{Q}(m_j)\big] \\
&\quad - 2\mathbb{E}\big[\sum_{k=0}^{T-1} \mathbf{Q}(m_j+k)^T \boldsymbol{\mu}(m_j+k)\big|\mathbf{Q}(m_j)\big] \qquad (2.12)
\end{aligned}$$

Now, at any time slot m, we define $D_E^{*\alpha}(m) = \text{argmax}_{\boldsymbol{\mu}\in\mathcal{M}_\alpha} \sum_{e\in E} Q_e(m)\mu_e$, i.e. $D_E^{*\alpha}(m)$ is the optimal link schedule at time slot m if the global NSI is $C_E[m] = \alpha$. From lines 4, 5 and 6 of Algorithm 1, we obtain that $\forall t > 0$:

$$P(R^\alpha(t) = D_E^{*\alpha}(t)|C_E[t] = \alpha) \geq 1/|\Xi| \geq 2^{-|E|} \qquad (2.13)$$

$$\sum_{e\in E} Q_e(t) D_e^\alpha(t) \geq \sum_{e\in E} Q_e(t) D_e^\alpha(t-1) \qquad (2.14)$$

Now, we define $\forall \alpha \in \Xi$:

$$Z_\alpha = \inf_{k\geq 0}\{k : C_E[m_j+k] = \alpha \text{ and } R^\alpha(m_j+k) = D_E^{*\alpha}(m_j+k)\} \qquad (2.15)$$

Consider any $k \in [0, T-1]$ and assume that $C_E[m_j+k] = \alpha$ for some $\alpha \in \Xi$. If

$Z_\alpha \le k \le T-1$, by (2.14) we have:

$$\begin{aligned}
&\sum_{e\in E} Q_e(m_j+k)D_e^\alpha(m_j+k) \ge \sum_{e\in E} Q_e(m_j+k)D_e^\alpha(m_j+k-1)\\
&\ge \sum_{e\in E} Q_e(m_j+k-1)D_e^\alpha(m_j+k-1) - \|\mathbf{Q}(m_j+k)-\mathbf{Q}(m_j+k-1)\|_1\\
&\ge \sum_{e\in E} Q_e(m_j+k-1)D_e^\alpha(m_j+k-1) - |E|(A_{max}+1).
\end{aligned}$$

Repeating the above inequality, we obtain:

$$\begin{aligned}
\sum_{e\in E} Q_e(m_j+k)D_e^\alpha(m_j+k) &\ge \sum_{e\in E} Q_e(m_j+Z_\alpha)D_e^\alpha(m_j+Z_\alpha) - (k-Z_\alpha)|E|(A_{max}+1)\\
&\ge \sum_{e\in E} Q_e(m_j+Z_\alpha)D_e^\alpha(m_j+Z_\alpha) - T|E|(A_{max}+1)\\
\therefore \sum_{e\in E} Q_e(m_j+k)\mu_e^{\mathbf{RUMW}}(m_j+k) &\ge \sum_{e\in E} Q_e(m_j+Z_\alpha)D_e^\alpha(m_j+Z_\alpha) - T|E|(A_{max}+1)
\end{aligned} \tag{2.16}$$

Recall that (2.16) holds for $k\in[Z_\alpha, T-1]$ and is conditioned on $C_E[m_j+k]=\alpha$. We thus obtain:

$$\begin{aligned}
&\mathbb{E}\big[\sum_{k=0}^{T-1}\mathbf{Q}(m_j+k)^T\boldsymbol{\mu}^{\mathbf{RUMW}}(m_j+k)\big|\{\mathbf{Q}(m_j+k)\}_{k=0}^{T-1},\{C_E[m_j+k]\}_{k=0}^{T-1}\big]\\
=&\mathbb{E}\Big[\sum_{\alpha\in\Xi}\big[\sum_{k:C_E[m_j+k]=\alpha}\mathbf{Q}(m_j+k)^T\boldsymbol{\mu}^{\mathbf{RUMW}}(m_j+k)\big|\{\mathbf{Q}(m_j+k)\}_{k=0}^{T-1},\{C_E[m_j+k]\}_{k=0}^{T-1}\big]\\
\overset{(2.16)}{\ge}&\mathbb{E}\Big[\sum_{\alpha\in\Xi}\big[-T|E|(A_{max}+1)+\mathbf{Q}(m_j+Z_\alpha)^T\boldsymbol{\mu}^{\mathbf{RUMW}}(m_j+Z_\alpha)\big]\\
&\times N(\alpha)\big|\{\mathbf{Q}(m_j+k)\}_{k=0}^{T-1},\{C_E[m_j+k]\}_{k=0}^{T-1}\Big],
\end{aligned} \tag{2.17}$$

where $N(\alpha)$ = "number of k's such that $C_E[m_j+k]=\alpha$ and $k\in[Z_\alpha, T-1]$". By definition of Z_α, we know that $\boldsymbol{\mu}^{\mathbf{RUMW}}(m_j+Z_\alpha) = D_E^{*\alpha}(m_j+Z_\alpha) =$

$\text{argmax}_{\boldsymbol{\mu}\in\mathcal{M}_\alpha}\sum_{e\in E}Q_e(m+Z_\alpha)\mu_e$. We thus bound (2.17) as follows:

$$\begin{aligned}
&\mathbb{E}\big[\sum_{k=0}^{T-1}\mathbf{Q}(m_j+k)^T\boldsymbol{\mu}^{\mathbf{RUMW}}(m_j+k)\big|\{\mathbf{Q}(m_j+k)\}_{k=0}^{T-1},\{C_E[m_j+k]\}_{k=0}^{T-1}\big]\\
\geq&\mathbb{E}\Big[\sum_{\alpha\in\Xi}\big[-T|E|(A_{max}+1)+\mathbf{Q}(m_j+Z_\alpha)^T\boldsymbol{\mu}^\alpha\big]N(\alpha)\big|\{\mathbf{Q}(m_j+k)\}_{k=0}^{T-1},\{C_E[m_j+k]\}_{k=0}^{T-1}\Big]\\
\geq&\mathbb{E}\Big[\sum_{\alpha\in\Xi}\big[-T|E|(A_{max}+1)-Z_\alpha|E|+\mathbf{Q}(m_j)^T\boldsymbol{\mu}^\alpha\big]N(\alpha)\big|\{\mathbf{Q}(m_j+k)\}_{k=0}^{T-1},\{C_E[m_j+k]\}_{k=0}^{T-1}\Big]\\
\geq&\mathbb{E}\Big[\sum_{\alpha\in\Xi}\big[-T|E|(A_{max}+2)+\mathbf{Q}(m_j)^T\boldsymbol{\mu}^\alpha\big]N(\alpha)\big|\{\mathbf{Q}(m_j+k)\}_{k=0}^{T-1},\{C_E[m_j+k]\}_{k=0}^{T-1}\Big]
\end{aligned}$$

Now, taking iterated expectation of the above, we obtain:

$$\mathbb{E}\big[\sum_{k=0}^{T-1}\mathbf{Q}(m_j+k)^T\boldsymbol{\mu}^{\mathbf{RUMW}}(m_j+k)\big|\mathbf{Q}[m_j]\big]\geq\sum_{\alpha\in\Xi}\big[-T|E|(A_{max}+2)+\mathbf{Q}(m_j)^T\boldsymbol{\mu}^\alpha\big]\mathbb{E}[N(\alpha)]. \tag{2.18}$$

The next Lemma establishes the evaluation of $\mathbb{E}[N(\alpha)]$.

Lemma 1. *We have $\forall\alpha\in\Xi$:*

$$\mathbb{E}[N(\alpha)]\geq p(\alpha)T-2^{|E|}+o(T). \tag{2.19}$$

Proof. We first equivalently characterize $N(\alpha)$ from its definition as follows:

$$N(\alpha)=\sum_{k=0}^{T-1}\mathbb{1}(C_E[m_j+k]=\alpha)-\sum_{k=0}^{Z_\alpha-1}\mathbb{1}(C_E[m_j+k]=\alpha) \tag{2.20}$$

Since both $T-1$ and $Z_\alpha-1$ are stopping times, the generalized Wald's identity for finite state Markov process [32] gives:

$$\begin{aligned}
&\mathbb{E}[\sum_{k=0}^{T-1}\mathbb{1}(C_E[m_j+k]=\alpha)]=p(\alpha)T+o(T)\\
&\mathbb{E}[\sum_{k=0}^{Z_\alpha-1}\mathbb{1}(C_E[m_j+k]=\alpha)]=p(\alpha)\mathbb{E}[Z_\alpha]+o(Z_\alpha).
\end{aligned}$$

Taking the expectation of (2.20) in view of the above identities, we obtain that:

$$\mathbb{E}[N(\alpha)] \geq p(\alpha)T - p(\alpha)\mathbb{E}[Z_\alpha] + o(T). \tag{2.21}$$

Furthermore, from (2.13) we have:

$$\begin{aligned}
&P(C_E[m_j+k] = \alpha, R^\alpha(m_j+k) = D_E^{*\alpha}(m_j+k)) \\
=&P(R^\alpha(m_j+k) = D_E^{*\alpha}(m_j+k) \big| C_E[m_j+k] = \alpha) P(C_E[m_j+k] = \alpha) \\
\geq& 2^{-|E|} p(\alpha).
\end{aligned}$$

The above bound and (2.15) imply that:

$$\mathbb{E}[Z_\alpha] \leq 1/(2^{-|E|}p(\alpha)) = 2^{|E|}p(\alpha)^{-1}.$$

Applying the above inequality to (2.21), we conclude the proof of the Lemma. □

Back to the main proof, using Lemma 1, we can bound (2.18) as:

$$\begin{aligned}
&\mathbb{E}\big[\sum_{k=0}^{T-1} \mathbf{Q}(m_j+k)^T \boldsymbol{\mu}^{\mathbf{RUMW}}(m_j+k) \big| \mathbf{Q}[m_j]\big] \\
\geq& \sum_{\alpha \in \Xi} \big[-T|E|(A_{max}+2) + \mathbf{Q}(m_j)^T \boldsymbol{\mu}^\alpha \big] \big(p(\alpha)T - 2^{|E|} + o(T) \big) \\
\geq& -T|E|(A_{max}+2)(T + 2^{|E|}o(T)) + T\mathbf{Q}(m_j)^T \sum_{\alpha \in \Xi} p(\alpha)\boldsymbol{\mu}^\alpha \big(1 - 2^{|E|}p(\alpha)^{-1}T^{-1} + o(1)p(\alpha)^{-1}\big)
\end{aligned} \tag{2.22}$$

Since $\lim_{T\to\infty} \big(2^{|E|}p(\alpha)^{-1}T^{-1} + o(1)p(\alpha)^{-1}\big) = 0$, by taking T large enough in (2.22), we obtain that:

$$\begin{aligned}
&\mathbb{E}\big[\sum_{k=0}^{T-1} \mathbf{Q}(m_j+k)^T \boldsymbol{\mu}^{\mathbf{RUMW}}(m_j+k) \big| \mathbf{Q}[m_j]\big] \\
\geq& -T|E|(A_{max}+2)(T + 2^{|E|}o(T)) + T\mathbf{Q}(m_j)^T \big[\sum_{\alpha \in \Xi} p(\alpha)\boldsymbol{\mu}^\alpha - \epsilon \mathbf{1} \big]
\end{aligned} \tag{2.23}$$

Since we perform UMW min-cost routing (line 3 of Algorithm 1), we have:

$$\begin{aligned}\mathbb{E}\big[\sum_{k=0}^{T-1}\mathbf{Q}(m_j+k)^T\mathbf{A}(m_j+k)\big|\mathbf{Q}(m_j)\big] &\leq \mathbb{E}\big[\sum_{k=0}^{T-1}\mathbf{Q}(m_j+k)^T\boldsymbol{\lambda}\big|\mathbf{Q}(m_j)\big] \\ &\leq B_1T^2+\sum_{k=0}^{T-1}\mathbf{Q}(m)^T\boldsymbol{\lambda} = B_1T^2+T\cdot\mathbf{Q}(m)^T\boldsymbol{\lambda} \\ &\overset{(2.11)}{\leq} B_1T^2+T\mathbf{Q}(m)^T(\sum_{\alpha\in\Xi}p(\alpha)\boldsymbol{\mu}^\alpha-\epsilon),\end{aligned} \tag{2.24}$$

where $B_1 = A_{max} + |E|$. Now plugging (2.23) and (2.24) into (2.12), we obtain:

$$\Delta(m_j) \leq BT + B_1T^2 + T|E|(A_{max}+2)(T+2^{|E|}o(T)) - \epsilon T\|\mathbf{Q}(m_j)\|_1.$$

By Foster's criteria, the virtual queue process are strongly stable. □

Finally, the strong stability of the virtual queue process under RUMW policy is sufficient to establish the throughput-optimality.

Theorem 3. *RUMW is throughput-optimal.*

Proof. The reasoning exactly follows the proof of [46, Theorem 4], whereby RUMW only differs from UMW in the proof of virtual queue process' strong stability as in Theorem 2 above. □

2.6 Chapter Summary

In this chapter, we present the first unified framework, based on the throughput-optimal UMW policy, for optimal network control in wireless SDN. Our framework satisfies the stringent requirements of the SDN architecture, which had hindered the adoption of traditional optimal control schemes such as BP, and can be flexibly applied to any SDN variant. The UMW algorithm can be further augmented to route around failed links in highly dynamic wireless networks. Extensive experiments on a real SDN system based on Mininet backbone illustrate UMW's superior performance compared

to conventional SP routing in terms of throughput, delay and robustness. Finally, we propose a linear complexity scheduling algorithm for dynamic network setting that is suitable for emerging wireless systems of low-power devices. This work can open up directions that adapt the UMW framework to ad-hoc SDN systems with highly dynamic environments [10], or stringent requirements in throughput and delay [22].

Chapter 3

Optimal Control for Distributed Wireless SDN

3.1 Distributed Wireless SDN Architecture

We recall the standard SDN architecture in Figure 3-1, which exemplifies the simplified setting of single domain. In particular, the control plane is decoupled from the data plane, which is in contrast to the traditional network architecture where control logic is embedded in the forwarding hardware.

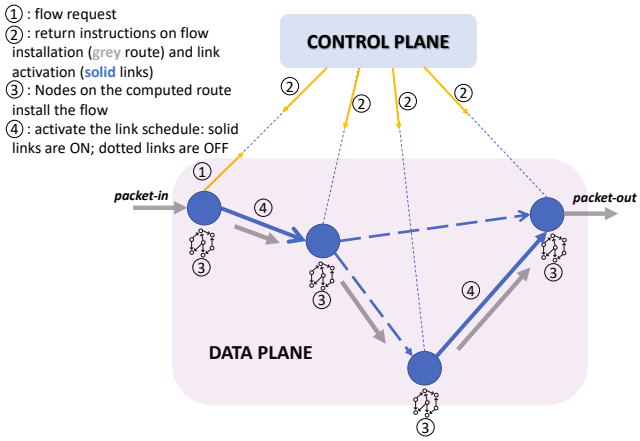


Figure 3-1: Basic SDN architecture and packet life cycle

Distributed wireless SDN has emerged to mitigate the scalability and reliability bottlenecks of standard SDN, and adapt to the modern wireless infrastructure. In

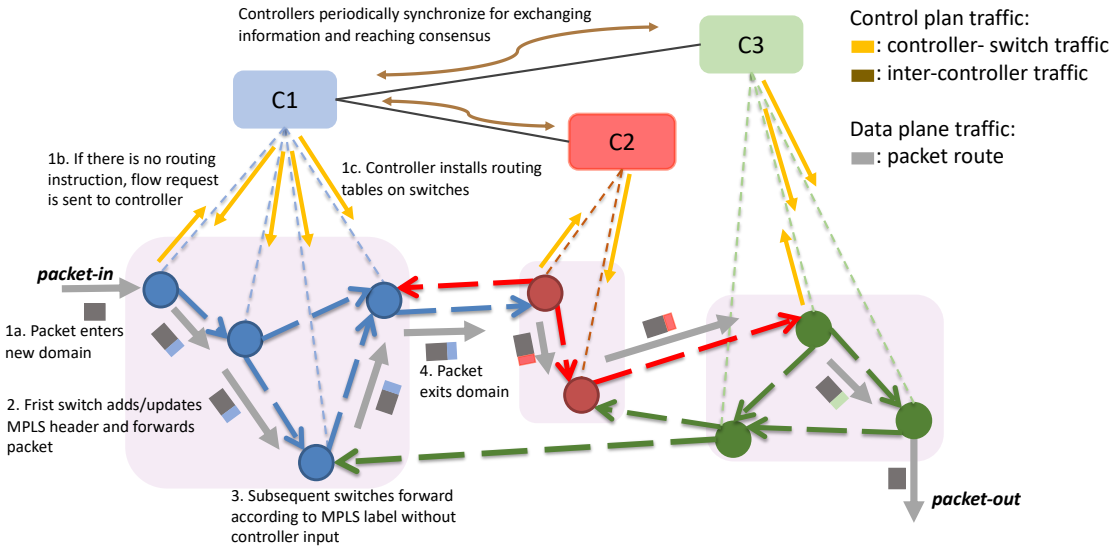


Figure 3-2: Distributed wireless SDN architecture and workflow

this setting, the underlying network topology is decomposed into inter-connected domains, each of which is an independent sub-network and managed by a separate SDN controller. The high-level architecture is depicted in Figure 3-2, which generalizes the basic wireless SDN with the following distinctive features:

Wireless inter-controller communication: The controllers, each of which manages only a sub-network domain, must communicate to exchange their local network information. The inter-controller network is separate from the underlying network infrastructure and is constrained by the operational wireless characteristics: i) the controllers may not have a centralized entity for facilitating coordination, and ii) the inter-controller topology may not be known to the controllers.

Controller synchronization: Unlike that in basic SDN, a controller in distributed wireless SDN only has instantaneous view of its local domain's NSI and statistics. Under the eventual consistency model [51], all the controllers periodically synchronize to maintain the global view of the *delayed* NSI and statistics. Additionally, the synchronization must be designed to accommodate the wireless inter-controller communication.

Inter-domain routing: Whenever a packet enters a new domain, the domain controller is triggered for flow installation if no routing instructions are available, and augments the packet with a MPLS header, which can either be new if the packet joins

the network for the first time, or replace the preceding domain's MPLS header. This process of packet traversal within a domain is summarized in steps 1, 2 and 3 in Figure 3-2. The partial control capability, whereby a controller can only route packets within its domain, makes inter-domain routing especially challenging and inter-dependent with the controller synchronization problem. In Figure 3-2 for example, even though the blue controller (C1) receiving the new packet can compute the optimal path, colored in grey, it must rely on the other controllers (red (C2) and green (C3)) to install the flow in the other domains; however, the blue controller can only reach consensus with other controllers via periodic synchronization.

Inter-domain wireless scheduling: Under the eventual consistency model, since a controller cannot observe the fresh state of certain inter-domain links, i.e. links with end nodes belonging to two different domains, which can interfere with its domain's internal links yet are managed by other controllers, interference is inevitable. It is thus important to design scheduling policies that minimize such interference, thereby maximizing the overall network throughput.

3.2 Preliminaries and Problem Formulation

3.2.1 System Model

Network Infrastructure

The data plane is a multi-hop wireless network with arbitrary topology represented by the directed graph $G = (V, E)$, where V is the set of nodes, i.e. SDN switches, and E is the set of directed point-to-point links. For simplicity, we assume each link has capacity 1. The network operation time is slotted; when there is no confusion, we refer to it as time slot. At any time slot, only certain subsets of links can be activated, according to the wireless interference constraint of the network. An incoming packet belongs to some class $c \in \mathcal{C}$ traffic, which is identified by its source node $s^{(c)} \in V$, the set of its required destination nodes $\mathcal{D}^{(c)} \subseteq V$ and the set $\mathcal{T}^{(c)}$ of all admissible

routes from $s^{(c)}$ to $\mathcal{D}^{(c)}$[1]. An admissible route $T^{(c)} \in \mathcal{T}^{(c)}$ is a tree rooted at the source node $s^{(c)}$ with the set of leaves formed by $\mathcal{D}^{(c)}$. We define the set of distinct classes of incoming traffic as $\mathcal{C}$. Packet arrivals are i.i.d. at every slot. At time slot t, $A^{(c)}(t)$ packets from class c arrive at source node $s^{(c)}$. The mean rate of arrival for class c is $\mathbb{E}[A^{(c)}(t)] = \lambda^{(c)}$. The total number of external packet arrivals to the entire network at any slot t is assumed to be bounded by a finite number A_{max}.

