A Control-Theoretic and Online Learning Approach to Self-Tuning Queue Management

Jiancheng Ye*, Kechao Cai†, Dong Lin*, Jiarong Li‡, Jianfei He§, and John C.S. Lui¶
*Network Technology Lab and Hong Kong Research Center, Huawei Technologies Co., Ltd.
†School of Electronics and Communication Engineering, Sun Yat-Sen University, China
‡Department of Electrical Engineering, Tsinghua University, China
§Department of Computer Science, The City University of Hong Kong
¶Department of Computer Science and Engineering, The Chinese University of Hong Kong
Email: {yejiancheng, lindong10}@huawei.com, caikch3@mail.sysu.edu.cn, jr-li16@tsinghua.org.cn,
jianfeihe2-c@my.cityu.edu.hk, cslui@cse.cuhk.edu.hk

***Abstract*—There is a growing trend that network applications not only require higher throughput, but also impose stricter delay requirements. The current Internet congestion control, which is driven by active queue management (AQM) algorithms interacting with the Transmission Control Protocol (TCP), has been playing an important role in supporting network applications. However, it still exhibits many open issues. Most of AQM algorithms only deploy a single-queue structure that cannot differentiate flows and easily leads to unfairness. Moreover, the parameter settings of AQM are often static, making them difficult to adapt to the dynamic network environments. In this paper, we propose a general framework for designing "self-tuning" queue management (SQM), which is adaptive to the changing environments and provides fair congestion control among flows. We first present a general architecture of SQM with fair queueing and propose a general fluid model to analyze it. To adapt to the stochastic environments, we formulate a stochastic network utility maximization (SNUM) problem, and utilize online convex optimization (OCO) and control theory to develop a distributed SQM algorithm which can self-tune different queue weights and control parameters. Numerical and packet-level simulation results show that our SQM algorithm significantly improves queueing delay and fairness among flows.**

I. Introduction

There has been a growing trend that network applications such as augmented reality (AR), virtual reality (VR), and interactive gaming require higher throughput, while imposing stricter end-to-end delay requirements. However, it is challenging to simultaneously satisfy these two requirements. When network flows with large size (i.e., elephant flows) compete with each other for limited link bandwidths, network congestion usually occurs, leading to increasing queueing delay and reduced network throughput due to packet loss. Thus, congestion control is crucial for the performance of network applications. In the Internet, the Transmission Control Protocol (TCP) primarily interacts with active queue management (AQM) algorithms to achieve congestion control [1].

Although research on congestion control driven by TCP and AQM has been active for more than two decades, there still exist many open and fundamental issues. Specifically, as the key to addressing the bufferbloat problem [2], the current AQM policies have the following issues. Firstly, existing AQM algorithms (such as RED [3], CoDel [4], and PIE [5]) mostly deploy a single queue in the router buffer such that all arriving packets from different flows have to share this single queue's resource. However, a single queue generally cannot differentiate flows and protect conservative flows from being affected by aggressive ones, which in turn leads to unfairness. For example, [6] showed that a FAST TCP [7] flow may acquire much larger portion of link bandwidth than a TCP Reno [8] flow if they share the same bottleneck link. SFQ-CoDel [9] uses multiple queues for monitoring different flows, but each queue is simply managed by the original CoDel algorithm which may have stability issues when using the fixed default parameter settings [10]. Secondly, a large number of existing works on AQM just analyzed a single-bottleneck network topology. There are only few related works that investigate more general multi-bottleneck scenarios, such as [11] and [12]. Thus, it is important to develop a general model and conduct detailed analysis for AQM algorithms managing multiple queues in multi-bottleneck networks. Thirdly, the parameter settings of existing AQM algorithms are often static, making them difficult to adapt to the dynamic network environments, which may further lead to system instability (see Section V-B). In fact, automating the parameter tuning of AQM has been emphasized by the Internet engineering task force (IETF) in RFC 7567 [13], but it is still an open problem.

To address the aforementioned issues and facilitate adaptive and fair congestion control, we first propose a general architecture of a "self-tuning" queue management (SQM) scheme. SQM maps elephant TCP flows to different queues, each of which is managed by an independent AQM algorithm with self-tuning capabilities. Scheduling across different queues can be performed by the weighted fair queueing (WFQ) [14] or deficit round robin (DRR) [15] with *tunable* queue weights. We then propose a general fluid model to analyze the interactions between TCP and SQM in a multi-bottleneck network. In order to adapt to the dynamic network environments, we further formulate a stochastic network utility maximization (SNUM) problem for the TCP/SQM system, and apply online convex optimization (OCO) [16] techniques to tackle it. Based

978-1-6654-6824-4/22/$31.00

on the solution to the SNUM and the stability analysis of the TCP/SQM system, we develop a novel SQM algorithm which can dynamically self-tune different queue weights and control parameters in a distributed manner.

Contributions: In this paper, we propose a general framework to design and analyze SQM schemes, which can facilitate adaptive and fair congestion control on the Internet so as to improve queueing delay and fairness among flows. To fill the aforementioned gaps, we make the following contributions:

- We first present a general architecture of SQM equipped with adaptive fair queueing so as to dynamically regulate different flows. We then propose a general fluid model to describe the interactions between TCP flows and SQM in a general network setting with multiple bottleneck routers and multiple queues in each router.
- We conduct equilibrium and stability analysis for the general TCP/SQM system. Specifically, we derive sufficient conditions for the existence and uniqueness of an equilibrium point in the system. Furthermore, we derive sufficient conditions for the local asymptotic stability of each SQM subsystem with feedback delays.
- To capture the dynamic nature of the network environments, we formulate a SNUM problem for the TCP/SQM system. The SNUM extends the classic (static) NUM [17] to stochastic environments. We then apply OCO techniques and develop an online gradient descent (OGD)-based algorithm to solve the SNUM. Moreover, based on the solution to the SNUM and the derived stability conditions, we propose a distributed SQM algorithm which can self-tune different queue weights and control parameters. To the best of our knowledge, this is the first work that applies OCO to the self-tuning of AQM.
- We perform both numerical and packet-level simulations, which verify our analysis and show that our SQM algorithm significantly improves queueing delay and fairness among flows, compared to existing AQM algorithms.

The rest of the paper is organized as follows. Section II presents a general architecture of SQM and a corresponding general model for the TCP/SQM system. In Section III, we conduct equilibrium and stability analysis for the TCP/SQM system. Section IV first formulates a SNUM problem for the TCP/SQM system and develops an OGD-based algorithm to solve it, and then further proposes our SQM algorithm. In Section V, we present both numerical and packet-level simulation results. Finally, we conclude the paper in Section VI.

II. System Design and Model

In this section, we propose a general architecture of SQM, and develop a fluid model for a general TCP/SQM system.

A. General Architecture of SQM

A general architecture of SQM is depicted in Fig. 1. It consists of a classifying & mapping component and a queue management component. Due to space limitation, we focus on the queue management component, while the classifying & mapping component can utilize existing algorithms for

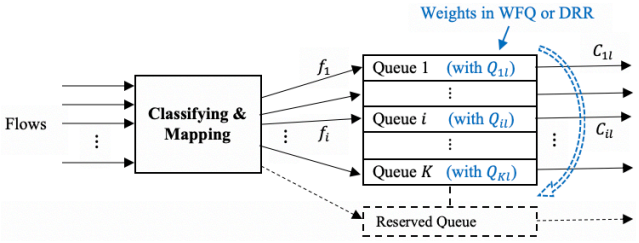


Fig. 1: A general architecture of a self-tuning queue management (SQM) scheme, where the queue weights Q_{il} and the control parameters of an AQM algorithm managing a queue can be tuned dynamically based on the network condition.

flow classification, such as ElephantTrap [18], HeavyKeeper [19], and hashing. Since a single-queue structure commonly used by existing AQM algorithms cannot differentiate flows and provide differentiated management, SQM utilizes a multi-queue structure where each queue serves a single flow and is managed by an independent AQM algorithm. Scheduling across different queues can be performed by WFQ [14] or DRR [15], according to the queue weights. Due to the dynamic nature of network environments, an AQM algorithm with fixed parameter settings cannot effectively adapt to the changing environments. Therefore, SQM enables self-tuning for both the control parameters of the AQM algorithms managing different queues and the queue weights in WFQ or DRR, so as to improve the system stability and performance.

The specific procedures of SQM can be described as follows. When flows arrive at a SQM-enabled router, they will first be classified and mapped to different queues based on their properties. Note that SQM mainly aims to control and manage long-lived TCP flows (i.e., elephant flows, especially aggressive ones) since they occupy the majority of the network traffic. Short-lived flows (i.e., mice flows) can be handled by a reserved queue with a reserved output rate and default queue management policy, say, drop-tail or CoDel. The classification and mapping for different flows can be done by using existing algorithms. If the number of classified flows exceeds the maximum number of queues (supported by specific hardware) denoted by $(K+1)$, only the top-K flows need to be mapped to individual queues and the rest can be handled by the aforementioned reserved queue. After the classification and mapping, each queue can at most contain one elephant flow. An independent self-tuning AQM algorithm is then used to manage the packets for each queue. Finally, WFQ or DRR with adaptive queue weights is utilized to perform scheduling across different queues.

