Optimal Network Protocol Selection for Competing Flows via Online Learning

Xiaoxi Zhang, *Member, IEEE,* Siqi Chen, *Student member, IEEE,*
Yunfan Zhang, Youngbin Im, *Member, IEEE,* Maria Gorlatova, *Member, IEEE,*
Sangtae Ha, *Senior Member, IEEE,* and Carlee Joe-Wong, *Member, IEEE*

Abstract—Today's Internet must support applications with increasingly dynamic and heterogeneous connectivity requirements, such as video streaming and the Internet of Things. Yet current network management practices generally rely on pre-specified network configurations, which may not be able to cope with dynamic application needs. Moreover, even the best-specified policies will find it difficult to cover all possible scenarios, given applications' increasing heterogeneity and dynamic network conditions, e.g., on volatile wireless links. In this work, we instead propose a model-free learning approach to find the optimal network policies for current network flow requirements. This approach is attractive as comprehensive models do not exist for how different policy choices affect flow performance under changing network conditions. However, it can raise new challenges for online learning algorithms: policy configurations can affect the performance of multiple flows sharing the same network resources, and this performance coupling limits the scalability and optimality of existing online learning algorithms. In this work, we extend multi-armed bandit frameworks to propose new online learning algorithms for protocol selection with provably sublinear regret under certain conditions. We validate the optimality and scalability of our algorithms through data-driven simulations and testbed experiments.

Index Terms—Network protocol selection, online learning, multi-armed bandits, online algorithms

1 Introduction

The Internet today is diversifying in terms of both the applications and devices that it aims to support, as well as the means for doing so. Applications like augmented reality, for instance, require increasingly low latencies [2], while the Internet-of-Things has dramatically expanded the range of devices connected to the Internet [3]. Fifth-generation (5G) wireless networks are simultaneously predicted to integrate several different access frequencies in an effort to boost capacity and coverage [4], [5]. To support devices and applications with different quality-of-service (QoS) requirements, today's 5G radio access networks (RAN) group the user applications into different service categories, i.e., eMBB, mMTC, and URLLC, so that distinct radio resources can be reserved (e.g., via network slices) in isolation from the allocation for devices served by other service classes [4], [6], [7]. Yet it is still far from clear how the network can enforce flow-specific heterogeneous requirements for each application sharing limited resources on heterogenous network links, e.g., application flows supported by the same service class traversing the 5G RAN and/or the backhaul networks [8].

Current network management practices generally rely on static pre-configurations, e.g., pre-specifying the routing algorithms for each router used to determine the paths that flows take in a network [9] or allowing users to specify the routes themselves (e.g., custom routes in Microsoft Azure's virtual networks [10]). Initiatives like network functions virtualization (NFV) and the RAN (radio access network) Intelligent Controller [11] aim to enable more flexible policies, but they still require manual intervention to change the preset network policies [12]. Moreover, it is unclear which policies can actually optimize over all possible scenarios and combinations of flow requirements. Comprehensive models for flow performance under different protocol choices often do not exist or are difficult to adapt to newly developed protocols, as we detail in example use cases below.

In this work, we recognize that pre-specified policies are likely insufficient to handle all possible scenarios. Instead, dynamically network management policies that can adapt to the non-stationary protocols' efficiency in processing flows and the changing network environment will improve the network resource utilization and overall gain of co-existing application flows. While a vast array of possible policies at various layers of the stack can be used for different types of networks, we focus on optimizing the configuration of protocols for flows on each link of a given network. The decision variables of the protocol configuration can include the types of pre-configured protocols (e.g., TCP/UDP, CSMA/ALOHA, or a priority class) and/or parameters of a given type of protocol (e.g., sending rate of TCP or backoff

- *X. Zhang is with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, Guangdong 510006, China, e-mail: zhangxx89@mail.sysu.edu.cn (Xiaoxi Zhang is the corresponding author.)*
- *Siqi Chen and Sangtae Ha are with the Department of Computer Science, University of Colorado Boulder, Boulder, CO 80309, USA, e-mail: {siqi.chen, sangtae.ha}@colorado.edu*
- *Yunfan Zhang and Maria Gorlatova are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA, email: {yunfan.zhang, maria.gorlatova}@duke.edu*
- *Youngbin Im is with the School of Electrical and Computer Engineering, Ulsan National Institute of Science and Technology, Ulsan, South Korea, e-mail:ybim@unist.ac.kr*
- *Carlee Joe-Wong is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA, e-mail: cjoewong@andrew.cmu.edu*

Manuscript received April xx, 2020; revised June 30, 2021.

time of a wireless channel access protocol). We suppose that a given set of protocol configurations is available and that each flow's achieved performance depends on the configuration combination chosen for all flows on shared links, in an unknown manner that depends on prevailing link conditions. Unfortunately, there are also no analytical flow resource competition models showing this dependency or theoretically performance-guaranteed online algorithms that can dynamically optimize protocols on each link with time-varying unknown network characteristics (e.g., bandwidth capacities). In this work, we develop new, model-free algorithms that learn the unknown relationship between protocol choices and flow performance under the presence of the above-mentioned time-varying inputs. Finally, we analytically and experimentally demonstrate that they learn the optimal assignment of protocols to flows over time.

1.1 Use Cases: Optimized Protocol Selection

Different types of protocols may be selected by different entities in a network, whether set on a link-by-link basis by intermediate routers for all flows on the same link (e.g., choosing a MAC protocol for a wireless link), or by each flow's source for its entire path (e.g., TCP/UDP protocols). We consider both scenarios, as exemplified by the use cases below (discussed in more detail in Table 1 of Section 2).

Configuration of 5G wireless networks is likely to require machine learning algorithms that can manage their high degree of configuration complexity [5]. For instance, one might determine the size of an uplink control channel between each UE (user equipment) and a cellular base station. While a larger control channel leads to less spectrum available for data transmission, 5G control channels may carry significant amounts of information about a UE's current condition that might be used to further optimize its service, e.g., with custom beamforming or scheduling. The control channel size would be determined individually for each UE, i.e., on a per-flow basis if each UE is associated with a single flow on the network.

MAC-layer protocol selection for individual flows naturally takes place on a link-by-link basis, and encompasses protocols such as error correction modes, sensor duty cycles, or re-transmission rates for dropped packets [13]. While models exist to analyze the performance of MAC protocols, the actual realized performance may be hardware-specific and thus difficult to predict in practice [14]. Moreover, new protocols not covered by current models may be required to handle the requirements of emerging applications like smart grids or cities [15]. Automated protocol selection can lessen the need for precise analytical models in such scenarios.

Network slicing, envisioned for next-generation 5G architectures [7], reserves shares of network resources for flows with strict performance requirements, e.g., AR and VR (augmented and virtual reality) applications. Choosing the right configuration to support these requirements, however, involves complex tradeoffs. For instance, allowing slices to share some physical resources makes the network more efficient but risks interference from other slices [7]. Flows that share a slice may also attempt to further optimize their performance, e.g., AR and VR applications may use protocols that are specialized to particular network conditions, including rate adaptation, caching, and transport control specializations [2]. Automated network management is one solution to the resulting configuration complexity. These protocols could be set by the AR/VR devices or by nodes or servers within the network. We validate our proposed algorithms in experiments on TCP protocol selection for AR applications in Section 5.2.

Transport-layer protocol optimization is a long-standing challenge that exemplifies the value of our learning-based approach. Indeed, we validate our algorithms with transport-layer protocol selection experiments in Section 5. It is well known that choosing different transport-layer protocols will affect flow performance, e.g., inducing different bandwidth allocations [16], but no comprehensive theoretical models exist for the bandwidth allocation when flows use an *arbitrary* combination of protocols, which can make it difficult to encode the optimal protocol selections a priori. Existing models may also exclude new congestion control protocols that propose to handle various network conditions and application requirements, further motivating a model-free learning approach [17]. In this scenario, the protocols are set by the flow sources or network proxies at intermediate nodes [18], [19], [20].

These examples fall into three types of protocol selection: *per-link* selection, or choosing the same protocol for all flows on a link (MAC protocols), *per-link and per-flow* selection, or choosing a protocol for each flow on a given link (wireless control channel size), and *per-flow* selection, choosing a protocol for each flow on its entire path through the network (TCP/UDP, network slicing). In this work we address the challenges that arise in all three selection types.

1.2 Our Contributions

We derive and validate, analytically and empirically, the *first algorithms that can learn the optimal assignment of network protocols that maximizes aggregate flow performance*. We define "performance" as the total flow completion time, but our framework generalizes to other objectives. In doing so, we solve several technical challenges:

- **Unpredictable, time-varying network conditions.** The exact relationship between protocol selection and flow performance depends on the presence of background traffic (for which we do not control the protocols used) and network conditions: for example, conventional TCP may not perform well on an unreliable wireless link [21], and protocol performance on an overlay link may be affected by (unknown) changes in the underlying physical topology. Since the exact network conditions may not be known, our algorithm must be able to dynamically manage the network flows arriving at different future time points by optimizing the protocol selection.
- **Competition between different flows.** When different protocols may be used for different flows, the choice of protocol for one flow affects the performance of other flows on the same link. This coupling is particularly hard to manage given the absence of models for how protocol choices affect flow performance and the nonlinearity of our objective with respect to the protocol decisions.

- **A large set of protocol choices.** The total number of possible protocol assignments to flows is exponential with the number of active flows on each link of a given network. Thus, we cannot sample all possible protocol assignments if directly following the existing online learning approaches, such as multi-armed bandit (MAB) framework [22]. Instead, we exploit the properties of the optimal protocol selection to reduce the sampling complexity.

We take the first steps towards meeting these challenges through the following **technical contributions**.

First, we formulate a total flow completion time minimization problem, which reveals the non-linearity of our objective function w.r.t. the decision variables (protocols) over the entire life-time of each flow and the dependency between the chosen protocols for different flows at any given time. This formulation is general enough that it can cover multiple use cases summarized in Table 1. We then construct a stochastic protocol interaction model and formulate the per flow transmission rate achieved by each protocol into three intuitive multiplicative parts that can be arbitrary and unknown: a bandwidth partition weight of the applied protocol, the total bandwidth utilization of the protocol combination, and the total available bandwidth capacity of our controlled flows. This model provides a foundation to understand the underlying effects of volatile network conditions on the protocol performance (see our first challenge) and models the interactions between different network flows (our second challenge).

Second, leveraging the above performance model, we propose a new extension of the multi-armed bandit (MAB) framework. We show that per-link protocol selection falls into conventional MAB models, but that per-flow protocol selection does not. A natural way to handle per-flow protocol selection is to view the selection of each feasible combination of protocols chosen for all flows in the network as an "arm" to be pulled. However, conventional MAB algorithms scale poorly with the number of arms (i.e., combinations of protocols); and they do not account for (1) a completion time objective, which is a highly nonlinear function of the protocol performance that also depends on past protocol selections, or (2) constraints (from bandwidth capacities) on the achievable reward that may be arbitrary and unknown. We extend the conventional MAB algorithm called UCB in Section 3.2 into a new online algorithm that can overcome the above challenges. The key idea is to select arms from a largely reduced set of protocol combinations by exploiting the properties of the optimization problem and to carefully decode each arm's "reward" from the transmission rates achieved by all present flows.

Third, we show in Section 4 that our algorithm asymptotically minimizes the total flow completion time under reasonable assumptions. Specifically, for the case where the total bandwidth utilization on any given link is uniform (independent of the decision variables) and the performance of different protocols are quantitatively upper-bounded, we prove that the *regret*, i.e., expected gap between our algorithm against the offline optimum, is logarithmic in the time horizon (Theorem 4.1) or the number of flows which arrive and depart onto a path at arbitrary times (Corollary 4.2). The regret is then improved for two other cases with different additional assumptions (Theorem 4.3 and 4.4). For the case with non-uniform bandwidth utilization, while the regret for arbitrary flow arrivals is intractable, we analyze the regret of a special case with at most two flows alive simultaneously (Theorem 4.5). These theoretical results are based on non-trivial extension of the analysis of existing MAB algorithms and online scheduling algorithms [23].

Finally, in Section 5, we validate our algorithms' optimality and scalability with experiments on transport protocol selection, showing that we achieve a 20% to 60% improvement over heuristic selection algorithms. To do so, we implement our learning algorithm on an Amazon EC2 testbed and a computing cluster at Duke University.

2 Problem Formulation

To formalize our problem, we consider a communication network with a pre-determined topology described by a graph $G(\mathcal{V}, \mathcal{L})$, where $\mathcal{V}$ represents the nodes, corresponding to the locations of routers in the network, and $\mathcal{L}$ represents the links. We divide time into discrete slots, each of which lasts $\epsilon > 0$ seconds. For our use cases, ϵ would likely be on the order of tens of seconds (cf. our transport protocol experiments in Section 5.2). Note that the topology may be a simple star topology of multiple devices connecting to a wireless base station, or a more complex overlay network in which the "links" are virtual ones with unknown underlying physical topologies (cf. Table 1).

Suppose n flow requests arrive at the network over the lifetime of the system, T. Let t_i denote the arrival time of each flow $i \in [n]$. Flows arriving within a time slot are scheduled at the beginning of the next time slot. Each flow i has a source and destination in $\mathcal{V}$, and a fixed path $\mathcal{P}_i$ that is exogenously determined at the time of its arrival. Each $\mathcal{P}_i$ consists of a set of nodes ($\in \mathcal{V}$), including a source and a destination node; and a set of links ($\in \mathcal{L}$) connecting them. Each flow i also has a fixed size π_i, *i.e.,* the amount of data (in bytes), to transfer along $\mathcal{P}_i$. The flows, for instance, could be downloads of files like computation results or AR holograms. We assume the flow sizes are known when the protocols are chosen, which is reasonable if a node assigns protocols for its existing incoming or outgoing flows [18], [19], [20]. On each link, there are also flows with unknown data sizes in the background that we cannot control, which could compete for the bandwidth with our controlled flows $i = 1, \cdots, n$ when they co-exit. Due to the presence of background traffic, each link l has an unknown bandwidth capacity, denoted by B_{lt}, (bps) in each time slot t.

Protocol selection. Our algorithm chooses the protocols for each flow i on its path, $\mathcal{P}_i$. To accommodate the per-link, per-flow, as well as per-link and per-flow protocol selection types discussed in Section 1.1, we suppose we have a total of M protocol choices on each link for each flow. Let x_{ilmt} denote a binary indicator variable, representing whether protocol $m \in [M]$ is chosen ($x_{ilmt} = 1$) for flow i on link l at time t, or not ($x_{ilmt} = 0$). Since each flow can use only one protocol on each link at a time, we constrain $\sum_{m\in[M]} x_{ilmt} = 1,\ \forall l \in \mathcal{P}_i, i \in [n], \forall t \in [T]$. We can similarly enforce choosing the same protocol for all flows on a given link or all links on a given flow's path. As

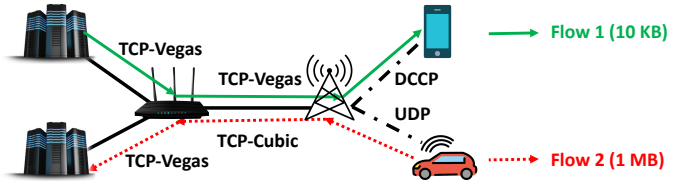


Fig. 1: A Simple Example of Selected Protocols

shown in Figure 1, in a simple IoT network with hybrid networks represented by black network edges: wired links between the servers, gateway, and base-station; a wireless link from the base-station to either mobile device. Suppose there are two flows, shown as Flow 1 and Flow 2 with the respective flow sizes in bytes, co-existing at the network with a shared link between the gateway and base-station. Flow 1 is a transfer of a small data batch requested from a VoIP application run on the mobile phone while Flow 2 transfers a large dataset collected by a sensing application run on a vehicle. We might want to apply DCCP and UDP on Flows 1 and 2 on the wireless links respectively and choose different TCP variants for the two flows at the wired links depending on our estimated transmission rates achieved by each flow under their applied protocols and our optimization goal, e.g., we want the smaller flow to be processed as fast as possible rather than aiming at a high extent of fairness which should be guaranteed by using the same TCP protocol on the shared link instead. Such hop-by-hop protocol implementation is an example of per-link and per-flow protocol selection type in Section 1.1 and can be realized using proxies as we discuss shortly after. In addition, if the network characteristics change in future time points, the above protocol combination may need to be changed dynamically.

We next briefly discuss how our formulation relates to the four use cases discussed in Section 1.1. Table 1 summarizes the network topology and example protocol choices considered in each use case. We assume that continuous protocol choices, e.g., the 5G control channel size and resources allocated to flows in a slice, are discretized.

In our 5G and MAC scenarios, we can view a "link" as a single wireless link to a base station, shared between several UEs connected to this base station. Network slice configurations and transport protocols, on the other hand, generally are not changed at every physical link in a flow's path, but can be changed at some intermediate points. For instance, network slices that cross multiple domains might need different configurations within each domain or even on wired vs. wireless links [7] to enforce slice performance requirements. To capture these choices, we can view each domain as a "link," similar to links in overlay networks where the overlay link may in fact consist of an arbitrary physical network topology. Transport-layer protocols can similarly be deployed on a link-by-link basis in overlay networks via network proxies that forward packets on behalf of the sender according to the best transport-layer protocol for the next link, e.g., as Dropbox and Google do in their datacenter networks [24], [25] or as may be needed for edge computing applications where latency-sensitive traffic

TABLE 1: Outline of formulation use cases.

Use case	Topology	Example protocols	Selection type
5G	Single link	Control channel size	Per-flow, Per-link
MAC	Single link	CSMA/CD, CSMA/CA	Per-link
Network slicing	Arbitrary	Slice resource reservation, priority class	Per-flow, per-link over domains
Transport layer	Arbitrary	TCP CUBIC, TCP Reno, UDP	Per-flow, per-link w/ proxies

is sent over a wireless link to an edge server proxy and then into the Internet [26], [27]. In fact, we show in Section 5.2 that using different transport protocols on different links can substantially improve performance.

Link transmission delay $\delta_{il}(\mathbf{x}_l)$. Let $r_{ilt}(\mathbf{x}_{lt})$ denote the effective transmission rate achieved on link l for flow i at time slot t, which is revealed after we choose the protocols on link l. Let $\mathcal{A}_{lt}$ denote the set of alive flows on link l at time t. As different flows may share one or more links and compete for the bandwidth on each link, $r_{ilt}(\mathbf{x}_{lt})$ is a function of the decisions x_{jlmt} over all protocols m and for all flows $j \in \mathcal{A}_{lt}$. Let $\delta_{il}(\mathbf{x}_l)$ denote the transmission delay on each link l of flow i incurred by transferring a total of π_i bytes of data, which is defined as $\delta_{il}(\mathbf{x}_l) = \max_{t_i \le t \le +\infty} t \cdot \mathbf{1}(\sum_{t_i \le t' \le t} \epsilon r_{ilt'}(\mathbf{x}_{lt'}) \le \pi_i)$, where $\mathbf{1}(X)$ equals 1 if X is true; and 0 otherwise.

Total completion time minimization. Let $\tau_i(\mathbf{x})$ denote the completion time of flow i, *i.e.*, its arrival time plus the total delay of serving flow i. We have $\tau_i(\mathbf{x}) = t_i + \max_{l \in \mathcal{P}_i} \delta_{il}(\mathbf{x}_l)$ + propogation and queueing delay. Since our protocol choices do not affect the propagation and queueing delay, minimizing the flow completion time is equivalent to minimizing the bottleneck transmission delay $\max_{l \in \mathcal{P}_i} \delta_{il}(\mathbf{x}_l)$. We therefore formulate an online optimization problem as:

$$\text{Minimize}_{\mathbf{x}} \quad \sum_{i \in [n]} \max_{l \in \mathcal{P}_i} \delta_{il}(\mathbf{x}_l) \tag{1}$$

$$\text{subject to:} \sum_{i \in \mathcal{A}_{lt}} r_{ilt}(\mathbf{x}_{lt}) \le B_{lt}, \ \forall l \in \mathcal{P}_i, i \in [n], t \in [T] \tag{2}$$

$$\sum_{m \in [M]} x_{ilmt} = 1, \ \forall l \in \mathcal{P}_i, i \in [n], t \in [T] \tag{3}$$

The objective (1) indicates that we attempt to minimize the total completion time of n flows, subject to a bandwidth capacity constraint (2) on each link in each time slot; the last constraint ensures only one protocol is chosen for each flow on each link at each time. We can impose additional constraints such as $x_{ilmt} = x_{jlmt}, \ \forall i, j, l, m, t$ to ensure that flows sharing a link must use the same protocol, modeling the per-link selection type.

This formulation reflects our research challenges: both $\delta_{il}(\mathbf{x}_l)$ and $r_{ilt}(\mathbf{x}_{lt})$ are unknown, time-varying and non-

TABLE 2: Notation Table

$i \in [n]$	index of flows
$m \in [M]$	index of protocols
$\mathcal{P}_i$	path that i traverses
x_{ilmt}	choose (=1) m for i on link l at t or not (=0)
π_i	size (in bits) of flow i
$\delta_{it}(\cdot)$	transmission delay of i on l
$\tau_i(\cdot)$	completion time of flow i
B_{lt}	bandwidth capacity of l at t
$r_{ilt}(\cdot)$	transmission rate achieved by i on l at t
$w_{lt}(\cdot)$	weight in achieving rates of protocol l at t
$u(\mathbf{x}_{lt}, B_{lt})$	total bandwidth utilization of flows on l at t

linear functions of the protocol choices of the coexisting flows of flow i, accounting for competition between flows, and both r_{ilt} and B_{lt} are unknown. Indeed, the bandwidth capacity B_{lt} on each link may even be arbitrarily chosen by the environment without any statistic patterns over time. Thus, the problem is complex and difficult to solve. In some cases, prior models exist for r_{ilt}, e.g., network utility maximization (NUM) frameworks for certain MAC and TCP protocol variants [28], [29]. However, as noted in Section 1.1's use cases, such models are not available for all protocols and may not be able to handle unknown bandwidth capacities, so in our formulation we assume the r_{ilt} are unknown random variables. To further help with the readability of the formulation and the algorithm design in the following section, we list key notation in Table 2.

3 Online Protocol Selection

Our key insight in solving the optimization problem (1) – (3) is to understand and exploit the relationship among transmission rates, joint protocol decisions of flows, and bandwidth capacities. As existing works do not explicitly model these relationships, we ask: Is there a pattern in the transmission rates of flows using different protocols on a link? If so, can we formulate the transmission rate functions $r_{ilt}(\mathbf{x}_{lt})$ in a general form that can be learned from historical observations of the transmission rates? How can we use these learned functions to design a scalable online algorithm for protocol selection, without knowing the bandwidth capacities?

Driven by these questions, we formulate a general stochastic model to characterize the transmission rate fluctuations in Section 3.1. We then adapt a multi-armed bandit approach to predict the parameters of transmission rate functions for each protocol in Sections 3.2 and 3.3. These predictions interact with an outer online protocol selection algorithm: the choices of which protocols to predict are driven by the goal of minimizing the overall completion time. Section 4 shows that our algorithms are asymptotically optimal under reasonable assumptions. We show that our algorithm empirically outperforms heuristics in more general scenarios in Section 5.

3.1 A Stochastic Model of Protocol Interactions

We first define a model of how the link bandwidth is divided between coexisting flows, depending on the protocols they use. As shown in (4) below, the total transmission rate achieved by all flows on a link will be the bandwidth capacity B_{lt} scaled by a utilization ratio $u(\mathbf{x}_{lt}, B_{lt})$ (≤ 1) which depends on B_{lt} and the protocol combination ($\mathbf{x}_{lt}$) used on l in t; the transmission rate achieved by each flow will be proportional to the *weight* of its corresponding protocol. Let e_m denote a singleton representing that only protocol m is selected. To account for fluctuations in the transmission rates achieved by each protocol, we assume that on each link l, the weight vector $(w_{lt}(e_1), \cdots, w_{lt}(e_M))$ is i.i.d. drawn from an unknown distribution, e.g., due to fluctuations in wireless signal strength or routing in an overlay network. Non-stationary distributions of the weight vector with a bounded number of abrupt change points can be handled based on existing methods and will be discussed in Section 3.2. We can then find the transmission rate for each flow on each link:

$$r_{ilt}(\mathbf{x}_{lt}) = \frac{w_{lt}(\vec{x}_{ilt})u(\mathbf{x}_{lt}, B_{lt})B_{lt}}{\sum_{i' \in \mathcal{A}_{lt}} w_{lt}(\vec{x}_{i'lt})} \quad (4)$$

$$\text{where: } u(\mathbf{x}_{lt}, B_{lt}) \leq 1, \forall \mathbf{x}_{lt}, l \in \mathcal{P}_i, i \in [n], t \in [T] \quad (5)$$

Our model for r_{ilt} is quite general, as we do not impose any conditions on the distributions of the weight vector aside from requiring that (4) is well-defined. The weight parameters reflect the relative difference of each protocol in competing for the bandwidth capacity, compared with other protocol choices. In a practical system, the transmission rate must be jointly affected by more than one factors, such as the bandwidth capacity of the link, the number of application flows co-exist on the same link, and also the relative difference of the protocols themselves, e.g., different congestion avoidance parameters and congestion control strategies within the protocols. We separate the transmission rates into different factors for better tractability to reasoning the impact of different factors that affect our transmission rates. We next explain how it captures the relationships between the protocol choices and achieved flow rates in Table 1's use cases.

In per-flow protocol selection (control channel size, network slicing, and transport layer protocols), both the weights and utilization are affected by the protocol choices. For example, the control channel sizes chosen by all flows ($\mathbf{x}_{lt}$) determine the overall utilization $u(\mathbf{x}_{lt}, B_{lt})$, which represents the total amount of resources available to transmit payload data. The weights $w_{lt}(\vec{x}_{ilt})$ for each flow i, which depend on the control channel size chosen for each flow, would then model the resources allocated to that particular flow, given the information conveyed in its control channel. In network slicing, the "protocol" would be a configuration for a given slice; e.g., the resources reserved for it, in which case the weights $w_{lt}(\vec{x}_{ilt})$ would simply be the fractions of resources allocated to each flow; or the priorities assigned to slices that share wireless PHY/MAC resources [7], which would then determine the weights $w_{lt}(\vec{x}_{ilt})$ according to the MAC scheduling algorithm used for that link (e.g., weighted proportional fairness). Finally, in the transport layer use case the weight model is consistent with NUM models of bandwidth allocation but can capture more TCP variants [28], [30]. The utilization factor can account for known effects where different combinations of TCP protocols utilize

Algorithm 1: Online Protocol Selection via Learning Bandwidth Competition – OPSBC

Input: $G(\mathcal{V}, \mathcal{L})$, n, α
Output: $\mathbf{x}$
Initialize: $\mathbf{x} = \mathbf{0}$, $\eta^{LCB} = \mathbf{1}$, $t = 0$
while *time slot* $1 \leq t \leq T$ *starts* **do**
  for *each link* $l \in \mathcal{L}$ **do**
    Update $\tilde{\pi}_{ilt}$ = the remaining size of each alive flow i;
    Update $i^* = \text{argmin}_{i \in \mathcal{A}_{lt}} \tilde{\pi}_{ilt}$;
    /* The flow with the smallest remaining size on l */
    Choose $(m^*, m^b) = \text{argmin}_{(m,m')} \eta_{lt}^{LCB}(m, m')$;
    /* Choose the best protocol combination on l */
    Update $x_{i^* l m^* t} = 1, x_{i l m^b t} = 1, \forall i \neq i^*$;
    /* Given (m^*, m^b), choose the dominant protocol m^* for flow i^* and the inferior protocol m^b for other alive flows on l */
    Update $\eta_{lt}(m, m')$ and $\eta_{lt}^{LCB}(m^*, m^b)$ using (7) ;
  end
end

the link to greater or lesser degrees [18], [19], [20], [31]. Fluctuations in the weights can model changes in how the flows interact on the underlying topology of an overlay link, which is likely not observable in practice.

In per-link selection where all flows on a link share the same protocol, the weights $w_{lt}(\vec{x}_{ilt})$ of the protocols chosen for each flow cancel out as they are the same for all flows. Thus, the choice of protocols only affects the achieved rate $r_{ilt}(\mathbf{x}_{lt})$ in (4) through the total link utilization $u(\mathbf{x}_{lt}, B_{lt})$. Utilization is, for instance, a standard metric for MAC protocol evaluation [13], [32] as it measures how the protocol handles potential collisions. The capacity B_{lt} may be thought of as a theoretical maximum link throughput.

3.2 Protocol Selection with Uniform Utilization

In this section, we assume uniform utilizations $u(\mathbf{x}_{lt}, B_{lt})$ for all protocol choices but allow B_{lt} to be adversarially chosen (*arbitrarily variant over time and across links*). We generalize the resulting algorithm to non-identical $u(\mathbf{x}_{lt}, B_{lt})$ in Section 3.3. We only consider per-flow selection in this section, as all protocol choices yield the same performance in per-link selection when we assume uniform utilization.

Algorithm intuition. If the distributions of the weight vectors w_{lt} and the current capacities B_{lt} are known, the problem (1–3) becomes a pure online decision making problem, where we do not know the arrival times and sizes of future flows and future bandwidth capacities. We then have the following intuition: Minimizing the total flow time (1) is equivalent to minimizing the number of alive flows at each time. Therefore, at any time, the offline optimum would make the flow with the shortest remaining time finish first so as to reduce the number of alive flows, as formally shown in the proof of Theorem 4.1 in our technical report [33]. Therefore, we propose to distributedly and greedily choose the protocols on each link at each time so as to minimize the remaining time of the flow with the smallest amount of untransferred data on the link. This strategy has the advantage of independently running on each node's router (or even on each link), simplifying its deployment.

Algorithm 1 formalizes our intuitions; we show its optimality in Section 4. Here, T represents the total number of time slots from the arrival of the first flow until the completion of the last (nth) flow, i.e., $T = \frac{\tilde{T}}{\epsilon}$, where $\tilde{T}$ is the makespan of the flows. Note that $\tilde{T}$ does not depend on our protocol choices when the bandwidth is fully utilized and the flows have fixed sizes. Algorithm 1 does not require knowledge of T, but T will affect the algorithm performance as shown in Section 4. In each time slot t (every ϵ seconds), our algorithm has two parts: 1) **learning**: independently predicting the weight vector on each link based on historical samples; and 2) **protocol selection**: on each link, choosing the protocol with the biggest estimated weight for the *shortest alive flow* and the one with the smallest estimated weight for all the other alive flows. The protocol selection is triggered for all flows in the network at that time. One can also run the algorithm less frequently by only selecting protocols when a new flow arrives or an existing flow completes, although this will delay the algorithm's convergence towards the optimum (cf. Corollary 4.2).

Distributed MAB algorithm for predictions. To predict the weight vector, we utilize the MAB framework, where in each round the learner chooses to play one of a set of arms and then observes the reward resulting from this choice. However, our protocol decisions do not directly map to arms. If we call a protocol decision vector for all flows an "arm," we obtain $M^{|\mathcal{L}| \times |\mathcal{A}|}$ arms ($|\mathcal{A}|$ represents the largest number of coexisting flows), which is too large to effectively sample. However, if we consider each individual protocol decision on each link as an arm and the corresponding weight as the reward, we cannot directly interpret the protocol weights as rewards. The reason is that we can only observe the rates for each flow, r_{ilt}, at each time and thus must infer the protocol weights from the r_{ilt} observations. However, this inference is challenging when the bandwidth capacities can vary over time, because the rates do not translate into weights for protocols that are *present at different times under time-varying bandwidth capacities.* For instance, for a given link, we might observe $10Mbps$ and $20Mbps$ achieved by protocols 1 and 2 at the first time and $50Mbps$ and $500Mbps$ achieved by protocols 3 and 4 at the next time. Since we only observe a subset of the available protocols at each time, we cannot interpret $(\frac{10}{580}, \frac{20}{580}, \frac{50}{580}, \frac{500}{580})$ as the rewards (weight samples) of these four arms.

To address the above concerns, for each link, we learn the protocol performance independently and only observe the weight ratio of each *pair* of protocols to predict the best and worst protocols in expectation each time. By defining each arm as a protocol pair, we obtain only $|\mathcal{L}| \times M^2$ arms. As explained in the algorithm intuition, the optimal solution uses only two protocols at a time for all alive flows, so there is no need for us to observe arbitrary combinations of protocols; it suffices to consider only protocol pairs. Let $\eta_{lt}(m, m')$ represent the ratio of the average flow rates

achieved by two protocols m and m' on link l in time t, defined as:

$$\eta_{lt}(m,m') = \frac{\text{average flow rate on } l \text{ under } m' \text{ at } t}{\text{average flow rate on } l \text{ under } m \text{ at } t}. \quad (6)$$

Our learning strategy is thus to run a Lower Confidence Bound (LCB) algorithm independently on each link to estimate $\mathbb{E}[\eta_{lt}(m,m')]$, which equals $\mathbb{E}\left[\frac{w_{lt}(e_{m'})}{w_{lt}(e_m)}\right]$. Let $x_{lt}(m,m')$ be the indicator variable which equals 1 if we choose the protocol pair (m,m'); and 0 otherwise. The LCB of $\mathbb{E}[\eta_{lt}(m,m')]$, denoted by $\eta_{lt}^{LCB}(m,m')$, is defined as

$$\eta_{lt}^{LCB}(m,m') = \frac{\sum_{t'=1}^{t}\eta_{lt'}(m,m')x_{lt'}(m,m')}{\sum_{t'=1}^{t}x_{lt'}(m,m')} - \sqrt{\frac{\alpha\log t}{\sum_{t'=1}^{t}x_{lt'}(m,m')}} \quad (7)$$

Note that if there is a number ($< T$) of time slots in which the distribution of the weight vector $\{w_{lt}(e_i)\}_{i=1}^{M}$ abruptly changes, one can adapt (7) to instead take the average of $\eta_{lt'}(m,m')$ over a fixed-size horizon rather than the entire history [34]. The algorithm pseudo-code and analysis are omitted for brevity since they are not the core contribution of this work.

Online protocol selection based on predictions. At time slot t, on each link l, once we estimate $\mathbb{E}[\eta_{lt}(m,m')]$ as $\eta_{lt}^{LCB}(m,m')$, we choose the protocol pair (m,m') that minimizes $\eta_{lt}^{LCB}(m,m')$. Such a pair is denoted as (m^*,m^b) (line 5 of Alg. 1). Our strategy is to assign m^* to the shortest alive flow (indexed by i^*) and m^b to all the other alive flows (line 6), as discussed in the algorithm intuition above. By doing so, we guarantee that the shortest flow gets the highest transmission rate if our predictions are accurate. More formally, we can show that:

Lemma 3.1. *Let $\mathbf{x}_{lt}^*$ and $\mathbf{x}_{lt}$ denote our protocol decision matrix and any other feasible protocol decision matrix at t, respectively. Then, we have:*

$$r_{i^*lt}(\mathbf{x}_{lt}^*) = B_{lt}\left(\frac{(|\mathcal{A}_{lt}|-1)w_{lt}(e_{m^b})+w_{lt}(e_{m^*})}{w_{lt}(e_{m^*})}\right)^{-1} \geq B_{lt}\left(1+\sum_{i\in\mathcal{A}_{lt}\setminus\{i^*\}}\frac{w_{lt}(\vec{x}_{ilt})}{w_{lt}(\vec{x}_{i^*lt})}\right)^{-1} = r_{i^*lt}(\mathbf{x}_{lt}),\ \forall \mathbf{x}_{lt} \quad (8)$$

In other words, the transmission rate achieved by the shortest flow i^* under our decisions at t ($\mathbf{x}_{lt}^*$) is no smaller than it would have been under any other decision at t ($\mathbf{x}_{lt}$), given the same decisions at other time slots. The proof follows immediately from the strategies in lines 5 – 6. Finally, we apply our protocols to the alive flows, and observe the transmission rates of each flow. We than update the LCB of $\eta_{lt}(m,m')$ according to (7) for use in the next time slot (line 7).

3.3 Protocol Selection with Non-uniform Utilization

We now consider the case when the utilizations $u(\mathbf{x}_{lt}, B_{lt})$ are non-uniform. In **per-link protocol selection**, $\mathbf{x}_{lt}$ degrades to a vector of the same protocol decision for all flows on a given link. Since we do not control the priorities of different flows in this case, our goal becomes to learn the protocol on each link that can achieve the highest bandwidth utilization $u(\mathbf{x}_{lt}, B_{lt})$. We can easily modify Algorithm 1 to distributedly predict the expected delay per bit (the "reward") under each protocol choice (the "arm") for each link using a similar LCB technique, and then assign the protocol with the lowest reward for all flows to use on the link. We then see that this formulation reduces to the traditional LCB framework [35], and thus can be shown to converge to the optimal protocol selection in the sense of achieving a $O(\log T)$ regret over T time slots relative to selecting the optimal protocol in each time slot. We omit the formal performance analysis for reasons of space. We test our algorithm performance for this case in Section 5.1.

In **per-flow protocol selection**, letting $u(\mathbf{x}_{lt}, B_{lt})$ and thus the total achieved rate depend on our protocol choices further complicates the problem of optimizing the total completion time. Even if we know the weight distributions, following Algorithm 1's intuition to choose the protocol with the highest relative weight for the shortest alive flow may lead to a low bandwidth utilization for all flows on the link and, depending on how much the utilization varies with the protocol choices, may even result in a low actual achieved rate for the shortest alive flow. To account for these effects, we assume that B_{lt} is i.i.d. over time for each link and then adapt our Algorithm 1 to learn the *actual rate achieved* (as defined in (4)) by flows using the dominant protocol in each pair of protocols, instead of the ratio of the protocol weights. Note that the achieved rate accounts for the bandwidth utilization, which is determined by the entire selected protocol combination, and the assumption of i.i.d. capacity mitigates the discrepancy between the rate samples of possibly different protocol sets chosen at different times (recall our four-protocol example in Section 3.2). For protocol pair (m,m'), the dominant protocol m is the one that incurs a smaller transmission delay per bit; we define $d_{lt}^{LCB}(m,m')$ as the LCB of the link transmission delay per bit (the inverse of the transmission rate) of protocol m. Formally, we replace Line 5 of Algorithm 1 with:

$$\text{Choose } (m^*,m') = \textbf{argmin}_{(m,m')}\ d_{lt}^{LCB}(m,m'). \quad (9)$$

Intuitively, we would expect this algorithm to yield the optimal protocol selection for a single path network if the utilization does not vary much with the protocol choice and thus our problem is close to that in Section 3.2, which Algorithm 1 solves optimally. In Section 4.4, we provide theoretical results to demonstrate this intuition.

4 Performance Guarantees

In this section, we provide theoretical performance bounds for Algorithm 1 when all flows follow the same path, e.g., slices might traverse the same sequence of network domains, or the "network" might be a single wireless link between Internet-of-Things or mobile devices and an edge server, as in the 5G and MAC use cases in Table 1. We validate our algorithms on AR hologram transmissions between end devices and an edge server in Section 5.2. For the case with uniform utilization ratio $u(\cdot)$, our theoretical guarantees hold for both i.i.d. unknown and arbitrarily unknown bandwidth capacities. We additionally propose and analyze the performance of two variants of Algorithm 1:

one designed to speed up the learning rate by leveraging background traffic (Section 4.2), and one designed to handle the case where the weight vector w_{lt} for link l depends on the link capacity B_{lt} (Section 4.3). Finally, theoretical results for arbitrary utilization $u(\cdot)$ are given in Section 4.4.

4.1 Regret Analysis for Uniform Bandwidth Utilization

We evaluate our algorithm by upper-bounding its *Regret*, *i.e.*, the expected difference of the total completion time between that achieved by our algorithm and the *offline optimum*, an omniscient algorithm that has perfect knowledge of: (1) the arrival times and sizes of all flows, (2) the bandwidth capacities over all time slots, and (3) the distributions of weight vectors. Formally, we define

$$Regret = \mathbb{E}\left[\sum_{i\in[n]} \tau_i(\mathbf{x}_i)\right] - \min_{\mathbf{x}} \mathbb{E}\left[\sum_{i\in[n]} \tau_i(\mathbf{x})\right] \quad (10)$$

The basic idea is to first evaluate the performance of our algorithm when we have accurate predictions of the weights but have no information of future flows and bandwidth capacities; and then upper-bound the difference between the algorithm performance with known weight vector distributions and one without any prior knowledge. We will use w_{ltm} and $w_{lt}(e_m)$ interchangeably for convenience.

In the following, we upper-bound the gap between the total completion time with our algorithm and with the expected optimum for the scenario where flows take the same path. All missing proofs of our theorems and the corollary can be found in our technical report [33].

Theorem 4.1. *Let B_{max} and B_{min} respectively denote the largest and smallest capacity over all links over all time, w_{max} and w_{min} denote the largest and smallest expected weights over all protocols and all links, and $\hat{\eta}_l^{max} = \max_{(m,m',t)} \mathbb{E}[\frac{w_{ltm}}{w_{ltm'}}]$. If the weight vector for each link is i.i.d., and we have $\frac{B_{lt}}{B_{l't}} \leq \frac{\hat{\eta}_l^{max}}{\hat{\eta}_{l'}^{max}} \leq \frac{B_{l't}}{B_{lt}}$, $\forall(l',l) : B_{l't} \geq B_{lt}, t \in [T]$, then the regret of our algorithm $OPSBC$ will be upper-bounded by*

$$O\left(\frac{\epsilon B_{max} w_{max} |\mathcal{P}| M^2 \log T}{B_{min} w_{min} \eta_{min}^2}\right),$$

The last assumption in Theorem 4.1 indicates that the relative competence (weights) of different protocols do not vary too much across links with respect to their bandwidth capacities. For example, if links l and l' have bandwidths of $1GB/s$ and $10GB/s$ respectively, then this assumption indicates that, if on *link* l, the best protocol can achieve 10 times the rate of the worst protocol, then the best protocol on *link* l' cannot have more than 100 times the rate of the worst protocol on l'. This assumption is reasonable in realistic settings and technically implies that all the flows have the same bottleneck link under our protocol choices.

Implications. Theorem 4.1 shows that the regret bound is quadratic in M and linear in $|\mathcal{P}|$, a vast improvement over the $M^{|\mathcal{A}||\mathcal{P}|}$ possible assignments if directly applying the MAB framework. Note that $\epsilon \log T = \frac{\tilde{T} \log T}{T}$, and we see that a higher learning rate (a larger T) leads to a smaller regret: as $T \to \infty$, implying an infinite number of opportunities to learn the bandwidth weights, the regret converges to 0. However, in practice, a higher learning rate may bring more frequent protocol changes. Thus, we can also only update protocol choices when flows arrive or depart, leading to the following regret bounds:

Corollary 4.2. *If any algorithm can only select protocols when either new flows arrive or old flows complete, we can adapt Alg. 1 to be triggered by flows' arrivals and departures, instead of updating the flow decisions in each time slot. If γ denotes the average flow arrival rate, the resulting regret will be upper-bounded by $O\left(\frac{B_{max} w_{max} |\mathcal{P}| M^2 \log n}{\gamma B_{min} w_{min} \eta_{min}^2}\right)$.*

4.2 Improved Theorem 4.1 Using Background Traffic

We next show that Theorem 4.1's regret bound may be improved with an alternative learning algorithm that leverages background traffic. Suppose that in each time slot, we can also observe (at least) one flow in the background traffic (denoted by flow i_0) that uses a fixed protocol, denoted by m_0 ($\in [M]$). For instance, this flow could represent a constant stream of data from an IoT device to a cloud or edge server. We take this flow as a "reference" flow against which we can compare protocol performance: specifically, we track the ratio of the transmission rate gained by using each individual protocol m to that gained by using m_0, rather than updating the prediction of each pair of protocols each time. This procedure reduces the number of arms by a factor of M compared to Algorithm 1.

More formally, for each l and t, we define $\tilde{\eta}_{lt}(e_m) = \frac{\bar{r}_{lt}(e_{m_0})}{\bar{r}_{lt}(e_m)} = \frac{B_{lt}\bar{w}_{lt}(e_{m_0})}{B_{lt}\bar{w}_{lt}(e_m)} = \frac{\tilde{w}_{lt}(e_{m_0})}{\tilde{w}_{lt}(e_m)}$. Here, $\bar{r}_{lt}(e_m)$ denotes the average rate achieved by flows using protocol m on l in t. Since m_0 is always present, we can finally have a good estimate of the vector $\left(\frac{w_{lt}(e_{m_0})}{w_{lt}(e_1)}, \frac{w_{lt}(e_{m_0})}{w_{lt}(e_2)}, \cdots, \frac{w_{lt}(e_{m_0})}{w_{lt}(e_M)}\right)$. Based on observations of $\tilde{\eta}_{lt}(e_m)$, lines 5 and 6 of our Algorithm 1 simplify as follows: We assign the protocols with the lowest and highest $\tilde{\eta}_{lt}^{LCB}(e_m)$ to the shortest alive flow and all the other flows, respectively. This strategy can also guarantee that when predictions are accurate, the current shortest flow gets the highest rate. Since we compare the protocols by independently learning the relative performance of each protocol compared with the protocol m_0, the number of arms is only $M \times |\mathcal{L}|$, and this adaption leads to a smaller regret shown as below, compared to Theorem 4.1. However, this strategy requires us to observe the rates of at least one flow that is always present in the network and always uses the same protocol, which may not be present in practice.

Theorem 4.3. *If all the flows share the same path, at least one flow in the background traffic that persistently uses a fixed protocol can be observed, the weight vector for each link are i.i.d., and $\frac{B_{lt}}{B_{l't}} \leq \frac{\hat{\eta}_l^{max}}{\hat{\eta}_{l'}^{max}} \leq \frac{B_{l't}}{B_{lt}}$, $\forall(l',l) : B_{l't} \geq B_{lt}, t \in [T]$, the regret of $OPSBC$ will be upper-bounded by $O\left(\frac{\epsilon B_{max} w_{max} |\mathcal{P}| M \log T}{B_{min} w_{min} \eta_{min}^2}\right)$.*

4.3 Improved Theorem 4.1 for I.i.d. Capacities

If the weight vector $\vec{w}_{lt}$ depends on the link capacity B_{lt}, the assumption that $\vec{w}_{lt}$ is i.i.d. chosen from a static distribution will not hold unless B_{lt} is also i.i.d. over time. In practice, it is reasonable to assume that link bandwidth is also i.i.d. chosen from an unknown distribution since the background traffic is often determined by random flow arrivals and departures. We show how to adapt Alg. 1 for this model.

Let $\delta_{lm^*}(m^*, m^b)$ denote the inverse of the fraction of bandwidth that protocol m^* can get when protocol pair (m^*, m^b) is chosen. Instead of updating the LCB of $\mathbb{E}\left[\frac{w_{lt}(e_{m^b})}{w_{lt}(e_{m^*})}\right]$ (line 7 in Alg. 1), we instead sequentially update the empirical average rate achieved by each protocol multiplied by the current number of alive flows over all times that the protocol is chosen to estimate $\delta_{lm^*}(m^*, m^b)$ and greedily choose the protocol pair with the smallest LCB of $\delta_{lm^*}(m^*, m^b)$. This modified algorithm has the following regret:

Theorem 4.4. *If all the flows share the same path, both the bandwidth capacities B_{ilt} and the weight vector for each link are i.i.d., and $\frac{B_{lt}}{B_{l't}} \leq \frac{\hat{\eta}_l^{max}}{\hat{\eta}_{l'}^{max}} \leq \frac{B_{l't}}{B_{lt}}$, $\forall(l', l) : B_{l't} \geq B_{lt}, t \in [T]$, the regret of $OPSBC$ will be upper-bounded by $O\left(\frac{\epsilon B_{max} w_{max} |\mathcal{P}| M \log T}{B_{min} w_{min} \eta_{min}^2}\right)$.*

Compared to Theorem 4.1, our regret bound is linear instead of quadratic in M, since we have M instead of M^2 arms.

4.4 Special Case Regret for Non-uniform Utilization

We finally analyze the regret of Algorithm 1 with adaptation (9) for two flows sharing the same bottleneck link. Before that, we define the *dominant throughput* to be the transmission rate achieved by flows using the dominant protocol in any protocol pair.

Assumption 1. *For the bottleneck link, given an optimal protocol pair p^* that has the largest dominant throughput in expectation and any other protocol pair p, let: r^*_{total} and r_{total} (r^*_d and r_d) denote the total transmission rates (dominant throughput) of p^* and p, respectively; $r^*(single)$ denotes the highest expected transmission rate of any single present flow. Then, we present the following assumption, which indicates that the difference in the total transmission rate is relatively small compared to the difference in the dominant protocol. For any protocol pair that is not the same as p^*, we have:*

$$\frac{r_{total} - r^*_{total}}{r^*_d - r_d} \leq 2\left(\frac{r^*_{total}(single)}{r^*_d} - 1\right) \tag{11}$$

Theorem 4.5. *Let u_{max} and u_{min} denote the maximum and minimum utilization ratio over all links. If there are always at most two flows present in the network sharing the same bottleneck link (the link with the least bandwidth capacity), both bandwidth capacity and utilization ratio for any given link are i.i.d. over time, and we have Assumptions 1 and $\frac{B_{lt}}{B_{l't}} \leq \frac{\hat{\eta}_l^{max} u_{max}(\mathbf{x}_l, B_{lt})}{\hat{\eta}_{l'}^{max} u_{max}(\mathbf{x}_l, B_{lt})} \leq \frac{B_{l't}}{B_{lt}}$, $\forall(l', l) : B_{l't} \geq B_{lt}, t$, then the regret of our Algorithm 1 with adaption (9) will be upper-bounded by*

$$O\left(\frac{\epsilon B_{max} u_{max} w_{max} |\mathcal{P}| M^2 \log(T u_{max}/u_{min})}{B_{min} u_{min} w_{min} \eta_{min}^2}\right).$$

Theorem 4.5 shows that when the difference in utilization between the optimal and non-optimal protocol pairs is small compared to the difference in dominant throughputs (Assumption 1), our modification of Algorithm 1 converges to the optimal protocol selection when only two flows are present. This condition is consistent with measurements of the average throughput achieved by different TCP variants provided in Table III of [36].

# of flows	Crab network		Star network	
	HomoPS	Random	HomoPS	Random
100	33.7%	19.9%	41.9%	35.3%
200	37.3%	22.8%	46.2%	41.3%
300	38.9%	24.0%	48.2%	44.9%
400	39.3%	24.2%	49.2%	45.3%
500	39.7%	24.6%	50.1%	45.6%

TABLE 3: Increase in the total flow time for heuristics compared with OPSBC.

5 Experimental Validation

In this section, we validate our theoretical results. We first compare our algorithms' performance to the offline optimum and heuristics in synthetic simulations (Section 5.1) and then use protocol performance traces from an `ns-3` [37] emulation, as well as conducting experiments on an Amazon EC2 testbed and a university testbed (Section 5.2).

We focus on transport-layer protocols, *e.g.*, UDP and multiple TCP variants, as the protocol choices in our experiments; but protocols on other layers of the network or other types of network policies can also be tested in each set of our experimental scenarios.

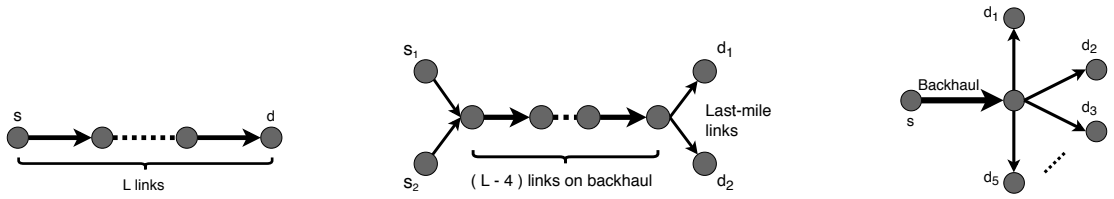
5.1 Numerical Performance

We evaluate Algorithm 1 in three settings: (1) per-flow protocol selection on a line network (validating Section 4's performance guarantees), (2) per-flow selection when flows take different paths, and (3) per-link selection with Section 3.3's proposed modification to Algorithm 1.

Simulation set-up. We consider the three network topologies shown in Figure 2 with three protocol choices on each link. The bandwidth capacities and weight parameters are drawn from uniform distributions. The expected bandwidth capacities are drawn from the range $[8, 16]$ (Mbps) in each timeslot. On each link, we set the weights as $\mathbb{E}[w_{ilt}] = \frac{1}{6}, \frac{1}{3}, \frac{1}{2}$ for each distinct protocol. The default variance of the bandwidth capacity and weights is $1/3$ and $1/12$. We simulate 500 flows arriving at the network according to a Poisson Process with an arrival rate of 0.8 and record the total accrued completion time as flows depart. By default, each flow takes a randomly chosen path in the network with size chosen uniformly within $[20, 60]$ (Mb). In this section, L is interchangeable with $|\mathcal{L}|$. We report the averaged results of 100 repetitions of each simulation.

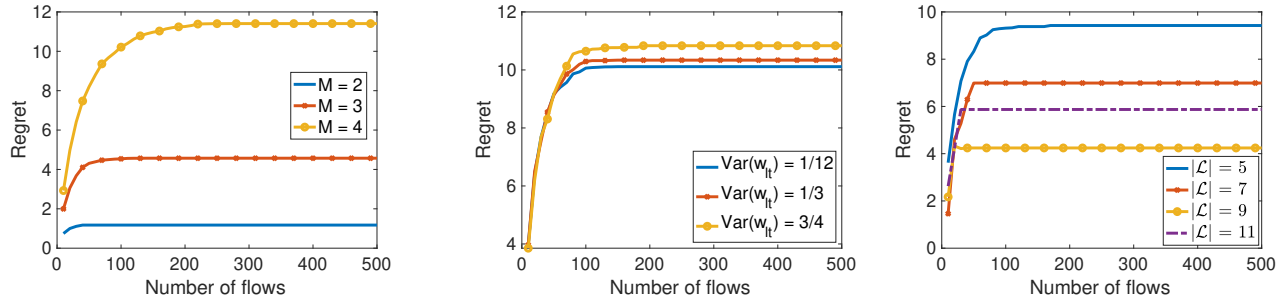
Per-flow algorithm regret on a line network. We first consider the regret for a line network (Figure 2a), defined as the average difference of the delay cost between our algorithm and the offline optimum. On a single path network as in this line network example, the offline optimum always assigns the protocol with the highest-in-hindsight bandwidth partition weight on the link to the flow with the shortest remaining time, and assigns the protocol with the lowest-in-hindsight weight to other alive flows.

Figure 3 indicates that the regret of Algorithm 1 is sublinear with the number of flows and of logarithmic order, as in Theorem 4.1. In Figure 3a, a larger number of protocol choices leads to a higher regret, which is consistent with our theoretical analysis. Furthermore, Alg. 1 converges at a slower rate toward the optimum with more protocol



(a) Line network. (b) Crab network. (c) Star network.

Fig. 2: Network topologies used in our synthetic simulations.



(a) Regret increases as the number of protocols M increases.

(b) Regret remains stable as the weight vectors w_{lt} vary more.

(c) Regret remains stable as the number of links $|\mathcal{L}|$ varies.

Fig. 3: The regret of OPSBC on a single-path-network is logarithmic with the number of flows.

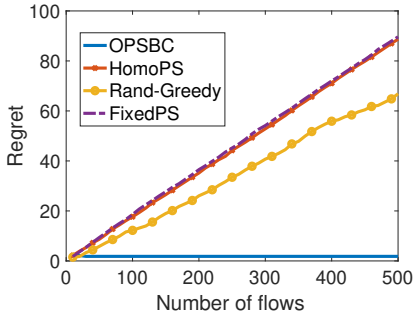


Fig. 4: Significantly lower regret of **OPSBC** than heuristics.

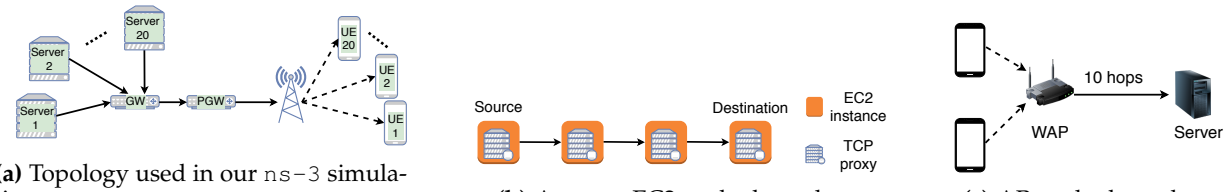
choices, which require more exploration before learning the optimal protocol pair. Figure 3b shows that the variance of the weight vector has only mild effects on our algorithm performance. In Figure 3c, we observe that the number of links affects the average regret non-monotonically. We hypothesize that this occurs because the completion time of each flow is determined by the delay on the bottleneck link. Therefore, an increased number of extra links can either increase or decrease the completion time depending on if they introduce a new network bottleneck.

Per-flow selection on various networks. To further evaluate Alg. 1, we design different heuristics for various network topologies. **HomoPS** always chooses the same protocol for all the alive flows on each link, which corresponds to using the same default protocol for all flows. Therefore, in expectation, flows will get an equal proportion of the bandwidth capacity on each link. In each time slot, **Rand-Greedy** randomly chooses a protocol pair for each link and adopts the same greedy protocol assignment strategy as Alg. 1, assigning the "best" protocol to the shortest alive flow and the other protocol to all other flows. **FixedPS** randomly chooses a protocol on all links for each flow to use through their entire lifetimes.

Figure 4 compares the average regret of **OPSBC** (Alg. 1) to that of our three heuristics on a line network. In contrast to the fast convergence of our Alg. 1, the heuristics' regrets increase almost linearly, indicating an inability to adapt to the network dynamics. More specifically, the performance of **FixedPS**, a predetermined protocol selection strategy, is similar to that of **HomoPS**, a completely fair strategy. This probably occurs because **FixedPS** has equal probabilities to choose each protocol for each flow on the bottleneck link, which may converge to a fair strategy in expectation over time. The regret of **Rand-Greedy** increases at a lower rate, demonstrating the good performance of our greedy strategy in assigning protocols.

For the crab and star networks, we compare another heuristic, **Random**, which randomly chooses a protocol for each flow on each link, with **OPSBC** and **HomoPS**. Table 3 shows that **Random** out-performs **HomoPS**, the fair strategy; and these heuristics have at least a 24% and 45% larger total completion time than our algorithm **OPSBC** for 500 flows on the two networks, respectively. These results imply that per-flow protocol selection can lower completion times compared to using the same protocol for all flows, especially when we learn the protocol performance on heterogeneous links as done by **OPSBC**.

For **per-link protocol selection**, we implement an extra benchmark by using a reinforcement learning algorithm, A2C, with a pre-trained model. More specifically, we consider the same line network used for Figure 3 and develop an environment where the network characteristics on each link independently evolve as a Markov Process, consisting of 20 distinct values of bandwidth capacities



(a) Topology used in our `ns-3` simulation.

(b) Amazon EC2 testbed topology.

(c) AR testbed topology.

Fig. 5: Network topologies used in our emulation and testbed experiments.

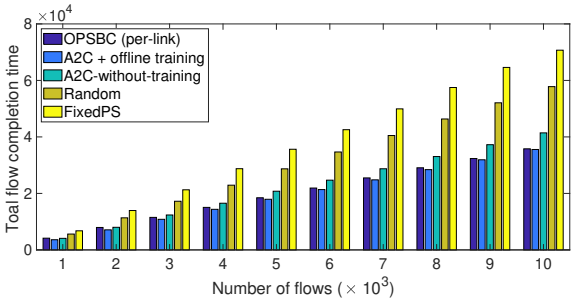
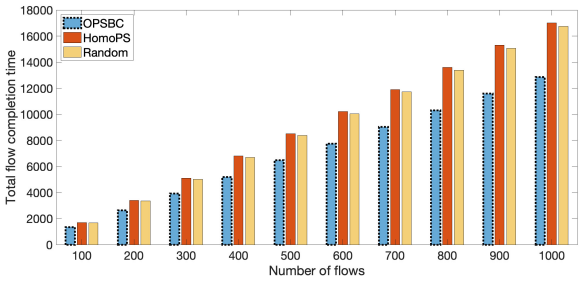
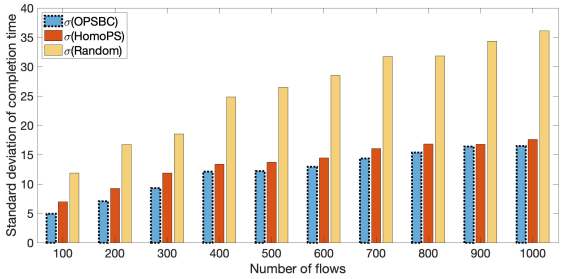


Fig. 6: OPSBC achieves similar flow-time (ms) to that of A2C with offline training for per-link selection on a line network.



(a) Flow completion time (ms).



(b) Standard deviation of flow completion time.

Fig. 7: OPSBC achieves lower flow-time with a lower standard deviation on `ns-3` data traces.

as states and a distribution of reward representing the total transmission rate that is dependent with protocol and bandwidth capacity. Since A2C allows different protocols to be chosen for different network states, we would expect it to outperform our OPSBC model, which simply aims to optimize over the expected state. Figure 6 shows that our algorithm **OPSBC** achieves a 16% to 50% flow completion time improvement over heuristics **Random**, **FixedPS** and an advanced reinforcement learning algorithm A2C without using a pre-trained model (**A2C-without-training**). Meanwhile, it shows that using offline training can indeed learn the optimal protocols faster (**A2C + offline training** with 10^4 timesteps of pre-training achieves a smaller completion time in the initial rounds), but our model-free Algorithm 1 with adaptation based on (9) can catch up with **A2C + offline training** quickly and achieves comparable ($< 1\%$ difference) completion time, even without the offline training that **A2C + offline training** requires. Therefore, our algorithm can benefit scenarios where collecting extensive offline data traces is impractical, e.g., if it is difficult to simulate realistic network conditions.

5.2 Experimental Results

We finally demonstrate that our Alg. 1 (**OPSBC**) is feasible for transport protocol selection in real network settings. We first test the algorithm on performance data from `ns-3` [37] TCP simulations and then demonstrate its efficacy in experiments on Amazon EC2.

Experiments with emulation data. Figure 5a shows the setup of our `ns-3` [37] simulator. We assume that 20 mobile nodes each receive data from a dedicated server over an LTE network with a crab topology shown in Figure 5a; the server-PGW links each have 100Mbps capacity while the PGW-BS link has 20 Mbps capacity, making it the network bottleneck. Each node can use one of five transport protocols: UDP, TCP CUBIC, TCP NewReno, TCP Vegas, or TCP Westwood. Nodes running UDP saturate their flows at 1 Mbps. We run 20 flows for each protocol pair in 30 different scenarios (*e.g.*, with randomly varying mobility of the mobile nodes) and measure their throughput and delay on each link, using these data traces to drive our algorithm. The flow sizes and arrival rates are chosen from the same distributions as in Section 5.1.

Figure 7a compares the total flow-time of **OPSBC** (Alg. 1) to that of the **HomoPS** and **Random** heuristics. These two heuristics show a 30% larger total flow-time than our Alg. 1 when 1000 flows are processed, which is consistent with their 24.6% increase in completion times for the crab topology with synthetic data (Table 3). We note that we have 5 protocol choices in our testbed emulation, instead of the 3 choices used for Table 3's results. Thus, we would expect to observe higher regret and less improvement over the heuristic (cf. Figure 3a) due to having more protocol choices. We also find the standard deviation of the total flow-time ($\sigma(\cdot)$) of **OPSBC**, **HomoPS** and **Random**. Figure 7b shows that **OPSBC** has the smallest standard deviations, indicating it consistently performs well over the randomly repeated experiments.

EC2 experiments. We implement a single-source-destination network with four nodes, each equipped with

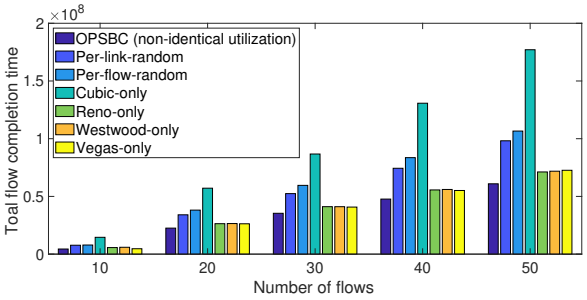


Fig. 8: OPSBC for non-identical utilization achieves lower flow-time (ms) than heuristics in EC2 experiments on Figure 5b's testbed.

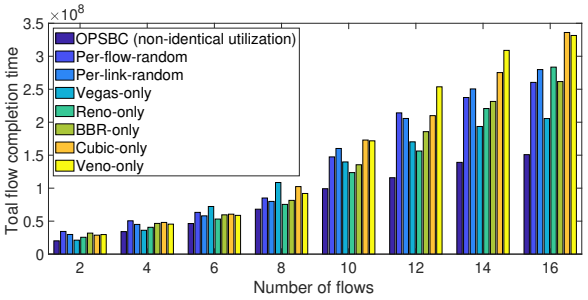


Fig. 9: OPSBC for non-identical utilization achieves lower flow-time (μs) than heuristics in AR hologram transmission experiments on Figure 5c's testbed.

a TCP proxy, deployed on four Amazon EC2 VM instances with 50 Mbps capacity on each link (Figure 5b). We generate 50 flows with sizes uniformly drawn from $[10, 3 \times 10^3]$ (Mb) arriving in five batches spaced 3 seconds apart. On each link, **OPSBC** can choose TCP CUBIC, Vegas, Reno, or Westwood. Figure 8 shows the resulting completion times. Despite the overhead at the node proxies, **OPSBC** achieves a 37.95%, 42.86%, and $> 16\%$ decrease in flow-time, compared to randomly selecting any single protocol for all flows to use on each link (**Per-link-random**), randomly selecting a protocol for each flow on each link (**Per-flow-random**), and simply using a single variant of TCP for all flows (**XX-only**), respectively. Note that in realistic scenarios, one cannot accurately predict which single protocol is the optimum in the future or expect that using one protocol solely (*e.g.*, **Vegas-only**) will always perform the best in a changing environment. Therefore, our dynamic strategy provides the most flexible way to select the optimal protocol for the current time and adapt over time.

AR experiments. We finally implement a two-source-and-one-destination network in the computing cluster at Duke University. Figure 5c shows the network topology, with two Pixel 3 XL cellphones that are connected to a WiFi access point (WAP) and running Augmented Reality (AR) applications. They receive holograms through the same path from an edge sever 10 hops away in the Duke Computing Cluster. We generate 16 hologram transmissions for each cellphone, and we uniformly at random sample the size of each hologram from $[0, 256]$ (MB) and the time interval between the end time of each transmission and the start time of the previous transmission from $[0, 15]$ (s). We allow **OPSBC** to choose TCP Veno, Reno, Vegas, Cubic, and Bottleneck Bandwidth and Round-trip propagation time (BBR) [38] for each transmission between a source to the destination. Figure 9 shows that **OPSBC** achieves at least a 26% decrease in flow-time, compared to 7 benchmarks: **XX-only** always uses protocol **XX** for all flows, **Per-link-random** randomly chooses a single protocol for all flows to use, and **Per-flow-random** randomly chooses a protocol for each alive flow.

6 Related Work

Machine learning for protocol selection. In recent years, machine learning has been adopted to design TCP and resource allocation algorithms in network settings. Winstein *et al.* [39] design Remy, a program that can generate distributed congestion control algorithms in a multi-user network and achieve desired outcomes, *e.g.,* high throughput. In light of this offline learning application, we believe that online learning of the protocol performance as in this work can further facilitate protocol selection in dynamic network environments. A new adaptable congestion control algorithm, called Verus [21], can dynamically adjust the congestion window size by continuously learning the short-term packet delay variations. Mao *et al.* [40] use reinforcement learning to allocate cloud resources in an online setting for minimizing job slow-downs. In contrast, we provide *theoretical* guarantees of our online algorithms' solution optimality, and numerically show that we achieve comparable performance to algorithms with offline training.

Online learning for network management. Online learning algorithms can make decisions in network settings when data gradually become available over time. Chen et al. [41] design novel algorithms for online convex optimization problems with switching costs. Zhang *et al.* [42] integrate the online gradient descent method into online cloud resource provisioning with theoretical guarantees. However, these studies assume full feedback on all feasible solutions, which is unavailable in our model. Thus, we instead take an MAB approach, which only uses information from the chosen decisions, leading to the famous *exploration-and-exploitation* trade-off. Combes *et al.* [43] propose MAB algorithms for ad-display optimization, considering users with fixed budgets. In [44], MAB algorithms have been extended for dynamic channel access. Unlike the algorithm design in these works, we do not directly optimize the rewards of all possible solution vectors, but *instead select a subset of solutions driven by the analysis of the offline optimum and use the rewards as predicted inputs needed by an additional strategy to optimize the protocols*. A few other works design MAB-based algorithms for optimizing transport or network layer protocols, but have different focus and theoretical challenges from ours. For instance, [45] maximizes the multi-path sending rates at a home network with interference domains considered for each link. Their optimization problem has a complex structure, which is proven to be NP-hard in the offline setting, due to the network topology and interference across links, and their MAB algorithm needs to deal with costly probing for the rewards. But our challenge is orthogonal – our decisions are coupled together across times due to the complex form of the flow completion time

function. [46] proposes three insightful myopic congestion control policies for end-to-end protocol design in wireless mesh network based on the restless MAB framework. Their objective function is written as the sum of terms, each of which only depends on the decision in a single timeslot, while ours is allowed to be an unknown and non-linear function of decisions across timeslots, and we validate our method using both theoretical regret analysis and testbed experiments. In [47], MAB model is established for making the decisions on channel selection in a wireless local area network, incorporating chaotically oscillating time series generated by semiconductor lasers. They leverage the specific problem structure in ultrafast photonic systems to design the MAB framework for throughput maximization and focus on the operation viability tested through experiments. We focus on a different set of network scenarios where the resources are divided by different application flows in a unknown manner and the objective function is non-linear with rewards over different time points.

Reinforcement learning (RL) has been adopted recently to optimize network policies in wireless (e.g., 5G) networks, such as TCP sending rates [48] or congestion windows [49], network slice configuration [50], [51], [52], MAC-layer resource allocation [53], [54], caching [55], offloading decisions [56], power management [57], and revenue maximization [58]. These works adopt state-of-the-art deep RL algorithms, which require a large and uncertain amount of time to train the neural network models, while our model-free approach is easier to implement, has much smaller algorithm running time, and can achieve comparable performance as shown in Section 5. Our MAB framework can moreover provide theoretical optimality guarantees, unlike most deep RL approaches. By leveraging existing protocols rather than using RL methods to design new ones as in [39], [52], [54], we further introduce less risk that the learned protocol will behave unexpectedly in practice, as most existing protocols can be expected not to cause catastrophic failures. New protocols designed by RL methods, on the other hand, may not have such assurances.

Protocol performance models. A large variety of TCP [31], [59] and MAC [14], [32] variants have been proposed to improve flow throughput while controlling congestion. Extensive experiments, for instance, have compared the performance of TCP variants, *e.g.,* TCP Cubic, TCP NewReno, etc. [29], [59], showing that these TCP variants have different performance on various types of network links. However, there are few analytical models for how these protocols interact with each other when they coexist in the network. NUM frameworks provide insight into protocol interactions at various layers of the stack when the per-flow resource allocation preferences are expressed with utility functions [13], [29], [30]. However, such utility functions only cover a few TCP variants and cannot adapt to changing network conditions.

7 Discussion and Conclusion

To cope with network flows' increasingly heterogeneous requirements and changing network characteristics, we propose a dynamic network management framework that leverages existing network protocols. We use model-free online learning to support automatic protocol selection for each individual flow, so as to optimize the overall flow completion time. Motivated by the deficiency of existing models for flow performance under different protocol choices, we propose a model to characterize coexisting flows' transmission rates under different protocols. We then extend multi-armed bandit algorithms to learn the rate function and predict an optimal assignment of protocols to flows at each time. Taking these real-time predictions as input, we then propose a provably optimal online protocol selection scheme that can minimize the aggregate flow completion time. The asymptotic optimality of our learning and assignment algorithm is validated through theoretical analysis and experiments. Substantial work, however, remains. On the theoretical side, additional efforts will be required to explore the lower-bound of the regret and adapt our mechanisms to independent decisions at different flow sources. On the practical side, testbed experiments for a larger scale network considering cross-layer protocol interactions, e.g., routing choices, should be carried out. Our work thus represents an initial step towards optimal online protocol selection for automated network management.

8 Acknowledgement

This work was supported by a Nokia gift, a Lord Foundation of North Carolina grant, an MSIT program IITP-2022-2017-0-01633 of South Korea, and grants NSF CSR-1903136, ONR N00014-21-1-2128, NSFC 62102460, and DARPA HR001117C0048.

References

[1] X. Zhang, S. Chen, Y. Im, M. Gorlatova, S. Ha, and C. Joe-Wong, "Towards automated network management: Learning the optimal protocol selection," in *In Proc. of IEEE ICNP*, 2019.

[2] C. Westphal, "IETF presentation: Challenges in networking to support augmented reality and virtual reality," Mar. 2017.

[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[4] C.-X. Wang, F. Haider, X. Gao, X.-H. You, Y. Yang, D. Yuan, H. Aggoune, H. Haas, S. Fletcher, and E. Hepsaydir, "Cellular architecture and key technologies for 5g wireless communication networks," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 122–130, 2014.

[5] A. Silver, "3 ways nokia is using machine learning in 5g networks," IEEE Spectrum, 2018.

[6] P. Popovski, K. F. Trillingsgaard, O. Simeone, and G. Durisi, "5g wireless network slicing for embb, urllc, and mmtc: A communication-theoretic view," *IEEE Access*, vol. 6, pp. 55 765 – 55 779, 2018.

[7] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, May 2017.

[8] N. Hassan, K. Yau, and C. Wu, "Edge computing in 5g: A review," *IEEE Access*, vol. 7, pp. 127 276 – 127 289, 2019.

[9] B. Hartpence, *Packet Guide to Routing and Switching*. O'Reilly Media, Inc, 2011.

[10] "Virtual network traffic routing," https://docs.microsoft.com/en-us/azure/virtual-network/virtual-networks-udr-overview, 2020.

[11] O-RAN Alliance, "O-ran: Towards an open andsmart ran," https://www.o-ran.org/, 2018.

[12] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[13] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 255–312, 2007.

[14] F. Ait Aoudia, M. Gautier, M. Magno, O. Berder, and L. Benini, "A generic framework for modeling mac protocols in wireless sensor networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 25, no. 3, pp. 1489–1500, 2017.

[15] A. A. Khan, M. H. Rehmani, and M. Reisslein, "Requirements, design challenges, and review of routing and mac protocols for cr-based smart grid systems," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 206–215, 2017.

[16] Ł. Budzisz, R. Stanojević, A. Schlote, F. Baker, and R. Shorten, "On the fair coexistence of loss-and delay-based tcp," *IEEE/ACM Transactions on Networking (TON)*, vol. 19, no. 6, pp. 1811–1824, 2011.

[17] C. Xu, J. Zhao, and G.-M. Muntean, "Congestion control design for multipath transport protocols: A survey," *IEEE communications surveys & tutorials*, vol. 18, no. 4, pp. 2948–2969, 2016.

[18] Q. Gao, J. Zhang, and S. V. Hanly, "Cross-layer rate control in wireless networks with lossy links: leaky-pipe flow, effective network utility maximization and hop-by-hop algorithms," *IEEE Transactions on wireless communications*, vol. 8, no. 6, 2009.

[19] N. Ehsan and M. Liu, "Modeling tcp performance with proxies," *Computer Communications*, vol. 27, no. 10, pp. 961–975, 2004.

[20] F. Le, E. Nahum, V. Pappas, M. Touma, and D. Verma, "Experiences deploying a transparent split tcp middlebox and the implications for nfv," in *In Proc. of ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*. ACM, 2015, pp. 31–36.

[21] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, and C. Görg, "Adaptive congestion control for unpredictable cellular networks," in *In Proc. of ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4. ACM, 2015, pp. 509–522.

[22] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *The Journal of Machine Learning Research*, vol. 12, pp. 2121 – 2159, 2011.

[23] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of ACM*, vol. 20, no. 1, pp. 46 – 61, 1973.

[24] R. Bhargava, "Evolution of Dropbox's edge network," Dropbox, 2017, https://blogs.dropbox.com/tech/2017/06/evolution-of-dropboxs-edge-network/.

[25] Google Cloud, "Overview of TCP forwarding," https://cloud.google.com/iap/docs/tcp-forwarding-overview, 2019.

[26] Y. Cai, F. R. Yu, and S. Bu, "Cloud computing meets mobile wireless communications in next generation cellular networks," *IEEE Network*, vol. 28, no. 6, pp. 54–59, 2014.

[27] G. Siracusano, R. Bifulco, S. Kuenzer, S. Salsano, N. B. Melazzi, and F. Huici, "On the fly tcp acceleration with miniproxy," in *Proceedings of the 2016 workshop on Hot topics in Middleboxes and Network Function Virtualization*. ACM, 2016, pp. 44–49.

[28] S. H. Low, "A duality model of tcp and queue management algorithms," *IEEE/ACM Transactions On Networking*, vol. 11, no. 4, pp. 525–536, 2003.

[29] B. Sikdar, S. Kalyanaraman, and K. S. Vastola, "Analytic models for the latency and steady-state throughput of tcp tahoe, reno, and sack," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 6, pp. 959–971, 2003.

[30] K. Nagaraj, D. Bharadia, H. Mao, S. Chinchali, M. Alizadeh, and S. Katti, "Numfabric: Fast and flexible bandwidth allocation in datacenters," in *In Proc. of ACM SIGCOMM*. ACM, 2016, pp. 188 – 201.

[31] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," *In Proc. of ACM SIGOPS operating systems review*, vol. 42, no. 5, pp. 64–74, 2008.

[32] I. Demirkol, C. Ersoy, F. Alagoz *et al.*, "Mac protocols for wireless sensor networks: a survey," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 115–121, 2006.

[33] "Optimal network protocol selection for competing flows via online learning," Dropbox, 2022. [Online]. Available: https://www.dropbox.com/s/25ae0nykg2i39pt/TechReport_2019.pdf?dl=0

[34] A. Garivier and E. Moulines, "On upper-confidence bound policies for non-stationary bandit problems," https://arxiv.org/abs/0805.3415, 2008.

[35] Y. Xia, T. Qin, W. Ma, N. Yu, and T.-Y. Liu, "Budgeted multi-armed bandits with multiple plays," in *In Proc. of IJCAI*, 2016.

[36] B. Turkovic, F. A. Kuipers, and S. Uhlig, "Fifty shades of congestion control: A performance and interactions evaluation," 2019. [Online]. Available: https://arxiv.org/pdf/1903.03852.pdf

[37] "NS-3 network simulator," https://www.nsnam.org/, 2018.

[38] M. Hock, R. Bless, and M. Zitterbart, "Experimental evaluation of BBR congestion control," in *Proc. IEEE ICNP'17*, 2017.

[39] K. Winstein and H. Balakrishnan, "Tcp ex machina: computer-generated congestion control," in *In Proc. of ACM SIGCOMM*, vol. 43, no. 4. ACM, 2013, pp. 123–134.

[40] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *In Proc. of the 15th ACM Workshop on Hot Topics in Networks*. ACM, 2016, pp. 50–56.

[41] N. Chen, J. Comden, Z. Liu, A. Gandhi, and A. Wierman, "Using predictions in online optimization: Looking forward with an eye on the past," in *In Proc. of ACM SIGMETRICS*, 2016.

[42] X. Zhang, C. Wu, Z. Li, and F. C. Lau, "Proactive vnf provisioning with multi-timescale cloud resources: Fusing online learning and online optimization," in *In Proc. of IEEE INFOCOM*, 2017.

[43] R. Combes, C. Jiang, and R. Srikant, "Bandits with budgets: Regret lower bounds and optimal algorithms," in *In Proc. of ACM SIGMETRICS*, 2015.

[44] Y. Liu and M. Liu, "An online approach to dynamic channel access and transmission scheduling," in *In Proc. of MobiHoc*, 2015.

[45] S. Henri, C. Vlachou, and P. Thiran, "Multi-armed bandit in action: Optimizing performance in dynamic hybrid networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1879–1892, 2018.

[46] A. A. A. Islam, S. I. Alam, V. Raghunathan, and S. Bagchi, "Multi-armed bandit congestion control in multi-hop infrastructure wireless mesh networks," in *In Proc. of IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 2012.

[47] S. Takeuchi, M. Hasegawa, K. Kanno, A. Uchida, N. Chauvet, and M. Naruse, "Dynamic channel selection in wireless communications via a multi-armed bandit algorithm using laser chaos time series," *Scientific Reports*, vol. 10, no. 1574, 2020.

[48] N. Jay, N. Rotman, B. Godfrey, M. Schapira, and A. Tamar, "A deep reinforcement learning perspective on internet congestion control," in *International Conference on Machine Learning*, 2019, pp. 3050–3059.

[49] Z. Xu, J. Tang, C. Yin, Y. Wang, and G. Xue, "Experience-driven congestion control: When multi-path tcp meets deep reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1325–1336, 2019.

[50] V. Sciancalepore, X. Costa-Perez, and A. Banchs, "Rl-nsb: Reinforcement learning-based 5g network slice broker," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1543–1557, 2019.

[51] Q. Liu, T. Han, and E. Moges, "Edgeslice: Slicing wireless edge computing network with decentralized deep reinforcement learning," https://arxiv.org/abs/2003.12911, 2020.

[52] Y. Hua, R. Li, Z. Zhao, X. Chen, and H. Zhang, "Gan-powered deep distributional reinforcement learning for resource management in network slicing," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 334–349, 2019.

[53] R. Ali, N. Shahin, Y. B. Zikria, B.-S. Kim, and S. W. Kim, "Deep reinforcement learning paradigm for performance optimization of channel observation–based mac protocols in dense wlans," *IEEE Access*, vol. 7, pp. 3500–3511, 2018.

[54] Y. Yu, T. Wang, and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1277–1290, 2019.

[55] W. Chien, H. Weng, and C. Lai, "Q-learning based collaborative cache allocation in mobile edge computing." *Future Gener. Comput. Syst.*, vol. 102, pp. 603–610, 2020.

[56] Y. Dai, D. Xu, K. Zhang, Y. Lu, S. Maharjan, and Y. Zhang, "Deep reinforcement learning for edge computing and resource allocation in 5g beyond," in *IEEE International Conference on Communication Technology*, 2019.

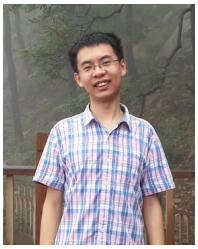
[57] N. Liu, Z. Li, J. Xu, Z. Xu, S. Lin, Q. Qiu, J. Tang, Y. Wang, and Y. Wang, "A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning," in *Proc. of IEEE ICDCS*, 2017.

[58] X. Foukas, M. K. Marina, and K. Kontovasilis, "Iris: Deep reinforcement learning driven shared spectrum access architecture for indoor neutral-host small cells," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1820–1837, 2019.

[59] I. Abdeljaouad, H. Rachidi, S. Fernandes, and A. Karmouch, "Performance analysis of modern tcp variants: A comparison of cubic, compound and new reno," in *Communications (QBSC), 2010 25th Biennial Symposium on*. IEEE, 2010, pp. 80–83.

Xiaoxi Zhang (S'13, M'18) is an associate professor with the School of Computer Science and Engineering, Sun Yat-sen University. Before joining SYSU, she was a Postdoctoral researcher of the Department of Electrical and Computer Engineering at Carnegie Mellon University, advised by Prof. Carlee Joe-Wong. She earned her Ph.D. degree in Computer Science from The University of Hong Kong in 2017, advised by Prof. Chuan Wu and Prof. Francis C.M. Lau. She received her B.E. degree in Electronics and Information Engineering from the Huazhong University of Science and Technology in 2013. Her research interests lie in the broad area of optimization and algorithm design for networked systems, including edge computing networks and distributed machine learning systems.

Siqi Chen (S'19) received his B.S. degree in computer science from Shanghai Jiao Tong University in 2016. He is currently the 2nd year Ph.D. student in the Department of Computer Science, University of Colorado Boulder. His research interests are related to mobile networking and systems, LTE and 5G architecture design/optimization, and next-generation Internet.



Yunfan Zhang received his B.S. degree in Computer Science from Duke University in 2020. His research interests include Computer Networks, AI for Systems, Internet of Things, AR/VR, and Edge Computing.

Youngbin Im (S'13, M'20) is an Assistant Professor in the School of Electrical and Computer Engineering at UNIST. Before joining UNIST, he was a postdoctoral researcher in the Department of Computer Science at University of Colorado Boulder from 2015 to 2019. He received his B.S. and Ph.D. degrees in computer science and engineering from Seoul National University in 2006 and 2014. His research interests include the next-generation Internet, video streaming, Internet protocols, data centers, wireless networks, IoT. He received the Best Paper Award from ACM MobiSys in 2019.

Maria Gorlatova (S'11, GS'12, M'15) received the B.Sc. (summa cum laude) and M.Sc. degrees in electrical engineering from the University of Ottawa, Ottawa, ON, Canada, and the Ph.D. degree in electrical engineering from Columbia University, New York, NY, USA. She is an Assistant Professor of Electrical and Computer Engineering and Computer Science with Duke University, Durham, NC, USA. She has several years of industry experience, and has been affiliated with Telcordia Technologies, Piscataway, NJ, USA, IBM, Armonk, NY, USA, and D. E. Shaw Research, New York, NY, USA. Her current research interests include architectures, algorithms, and protocols for emerging pervasive technologies. Dr. Gorlatova was a recipient of the Google Anita Borg USA Fellowship, the Canadian Graduate Scholar NSERC Fellowships, and the Columbia University Presidential Fellowship. She was a co-recipient of the 2011 ACMSenSys Best Student Demonstration Award, the 2011 IEEE Communications Society Award for Advances in Communications, and the 2016 IEEE Communications Society Young Author Best Paper Award.

Sangtae Ha (S'07–M'09–SM'12) is currently an Associate Professor with the Department of Computer Science, University of Colorado at Boulder. In 2009, he co-founded the EDGE Lab, Princeton University, as its first Associate Director and led its research team as an Associate Research Scholar with Princeton University from 2010 to 2013. He is also a Co-Founder and the founding CTO/VP Engineering of DataMi, a startup company on mobile networks. He also co-founded Zoomi, Inc., an artificial-intelligence-based learning analytics company. He received the INFORMS ISS Design Science Award in 2014 and the Best Paper Awards at ACM MobiSys in 2019 and 2021.

Carlee Joe-Wong (S'11, M'16) is the Robert E. Doherty Associate Professor of Electrical and Computer Engineering at Carnegie Mellon University. She received her A.B. degree (magna cum laude) in Mathematics, and M.A. and Ph.D. degrees in Applied and Computational Mathematics, from Princeton University in 2011, 2013, and 2016, respectively. Her research interests lie in optimizing various types of networked systems, including applications of machine learning and pricing to cloud computing, mobile/wireless networks, and ridesharing networks. From 2013 to 2014, she was the Director of Advanced Research at DataMi, a startup she co-founded from her research on mobile data pricing. She received the NSF CAREER award in 2018 and the ARO Young Investigator award in 2019.