To model time-variation, we consider the ON-OFF model where a link can be in one of the two states, ON or OFF[2]. We denote by $C_e[t]$ the state of link $e \in E$ at time slot t:

$$C_e[t] = \begin{cases} 1, \text{ if } e \text{ is ON at time } t \\ 0, \text{ if } e \text{ is OFF at time } t \end{cases}.$$

For any link subset $E' \subseteq E$, we define $C_{E'}[t] = \{C_e[t]\}_{e \in E'}$ as the vector of links' states of E'. At a given time, the network can be in any configuration $C_E[t] = \alpha \in \{0,1\}^{|E|}$. Each element α corresponds to a sub-graph $\mathcal{G}(V, E_\alpha) \subseteq \mathcal{G}(V, E)$, with $E_\alpha \subseteq E$, denoting the set of links that are ON. The network-configuration process $\{C_E[t]\}_{t \geq 0}$ evolves in discrete-time according to a stationary ergodic process with the stationary distribution $\{p(\alpha)\}_{\alpha \in \{0,1\}^{|E|}}$, where $\sum_{\alpha \in \{0,1\}^{|E|}} p(\alpha) = 1$.

In the distributed wireless SDN setting, m controllers $D_1, D_2, ...,$ and D_m collectively manage the entire network infrastructure. We interchangeably refer to D_i as the i^{th} domain. The underlying network topology $G = (V, E)$ is decomposed into independent and inter-connected domains $G = \cup_{i=1}^{m} G_i$ with $G_i = (V_i, E_i)$, whereby the sub-graph G_i is associated with the controller D_i. The decomposition must satisfy $E_i = \{e = (u, v) \big| u \in V_i\}$, so that any node $u \in V_i$ within the domain of D_i can transmit over links out-going from it. The controllers synchronize every τ time slots, where $\tau_j = j\tau$ is the j^{th} synchronization point. For analytical simplicity, we assume that the global NSI changes at every synchronization point τ_j and remains the same

[1]This captures unicast, multicast or broadcast traffic. In the case of unicast, $D^{(c)}$ corresponds to a single node.

[2]We use this ON-OFF model for simplicity of presentation. Generalization to more sophisticated models is straightforward (albeit cumbersome).

until the next synchronization point, i.e.

$$C_E[\tau_j] = C_E[\tau_j + 1] = ... = C_E[\tau_{j+1} - 1]. \tag{3.1}$$

We assume that the network state is random and can be described as a finite-state Markov chain, i.e.

$$P(C_e[\tau_j]\big|C_e[\tau_{j-1}], ..., C_e[\tau_0]) = P(C_e[\tau_j]\big|C_e[\tau_{j-1}]). \tag{3.2}$$

Furthermore, at time τ_j, any controller D_i gets access to the instantaneous view of its domain's NSI $C_{E_i}[\tau_j]$ and delayed view of the global NSI $C_E[\tau_{j-1}]$, which precedes the current time by τ time slots and is the result of the closest controller synchronization point at time τ_{j-1}. By (3.1) and (3.2), at any time t, the controller D_i always has the fresh local NSI $C_{E_i}[t]$ and delayed global NSI $C_E[t-\tau]$, the dynamics of which can be characterized by the Markov probability $P(C_E[t)]\big|C_E[t-\tau])$.

For the data plane network, we assume a general collision model for wireless inter-switch interference. If two links interfere with each other, simultaneous transmissions over the two links will lead to a collision and no packet will get through. Denote by I_e the set of links that interfere with link $e \in E$ and by $D_e(t)$ the decision of whether to activate link $e \in E$ at time slot t:

$$D_e(t) = \begin{cases} 1, \text{ if } e \text{ is activated at time } t \\ 0, \text{ if } e \text{ is not activated at time } t \end{cases}.$$

For any link subset $E' \subseteq E$, we define $D_{E'}(t) = \{D_e(t)\}_{e \in E'}$ as the link activation vector of E'. Link e successfully transmits a packet at time t if the following conditions hold:

- Link e is ON (i.e. $C_e[t] = 1$) and activated (i.e. $D_e(t) = 1$) at the same time. This is equivalent to $C_e[t] \cdot D_e(t) = 1$.

- No interfering links initiate packet transmission, i.e. $C_{e'}[t] \cdot D_{e'}(t) = 0, \forall e' \in I_e$.

The effective service rate of edge e is thus characterized by:

$$\mu_e(t) = C_e[t] \cdot D_e(t) \prod_{e' \in I_e} (1 - C_{e'}[t] \cdot D_{e'}(t)), \tag{3.3}$$

where $\mu_e(t) = 1$ indicates that link e successfully transmits a packet at time slot t and vice versa. We also assume that the effective service rate is known to the local controller at the end of time slot t. From the system perspective, this can be attained by simply having the nodes actively listen to the channel feedback and then send an acknowledgement to the controllers upon successful packet transmission. In the case that the link is successfully activated for transmission, i.e. $\mu_e(t) = 1$, yet there is no packet backlogged, the sending node can transmit a dummy packet, which will be discarded right upon its reception, to signal the channel for feedback.

As discussed in Section 2.1, a complete algorithm for distributed wireless SDN must also accommodate:

- *SDN routing requirement:* Traditional wireless network routing schemes, including the throughput optimal Back Pressure (BP) policy, admit hop-by-hop routing decisions that are made along the way of packets' traversal. However, the SDN's workflow requires the route per packet to be established immediately upon the packet's arrival at the domain and fixed afterward throughout the packet's intra-domain traversal until exiting the domain.

- *Limited control of physical queues:* Prevalent optimal control schemes such as BP heavily rely on the physical queues of backlogged packets. While operations on physical queues are well supported by traditional networks, SDN's switches lack logical capability for managing the physical queues or distilling the statistics therein.

- *Communication-efficient wireless controller synchronization:* the time allowed for synchronization is bounded by the synchronization period τ.

- *Inter-domain routing:* the controllers must also reach consensus for flow installations via periodic synchronization.

We refer to the above challenges as the SDN *system idiosyncrasies*, which are inter-dependent and additional to the traditional network problem.

Inter-Controller Network

The inter-controller network is separated from the underlying network infrastructure and denoted by $G_c = (V_c, E_c)$, where V_c is the set of controllers and E_c is the set of bi-directional controller-to-controller links. Let $N(D_i)$ be the set of neighbours of controller D_i. We characterize G_c by an $|V_c| \times |V_c|$ graph matrix P such that $P_{ij} > 0$ if $(D_i, D_j) \in E_c$ and $P_{ij} = 0$ otherwise. For technical reason, we assume P is stochastic and has the largest eigenvalue equal 1, while the remaining $n - 1$ eigenvalues are less than 1. Such P always exists, if G_c is connected and nonbipartite. We assume the inter-controller communication to be synchronous and, for generality, operate on the time-scale independent of that of the inter-switch network; the time is divided into frames and, for clarity, we always refer to it as *inter-controller time frame*, which also corresponds to a round of communication. At any inter-controller time frame, each controller is allowed to communicate with only its neighbours. Toward completing certain inter-controller task (e.g. controller synchronization), we characterize the communication complexity by the number of rounds, i.e. inter-controller time frames, and the total number of messages sent over all links during the entire execution time. We assume the multi-port model, whereby each controller can send to or receive from all of its neighbours simultaneously in one inter-controller time frame. The communication complexity analyzed for this model can be translated via sequentialization to the one-port model, where each controller can communicate with at most one controller within a round, with just a multiplicative factor of d^*- maximum node degree of G_c.

3.2.2 Policy Space and Problem Statement

For any decision variable[3], we add the superscript π to acknowledge that it is under the action of the policy π. An admissible policy π, which is mutually deployed by the

[3]These variables include $D_e(t), \mu_e(t)$ and $A_e(t)$, which will be presented later in Section 3.3.2, for any $e \in E$.

m controllers, executes the following actions at every time slot t:

- ROUTE COMPUTATION: Controller D_i computes the route $T^{(c)}(t) \in \mathcal{T}^{(c)}$ for any new packet in class $c \in \mathcal{C}$ that arrives at its domain. All packets in class c arriving at the network in the current time slot are then prescribed such route $T^{(c)}(t)$ throughout their deployment in the network[4].

- SCHEDULING: Based on $C_{E_i}[t]$ and $C_E[t-\tau]$, the controller D_i independently of other controllers activates the link activation vector $D^{\pi}_{E_i}(t)$.

- PACKET TRANSMISSION: Switches transmit packets over the activated links $e \in E_i$ if $D^{\pi}_e(t) = 1$.

We denote by Π the set of all admissible policies under the distributed wireless SDN setting. The set Π includes policies which can use all past and future packet arrival information. Let $R^{(c)}(t)$ be the number of distinct packets of class $c \in \mathcal{C}$ that have reached all of the destination nodes $i \in \mathcal{D}^{(c)}$ by time t. We say that an arrival rate vector $\boldsymbol{\lambda} = \{\lambda^{(c)}\}_{c\in\mathcal{C}}$ is supported by policy π if under the action of π and for any $c \in \mathcal{C}$, the destination nodes commonly receive the distinct packets of class c at the rate of $\lambda^{(c)}$.

Definition 2. *An arrival rate vector $\boldsymbol{\lambda} = \{\lambda^{(c)}\}_{c\in\mathcal{C}}$ is supported by policy π if under the policy π:*

$$\liminf_{t\to\infty} \frac{R^{(c)}(t)}{t} = \lambda^{(c)}, \quad \forall c \in \mathcal{C}, w.p.1 \tag{3.4}$$

Finally, we define the network-layer throughput region $\boldsymbol{\Lambda}$ to be the set of all supportable arrival rate vectors, i.e.

$$\boldsymbol{\Lambda} = \{\boldsymbol{\lambda} \in \mathbb{R}_+^{|\mathcal{C}|} : \exists \pi \in \Pi \text{ that supports } \boldsymbol{\lambda}\} \tag{3.5}$$

[4]This specification on routing policy is meant to accommodate the SDN routing requirement as described in the previous Section.

Definition 3 (Throughput-optimality)**.** *A policy $\pi \in \Pi$ is throughput-optimal if it supports any arrival rate vector $\boldsymbol{\lambda} \in int(\boldsymbol{\Lambda})$, i.e. in the interior of the throughput region.*

In this work, we aim to develop a control scheme for the distributed wireless SDN that is throughput-optimal, and simultaneously satisfies all the SDN system idiosyncrasies.

3.3 Optimal Network Control for Distributed Wireless SDN

In this section, we present a unified optimal network control framework for distributed wireless SDN, termed Distributed Universal Max-Weight (DUMW), and establish its throughput-optimality. The algorithmic development is based on the design of a novel network control policy for the model of interest, and its adaptation to the SDN system idiosyncrasies. To this end, our proposed network control framework DUMW non-trivially extends UMW to support distributed control under the considered dynamic network setting, i.e. inter-domain routing and link scheduling under heterogeneously delayed NSI, while accommodating the wireless controller synchronization. All the proofs in this Section are omitted due to space limitation.

3.3.1 Centralized Universal Max-Weight (UMW)

We first review the Universal Max-Weight (UMW) policy [46] which was shown to be throughput-optimal under a mix of unicast, multicast and broadcast traffic. In particular, UMW permits algorithmic structure directly congruent with SDN routing requirement and leverages easy-to-track virtual queues in place of physical queues. However, while having the potential for being the solution for SDN network control, the UMW policy in its original form is centralized in nature, which is incompatible with inter-domain operations and is not designed to deal with sophisticated dynamic networks.

Setting and Virtual Queue Process of UMW

The original setting of [46] assumes the centralized view of the global NSI which changes every time slot according to some prescribed Markov chain; this is captured by our generalized model (in Section 3.2.1) for the specific case of $m = 1$ domain controller and step size of $\tau = 1$. Under this setting, UMW utilizes the virtual queue process $\mathbf{Q}(t) = \{Q_e(t)\}_{e \in E}$, which relaxes the precedence constraints of multi-hop networks to dynamically route packets and schedule link activations. Unlike the conventional concept of physical queues, in which the queue counter is incremented only when physical packets arrive at the edge in the current time slot [50], the virtual queue of edge e is incremented immediately upon a packet arrival as long as its prescribed route passes through e. Formally, for all $A^{(c)}(t)$ packets of class c arriving at the source node $s^{(c)}$ during time slot t, the UMW policy prescribes them a route $T^{(c)}(t) \in \mathcal{T}^{(c)}$, along which these packets are routed throughout their traversal in the network. The total number of virtual packet arrivals to the virtual queue Q_e at time t is:

$$A_e(t) = \sum_{c \in \mathcal{C}} A^{(c)}(t) \mathbb{1}(e \in T^{(c)}(t)), \quad \forall e \in E. \tag{3.6}$$

Recall from Section 3.2.1 that $D_e(t)$ and $\mu_e(t)$ are respectively the decision variable and the effective service rate of link e. Then the virtual queue process evolves as:

$$Q_e(t+1) = \big(Q_e(t) + A_e(t) - \mu_e(t)\big)^+, \quad e \in E. \tag{3.7}$$

Utilizing the virtual queues, the UMW scheme performs routing and dynamic scheduling on the physical network by solving weighted min-cost and max-weight problems as follows.

Routing: For any class $c \in \mathcal{C}$ packet at time t, select route $T^{(c)}(t) \in \mathcal{T}^{(c)}$ that solves the weighted min-cost problem:

$$T^{(c)}(t) \in \underset{T^{(c)} \in \mathcal{T}^{(c)}}{\operatorname{argmin}} \left(\sum_{e \in E} Q_e(t) \mathbb{1}(e \in T^{(c)}) \right). \tag{3.8}$$

Thus the routing algorithm uses shortest paths based on virtual queue lengths. Upon the arrival of a new packet, its route is immediately computed according to (3.8) and fixed throughout execution. This unique algorithmic structure of UMW makes it compatible with the *SDN routing requirement*.

Scheduling: Denote by $\mathcal{M} \in \{0,1\}^{|E|}$ the set of all admissible link activations. For $\mathbf{x} \in \mathcal{M}$, we have $x_e x_{e'} = 0, \forall e' \in I_e$, i.e. no pair of interfering links can be simultaneously activated. The UMW policy selects the link activation vector $D_E(t) \in \{0,1\}^{|E|}$ that solves the max-weight problem:

$$D_E(t) \in \underset{\mathbf{x}\in\mathcal{M}}{\operatorname{argmax}} \left(\sum_{e\in E} Q_e(t) \cdot C_e[t] \cdot x_e \right). \tag{3.9}$$

Thus the scheduling algorithm activates the schedule of maximum weight using virtual queue lengths. While dynamically routing and scheduling packets via (3.8) and (3.9) to stabilize the virtual queues, UMW transmits packets in the physical network and is guaranteed to achieve the full capacity region [46].

Limitations of UMW

The analytical model of UMW lacks the generality to readily be extended to distributed control. The queue dynamics (3.7) cannot exemplify the distributed view of the network, whereby each controller has only local network statistics and information. Moreover, the scheduling algorithm (3.9) requires the fresh global NSI $C_E[t]$, which is not available to controllers in the distributed setting. The dependence on $\mathcal{M}$ as above also cannot capture the inter-domain characteristics: for example, even if a controller D_i decides to activate link $e \in E$, i.e. $x_e = 1$, it cannot control or even observe the interfering links handled by other domains, i.e. the values of $x_{e'}$ for $e' \in I_e \cap \{E \setminus E_i\}$. On the other hand, our new analytical characterization of effective service rate (3.3) in place of $\mathcal{M}$ captures interference from inter-domain links.

3.3.2 The Virtual Queue Process of Distributed UMW (DUMW)

Next, we develop the Distributed UMW (DUMW) framework that non-trivially extends UMW to the setting of distributed wireless SDN. We hereby define some notations and formally present our generalized virtual queue dynamics. The set $\mathcal{C}$ of packet classes can be decomposed into mutually exclusive sets $\mathcal{C} = \cup_{i=1}^{m} \mathcal{C}_i$ with $\mathcal{C}_i = \{c \in \mathcal{C} : s^{(c)} \in V_i\}$, whereby any packet of class $c \in \mathcal{C}_i$ enters the network through the domain D_i, which manages the source node $s^{(c)}$ of the packet. At any time slot t, a policy π prescribes any packet of class $c \in \mathcal{C}$ an admissible route $T^{(c)}(t) \in \mathcal{T}^{(c)}$. Controller D_i is in charge of packets arriving to its domain, i.e. class-c packets with $c \in \mathcal{C}_i$; consequently, the controller D_i computes the route $T^{(c)}(t)$ for such packets and keeps track of the total virtual packet arrival from the classes $c \in \mathcal{C}_i$:

$$A_e^{\pi i}(t) = \sum_{c \in \mathcal{C}_i} A^{(c)}(t) \mathbb{1}(e \in T^{(c)}), \quad \forall e \in E. \tag{3.10}$$

Summing up over all the domains, we obtain the total virtual packet arrivals from all classes as:

$$A_e^{\pi}(t) = \sum_{i=1}^{m} A_e^{\pi i}(t), \quad \forall e \in E. \tag{3.11}$$

Recall from Section 3.2.1 that $\mu_e^{\pi}(t)$ is the effective service rate of link e. At time t, the virtual queue for link $e \in E$, under the action of policy π, would be incremented by $A_e^{\pi}(t)$ due to the routing decisions, and decremented by $\mu_e^{\pi}(t)$ due to the scheduling decisions. However, since domain controller D_i only has information on $A_e^{\pi i}(t)$ and, if $e \in E_i$, $\mu_e^{\pi}(t)$, the controllers must exchange information to maintain the virtual queues. Moreover, the exchange must be synchronized in order for the controllers to make consistent routing decisions. To model the periodic synchronization, we allow the virtual queues to be updated only at the synchronization points τ_j $(j = 1, 2, ...)$,

and thus obtain the τ-step evolution of the virtual queue process as:

$$Q_e(\tau_{j+1}) = \Bigg(Q_e(\tau_j) + \sum_{t=\tau_j}^{\tau_{j+1}-1} [A_e^{\pi}(t) - \mu_e^{\pi}(t)]\Bigg)^+, \forall e \in E$$

$$\overset{(3.11)}{=} \Bigg(Q_e(\tau_j) + \sum_{t=\tau_j}^{\tau_{j+1}-1} [\sum_{i=1}^{m} A_e^{\pi i}(t) - \mu_e^{\pi}(t)]\Bigg)^+, \forall e \in E. \tag{3.12}$$

It is notable that our virtual queue process generalizes that of [46], whereby for the setting of $m = 1$ domain controller and step size of $\tau = 1$, the recursion (3.12) reduces to the queue dynamics (3.7) of UMW. The DUMW policy is then designed to stabilize the virtual queue process $\{\mathbf{Q}(t)\}_{t\geq 0}$ for any arrival rate vector $\boldsymbol{\lambda} \in int(\boldsymbol{\Lambda})$, which is further shown in Section 3.3.6 to be sufficient for throughput-optimality. In particular, the stability of the virtual queues is obtained by minimizing the τ-step drift of the following quadratic Lyapunov function:

$$L(\mathbf{Q}(t)) = \mathbf{Q}(t)^T \mathbf{Q}(t) = \sum_{e \in E} Q_e(t)^2. \tag{3.13}$$

For any policy π, we consider the τ-step Lyapunov drift of $L(.)$ conditioned on the virtual queue lengths as follows:

$$\Delta^{\pi}(\tau_j) = \mathbb{E}\big[L(\mathbf{Q}(\tau_{j+1})) - L(\mathbf{Q}(\tau_j))\big|\mathbf{Q}(\tau_j)\big], \tag{3.14}$$

where we recall that $\tau_j = j\tau$. Unlike a traditional control problem, the solution for distributed wireless SDN requires the interplay between network control and algorithmic adaptation to the SDN system idiosyncrasies. In the next Section, we present our algorithmic design for controller synchronization that fully addresses the SDN system idiosyncrasies and is the backbone of DUMW's network control operations.

3.3.3 Controller Synchronization and Virtual Queue Estimates

Besides resolving consensus for inter-domain routing, controller synchronization helps maintaining the delayed global NSI and virtual queue updates, all of which are required for making DUMW's dynamic routing and scheduling decisions. At time slot τ_j, the NSI changes to $C_E[\tau_j]$ triggering the new synchronization round. Consequently, each controller D_i executes the SYNC operation (described below) in order to exchange its fresh local NSI $C_{E_i}[\tau_j]$ and local statistics required for virtual queue update, and retrieve the information from the last synchronization point τ_{j-1} (computed by the past SYNC).

