Optimal Network Control with Adversarial Uncontrollable Nodes

Qingkai Liang, Eytan Modiano
LIDS, MIT

ABSTRACT

The effectiveness of many well-known network control algorithms (e.g., MaxWeight) relies on the premise that all of the nodes are fully controllable. However, these algorithms may yield poor performance in a partially-controllable network where a subset of nodes are uncontrollable and may take arbitrary (possibly adversarial) actions. Such a partially-controllable model is of increasing importance in real-world networked systems such as overlay-underlay networks and uncooperative wireless networks. In this paper, we study two fundamental network optimization problems in a network with adversarial uncontrollable nodes. First, we investigate the network stability problem where the objective is to stabilize the network whenever possible. We develop a lower bound on the total queue length that can be achieved by any causal policy, and propose a throughput-optimal algorithm, called Tracking-MaxWeight (TMW), which enhances the original MaxWeight algorithm with an explicit learning of the behavior of uncontrollable nodes. Second, we study the network utility maximization problem where the objective is to maximize cumulative utility subject to stability constraints. We provide a lower bound on the utility-delay tradeoff, and develop the Tracking Drift-plus-Penalty (TDP) algorithm that achieves tunable utility-delay tradeoffs.

1 INTRODUCTION

Network optimization has been an active area of research for many decades due to its wide applicability in practice. The objective is usually to achieve network stability or maximize network utility under unknown system statistics (e.g., exogenous arrival rates). Many efficient network optimization algorithms have been developed in the past few decades, such as the well-known MaxWeight algorithm [24]. The effectiveness of these algorithms usually relies on the premise that all of the nodes in a network are fully controllable and adopt some optimal algorithm. Unfortunately, an increasing number of real-world networked systems are only *partially controllable*, where a subset of nodes are not managed by the network operator and use some unknown (possibly adversarial) network control policy, such as overlay-underlay networks.

This work was supported by NSF Grant CNS-1524317 and by DARPA I2O and Raytheon BBN Technologies under Contract No. HROO l l-l 5-C-0097.

Mobihoc '19, Catania, Italy

DOI: 10.1145/3323679.3326508

An overlay-underlay network consists of overlay nodes and underlay nodes [9, 20, 22]. The overlay nodes can implement state-of-the-art algorithms while the underlay nodes are uncontrollable and could take arbitrary (possibly adversarial) actions. Figure 1 shows an overlay-underlay network where the communications among overlay nodes rely on the uncontrollable underlay nodes. Overlay networks have been used to improve the performance and capabilities of computer networks for a long time (e.g., content delivery [23]).

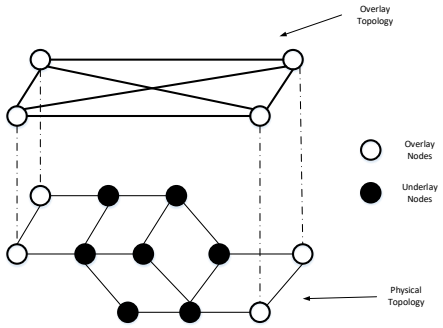


Figure 1: An example of overlay-underlay networks.

Due to the unknown behavior of uncontrollable nodes, the existing routing algorithms may yield poor performance in a partially-controllable environment. For example, Figure 2 shows an example where the well-known Backpressure routing algorithm [24] fails to deliver the maximum throughput when some nodes are uncontrollable. In particular, uncontrollable node 3 adopts a policy that does not preserve the flow conservation law such that its backlog builds up, but uncontrollable node 2 hides this backlog information from node 1. As a result, if node 1 uses Backpressure routing, it always transmits packets to node 2, although these packets will never be delivered. A smarter algorithm should be able to learn the behavior of the uncontrollable nodes such that node 1 only sends packets along route $1 \rightarrow 5 \rightarrow 4$.

As a result, it is important to develop new network control algorithms that can achieve consistently good performance in a partially-controllable environment. In this paper, we investigate efficient network optimization algorithms that achieve guaranteed performance in the presence of uncontrollable nodes that may exhibit arbitrary (possibly adversarial) behavior. In particular, we study two fundamental problems in network optimization.

First, we investigate the network stability problem where the objective is to achieve network stability whenever the system is inside the stability region. We provide a lower bound on the total queue length that could be achieved by any causal policy with respect to a "niceness" metric for the actions taken by uncontrollable nodes.

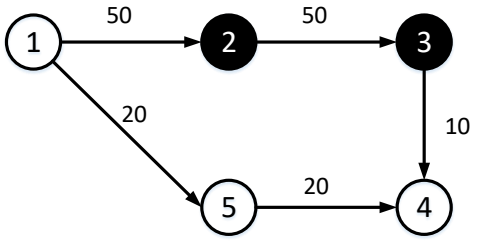


Figure 2: Counterexample where the well-known Backpressure routing algorithm fails to deliver the maximum throughput in a partially-controllable network. The number next to each link is its capacity. Each node can transmit only to one of its neighbors in each time slot. There is only one flow: $1 \rightarrow 4$ (at rate 20). Black nodes are uncontrollable nodes. Node 2 transmits any packet it received to node 3 at full rate, so that its queue length is always zero; node 3 adopts a non-work-conserving policy that holds any packet it received. When node 1 uses Backpressure routing, it always transmits packets to node 2 since its queue length is always zero, which hides the fact that backlog builds up at node 3.

Then we propose a new algorithm called Tracking-MaxWeight (TMW), which enhances the well-known MaxWeight algorithm [24] with an explicit learning of the policy used by uncontrollable nodes. We provide a queue length upper bound for Tracking-MaxWeight and show that Tracking-MaxWeight is throughput-optimal even in the presence of adversarial uncontrollable nodes.

Second, we study the network utility maximization problem where the objective is to maximize network utility (e.g., throughput and fairness) subject to the constraint that the network be stable. In this context, the performance is measured with respect to the tradeoff between delay (i.e., total queue length) and utility regret (i.e., the utility difference between a causal policy and the optimal oracle policy that knows the future). We provide a lower bound on the utility-delay tradeoff that can be achieved by any causal algorithm and develop a new Tracking-Drift-plus-Penalty (TDP) algorithm that achieves good utility-delay tradeoffs in the presence of adversarial uncontrollable nodes.

1.1 Related Work

1.1.1 Network Optimization in Partially-Controllable Networks. Most of the existing works on network optimization in a partially-controllable environment are in the context of overlay-underlay networks. An important feature of overlay-underlay networks is that underlay nodes are uncontrollable and may take arbitrary control actions. The objective is to find efficient control policies for the controllable overlay nodes in order to optimize certain performance metrics (e.g., throughput). In [9], the authors showed that the well-known BackPressure algorithm [24], which was shown to be throughput-optimal in a wide range of scenarios, may lead to a loss in throughput when used in an overlay-underlay setting, and proposed a heuristic routing algorithm for overlay nodes called Overlay Backpressure Policy (OBP). An optimal backpressure-type routing algorithm for a special case, where the underlay paths do not overlap with each other, was given in [20]. Recently, [22] showed that the overlay routing algorithms proposed in [9][20] are not throughput-optimal in general, and developed the Optimal Overlay Routing Policy (OORP) for overlay nodes. However, all of the existing overlay routing algorithms [9, 20, 22] impose very stringent assumptions about the behavior of underlay nodes. In particular, the underlay nodes are required to use fixed-path routing (e.g., shortest-path routing) and maintain stability whenever possible, which fails to account for many important underlay policies (e.g., underlay nodes may use multi-path routing).