B. General Fluid Model for TCP/SQM System

In this subsection, we develop a general fluid model to describe the interactions between long-lived TCP flows and SQM routers in a general network setting. To be consistent with our subsequent formulation of SNUM problem, the total time under consideration is composed of a number of periods indexed by an integer t, where $t = 1, 2, ..., T$. In a period t, we use τ to denote a time instant within that period.

Let us formally define the fluid model which describes the dynamics of a general TCP/SQM system in a period t. Specifically, let $N_t(\tau)$ be the number of TCP flows at time τ in period t and L_t be the number of bottleneck links in period t in the network. Let $A_t(i)$ be the set of bottleneck links that flow i traverses in period t, and $B_t(l)$ be the set of flows that pass through link l in period t. The rate of TCP flow i at time τ in period t is denoted by $x_{i,t}(\tau)$. We use $U_{i,t}(x_{i,t})$ to denote a utility function for TCP flow i in terms of its rate in period t, and choose the widely used α-fair utility function [20] defined as follows (note that our model can be easily extended to a weighted α-fair utility function):

$$U_{i,t}(x_{i,t}) = \begin{cases} \log x_{i,t}, & \alpha = 1, \\ (1-\alpha)^{-1} x_{i,t}^{1-\alpha}, & \alpha \neq 1,\ \alpha \geq 0. \end{cases} \tag{1}$$

Note that α-fair utility function can cover many fairness policies. For example, it corresponds to maximum throughput when $\alpha = 0$, to proportional fairness when $\alpha = 1$, and can achieve max-min fairness when $\alpha \to \infty$ [21]. Moreover, it also includes a number of TCP algorithms as special cases, such as FAST TCP [7], TCP Vegas [22], and Scalable TCP [23] when $\alpha = 1$, or TCP Reno [8] when $\alpha = 2$ [24].

A summary of notations used in our model is shown in Table I. The round-trip time of TCP flow i in period t is denoted by $D_{i,t}$. We use $q_{il,t}(\tau)$ and $p_{il,t}(\tau)$ to denote the length of queue i at link l at time τ and the price of queue i at link l at time τ, respectively. Here, the queue price $p_{il,t}(\tau)$ can be packet dropping probability or queueing delay, depending on the specific TCP and queue management algorithms used. Let C_l be the capacity of link l and $C_{il,t}(\tau)$ be the output rate of queue i at link l at time τ in period t. We use $\beta_{il,t}$ to denote the target queue length for queue i at link l in period t, and let $\sigma_{il,t}$ be a parameter for controlling the drop probability of queue i at link l in period t. They can be considered as control parameters of the AQM algorithm managing queue i. In particular, $\beta_{il,t}$ is used to control the average queue length to a predefined value. The weight (or quantum if DRR is used) for queue i at link l in period t is denoted by $Q_{il,t}$.

Using (1) as the utility function, our proposed general model for the TCP/SQM system which can accommodate different versions of TCP flows in period t is described by a set of delay differential equations (DDEs) in (2)–(5):

$$\dot{x}_{i,t}(\tau) = k_{i,t}\left(1 - \frac{\sum_{l \in A_t(i)} p_{il,t}(\tau - D_{i,t})}{U'_{i,t}(x_{i,t}(\tau))}\right), \quad i \in \mathcal{N}_t, \tag{2}$$

$$\dot{q}_{il,t}(\tau) = \begin{cases} x_{i,t}(\tau) - C_{il,t}(\tau), & q_{il,t}(\tau) > 0, \\ \left[x_{i,t}(\tau) - C_{il,t}(\tau)\right]^+, & q_{il,t}(\tau) = 0, \end{cases} i \in B_t(l), l \in \mathcal{L}_t, \tag{3}$$

$$p_{il,t}(\tau) = \left[\frac{q_{il,t}(\tau) - \beta_{il,t}}{\sigma_{il,t}}\right]^+, \quad i \in B_t(l), \quad l \in \mathcal{L}_t, \tag{4}$$

$$C_{il,t}(\tau) = \frac{Q_{il,t}(\tau)}{\sum_{j \in B_t(l)} Q_{jl,t}(\tau)} C_l, \quad i \in B_t(l), \quad l \in \mathcal{L}_t, \tag{5}$$

where $\mathcal{N}_t$ denotes the set of flows appearing in period t and $\mathcal{L}_t$ represents the set of bottleneck links in period t.

TABLE I: A Summary of Notations

Notation	Meaning
$N_t(\tau)$	Number of TCP flows at time τ in period t
L_t	Number of bottleneck links in period t
$x_{i,t}(\tau)$	Rate of TCP flow i at time τ in period t
$U_{i,t}(x_{i,t})$	Utility function of TCP flow i in period t
$D_{i,t}$	Round-trip time of TCP flow i in period t
$q_{il,t}(\tau)$	Length of queue i at link l at time τ in period t
$p_{il,t}(\tau)$	Price (e.g., drop probability) of queue i at link l at time τ
C_l	Capacity of link l
$C_{il,t}(\tau)$	Output rate of queue i at link l at time τ in period t
$\beta_{il,t}$	Target queue length for queue i at link l in period t
$\sigma_{il,t}$	Control parameter for drop probability of queue i at link l
$Q_{il,t}$	Weight or quantum for queue i at link l in period t
$A_t(i)$	Set of bottleneck links that flow i traverses in period t
$B_t(l)$	Set of flows that pass through link l in period t
M_t	Total number of non-empty queues in period t

Equations (2)–(3) describe the evolutions (time derivatives) of flow rates and queue lengths based on their interactions, while the instantaneous queue prices and the instantaneous output rates of the queues are determined by (4)–(5).

It is important to point out that (2) is a modified version of a well-known differential equation describing the dynamics of TCP flows in the literature such as [7], [17], and [25]. It describes the evolution of the rate of flow i based on the relation between the end-to-end queue price (e.g., drop probability) and the marginal utility $U'_{i,t}$ (i.e., the derivative of $U_{i,t}$ with respect to $x_{i,t}(\tau)$). In (2), $k_{i,t} > 0$ is a constant of gain parameter which is determined by a specific TCP algorithm used by flow i. For example, TCP Reno was shown to have $k_{i,t} = \frac{1}{D_{i,t}^2}$ [7], [25]. Note that comparing to the delay-free models in [7], [17], and [25], we further consider feedback delays in (2) where $D_{i,t}$ is the average round-trip time of flow i.

The evolution of the length of queue i at link l is modeled by (3). To simplify the modeling and without loss of generality, flow i is mapped to queue i with length $q_{il,t}(\tau)$ if it traverses link l in period t. Note that this assumption does not introduce a restriction to the mapping. When flow i is arbitrarily mapped to a queue j at link l, one can locally renumber queue j to queue i. Define $[a]^+ = \max\{a, 0\}$. Since $q_{il,t}(\tau) \geq 0$, the time derivative of $q_{il,t}(\tau)$ cannot be negative when $q_{il,t}(\tau) = 0$.

The price of queue i at link l is described by (4). Here, we mainly consider drop probability as the queue price (though (4) can also describe queueing delay by letting $\beta_{il,t} = 0$ and $\sigma_{il,t} = C_{il,t}(\tau)$) since AQM algorithms generally carry out congestion control by dropping packets. In (4), the parameters $\beta_{il,t}$ and $\sigma_{il,t}$ control the instantaneous drop probability of queue i. A number of existing AQM algorithms with a target queue length or delay, such as CoDel [4] and RED [3], employ packet dropping policies that satisfy the general form (4).

The output rate of queue i at link l is expressed in (5). Recall that either WFQ or DRR can be used for scheduling in SQM. DRR [15] assigns a quantum to each queue and performs scheduling across queues based on their quanta. In

the long run, a quantum in DRR can be considered as a queue weight when calculating the output rate of a queue. Let $Q_{il,t}(\tau)$ be an instantaneous weight or quantum for queue i at link l at time τ. We have $Q_{il,t}(\tau) \le Q_{il,t}$. Note that we do not model the reserved queue since it mainly contains short-lived mice flows which generally cannot be controlled. Thus, C_l in (5) excludes the reserved capacity for the reserved queue.

III. Equilibrium and Stability Analysis

This section analyzes two important properties of the TCP/SQM system (2)–(5), namely, equilibrium and stability.