Algorithmic Development of SYNC

At the synchronization point τ_j, each controller observes its fresh local NSI $C_{E_i}[\tau_j]$ and has the past local statistics, comprised of virtual arrival packets $\{A_e^{\pi i}(q)\}_{q=\tau_{j-1}}^{\tau_j-1}, \forall e \in E$ and local service rates $\{\mu_e^{\pi}(q)\}_{q=\tau_{j-1}}^{\tau_j-1}, \forall e \in E_i$. Besides exchanging the NSI, the controllers initiate the calculation of $\mathbf{Q}(\tau_j)$ for virtual queue update. From (3.12), this requires the inter-controller global computation of $\sum_{t=\tau_{j-1}}^{\tau_j-1} \mu_e^{\pi}(t)$ and $\sum_{i=1}^{m} \sum_{t=\tau_{j-1}}^{\tau_j-1} A_e^{\pi i}(t)$ for all edges $e \in E$. We first show how the controllers can collaboratively deploy the well-studied MAX gossip protocol [34] for maintaining the (delayed) view of the global NSI $C_e[\tau_j]$ and computing $\sum_{t=\tau_{j-1}}^{\tau_j-1} \mu_e^{\pi}(t)$. At inter-controller time frame k, each controller D_i maintains $c_e^i[k]$ and $s_e^i[k]$ respectively as its estimates of $C_e[\tau_j]$ and $\sum_{t=\tau_{j-1}}^{\tau_j-1} \mu_e^{\pi}(t)$ for all $e \in E$. Based on the NSI and statistics of its local links in E_i, each controller D_i initializes:

$$c_e^i[0] = \begin{cases} C_e[\tau_j] & \text{, if } e \in E_i \\ 0 & \text{, otherwise} \end{cases}, \tag{3.15}$$

$$s_e^i[0] = \begin{cases} \sum_{t=\tau_{j-1}}^{\tau_j-1} \mu_e^{\pi}(t) & \text{, if } e \in E_i \\ 0 & \text{, otherwise} \end{cases}. \tag{3.16}$$

These values represent controller D_i's local information with respect to NSI and service rates. Now, observe that the global NSI $C_e[\tau_j]$ and $\sum_{t=\tau_{j-1}}^{\tau_j-1} \mu_e^\pi(t)$ can be written respectively as:

$$C_e[\tau_j] = \max_{i\in[1,m]} c_e^i[0] \text{ and } \sum_{t=\tau_{j-1}}^{\tau_j-1} \mu_e^\pi(t) = \max_{i\in[1,m]} s_e^i[0], \tag{3.17}$$

since every $c_e^i[0]$ (resp. $s_e^i[0]$) can be either 0 or the true value $C_e[\tau_j]$ (resp. $\sum_{t=\tau_{j-1}}^{\tau_j-1} \mu_e^\pi(t)$). In order to distributedly compute the global maximum, at every inter-controller time frame k, each controller D_i first sends $\mathbf{c}^i[k-1]$ and $\mathbf{s}^i[k-1]$ to all the neighbours $N(D_i)$. After receiving the messages, D_i proceeds to update its estimates as $c_e^i[k] = \max_{h:D_h\in D_i\cup N(D_i)} c_e^h[k-1]$ and $s_e^i[k] = \max_{h:D_h\in D_i\cup N(D_i)} s_e^h[k-1]$ for all $e \in E$, i.e. taking the max of its own value and all other neighbours. It can be shown [34] that, after $O(|V_c|)$ rounds, each controller D_i has $c_e^i[k]$ and $s_e^i[k]$ respectively as the exact values of $C_e[\tau_j]$ and $\sum_{t=\tau_{j-1}}^{\tau_j-1} \mu_e^\pi(t)$. The process incurs $O(|V_c||E_c|)$ messages.

We are now left with the inter-controller computation of $\sum_{i=1}^m \sum_{t=\tau_{j-1}}^{\tau_j-1} A_e^{\pi i}(t)$. Since each controller D_i can locally compute the vector $\mathbf{a}^i \in \mathbb{R}_+^{|E|}$ of partial sums:

$$a_e^i = \sum_{t=\tau_{j-1}}^{\tau_j-1} A_e^{\pi i}(t), \quad \forall e \in E, \tag{3.18}$$

this problem reduces to computing the global sum $\sum_{i=1}^m a_e^i$ across all the m controllers. Unlike the MAX gossip protocol, the summing gossip for wireless setting may incur error to the estimator of the global sum [4]. We purposefully require and thus design the SYNCH operation to ensure that every controller gets the same estimate $\widetilde{A}_e$ of $\sum_{i=1}^m \sum_{t=\tau_{j-1}}^{\tau_j-1} A_e^{\pi i}(t)$ and denote by $\epsilon_e(\tau_j)$ the estimation error, i.e.

$$|\widetilde{A}_e - \sum_{i=1}^m \sum_{t=\tau_{j-1}}^{\tau_j-1} A_e^{\pi i}(t)| \le \epsilon_e(\tau_j), \quad \forall e \in E. \tag{3.19}$$

We further define the following quantities of gossip errors:

$$\epsilon(\tau_j) = \sum_{e \in E} \epsilon_e(\tau_j), \quad \bar{\epsilon}(j) = \sum_{q=1}^{j-1} \epsilon(\tau_q). \tag{3.20}$$

Next, we present our algorithms for the distributed computation of the global sum $\sum_{i=1}^{m} a_e^i$, under two settings: *semi-static wireless environment* and *highly dynamic wireless environment.*

Semi-static wireless environment: We design the mechanism, termed TREE-SUM, that finds the global sum precisely under the semi-static setting where controllers support unique node identities, are capable of coordination, and have access to the inter-controller topology G_c. First, we designate a controller, say D_1, as the root controller and compute a spanning tree T_c, rooted at D_1, of the inter-controller topology G_c. Let T_c have depth d_c and denote by L_i ($i \in [0, d_c]$) be the set of controllers at the i^{th} level. Second, all the controllers accumulate the result until the root of the tree is reached. Formally, for $l = 1 \rightarrow d_c$ iteratively, every controller $D_i \in L_{d_c - l + 1}$ sends $\mathbf{a}^i$ along the tree to its "parent" controller $D_j \in L_{d_c - l}$ in the next level; D_j then adds up all the received values to its original partial values as:

$$a_e^j \leftarrow a_e^j + \sum_{i: D_i \in N(D_j) \cap L_{d_c - l + 1}} a_e^i, \quad \forall e \in E.$$

At the end of the d_c inter-controller time frames, the root controller D_1 will have a_e^1 as the exact value of the global sum $\sum_{i=1}^{m} \sum_{t=\tau_{j-1}}^{\tau_j - 1} A_e^{\pi i}(t)$, which also takes d_c inter-controller time frames. Finally, it broadcasts $\mathbf{a}^1$ along the tree T_c to all the controllers. The total number of communication rounds is $2d_c = O(|V_c|)$, and the total number of messages is $O(|E_c|)$. Moreover, under this mechanism, $\epsilon_e(\tau_j) = 0, \forall e \in E$.

Highly dynamic wireless environment: We now describe our second mechanism, termed GOSSIP-SUM, that finds the global sum approximately under the setting constrained by stringent operational wireless characteristics: i) controllers have limited capability, where no centralized identity or specialized coordination is allowed, and ii) the inter-controller network topology is not known to any of the controllers. The design

of GOSSIP-SUM is comprised of two phases: the *estimation* phase and the *consensus* phase. In the *estimation* phase, all the controllers deploy the Distributed Synchronous Algorithm (DSA) of [4][5] to compute their discrepant estimates of the global sum. Under DSA, each controller D_i maintains $x_e^i[k]$ as its estimate of the true global sum $\sum_{i=1}^{m}\sum_{t=\tau_{j-1}}^{\tau_j-1} A_e^{\pi i}(t)$ at inter-controller time frame k. Denote by $k(\epsilon,\tau_j)$ the number of inter-controller time frames that the DSA runs for the current synchronization point τ_j to achieve arbitrarily small error $\epsilon > 0$ in the sense of Lemma 4. At the end of DSA's last iteration $k(\epsilon,\tau_j)$, every controller D_i separately has $x_e^i[k(\epsilon,\tau_j)]$ as the estimate of the global sum $\sum_{i=1}^{m} a_e^i$. Since the estimators $x_e^1[k(\epsilon,\tau_j)], x_e^2[k(\epsilon,\tau_j)], ..., x_e^m[k(\epsilon,\tau_j)]$ are potentially dissimilar under DSA, the consensus phase next enforces all the controllers to get the same estimate $\widetilde{A}_e$ of $\sum_{i=1}^{m} a_e^i$, which is crucial for inter-domain routing as discussed later in Section 3.3.4. In the consensus phase, the controllers deploy the MAX gossip protocol [34] to mutually obtain $\widetilde{A}_e = \max_{i\in[1,m]}\{x_e^i[k(\epsilon,\tau_j)]\}$ as the same estimate of the global sum $\sum_{i=1}^{m} a_e^i$. The process is similar to deploying the MAX gossip protocol to compute (3.17). Now, to analyze the complexity of GOSSIP-SUM, we present some notations for technical exposition. Recall that the inter-controller network G_c is characterized by the matrix P and has maximum node degree of d^*. We consider the diagonal matrix H with $H_{ii} = \sum_{j=1}^{|V_c|}(P_{ij} + P_{ji})$ and $\bar{d} = \frac{1}{d^*}\big(1 - \frac{1}{2d^*}\big)^{d^*-1}$. Now, we let $W = I - \frac{\bar{d}}{8}H + \frac{\bar{d}}{8}(P + P^T)$ and denote by $\lambda_2(W)$ the second largest eigenvalue of W. The next Theorem establishes the number of inter-controller time frames required for GOSSIP-SUM to achieve the desired estimation error in expectation.

Theorem 4. *For any $\epsilon > 0$, if the DSA in* GOSSIP-SUM *at the synchronization point τ_j runs for $k(\epsilon,\tau_j) = \Theta\big(\log(m\epsilon^{-1})/\log(\lambda_2(W)^{-1})\big)$ inter-controller time frames, then the following bound holds:*

$$\mathbb{E}[\epsilon_e(\tau_j)] \leq \epsilon\tau(m+2)A_{max}, \quad \forall e \in E.$$

Based on Theorem 4, we design GOSSIP-SUM as follows:

[5]The DSA is designed to compute global average, yet it can be equivalently converted to global sum in our case with a multiplicative factor.

Construction 1. *At synchronization point τ_j,* GOSSIP-SUM *runs DSA for $k(j^{-2}, \tau_j)$ inter-controller time frames.*

Lemma 2. GOSSIP-SUM *under Construction 1 satisfies* $\lim_{K\to\infty} \frac{\sum_{j=1}^{K} \mathbb{E}[\bar{\epsilon}(j)]}{K} < \infty$. *Moreover, at synchronization point τ_K, it requires $O(\frac{\log(mK)}{\log(\lambda_2(W)^{-1})} + |V_c|)$ inter-controller time frames and a total of $O(\frac{|E_c|\log(mK)}{\log(\lambda_2(W)^{-1})} + |V_c||E_c|)$ messages.*

The guarantee in Lemma 2 is necessary for establishing the stability of the virtual queue process (needed to prove Theorem 5). The complexity on inter-controller time frames that scales with $O(\log(K))$ is not overly restrictive since the controllers can support much faster time-scale than the switches.

Virtual Queue Estimate

Due to the delayed queue update from the periodic synchronization and the error incurred by the wireless gossip algorithm, the controllers cannot have access to the exact and fresh values of the virtual queues $\mathbf{Q}$ in (3.12); instead, they maintain the potentially inexact estimate $\widetilde{\mathbf{Q}}$. At the synchronization point τ_j, each controller D_i, upon retrieving $C_E[\tau_{j-1}], \boldsymbol{\mu}$ (whereby $\mu_e = \sum_{t=\tau_{j-2}}^{\tau_{j-1}-1} \mu_e^{\pi}(t)$) and $\widetilde{\mathbf{A}}$ (whereby $\widetilde{A}_e = \max_{i\in[1,m]}\{x_e^i[k(\epsilon, \tau_{j-1})]\}$), proceeds to update:

$$\widetilde{Q}_e(\tau_{j-1}) = \left(\widetilde{Q}_e(\tau_{j-2}) + \widetilde{A}_e - \mu_e\right)^+, \quad \forall e \in E. \tag{3.21}$$

At any time $t \in [\tau_j, \tau_{j+1})$, each controller D_i thus has only the estimate of the delayed virtual queues, i.e. $\widetilde{\mathbf{Q}}(\tau_{j-1})$, for making routing and scheduling decisions.

3.3.4 Inter-Domain Routing

In order to address the inter-domain flow installation, we require every controller D_i to compute the min-cost route selection for *any* class $c \in \mathcal{C}$ at time τ_j as:

$$T^{(c)}(\tau_j) \in \underset{T^{(c)}\in\mathcal{T}^{(c)}}{\operatorname{argmin}} \left(\sum_{e\in E} \widetilde{Q}_e(\tau_{j-1}) \cdot \mathbb{1}(e \in T^{(c)}) \right), \tag{3.22}$$

and accordingly install forwarding rules onto its local switches, i.e. nodes $V_i \subseteq V$. This means that even the packets of class $c \notin \mathcal{C}_i$, i.e. not entering the network through D_i, are also supported by D_i for the flow installation on V_i. Since every controller D_i maintains the mutual view of $\widetilde{\mathbf{Q}}(\tau_{j-1})$, they can obtain and solve the same problem (3.22). By imposing the same deterministic tie-breaking rules for optimizing (3.22), we ensure that all the controllers select the same routes $T^{(c)}(\tau_j)$. Moreover, for any time slot $t \in (\tau_j, \tau_{j+1})$, we reuse the old routes computed from the latest synchronization point, i.e. setting $T^{(c)}(t) = T^{(c)}(\tau_j), \forall c \in \mathcal{C}$, thereby requiring *no* flow installation during this period. While our approach incurs some overhead due to repetitive computation, which is inevitable in inter-domain routing [55] and not a barrier in distributed SDN, it fully resolves the consensus problem for inter-domain flow installation and alleviates the controller-switch communication via infrequent flow installation, which is a major scalability bottleneck for large-scale SDN [51].

3.3.5 Inter-Domain Scheduling

Algorithm 2: SCHEDULE

Input: t, D_i, $C_{E_i}[t]$, $C_E[t-\tau]$, $\widetilde{\mathbf{Q}}(\tau_{j-1})$
Output: Link activation vector $D_{E_i}(t) \in \{0,1\}^{|E_i|}$ for domain D_i.

1 Form the view of fresh local NSI as $C_{E_i}[t] = \gamma_{E_i}$ and delayed global NSI as $C_E[t-\tau] = \alpha$.

2 Define k_e such that $e \in V_{k_e}$. Consider the binary vector variable $\mathbf{x} = \{x(e,\xi,\alpha)\}_{e\in E, \xi\in\{0,1\}^{|E_{k_e}|}} \in \{0,1\}^{M_0}$ with $M_0 = \sum_{j=1}^{m} |E_j| 2^{|E_j|}$. Solve the optimization:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmax}} \Big\{ \sum_{\beta\in\{0,1\}^{|E|}} P(C_E[t] = \beta \big| C_E[t-\tau] = \alpha) \times$$
$$\sum_{e\in E} \widetilde{Q}_e(\tau_{j-1}) \beta_e \cdot x(e, \beta_{E_{k_e}}, \alpha) \prod_{e'\in I_e} (1 - x(e', \beta_{E_{k_{e'}}}, \alpha)) \Big\}$$

3 Set $D_e(t) = x^*(e, \gamma_{E_i}, \alpha), \quad \forall e \in E_i$.

4 **Return** the link activation vector $D_{E_i}(t)$

The inter-domain scheduling problem can be characterized as scheduling with heterogeneously delayed NSI. Though sharing similarities with the literature [42], the

system within our interest is more generalized: [42] can be viewed as a special case of our problem where every domain only has one node. On the other hand, we allow every domain to handle an arbitrary set of nodes, thereby imposing the hierarchical domain structure. Consequently, decisions for nodes inside one domain can be inter-dependent in our problem, which is distinctive from [42] where each node makes independent decisions. We hereby present our inter-domain scheduling policy, termed SCHEDULE, in Algorithm 2. At time slot $t \in [\tau_j, \tau_{j+1})$, each controller D_i formulates the same optimization problem from the the common information, which includes the delayed global NSI $C_E[t-\tau]$ and the approximate virtual queue $\widetilde{\mathbf{Q}}(\tau_{j-1})$; under the same deterministic tie-breaking rule, all the controllers then obtain the mutual optimal solution $\mathbf{x}^*$. Given $C_E[t-\tau] = \alpha$, the index $x^*(e, \xi, \alpha) \in \{0, 1\}$ corresponds to the (optimal) decision of whether to activate link $e \in E$ if the fresh local NSI of the domain D_{k_e} managing e is instantaneously observed as $C_{E_{k_e}}[t] = \xi$. Thus, controller D_i, upon observing its fresh local NSI as $C_{E_i}[t] = \gamma_{E_i}$, sets its link activation vector according to $D_e(t) = x^*(e, \gamma_{E_i}, \alpha), \forall e \in E_i$. The SCHEDULE algorithm is novel and the first optimal scheduling policy for the setting of heterogeneously delayed NSI *with* hierarchy. Under this setting, the only applicable algorithm in the literature [58][6] does not leverage fresh local NSI and is thus sub-optimal.

3.3.6 The DUMW Framework and Throughput-Optimality

We depict the full DUMW framework in Algorithm 3, which combines all the technical operations described previously, and proceeds to establish the throughput-optimality of DUMW.

Theorem 5. *DUMW is throughput-optimal.*

Theorem 5 is derived by first minimizing an upperbound of (3.14) and deploying the Lyapunov drift analysis in order to show that the virtual queue process under DUMW is strongly stable for any arrival rate $\boldsymbol{\lambda} \in int(\boldsymbol{\Lambda})$, i.e. $\limsup_{K\to\infty} \frac{1}{K} \sum_{j=0}^{K-1} \sum_{e\in E} \mathbb{E}[Q_e(\tau_j)] <$

[6] We can adapt [58] to the UMW framework by using virtual queues instead of physical queues; this will result in the same achievable throughput region.

Algorithm 3: Distributed **UMW** (**DUMW**) framework

for $t = 1, ..., T$ *each domain* D_i **do**
  if $t = \tau_j \in \mathcal{T} = \{\tau_1, \tau_2, ..., \tau_K\}$ **then**
    Initialize $\mathbf{c}^i[0], \mathbf{s}^i[0], \mathbf{a}^i[0]$ as in (3.15), (3.16), (3.18)
    [Synchronization] Start the new round τ_j and retrieve from the previous round τ_{j-1}: $(C_E[\tau_{j-1}], \boldsymbol{\mu}, \widetilde{\mathbf{A}}) = \text{SYNC}\big(\tau_j, D_i, \mathbf{c}^i[0], \mathbf{s}^i[0], \mathbf{a}^i[0]\big)$
    Update the (approximate) virtual queues as (3.21).
    [Flow installation] Solve for $T^{(c)}(\tau_j), \forall c \in \mathcal{C}$ from (3.22) and install all such flows on $V_i \subseteq V$.
  end
  [Routing] Reuse routes $T^{(c)}(t) = T^{(c)}(\tau_j), \forall c \in \mathcal{C}$.
  [Scheduling] Activate the link activation vector: $D_{E_i}(t) = \text{SCHEDULE}(t, D_i, C_{E_i}[t], C_E[t-\tau], \widetilde{\mathbf{Q}}(\tau_{j-1}))$
end

∞. The key components of the proof leverage the bounded queue delay, the results on gossip error (Theorem 4), and the optimality of our novel scheduling algorithm SCHEDULE to obtain the strong stability of virtual queues, which then provably implies the stability of physical queues and thus the throughput-optimality.

3.4 Numerical Simulation

In all simulations, we report the total average physical queue, which differs from the virtual queue used by DUMW and illustrates the number of packets backlogged in the system. Unless specified otherwise, the DUMW used in our tests deploys the TREE-SUM mechanism. We consider the centralized UMW (Section 3.3.1) as an *unrealistic* baseline. In all experiments, our setting assumes fully-connected inter-controller topology G_c, node-exclusive wireless interference constraints [6], unit link capacity, and unicast traffic, which is specified by a source-destination (s-d) pair assuming Poisson arrivals with the same packet generation rate λ. For abbreviation, we denote the link statistics by $P_e(a|b) = P(C_e[t] = a | C_e[t-\tau] = b)$.

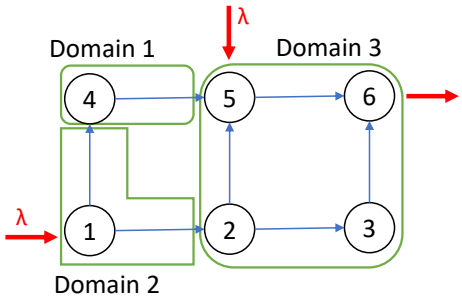


Figure 3-3: The tested 2×3 grid is decomposed into three domains.

DUMW on Dynamic Network

Consider the 2×3 grid decomposed into three domains, as in Figure 3-3, with two s-d pairs (1, 6) and (5, 6), and $\tau = 50$. We explore the throughput capacity of DUMW on the dynamic network with $P_e(1|0) = P_e(1|1) = 0.5$. The only applicable algorithm in the literature [58] only uses the delayed global NSI $C_E[t - \tau]$ for scheduling and is thus sub-optimal in throughput. As illustrated in Figure 3-4, DUMW gains noticeable throughput improvement compared to the literature by leveraging the fresh local NSI $C_{E_i}[t]$.

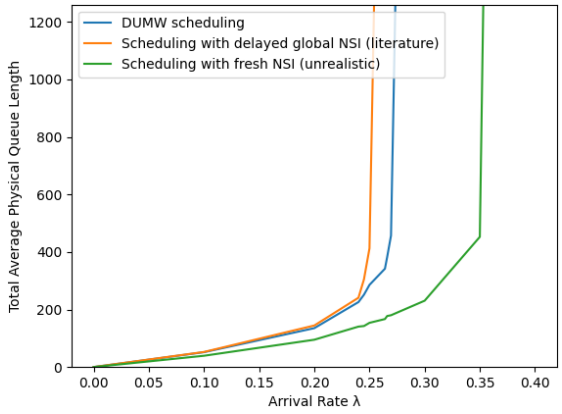


Figure 3-4: DUMW notably gains throughput by leveraging fresh local NSI.

DUMW versus Centralized Controller

For a fair comparison with centralized UMW, we test DUMW (now with both TREE-SUM and GOSSIP-SUM mechanisms) on the same network with reliable links $P_e(1|0) = P_e(1|1) = 1$, since in this setting DUMW also knows the fresh global NSI $C_E[t]$. Results in Figure 3-5 illustrate that DUMW achieves the same throughput

capacity as the centralized controller. The higher physical queue value of DUMW is merely due to the fact that DUMW performs distributed control to reduce the communication overhead.

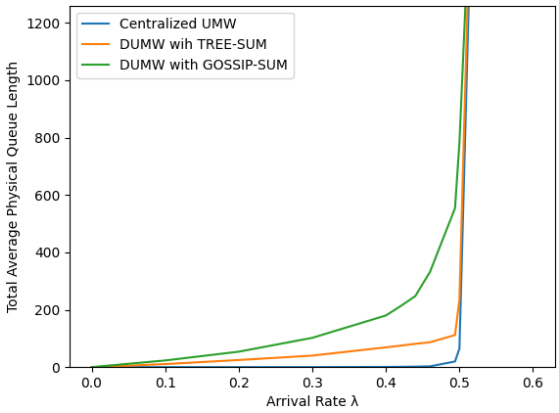


Figure 3-5: DUMW gets the same throughput region as the centralized controller.

Scalability of DUMW

From Section 3.3.4, DUMW drastically alleviates the controller-switch communication via infrequent flow installation with period τ. We experiment on the 9×9 grid (81 nodes and 144 edges) with reliable links and three s-d pairs, whereby three corner nodes send to the last corner node, and high arrival rate $\lambda = 0.3$ (the capacity is 0.33). Figure 3-6 illustrates the system stability and the tradeoff between physical queue values, i.e. estimates of packet delay, and different levels of τ. We note that the queues remain stable even with large values of τ, implying the scalability of DUMW.

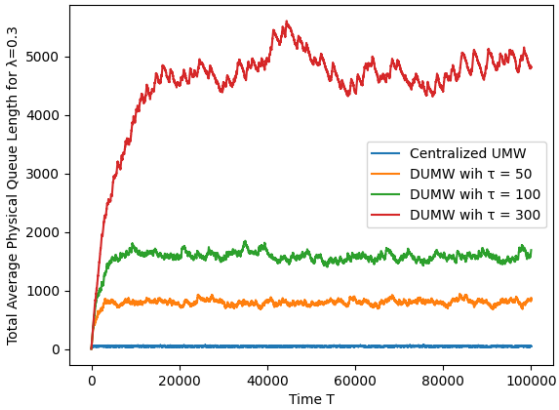


Figure 3-6: Moderately large-scale network test with infrequent flow installation.

3.5 Chapter Summary

In this chapter, we propose the first network control framework for distributed wireless SDN. The control framework is throughput-optimal and alleviates the communication overhead, which has been a major bottleneck for large-scale SDN. Extensive experiments confirm throughput-optimality and favorable scalability of the algorithm.

3.6 Chapter Appendix

3.6.1 Proof of Theorem 4

In the estimation phase of the GOSSIP-SUM mechanism for the synchronization point τ_j, each controller D_i sets $x_e^i[0] = m \cdot \sum_{t=\tau_{j-1}}^{\tau_j - 1} A_e^{\pi i}(t)$ and follows the DSA to maintain $x_e^i[k]$ as the estimate of the true global sum $x_{global} = \sum_{i=1}^{m} \sum_{t=\tau_{j-1}}^{\tau_j - 1} A_e^{\pi i}(t)$ at inter-controller time frame k. Now we present the following adapted [4, Theorem 5][7] that provides the guarantee of DSA.

Theorem 6. *At the end of inter-controller time frame k in the estimation phase of* GOSSIP-SUM*, we have:*

$$P\left(\frac{\sqrt{\sum_{i=1}^{m} [x_e^i[k] - x_{global}]^2}}{\sqrt{\sum_{i=1}^{m} x_e^i[0]^2}} \leq \gamma_1 \right) \geq 1 - \epsilon$$

for $k = \Theta(\frac{\log \epsilon^{-1} + \log \gamma_1^{-1}}{\log \lambda_2(W)^{-1}})$.

From Theorem 6, we establish the following supplementary Lemma.

Lemma 3. *At the end of inter-controller time frame k in the estimation phase of* GOSSIP-SUM*, we have the following guarantee for the global sum estimator of any controller D_i:*

$$P\left(x_e^i[k] \in [x_{global} \cdot (1 - \epsilon), x_{global} \cdot (1 + \epsilon)] \right) \geq 1 - \epsilon$$

[7] The DSA is designed to compute global average, yet it can be equivalently converted to global sum in our case with a multiplicative factor of m in the initialization.

for $k = \Theta(\frac{\log(m \cdot \epsilon^{-1})}{\log \lambda_2(W)^{-1}})$.

Proof. For the $k = \Theta(\frac{\log \epsilon^{-1} + \log \gamma_1^{-1}}{\log \lambda_2(W)^{-1}})$ as defined in Theorem 6, we have:

$$P\left(\sqrt{\sum_{i=1}^{m}[x_e^i[k] - x_{global}]^2} \leq \gamma_1 \sqrt{\sum_{i=1}^{m} x_e^i[0]^2}\right) \geq 1 - \epsilon.$$

For any $i \in [1, m]$, we have:

$$|x_e^i[k] - x_{global}| \leq \sqrt{\sum_{i=1}^{m}[x_e^i[k] - x_{global}]^2}.$$

Furthermore, we have:

$$\sqrt{\sum_{i=1}^{m} x_e^i[0]^2} \leq \sum_{i=1}^{m} x_e^i[0] = m \cdot x_{global}.$$

Combining all the above, we obtain that:

$$P\left(x_e^i[k] \in [x_{global} \cdot (1 - m \cdot \gamma_1), x_{global} \cdot (1 + m \cdot \gamma_1)]\right) \geq 1 - \epsilon.$$

By choosing $\gamma_1 = \epsilon/m$, we obtain the final result. □

Back to the main proof, we proceed to bound the gossip error in expectation. Since $x_e^i[0] = m \cdot \sum_{t=\tau_{j-1}}^{\tau_j - 1} A_e^{\pi i}(t) \leq m \cdot (\tau_j - \tau_{j-1}) A_{max} = m \cdot \tau A_{max}$ and each domain D_i only averages itself with the neighbours during the estimation phase of GOSSIP-SUM, we obtain that $x_e^i[k] \leq m \cdot \tau A_{max}$ for any k. In the consensus phase, after running the DSA for $k(\epsilon, \tau_j)$ inter-controller time frames in the estimation phase, the controllers use $\widetilde{A}_e = \max_{i \in [1,m]}\{x_e^i[k(\epsilon, \tau_j)]\}$ as the mutual estimate of the true global sum. Consequently, we have:

$$\widetilde{A}_e \leq m \cdot \tau A_{max}. \tag{3.23}$$

Now, note that:

$$x_{global} = \sum_{i=1}^{m} \sum_{t=\tau_{j-1}}^{\tau_j - 1} A_e^{\pi i}(t) = \sum_{t=\tau_{j-1}}^{\tau_j - 1} \sum_{i=1}^{m} A_e^{\pi i}(t) \leq \tau A_{max} \tag{3.24}$$

We thus can bound the (worst-case) gossip error as:

$$\epsilon_e(\tau_j) = |\widetilde{A}_e - x_{global}| \leq \widetilde{A}_e + x_{global} \overset{(3.23)+(3.24)}{\leq} (m+1)\tau A_{max}. \tag{3.25}$$

From Lemma 3 and the fact that $\widetilde{A}_e = \max_{i \in [1,m]}\{x_e^i[k(\epsilon, \tau_j)]\}$, we have:

$$P\Bigg(\widetilde{A}_e \in [x_{global} \cdot (1-\epsilon), x_{global} \cdot (1+\epsilon)]\Bigg) \geq 1 - \epsilon. \tag{3.26}$$

For abbreviation, we let $p_1 = P(\epsilon_e(\tau_j) \leq \epsilon \cdot \tau A_{max})$. From (3.24) and recalling that $\epsilon_e(\tau_j) = |\widetilde{A}_e - x_{global}|$, we obtain:

$$\begin{aligned} p_1 &\geq P(\epsilon_e(\tau_j) \leq \epsilon \cdot x_{global}) \\ &= P\Bigg(\widetilde{A}_e \in [x_{global} \cdot (1-\epsilon), x_{global} \cdot (1+\epsilon)]\Bigg) \overset{(3.26)}{\geq} 1 - \epsilon. \end{aligned} \tag{3.27}$$

Finally, we bound the expected gossip error as follows:

$$\begin{aligned} \mathbb{E}[\epsilon_e(\tau_j)] &= p_1 \mathbb{E}[\epsilon_e(\tau_j) \big| \epsilon_e(\tau_j) \leq \epsilon \cdot \tau A_{max}] + (1-p_1)\mathbb{E}[\epsilon_e(\tau_j) \big| \epsilon_e(\tau_j) > \epsilon \cdot \tau A_{max}] \\ &\overset{(3.25)}{\leq} p_1 \epsilon \tau A_{max} + (1-p_1)(m+1)\tau A_{max} \\ &= \epsilon \tau A_{max} + (1-p_1)(m+1-\epsilon)\tau A_{max} \\ &\overset{(3.27)}{\leq} \epsilon \tau A_{max} + \epsilon(m+1)\tau A_{max} \\ &= \epsilon \tau (m+2) A_{max} \end{aligned}$$

3.6.2 Proof of Lemma 2

Under Construction 1, we obtain from Theorem 4 that:

$$\sum_{j=1}^{K}\mathbb{E}[\bar{\epsilon}(j)] \overset{(3.20)}{=} \sum_{j=1}^{K}\sum_{q=1}^{j-1}\sum_{e\in E}\mathbb{E}[\epsilon(\tau_q)] \leq |E|\tau(m+2)A_{max}\sum_{j=1}^{K}\sum_{q=1}^{j-1}\frac{1}{q^2}$$
$$\leq |E|\tau(m+2)A_{max}\sum_{j=1}^{K}\frac{\pi^2}{6} = K\cdot|E|\tau(m+2)A_{max}\pi^2/6,$$

where the last inequality follows the result of the well-known Basel problem. Dividing both sides by K and taking the limit $K\to\infty$, we have:

$$\lim_{K\to\infty}\frac{\sum_{j=1}^{K}\mathbb{E}[\bar{\epsilon}(j)]}{K} < \infty.$$

At synchronization point τ_K, the GOSSIP-SUM mechanism is comprised of:

- *The estimation phase:* The DSA is run for $k(K^{-2},\tau_K)$ inter-controller time frames, which incur $\Theta(\frac{\log(m\cdot\epsilon^{-1})}{\log\lambda_2(W)^{-1}})$ communication round. Since in each communication round there are at most $O(|E_c|)$ pairs of controllers and thus number of messages exchanged, the total number of exchanged messages is $O(\frac{|E_c|\log(m\cdot\epsilon^{-1})}{\log\lambda_2(W)^{-1}})$.
- *The consensus phase:* The controllers compute the global maximum via the MAX protocol [34], which incurs $O(|V_c|)$ communication rounds, and $O(|V_c||E_c|)$.

Combining the cost of the above two phases, we get the final complexity of the synchronization point τ_K as $O(\frac{\log(mK)}{\log(\lambda_2(W)^{-1})}+|V_c|)$ inter-controller time frames and a total of $O(\frac{|E_c|\log(mK)}{\log(\lambda_2(W)^{-1})}+|V_c||E_c|)$ messages.

3.6.3 Errors of the queue estimates

From (3.19), we bound the error of our virtual queue estimate as follows:

Lemma 4. *At any time slot τ_j, we have*

$$|\widetilde{Q}_e(\tau_k)-Q_e(\tau_k)| \leq \sum_{q=1}^{k}\epsilon_e(\tau_q), \quad \forall e\in E \tag{3.28}$$

Proof. We prove (3.28) by induction on k.

Base case $k = 0$: Initially, we have $\widetilde{Q}_e(\tau_0) = Q_e(\tau_0) = 0$, so the statement trivially holds.

Inductive step from $k-1$ to k: Recall from (3.21) that:

$$\widetilde{Q}_e(\tau_k) = \left(\widetilde{Q}_e(\tau_{k-1}) + \widetilde{A}_e - \sum_{t=\tau_{j-1}}^{\tau_k - 1} \mu_e^{\pi}(t)\right)^+, \quad \forall e \in E, \tag{3.29}$$

where as in (3.19) the estimate $\widetilde{A}_e$ satisfies:

$$|\widetilde{A}_e - \sum_{i=1}^{m} \sum_{t=\tau_{k-1}}^{\tau_k - 1} A_e^{\pi i}(t)| \leq \epsilon_e(\tau_k), \quad \forall e \in E. \tag{3.30}$$

By inductive hypothesis, we have:

$$\widetilde{Q}_e(\tau_{k-1}) \leq Q_e(\tau_{k-1}) + \sum_{q=1}^{k-1} \epsilon_e(\tau_q), \quad \forall e \in E.$$

Plugging the above to (3.29), we obtain $\forall e \in E$:

$$\begin{aligned}
\widetilde{Q}_e(\tau_k) &\leq \left(Q_e(\tau_{k-1}) + \sum_{q=1}^{k-1} \epsilon_e(\tau_q) + \widetilde{A}_e - \sum_{t=\tau_{j-1}}^{\tau_k - 1} \mu_e^{\pi}(t)\right)^+ \\
&\overset{(3.30)}{\leq} \left(Q_e(\tau_{k-1}) + \sum_{q=1}^{k} \epsilon_e(\tau_q) + \sum_{i=1}^{m} \sum_{t=\tau_{k-1}}^{\tau_k - 1} A_e^{\pi i}(t) - \sum_{t=\tau_{j-1}}^{\tau_k - 1} \mu_e^{\pi}(t)\right)^+ \\
&\leq \left(Q_e(\tau_{k-1}) + \sum_{i=1}^{m} \sum_{t=\tau_{k-1}}^{\tau_k - 1} A_e^{\pi i}(t) - \sum_{t=\tau_{j-1}}^{\tau_k - 1} \mu_e^{\pi}(t)\right)^+ + \sum_{q=1}^{k} \epsilon_e(\tau_q) \\
&= Q_e(\tau_k) + \sum_{q=1}^{k} \epsilon_e(\tau_q).
\end{aligned}$$

With similar reasoning, we can prove that $\widetilde{Q}_e(\tau_k) \geq Q_e(\tau_k) - \sum_{q=1}^{k} \epsilon_e(\tau_q)$. Finally, we conclude that:

$$|\widetilde{Q}_e(\tau_k) - Q_e(\tau_k)| \leq \sum_{q=1}^{k} \epsilon_e(\tau_q), \quad \forall e \in E$$

□

The DUMW policy leverages the estimates of the *delayed* queue values for making decisions. The following Lemma bounds the impact of delayed queue values.

Lemma 5. *We have the following bounds:*

$$\|\mathbf{Q}(\tau_{k-1}) - \mathbf{Q}(\tau_k)\|_1 \leq \tau(A_{max} + |E|), \tag{3.31}$$

$$\|\widetilde{\mathbf{Q}}(\tau_{k-1}) - \mathbf{Q}(\tau_k)\|_1 \leq \bar{\epsilon}(k) + \tau(A_{max} + |E|), \tag{3.32}$$

Proof. From (3.12), we have:

$$|Q_e(\tau_k) - Q_e(\tau_{k-1})| \leq \sum_{t=\tau_{k-1}}^{\tau_k-1} A_e^\pi(t) + \sum_{t=\tau_{k-1}}^{\tau_k-1} \mu_e^\pi(t) \leq \sum_{t=\tau_{k-1}}^{\tau_k-1} A_e^\pi(t) + \tau.$$

Summing up over all $e \in E$ and noting that $\sum_{e\in E} A_e^\pi(t) \leq A_{max}$:

$$\|\mathbf{Q}(\tau_{k-1}) - \mathbf{Q}(\tau_k)\|_1 \leq \tau A_{max} + \tau|E| = \tau(A_{max} + |E|).$$

By triangular inequality, Lemma 4 and the above (3.31), we have:

$$\begin{aligned}\|\widetilde{\mathbf{Q}}(\tau_{k-1}) - \mathbf{Q}(\tau_k)\|_1 &\leq \|\widetilde{\mathbf{Q}}(\tau_{k-1}) - \mathbf{Q}(\tau_{k-1})\|_1 + \|\mathbf{Q}(\tau_{k-1}) - \mathbf{Q}(\tau_k)\|_1 \\ &\leq \sum_{e\in E}\sum_{q=1}^{k-1} \epsilon_e(\tau_q) + \tau(A_{max} + |E|) \\ &\overset{(3.20)}{=} \bar{\epsilon}(k) + \tau(A_{max} + |E|).\end{aligned}$$

□

3.6.4 Characterization of Throughput Region

Recall that Π the set of all admissible policies under the distributed wireless SDN setting. Moreover, we consider Π_s as the set of policies in Π that make *stationary* scheduling decisions. The scheduling of any $\pi \in \Pi_s$ can be further characterized and have additional properties as follows:

- *Stationary scheduling:* At any time t and for any $i \in [1, m]$, given $C_{E_i}[t] = \beta_{E_i} \in \{0,1\}^{|E_i|}$ and $C_E[t-\tau] = \alpha \in \{0,1\}^{|E|}$, the controller D_i activates the link activation vector $\gamma_{E_i} \in \{0,1\}^{|E_i|}$ with certain probability $P(D^\pi_{E_i}(t) = \gamma_{E_i} \big| C_{E_i}[t] = \beta_{E_i}, C_E[t-\tau] = \alpha) := p^\pi(\gamma_{E_i}|\beta_{E_i}, \alpha)$.

- *Independence across domains:* The decisions made by the domains are independent of each other:

$$\begin{aligned} P(D^\pi_E(t) = \gamma \big| C_E[t] = \beta, C_E[t-\tau] = \alpha) &= \prod_{i=1}^m P(D^\pi_{E_i}(t) = \gamma_{E_i} \big| C_E[t] = \beta, C_E[t-\tau] = \alpha) \\ &= \prod_{i=1}^m P(D^\pi_{E_i}(t) = \gamma_{E_i} \big| C_{E_i}[t] = \beta_{E_i}, C_E[t-\tau] = \alpha) \\ &= \prod_{i=1}^m p^\pi(\gamma_{E_i}|\beta_{E_i}, \alpha), \end{aligned}$$

  where the second equality is by the fact that each domain controller D_i relies on only its fresh local NSI $C_{E_i}[t]$ and delayed global NSI $C_E[t-\tau]$ for making scheduling decisions.

Definition 4. *For network state $\alpha \in \Xi$, and policy $\pi \in \Pi_s$, we define:* $S(\alpha, \pi) = \{\mathbb{E}[\mu^\pi_e(t) \big| C[t-1] = \alpha]\}_{e \in E} = \{\mathbb{E}[C_e[t] \cdot D^\pi_e(t) \prod_{e' \in I_e} (1 - C_{e'}[t] \cdot D^\pi_{e'}(t)) \big| C[t-1] = \alpha]\}_{e \in E}$.

Definition 5 (Capacity Region)**.** *We define the set $\mathbf{\Lambda}$ to be the set of all arrival vectors $\boldsymbol{\lambda} \in \mathbb{R}^{|\mathcal{C}|}_+$, for which there exists non-negative scalars $\{\lambda^{(c)}_i\}$, indexed by admissible routes $T^{(c)}_i \in \mathcal{T}^{(c)}$, and $\forall \alpha \in \Xi$ there exists $\boldsymbol{\mu}^\alpha \in conv_{\pi \in \Pi_s}(S(\alpha, \pi))$ such that,*

$$\lambda^{(c)} = \sum_{T^{(c)}_i \in \mathcal{T}^{(c)}} \lambda^{(c)}_i, \quad \forall c \in \mathcal{C} \tag{3.33}$$

$$\lambda_e \overset{(def.)}{=} \sum_{(i,c): e \in T^{(c)}_i, T^{(c)}_i \in \mathcal{T}^{(c)}} \lambda^{(c)}_i \leq \sum_{\alpha \in \Xi} p(\alpha) \mu^\alpha_e, \quad \forall e \in E \tag{3.34}$$

Theorem 7 (Network Capacity)**.** *The network-layer capacity region is characterized by the set $\mathbf{\Lambda}$, up to its boundary.*

Proof. Proof of Theorem consists of converse and achievability. The proof of converse follows Lemma 6, which shows that any admissible arrival rate vector $\boldsymbol{\lambda}$ is inside $\mathbf{\Lambda}$.

The proof of achievability follows Theorem 5 which shows that DUMW achieves any arrival rate in the interior of the set $\mathbf{\Lambda}$. $\square$

Lemma 6. *For any admissible arrival rate vector* $\boldsymbol{\lambda}$, $\forall \alpha \in \Xi$ *there exists* $\boldsymbol{\mu}^{\alpha} \in conv_{\pi \in \Pi_s}(S(\alpha, \pi))$ *that satisfy* (3.33) *and* (3.34).

Proof. Consider any admissible arrival rate vector $\boldsymbol{\lambda}$ supported by some policy π. WLOG, we may assume the policy π to be stationary and the associated DTMC to be ergodic.

Let $A_i^{(c)}(t)$ and $A^{(c)}(t)$ be respectively the total number of packets from class c up to time t that have finished their routing along the route $T_i^{(c)}$ and have arrived at the source $s^{(c)}$. We first note that the number of serviced packets of class c is upperbounded by the total arrival of that class, i.e.

$$A^{(c)}(t) \geq \sum_{T_i^{(c)} \in \mathcal{T}^{(c)}} A_i^{(c)}(t) = R^{(c)}(t). \tag{3.35}$$

Dividing both sides of (3.35) by t and taking the limit $t \to \infty$, we obtain that w.p.1:

$$\lambda^{(c)} \overset{(a)}{=} \lim_{t\to\infty} \frac{A^{(c)}(t)}{t} \geq \lim_{t\to\infty} \frac{\sum_{T_i^{(c)} \in \mathcal{T}^{(c)}} A_i^{(c)}(t)}{t} = \lim_{t\to\infty} \frac{R^{(c)}(t)}{t} \overset{(b)}{=} \lambda^{(c)}, \tag{3.36}$$

where (a) is by SLLN, and (d) follows the definition of supportable arrival rate vector $\boldsymbol{\lambda}$. We thus conclude that w.p.1:

$$\lim_{t\to\infty} \frac{\sum_{T_i^{(c)} \in \mathcal{T}^{(c)}} A_i^{(c)}(t)}{t} = \lambda^{(c)} \tag{3.37}$$

Since π is stationary and the associated DTMC is ergodic, the time-average limits $\lim_{t\to\infty} \frac{1}{t} A_i^{(c)}(t)$ exist a.s.. We consequently define:

$$\lambda_i^{(c)} := \lim_{t\to\infty} \frac{1}{t} A_i^{(c)}(t). \tag{3.38}$$

Plugging (3.38) to (3.37), we obtain that w.p.1:

$$\lambda^{(c)} = \sum_{T_i^{(c)} \in \mathcal{T}^{(c)}} \lambda_i^{(c)},$$

which validates (3.33). It is left to show that (3.34) holds. Now for any edge $e \in E$, the total number of packets that have have finished their routing along the the routes $T_i^{(c)}$ such that $e \in T_i^{(c)}$ is upperbounded by the total service of the edge e, i.e.

$$\sum_{(i,c): e \in T_i^{(c)}, T_i^{(c)} \in \mathcal{T}^{(c)}} A_i^{(c)}(t) \leq \sum_{\tau=1}^{t} \mu_e^{\pi}(\tau)$$

$$\sum_{(i,c): e \in T_i^{(c)}, T_i^{(c)} \in \mathcal{T}^{(c)}} \frac{1}{t} A_i^{(c)}(t) \leq \frac{1}{t} \sum_{\tau=1}^{t} \mu_e^{\pi}(\tau). \tag{3.39}$$

Let $\Xi = \{\alpha \in \{0,1\}^{|E|} : p(\alpha) > 0\}$. The RHS of (3.39) can be further expressed as:

$$\begin{aligned} \frac{1}{t} \sum_{\tau=1}^{t} \mu_e^{\pi}(\tau) &= \sum_{\alpha \in \Xi} \frac{|\{\tau : C[\tau-1] = \alpha\}|}{t} \cdot \left(\sum_{\tau: \tau \leq t, C[\tau-1]=\alpha} \frac{1}{|\{\tau : C[\tau-1] = \alpha\}|} \mu_e^{\pi}(\tau) \right) \\ &= \sum_{\alpha \in \Xi} \hat{P}_t(\alpha) \cdot H_{\alpha,e}(t) \end{aligned} \tag{3.40}$$

where we consequently define:

$$\hat{P}_t(\alpha) = \frac{|\{\tau : C[\tau-1] = \alpha\}|}{t} \tag{3.41}$$

$$H_{\alpha,e}(t) = \sum_{\tau: C[\tau-1]=\alpha}, \frac{1}{|\{\tau : C[\tau-1] = \alpha\}|} \mu_e^{\pi}(\tau), \tag{3.42}$$

and let $H_{\alpha}(t) = \{H_{\alpha,e}(t)\}_{e \in E}$. Since topology state is assumed to evolve according to a finite state and irreducible Markov chain, we obtain that w.p.1:

$$\lim_{t \to \infty} \hat{P}_t(\alpha) = \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=1}^{t} \mathbb{1}\{C[\tau-1] = \alpha\} = p(\alpha). \tag{3.43}$$

We proceed to show $H_\alpha(t) \in conv_{\pi\in\Pi_s}(S(\alpha,\pi))$ when $t\to\infty$. Observe that:

$$\begin{aligned}
H_{\alpha,e}(t) &= \sum_{\tau:C[\tau-1]=\alpha} \frac{1}{\left|\{\tau : C[\tau-1]=\alpha\}\right|}\mu_e^\pi(\tau) \\
&= \sum_{\beta,M\in\{0,1\}^{|E|}} \left(\sum_{\tau:C[\tau-1]=\alpha,C[\tau]=\beta,D^\pi(\tau)=M} \frac{1}{\left|\{\tau : C[\tau-1]=\alpha\}\right|}\mu_e^\pi(\tau)\right) \\
&\overset{(3.3)}{=} \sum_{\beta,M\in\{0,1\}^{|E|}} \left(\sum_{\tau:C[\tau-1]=\alpha,C[\tau]=\beta,D^\pi(\tau)=M} \frac{1}{|\{\tau : C[\tau-1]=\alpha\}|}\cdot\beta_e M_e \prod_{e'\in I_e}(1-\beta_{e'}M_{e'})\right) \\
&= \sum_{\beta,M\in\{0,1\}^{|E|}} \left(\frac{|\{\tau : C[\tau-1]=\alpha, C[\tau]=\beta, D^\pi(\tau)=M\}|}{|\{\tau : C[\tau-1]=\alpha\}|}\cdot\beta_e M_e \prod_{e'\in I_e}(1-\beta_{e'}M_{e'})\right) \\
&= \sum_{\beta,M\in\{0,1\}^{|E|}} \left(\hat{P}_t(D^\pi(\tau)=M, C[\tau]=\beta \big| C[\tau-1]=\alpha)\cdot\beta_e M_e \prod_{e'\in I_e}(1-\beta_{e'}M_{e'})\right),
\end{aligned} \tag{3.44}$$

In the last equation (3.44), we consider the following empirical distributions:

$$\begin{aligned}
\hat{P}_t(D^\pi(\tau)=M, C[\tau]=\beta, C[\tau-1]=\alpha) &= \frac{|\{\tau : C[\tau-1]=\alpha, C[\tau]=\beta, D^\pi(\tau)=M\}|}{t} \\
\hat{P}_t(D^\pi(\tau)=M, C[\tau]=\beta \big| C[\tau-1]=\alpha) &= \frac{|\{\tau : C[\tau-1]=\alpha, C[\tau]=\beta, D^\pi(\tau)=M\}|}{|\{\tau : C[\tau-1]=\alpha\}|} \\
&\overset{(3.41)}{=} \frac{\hat{P}_t(D^\pi(\tau)=M, C[\tau]=\beta, C[\tau-1]=\alpha)}{\hat{P}_t(\alpha)},
\end{aligned} \tag{3.45}$$

where by the fact that $p(\alpha)>0$ for $\alpha\in\Xi$ and (3.43), for t large enough we have $\hat{P}_t(\alpha)>0$, so that (3.45) is well-defined. Since π is stationary and the associated DTMC is ergodic, it well-defines and enforces the existence of $P(D^\pi(\tau)=M\big|C[\tau-1]=\alpha, C[\tau]=\beta)$. Consequently, the following probability also exists and is well-defined:

$$\begin{aligned}
P(D^\pi(\tau)=M, C[\tau-1]=\alpha, C[\tau]=\beta) =& P(C[\tau-1]=\alpha)\cdot P(C[\tau]=\beta\big|C[\tau-1]=\alpha) \\
&\cdot P(D^\pi(\tau)=M\big|C[\tau-1]=\alpha, C[\tau]=\beta).
\end{aligned}$$

We thus obtain w.p.1 that:

$$\begin{aligned}
&\lim_{t\to\infty} \hat{P}_t(D^\pi(\tau) = M, C[\tau] = \beta, C[\tau-1] = \alpha) \\
=&\lim_{t\to\infty} \frac{1}{t}\sum_{\tau=1}^{t} \mathbb{1}\{D^\pi(\tau) = M, C[\tau] = \beta, C[\tau-1] = \alpha\} \\
=&P(D^\pi(\tau) = M, C[\tau] = \beta, C[\tau-1] = \alpha) \qquad (3.46)
\end{aligned}$$

Combining (3.43), (3.45) and (3.46), we conclude that w.p.1:

$$\lim_{t\to\infty} \hat{P}_t(D^\pi(\tau) = M, C[\tau] = \beta \big| C[\tau-1] = \alpha) = P(D^\pi(\tau) = M, C[\tau] = \beta \big| C[\tau-1] = \alpha).$$

Now taking $t \to \infty$ on both sides of (3.44), we obtain that w.p.1:

$$\begin{aligned}
\lim_{t\to\infty} H_{\alpha,e}(t) &= \sum_{\beta, M\in\{0,1\}^{|E|}} \left(P(D^\pi(\tau) = M, C[\tau] = \beta \big| C[\tau-1] = \alpha) \cdot \beta_e M_e \prod_{e'\in I_e} (1 - \beta_{e'} M_{e'}) \right) \\
&= \mathbb{E}\big[C_e[t] \cdot D_e^\pi(t) \prod_{e'\in I_e} (1 - C_{e'}[t] \cdot D_{e'}^\pi(t)) \big| C[t-1] = \alpha\big] \\
&= S(\alpha, \pi)_e, \qquad (3.47)
\end{aligned}$$

where the last line follows Definition 4. Using (3.43) and (3.47), we obtain the limit of (3.40) as $t \to \infty$ w.p.1:

$$\lim_{t\to\infty} \frac{1}{t}\sum_{\tau=1}^{t} \mu_e^\pi(\tau) = \sum_{\alpha\in\Xi} P(\alpha) \cdot S(\alpha, \pi)_e. \qquad (3.48)$$

Applying (3.38) and (3.48) to (3.39) as $t \to \infty$, we obtain w.p.1:

$$\lambda_e \overset{(def.)}{=} \sum_{(i,c): e\in T_i^{(c)}, T_i^{(c)}\in\mathcal{T}^{(c)}} \lambda_i^{(c)} \leq \sum_{\alpha\in\Xi} P(\alpha) \cdot S(\alpha, \pi)_e,$$

which validates (3.34) wherein we take $\boldsymbol{\mu}^\alpha = S(\alpha, \pi)$.

□

3.6.5 Proof of Theorem 5

We first establish Theorem 8 that shows the strong stability of the virtual queue process under the DUMW policy, which is crucial for later proving the rate-stability of the physical queues and thus the throughput-optimality.

Theorem 8. *The DUMW policy (Algorithm 3) strongly stabilizes the virtual queue process, i.e.*

$$\limsup_{K\to\infty} \frac{1}{K} \sum_{j=0}^{K-1} \sum_{e\in E} \mathbb{E}[Q_e(\tau_j)] < \infty,$$

Proof. Consider any arrival rate vector $\boldsymbol{\lambda} \in int(\boldsymbol{\Lambda})$. Then there exists some scalar $\epsilon > 0$ and vectors $\boldsymbol{\mu}^\alpha \in conv_{\pi'\in\Pi_s}(S(\alpha,\pi'))$ for all $\alpha \in \Xi$, such that we can decompose the total arrival for each class $c \in \mathcal{C}$ into a finite number of routes in the sense of (3.33), and such that:

$$\lambda_e \overset{(def.)}{=} \sum_{(i,c):e\in T_i^{(c)}, T_i^{(c)}\in\mathcal{T}^{(c)}} \lambda_i^{(c)} \leq \sum_{\alpha\in\Xi} p(\alpha)\mu_e^\alpha - \epsilon, \quad \forall e \in E \tag{3.49}$$

By the Carathéodory's theorem, every $\boldsymbol{\mu}^\alpha \in conv_{\pi'\in\Pi_s}(S(\alpha,\pi'))$ can be expressed as:

$$\boldsymbol{\mu}^\alpha = \sum_{\pi'\in Y\subseteq\Pi_s} p_{\pi'}^\alpha S(\alpha,\pi'), \tag{3.50}$$

where Y is a finite set of policies in Π_s and the non-negative $p_{\pi'}^\alpha$'s are such that

$$\sum_{\pi'\in Y\subseteq\Pi_s} p_{\pi'}^\alpha = 1. \tag{3.51}$$

Recall the quadratic Lyapunov function in terms of the virtual queue lengths:

$$L(Q(m)) = Q(m)^T Q(m) = \sum_{e\in E} Q_e(m)^2.$$

For any policy π, we consider the τ-step Lyapunov drift of $L(.)$ (recall that $\tau_j = j\cdot\tau$)

conditioned on the virtual queue lengths as follows:

$$\Delta^{\pi}(\tau_j) = \mathbb{E}\big[L(Q(\tau_{j+1})) - L(Q(\tau_j))\big|Q(\tau_j)\big] \tag{3.52}$$

From the virtual queue dynamics, we first have $\forall e \in E$:

$$\begin{aligned} Q_e(\tau_{j+1})^2 &\leq \left(Q_e(\tau_j) + \sum_{t=\tau_j}^{\tau_{j+1}-1}[\sum_{i=1}^{m} A_e^{\pi i}(t) - \mu_e^{\pi}(t)]\right)^2 \\ &= Q_e(\tau_j)^2 + \left(\sum_{t=\tau_j}^{\tau_{j+1}-1}[\sum_{i=1}^{m} A_e^{\pi i}(t) - \mu_e^{\pi k_e}(t)]\right)^2 + 2Q_e(\tau_j)\sum_{t=\tau_j}^{\tau_{j+1}-1}[\sum_{i=1}^{m} A_e^{\pi i}(t) - \mu_e^{\pi}(t)]. \end{aligned}$$

Combining with the following simple bound:

$$\begin{aligned} \left(\sum_{t=\tau_j}^{\tau_{j+1}-1}[\sum_{i=1}^{m} A_e^{\pi i}(t) - \mu_e^{\pi}(t)]\right)^2 &\leq \left(\sum_{t=\tau_j}^{\tau_{j+1}-1}[\sum_{i=1}^{m} A_e^{\pi i}(t) + \mu_e^{\pi}(t)]\right)^2 \\ &\leq \left(\sum_{q=\tau_j}^{\tau_{j+1}-1}[A_{max} + \mu_{max}]\right)^2 = \tau^2(A_{max} + \mu_{max})^2, \end{aligned}$$

we obtain that:

$$Q_e(\tau_{j+1})^2 - Q_e(\tau_j)^2 \leq \tau^2(A_{max} + \mu_{max})^2 + 2Q_e(\tau_j)\sum_{t=\tau_j}^{\tau_{j+1}-1}[\sum_{i=1}^{m} A_e^{\pi i}(t) - \mu_e^{\pi}(t)]$$

Summing up over all edges $e \in E$ and taking the expectation conditioned on $Q(\tau_j)$, we have:

$$\begin{aligned} \Delta^{\pi}(\tau_j) \leq \tau^2|E|(A_{max} + \mu_{max})^2 &+ 2\sum_{t=\tau_j}^{\tau_{j+1}-1}\mathbb{E}\big[\sum_{e\in E} Q_e(\tau_j)(\sum_{i=1}^{m} A_e^{\pi i}(t))\big|Q(\tau_j)\big] \\ &- 2\sum_{t=\tau_j}^{\tau_{j+1}-1}\mathbb{E}\big[\sum_{e\in E} Q_e(\tau_j)\mu_e^{\pi}(t)\big|Q(\tau_j)\big] \end{aligned} \tag{3.53}$$

Construction of the randomized policy RAND: For any class c packet with $s^{(c)} \in D_i$ for some $i \in [1, m]$ (i.e. the packet arrives at node $s^{(c)}$ belonging to domain D_i), the **RAND** policy routes it along the path $T_j^{(c)} \in \mathcal{T}^{(c)}$ with probability $\frac{\lambda_j^{(c)}}{\lambda^{(c)}}$. For

scheduling, in view of (3.50), at every time slot t the **RAND** policy observes the delayed global NSI as $C[t-\tau] = \alpha$ and activates link according to policy π' with probability $p^{\alpha}_{\pi'}$. Note that $A_e^{\mathbf{RAND}i}(t)$ is the number of packets that arrive at domain D_i and are routed through edge e by the **RAND** policy. We obtain that:

$$\mathbb{E}[A_e^{\mathbf{RAND}i}(t)] = \mathbb{E}[\sum_{c\in\mathcal{C}:s^{(c)}\in D_i} A^{(c)}(t)\mathbb{1}(e\in T^{(c)}(t))] = \sum_{(i,c):e\in T_i^{(c)},s^{(c)}\in D_i} \lambda_i^{(c)} := \lambda_e^i$$

From our definition of λ_e^i above, we have:

$$\mathbb{E}[A_e^{\mathbf{RAND}}(t)] = \sum_{i=1}^{m} \lambda_e^i = \lambda_e. \tag{3.54}$$

Under DUMW, each domain D_i finds the routing $T^{(c)}(t)$ that, for $t \in [\tau_j, \tau_{j+1})$, minimizes

$$\Big(\sum_{e\in E} \widetilde{Q}_e(\tau_{j-1})\cdot \mathbb{1}(e\in T^{(c)})\Big), \quad \forall c\in\mathcal{C}_i \tag{3.55}$$

as in (3.22). Consequently, under DUMW, domain D_i minimizes the following objective:

$$\begin{aligned}
&\sum_{c\in\mathcal{C}_i} A^{(c)}(t)\Big(\sum_{e\in E} \widetilde{Q}_e(\tau_{j-1})\cdot \mathbb{1}(e\in T^{(c)})\Big) &(3.56)\\
=&\sum_{e\in E} \widetilde{Q}_e(\tau_{j-1})\Big(\sum_{c\in\mathcal{C}_i} A^{(c)}(t)\mathbb{1}(e\in T^{(c)})\Big)\\
=&\sum_{e\in E} \widetilde{Q}_e(\tau_{j-1})A_e^{\pi i}(t), &(3.57)
\end{aligned}$$

since the separable objective (3.56) is the weighted sum of (3.55) over $c\in\mathcal{C}$. As domain D_i under DUMW minimizes (3.57), we obtain that:

$$\begin{aligned}
&\sum_{e\in E} \widetilde{Q}_e(\tau_{j-1})A_e^{\mathbf{DUMW}i}(t) \le \sum_{e\in E} \widetilde{Q}_e(\tau_{j-1})A_e^{\mathbf{RAND}i}(t)\\
\therefore &\sum_{e\in E} \widetilde{Q}_e(\tau_{j-1})A_e^{\mathbf{DUMW}}(t) \le \sum_{e\in E} \widetilde{Q}_e(\tau_{j-1})A_e^{\mathbf{RAND}}(t), &(3.58)
\end{aligned}$$

where the last inequality follows the identity (3.11). Now using Lemma 5, we obtain:

$$\begin{aligned}\sum_{e\in E}\widetilde{Q}_e(\tau_{j-1})A_e^{\mathbf{DUMW}}(t) &\geq \sum_{e\in E}Q_e(\tau_j)A_e^{\mathbf{DUMW}}(t) - A_{max}\|\widetilde{\mathbf{Q}}(\tau_{j-1}) - \mathbf{Q}(\tau_j)\|_1 \\ &\overset{(3.32)}{\geq} \sum_{e\in E}Q_e(\tau_j)A_e^{\mathbf{DUMW}}(t) - \tau A_{max}(A_{max}+|E|) - A_{max}\bar{\epsilon}(j),\end{aligned} \tag{3.59}$$

and similarly:

$$\begin{aligned}\sum_{e\in E}\widetilde{Q}_e(\tau_{j-1})A_e^{\mathbf{RAND}}(t) &\leq \sum_{e\in E}Q_e(\tau_j)A_e^{\mathbf{RAND}}(t) + A_{max}\|\widetilde{\mathbf{Q}}(\tau_{j-1}) - \mathbf{Q}(\tau_j)\|_1 \\ &\overset{(3.32)}{\leq} \sum_{e\in E}Q_e(\tau_j)A_e^{\mathbf{RAND}}(t) + \tau A_{max}(A_{max}+|E|) + A_{max}\bar{\epsilon}(j).\end{aligned} \tag{3.60}$$

Combining (3.58), (3.59) and (3.60), we obtain that:

$$\sum_{e\in E}Q_e(\tau_j)A_e^{\mathbf{DUMW}}(t) \leq \sum_{e\in E}Q_e(\tau_j)A_e^{\mathbf{RAND}}(t) + 2\tau A_{max}(A_{max}+|E|) + 2A_{max}\bar{\epsilon}(j).$$

Taking expectation conditioned on $\mathbf{Q}(\tau_j)$ and using (3.54), we have:

$$\begin{aligned}&\mathbb{E}\big[\sum_{e\in E}Q_e(\tau_j)A_e^{\mathbf{DUMW}}(t)\big|\mathbf{Q}(\tau_j)\big] \leq \mathbf{Q}(\tau_j)^T\boldsymbol{\lambda} + 2\tau A_{max}(A_{max}+|E|) + 2A_{max}\mathbb{E}[\bar{\epsilon}(j)] \\ \therefore \sum_{t=\tau_j}^{\tau_{j+1}-1}&\mathbb{E}\big[\sum_{e\in E}Q_e(\tau_j)A_e^{\mathbf{DUMW}}(t)\big|\mathbf{Q}(\tau_j)\big] \leq \tau\mathbf{Q}(\tau_j)^T\boldsymbol{\lambda} + 2\tau^2 A_{max}(A_{max}+|E|) + 2\tau A_{max}\mathbb{E}[\bar{\epsilon}(j)].\end{aligned} \tag{3.61}$$

Recall that Π_s is the space of stationary *admissible* policies making decisions independently of the virtual queue lengths. Besides Π_s, we consider $\widetilde{\Pi}_s$ as the space of stationary *admissible* policies making decisions based on not only the NSI (i.e. decisions for domain i^{th} depend on fresh local NSI $C_{E_i}[t]$ and delayed global NSI $C_E[t-\tau]$), but also the queue length information $Q(t)$. Since $\widetilde{\Pi}_s$ is the same as Π_s with the exception that policies in $\widetilde{\Pi}_s$ has additional knowledge in terms of queue

lengths for making decision, the maximum effective service rate achieved by policies in $\widetilde{\Pi}_s$ is better than by policies in Π_s:

$$\begin{aligned}
&\max_{\pi\in\widetilde{\Pi}_s}\mathbb{E}\bigg[\sum_{e\in E}Q_e(\tau_j)\cdot\mu_e^{\pi}(t)\big|C[t-\tau]=\alpha,\mathbf{Q}(\tau_j)\bigg]\\
&\geq\max_{\pi\in\Pi_s}\mathbb{E}\bigg[\sum_{e\in E}Q_e(\tau_j)\cdot\mu_e^{\pi}(t)\big|C[t-\tau]=\alpha,\mathbf{Q}(\tau_j)\bigg]\\
&=\max_{\pi\in\Pi_s}\mathbb{E}\bigg[\sum_{e\in E}Q_e(\tau_j)\cdot\mu_e^{\pi}(t)\big|C[t-\tau]=\alpha\bigg],
\end{aligned}\tag{3.62}$$

where the last line is by the fact that policies in Π_s make decision independently of the virtual queue lengths. Next, from Lemma 9, the DUMW policy satisfies $\forall t\in[\tau_j,\tau_{j+1})$:

$$\begin{aligned}
&\mathbb{E}\bigg[\sum_{e\in E}Q_e(\tau_j)\cdot\mu_e^{\mathbf{DUMW}}(t)\big|C[t-\tau]=\alpha,\mathbf{Q}(\tau_j)\bigg]\\
&\geq\max_{\pi\in\widetilde{\Pi}_s}\mathbb{E}\bigg[\sum_{e\in E}Q_e(\tau_j)\cdot\mu_e^{\pi}(t)\big|C[t-\tau]=\alpha,\mathbf{Q}(\tau_j)\bigg]-2\mathbb{E}[\bar{\epsilon}(j)]-2\tau(A_{max}+|E|).
\end{aligned}\tag{3.63}$$

Combining (3.62) and (3.63), we obtain that:

$$\begin{aligned}
&\mathbb{E}\bigg[\sum_{e\in E}Q_e(\tau_j)\cdot\mu_e^{\mathbf{DUMW}}(t)\big|C[t-\tau]=\alpha,\mathbf{Q}(\tau_j)\bigg]\\
&\geq\max_{\pi\in\Pi_s}\mathbb{E}\bigg[\sum_{e\in E}Q_e(\tau_j)\cdot\mu_e^{\pi}(t)\big|C[t-\tau]=\alpha\bigg]-2\mathbb{E}[\bar{\epsilon}(j)]-2\tau(A_{max}+|E|)\\
&\geq\mathbb{E}\bigg[\sum_{e\in E}Q_e(\tau_j)\cdot\mu_e^{\pi'}(t)\big|C[t-\tau]=\alpha\bigg]-2\mathbb{E}[\bar{\epsilon}(j)]-2\tau(A_{max}+|E|),\quad\forall\pi'\in\Pi_s\\
&=Q(\tau_j)^TS(\alpha,\pi')-2\mathbb{E}[\bar{\epsilon}(j)]-2\tau(A_{max}+|E|),\quad\forall\pi'\in\Pi_s\\
\therefore\;&p_{\pi'}^{\alpha}\mathbb{E}\bigg[\sum_{e\in E}Q_e(\tau_j)\cdot\mu_e^{\mathbf{DUMW}}(t)\big|C[t-\tau]=\alpha,\mathbf{Q}(\tau_j)\bigg]\\
&\geq Q(\tau_j)^T\big(p_{\pi'}^{\alpha}S(\alpha,\pi')\big)-p_{\pi'}^{\alpha}\big[2\mathbb{E}[\bar{\epsilon}(j)]-2\tau(A_{max}+|E|)\big].
\end{aligned}$$

Summing up the above in view of (3.50) and (3.51), we obtain:

$$\mathbb{E}\Big[\sum_{e\in E} Q_e(\tau_j)\cdot \mu_e^{\mathbf{DUMW}}(t)\big|C[t-\tau]=\alpha, \mathbf{Q}(\tau_j)\Big] \geq Q(\tau_j)^T\boldsymbol{\mu}^\alpha - 2\mathbb{E}[\bar{\epsilon}(j)] - 2\tau(A_{max}+|E|)$$

Taking expectation w.r.t. $\alpha\in\Xi$ of the above, we have:

$$\begin{aligned}\mathbb{E}\Big[\sum_{e\in E} Q_e(\tau_j)\cdot \mu_e^{\mathbf{DUMW}}(t)\big|\mathbf{Q}(\tau_j)\Big] &\geq \mathbf{Q}(\tau_j)^T\big(\sum_{\alpha\in\Xi} p(\alpha)\cdot\boldsymbol{\mu}^\alpha\big) - 2\mathbb{E}[\bar{\epsilon}(j)] - 2\tau(A_{max}+|E|)\\ &\overset{(3.49)}{\geq} \mathbf{Q}(\tau_j)^T(\boldsymbol{\lambda}+\epsilon\mathbf{1}) - 2\mathbb{E}[\bar{\epsilon}(j)] - 2\tau(A_{max}+|E|)\end{aligned}$$

Summing up the above over $t=\tau_j\to\tau_{j+1}-1$, we obtain:

$$\sum_{t=\tau_j}^{\tau_{j+1}-1}\mathbb{E}\Big[\sum_{e\in E} Q_e(\tau_j)\cdot \mu_e^{\mathbf{DUMW}}(t)\big|\mathbf{Q}(\tau_j)\Big] \geq \tau\mathbf{Q}(\tau_j)^T(\boldsymbol{\lambda}+\epsilon\mathbf{1}) - 2\tau\mathbb{E}[\bar{\epsilon}(j)] - 2\tau^2(A_{max}+|E|) \tag{3.64}$$

Now applying the bounds (3.61) and (3.64) into (3.53), we have:

$$\begin{aligned}\Delta^{\mathbf{DUMW}}(\tau_j) &\leq \tau^2|E|(A_{max}+\mu_{max})^2 + 2\tau^2(A_{max}+|E|) + 4\tau^2A_{max}(A_{max}+|E|)\\ &\quad + 4\tau A_{max}\mathbb{E}[\bar{\epsilon}(j)] + 2\tau\bar{\epsilon}(j) - 2\tau\epsilon\mathbf{Q}(\tau_j)^T\mathbf{1}\\ &= \tau^2C_1 + (4\tau A_{max}+2\tau)\mathbb{E}[\bar{\epsilon}(j)] - 2\tau\epsilon\|\mathbf{Q}(\tau_j)\|_1,\end{aligned} \tag{3.65}$$

where $C_1 = |E|(A_{max}+\mu_{max})^2+2(A_{max}+|E|)+4A_{max}(A_{max}+|E|)$. Taking expectation on both sides of (3.65) w.r.t the virtual queue lengths $\mathbf{Q}(\tau_j)$, we bound the expected drift as:

$$\mathbb{E}L(\mathbf{Q}(\tau_{j+1})) - \mathbb{E}L(\mathbf{Q}(\tau_j)) \leq \tau^2C_1 + (4\tau A_{max}+2\tau)\mathbb{E}[\bar{\epsilon}(j)] - 2\tau\epsilon\sum_{e\in E}\mathbb{E}[Q_e(\tau_j)]$$

Summing the above from $j=0\to K-1$ and noting that $L(\mathbf{Q}(\tau_K))\geq 0$ and

$L(\mathbf{Q}(\tau_0)) = 0$, we obtain that:

$$\frac{1}{K}\sum_{j=0}^{K-1}\sum_{e\in E}\mathbb{E}[Q_e(\tau_j)] \leq \frac{\tau C_1}{2\epsilon} + \frac{2A_{max}+1}{\epsilon}\cdot\frac{\sum_{j=0}^{K-1}\mathbb{E}[\bar{\epsilon}(j)]}{K}. \tag{3.66}$$

If DUMW is under the error-free TREE-SUM mechanism, then $\frac{\sum_{j=0}^{K-1}\mathbb{E}[\bar{\epsilon}(j)]}{K} = 0$. Else if DUMW is under the GOSSIP-SUM mechanism, then by Lemma 2 we have $\lim_{K\to\infty}\frac{\sum_{j=0}^{K-1}\mathbb{E}[\bar{\epsilon}(j)]}{K} < \infty$. Thus taking lim sup of (3.66), we conclude that:

$$\limsup_{K\to\infty}\frac{1}{K}\sum_{j=0}^{K-1}\sum_{e\in E}\mathbb{E}[Q_e(\tau_j)] < \infty,$$

i.e. the virtual queues are strongly stable under DUMW. □

Back to the main proof, besides the τ-step virtual queue process $\{\mathbf{Q}(\tau_j)\}_{j\geq 0}$, we now also consider 1-step virtual queue process $\{\hat{\mathbf{Q}}(t)\}_{t\geq 0}$ as follows:

$$\hat{Q}_e(t+1) = \left(\hat{Q}_e(t) + A_e^{\pi}(t) - \mu_e^{\pi}(t)\right)^+, \forall e \in E. \tag{3.67}$$

The next Lemma 7 derives the strong stability of $\{\hat{\mathbf{Q}}(t)\}_{t\geq 0}$ from the strong stability of $\{\mathbf{Q}(\tau_j)\}_{j\geq 0}$. Since the 1-step virtual queue process $\{\hat{\mathbf{Q}}(t)\}_{t\geq 0}$ with the dynamics as in (3.67) is exactly the same as that considered in [46], its strong stability immediately implies the throughput-optimality of DUMW following the results from [46].

Lemma 7. *Under the DUMW policy, the virtual queue process $\{\hat{\mathbf{Q}}(t)\}_{t\geq 0}$ is strongly-stable, i.e.*

$$\limsup_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{e\in E}\mathbb{E}[\hat{Q}_e(t)] < \infty.$$

Proof. For any $T > 0$, we consider $T_0 = (T-1) \mod \tau$ and express $T-1 = K\cdot\tau + T_0$

for some integer K. From Lemma 11, we have $\forall j \in [0, K-1]$:

$$\sum_{t=j\tau}^{(j+1)\tau-1} \sum_{e\in E} \mathbb{E}[\hat{Q}_e(t)] \leq \sum_{t=j\tau}^{(j+1)\tau-1} \sum_{e\in E} \big(\mathbb{E}[Q_e(\tau_j)] + \tau A_{max}\big)$$
$$= \tau^2|E|A_{max} + \tau \sum_{e\in E} \mathbb{E}[Q_e(\tau_j)]. \tag{3.68}$$

Noting that $T - 1 - K\tau = T_0 < \tau$, we similarly have:

$$\sum_{t=K\tau}^{T-1} \sum_{e\in E} \mathbb{E}[\hat{Q}_e(t)] \leq \sum_{t=K\tau}^{T-1} \sum_{e\in E} \big(\mathbb{E}[Q_e(\tau_K)] + \tau A_{max}\big)$$
$$\leq \tau^2|E|A_{max} + \tau \sum_{e\in E} \mathbb{E}[Q_e(\tau_{\lfloor (T-1)/\tau \rfloor})]. \tag{3.69}$$

Summing up (3.68) over $j = 0 \to K-1$ and (3.69), we obtain that for $T > 1$:

$$\frac{1}{T}\sum_{t=0}^{T-1} \sum_{e\in E} \mathbb{E}[\hat{Q}_e(t)] \leq \frac{\lfloor (T-1)/\tau \rfloor + 1}{T}\tau^2|E|A_{max} + \tau \frac{\sum_{j=0}^{\lfloor (T-1)/\tau \rfloor} \sum_{e\in E} \mathbb{E}[Q_e(\tau_j)]}{T}$$
$$\leq 2\tau|E|A_{max} + \frac{\sum_{j=0}^{\lfloor (T-1)/\tau \rfloor} \sum_{e\in E} \mathbb{E}[Q_e(\tau_j)]}{\lfloor (T-1)/\tau \rfloor}.$$

Taking lim sup with $T \to \infty$ and using Theorem 8, we conclude that under DUMW:

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \sum_{e\in E} \mathbb{E}[\hat{Q}_e(t)] < \infty.$$

□

We now also consider the physical queues $\{\mathbf{U}(t)\}_{t\geq 0}$ [50], whereby $U_e(t)$ captures the number of physical packets backlogged at edge $e \in E$ at time slot t. The strong stability of $\{\hat{\mathbf{Q}}(t)\}_{t\geq 0}$ (Lemma 7) is sufficient to establish the rate-stability of the physical queues in the following Theorem.

Theorem 9. *Under the DUMW policy, the physical queues are rate-stable [33] for*

any arrival rate vector $\boldsymbol{\lambda} \in int(\boldsymbol{\Lambda})$*, i.e.*

$$\lim_{t\to\infty} \frac{\sum_{e\in E} U_e(t)}{t} = 0, \quad w.p.1$$

Proof. The proof directly follows the proof of [46, Theorem 3], which requires the strong stability of the 1-step virtual queue process $\{\hat{\mathbf{Q}}(t)\}_{t\geq 0}$ proven in Lemma 7. □

Remark 1. *While [46] proved Theorem 9 for the case of Extended Nearest To Origin (ENTO) packet scheduling, we strongly note that the results also hold for FIFO packet scheduling. For the proof sketch, using SLLN and the strong stability of the virtual queue process, we can show that the FIFO fluid model is stable [5], which then implies the stability of the physical network [9].*

Now, we can bound the number of distinct packets $R^{(c)}(t)$ of class $c \in \mathcal{C}$ that have reached all of the destination nodes as follows:

$$\sum_{q=0}^{t} A^{(c)}(t) - \sum_{e\in E} U_e(t) \leq R^{(c)}(t) \leq \sum_{q=0}^{t} A^{(c)}(t), \tag{3.70}$$

where the lower bound follows the fact that any packet out of the $\sum_{q=0}^{t} A^{(c)}(t)$ packets that have entered the network must either have reached the final destination, i.e. counted in $R^{(c)}(t)$, or be present in some physical queue, i.e. counted in $\sum_{e\in E} U_e(t)$. Dividing both sides of (3.70) by t, taking the limit $t \to \infty$, using Theorem 9, and noting that $\mathbb{E}A^{(c)}(t) = \lambda^{(c)}$, we obtain that under DUMW policy:

$$\liminf_{t\to\infty} \frac{R^{(c)}(t)}{t} = \lambda^{(c)}, \quad \forall c \in \mathcal{C}, w.p.1$$

By definition, we conclude that DUMW is throughput-optimal.

3.6.6 Supplementary Lemmas

For technical exposition, We define the function:

$$G(\mathbf{x},\alpha,\mathbf{Q}) = \sum_{\beta\in\Xi} P(C[t]=\beta \big| C[t-\tau]=\alpha) \times \left[\sum_{e\in E} Q_e\cdot\beta_e\cdot x(e,\beta_{E_{k_e}},\alpha)\prod_{e'\in I_e}(1-\beta_{e'}x(e',\beta_{E_{k'_e}},\alpha))\right]. \tag{3.71}$$

Furthermore, given $\alpha\in\{0,1\}^{|E|}$, for notation, we define $S_\alpha = \{\mathbf{x}\in\{0,1\}^{M_0} : \mathbf{x} = \{x(e,\xi,\alpha)\}_{e\in E,\xi\in\{0,1\}^{|E_{k_e}|}}\}$ with $M_0=\sum_{j=1}^m |E_j|2^{|E_j|}$ be the set of binary vectors $\mathbf{x}$ of M_0 entries indexed by e,Ξ and α. Intuitively, as in SCHEDULE, $x(e,\xi,\alpha)$ corresponds to the scheduling decision of whether to activate link e given the observation of fresh local NSI as Ξ and delayed global NSI as α.

Properties of $G(\mathbf{x},\alpha,\mathbf{Q})$

Lemma 8. *Let* $\mathbf{x}_1 = \text{argmax}_{\mathbf{x}\in S_\alpha} G(\mathbf{x},\alpha,\mathbf{Q})$ *and* $\mathbf{x}_2 = \text{argmax}_{\mathbf{x}\in S_\alpha} G(\mathbf{x},\alpha,\mathbf{Q}')$. *We have the followings:*

$$|G(\mathbf{x},\alpha,\mathbf{Q}) - G(\mathbf{x},\alpha,\mathbf{Q}')| \le \|\mathbf{Q}-\mathbf{Q}'\|_1, \forall \mathbf{x}\in S_\alpha \tag{3.72}$$

$$|\max_{\mathbf{x}\in S_\alpha} G(\mathbf{x},\alpha,\mathbf{Q}) - \max_{\mathbf{x}\in S_\alpha} G(\mathbf{x},\alpha,\mathbf{Q}')| \le \|\mathbf{Q}-\mathbf{Q}'\|_1 \tag{3.73}$$

$$G(\mathbf{x}_1,\alpha,\mathbf{Q}') \ge \max_{\mathbf{x}\in S_\alpha} G(\mathbf{x},\alpha,\mathbf{Q}') - 2\|\mathbf{Q}-\mathbf{Q}'\|_1 \tag{3.74}$$

Proof. For (3.72), we have:

$$
\begin{aligned}
&G(\mathbf{x}_1, \alpha, \mathbf{Q}) \\
&= \sum_{\beta \in \Xi} P(C[t] = \beta \big| C[t-\tau] = \alpha) \times \left[\sum_{e \in E} Q_e \cdot \beta_e x(e, \beta_{E_{k_e}}, \alpha) \prod_{e' \in I_e} (1 - \beta_{e'} x(e', \beta_{E_{k'_e}}, \alpha)) \right] \\
&= G(\mathbf{x}, \alpha, \mathbf{Q}') \\
&+ \sum_{\beta \in \Xi} P(C[t] = \beta \big| C[t-\tau] = \alpha) \times \left[\sum_{e \in E} (Q_e - Q'_e) \cdot \beta_e x(e, \beta_{E_{k_e}}, \alpha) \prod_{e' \in I_e} (1 - \beta_{e'} x(e', \beta_{E_{k'_e}}, \alpha)) \right] \\
&\leq G(\mathbf{x}, \alpha, \mathbf{Q}') + \sum_{\beta \in \Xi} P(C[t] = \beta \big| C[t-\tau] = \alpha) \times \left[\sum_{e \in E'} |Q_e - Q'_e| \right] \\
&= G(\mathbf{x}, \alpha, \mathbf{Q}') + \sum_{\beta \in \Xi} P(C[t] = \beta \big| C[t-\tau] = \alpha) \times \|\mathbf{Q} - \mathbf{Q}'\|_1 \\
&= G(\mathbf{x}, \alpha, \mathbf{Q}') + \|\mathbf{Q} - \mathbf{Q}'\|_1.
\end{aligned}
$$

By symmetry, we obtain that: $\max_{\mathbf{x} \in S_\alpha} G(\mathbf{x}, \alpha, \mathbf{Q}') \leq G(\mathbf{x}, \alpha, \mathbf{Q}) + \|\mathbf{Q} - \mathbf{Q}'\|_1$, which concludes the proof of (3.73).

For (3.73), we have:

$$
\begin{aligned}
\max_{\mathbf{x} \in S_\alpha} G(\mathbf{x}, \alpha, \mathbf{Q}) = G(\mathbf{x}_1, \alpha, \mathbf{Q}) &\overset{(3.72)}{\leq} G(\mathbf{x}_1, \alpha, \mathbf{Q}) + \|\mathbf{Q} - \mathbf{Q}'\|_1 \\
&\leq \max_{\mathbf{x} \in S_\alpha} G(\mathbf{x}, \alpha, \mathbf{Q}') + \|\mathbf{Q} - \mathbf{Q}'\|_1.
\end{aligned}
$$

By symmetry, we obtain that: $\max_{\mathbf{x} \in S_\alpha} G(\mathbf{x}, \alpha, \mathbf{Q}') \leq \max_{\mathbf{x} \in S_\alpha} G(\mathbf{x}, \alpha, \mathbf{Q}) + \|\mathbf{Q} - \mathbf{Q}'\|_1$, which concludes the proof of (3.73).

For (3.74), we have:

$$
\begin{aligned}
G(\mathbf{x}_1, \alpha, \mathbf{Q}') &\overset{(3.73)}{\geq} G(\mathbf{x}_1, \alpha, \mathbf{Q}) - \|\mathbf{Q} - \mathbf{Q}'\|_1 \\
&= \max_{\mathbf{x} \in S_\alpha} G(\mathbf{x}, \alpha, \mathbf{Q}) - \|\mathbf{Q} - \mathbf{Q}'\|_1 \\
&\overset{(3.73)}{\geq} \max_{\mathbf{x} \in S_\alpha} G(\mathbf{x}, \alpha, \mathbf{Q}') - 2\|\mathbf{Q} - \mathbf{Q}'\|_1
\end{aligned}
$$

□

Optimality of the Scheduling Algorithm SCHEDULE

Lemma 9. *The DUMW policy satisfies* $\forall t \in [\tau_j, \tau_{j+1})$*:*

$$\mathbb{E}\Big[\sum_{e\in E} Q_e(\tau_j)\cdot \mu_e^{\boldsymbol{DUMW}}(t)\Big|C[t-\tau]=\alpha, \mathbf{Q}(\tau_j)\Big]$$
$$\geq \max_{\pi\in\widetilde{\Pi}_s} \mathbb{E}\Big[\sum_{e\in E} Q_e(\tau_j)\cdot \mu_e^{\pi}(t)\Big|C[t-\tau]=\alpha, \mathbf{Q}(\tau_j)\Big] - 2\mathbb{E}[\bar{\epsilon}(j)] - 2\tau(A_{max}+|E|)$$

Proof. We first prove that there exists an optimal policy:

$$\pi^* = \operatorname*{argmax}_{\pi\in\widetilde{\Pi}_s} \mathbb{E}\Big[\sum_{e\in E} Q_e(\tau_j)\cdot \mu_e^{\pi}(t)\Big|C[t-\tau]=\alpha, \mathbf{Q}(\tau_j)\Big], \tag{3.75}$$

which is *deterministic*. We know that any policy $\pi \in \widetilde{\Pi}_s$ can be characterized by the set of feasible probabilities $\big\{P(D_{E_i}^{\pi}(\tau)=M_{E_i}\big|C[t-\tau]=\alpha, C_{E_i}[t]=\beta_{E_i}, \mathbf{Q}(t))\big\}$ with $M_{E_i}\in\{0,1\}^{|E_i|}\ \forall i\in[1,m]$ and $\alpha,\beta\in\{0,1\}^{|E|}$. For abbreviation, we let $p_i^{\pi}(M_{E_i}\big|\beta_{E_i},\alpha,\mathbf{Q}(t)) = P(D_{E_i}^{\pi}(\tau)=M_{E_i}\big|C[t-\tau]=\alpha, C_{E_i}[t]=\beta_{E_i},\mathbf{Q}(t))$. The objective (3.75) can be expressed as:

$$\mathbb{E}\Big[\sum_{e\in E} Q_e(\tau_j)\cdot \mu_e^{\pi}(t)\Big|C[t-\tau]=\alpha, \mathbf{Q}(\tau_j)\Big] \tag{3.76}$$
$$=\sum_{\beta\in\Xi} P(C[t]=\beta\big|C[t-\tau]=\alpha)\times \mathbb{E}\Big[\sum_{e\in E} Q_e(\tau_j)\cdot \mu_e^{\pi}(t)\Big|C[t]=\beta, C[t-\tau]=\alpha, \mathbf{Q}(\tau_j)\Big] \tag{3.77}$$
$$=\sum_{\beta\in\Xi} P(C[t]=\beta\big|C[t-\tau]=\alpha)\times\Bigg[\sum_{M\in\{0,1\}^{|E|}} P(D^{\pi}(\tau)=M\big|C[t-\tau]=\alpha, C[t]=\beta, \mathbf{Q}(\tau_j))$$
$$\times \underbrace{\mathbb{E}\Big[\sum_{e\in E} Q_e(\tau_j)\cdot \mu_e^{\pi}(t)\Big|D^{\pi}(\tau)=M, C[t]=\beta, C[t-\tau]=\alpha, \mathbf{Q}(\tau_j)\Big]}_{F(M,\beta,\alpha,\mathbf{Q}(\tau_j))}\Bigg] \tag{3.78}$$
$$=\sum_{\beta\in\Xi} P(C[t]=\beta\big|C[t-\tau]=\alpha)\times\Bigg[\sum_{M\in\{0,1\}^{|E|}}\prod_{i=1}^{m} p_i^{\pi}(M_{E_i}\big|\beta_{E_i},\alpha,\mathbf{Q}(\tau_j))\times F(M,\beta,\alpha,\mathbf{Q}(\tau_j))\Bigg], \tag{3.79}$$

where (3.77) and (3.78) are by law of iterated expectation, and (3.79) is by the fact

that decisions across domains are made independently. Furthermore, we note that $F(M, \beta, \alpha, \mathbf{Q}(\tau_j))$ is non-negative and can be evaluated as:

$$F(M, \beta, \alpha, \mathbf{Q}(\tau_j)) = \sum_{e \in E} Q_e(\tau_j) \cdot \beta_e M_e \prod_{e' \in I_e} (1 - \beta_{e'} M_{e'}). \tag{3.80}$$

Now take any optimal policy $\widetilde{\pi}$ maximizing (3.76), i.e.:

$$\begin{aligned} \widetilde{\pi} &= \underset{\pi \in \widetilde{\Pi}_s}{\operatorname{argmax}} \, \mathbb{E}\Big[\sum_{e \in E} Q_e(\tau_j) \cdot \mu_e^{\pi}(t) \Big| C[t-\tau] = \alpha, \mathbf{Q}(\tau_j) \Big]. \qquad (3.81) \\ &= \underset{\{p_i^{\pi}(|)\}}{\operatorname{argmax}} \sum_{\beta \in \Xi} P(C[t] = \beta \big| C[t-\tau] = \alpha) \\ &\qquad \times \Big[\sum_{M \in \{0,1\}^{|E|}} \prod_{i=1}^{m} p^{\pi}(M_{E_i} \big| \beta_{E_i}, \alpha, Q(\tau_j)) \times F(M, \beta, \alpha, \mathbf{Q}(\tau_j)) \Big] \end{aligned}$$

Now fix any $\beta_{E_i} \in \{0,1\}^{|E_i|}$ and $\alpha \in \{0,1\}^{|E|}$. We show that from $\widetilde{\pi}$ we can construct a new optimal policy $\widetilde{\pi}'$ (i.e. maximizing (3.76)) that makes *deterministic* scheduling decisions for the domain i^{th} given the observation of its fresh local NSI as $C_{E_i}[t] = \beta_{E_i}$, , delayed global NSI as $C_E[t-\tau] = \alpha$, and virtual queue values $\mathbf{Q}(\tau_j)$. Since (3.79) is multi-linear in term of the variables $p_i^{\pi}(M_{E_i} \big| \beta_{E_i}, \alpha, \mathbf{Q}(\tau_j))$'s, we can write (3.76) as:

$$\mathbb{E}\Big[\sum_{e \in E} Q_e(\tau_j) \cdot \mu_e^{\widetilde{\pi}}(t) \Big| C[t-\tau] = \alpha, \mathbf{Q}(\tau_j) \Big] = \sum_{M' \in \{0,1\}^{|E_i|}} p_i^{\widetilde{\pi}}(M' | \beta_{E_i}, \alpha, \mathbf{Q}(\tau_j)) \cdot C_{M'}^{\widetilde{\pi}}, \tag{3.82}$$

where:

$$\begin{aligned} C_{M'}^{\widetilde{\pi}} &= \sum_{M \in \{0,1\}^{|E_i|} : M_{E_i} = M'} \Big[\sum_{\beta \in \Xi} P(C[t] = \beta \big| C[t-\tau] = \alpha) \\ &\qquad \times \prod_{j \neq i}^{m} p_j^{\pi}(M_{E_j} \big| \beta_{E_j}, \alpha, \mathbf{Q}(\tau_j)) \times F(M, \beta, \alpha, \mathbf{Q}(\tau_j)) \Big], \\ &\sum_{M' \in \{0,1\}^{|E_i|}} p_i^{\widetilde{\pi}}(M' | \beta_{E_i}, \alpha, \mathbf{Q}(\tau_j)) = 1 \qquad (3.83) \end{aligned}$$

All $C_{M'}^{\widetilde{\pi}}$ for $M' \in \{0,1\}^{|E_i|}$ do not depend on the variables $p_i^{\widetilde{\pi}}(|)$ in their formulas. Let

$M_1' = \text{argmax}_{M' \in \{0,1\}^{|E_i|}} C_{M'}^{\widetilde{\pi}}$ Consider the policy $\widetilde{\pi}'$ such that:

$$p_i^{\widetilde{\pi}'}(M'|\beta_{E_i}, \alpha, \mathbf{Q}(\tau_j)) = \begin{cases} 1, \text{ if } M' = M_1' \\ 0, \text{ otherwise} \end{cases} \tag{3.84}$$

$$p_j^{\widetilde{\pi}'}(M'|\beta'_{E_j}, \alpha', \mathbf{Q}(\tau_j)) = p_j^{\widetilde{\pi}}(M'|\beta'_{E_j}, \alpha', \mathbf{Q}(\tau_j)), \quad \forall (\beta'_{E_j}, \alpha') \neq (\beta_{E_i}, \alpha) \tag{3.85}$$

Since $\widetilde{\pi}'$ only differs from $\widetilde{\pi}$ in $p_i^{\widetilde{\pi}}(.|\beta_{E_i}, \alpha, \mathbf{Q}(\tau_j))$, we have $C_{M'}^{\widetilde{\pi}'} = C_{M'}^{\widetilde{\pi}}$ for all $M' \in \{0,1\}^{|E_i|}$. Thus,

$$\begin{aligned}
&\mathbb{E}\Big[\sum_{e\in E} Q_e(\tau_j) \cdot \mu_e^{\widetilde{\pi}'}(t)\Big|C[t-\tau] = \alpha, \mathbf{Q}(\tau_j)\Big] \\
&= \sum_{M' \in \{0,1\}^{|E_i|}} p_i^{\widetilde{\pi}'}(M'|\beta_{E_i}, \alpha, \mathbf{Q}(\tau_j)) \cdot C_{M'}^{\widetilde{\pi}'} = C_{M_1'}^{\widetilde{\pi}} && (3.86) \\
&\geq \sum_{M' \in \{0,1\}^{|E_i|}} p_i^{\widetilde{\pi}}(M'|\beta_{E_i}, \alpha, \mathbf{Q}(\tau_j)) \cdot C_{M}^{\widetilde{\pi}} && (3.87) \\
&= \mathbb{E}\Big[\sum_{e\in E} Q_e(\tau_j) \cdot \mu_e^{\widetilde{\pi}}(t)\Big|C[t-\tau] = \alpha, \mathbf{Q}(\tau_j)\Big], && (3.88)
\end{aligned}$$

where (3.87) is by (3.82) and $C_{M_1'}^{\widetilde{\pi}} \geq C_{M'}^{\widetilde{\pi}}, \forall M' \in \{0,1\}^{|E_i|}$. Now, (3.88) and the fact that $\widetilde{\pi}$ is an optimal policy (as in (3.81)) imply that $\widetilde{\pi}'$ is also an optimal policy, i.e.:

$$\widetilde{\pi}' = \underset{\pi \in \widetilde{\Pi}_s}{\text{argmax}}\, \mathbb{E}\Big[\sum_{e\in E} Q_e(\tau_j) \cdot \mu_e^{\pi}(t)\Big|C[t-\tau] = \alpha, \mathbf{Q}(\tau_j)\Big],$$

with the property that $\widetilde{\pi}'$ makes *deterministic* decisions given β_{E_i}, α and $Q(t)$ according to (3.84). Repeating the process for all β_{E_j}'s, we obtain a deterministic optimal policy π^*. Now consider $\widetilde{\Pi}_{det} = \{\pi \in \widetilde{\Pi}_s : \pi \text{ is deterministic}\}$. Our original problem of maximizing the expected service rate can be equivalently characterized as:

$$\max_{\pi \in \widetilde{\Pi}_s} \mathbb{E}\Big[\sum_{e\in E} Q_e(\tau_j) \cdot \mu_e^{\pi}(t)\Big|C[t-\tau] = \alpha, \mathbf{Q}(\tau_j)\Big] = \max_{\pi \in \widetilde{\Pi}_{det}} \mathbb{E}\Big[\sum_{e\in E} Q_e(\tau_j) \cdot \mu_e^{\pi}(t)\Big|C[t-\tau] = \alpha, \mathbf{Q}(\tau_j)\Big] \tag{3.89}$$

Take any $\pi \in \widetilde{\Pi}_{det}$. Consider any $\beta, \alpha \in \{0,1\}^{|E|}$ representing the fresh and delayed

global NSI and the decomposition of fresh global NSI into fresh local NSI as $\beta = \cup_{i=1}^{m} \beta_{E_i}$ with $\beta_{E_i} \in \{0,1\}^{|E_i|}$. Since π is deterministic, there exists a unique schedule vector $M^{\pi}_{\beta_{E_i},\alpha} \in \{0,1\}^{|E_i|}$ such that $p_i^{\pi}(M^{\pi}_{\beta_{E_i},\alpha}|\beta_{E_i}, \alpha, \mathbf{Q}(\tau_j)) = 1$ and $p_i^{\pi}(M'|\beta_{E_i}, \alpha, \mathbf{Q}(\tau_j)) = 0, \forall M' \neq M^{\pi}_{\beta_{E_i},\alpha}$. We denote the decision of π on whether to activate link e based on the observation of NSI as:

$$x^{\pi}(e, \beta_{E_i}, \alpha) = (M^{\pi}_{\beta_{E_i},\alpha})_e \tag{3.90}$$

Following the uniqueness of $M^{\pi}_{\beta_{E_i},\alpha}$, we uniquely define global schedule vector $M^{\pi}_{\beta,\alpha} = \cup_{i=1}^{m} (M^{\pi}_{\beta,\alpha})_{E_i} \in \{0,1\}^{|E|}$ with $(M^{\pi}_{\beta,\alpha})_{E_i} = M^{\pi}_{\beta_{E_i},\alpha}$. We then obtain that $\forall M \in \{0,1\}^{|E|}$:

$$\begin{aligned}
&P(D^{\pi}(\tau) = M \big| C[t-\tau] = \alpha, C[t] = \beta, \mathbf{Q}(\tau_j)) \\
=&\prod_{i=1}^{m} p_i^{\pi}(M_{E_i} \big| \beta_{E_i}, \alpha, QQ(\tau_j)) = \begin{cases} 1, \text{ if } M = M^{\pi}_{\beta,\alpha} \\ 0, \text{ otherwise} \end{cases}
\end{aligned}$$

From the above, we further expand (3.79) $\forall \pi \in \widetilde{\Pi}_{det}$ as:

$$\begin{aligned}
&\mathbb{E}\Big[\sum_{e \in E} Q_e(\tau_j) \cdot \mu_e^{\pi}(t) \big| C[t-\tau] = \alpha, \mathbf{Q}(\tau_j) \Big] \\
&= \sum_{\beta \in \Xi} P(C[t] = \beta \big| C[t-\tau] = \alpha) \times \Big[F(M^{\pi}_{\beta,\alpha}, \beta, \alpha, \mathbf{Q}(\tau_j)) \Big] \\
&\overset{(3.80)}{=} \sum_{\beta \in \Xi} P(C[t] = \beta \big| C[t-\tau] = \alpha) \\
&\times \Big[\sum_{e \in E} Q_e(\tau_j) \cdot \beta_e \cdot (M^{\pi}_{\beta,\alpha})_e \prod_{e' \in I_e} (1 - \beta_{e'} (M^{\pi}_{\beta,\alpha})_{e'}) \Big] \\
&\overset{(3.90)}{=} \sum_{\beta \in \Xi} P(C[t] = \beta \big| C[t-\tau] = \alpha) \\
&\times \Big[\sum_{e \in E} Q_e(\tau_j) \cdot \beta_e \cdot x^{\pi}(e, \beta_{E_{k_e}}, \alpha) \prod_{e' \in I_e} (1 - \beta_{e'} x^{\pi}(e', \beta_{E_{k'_e}}, \alpha)) \Big] \\
&\overset{(3.71)}{=} G(\mathbf{x}^{\pi}, \alpha, \mathbf{Q}(\tau_j))
\end{aligned}$$

$$\therefore \max_{\pi \in \widetilde{\Pi}_{det}} \mathbb{E}\Big[\sum_{e \in E} Q_e(\tau_j) \cdot \mu_e^{\pi}(t) \big| C[t-\tau] = \alpha, \mathbf{Q}(\tau_j) \Big] = \max_{\mathbf{x}^{\pi}} G(\mathbf{x}^{\pi}, \alpha, \mathbf{Q}(\tau_j))$$

Combining with (3.89), we conclude that:

$$\max_{\pi\in\widetilde{\Pi}_s}\mathbb{E}\bigg[\sum_{e\in E}Q_e(\tau_j)\cdot\mu_e^{\pi}(t)\big|C[t-\tau]=\alpha,\mathbf{Q}(\tau_j)\bigg]=\max_{\mathbf{x}^{\pi}}G(\mathbf{x}^{\pi},\alpha,\mathbf{Q}(\tau_j)),\tag{3.91}$$

i.e. finding the optimal policy $\pi\in\widetilde{\Pi}_s$ corresponds to finding the optimal schedule vector $\mathbf{x}^{\pi}$ maximizing $G(\mathbf{x}^{\pi},\alpha,\mathbf{Q}(\tau_j))$. Now, from the scheduling algorithm SCHEDULE of DUMW, we know that DUMW solves $\max_{\mathbf{x}^{\pi}}G(\mathbf{x}^{\pi},\alpha,\widetilde{\mathbf{Q}}(\tau_{j-1}))$. Applying Lemma 8, we get:

$$\begin{aligned}
&\mathbb{E}\bigg[\sum_{e\in E}Q_e(\tau_j)\cdot\mu_e^{\mathbf{DUMW}}(t)\big|C[t-\tau]=\alpha,\mathbf{Q}(\tau_j)\bigg]=G(\mathbf{x}^{\mathbf{DUMW}},\alpha,\mathbf{Q}(\tau_j))\\
&\geq\max_{\mathbf{x}^{\pi}}G(\mathbf{x}^{\pi},\alpha,\mathbf{Q}(\tau_j))-2\mathbb{E}\|\widetilde{\mathbf{Q}}(\tau_{j-1})-\mathbf{Q}(\tau_j)\|_1\\
&\overset{(3.32)}{\geq}\max_{\mathbf{x}^{\pi}}G(\mathbf{x}^{\pi},\alpha,\mathbf{Q}(\tau_j))-2\mathbb{E}[\bar{\epsilon}(j)]-2\tau(A_{max}+|E|)\\
&\overset{(3.91)}{\geq}\max_{\pi\in\widetilde{\Pi}_s}\mathbb{E}\bigg[\sum_{e\in E}Q_e(\tau_j)\cdot\mu_e^{\pi}(t)\big|C[t-\tau]=\alpha,\mathbf{Q}(\tau_j)\bigg]-2\mathbb{E}[\bar{\epsilon}(j)]-2\tau(A_{max}+|E|),
\end{aligned}$$

which concludes the proof. □

Skorokhod Map Representation of $\{\mathbf{Q}(\tau_j)\}_{j\geq0}$

Lemma 10. *Define $A_e^{\pi}(t_1,t_2)=\sum_{q=t_1}^{t_2-1}A_e^{\pi}(q)$ and $\mu_e^{\pi}(t_1,t_2)=\sum_{q=t_1}^{t_2-1}\mu_e^{\pi}(q)$. The discrete-time Skorokhod map representation of the virtual queue process $\{\mathbf{Q}(\tau_j)\}_{j\geq0}$ can be expressed as:*

$$Q_e(\tau_j)=\bigg(\sup_{1\leq q\leq j}\big(A_e^{\pi}(\tau_j-q\tau,\tau_j)-\mu_e^{\pi}(\tau_j-q\tau,\tau_j)\big)\bigg)^{+},\forall e\in E$$

Proof. From the virtual queue dynamics (3.12), we have $\forall e\in E$:

$$\begin{aligned}
Q_e(\tau_j)&=\Big(Q_e(\tau_{j-1})+A_e^{\pi}(\tau_{j-1},\tau_j)-\mu_e^{\pi}(\tau_{j-1},\tau_j)\Big)^{+}\\
&\geq Q_e(\tau_{j-1})+A_e^{\pi}(\tau_{j-1},\tau_j)-\mu_e^{\pi}(\tau_{j-1},\tau_j)\\
&=Q_e(\tau_{j-1})+A_e^{\pi}(\tau_j-\tau,\tau_j)-\mu_e^{\pi}(\tau_j-\tau,\tau_j),
\end{aligned}$$

where for the last line we recall that $\tau_k = k\tau, \forall k$. Iterating the above for $q \in [1, j]$ times and using the fact that the virtual queue values are non-negative, we obtain that $\forall e \in E$:

$$\begin{aligned} Q_e(\tau_j) &\geq Q_e(\tau_{j-q}) + A_e^\pi(\tau_j - q\tau, \tau_j) - \mu_e^\pi(\tau_j - q\tau, \tau_j) \\ &\geq A_e^\pi(\tau_j - q\tau, \tau_j) - \mu_e^\pi(\tau_j - q\tau, \tau_j) \end{aligned}$$

Taking sup of the above over $q \in [1, j]$ and using the fact that the virtual queue values are always non-negative, we have:

$$Q_e(\tau_j) \geq \left(\sup_{q \in [1,j]} \left\{ A_e^\pi(\tau_j - q\tau, \tau_j) - \mu_e^\pi(\tau_j - q\tau, \tau_j) \right\} \right)^+. \tag{3.92}$$

Now, we proceed to show that (3.92) holds with equality. If $Q_e(\tau_j) = 0$, then the non-negative RHS of (3.92) must be equal to 0, implying equality in (3.92). Else if $Q_e(\tau_j) > 0$, we consider the latest time τ_{j-q} (with $q \in [1, j]$) prior to τ_j such that $Q_e(\tau_{j-q}) = 0$. Such τ_{j-q} must exist since the system starts with $Q_e(\tau_0) = 0$. Consequently, $Q(\tau_{j-q'}) > 0$ for any $\tau_{j-q} < \tau_{j-q'} \leq \tau_j$ (or equivalently for $q' < q$). Therefore, the virtual queues $Q(\tau_{j-q'})$ for $q' \in [0, q)$ are positive and thus have the dynamics from (3.12) as:

$$Q_e(\tau_{j-q'}) = Q_e(\tau_{j-q'-1}) + A_e^\pi(\tau_{j-q'-1}, \tau_j) - \mu_e^\pi(\tau_{j-q'-1}, \tau_j). \tag{3.93}$$

Iterating (3.93) for the interval $q' \in [0, q)$, we obtain that:

$$\begin{aligned} Q_e(\tau_j) &= Q_e(\tau_{j-q}) + A_e^\pi(\tau_{j-q}, \tau_j) - \mu_e^\pi(\tau_{j-q}, \tau_j) \\ &= A_e^\pi(\tau_{j-q}, \tau_j) - \mu_e^\pi(\tau_{j-q}, \tau_j), \end{aligned}$$

which implies the equality of (3.92) and thus concludes the proof of this Lemma. □

Properties of the 1-Step Virtual Queue Process $\{\hat{\mathbf{Q}}(t)\}_{t\geq 0}$

Lemma 11. *We have the following bounds $\forall e \in E$:*

$$|\hat{Q}_e(t_1) - \hat{Q}_e(t_2)| \leq |t_1 - t_2|(A_{max} + 1) \tag{3.94}$$

$$Q_e(\tau_j) \leq \hat{Q}_e(\tau_j) \leq Q_e(\tau_j) + \tau A_{max}, \tag{3.95}$$

Proof. (3.94) trivially holds for $t_1 = t_2$. If $t_1 \neq t_2$, WLOG, we assume that $t_1 > t_2$. From the queue dynamics (3.67), we have:

$$\hat{Q}_e(t_1) \geq \hat{Q}_e(t_1 - 1) + A_e^\pi(t_1 - 1) - \mu_e^\pi(t_1 - 1) \geq \hat{Q}_e(t_1 - 1) - 1,$$
$$\hat{Q}_e(t_1) \leq \hat{Q}_e(t_1 - 1) + A_e^\pi(t_1 - 1) \leq \hat{Q}_e(t_1 - 1) + A_{max}.$$

Iterating the above, we obtain that:

$$\hat{Q}_e(t_1) \geq \hat{Q}_e(t_2) - (t_1 - t_2),$$
$$\hat{Q}_e(t_1) \leq \hat{Q}_e(t_2) + (t_1 - t_2)A_{max}.$$

Combining the two above, we obtain (3.94).

Now, we proceed to prove (3.95). Define $A_e^\pi(t_1, t_2) = \sum_{q=t_1}^{t_2-1} A_e^\pi(q)$ and $\mu_e^\pi(t_1, t_2) = \sum_{q=t_1}^{t_2-1} \mu_e^\pi(q)$. From Lemma 10, we express the virtual queue process $\{\mathbf{Q}(\tau_j)\}_{j\geq 0}$ as:

$$Q_e(\tau_j) = \left(\sup_{1\leq q\leq j} \left(A_e^\pi(\tau_j - q\tau, \tau_j) - \mu_e^\pi(\tau_j - q\tau, \tau_j) \right) \right)^+, \forall e \in E. \tag{3.96}$$

From the discrete time Skorokhod map representation of $\hat{\mathbf{Q}}(\tau_j)$ in [46], we have $\forall e \in E$:

$$\begin{aligned} \hat{Q}_e(\tau_j) &= \left(\sup_{1\leq l\leq \tau_j} \left(A_e^\pi(\tau_j - l, \tau_j) - \mu_e^\pi(\tau_j - l, \tau_j) \right) \right)^+ \\ &\overset{(3.96)}{=} \sup \left(Q_e(\tau_j), \sup_{1\leq l\leq \tau_j:\ l\not\mid \tau} \left(A_e^\pi(\tau_j - l, \tau_j) - \mu_e^\pi(\tau_j - l, \tau_j) \right) \right) \\ &\geq Q_e(\tau_j). \end{aligned} \tag{3.97}$$

For any $l \in [1, \tau_j]$, we consider $u \in [1, j]$ be the smallest number such that $u\tau - l \geq 0$. Such u exists since $\tau_j - l = j\tau - l \geq 0$. Furthermore, we must have $u\tau - l \leq \tau$; otherwise $(u-1)\tau > l$ contradicting the fact that u is the smallest number such that $u\tau - l \geq 0$. Thus, we have:

$$0 \leq u\tau - l \leq \tau \tag{3.98}$$

Noting that $\tau_j - l \leq \tau_j - u\tau$, we bound:

$$\begin{aligned} A_e^\pi(\tau_j - l, \tau_j) - \mu_e^\pi(\tau_j - l, \tau_j) &= [A_e^\pi(\tau_j - u\tau, \tau_j) - \mu_e^\pi(\tau_j - u\tau, \tau_j)] \\ &\quad + [A_e^\pi(\tau_j - l, \tau_j - u\tau) - \mu_e^\pi(\tau_j - l, \tau_j - u\tau)] \\ &\overset{(3.96)}{\leq} Q_e(\tau_j) + (u\tau - l)A_{max} + 0 \\ &\overset{(3.96)}{\leq} Q_e(\tau_j) + \tau A_{max}. \end{aligned}$$

Taking sup of the above over all $l \in [1, \tau_j]$, we obtain in view of (3.97) that:

$$\hat{Q}_e(\tau_j) \leq Q_e(\tau_j) + \tau A_{max}.$$

□

Bibliography

[1] Carlos A.B. Macapuna, Christian Esteve Rothenberg, and Magalhães F. Maurício. In-packet bloom filter based data center networking with distributed openflow controllers. In *2010 IEEE Globecom Workshops*, pages 584–588, 2010.

[2] Fetia Bannour, Sami Souihi, and Abdelhamid Mellouk. Distributed sdn control: Survey, taxonomy, and challenges. *IEEE Communications Surveys & Tutorials*, 20(1):333–354, 2018.

[3] Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz, Brian O'Connor, Pavlin Radoslavov, William Snow, and Guru Parulkar. Onos: Towards an open, distributed sdn os. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, page 1–6, New York, NY, USA, 2014. Association for Computing Machinery.

[4] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, 2006.

[5] Maury Bramson. Convergence to equilibria for fluid models of fifo queueing networks. *Queueing Systems*, 22:5–45, 1996.

[6] Loc X. Bui, Sujay Sanghavi, and R. Srikant. Distributed link scheduling with constant overhead. *IEEE/ACM Transactions on Networking*, 17(5):1467–1480, 2009.

[7] Guozhen Cheng, Hongchang Chen, Zhiming Wang, and Shuqiao Chen. Dha: Distributed decisions on the switch migration toward a scalable sdn control plane. In *2015 IFIP Networking Conference (IFIP Networking)*, pages 1–9, 2015.

[8] Salvatore Costanzo, Laura Galluccio, Giacomo Morabito, and Sergio Palazzo. Software defined wireless networks: Unbridling sdns. In *2012 European Workshop on Software Defined Networking*, pages 1–6, 2012.

[9] J. G. Dai. On Positive Harris Recurrence of Multiclass Queueing Networks: A Unified Approach Via Fluid Limit Models. *The Annals of Applied Probability*, 5(1):49 – 77, 1995.

[10] Laura Galluccio, Sebastiano Milardo, Giacomo Morabito, and Sergio Palazzo. Sdn-wise: Design, prototyping and experimentation of a stateful sdn solution for

wireless sensor networks. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 513–521, 2015.

[11] P. Giaccone, B. Prabhakar, and D. Shah. Randomized scheduling algorithms for high-aggregate bandwidth switches. *IEEE Journal on Selected Areas in Communications*, 21(4):546–559, 2003.

[12] Syed Sherjeel A. Gilani, Amir Qayyum, Rao Naveed Bin Rais, and Mukhtiar Bano. Sdnmesh: An sdn based routing architecture for wireless mesh networks. *IEEE Access*, 8:136769–136781, 2020.

[13] Carlos Gonzalez, Olivier Flauzac, Florent Nolot, and Antonio Jara. A novel distributed sdn-secured architecture for the iot. In *2016 International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 244–249, 2016.

[14] Deepthi Gopi, Samuel Cheng, and Robert Huck. Comparative analysis of sdn and conventional networks using routing protocols. In *2017 International Conference on Computer, Information and Telecommunication Systems (CITS)*, pages 108–112, 2017.

[15] Zehua Guo, Mu Su, Yang Xu, Zhemin Duan, Luo Wang, Shufeng Hui, and H. Chao. Improving the performance of load balancing in software-defined networks through load variance-based synchronization. *Computer Networks*, 68:95–109, 08 2014.

[16] Arpit Gupta, Laurent Vanbever, Muhammad Shahbaz, Sean P. Donovan, Brandon Schlinker, Nick Feamster, Jennifer Rexford, Scott Shenker, Russ Clark, and Ethan Katz-Bassett. Sdx: A software defined internet exchange. *SIGCOMM Comput. Commun. Rev.*, 44(4):551–562, aug 2014.

[17] Robert Hanmer, Lalita Jagadeesan, Veena Mendiratta, and Heng Zhang. Friend or foe: Strong consistency vs. overload in high-availability distributed systems and sdn. In *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 59–64, 2018.

[18] Israat Haque and MA Moyeen. Revive: A reliable software defined data plane failure recovery scheme. In *2018 14th International Conference on Network and Service Management (CNSM)*, pages 268–274, 2018.

[19] Changhee Joo, Xiaojun Lin, and Ness B. Shroff. Greedy maximal matching: Performance limits for arbitrary network graphs under the node-exclusive interference model. *IEEE Transactions on Automatic Control*, 54(12):2734–2744, 2009.

[20] Teemu Koponen, Martin Casado, Natasha Gude, Jeremy Stribling, Leon Poutievski, Min Zhu, Rajiv Ramanathan, Yuichiro Iwata, Hiroaki Inoue, Takayuki Hama, and Scott Shenker. Onix: A distributed control platform for large-scale production networks. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, page 351–364, USA, 2010. USENIX Association.

[21] Vasileios Kotronis, Rowan Klöti, Matthias Rost, Panagiotis Georgopoulos, Bernhard Ager, Stefan Schmid, and Xenofontas Dimitropoulos. Stitching inter-domain paths over ixps. In *Proceedings of the Symposium on SDN Research*, SOSR '16, New York, NY, USA, 2016. Association for Computing Machinery.

[22] Rakesh Kumar, Monowar Hasan, Smruti Padhy, Konstantin Evchenko, Lavanya Piramanayagam, Sibin Mohan, and Rakesh B. Bobba. End-to-end network delay guarantees for real-time systems using sdn. In *2017 IEEE Real-Time Systems Symposium (RTSS)*, pages 231–242, 2017.

[23] Mohamed Labraoui, Michael Mathias Boc, and Anne Fladenmuller. Software defined networking-assisted routing in wireless mesh networks. In *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 377–382, 2016.

[24] Rémy Lapeyrade, Marc Bruyère, and Philippe Owezarski. Openflow-based migration and management of the touix ixp. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pages 1131–1136, 2016.

[25] J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26(1):22–35, 1978.

[26] Dan Levin, Andreas Wundsam, Brandon Heller, Nikhil Handigol, and Anja Feldmann. Logically centralized? state distribution trade-offs in software defined networks. *HotSDN'12 - Proceedings of the 1st ACM International Workshop on Hot Topics in Software Defined Networks*, 08 2012.

[27] Dawei Li, Jie Wu, and Dajin Wang. Single-link failure recovery with or without software-defined networking switches. In *2018 International Conference on Information and Computer Technologies (ICICT)*, pages 87–91, 2018.

[28] Ying-Dar Lin, Hung-Yi Teng, Chia-Rong Hsu, Chun-Chieh Liao, and Yuan-Cheng Lai. Fast failover and switchover for link failures and congestion in software defined networks. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6, 2016.

[29] Yangyang Liu, Zhao Laiping, Jingyu Hua, Wenyu Qu, Suohao Zhang, and Sheng Zhong. Distributed traffic engineering for multi-domain sdn without trust. *IEEE Transactions on Cloud Computing*, pages 1–1, 2021.

[30] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, mar 2008.

[31] Mar). An Instant Virtual Network on your Laptop (or other PC). Mininet. (2013.

[32] George V. Moustakides. Extension of wald's first lemma to markov processes. *Journal of Applied Probability*, 36(1):48–59, 1999.

[33] Michael J. Neely. *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan and Claypool Publishers, 2010.

[34] Behrang Monajemi Nejad, Sid Ahmed Attia, and Jorg Raisch. Max-consensus in a max-plus algebraic setting: The case of fixed communication topologies. In *2009 XXII International Symposium on Information, Communication and Automation Technologies*, pages 1–7, 2009.

[35] Bruno Astuto A. Nunes, Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys Tutorials*, 16(3):1617–1634, 2014.

[36] Yustus Eko Oktian, SangGon Lee, HoonJae Lee, and JunHuy Lam. Distributed sdn controller system: A survey on design choice. *Computer Networks*, 121:100–111, 2017.

[37] Aurojit Panda, Wenting Zheng, Xiaohe Hu, Arvind Krishnamurthy, and Scott Shenker. Scl: Simplifying distributed sdn control planes. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, NSDI'17, page 329–345, USA, 2017. USENIX Association.

[38] George Petropoulos, Fragkiskos Sardis, Spiros Spirou, and Toktam Mahmoodi. Software-defined inter-networking: Enabling coordinated qos control across the internet. In *2016 23rd International Conference on Telecommunications (ICT)*, pages 1–5, 2016.

[39] Konstantinos Poularakis, Qiaofeng Qin, Liang Ma, Sastry Kompella, Kin Kwong Leung, and Leandros Tassiulas. Learning the optimal synchronization rates in distributed sdn control architectures. *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 1099–1107, 2019.

[40] Heng Qi and Keqiu Li. *Software Defined Networking Applications in Distributed Datacenters*. 01 2016.

[41] Qiaofeng Qin, Konstantinos Poularakis, George Iosifidis, and Leandros Tassiulas. Sdn controller placement at the edge: Optimizing delay and overheads. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 684–692, 2018.

[42] Akula Aneesh Reddy, Siddhartha Banerjee, Aditya Gopalan, Sanjay Shakkottai, and Lei Ying. On distributed scheduling with heterogeneously delayed network-state information. *Queueing Syst. Theory Appl.*, 72(3–4):193–218, dec 2012.

[43] Ermin Sakic and Wolfgang Kellerer. Decoupling of distributed consensus, failure detection and agreement in sdn control plane. In *2020 IFIP Networking Conference (Networking)*, pages 467–475, 2020.

[44] Zhaogang Shu, Jiafu Wan, Jiaxiang Lin, Shiyong Wang, Di Li, Seungmin Rho, and Changcai Yang. Traffic engineering in software-defined networking: Measurement and management. *IEEE Access*, 4:3246–3256, 2016.

[45] Liran Sidki, Yehuda Ben-Shimol, and Akiva Sadovski. Fault tolerant mechanisms for sdn controllers. In *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 173–178, 2016.

[46] Abhishek Sinha and Eytan Modiano. Optimal control for generalized network-flow problems. *IEEE/ACM Transactions on Networking*, 26(1):506–519, 2018.

[47] Jonathan Stringer, Dean Pemberton, Qiang Fu, Christopher Lorier, Richard Nelson, Josh Bailey, Carlos N. A. Corrêa, and Christian Esteve Rothenberg. Cardigan: Sdn distributed routing fabric going live at an internet exchange. In *2014 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7, 2014.

[48] Adrian S.-W. Tam, Kang Xi, and H. Jonathan Chao. Use of devolved controllers in data center networks. In *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 596–601, 2011.

[49] L. Tassiulas. Linear complexity algorithms for maximum throughput in radio networks and input queued switches. In *Proceedings. IEEE INFOCOM '98, the Conference on Computer Communications. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Gateway to the 21st Century (Cat. No.98*, volume 2, pages 533–539 vol.2, 1998.

[50] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):1936–1948, 1992.

[51] Amin Tootoonchian and Yashar Ganjali. Hyperflow: A distributed control plane for openflow. In *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, INM/WREN'10, page 3, USA, 2010. USENIX Association.

[52] Niels L. M. Van Adrichem, Benjamin J. Van Asten, and Fernando A. Kuipers. Fast recovery in software-defined networks. In *2014 Third European Workshop on Software Defined Networks*, pages 61–66, 2014.

[53] BI Jun WANG Yangyang. Survey of mechanisms for inter-domain sdn. *ZTE Communications*, 15(3):8, 2017.

[54] Di Wu, Xiang Nie, Eskindir Asmare, Dmitri I. Arkhipov, Zhijing Qin, Renfa Li, Julie A. McCann, and Keqin Li. Towards distributed sdn: Mobility management and flow scheduling in software defined urban iot. *IEEE Transactions on Parallel and Distributed Systems*, 31(6):1400–1418, 2020.

[55] Yang Xu, Marco Cello, I-Chih Wang, Anwar Walid, Gordon Wilfong, Charles H.-P. Wen, Mario Marchese, and H. Jonathan Chao. Dynamic switch migration in distributed software-defined networks to achieve controller load balance. *IEEE Journal on Selected Areas in Communications*, 37(3):515–529, 2019.

[56] KK Yap, Murtaza Motiwala, Jeremy Rahe, Steve Padgett, Matthew Holliman, Gary Baldus, Marcus Hines, TaeEun Kim, Ashok Narayanan, Ankur Jain, Victor Lin, Colin Rice, Brian Rogan, Arjun Singh, Bert Tanaka, Manish Verma, Puneet Sood, Mukarram Tariq, Matt Tierney, Dzevad Trumic, Vytautas Valancius, Calvin Ying, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. Taking the edge off with espresso: Scale, reliability and programmability for global internet peering. 2017.

[57] Soheil Hassas Yeganeh, Amin Tootoonchian, and Yashar Ganjali. On scalability of software-defined networking. *IEEE Communications Magazine*, 51(2):136–141, 2013.

[58] Lei Ying and Sanjay Shakkottai. On throughput optimality with delayed network-state information. *IEEE Transactions on Information Theory*, 57(8):5116–5132, 2011.

[59] Ziyao Zhang, Liang Ma, Kin Kwong Leung, Franck Le, Sastry Kompella, and Leandros Tassiulas. How better is distributed sdn? an analytical approach. *ArXiv*, abs/1712.04161, 2017.

[60] Ziyao Zhang, Liang Ma, Konstantinos Poularakis, Kin K. Leung, Jeremy Tucker, and Ananthram Swami. Macs: Deep reinforcement learning based sdn controller synchronization policy design. In *2019 IEEE 27th International Conference on Network Protocols (ICNP)*, pages 1–11, 2019.

[61] Ziyao Zhang, Liang Ma, Konstantinos Poularakis, Kin Kwong Leung, and Lingfei Wu. Dq scheduler: Deep reinforcement learning based controller synchronization in distributed sdn. *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pages 1–7, 2019.

[62] Yuanhao Zhou, Mingfa Zhu, Limin Xiao, Li Ruan, Wenbo Duan, Deguo Li, Rui Liu, and Mingming Zhu. A load balancing strategy of sdn controller based on distributed decision. In *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 851–856, 2014.