1.1.2 Network Optimization in Adversarial Environments. The study of adversarial network models dates back more than two decades. Cruz [7] provided the first concrete example of networks with adversarial dynamics, which were later generalized by Borodin *et al.* [4] under the *Adversarial Queuing Theory* (AQT) framework. In AQT, in each time slot, the adversary injects a set of packets at some of the nodes. In order to avoid trivially overloading the system, the AQT framework imposes a stringent *window constraints*: the maximum traffic injected in every link over any window of W time slots should not exceed the amount of traffic that the link can serve during that interval. The AQT model has given rise to a large number of results since its introduction, most of which are about network stability under several simple scheduling policies such as FIFO (see [6] for a review of these results). However, the AQT model assumes that only packet injections are adversarial while the underlying network topology and link states remain fixed. Andrews and Zhang [2, 3] extended the AQT model to single-hop dynamic wireless networks, where both packets injections and link states are controlled by an adversary, and proved the stability of the MaxWeight algorithm in this context. Jung *et al.* [1, 13] further extended the results of [2, 3] to multi-hop dynamic networks. While the above-mentioned works focused on network stability, Neely [17] investigated the network utility maximization problem in an adversarial environment. Algorithm performance was measured with respect to a so-called "W-slot look-ahead policy". Such a policy has perfect knowledge about network dynamics over the next W slots and it is assumed that under this policy the total arrivals to each queue should not exceed the total amount of service offered to that queue during every window of W slots.

Note that all of the aforementioned works considered the same "window-based" adversarial model due to its analytical tractability. Some recent works relaxed the window constraints. For example, Walton [25] investigated queue stability under an arbitrary arrival process; Paschos and Tassiulas [21] studied the problem of stabilizing queues under a mixture of stochastic and adversarial traffic injections. However, these results are limited to very specific queueing systems and only exogenous traffic injections are adversarial. Recently, Liang and Modiano [10, 11] studied various network optimization problems under a generalized adversarial network model that relaxed the window constraints. They analyzed the worst-case performance of different network control algorithms (e.g., MaxWeight [24] and Drift-plus-Penalty [18]) under their proposed adversarial models with respect to the notions of queue length regret and utility regret.

Despite the advances in adversarial network optimization, none of the existing works can be applied in a partially-controllable network. For example, in Figure 2, it has been shown that MaxWeight may achieve very poor performance even in face of a mildly-behaved uncontrollable nodes, not to mention its worse-case performance in the presence of adversarial uncontrollable nodes.

Another related area of research is Constrained Online Learning (COL) [8, 15, 19, 26] where a reward and a set of cost functions are generated by an adversary and the objective is to maximize the cumulative rewards while keeping the long-term costs below a certain threshold. However, the COL framework is stateless, meaning that revealed reward and cost functions do not depend on any system state, whereas most networking problems have queue length as a system state whose evolution follows the Lindley recursion [14]. As a result, the COL framework cannot be directly applied to our context. Some recent works [5] on COL also demonstrated their applications in networking problems, but the queueing dynamics is often relaxed such that the problems become stateless.

1.2 Our Contributions

We investigate network optimization under a generalized partially-controllable network model whereas existing works (e.g., [9, 20, 22]) imposed very stringent assumptions about the behavior of uncontrollable nodes for analytical tractability (e.g., underlay nodes use shortest path routing). In addition, our framework allows for both adversarial environment (e.g., adversarial exogenous packet injections and link states) and adversarial uncontrollable nodes, thus greatly extending previous adversarial network models that do not account for uncontrollable nodes (e.g., [2, 10, 11, 17]). As far as we know, this is the first work that establishes network optimization results under the generalized adversarial partially-controllable network model.

Moreover, we develop lower bounds on the total queue length and the utility-delay tradeoff in the presence of adversarial uncontrollable nodes. These lower bounds reveal the fundamental performance limits of a network with adversarial uncontrollable nodes.

Finally, we propose new network optimization algorithms that achieve guaranteed good performance even in the presence of adversarial uncontrollable nodes. The analysis of our algorithms is fundamentally different from the traditional equilibrium-based arguments used in stochastic network optimization, since there may not exist any steady state in the presence of adversarial uncontrollable nodes. Our proof first establishes sampled-path bounds under stringent window-based assumptions, and then carefully relaxes the assumptions by using a load-shedding technique.

2 SYSTEM MODEL

2.1 Asymptotic Notations

Let f and g be two functions defined on some set of real numbers. Then $f(x) = O(g(x))$ if $\limsup_{x\to\infty} \frac{|f(x)|}{g(x)} < \infty$. Similarly, $f(x) = \Omega(g(x))$ if $\liminf_{x\to\infty} \frac{f(x)}{g(x)} > 0$. Also, $f(x) = \Theta(g(x))$ if $f(x) = O(g(x))$ and $f(x) = \Omega(g(x))$. In addition, $f(x) = o(g(x))$ if $\lim_{x\to\infty} \frac{f(x)}{g(x)} = 0$, and in this case we say that $f(x)$ is sublinear in $g(x)$.

2.2 Network Model

Consider a networked system with N nodes (the set of all nodes is denoted by $\mathcal{N}$). Time is slotted with a finite time horizon $\mathcal{T} = \{0, 1, \cdots, T-1\}$. There are K flows in the network and each node i maintains a queue for buffering undelivered packets for each flow k (the queue is denoted by (i,k)). As a result, there are NK queues in the network, and we denote by $\mathbf{Q}(t)$ the queue length vector at the beginning of slot t, where its element $Q_{ik}(t)$ represents the queue length for flow k at node i in slot t.

Let ω_t be the *network event* that occurs in slot t, which includes information about the current network parameters, such as a vector of channel conditions for each link and a vector of exogenous arrivals to each queue. In particular, the vector of exogenous packet arrivals is denoted by $\mathbf{a}(\omega_t) = \{a_{ik}(t)\}_{i,k}$, where $a_{ik}(t)$ is the number of exogenous arrivals to queue (i,k) in slot t.

At the beginning of time slot t, after observing the current network event ω_t and the current queue length vector $\mathbf{Q}(t)$, each node i needs to make a routing decision $f_{ijk}(t)$ indicating the offered transmission rate for flow k over link $i \to j$. The corresponding network routing vector is denoted by $\boldsymbol{f}(t) = \{f_{ijk}(t)\}_{i,j,k}$.

There are two types of nodes in the network: controllable nodes (the set of controllable nodes is denoted by C) and uncontrollable nodes (the set of uncontrollable nodes is denoted by $\mathcal{U}$). The network operator can only control the routing behavior for controllable nodes while the routing actions taken by uncontrollable nodes cannot be regulated and are only observable at the end of each time slot. In this case, the routing vector $\boldsymbol{f}(t)$ can be decomposed into two parts: $\boldsymbol{f}(t) = (\boldsymbol{f}^c(t), \boldsymbol{f}^u(t))$. Here, $\boldsymbol{f}^c(t) = \{f_{ijk}(t)\}_{i\in C}$ represents the routing decisions made by controllable nodes (referred to as the **controllable action**) and $\boldsymbol{f}^u(t) = \{f_{ijk}(t)\}_{i\in\mathcal{U}}$ corresponds to the routing decisions made by uncontrollable nodes (referred to as the **uncontrollable action**). The routing vectors $\boldsymbol{f}^c(t)$ and $\boldsymbol{f}^u(t)$ are constrained within some action spaces $\mathcal{F}^c_{\omega_t}$ and $\mathcal{F}^u_{\omega_t}$, respectively, that may depend on the current network event ω_t. The action space for all nodes is denoted by $\mathcal{F}_{\omega_t} = \mathcal{F}^c_{\omega_t} \cup \mathcal{F}^u_{\omega_t}$. The action space can be used to specify routing constraints (e.g., the total transmission rate over each link should not exceed its capacity) or describe scheduling constraints (e.g., each node can only transmit to one of its neighbors in each slot).

Note that when there is not enough backlog to transmit, the actual number of transmitted packets may be less than the offered transmission rate. In particular, we denote by $\widetilde{f}_{ijk}(Q_{ik}(t))$ (or simply $\widetilde{f}_{ijk}(t)$ if the context is clear) the actual number of transmitted packets for flow k over link $i \to j$ in slot t under the current queue length $Q_{ik}(t)$. Clearly, we have $\widetilde{f}_{ijk}(Q_{ik}(t)) \le \min\{f_{ijk}(t), Q_{ik}(t)\}$. We further assume that the routing decision can always be chosen to respect the backlog constraints (but the actual actions may not necessarily be queue-respecting). This can be done simply by never attempting to transmit more data than we have. Under such notations, the queuing dynamics are given by

$$\begin{aligned} &Q_{ik}(t+1) \\ =&Q_{ik}(t) + a_{ik}(t) + \sum_{j\in\mathcal{N}} \widetilde{f}_{jik}(t) - \sum_{j\in\mathcal{N}} \widetilde{f}_{ijk}(t) \\ \le&\Big[Q_{ik}(t) + a_{ik}(t) + \sum_{j\in\mathcal{N}} f_{jik}(t) - \sum_{j\in\mathcal{N}} f_{ijk}(t)\Big]^+, \end{aligned}$$

where $[z]^+ = \max\{z, 0\}$.

The sequence of network events $\{\omega_t\}_{t\leq T}$ and uncontrollable actions $\{\boldsymbol{f}^u(t)\}_{t\leq T}$ are generated according to an *arbitrary* process (possibly non-stationary or even adversarial), except for the boundedness assumption that the amount of exogenous arrivals and the offered transmission rate in each time slot are bounded by some constant D, i.e.,

$$0 \leq a_{ik}(t) \leq D,\ 0 \leq f_{ijk}(t) \leq D,\ \forall i, j, k.$$

A policy π generates a sequence of routing actions $\pi = \{\boldsymbol{f}^c(t)\}_{t\leq T}$ within the time horizon. In each slot t, the queue length vector under policy π is denoted by $\mathbf{Q}^\pi(t)$. A *causal* policy is one that generates the current controllable action $\boldsymbol{f}^c(t)$ only based on the knowledge up until the current slot t. In contrast, a *non-causal* (oracle) policy may generate the current controllable action based on knowledge of the future.

One of the most important notions in network optimization theory is *network stability*. In this paper, we focus on rate stability, defined as follows.

Definition 2.1. A network is rate-stable if

$$\lim_{T\to\infty} \frac{\sum_{i,k} Q_{ik}(T)}{T} = 0.$$

In other words, rate stability means that as the time horizon T becomes large the average arrival rate asymptotically approaches the average departure rate for each queue. From a finite-time perspective, rate stability requires that the total queue length grows *sublinearly* with the time horizon T, i.e., $\sum_{i,k} Q_{ik}(T) = o(T)$.

In this paper, we study the following two fundamental network optimization problems.

- **Network Stability.** Our objective is to find a causal policy such that rate stability can be achieved whenever the system is inside the stability region. The corresponding results are presented in Section 3.
- **Network Utility Maximization.** Our objective is to maximize a network utility function (to be discussed later) subject to the constraint that the network should be rate-stable. The corresponding results are provided in Section 4.

3 NETWORK STABILITY

In this section, we focus on the network stability problem where our objective is to find a causal policy such that stability can be achieved whenever possible. Note that even under the mild case where uncontrollable nodes use a stationary policy, existing network optimization algorithms may fail to stabilize the network. For example, as is illustrated in Figure 2, the well-known Backpressure algorithm achieves low throughput when some uncontrollable node uses a fixed control policy. Such a failure is due to the fact that some uncontrollable node uses a non-stabilizing policy that does not preserve flow conservation but the Backpressure algorithm is not aware of this non-stabilizing behavior.

In the following, we first characterize a lower bound on the total queue length that can be achieved by any causal policy in an adversarial partially-controllable network (Section 3.1). Then we develop an algorithm called *Tracking-MaxWeight* that stabilizes a network with adversarial uncontrollable nodes whenever possible in Section 3.2 . Finally, we characterize the stability region for any adversarial partially-controllable network in Section 3.3.

3.1 Lower Bound on Queue Length

We first introduce an important metric that measures the "niceness" of a sequence of uncontrollable routing actions $\{\boldsymbol{f}^u(t)\}_{t\leq T}$:

$$V_T\Big(\{\boldsymbol{f}^u(t)\}_{t\leq T}\Big) \triangleq \max_{t\leq T} \sum_{i,k} Q^*_{ik}(t),$$

where $\mathbf{Q}^*(t)$ is the queue length vector achieved by the optimal oracle policy in slot t under the given sequence of uncontrollable routing actions $\{\boldsymbol{f}^u(t)\}_{t\leq T}$. In other words, $V_T\Big(\{\boldsymbol{f}^u(t)\}_{t\leq T}\Big)$ is the peak queue length during the sample path of the *optimal oracle policy* that knows the future and minimizes the total queue length $\sum_{i,k} Q_{ik}(T)$. Note that if there are multiple optimal solutions, the one with the largest $V_T(\cdot)$ is considered. For convenience, we may simply write $V_T\Big(\{\boldsymbol{f}^u(t)\}_{t\leq T}\Big)$ as V_T if the context is clear.

The following theorem gives a lower bound on the total queue length that can be achieved by any causal policy.

Theorem 3.1. *For any causal policy π, there exists a sequence of uncontrollable routing actions $\{\boldsymbol{f}^u(t)\}_{t\leq T}$ such that the total queue length $\sum_{i,k} Q^\pi_{ik}(T) = \Omega(V_T)$.*

Proof. For any causal policy π, we construct a sequence of uncontrollable routing actions such that the total queue length never exceeds V_T under the optimal oracle policy, but the total queue length under the given causal policy π is at least cV_T for a constant $c > 0$. The detailed proof is presented in the technical report [12]. □

Theorem 3.1 shows that if $V_T = \Omega(T)$, then no causal policy can achieve sublinear queue length. On the other hand, if $V_T = o(T)$, there *might* exist some causal policy that attains sublinear queue length, which we investigate in the next section.

3.2 Tracking-MaxWeight Algorithm

Now we introduce an algorithm that achieves sublinear queue length whenever $V_T = o(T)$. The algorithm is called Tracking-MaxWeight (TMW), which enhances the original MaxWeight algorithm [24] with an explicit learning of the policy used by uncontrollable nodes. Throughout this section, we let $\{\boldsymbol{f}^u(t)\}_{t\leq T}$ be the sequence of routing actions that are actually executed by uncontrollable nodes.

The details of the TMW algorithm are presented in Algorithm 1. In each slot t, the TMW algorithm generates the routing actions $\mathbf{g}^c(t) = \{g_{ijk}(t)\}_{i\in C}$ for controllable nodes, and also produces an "imagined" routing action $\mathbf{g}^u(t) = \{g_{ijk}(t)\}_{i\in\mathcal{U}}$ for uncontrollable nodes, by solving the optimization problem (3) (we will explain this problem later). With these calculated actions, the TMW algorithm then updates two virtual queues. The first virtual queue $\mathbf{X}(t)$ tries to emulate the physical queue $\mathbf{Q}(t)$ but assumes that the imagined uncontrollable action $\mathbf{g}^u(t)$ is applied (while the physical queue is updated using the true uncontrollable action $\boldsymbol{f}^u(t)$). The second virtual queue $\mathbf{Y}(t)$ tracks the cumulative difference between the imagined uncontrollable actions $\{\mathbf{g}^u(t)\}_{t\geq 0}$ and the actual uncontrollable actions $\{\boldsymbol{f}^u(t)\}_{t\geq 0}$. In particular, we use $\Delta_{ijk}(t)$ to

measure the difference between the imagined routing action $g_{ijk}(t)$ and the true routing action $f_{ijk}(t)$ taken by uncontrollable node $i \in \mathcal{U}$, which is given by

$$\Delta_{ijk}(t) = g_{ijk}(t) - \widetilde{f}_{ijk}(t), \ \forall i \in \mathcal{U}, \tag{1}$$

where $\widetilde{f}_{ijk}(t)$ is the actual number of transmitted packets under the true routing action $f_{ijk}(t)$ given the current queue backlog $\mathbf{Q}(t)$. Note that for each controllable node $i \in C$, we simply set $\Delta_{ijk}(t) = 0$.

The optimization problem (3) aims at maximizing a weighted sum of flow variables, which is similar to the optimization problem solved in the original MaxWeight algorithm [24] except for the setting of weights. In the original MaxWeight algorithm, the weight is $W_{ijk}(t) = Q_{ik}(t) - Q_{jk}(t)$, which corresponds to the physical queue backlog differential, while in the Tracking-MaxWeight algorithm the weight $W_{ijk}(t) = X_{ik}(t) - X_{jk}(t) - Y_{ijk}(t)$ accounts for both the backlog differential for virtual queue $\mathbf{X}(t)$ and the backlog of virtual queue $\mathbf{Y}(t)$. The derivation of (3) is based on the minimization of quadratic Lyapunov drift terms for the two virtual queues:

$$\begin{aligned} \min_{\mathbf{g}(t)\in\mathcal{F}_{\omega_t}} \ & \sum_{i,k} X_{ik}(t)\Big[a_{ik}(t) + \sum_j g_{jik}(t) - \sum_j g_{ijk}(t)\Big] \\ & + \sum_{i,j,k} Y_{ijk}(t)\Big(g_{ijk}(t) - \widetilde{f}_{ijk}(t)\Big), \end{aligned} \tag{2}$$

where the first term corresponds to the Lyapunov drift of virtual queue $\mathbf{X}(t)$ and the second term corresponds to the Lyapunov drift of virtual queue $\mathbf{Y}(t)$. Note that the minimization is done over controllable actions $\mathbf{g}^c(t)$ and "imagined" uncontrollable actions $\mathbf{g}^u(t)$. Cleaning up irrelevant constants, i.e., $a_{ik}(t)$ and $\widetilde{f}_{ijk}(t)$, and rearranging terms yield the optimization problem (3).

Algorithm 1 Tracking-MaxWeight (TMW)

1: In each slot t, observe the current network event ω_t and solve the following optimization problem to obtain the controllable action $\mathbf{g}^c(t)$ and the *imagined* uncontrollable action $\mathbf{g}^u(t)$:

$$\max_{\mathbf{g}(t)\in\mathcal{F}_{\omega_t}} \sum_{(i,j)} \sum_k g_{ijk}(t) W_{ijk}(t), \tag{3}$$

where

$$W_{ijk}(t) = X_{ik}(t) - X_{jk}(t) - Y_{ijk}(t).$$

2: Controllable nodes execute the routing decision $\mathbf{g}^c(t)$.

3: Observe the true routing action $\boldsymbol{f}^u(t)$ taken by uncontrollable nodes and update virtual queues:

$$X_{ik}(t+1) = \Big[X_{ik}(t) + a_{ik}(t) + \sum_{j\in\mathcal{N}} g_{jik}(t) - \sum_{j\in\mathcal{N}} g_{ijk}(t)\Big]^+$$

$$Y_{ijk}(t+1) = Y_{ijk}(t) + \Delta_{ijk}(t)$$

where $\Delta_{ijk}(t)$ is defined in (1).

The following theorem gives the performance of TMW.

Theorem 3.2. *The worst-case queue length achieved by Tracking-MaxWeight is*

$$\sum_{i,k} Q_{ik}(T) = O\Big(T^{2/3} V_T^{1/3}\Big).$$

Proof. The high-level idea is to first prove a queue length upper bound achieved by Tracking-MaxWeight under a stronger "niceness" assumption, called W-constraints, and then show that the V_T metric can be used to relax the W-constraints. The detailed proof is presented in the technical report [12]. □

An important observation about the above theorem is that Tracking-MaxWeight achieves sublinear queue length whenever $V_T = o(T)$. Noticing that sublinear queue length cannot be achieved by any causal policy if $V_T = \Omega(T)$ (see Theorem 3.1), we have the following corollary.

Corollary 3.3. *Sublinear queue length is achievable if and only if $V_T = o(T)$.*

This corollary shows that Tracking-MaxWeight achieves sublinear queue length whenever possible. We will formalize this observation in terms of the notion of stability region in Section 3.3.

3.3 Stability Region

In this section, we characterize the stability region for a network with adversarial uncontrollable nodes. The notion of stability region describes a necessary and sufficient condition such that rate stability could be achieved, as is given in the following theorem.

Theorem 3.4. *A network can be stabilized by some causal policy under a given sequence of uncontrollable actions $\{\boldsymbol{f}^u(t)\}_{t\le T}$ if and only if $V_T\Big(\{\boldsymbol{f}^u(t)\}_{t\le T}\Big) = o(T)$.*

Proof. The sufficiency follows from Theorem 3.2, and the necessity is proved in the technical report [12] □

We say that a network control algorithm is *throughput-optimal* if it achieves rate stability whenever the network is inside the stability region. Combining Theorem 3.2 and Theorem 3.4, we have the following corollary.

Corollary 3.5. *The TMW algorithm is throughput-optimal with respect to any given sequence of uncontrollable actions $\{\boldsymbol{f}^u(t)\}_{t\le T}$. That is, as long as there exists some causal policy that achieves rate stability under $\{\boldsymbol{f}^u(t)\}_{t\le T}$, the TMW algorithm is also stable.*

4 NETWORK UTILITY MAXIMIZATION

In the previous section, we studied the network stability problem where the objective is to achieve rate stability whenever the network is inside the stability region. However, when exogenous packet arrivals are outside the stability region, admission control needs to be performed in order to maintain network stability. In this section, we extends the previous network stability problem by accounting for joint admission control and routing decisions, such that a certain *network utility* metric can be maximized while keeping the network stable. This is referred to as the **Network Utility Maximization (NUM)** problem.

Specifically, let $\boldsymbol{\gamma}(t)$ be the vector of admitted exogenous arrivals in slot t, where $\gamma_{ik}(t) \in [0, a_{ik}(t)]$ is the number of admitted packets for flow k to node i in slot t. Similar to the routing vector $\boldsymbol{f}(t)$, the admission control vector $\boldsymbol{\gamma}(t)$ can be decomposed into two parts: $\boldsymbol{\gamma}(t) = (\boldsymbol{\gamma}^c(t), \boldsymbol{\gamma}^u(t))$ where $\boldsymbol{\gamma}^c(t)$ and $\boldsymbol{\gamma}^u(t)$ is the vector of admitted arrivals to controllable nodes and uncontrollable nodes,

respectively. Note that $\boldsymbol{\gamma}^u(t)$ is determined by the uncontrollable nodes and cannot be regulated by the network operator. In this context, the couple $(\boldsymbol{\gamma}^c(t), \boldsymbol{f}^c(t))$ is called a **controllable action** while $(\boldsymbol{\gamma}^u(t), \boldsymbol{f}^u(t))$ is referred to as an **uncontrollable action**.

Denote by $U(\boldsymbol{\gamma}(t)) = \sum_{i,k} U_{ik}(\gamma_{ik}(t))$ the network utility function, where each $U_{ik}(\gamma_{ik}(t))$ is the utility gained by admitting $\gamma_{ik}(t)$ exogenous arrivals in flow k to node i. Typical examples include $U(\boldsymbol{\gamma}(t)) = \sum_{i,k} \gamma_{ik}(t)$ (total throughput), $U(\boldsymbol{\gamma}(t)) = \sum_{i,k} \log(1 + \gamma_{ik}(t))$ (proportional fairness), etc. Since the admitted arrival vector $\boldsymbol{\gamma}(t)$ can be decomposed into $\boldsymbol{\gamma}(t) = (\boldsymbol{\gamma}^c(t), \boldsymbol{\gamma}^u(t))$, we may also write the network utility function as $U(\boldsymbol{\gamma}(t)) = U(\boldsymbol{\gamma}^c(t), \boldsymbol{\gamma}^u(t))$. In addition, we assume that the utility function is bounded:

$$U_{\min} \le U(\boldsymbol{\gamma}) \le U_{\max}, \ \forall \boldsymbol{\gamma}.$$

A policy π generates a sequence of admission control and routing decisions for controllable nodes, i.e., $\pi = \{\boldsymbol{\gamma}^c(t), \boldsymbol{f}^c(t)\}_{t \le T}$. Let $\{\boldsymbol{\gamma}^u(t), \boldsymbol{f}^u(t)\}_{t \le T}$ be the sequence of uncontrollable actions. Our objective is to find a causal control policy for controllable nodes that maximizes the cumulative network utility while keeping the total queue length small.

NUM:

$$\max_{\boldsymbol{\gamma}^c(t), \boldsymbol{f}^c(t)} \quad \sum_{t \le T} U(\boldsymbol{\gamma}^c(t), \boldsymbol{\gamma}^u(t)) \tag{4}$$

$$\text{s.t.} \ \sum_{t \le T} \left(\gamma_{ik}(t) + \sum_j \widetilde{f}_{jik}(t) - \sum_j \widetilde{f}_{ijk}(t) \right) = 0, \ \forall i, k. \tag{5}$$

The objective (4) is to maximize the total network utility gained within the time horizon. The constraint (5) requires that the total actual arrivals to each queue should not exceed the total amount of actual departures from that queue during the time horizon. Any optimal solution to **NUM** is a utility maximizing policy subject to the constraint (5) that it clears all the backlogs within the time horizon. We assume that there is at least one feasible solution to **NUM**.

Note that a network with adversarial uncontrollable nodes may not have any steady state or well-defined time averages, so we introduce the notion of *utility regret* to measure the finite-time performance achieved by a network control policy.

Definition 4.1. Given the time horizon T, the utility regret achieved by a policy $\pi = \{\boldsymbol{\gamma}^c(t), \boldsymbol{f}^c(t)\}_{t \le T}$ under a sequence of uncontrollable actions $\{\boldsymbol{\gamma}^u(t), \boldsymbol{f}^u(t)\}_{t \le T}$ is defined to be

$$\mathcal{R}_T^\pi = \sum_{t \le T} U(\boldsymbol{\gamma}^{c*}(t), \boldsymbol{\gamma}^u(t)) - \sum_{t \le T} U(\boldsymbol{\gamma}^c(t), \boldsymbol{\gamma}^u(t)), \tag{6}$$

where $\{\boldsymbol{\gamma}^{c*}(t)\}_{t \le T}$ is an optimal solution to ***NUM*** generated by an "oracle" that knows the sequence of uncontrollable actions in advance.

Intuitively, the notion of utility regret captures the utility difference between a causal policy and an ideal T-slot lookahead non-causal policy. Note that a causal policy may trivially maximize the network utility by simply ignoring the constraint (5) (e.g., admitting all exogenous traffic) such that the utility regret become zero or even negative[1]. However, such an action may significantly violate the constraint (5) and lead to large total queue length. As a result, there is a tradeoff between the utility regret and the total queue length achieved by a causal control policy, referred to as the *utility-delay tradeoff*. In the following, our objective is to find a causal policy that achieves good utility-delay tradeoffs where the constraint (5) may be violated by a bounded amount (note that as the benchmark in defining utility regret, the optimal oracle policy for **NUM** still need to satisfy the constraint (5)).

A desirable first order characteristic of a "good" policy π is that it simultaneously achieves sublinear utility regret and sublinear queue length w.r.t. the time horizon T, i.e., $\mathcal{R}_T^\pi = o(T)$ and $\sum_{i,k} Q_{ik}^\pi(T) = o(T)$. Sublinear utility regret guarantees that $\mathcal{R}_T^\pi / T \to 0$ as the time horizon $T \to \infty$, meaning that the time-average utility gained under policy π asymptotically approaches that under the optimal non-causal policy. In other words, the long-term time-average utility is optimal. Sublinear queue length ensures $\sum_{i,k} Q_{ik}^\pi(T)/T \to 0$ as $T \to \infty$, which is equivalent to rate stability. Note that simultaneously achieving sublinear utility regret and sublinear queue length is equivalent to maximizing long-term time-average utility subject rate stability, which is the goal of traditional stochastic network optimization [16].

In addition, similar to that in the network stability problem, our analysis utilizes the "niceness" metric V_T for a sequence of uncontrollable actions $\{\boldsymbol{\gamma}^u(t), \boldsymbol{f}^u(t)\}_{t \le T}$:

$$V_T\left(\{\boldsymbol{\gamma}^u(t), \boldsymbol{f}^u(t)\}_{t \le T}\right) \triangleq \max_{t \le T} \sum_{i,k} Q_{ik}^*(t),$$

where $\mathbf{Q}^*(t)$ is the queue length vector achieved by the optimal policy to **NUM** in slot t under the given sequence of uncontrollable actions. Note that if there are multiple optimal solutions to **NUM**, the one with the largest $V_T(\cdot)$ is considered. We may simply write $V_T\left(\{\boldsymbol{\gamma}^u(t), \boldsymbol{f}^u(t)\}_{t \le T}\right)$ as V_T if the context is clear.

In the following, we first provide a lower bound on the utility-delay tradeoff in Section 4.1. Then we develop an efficient algorithm for solving **NUM** and analyze its performance in Section 4.2.

4.1 Lower Bound on Utility-Delay Tradeoff

The following theorem provides a lower bound on the utility-delay tradeoff in a network with adversarial uncontrollable nodes

Theorem 4.2. *For any causal policy π, there exists a sequence of uncontrollable actions $\{\boldsymbol{\gamma}^u(t), \boldsymbol{f}^u(t)\}_{t \le T}$ such that*

$$\mathcal{R}_T^\pi + c \sum_{i,k} Q_{ik}^\pi(T) \ge c' V_T,$$

where $c, c' > 0$ are constants.

Proof. The proof constructs a similar example to that used in the proof of Theorem 3.1 but accounts for admission control actions. See the technical report [12] for details. □

Theorem 4.2 shows that if $V_T = \Omega(T)$, then no causal policy can simultaneously achieve sublinear utility regret and sublinear queue length. On the other hand, if $V_T = o(T)$, there *might* exist some causal policy that attains sublinear utility regret and sublinear queue length simultaneously, which we investigate in the next section.

[1] The negative regret may happen since we are comparing against an optimal oracle policy for **NUM** that must satisfy the constraint (5)

4.2 Tracking Drift-plus-Penalty (TDP) Algorithm

Now we develop an algorithm that achieves sublinear utility regret and sublinear queue length whenever $V_T = o(T)$. The algorithm is called Tracking Drift-plus-Penalty (TDP), which enhances the well-known Drift-plus-Penalty algorithm [18] with an explicit tracking of the actions taken by uncontrollable nodes. Throughout this section, let $\{\boldsymbol{\gamma}^u(t), \boldsymbol{f}^u(t)\}_{t\leq T}$ be the sequence of actions that were actually taken by uncontrollable nodes in the time horizon.

The detailed algorithm description is given in Algorithm 2. It is almost the same as the Tracking-MaxWeight algorithm (see Algorithm 1) except that the drift minimization problem (2) is replaced by the minimization of a drift-plus-penalty term:

$$\begin{aligned}\min_{\boldsymbol{\gamma}^c(t),\mathbf{g}(t)} \; & \sum_{i,k} X_{ik}(t)\Big[\gamma_{ik}(t) + \sum_j g_{jik}(t) - \sum_j g_{ijk}(t)\Big] \\ & + \sum_{i,j,k} Y_{ijk}(t)\Big(g_{ijk}(t) - \widetilde{f}_{ijk}(t)\Big) \\ & - VU(\boldsymbol{\gamma}^c(t), \boldsymbol{\gamma}^u(t)),\end{aligned} \tag{7}$$

where $VU(\boldsymbol{\gamma}^c(t), \boldsymbol{\gamma}^u(t))$ is the additional penalty term for utility maximization. The parameter $V > 0$ tunes the utility-delay tradeoff: the larger V is, the more penalty it creates for utility loss, thus achieving larger utility at the price of a larger queue length. Note that $\boldsymbol{\gamma}^c(t)$ and $\boldsymbol{g}(t)$ can be optimized separately and thus we further decompose (7) into two sub-problems: the admission control problem (step 1) and the routing/scheduling problem (step 3). Note also that the optimization problems presented in step 1 and step 3 have cleaned up irrelevant constants (i.e., $\widetilde{f}_{ijk}(t)$ and $\boldsymbol{\gamma}^u(t)$), and rearranged terms to simplify the expressions. Finally, it should be mentioned that the TDP algorithm does not produce any "imagined admission control action" for the true uncontrollable admission control action $\boldsymbol{\gamma}^u(t)$ (only "imagined routing action" $\mathbf{g}^u(t)$ is needed).

The performance of the TDP algorithm is given in the following theorem.

Theorem 4.3. *The TDP algorithm with parameter V achieves $O\Big(\frac{V_T^{2/3}T^{4/3}}{V} + \frac{V_T^{1/3}T^{7/6}}{V^{1/2}}\Big)$ utility regret and the total queue length is $O\Big(V_T^{1/3}T^{2/3} + T^{1/2}V^{1/2} + V^{1/4}V_T^{1/6}T^{7/12}\Big)$.*

Proof. The proof first derives an upper bound on the drift-plus-penalty term. Then we carefully constructs a quadratic inequality with respect to the maximum virtual queue length $\sum_{i,k} X_{ik}(T)$ using the drift-plus-penalty upper bound and the definition of V_T, which further yields an upper bound on $\sum_{i,k} X_{ik}(T)$. Finally, we bound the virtual queue length $\sum_{i,j,k} |Y_{ijk}(T)|$, the physical queue length $\sum_{i,k} Q_{ik}(T)$ and the utility regret $\mathcal{R}_T$. The detailed proof is presented in the technical report [12]. □

There are several observations about Theorem 4.3. First, by selecting a proper value of V, the TDP algorithm simultaneously achieves sublinear utility regret and sublinear queue length whenever $V_T = o(T)$. For example, if $V_T = \Theta(T^{1/2})$ and we set $V = \Theta(T^{4/5})$, then the utility regret is $O(T^{14/15})$ and the total queue length is $O(T^{9/10})$. Noticing that sublinear utility regret and sublinear queue length cannot be simultaneously achieved by any causal policy if $V_T = \Omega(T)$ (Theorem 4.2), we have the following corollary.

Algorithm 2 Tracking Drift-plus-Penalty (TDP)

Input: parameter $V > 0$ that tunes the utility-delay tradeoff

1: (Admission Control) For each controllable node $i \in C$, determine the amount of admitted traffic $\gamma_{ik}(t)$ by solving the following problem

$$\max_{\gamma_{ik}(t)\leq a_{ik}(t)} \sum_k \Big[VU_{ik}\Big(\gamma_{ik}(t)\Big) - \gamma_{ik}(t)X_{ik}(t)\Big]. \tag{8}$$

2: (Routing and Scheduling) Determine the controllable routing action $\mathbf{g}^c(t)$ and the *imagined* uncontrollable routing action $\mathbf{g}^u(t)$ by solving the following problem:

$$\max_{\mathbf{g}(t)\in\mathcal{F}_{\omega t}} \sum_{(i,j)} \sum_k g_{ijk}(t)W_{ijk}(t), \tag{9}$$

where

$$W_{ijk}(t) = X_{ik}(t) - X_{jk}(t) - Y_{ijk}(t).$$

3: Controllable nodes execute the routing decision $\mathbf{g}^c(t)$.

4: Observe the true routing action $\boldsymbol{f}^u(t)$ taken by uncontrollable nodes and update virtual queues:

$$X_{ik}(t+1) = \Big[X_{ik}(t) + \gamma_{ik}(t) + \sum_{j\in\mathcal{N}} g_{jik}(t) - \sum_{j\in\mathcal{N}} g_{ijk}(t)\Big]^+$$

$$Y_{ijk}(t+1) = Y_{ijk}(t) + \Delta_{ijk}(t)$$

where $\Delta_{ijk}(t)$ is defined in (1).

Corollary 4.4. *Sublinear utility regret and sublinear queue length are simultaneously achievable if and only if $V_T = o(T)$.*

Moreover, by the definition of utility regret and rate stability, we can conclude that the TDP algorithm achieves the optimal long-term average utility while keeping the network rate stable whenever possible.

5 SIMULATION RESULTS

In this section, we numerically evaluate the performance of TMW and TDP in different scenarios.

5.1 Scenario I: Stochastic Partially-Controllable Network

We first study a mild case where uncontrollable nodes use some fixed randomized policy. Specifically, consider the partially-controllable network shown in Figure 3. There are two flows: $1 \to 4$ and $6 \to 4$. Each node in the network needs to make a routing and scheduling decision in every time slot. The constraint is that each node can transmit to only one of its neighbors in each time slot and the transmission rate over each link cannot exceed its capacity. Node 2 and node 3 are uncontrollable nodes that use randomized policies. Specifically, uncontrollable node 2 uses a randomized routing algorithm that transmits any packets it received to either node 3 or node 5 with an equal probability in each time slot. Uncontrollable node 3 uses a randomized scheduling policy that serves flow $1 \to 4$ or flow $6 \to 4$ with an equal probability in each time slot. The arrival rate of flow $6 \to 4$ is 5. In this case, it can be shown that the maximum supportable arrival rate for flow $1 \to 4$ is 25 given the routing constraints and the behavior of uncontrollable nodes.

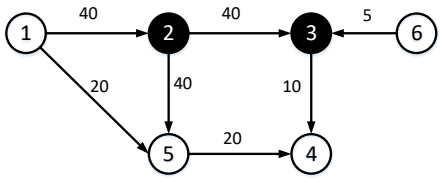


Figure 3: Network topology used in simulation scenario I. The number next to each link is its capacity. Each node can only transmit to one of its neighbors in each slot. Black nodes are uncontrollable nodes that use randomized policies.

In Figure 4(a), we compare Tracking-MaxWeight with the well-known MaxWeight algorithm (i.e., BackPressure routing), in terms of the supportable rate for flow $1 \to 4$. Specifically, Figure 4(a) shows the total queue length achieved by MaxWeight and Tracking-MaxWeight under different system loads (if the load is ρ, then the arrival rate of flow $1 \to 4$ is 25ρ while the arrival rate of flow $6 \to 4$ is fixed to 5). It is observed that MaxWeight can only support a load of 0.4 (the queue length under MaxWeight blows up at load ≈ 0.4). By comparison, our Tracking-MaxWeight achieves the optimal throughput, i.e., $\rho = 1$.

We further examine the behavior of the Tracking-MaxWeight algorithm in Figure 4(b) and Figure 4(c). Specifically, Figure 4(b) shows the queue length trajectory for the physical queue $\mathbf{Q}(t)$ and the two virtual queues $\mathbf{X}(t), \mathbf{Y}(t)$. As our theory predicts, both the physical queue $\mathbf{Q}(t)$ and the two virtual queues $\mathbf{X}(t), \mathbf{Y}(t)$ are stable under the TMW algorithm. In addition, Figure 4(b) also validates the queue length upper bound of the TMW algorithm (Theorem 3.2). Figure 4(c) shows the learning curve of the TMW algorithm for the uncontrollable policy used by node 3. In particular, node 3 uses randomized scheduling that serves flow $1 \to 4$ and flow $6 \to 4$ with an equal probability 0.5. It is observed in Figure 4(c) that the TMW algorithm quickly learns the service probability for flow $1 \to 4$ at node 3 (i.e., the "imagined uncontrollable action" in TMW approaches the true uncontrollable action).

5.2 Scenario II: Adaptive Adversarial Nodes (stability)

Next, we study an adversarial scenario where uncontrollable nodes behave maliciously. Consider a similar example as used in the proof of Theorem 3.1, where the network topology is shown in Figure **??**. Node 2 is uncontrollable while the remaining nodes are controllable. There are two flows in the network: $1 \to 3$ (flow index $k = 1$) and $1 \to 4$ (flow index $k = 2$). The controllable action is the routing decision for node 1. The only constraint is the link capacity constraint, i.e., $\sum_k f_{ijk}(t) \le C_{ij}$. In our construction, the capacity of each link is 2, i.e., $C_{ij} = 2$ for any $i \to j$. Time is divided into frames of $Z = V_T$ slots. In the first $Z/2$ slots of each frame r, the exogenous arrival rate for both flows is 2, and the uncontrollable node (i.e., node 2) allocates full link capacity to the two flows:

$$f_{231}(t) = 2,\ f_{242}(t) = 2,\ \forall t = rZ, \cdots, rZ + Z/2 - 1.$$

In the remaining $Z/2$ slots of each frame, there are no exogenous arrivals to either flow, and uncontrollable node 2 allocates zero rate to the flow with a larger total backlog in the network while allocates full rate to the other flow. Specifically, suppose that flow k' has a larger total queue backlog and flow k'' has a smaller backlog. Then the routing decision at uncontrollable node 2 is

$$f_{23k'}(t) = 0,\ f_{24k''}(t) = 2,\ \forall t = rZ + Z/2, \cdots, rZ + Z - 1.$$

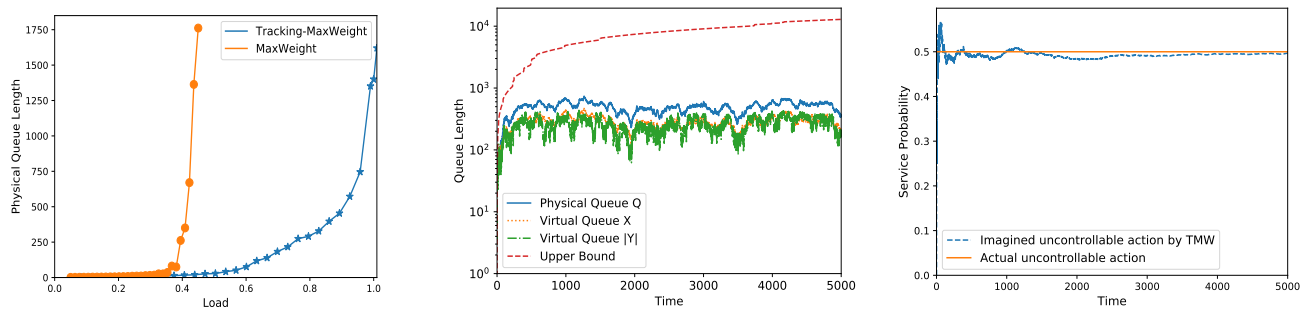
If the two flows have the same queue backlog in the network, ties are broken randomly. Our objective is to stabilize the network.

Figure 5 shows the performance of Tracking-MaxWeight and MaxWeight in this scenario. The queue length lower bound, as given by Theorem 3.1, is also plotted in the figures for comparison. Specifically, Figures 5(a)-5(c) illustrate the growth of queue length w.r.t. the time horizon T under different values of V_T. When $V_T = \Theta(1)$, i.e., V_T does not grow with the time horizon T, the queue lengths achieved by both algorithms remain at some constants when T is sufficiently large. When $V_T = \Theta(\sqrt{T})$, the queue lengths achieved by both algorithms increase with the time horizon T, yet the growth rate is sublinear. When $V_T = \Theta(T)$, both algorithms have linearly-increasing queue length w.r.t. T. In fact, even the queue length lower bound becomes linear in T, implying that no algorithms can have sublinear queue length in this case. In addition, Tracking-MaxWeight achieves a smaller queue length than MaxWeight, although their overall queue length growth rates are quantitatively the same, given the adaptive adversarial behavior of the uncontrollable node. Figure 5(d) shows the growth of queue length w.r.t. the increase of the niceness metric V_T, where we fix the time horizon $T = 10^4$ slots. It is observed that the Tracking-MaxWeight algorithm empirically achieves a queue length that grows linearly in V_T, which shows that the analysis in Theorem 3.2 is not tight in this scenario.

5.3 Scenario III: Adaptive Adversarial Nodes (utility)

Finally, we evaluate the performance of the Tracking Drift-plus-Penalty (TDP) algorithm for the network utility maximization problem in a similar scenario as in Section 5.2, where the uncontrollable node behave maliciously. The difference is that node 1 also needs to make an admission control decision to determine the amount of traffic entering the network for the two flows. Let $\gamma^{(k)}(t)$ be the amount of admitted packets for flow k in slot t. The utility function is $U(\boldsymbol{\gamma}(t)) = \sum_k \log(1 + \gamma^{(k)}(t))$ (proportional fairness). Such a scenario is also similar to the example that we use for proving the lower bound on utility-delay tradeoff (see Theorem 4.2).

Figure 6 illustrates the growth rate of the total queue length and the utility regret with the time horizon T under the TDP algorithm. First, when $V_T = \Theta(\sqrt{T})$, the TDP algorithm can simultaneously achieve sublinear utility regret and sublinear queue length, if the parameter V is set appropriately (for example, $V = \Theta(T^{3/4})$). Note that setting V to some very large value (e.g., $V = \Theta(T^2)$) still achieves sublinear utility regret and sublinear queue length, though the theoretical bound on queue length (see Theorem 4.3) is at least linear in T when $V = \Omega(T)$, which shows that the performance upper bound is not tight in this scenario. When $V_T = \Theta(T)$, the TDP algorithm fails to achieve desirable performance: either the utility regret or the queue length grows linearly with T. In fact, the lower bound in

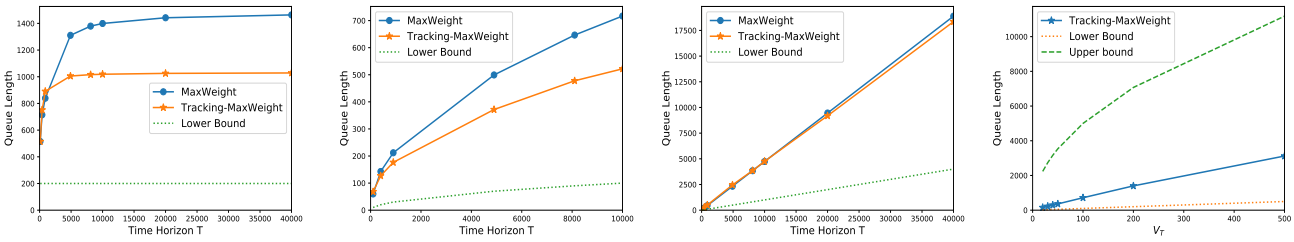


(a) Throughput performance of MaxWeight and Tracking-MaxWeight.

(b) Queue length under the TMW algorithm (load = 0.99).

(c) The TMW algorithm quickly learns that node 3 serves flow $1 \longrightarrow 4$ with probability 0.5.

Figure 4: Performance of the Tracking-MaxWeight (TMW) algorithm in Scenario 1.



(a) $V_T = O(1)$

(b) $V_T = O(\sqrt{T})$

(c) $V_T = O(T)$

(d) Queue Length vs. V_T

Figure 5: Performance of the Tracking-MaxWeight (TMW) algorithm in Scenario II. (a)-(c): growth of queue length w.r.t. the time horizon T under different values of V_T; (d) growth of queue length w.r.t. the niceness metric V_T where we fix the time horizon $T = 10^4$ slots.

Theorem 4.2 shows that no causal policy can achieve both sublinear utility regret and sublinear queue length if $V_T = \Theta(T)$.

Figure 7 shows the utility-delay tradeoff achieved by the TDP algorithm, where we fix the time horizon to be $T = 10^4$ slots and $V_T = \Theta(\sqrt{T})$. The lower bound (Theorem 4.2) and the performance upper bound (Theorem 4.3) are validated in the figure.

6 CONCLUSIONS

In this paper, we study efficient network control algorithms for a partially-controllable network with adversarial uncontrollable nodes. First, we investigate the network stability problem and provide a lower bound on the total queue length that can be achieved by any causal policy. A throughput-optimal algorithm, called Tracking-MaxWeight, is developed to stabilize any adversarial partially-controllable network. Next, we study the network utility maximization problem and develop a lower bound on the utility-delay tradeoff. We propose the Tracking Drift-plus-Penalty (TDP) algorithm and show that TDP achieves good utility-delay tradeoffs in any adversarial partially-controllable network.

REFERENCES

[1] Matthew Andrews, Kyomin Jung, and Alexander Stolyar. 2007. Stability of the Max-weight Routing and Scheduling Protocol in Dynamic Networks and at Critical Loads. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing (STOC '07)*. ACM, 145–154.

[2] M. Andrews and L. Zhang. 2002. Scheduling over a time-varying user-dependent channel with applications to high speed wireless data. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.* 293–302. https://doi.org/10.1109/SFCS.2002.1181952

[3] M. Andrews and L. Zhang. 2006. Scheduling Over Nonstationary Wireless Channels With Finite Rate Sets. *IEEE/ACM Transactions on Networking* 14, 5 (Oct 2006), 1067–1077. https://doi.org/10.1109/TNET.2006.882835

[4] Allan Borodin, Jon Kleinberg, Prabhakar Raghavan, Madhu Sudan, and David P Williamson. 2001. Adversarial queuing theory. *Journal of the ACM (JACM)* 48, 1 (2001), 13–38.

[5] Tianyi Chen, Qing Ling, and Georgios B Giannakis. 2017. An online convex optimization approach to dynamic network resource allocation. *arXiv preprint arXiv:1701.03974* (2017).

[6] Vicent Cholvi and Juan Echagüe. 2007. Stability of FIFO networks under adversarial models: State of the art. *Computer Networks* 51, 15 (2007), 4460–4474.

[7] Rene L Cruz. 1991. A calculus for network delay. I. Network elements in isolation. *IEEE Transactions on information theory* 37, 1 (1991), 114–131.

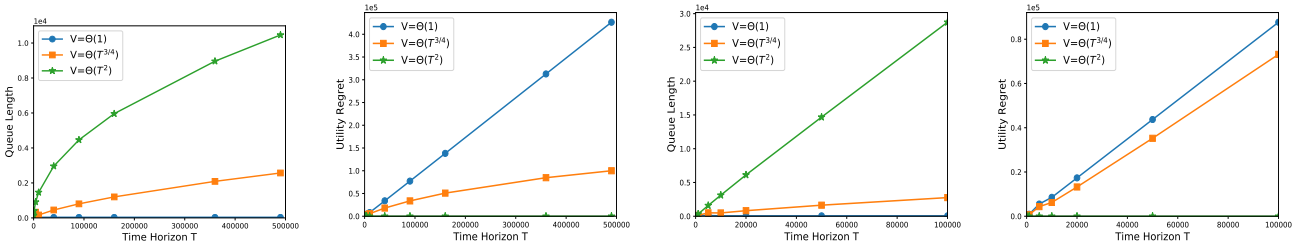
[8] Rodolphe Jenatton, Jim Huang, and Cédric Archambeau. 2015. Adaptive algorithms for online convex optimization with long-term constraints. *arXiv preprint arXiv:1512.07422* (2015).

[9] Nathaniel M Jones, Georgios S Paschos, Brooke Shrader, and Eytan Modiano. 2017. An overlay architecture for throughput optimal multipath routing. *IEEE/ACM Transactions on Networking* (2017).

[10] Qingkai Liang and Eytan Modiano. 2018. Minimizing Queue Length Regret Under Adversarial Network Models. *ACM SIGMETRICS* (2018).

[11] Qingkai Liang and Eytan Modiano. 2018. Network Utility Maximization in Adversarial Environments. *IEEE INFOCOM* (2018).

[12] Qingkai Liang and Eytan Modiano. 2019. Technical Report:. https://www.dropbox.com/s/x952j5vgny6hbou/uncontrollable-adversarial-tech-report.pdf?dl=0. (2019).



(a) Queue Length $V_T = \Theta(\sqrt{T})$ (b) Utility Regret $V_T = \Theta(\sqrt{T})$ (c) Queue Length $V_T = \Theta(T)$ (d) Utility Regret $V_T = \Theta(T)$

Figure 6: Performance of the Tracking Drift-plus-Penalty (TDP) algorithm in Scenario III.

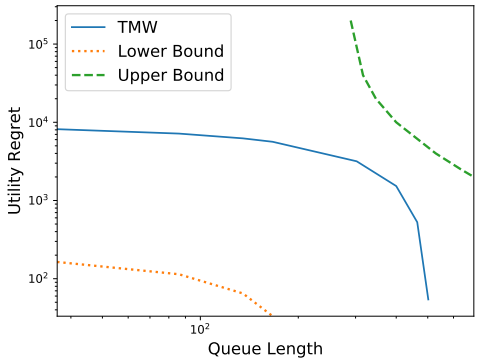


Figure 7: Tradeoffs between utility and total queue length (double log scale) achieved by the TDP algorithm in Scenario III. The time horizon is fixed to be $T = 10^4$ slots and $V_T = \Theta(\sqrt{T})$.

[13] Sungsu Lim, Kyomin Jung, and Matthew Andrews. 2014. Stability of the Max-weight Protocol in Adversarial Wireless Networks. *IEEE/ACM Trans. Netw.* 22, 6 (Dec. 2014), 1859–1872.

[14] David V Lindley. 1952. The theory of queues with a single server. In *Mathematical Proceedings of the Cambridge Philosophical Society*, Vol. 48. Cambridge University Press, 277–289.

[15] Mehrdad Mahdavi, Rong Jin, and Tianbao Yang. 2012. Trading regret for efficiency: online convex optimization with long term constraints. *Journal of Machine Learning Research* 13, Sep (2012), 2503–2528.

[16] Michael J Neely. 2010. Stochastic network optimization with application to communication and queueing systems. *Synthesis Lectures on Communication Networks* 3, 1 (2010), 1–211.

[17] Michael J Neely. 2010. Universal scheduling for networks with arbitrary traffic, channels, and mobility. In *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 1822–1829.

[18] Michael J Neely, Eytan Modiano, and Chih-Ping Li. 2008. Fairness and optimal stochastic control for heterogeneous networks. *IEEE/ACM Transactions on Networking (TON)* 16, 2 (2008), 396–409.

[19] Michael J Neely and Hao Yu. 2017. Online Convex Optimization with Time-Varying Constraints. *arXiv preprint arXiv:1702.04783* (2017).

[20] Georgios S Paschos and Eytan Modiano. 2014. Throughput optimal routing in overlay networks. In *Communication, Control, and Computing (Allerton), 2014 52nd Annual Allerton Conference on*. IEEE, 401–408.

[21] George S Paschos and Leandros Tassiulas. 2016. Sustainability of Service Provisioning Systems under Stealth DoS Attacks. *IEEE Transactions on Control of Network Systems* (2016).

[22] Anurag Rai, Rahul Singh, and Eytan Modiano. 2016. A Distributed Algorithm for Throughput Optimal Routing in Overlay Networks. *arXiv preprint arXiv:1612.05537* (2016).

[23] Ramesh K Sitaraman, Mangesh Kasbekar, Woody Lichtenstein, and Manish Jain. 2014. Overlay networks: An akamai perspective. *Advanced Content Delivery, Streaming, and Cloud Services* 51, 4 (2014), 305–328.

[24] Leandros Tassiulas and Anthony Ephremides. 1992. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE transactions on automatic control* 37, 12 (1992), 1936–1948.

[25] NS Walton. 2014. Two queues with non-stochastic arrivals. *Operations Research Letters* 42, 1 (2014), 53–57.

[26] Hao Yu, Michael Neely, and Xiaohan Wei. 2017. Online Convex Optimization with Stochastic Constraints. In *Advances in Neural Information Processing Systems*. 1427–1437.