A. Equilibrium Analysis

First of all, we investigate the existence and uniqueness of an equilibrium in the system. By definition, the equilibrium of the system (2)–(5) in period t is determined by $\dot{x}_{i,t} = 0$ and $\dot{q}_{il,t} = 0$ for all $i \in \mathcal{N}_t$ and $l \in \mathcal{L}_t$. Note that variables without the time index τ in (2)–(5) possess fixed values during period t. We assume that the set $\mathcal{N}_t$ contains a fixed number of flows denoted by N_t (i.e., $|\mathcal{N}_t| = N_t$), when the system is in equilibrium in period t. Moreover, the set $\mathcal{L}_t$ contains L_t bottleneck links in period t (i.e., $|\mathcal{L}_t| = L_t$). Thus, the equilibrium equations of the system (2)–(5) are as follows:

$$\frac{1}{(x^*_{i,t})^\alpha} = \sum_{l \in A_t(i)} p^*_{il,t}, \quad i \in \mathcal{N}_t, \tag{6}$$

$$x^*_{i,t} = C_{il,t}, \quad i \in B_t(l), \quad l \in \mathcal{L}_t, \tag{7}$$

where

$$C_{il,t} = \frac{Q_{il,t}}{\sum_{j \in B_t(l)} Q_{jl,t}} C_l, \quad i \in B_t(l), \quad l \in \mathcal{L}_t. \tag{8}$$

When the system is in equilibrium, the time index τ can be ignored. We use $x^*_{i,t}$ and $p^*_{il,t}$ to denote the equilibrium rates and the equilibrium drop probabilities, respectively. Then, all $x^*_{i,t}$ and $p^*_{il,t}$ ($i \in \mathcal{N}_t$, $l \in \mathcal{L}_t$) form an equilibrium point of the system, which is determined by (6)–(7). Note that to derive (6) from (2), we make use of the α-fair utility function (1). In (7), $C_{il,t}$ denotes the output rate of queue i at link l in equilibrium, which can be expressed by (8). Comparing to (5), (8) uses the equilibrium values of weights $Q_{il,t}$ to compute the achievable output rates for the queues. Note that since (8) only considers bottleneck links that are fully utilized, the values of $Q_{il,t}$ should satisfy (8).

Let $M_t = \sum_{l \in \mathcal{L}_t} \left|B_t(l)\right|$ be the total number of non-empty queues in period t. Let $\overline{\boldsymbol{R}}_{\boldsymbol{t}}$ be an $N_t \times M_t$ augmented routing matrix in period t, which is expressed as:

$$\overline{\boldsymbol{R}}_{\boldsymbol{t}} = \begin{bmatrix} \boldsymbol{R}_{\boldsymbol{1}}^{(\boldsymbol{1})} & \cdots & \boldsymbol{R}_{\boldsymbol{1}}^{(\boldsymbol{L_t})} \\ \vdots & \ddots & \vdots \\ \boldsymbol{R}_{\boldsymbol{N_t}}^{(\boldsymbol{1})} & \cdots & \boldsymbol{R}_{\boldsymbol{N_t}}^{(\boldsymbol{L_t})} \end{bmatrix}, \tag{9}$$

where the vectors $\boldsymbol{R}_{\boldsymbol{i}}^{(\boldsymbol{l})} = [R_{i1l}, R_{i2l}, ..., R_{i|B_t(l)|l}]$ (for $i = 1, 2, ..., N_t$ and $l = 1, 2, ..., L_t$) and their entries $R_{ikl} = 1$ if flow i uses queue k at link l, and $R_{ikl} = 0$ otherwise. Note that to compress the expression of $\overline{\boldsymbol{R}}_{\boldsymbol{t}}$, we have renumbered the flows from 1 to N_t and the bottleneck links from 1 to L_t. In addition, once all the flows have been mapped to their corresponding queues, we can further renumber the non-empty queues at a link l from 1 to $|B_t(l)|$.

Define an $M_t \times 1$ equilibrium drop probability vector as:

$$\boldsymbol{P}^*_{\boldsymbol{t}} = \left[\boldsymbol{P}^{(\boldsymbol{1})}, \boldsymbol{P}^{(\boldsymbol{2})}, ..., \boldsymbol{P}^{(\boldsymbol{L_t})}\right]^T, \tag{10}$$

where $\boldsymbol{P}^{(\boldsymbol{l})} = \left[p^*_{1l,t}, p^*_{2l,t}, ..., p^*_{|B_t(l)|l,t}\right]$ (for $l = 1, 2, ..., L_t$).

Define an $N_t \times 1$ marginal utility vector as:

$$\boldsymbol{u}^*_{\boldsymbol{t}} = \left[\frac{1}{(x^*_{1,t})^\alpha}, \frac{1}{(x^*_{2,t})^\alpha}, ..., \frac{1}{(x^*_{N_t,t})^\alpha}\right]^T. \tag{11}$$

Then, (6) can be written in the following matrix form:

$$\overline{\boldsymbol{R}}_{\boldsymbol{t}} \boldsymbol{P}^*_{\boldsymbol{t}} = \boldsymbol{u}^*_{\boldsymbol{t}}. \tag{12}$$

An equilibrium point of the system (2)–(5) is determined by (6)–(7). One can investigate the existence and uniqueness of an equilibrium by analyzing the number of solutions to (6)–(7). Since (7) actually gives a solution for the equilibrium rates $x^*_{i,t}$, we only need to study the solution for the equilibrium drop probability vector $\boldsymbol{P}^*_{\boldsymbol{t}}$ in the matrix form (12).

First of all, we present a lemma about the property of $\overline{\boldsymbol{R}}_{\boldsymbol{t}}$.

Lemma 1. *The augmented routing matrix $\overline{\boldsymbol{R}}_{\boldsymbol{t}}$ has full rank.*

Proof: A detailed proof is given in Appendix A. ∎

Then, the following theorem provides a sufficient condition for a unique equilibrium point in the system.

Theorem 1. *If $N_t = M_t$, then there exists a unique equilibrium point in the TCP/SQM system* (2)–(5).

Proof: A detailed proof is given in Appendix B. ∎

Theorem 1 implies that if every TCP flow only traverses one bottleneck link (not necessarily the same one), then the TCP/SQM system has a unique equilibrium point. From Theorem 1, we can easily derive the following corollary.

Corollary 1. *If $L_t = 1$, then the corresponding single-bottleneck TCP/SQM system has a unique equilibrium point.*

Proof: Since $L_t = 1$ implies that $N_t = M_t$, the conclusion can be derived from Theorem 1. ∎

Theorem 2. *If $N_t < M_t$, then there are infinitely many equilibrium points in the TCP/SQM system* (2)–(5).

Proof: A detailed proof is given in Appendix C. ∎

Theorem 2 implies that the equilibrium point of the system is not unique if some flows traverse multiple bottleneck links. In particular, there exist infinitely many equilibrium drop probability vectors $\boldsymbol{P}^*_{\boldsymbol{t}}$ in this case.

B. Local Stability Analysis

After analyzing the equilibrium of the TCP/SQM system, we now conduct stability analysis for the system. Stability analysis can include both the stability of the entire TCP/SQM system (2)–(5), as well as the stability of each SQM subsystem at a bottleneck link. Due to space limitation, here we only present the stability analysis for each SQM subsystem.

Note that the system (2)–(5) is nonlinear. To study the stability of a nonlinear system, a prevalent approach (e.g., [26], [11], and [12]) is to perform linearization for the system and then analyze the local stability of the linearized system. We follow the approach used in [11], and present a local stability analysis for each single-bottleneck TCP/SQM subsystem with feedback delays. Note that comparing to the model in [11] which uses Reno as the TCP algorithm and considers a single-queue AQM scheme at each link, the proposed TCP/SQM model is more general.

First, we formulate a single-bottleneck TCP/SQM subsystem at link l based on the original system (2)–(5) as follows:

$$\dot{x}_{i,t}(\tau) = k_{i,t}\left(1 - x_{i,t}^{\alpha}(\tau)\left(p_{il,t}(\tau - D_{i,t}) + \sum_{\substack{j \in A_t(i) \\ j \neq l}} p_{ij,t}^{*}\right)\right), \quad (13)$$

$$\dot{q}_{il,t}(\tau) = \begin{cases} x_{i,t}(\tau) - C_{il,t}(\tau), & q_{il,t}(\tau) > 0, \\ \left[x_{i,t}(\tau) - C_{il,t}(\tau)\right]^{+}, & q_{il,t}(\tau) = 0, \end{cases} \quad (14)$$

where $i \in B_t(l)$, l is fixed for a specific link under consideration, $p_{ij,t}^{*}$ denotes the equilibrium drop probability of queue i at link $j \neq l$ that flow i traverses, $p_{il,t}(\tau - D_{i,t})$ and $C_{il,t}(\tau)$ are still described by (4) and (5) (with a fixed l), respectively, and the utility function (1) is used to derive (13).

The subsystem (13)–(14) only considers the evolution of the queue lengths (and corresponding drop probabilities) at link l and the evolution of the rates of the flows traversing link l. The drop probabilities at other links $j \neq l$ that a flow i traverses are fixed to the equilibrium values. This can be considered as an approximation for the original system [11].

Next, we linearize the subsystem (13)–(14) at the equilibrium point and obtain:

$$\delta\dot{x}_{i,t}(\tau) = -\frac{\alpha k_{i,t}}{x_{i,t}^{*}}\delta x_{i,t}(\tau) - \frac{k_{i,t}(x_{i,t}^{*})^{\alpha}}{\sigma_{il,t}}\delta q_{il,t}(\tau - D_{i,t}), \quad (15)$$

$$\delta\dot{q}_{il,t}(\tau) = \delta x_{i,t}(\tau) - \frac{1}{D_{i,t}}\delta q_{il,t}(\tau), \quad (16)$$

where $i \in B_t(l)$, l is fixed, $\delta x_{i,t}(\tau) = x_{i,t}(\tau) - x_{i,t}^{*}$, and $\delta q_{il,t}(\tau) = q_{il,t}(\tau) - q_{il,t}^{*}$ ($q_{il,t}^{*}$ is the equilibrium queue length).

For simplicity, we also renumber the flows and non-empty queues in a subsystem at link l such that $i = 1, 2, .., |B_t(l)|$. To express the linearized subsystem (15)–(16) in the matrix form, we further define some matrices as follows.

Define a $2|B_t(l)| \times 2|B_t(l)|$ matrix for link l as:

$$\boldsymbol{F_{l,t}} = \begin{bmatrix} \boldsymbol{F_1} & \boldsymbol{0} \\ \boldsymbol{I_1} & \boldsymbol{F_2} \end{bmatrix}, \quad (17)$$

where $\boldsymbol{F_1} = \mathrm{diag}\big(-\frac{\alpha k_{i,t}}{x_{i,t}^{*}}\big)$ is a $|B_t(l)| \times |B_t(l)|$ diagonal matrix with entries $\big(-\frac{\alpha k_{i,t}}{x_{i,t}^{*}}\big)$ in the main diagonal, $\boldsymbol{I_1}$ is a $|B_t(l)| \times |B_t(l)|$ identity matrix, $\boldsymbol{F_2} = \mathrm{diag}\big(-\frac{1}{D_{i,t}}\big)$ is a $|B_t(l)| \times |B_t(l)|$ diagonal matrix, and $\boldsymbol{0}$ is a $|B_t(l)| \times |B_t(l)|$ zero matrix.

Define a set of $2|B_t(l)| \times 2|B_t(l)|$ matrices for link l as:

$$\boldsymbol{E_{il,t}} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{E_{i1}} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}, \quad i = 1, 2, .., |B_t(l)|, \quad (18)$$

where $\boldsymbol{E_{i1}} = \mathrm{diag}\big((E_{i1})_j\big)$ is a $|B_t(l)| \times |B_t(l)|$ diagonal matrix with the following entries in the main diagonal

$$(E_{i1})_j = \begin{cases} -\frac{k_{i,t}(x_{i,t}^{*})^{\alpha}}{\sigma_{il,t}}, & \text{if } j = i, \\ 0, & \text{otherwise.} \end{cases}$$

Define a $2|B_t(l)| \times 1$ vector for link l as:

$$\boldsymbol{y_{l,t}}(\tau) = \left[\delta x_{1,t}(\tau), ..., \delta x_{|B_t(l)|,t}(\tau), \delta q_{1l,t}(\tau), ..., \delta q_{|B_t(l)|l,t}(\tau)\right]^{T}. \quad (19)$$

Then, the linearized subsystem (15)–(16) can be written in the following matrix form:

$$\dot{\boldsymbol{y}}_{\boldsymbol{l,t}}(\tau) = \boldsymbol{F_{l,t}}\boldsymbol{y_{l,t}}(\tau) + \sum_{i=1}^{|B_t(l)|} \boldsymbol{E_{il,t}}\boldsymbol{y_{l,t}}(\tau - D_{i,t}). \quad (20)$$

Next, we construct a Lyapunov function for the subsystem (20). To this end, we first present the following lemma about the property of the matrix $\boldsymbol{F_{l,t}}$.

Lemma 2. *$\boldsymbol{F_{l,t}}$ is a stable matrix, i.e., every eigenvalue of $\boldsymbol{F_{l,t}}$ has negative real part.*

Proof: A detailed proof is given in Appendix D. ∎

Since $\boldsymbol{F_{l,t}}$ is a stable matrix, there exists a unique positive definite matrix $\boldsymbol{A_{l,t}}$ that satisfies the following Lyapunov equation [27]:

$$\boldsymbol{F_{l,t}^{T}}\boldsymbol{A_{l,t}} + \boldsymbol{A_{l,t}}\boldsymbol{F_{l,t}} = -\boldsymbol{I} \quad (21)$$

where $\boldsymbol{I}$ is a $2|B_t(l)| \times 2|B_t(l)|$ identity matrix.

Using the matrix $\boldsymbol{A_{l,t}}$ in (21), we construct the following candidate Lyapunov function for the subsystem (20):

$$V(\boldsymbol{y_{l,t}}) = \boldsymbol{y_{l,t}^{T}}\boldsymbol{A_{l,t}}\boldsymbol{y_{l,t}}. \quad (22)$$

We now present a theorem that states a sufficient condition for the local asymptotic stability of the subsystem.

Theorem 3. *The subsystem* (20) *is locally asymptotically stable if*

$$\sum_{i=1}^{|B_t(l)|} \frac{k_{i,t}(x_{i,t}^{*})^{\alpha}}{\sigma_{il,t}} < \frac{1}{2\rho J_{l,t}\|\boldsymbol{A_{l,t}}\|} \quad (23)$$

where $\|\boldsymbol{A_{l,t}}\|$ denotes the 2-norm of $\boldsymbol{A_{l,t}}$, $J_{l,t} = \sqrt{\frac{\lambda_{max}(\boldsymbol{A_{l,t}})}{\lambda_{min}(\boldsymbol{A_{l,t}})}}$, $\lambda_{max}(\boldsymbol{A_{l,t}})$ and $\lambda_{min}(\boldsymbol{A_{l,t}})$ are the largest and smallest eigenvalues of $\boldsymbol{A_{l,t}}$, respectively, and $\rho > 1$ is a constant.

Proof: A detailed proof is given in Appendix E. ∎

Theorem 3 provides a guidance for choosing proper control parameters $\sigma_{il,t}$ for the queues of SQM, such that the SQM subsystem is locally asymptotically stable.

IV. SNUM Problem and SQM Algorithm

In this section, we first formulate a SNUM problem for the TCP/SQM system, and then propose our SQM algorithm.

A. Problem Formulation for SNUM

In the literature on congestion control, network utility maximization (NUM) [17] is frequently formulated to determine the optimal flow rates of a network. The classical NUM is a "static" optimization problem where all the network variables are assumed to have static values. In reality, network variables (such as the number of flows in a network) are usually time-varying, rendering the solution of the static NUM problem impractical for dynamic environments. To tackle this issue, we formulate a stochastic version of NUM, named SNUM, where the number of flows in the network is a random variable.

Following the formulation of the TCP/SQM model, the total time under consideration is composed of a number of periods indexed by t ($t = 1, 2, ..., T$). In every period t, we assume that the number of flows can reach a steady value denoted by N_t when the system is in equilibrium. Across different periods, the random variable N_t changes according to an unknown distribution. We further assume that the maximum number of flows that can be supported by the TCP/SQM system in all periods is denoted by K, and that the maximum number of bottleneck links in the system is denoted by L.

Let $\boldsymbol{x_t} = [x_{1,t}, x_{2,t}, ..., x_{K,t}]^T$ be the rate vector in period t. Note that $\boldsymbol{x_t}$ includes the equilibrium (or average) rates of all the K possible flows in period t. For non-existent flows j in period t, we let $x_{j,t} = \varepsilon$, where ε is a small positive constant such as one. We do not set the rate to zero in this case because we need to use the utility function (1) for all flows, which is undefined at zero. Thus, we have $\boldsymbol{x_t} \succeq \boldsymbol{\varepsilon}$. Let $\boldsymbol{C} = [C_1, C_2, ..., C_L]^T$ be the link capacity vector. Define a $K \times L$ routing matrix $\boldsymbol{R_t}$ in period t, where entries $R_{il} = 1$ if flow i uses link l, and $R_{il} = 0$ otherwise. Note that $\boldsymbol{R_t}$ is different from the augmented routing matrix $\overline{\boldsymbol{R}}_{\boldsymbol{t}}$ in (9).

Define a time-varying aggregate utility function U_t as:

$$U_t(\boldsymbol{x_t}) = \sum_{i=1}^{K} \widetilde{U}_{i,t}(x_{i,t}), \tag{24}$$

where

$$\widetilde{U}_{i,t}(x_{i,t}) = \begin{cases} U_{i,t}(x_{i,t}), & \text{if } i \in \mathcal{N}_t, \\ U_{i,t}(\varepsilon), & \text{otherwise.} \end{cases} \tag{25}$$

$U_{i,t}$ is defined in (1). $\mathcal{N}_t$ denotes the set of N_t existing flows in period t. For non-existent flows, they have constant utilities $U_{i,t}(\varepsilon)$, and we can ignore them by assigning zero coefficients.

The SNUM problem for the TCP/SQM system can be specified as follows:

$$\max_{\boldsymbol{x_t}} \sum_{t=1}^{T} U_t(\boldsymbol{x_t}) \tag{26}$$

$$\text{subject to} \quad \boldsymbol{R_t^T x_t} \preceq \boldsymbol{C}, \quad t = 1, 2, ..., T. \tag{27}$$

Note that the objective function (26) considers cumulative utilities from all periods, and this cumulative form of objective function is widely used in the online optimization literature, e.g., [16], [28], and [29]. Moreover, U_t is time-varying and depends on the random variable N_t, thereby introducing difficulties to apply static optimization techniques. Since U_t is the sum of some concave functions $U_{i,t}$ and the constraints (27) are linear, the SNUM problem (26)–(27) belongs to the OCO category. Thus, we can utilize OCO techniques to solve it.

Algorithm 1 Online Solver for SNUM

Input: T, initial rate vector $\boldsymbol{x_1} \in \mathcal{X}$, step size $\eta_t = \frac{D}{G\sqrt{t}}$.

1: **for** $t = 1, 2, ..., T$ **do**
2: Set the rate vector $\boldsymbol{x_t}$.
3: Observe N_t existing flows and compute $\nabla U_t(\boldsymbol{x_t})$.
4: Update: $\boldsymbol{y_{t+1}} = \boldsymbol{x_t} + \eta_t \nabla U_t(\boldsymbol{x_t})$.
5: Project: $\boldsymbol{x_{t+1}} = \Pi_{\mathcal{X}}(\boldsymbol{y_{t+1}})$.
6: **end for**

Let $\mathcal{X}$ be the feasible region, which is determined by (27) and $\boldsymbol{x_t} \succeq \boldsymbol{\varepsilon}$ (i.e., each $x_{i,t} \geq \varepsilon$). It is easy to see that $\mathcal{X}$ is a bounded convex set. Let D be an upper bound on the diameter of the feasible region $\mathcal{X}$. Since $\mathcal{X}$ is bounded, it satisfies that:

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}, \quad \|\boldsymbol{x} - \boldsymbol{y}\| \leq D. \tag{28}$$

Note that $U'_{i,t}(x_{i,t}) = \frac{1}{x_{i,t}^{\alpha}}$ and $x_{i,t}$ has non-zero and bounded value[1]. Therefore, the gradient of $U_t(\boldsymbol{x_t})$, denoted by $\nabla U_t(\boldsymbol{x_t})$, exists and is bounded. Let G be an upper bound on the norm of the gradient $\nabla U_t(\boldsymbol{x_t})$, i.e.,

$$\|\nabla U_t(\boldsymbol{x_t})\| \leq G, \quad \text{for any } \boldsymbol{x_t} \in \mathcal{X}. \tag{29}$$

Given specific network settings, it is easy to determine the values of D and G.

B. OGD-based Algorithm for Solving SNUM

Instead of using reinforcement learning methods [30], we now utilize OCO techniques to solve our SNUM problem. In the OCO framework [16], an online player sequentially makes decisions. At iteration t, the player has to choose a decision $\boldsymbol{x_t} \in \mathcal{X}$ before observing the corresponding function U_t. The performance of an algorithm for OCO is formally measured by the regret, which is defined as follows:

$$\text{Regret}(T) = \max_{\boldsymbol{x^*} \in \mathcal{X}} \sum_{t=1}^{T} U_t(\boldsymbol{x^*}) - \sum_{t=1}^{T} U_t(\boldsymbol{x_t}) \tag{30}$$

where $\boldsymbol{x^*} \in \arg\max_{\boldsymbol{x} \in \mathcal{X}} \sum_{t=1}^{T} U_t(\boldsymbol{x})$ is the best fixed decision in hindsight (i.e., with full knowledge of all U_t).

The goal of OCO is to develop an online learning algorithm such that the regret is sublinear with respect to T, i.e., $\text{Regret}(T) = o(T)$. This implies that the algorithm performs as well as the best fixed decision in hindsight on the average.

The online gradient descent (OGD) [31] is an OCO algorithm which achieves sublinear regret. Here we propose an OGD-based algorithm to solve the SNUM in Algorithm 1.

In Algorithm 1, a decision for the rate vector $\boldsymbol{x_t}$ is made at the beginning of period t. This is done before the corresponding $U_t(\boldsymbol{x_t})$ is available. After $\boldsymbol{x_t}$ is made, one can observe N_t

[1]For the TCP/SQM system, we can set $\varepsilon = 1$ which means that the rates of non-existent flows in period t are fixed to 1. The rates of existing flows are initialized at 1 and usually significantly larger than it as the system evolves. By doing so, the upper bound G in (29) does not become too large.

Algorithm 2 SQM Algorithm at Link l

Input: $\boldsymbol{x_{t+1}^{(l)}} \subseteq \boldsymbol{x_{t+1}}$ obtained in period t of Algorithm 1. At the end of period t, perform self-tuning:

1: Set the queue weights $Q_{il,t+1}$ at link l such that: $\frac{Q_{il,t+1}}{\sum_{j\in B_t(l)} Q_{jl,t+1}} C_l = x_{i,t+1}$ $(i \in B_t(l))$, e.g., $Q_{il,t+1} = x_{i,t+1}$.

2: Adjust the control parameters $\sigma_{il,t+1}$ at link l such that: $\sum_{i=1}^{|B_t(l)|} \frac{k_{i,t+1}(x_{i,t+1})^{\alpha}}{\sigma_{il,t+1}} < \frac{1}{2\rho J_{l,t+1}\|\boldsymbol{A_{l,t+1}}\|}$.

existing flows and compute the gradient $\nabla U_t(\boldsymbol{x_t})$ in period t. Then, the next decision $\boldsymbol{x_{t+1}}$ is derived by using the current gradient $\nabla U_t(\boldsymbol{x_t})$ and projecting it back onto the feasible set $\mathcal{X}$. The operator $\Pi_{\mathcal{X}}(\cdot)$ denotes the projection onto $\mathcal{X}$.

Next, we present a theorem about the regret of Algorithm 1.

Theorem 4. *Algorithm 1 with step size* $\eta_t = \frac{D}{G\sqrt{t}}$ $(t = 1, 2, ..., T)$ *yields* $O(\sqrt{T})$ *sublinear regret. Specifically,*

$$Regret(T) = \max_{\boldsymbol{x^*}\in\mathcal{X}} \sum_{t=1}^{T} U_t(\boldsymbol{x^*}) - \sum_{t=1}^{T} U_t(\boldsymbol{x_t}) \leq \frac{3}{2} GD\sqrt{T}. \quad (31)$$

Proof: A detailed proof is given in Appendix F. ∎

Remark 1: We make use of OGD to solve the SNUM since it is easy to determine its step size. There exist other advanced algorithms that can be utilized to solve the SNUM, such as the Primal-Dual algorithm in [28]. We do try it and find that it is challenging to select a proper step size for it.

C. Proposed SQM Algorithm

We now present our SQM algorithm based on the solution to the SNUM and the stability condition (23) in Algorithm 2.

The SQM algorithm in Algorithm 2 can be run iteratively at every bottleneck link l in a distributed manner. In contrast, Algorithm 1 is a centralized algorithm (which can be run by a controller or a designated router in the network) since it needs to collect the observed information about existing flows from all the bottleneck links (managed by the SQM algorithms).

During every update period t, the SQM algorithm at link l makes use of Algorithm 1 to obtain an interim decision for the rate vector in the SNUM. Thus, one can consider that the SQM algorithm at every link l can be invoked separately after Line 5 of Algorithm 1 for every period t. Specifically, $\boldsymbol{x_{t+1}}$ denotes the interim rate vector computed in period t of Algorithm 1. Let $\boldsymbol{x_{t+1}^{(l)}} \subseteq \boldsymbol{x_{t+1}}$ be a vector which only contains the rates (i.e., $x_{i,t+1}, i \in B_t(l)$) of the flows that traverse link l in period t. Then, at the end of period t, the queue weights $Q_{il,t+1}$ and the control parameters $\sigma_{il,t+1}$ can be self-tuned by SQM as follows. Since $\boldsymbol{x_{t+1}}$ is computed for the objective of maximizing the cumulative utilities via OCO, it can serve as a good indicator for the rate allocation. Thus, we update the queue weights based on $\boldsymbol{x_{t+1}}$ in Line 1 and aim to use the updated queue weights to drive the system towards an efficient equilibrium. To ensure the stability of the SQM subsystem at link l, we also update the control parameters $\sigma_{il,t+1}$ based on the stability condition (23) in Line 2. Specifically, we can construct the matrix $\boldsymbol{F_{l,t+1}}$ based on (17) with $B_{t+1}(l) = B_t(l)$, where the elements in (17) can be estimated or predetermined. We can then solve the corresponding Lyapunov equation (21) for $\boldsymbol{A_{l,t+1}}$ via MATLAB (note that MATLAB provides an interface to call it from C++). Alternatively, there are many efficient methods for solving Lyapunov equations in the literature. The eigenvalues of $\boldsymbol{A_{l,t+1}}$ can also be computed via MATLAB (or related efficient methods in the literature), so as to obtain $J_{l,t+1}$ and the 2-norm of $\boldsymbol{A_{l,t+1}}$. Note that by the end of period t, one can only observe the existing flows $i \in B_t(l)$. The updating for $Q_{il,t+1}$ can be considered as an action for setting $\boldsymbol{x_{t+1}}$ in the next period $(t + 1)$ in Algorithm 1.

Remark 2: Although Algorithm 1 is a centralized algorithm (since it needs to solve the SNUM problem globally), it is simple and easy to implement in a single autonomous system (AS). We leave the decentralization of it as our future work, which usually involves some message exchanges among distributed SQM routers. By passing a subset of $\boldsymbol{x_{t+1}}$ to every bottleneck link, the SQM algorithm (Algorithm 2) at every link can be run locally and separately.

V. Performance Evaluation

This section presents our simulation results obtained from MATLAB and *ns* [32].

The simulation settings are as follows. The network topology of our simulations is shown in Fig. 2, where the square nodes represent routers. To save space, we do not show hosts but they are indeed connected to the routers. In particular, the link connecting routers $R2$ and $R3$ denoted by l_{23} and the link connecting $R3$ and $R4$ denoted by l_{34} are two bottleneck links, and they have the same capacity. $R2$, $R3$, and $R4$ can deploy AQM algorithms such as CoDel [4], PIE [5], SFQ-CoDel [9], and our proposed SQM in the simulations. There are a number of TCP flows in the network, traversing from $R1$ to $R4$, from $R5$ to $R6$, or from $R6$ to $R4$. The long-lived TCP flows with different round-trip times (RTTs) are generated by FTP applications, and they can use the popular loss-based TCP algorithms: NewReno [33] or CUBIC [34].

A. Numerical Results via MATLAB

First, we verify our theoretical analysis by conducting MATLAB simulations. For simplicity, let us focus on a single-bottleneck case in this subsection, that is, the flows from $R6$ to $R4$ do not exist such that l_{34} is not a bottleneck link. We use the following settings to study the equilibrium and stability of the TCP/SQM system (2)–(5) (the period index t is removed here). There are ten TCP flows (with rates $x_i(\tau)$) traversing the bottleneck link l_{23} with a capacity $C_{l_{23}} = 1000$ packets/s. The RTTs of the flows (i.e., D_i) range from 0.100 s to 0.150 s. Let $\alpha = 2$, which means that the flows are Reno or NewReno flows. Thus, we can set the constants $k_i = \frac{1}{D_i^2}$. Let the weights of the ten corresponding queues be $Q_1 = Q_2 = Q_3 = Q_4 = Q_5 = 1500$ and $Q_6 = Q_7 = Q_8 = Q_9 = Q_{10} = 1000$. Then, we can use the *dde23* solver [35] to solve the DDEs (2)–(5) with these specific settings.

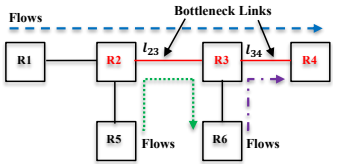


Fig. 2: Simulation topology.

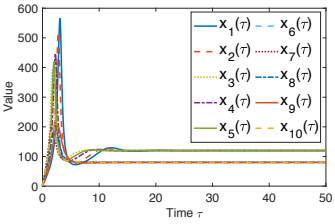


Fig. 3: Results via MATLAB.

Fig. 3 plots the solution for the rates $x_i(\tau)$ evaluated in the interval [0, 50]. It can be seen that as time τ increases, $x_i(\tau)$ can converge to the equilibrium values $x_1 = x_2 = x_3 = x_4 = x_5 = 120$ and $x_6 = x_7 = x_8 = x_9 = x_{10} = 80$. These equilibrium rates are consistent with the above assignment of queue weights. This shows that one can drive the TCP/SQM system towards a desirable equilibrium by setting proper queue weights.

B. Packet-Level (ns) Simulation Results

We then conduct packet-level simulations using *ns* [32] to compare our SQM with the popular CoDel, PIE, and SFQ-CoDel in terms of queueing delay and average throughput.

Using the topology in Fig. 2 with two bottleneck links l_{23} and l_{34}, we now consider Scenario 1, where the maximum number of TCP flows in the network is $K = 40$. Initially, there are five flows traversing from $R5$ to $R6$ and another five flows going from $R6$ to $R4$. Subsequently, new flows randomly arrive and traverse from $R1$ to $R4$, and they will end with different finishing times. The number of newly arriving flows in period t is randomly drawn from the range [0, 5]. All flows use the NewReno algorithm. The RTTs of the flows range from 0.080 s to 0.180 s. Let the link capacities be $C_{l_{23}} = C_{l_{34}} = 4166$ packets/s (i.e., 50 Mbps). CoDel, PIE, and SFQ-CoDel use the default settings, and PIE has an auto-tuning mechanism for its control parameters [5]. The target queueing delay is set to 5 ms for all algorithms. A period t in SQM lasts for 5 seconds by default. This can be reset whenever needed.

Due to space limitation, here we only present the results for the queueing delay at the more congested link l_{23} (compared to l_{34}). Specifically, Fig. 4 illustrates the evolution of the queueing delay produced by the AQM algorithms at l_{23}. It can be seen that CoDel and PIE exhibit a number of notable spikes on the queueing delay when the number of flows changes randomly over time. SFQ-CoDel suffers more and shows highly fluctuating queueing delay with a larger mean value. In contrast, our SQM can consistently stabilize the queueing delay by properly tuning its control parameters.

Next, we present Scenario 2, which aims to investigate the throughput of flows and the fairness among them, given that a specific AQM is used. Here we are interested in the case where flows can use different TCP algorithms. Specifically, five NewReno flows from $R5$ to $R6$ and another five NewReno flows from $R6$ to $R4$ start at the beginning of the simulation. Then, two CUBIC flows from $R1$ to $R4$ start at 30 s, and three more CUBIC flows join at 55 s. We investigate the rate allocation of these flows regulated by the AQM algorithms.

Fig. 5 shows the average throughput of the NewReno flows (from $R5$ to $R6$) and the CUBIC flows (from $R1$ to $R4$) when using different AQM algorithms during [10 s, 80 s]. One can observe that the CUBIC flows aggressively acquire much higher throughput while substantially lowering the throughput of the NewReno flows, if CoDel or PIE is used as the AQM algorithm. The reason is that CoDel and PIE employ a single-queue structure that cannot differentiate flows, which may lead to unfairness. In contrast, SQM and SFQ-CoDel can provide fair rate allocation for different flows. Note that unlike SFQ-CoDel which uses the same static weight for all queues, our SQM is more flexible and can self-tune different queue weights. It is also applicable when one aims to provide differentiated services for flows or minimize the flow completion time (FCT) using a different form of utility function [36].

Here, the packet-level simulation results demonstrate that our SQM not only stabilizes the queueing delay, but also improves fairness among flows.

VI. Conclusion

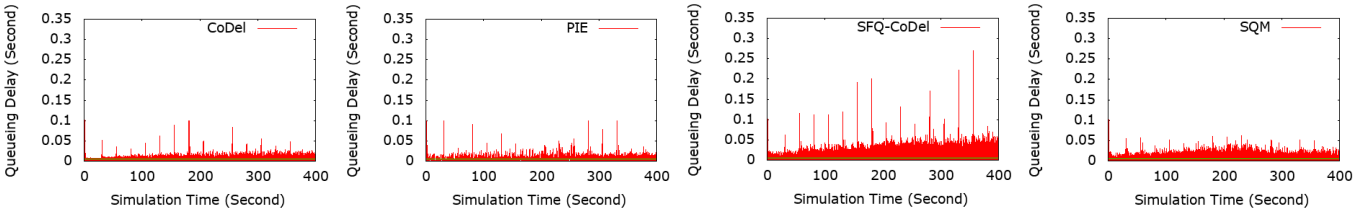
In this paper, we have proposed a general framework for a self-tuning queue management (SQM) scheme, which is adaptive to the stochastic network environments and provides fair congestion control among flows. We first presented a general architecture of SQM with adaptive fair queueing and proposed a fluid model to analyze a general TCP/SQM system. We then conducted equilibrium and stability analysis for the general TCP/SQM system by showing the existence and uniqueness of an equilibrium point in the system, and deriving sufficient conditions for the local asymptotic stability of each SQM subsystem with feedback delays. To adapt to the stochastic environments, we further formulated a SNUM problem for the TCP/SQM system and utilized OCO techniques to tackle it. Finally, we developed a distributed SQM algorithm which can self-tune different queue weights and control parameters. By numerical and packet-level simulations, we not only verified our theoretical analysis, but also demonstrated that our SQM algorithm can significantly improve queueing delay and fairness among flows, compared to existing AQM algorithms.

Appendix A
Proof of Lemma 1

Proof: From the definition of $\overline{\boldsymbol{R}}_{\boldsymbol{t}}$ in (9), one can know that there must be at least one entry with the value of 1 in every row since every flow i ($i = 1, 2, ..., N_t$) is mapped to at least one queue in the system. Moreover, for every flow i, the columns of its non-zero entries must be different from that of all other flows' non-zero entries. The reason is that different flows are mapped to different queues even if they traverse the same link and every queue contains one flow. Thus, we can conclude that the N_t rows of $\overline{\boldsymbol{R}}_{\boldsymbol{t}}$ are linearly independent. That is, $\overline{\boldsymbol{R}}_{\boldsymbol{t}}$ has full rank and $\text{rank}(\overline{\boldsymbol{R}}_{\boldsymbol{t}}) = N_t$ since $N_t \leq M_t$. ■

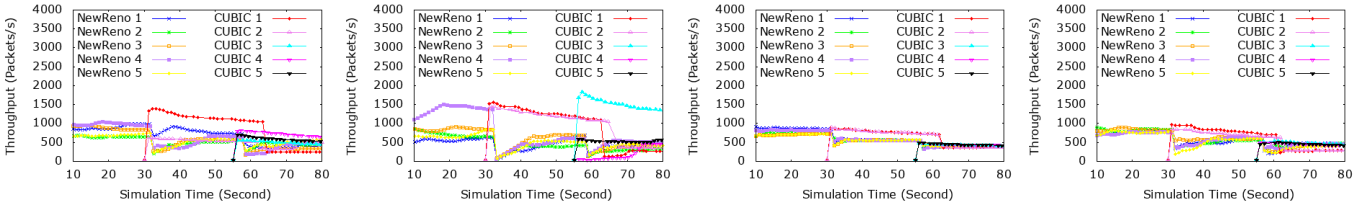
Appendix B
Proof of Theorem 1

Proof: If $N_t = M_t$, $\overline{\boldsymbol{R}}_{\boldsymbol{t}}$ is a square matrix with full rank and thus has an inverse $\overline{\boldsymbol{R}}_{\boldsymbol{t}}^{-1}$. Then, we can directly solve (12)



(a) CoDel. (b) PIE. (c) SFQ-CoDel. (d) SQM.

Fig. 4: Scenario 1: queueing delay at the bottleneck link l_{23} in a stochastic environment.



(a) Using CoDel. (b) Using PIE. (c) Using SFQ-CoDel. (d) Using SQM.

Fig. 5: Scenario 2: average throughput of NewReno and CUBIC flows.

for $\boldsymbol{P}_t^*$ and obtain $\boldsymbol{P}_t^* = \overline{\boldsymbol{R}}_t^{-1}\boldsymbol{u}_t^*$, which is uniquely determined since $\boldsymbol{u}_t^*$ is already fixed. On the other hand, $N_t = M_t$ indicates that every flow only traverses one bottleneck link. Thus, (7) gives a unique solution for the equilibrium rates $x_{i,t}^*$ as $C_{il,t}$ is uniquely determined by (8). As a result, there exists a unique equilibrium point in the TCP/SQM system when $N_t = M_t$. ■

Appendix C
Proof of Theorem 2

Proof: Applying the Rouché-Capelli Theorem [37], (12) has a solution if and only if $\text{rank}(\overline{\boldsymbol{R}}_t) = \text{rank}(\overline{\boldsymbol{R}}_t|\boldsymbol{u}_t^*)$, where $[\overline{\boldsymbol{R}}_t|\boldsymbol{u}_t^*]$ denotes the augmented matrix of (12). Since $[\overline{\boldsymbol{R}}_t|\boldsymbol{u}_t^*]$ is an $N_t \times (M_t+1)$ matrix and $N_t < (M_t+1)$, we still have $\text{rank}(\overline{\boldsymbol{R}}_t|\boldsymbol{u}_t^*) = N_t$. Thus, we obtain $\text{rank}(\overline{\boldsymbol{R}}_t) = \text{rank}(\overline{\boldsymbol{R}}_t|\boldsymbol{u}_t^*) = N_t$, which implies that (12) has a solution. However, since the rank of $\overline{\boldsymbol{R}}_t$ is less than the number of variables (i.e., $\text{rank}(\overline{\boldsymbol{R}}_t) = N_t < M_t$), (12) actually has infinitely many solutions for $\boldsymbol{P}_t^*$ according to the Rouché-Capelli Theorem. Therefore, we can conclude that there are infinitely many equilibrium points in the TCP/SQM system when $N_t < M_t$. ■

Appendix D
Proof of Lemma 2

Proof: Let λ be an eigenvalue of $\boldsymbol{F}_{\boldsymbol{l},t}$. By definition, λ satisfies the equation: $\left|\boldsymbol{F}_{\boldsymbol{l},t} - \lambda\boldsymbol{I}\right| = 0$, where $\boldsymbol{I}$ is a $2|B_t(l)| \times 2|B_t(l)|$ identity matrix. We can iteratively expand $\left|\boldsymbol{F}_{\boldsymbol{l},t} - \lambda\boldsymbol{I}\right|$ along the last column using the Laplace expansion as follows:

$$\left|\boldsymbol{F}_{\boldsymbol{l},t} - \lambda\boldsymbol{I}\right| = \prod_{i=1}^{|B_t(l)|}\left(-\frac{1}{D_{i,t}} - \lambda\right)\cdot\left|\text{diag}\left(-\frac{\alpha k_{i,t}}{x_{i,t}^*} - \lambda\right)\right| = 0. \tag{32}$$

Therefore, we obtain $\lambda_i = -\frac{\alpha k_{i,t}}{x_{i,t}^*}$ and $\lambda_{|B_t(l)|+i} = -\frac{1}{D_{i,t}}$ for $i = 1, ..., |B_t(l)|$. Since every eigenvalue of $\boldsymbol{F}_{\boldsymbol{l},t}$ has negative real part, $\boldsymbol{F}_{\boldsymbol{l},t}$ is a stable matrix. ■

Appendix E
Proof of Theorem 3

Proof: Here we apply similar proof techniques used in [11]. By definition, we just need to show that (22) is indeed a Lyapunov function, i.e., $\dot{V}(\boldsymbol{y}_{\boldsymbol{l},t}) < 0$ if (23) is satisfied.

Let $D_{max} = \max\{D_{1,t}, D_{2,t} ..., D_{|B_t(l)|,t}\}$. Applying the following Razumikhin condition [38] with some constant $\rho > 1$:

$$V(\boldsymbol{y}_{\boldsymbol{l},t}(\xi)) \le \rho^2 V(\boldsymbol{y}_{\boldsymbol{l},t}(\tau)), \quad \text{for } \tau - D_{max} \le \xi \le \tau, \tag{33}$$

we can obtain:

$$\lambda_{min}(\boldsymbol{A}_{\boldsymbol{l},t})\|\boldsymbol{y}_{\boldsymbol{l},t}(\xi)\|^2 \le \rho^2\lambda_{max}(\boldsymbol{A}_{\boldsymbol{l},t})\|\boldsymbol{y}_{\boldsymbol{l},t}(\tau)\|^2. \tag{34}$$

Thus,

$$\|\boldsymbol{y}_{\boldsymbol{l},t}(\xi)\| \le \rho J_{l,t}\|\boldsymbol{y}_{\boldsymbol{l},t}(\tau)\|. \tag{35}$$

From (18), we have:

$$\|\boldsymbol{E}_{\boldsymbol{il},t}\| = \frac{k_{i,t}(x_{i,t}^*)^\alpha}{\sigma_{il,t}}. \tag{36}$$

Using (21), (35), and (36), we can derive:

$$\begin{aligned}\dot{V}(\boldsymbol{y}_{\boldsymbol{l},t}) &= -\boldsymbol{y}_{\boldsymbol{l},t}^T(\tau)\boldsymbol{I}\,\boldsymbol{y}_{\boldsymbol{l},t}(\tau) + 2\sum_{i=1}^{|B_t(l)|}\boldsymbol{y}_{\boldsymbol{l},t}^T(\tau - D_{i,t})\boldsymbol{E}_{\boldsymbol{il},t}^T\boldsymbol{A}_{\boldsymbol{l},t}\boldsymbol{y}_{\boldsymbol{l},t}(\tau)\\ &\le -\|\boldsymbol{y}_{\boldsymbol{l},t}(\tau)\|^2 + 2\rho J_{l,t}\sum_{i=1}^{|B_t(l)|}\|\boldsymbol{A}_{\boldsymbol{l},t}\|\|\boldsymbol{E}_{\boldsymbol{il},t}\|\|\boldsymbol{y}_{\boldsymbol{l},t}(\tau)\|^2\\ &= -\left(1 - 2\rho J_{l,t}\|\boldsymbol{A}_{\boldsymbol{l},t}\|\sum_{i=1}^{|B_t(l)|}\frac{k_{i,t}(x_{i,t}^*)^\alpha}{\sigma_{il,t}}\right)\|\boldsymbol{y}_{\boldsymbol{l},t}(\tau)\|^2.\end{aligned} \tag{37}$$

One can now observe that if (23) is satisfied, $\dot{V}(\boldsymbol{y}_{\boldsymbol{l},t}) < 0$ when $\boldsymbol{y}_{\boldsymbol{l},t}(\tau) \neq 0$. Thus, (22) is indeed a Lyapunov function, such that the subsystem is locally asymptotically stable. ■

APPENDIX F
PROOF OF THEOREM 4

Proof: For simplicity, define $\nabla_t \triangleq \nabla U_t(\boldsymbol{x_t})$. By the concavity of U_t, we have:

$$U_t(\boldsymbol{x^*}) - U_t(\boldsymbol{x_t}) \leq \nabla_t^T(\boldsymbol{x^*} - \boldsymbol{x_t}). \tag{38}$$

Using the Pythagorean theorem about the property of the projection operator [16], we can derive an upper bound for $\nabla_t^T(\boldsymbol{x^*} - \boldsymbol{x_t})$ as follows:

$$\|\boldsymbol{x_{t+1}} - \boldsymbol{x^*}\|^2 = \left\|\Pi_{\mathcal{X}}(\boldsymbol{x_t} + \eta_t \nabla_t) - \boldsymbol{x^*}\right\|^2 \leq \|\boldsymbol{x_t} + \eta_t \nabla_t - \boldsymbol{x^*}\|^2. \tag{39}$$

Expanding the right-hand side of (39) gives:

$$\|\boldsymbol{x_{t+1}} - \boldsymbol{x^*}\|^2 \leq \|\boldsymbol{x_t} - \boldsymbol{x^*}\|^2 + \eta_t^2\|\nabla_t\|^2 + 2\eta_t\nabla_t^T(\boldsymbol{x_t} - \boldsymbol{x^*}). \tag{40}$$

Using (29), we then derive:

$$2\nabla_t^T(\boldsymbol{x^*} - \boldsymbol{x_t}) \leq \frac{\|\boldsymbol{x_t} - \boldsymbol{x^*}\|^2 - \|\boldsymbol{x_{t+1}} - \boldsymbol{x^*}\|^2}{\eta_t} + \eta_t G^2. \tag{41}$$

Let $\eta_t = \frac{D}{G\sqrt{t}}$ (with $\frac{1}{\eta_0} \triangleq 0$) and summing (38) and (41) from $t = 1$ to T, we can derive:

$$\begin{aligned} 2\,\text{Regret}(T) &\leq \sum_{t=1}^{T} \frac{\|\boldsymbol{x_t} - \boldsymbol{x^*}\|^2 - \|\boldsymbol{x_{t+1}} - \boldsymbol{x^*}\|^2}{\eta_t} + G^2\sum_{t=1}^{T}\eta_t \\ &\leq \sum_{t=1}^{T} \|\boldsymbol{x_t} - \boldsymbol{x^*}\|^2\Big(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\Big) + G^2\sum_{t=1}^{T}\eta_t \\ &\leq D^2\frac{1}{\eta_T} + G^2\sum_{t=1}^{T}\eta_t \leq 3GD\sqrt{T}. \end{aligned}$$

Therefore, $\text{Regret}(T) \leq \frac{3}{2}GD\sqrt{T}$. ∎

REFERENCES

[1] S. H. Low, "A duality model of TCP and queue management algorithms," *IEEE/ACM Trans. Networking*, Vol. 11, No. 4, pp. 525-536, Aug. 2003.

[2] J. Gettys and K. Nichols, "Bufferbloat: Dark buffers in the Internet," *Communications of the ACM*, Vol. 55, No. 1, pp. 57-65, Jan. 2012.

[3] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, Vol. 1, No. 4, pp. 397-413, Aug. 1993.

[4] K. Nichols and V. Jacobson, "Controlling queue delay," *ACM Queue*, Vol. 10, No. 5, pp. 1-15, May 2012.

[5] R. Pan, P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg, "PIE: A lightweight control scheme to address the bufferbloat problem," *Proc. of IEEE HPSR*, pp. 148-155, Jul. 2013.

[6] A. Tang, X. Wei, S. H. Low, and M. Chiang, "Equilibrium of heterogeneous congestion control: Optimality and stability," *IEEE/ACM Trans. Networking*, Vol. 18, No. 3, pp. 844-857, Jun. 2010.

[7] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: Motivation, architecture, algorithms, performance," *IEEE/ACM Trans. Networking*, Vol. 14, No. 6, pp. 1246-1259, Dec. 2006.

[8] V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM Computer Comm. Review*, Vol. 18, No. 4, pp. 314-329, Aug. 1988.

[9] K. Nichols, sfqCodel - The Controlled-Delay Active Queue Management algorithm with stochastic binning. [Online]. Available: http://www.pollere.net/Txtdocs/sfqcodel.cc

[10] J. Ye and K.-C. Leung, "Adaptive and stable delay control for combating bufferbloat: Theory and algorithms," *IEEE Systems Journal*, Vol. 14, No. 1, pp. 1285-1296, Mar. 2020.

[11] J. Ye, K.-C. Leung, and S. H. Low, "Combating bufferbloat in multi-bottleneck networks: Theory and algorithms," *IEEE/ACM Trans. Networking*, Vol. 29, No. 4, pp. 1477-1493, Aug. 2021.

[12] L. Wang, L. Cai, X. Liu, X. Shen, and J. Zhang, "Stability analysis of multiple-bottleneck networks," *Computer Networks*, Vol. 53, No. 3, pp. 338-352, Feb. 2009.

[13] F. Baker and G. Fairhurst, "IETF recommendations regarding active queue management," *RFC 7567, IETF*, Jul. 2015.

[14] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *ACM SIGCOMM Computer Comm. Review*, Vol. 19, No. 4, pp. 1-14, Aug. 1989.

[15] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin," *ACM SIGCOMM Computer Comm. Review*, Vol. 25, No. 4, pp. 231-242, Oct. 1995.

[16] E. Hazan, "Introduction to online convex optimization," *Found. Trends Optimization*, Vol. 2, No. 3-4, pp. 157-325, 2016.

[17] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, Vol. 49, No. 3, pp. 237-252, Mar. 1998.

[18] Y. Lu, M. Wang, B. Prabhakar, and F. Bonomi, "ElephantTrap: A low cost device for identifying large flows," *Proc. of IEEE HOTI*, Sep. 2007.

[19] T. Yang, H. Zhang, J. Li, J. Gong, S. Uhlig, S. Chen, and X. Li, "HeavyKeeper: An accurate algorithm for finding top-k elephant flows," *IEEE/ACM Trans. Networking*, Vol. 27, No. 5, pp. 1845-1858, Oct. 2019.

[20] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Trans. Networking*, Vol. 8, No. 5, pp. 556-567, Oct. 2000.

[21] T. Bonald and L. Massoulié, "Impact of fairness on Internet performance," *Proc. of ACM SIGMETRICS*, Jun. 2001.

[22] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," *IEEE Jour. Selected Areas in Commun.*, Vol. 13, No. 8, pp. 1465-1480, Oct. 1995.

[23] T. Kelly, "Scalable TCP: improving performance in highspeed wide area networks," *ACM SIGCOMM Computer Comm. Review*, Vol. 33, No. 2, pp. 83-91, Apr. 2003.

[24] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proc. of the IEEE*, Vol. 95, No. 1, pp. 255-312, Jan. 2007.

[25] C. Lai, S. H. Low, K.-C. Leung, and V. O. K. Li, "Pricing link by time," *ACM SIGMETRICS Performance Eval. Review*, Vol. 42, No. 1, pp. 421-433, Jun. 2014.

[26] C. V. Hollot, V. Misra, D. Towsley, and W. Gong, "Analysis and design of controllers for AQM routers supporting TCP flows," *IEEE Trans. Automatic Control*, Vol. 47, No. 6, pp. 945-959, Jun. 2002.

[27] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2002.

[28] X. Cao and K. J. R. Liu, "Online convex optimization with time-varying constraints and bandit feedback," *IEEE Trans. Automatic Control*, Vol. 64, No. 7, pp. 2665-2680, Jul. 2019.

[29] S. Shahrampour and A. Jadbabaie, "Distributed online optimization in dynamic environments using mirror descent," *IEEE Trans. Automatic Control*, Vol. 63, No. 3, pp. 714-725, Mar. 2018.

[30] J. Wang, Y. Liu, S. Niu, and H. Song, "Reinforcement learning optimized throughput for 5G enhanced swarm UAS networking," *Proc. of IEEE ICC*, Jun. 2021.

[31] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," *Proc. of ICML*, pp. 928-936, