Optimal Control for Distributed Wireless SDN

Quang Minh Nguyen, Eytan H. Modiano
LIDS, Massachusetts Institute of Technology, Cambridge, MA, USA
Email: {nmquang, modiano}@mit.edu

***Abstract*—Distributed wireless Software Defined Networking (SDN) has emerged as a solution to the scalability bottleneck of traditional SDN. Despite its inherent importance, optimal network control for 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.**

In this paper, we propose the Distributed Universal Max-Weight (DUMW) algorithm as a novel optimal control framework for distributed wireless SDN. The DUMW algorithm is theoretically throughput-optimal and practically congruent with SDN system idiosyncrasies. Our algorithmic development non-trivially extends the throughput-optimal Universal Max-Weight (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.

I. Introduction

Software-Defined Networking (SDN) [1] emerged from the urgent need for a programmable networking framework with adaptive reconfiguration to meet modern networking requirements. Its flexibility has facilitated the implementation of widespread network management functionalities such as routing, load-balancing and traffic engineering. 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 with the requirements for dynamic control and distributed operations. Nonetheless, there has been increasing interest in wireless SDN thanks to the surge in mobile communication and wireless infrastructures [2]. 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 and suffers from the single point-of-failure problem. Distributed SDN [3] has emerged to mitigate the scalability and reliability bottlenecks. In this work, we focus on designing optimal control framework for distributed SDN.

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 synchronize with each other to partially or fully maintain their global view of the network, which can be utilized to enhance decision making for inter-domain tasks. While there have been several consistency models considered in the literature, the two most predominant classes are: *strong consistency* and *eventual consistency* [3]. By requiring all the controllers to be synchronized at any time, strongly consistent protocols maintain fresh global network information for optimizing application performance. However, its practicality is hindered by the unreliable nature of network communications and the prohibitively high overhead for frequent controller coordination. 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, yet the inconsistent network information significantly degrades the inter-domain application performance. Improving the performance of ad-hoc inter-domain tasks via customized algorithms and synchronization strategies is an active area of research [4]. While there have been several applications considered in the literature, such as traffic engineering, load-balancing or utility maximization, optimal network control comprised of routing and scheduling is the most prominent for being the backbone of network operations. To this end, we study the optimal network control framework for distributed wireless SDN under the eventual consistency model.

A. Open Problems and Motivation

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, where much of the work is focused on optimizing the controller-synchronization rate. 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

This work was funded by ONR grant award N00014-20-1-2119, and by NSF grants CNS-2148128 and CNS-1907905.

ISBN 978-3-903176-63-8 © 2024 IFIP.

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. To fill this gap, we thus investigate the unsolved yet critical problem of optimal network control for distributed wireless SDN. Our goal is to develop a new algorithm that is theoretically throughput-optimal and practically congruent with the distributed SDN architecture and the wireless environments.

B. Contributions

In this paper, we propose a novel unified optimal control framework for distributed wireless SDN that fully addresses the aforementioned challenges. Our contributions are three-fold:

- We formulate the problem of optimal inter-domain routing and scheduling for distributed wireless SDN. 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.
- We present the Distributed Universal Max-Weight (DUMW) algorithm for distributed wireless SDN that is throughput-optimal and can handle generalized wireless interference constraints. Unlike conventional control schemes, DUMW leverages easy-to-track virtual queues, in place of physical queues whose operations are not supported by SDN switches. Moreover, 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.

II. Software-Defined Network (SDN)

A. Wireless SDN Architecture

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

Data plane: The data plane is comprised of network nodes, called switches, which are all connected to the control plane and interconnected by links to form the underlying network topology. The switches passively execute the instructions received from the programmable control plane.

Control plane: The control plane is in charge of all logical operations and essential functionalities of the network. Gathering NSI from the data plane, the control plane computes the routing and scheduling decisions, to be sent to the switches for packet routing and link activation.

Packet life cycle and MPLS routing: Whenever the first packet of a flow arrives at a switch, a flow request is generated by the switch and sent to the control plane, which computes the flow installation rules and scheduling decisions. The control plane then deploys the packet forwarding rules on the switches and notifies the switches adjacent to activated links to transmit packets. Subsequent packets of the same flow do not generate flow request and are routed via the Multiprotocol Label Switching (MPLS) mechanism. After computing the route for the first packet of a flow, the control plane updates the routing tables on all the participating switches and augments incoming packet with a MPLS label at the source node to which packets arrive. The switches then look up the routing tables for the entries matching the MPLS labels to forward packets.

Wireless model: The inter-switch communication corresponds to the network link activation for packet transmission, whereby wireless interference constraints 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 or enhanced coding and retransmission at the physical and link layers.

B. Distributed Wireless SDN Architecture

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

Wireless inter-controller communication: The controllers, each of which manages a sub-network domain, must communicate to exchange their local network information. The inter-controller network is separate from the underlying network.

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, 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 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 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

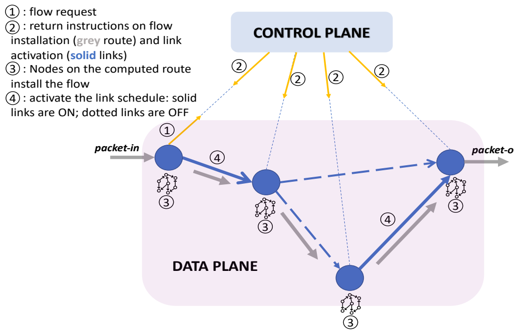
① : flow request
② : return instructions on flow installation (grey route) and link activation (solid links)
③ : Nodes on the computed route install the flow
④ : activate the link schedule: solid links are ON; dotted links are OFF
CONTROL PLANE
②
②
②
②
①
packet-in
packet-out
④
④
③
③
③
③
DATA PLANE

Fig. 1: Basic SDN architecture and packet life cycle

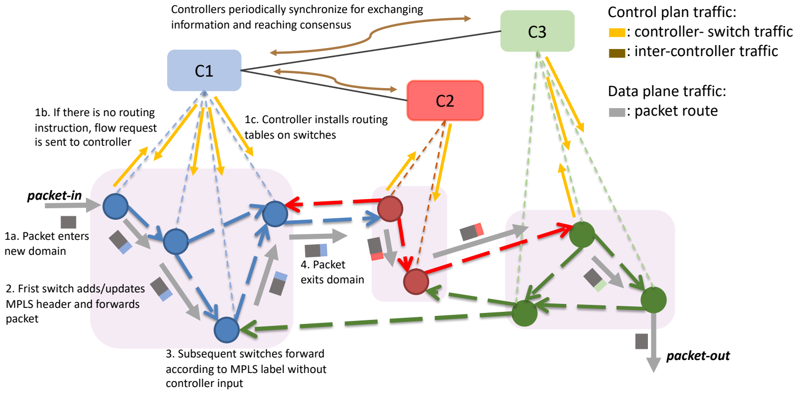
Controllers periodically synchronize for exchanging information and reaching consensus
C1
C2
C3
Control plan traffic:
▬: controller- switch traffic
▬: inter-controller traffic
Data plane traffic:
▬: packet route
1b. If there is no routing instruction, flow request is sent to controller
1c. Controller installs routing tables on switches
packet-in
1a. Packet enters new domain
2. Frist switch adds/updates MPLS header and forwards packet
4. Packet exits domain
3. Subsequent switches forward according to MPLS label without controller input
packet-out

Fig. 2: Distributed wireless SDN architecture and workflow

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.

III. Preliminaries and Problem Formulation

A. System Model

1) 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 links. For simplicity, we assume each link has capacity 1. Time is slotted. 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 $\mathcal{D}^{(c)}$. 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 at any slot t is assumed to be bounded by a finite number A_{max}. To model time-variation, we consider the model where a link can be in one of the two states, ON or OFF. 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. 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 outgoing 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 till time τ_{j+1}:

$$C_E[\tau_j] = C_E[\tau_j + 1] = ... = C_E[\tau_{j+1} - 1]. \quad (1)$$

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

$$P(C_e[\tau_j] \big| C_e[\tau_{j-1}], ..., C_e[\tau_0]) = P(C_e[\tau_j] \big| C_e[\tau_{j-1}]). \quad (2)$$

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 (1) and (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 $e \in E$:

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

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:

This captures unicast, multicast or broadcast traffic.

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

- 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:

$$\mu_e(t) = C_e[t] \cdot D_e(t) \prod_{e' \in I_e} (1 - C_{e'}[t] \cdot D_{e'}(t)), \tag{3}$$

where $\mu_e(t) = 1$ indicates that link e successfully transmits a packet at time slot t and vice versa. We impose no structural restriction on the set I_e and thus capture a wide range of practical wireless models including primary interference, k-hop interference, and protocol interference. We 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 II, 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.

2) 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 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.

B. Policy Space and Problem Statement

For any decision variable, 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 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 prescribed such route $T^{(c)}(t)$ throughout their deployment in the network.
- 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 $D^{\pi}_e(t) = 1$.

Denote by Π the set of admissible policies, which can use all past and future packet arrival information, under the distributed wireless SDN setting. 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 1. *An arrival rate vector $\boldsymbol{\lambda} = \{\lambda^{(c)}\}_{c \in \mathcal{C}}$ is supported by policy π if under the policy π:*

$$\liminf_{t \to \infty} \frac{R^{(c)}(t)}{t} = \lambda^{(c)}, \quad \forall c \in \mathcal{C}, w.p.1 \tag{4}$$

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

$$\boldsymbol{\Lambda} = \{\boldsymbol{\lambda} \in \mathbb{R}_+^{|\mathcal{C}|} : \exists \pi \in \Pi \text{ that supports } \boldsymbol{\lambda}\} \tag{5}$$

Definition 2 (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.

IV. OPTIMAL NETWORK CONTROL FOR DISTRIBUTED WIRELESS SDN

In this section, we present a network control framework for distributed wireless SDN, termed Distributed Universal Max-Weight (DUMW), and establish its throughput-optimality.

A. *Universal Max-Weight (UMW)*

We first present the throughput-optimal Universal Max-Weight (UMW) policy [5] which permits algorithmic structure directly congruent with SDN routing requirement and leverages easy-to-track virtual queues in place of physical queues. However, 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.

1) Setting of UMW: The original setting of [5] 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 III-A) for the 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. Unlike the conventional physical queues, in which the queue counter is incremented when physical packets arrive at the edge in the current time slot, the virtual queue of edge e is incremented upon a packet arrival as long as its prescribed route passes through e. Formally, at time slot t, for all $A^{(c)}(t)$ packets of class c arriving at the source node $s^{(c)}$, UMW prescribes them a route $T^{(c)}(t) \in \mathcal{T}^{(c)}$, along which these packets are routed throughout their traversal in the network. The number of packet arrivals to the virtual queue is:

$$A_e(t) = \sum_{c \in \mathcal{C}} A^{(c)}(t) \mathbb{1}(e \in T^{(c)}(t)), \quad \forall e \in E. \tag{6}$$

Then the virtual queue process evolves as:

$$Q_e(t+1) = \big(Q_e(t) + A_e(t) - \mu_e(t)\big)^+, \quad \forall e \in E. \tag{7}$$

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:

$$T^{(c)}(t) \in \underset{T^{(c)} \in \mathcal{T}^{(c)}}{\operatorname{argmin}} \Bigg(\sum_{e \in E} Q_e(t) \mathbb{1}(e \in T^{(c)}) \Bigg). \tag{8}$$

Upon the arrival of a new packet, its route is immediately computed according to (8) and fixed throughout execution, which makes UMW 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:

$$D_E(t) \in \underset{\mathbf{x} \in \mathcal{M}}{\operatorname{argmax}} \Bigg(\sum_{e \in E} Q_e(t) \cdot C_e[t] \cdot x_e \Bigg). \tag{9}$$

2) Limitations of UMW: The analytical model of UMW lacks the generality to readily be extended to distributed control. The queue dynamics (7) cannot exemplify the distributed view of the network, whereby each controller has only local network statistics and information. Moreover, the scheduling algorithm (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) in place of $\mathcal{M}$ captures interference from inter-domain links.

B. *The Virtual Queue Process of DUMW*

We develop the Distributed UMW (DUMW) framework that non-trivially extends UMW [5] to the distributed wireless SDN setting. We hereby define some notations and formally present our 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 $T^{(c)}(t)$ and keeps track of the total virtual packet arrival from the classes $c \in \mathcal{C}_i$:

$$A_e^{\pi i}(t) = \sum_{c \in \mathcal{C}_i} A^{(c)}(t) \mathbb{1}(e \in T^{(c)}), \quad \forall e \in E. \tag{10}$$

The total virtual packet arrivals from all classes are:

$$A_e^{\pi}(t) = \sum_{i=1}^{m} A_e^{\pi i}(t), \quad \forall e \in E. \tag{11}$$

Recall from Section III-A1 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 periodic synchronization, we allow the virtual queues to be updated only at the synchronization points τ_j $(j = 1, 2, \ldots)$, and obtain the τ-step evolution of the virtual queue process as:

$$\begin{aligned} Q_e(\tau_{j+1}) &= \Bigg(Q_e(\tau_j) + \sum_{t=\tau_j}^{\tau_{j+1}-1} [A_e^{\pi}(t) - \mu_e^{\pi}(t)] \Bigg)^+ \\ &\overset{(11)}{=} \Bigg(Q_e(\tau_j) + \sum_{t=\tau_j}^{\tau_{j+1}-1} [\sum_{i=1}^{m} A_e^{\pi i}(t) - \mu_e^{\pi}(t)] \Bigg)^+, \forall e \in E. \end{aligned} \tag{12}$$

It is notable that our virtual queue process generalizes that of [5], whereby for $m = 1$ domain controller and step size of $\tau = 1$, the recursion (12) reduces to the queue dynamics (7) of UMW. DUMW is then designed to stabilize the virtual queue process $\{\mathbf{Q}(t)\}_{t \geq 0}$ for any arrival rate $\boldsymbol{\lambda} \in int(\boldsymbol{\Lambda})$.

C. Controller Synchronization and Virtual Queue Estimates

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).

1) 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 (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 MAX gossip protocol [6] 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 its NSI and the local statistics, each controller D_i initializes:

$$c_e^i[0] = C_e[\tau_j] \cdot \mathbb{1}(e \in E_i), \tag{13}$$

$$s_e^i[0] = \Big(\sum_{t=\tau_{j-1}}^{\tau_j-1} \mu_e^{\pi}(t) \Big) \cdot \mathbb{1}(e \in E_i). \tag{14}$$

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:

$$C_e[\tau_j] = \max_{i \in [1,m]} c_e^i[0] \text{ and } \sum_{t=\tau_{j-1}}^{\tau_j-1} \mu_e^{\pi}(t) = \max_{i \in [1,m]} s_e^i[0], \tag{15}$$

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 [6] 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:

$$a_e^i = \sum_{t=\tau_{j-1}}^{\tau_j-1} A_e^{\pi i}(t), \quad \forall e \in E, \tag{16}$$

this problem reduces to computing across all the m controllers the global sum $\sum_{i=1}^{m} a_e^i$. We design the mechanism, termed TREE-SUM, that finds the global sum precisely under the semi-static wireless 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 $l = 1 \to d_c$ 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 the received values to its original partial values:

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

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 $O(|E_c|)$.

D. 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:

$$T^{(c)}(\tau_j) \in \operatorname*{argmin}_{T^{(c)} \in \mathcal{T}^{(c)}} \left(\sum_{e \in E} Q_e(\tau_{j-1}) \cdot \mathbb{1}(e \in T^{(c)}) \right), \tag{17}$$

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 $\mathbf{Q}(\tau_{j-1})$, they can obtain and solve the same problem (17). By imposing the same deterministic tie-breaking rules for optimizing (17), we ensure that all the controllers select the same routes $T^{(c)}(\tau_j)$. 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.

E. Inter-Domain Scheduling

The inter-domain scheduling problem can be characterized as scheduling with heterogeneously delayed NSI. Though sharing similarities with the literature [7], the system within our interest is more generalized: [7] can be viewed as a special case where every domain only has one node. 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 [7] where each node makes independent decisions. We present our inter-domain scheduling policy, termed SCHEDULE, in Algorithm 1. At time slot $t \in [\tau_j, \tau_{j+1})$, each controller D_i formulates the same optimization problem from the common information, which includes the delayed global NSI $C_E[t-\tau]$ and the virtual queue $\mathbf{Q}(\tau_{j-1})$; under the same deterministic tie-breaking rule, all the controllers obtain the mutual optimal solution $\mathbf{x}^*$. The SCHEDULE algorithm is novel and the first optimal scheduling policy for the setting of heterogeneously delayed NSI *with* hierarchy. The only applicable algorithm in the literature [8] does not leverage fresh local NSI and is thus sub-optimal.

```
Algorithm 1: SCHEDULE
  Input: $t$, $D_i$, $C_{E_i}[t]$, $C_E[t-\tau]$, $\mathbf{Q}(\tau_{j-1})$
  Output: Link activation vector $D_{E_i}(t) \in \{0,1\}^{|E_i|}$ for domain $D_i$.
  Form the view of fresh local NSI as $C_{E_i}[t] = \gamma_{E_i}$ and delayed global NSI as $C_E[t-\tau] = \alpha$.
  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:
    $$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmax}} \Big\{ \sum_{\beta \in \{0,1\}^{|E|}} P(C_E[t] = \beta | C_E[t-\tau] = \alpha) \times \sum_{e \in E} Q_e(\tau_{j-1}) \beta_e x(e, \beta_{E_{k_e}}, \alpha) \prod_{e' \in I_e} (1 - \beta_{e'} x(e', \beta_{E_{k_{e'}}}, \alpha)) \Big\}$$
  Set $D_e(t) = x^*(e, \gamma_{E_i}, \alpha), \quad \forall e \in E_i$.
  Return the link activation vector $D_{E_i}(t)$
```

F. The DUMW Framework and Throughput-Optimality

We depict the full DUMW framework in Algorithm 2 and proceed to establish the throughput-optimality of DUMW.

```
Algorithm 2: 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 (13), (14), (16).
        [Synchronization] Start the new round $\tau_j$ and retrieve from the previous round $\tau_{j-1}$:
          $C_E[\tau_{j-1}], \boldsymbol{\mu}, \widetilde{\mathbf{A}}) = \text{SYNC}(\tau_j, D_i, \mathbf{c}^i[0], \mathbf{s}^i[0], \mathbf{a}^i[0])$
        Update the virtual queues as (12).
        [Flow installation] Solve for $T^{(c)}(\tau_j), \forall c \in \mathcal{C}$ from (17) 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
```

Theorem 1. *DUMW is throughput-optimal.*

Theorem 1 is derived by first 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. $\limsup_{K \to \infty} \frac{1}{K} \sum_{j=0}^{K-1} \sum_{e \in E} \mathbb{E}[Q_e(\tau_j)] < \infty$. The key components of the proof leverage the bounded queue delay and the optimality of our novel SCHEDULE algorithm to obtain the strong stability of virtual queues, which provably implies the stability of physical queues and thus the throughput-optimality.

V. 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. Our setting assumes fully-connected inter-controller topology G_c, node-exclusive wireless interference constraints, unit link capacity, and unicast traffic, specified by a source-destination (s-d) pair with Poisson arrivals following the same packet generation rate λ. For abbreviation, we denote the link statistics by $P_e(a|b) = P(C_e[t] = a | C_e[t-\tau] = b)$ and consider the dynamic network with $P_e(1|0) = P_e(1|1) = 0.5$. We consider the 2×3 grid decomposed into three domains (Figure 3) with two s-d pairs (1, 6) and (5, 6), and $\tau = 50$ in order to demonstrate the noticeable throughput gain of DUMW even on this moderate scale.

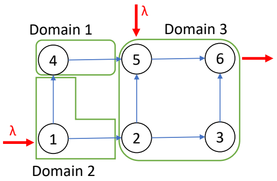

Fig. 3: The tested 2×3 grid is decomposed into three domains.

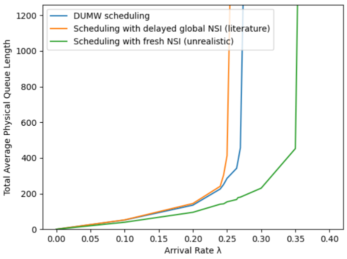

Fig. 4: DUMW notably gains throughput by leveraging fresh local NSI.

Throughput-optimality of DUMW: We consider the centralized UMW [5] as an *unrealistic* baseline. The only applicable algorithm in the literature [8] only uses the delayed global NSI $C_E[t-\tau]$ and is thus sub-optimal in throughput. As illustrated in Figure 4, DUMW gains noticeable throughput improvement compared to the literature by leveraging the fresh local NSI.

References

[1] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, “A survey of software-defined networking: Past, present, and future of programmable networks,” *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[2] Q. M. Nguyen, M. S. Rahman, X. Fu, S. Kompella, J. Macker, and E. H. Modiano, “An optimal network control framework for wireless sdn: From theory to implementation,” in *MILCOM 2022 - 2022 IEEE Military Communications Conference (MILCOM)*, 2022, pp. 102–109.

[3] F. Bannour, S. Souihi, and A. Mellouk, “Distributed sdn control: Survey, taxonomy, and challenges,” *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 333–354, 2018.

[4] A. Panda, W. Zheng, X. Hu, A. Krishnamurthy, and S. Shenker, “Scl: Simplifying distributed sdn control planes,” in *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI’17. USA: USENIX Association, 2017, p. 329–345.

[5] A. Sinha and E. Modiano, “Optimal control for generalized network-flow problems,” *IEEE/ACM Transactions on Networking*, vol. 26, no. 1, pp. 506–519, 2018.

[6] B. M. Nejad, S. A. Attia, and J. 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*, 2009, pp. 1–7.

[7] A. A. Reddy, S. Banerjee, A. Gopalan, S. Shakkottai, and L. Ying, “On distributed scheduling with heterogeneously delayed network-state information,” *Queueing Syst. Theory Appl.*, vol. 72, no. 3–4, p. 193–218, dec 2012. [Online]. Available: https://doi.org/10.1007/s11134-012-9312-z

[8] L. Ying and S. Shakkottai, “On throughput optimality with delayed network-state information,” *IEEE Transactions on Information Theory*, vol. 57, no. 8, pp. 5116–5132, 2011.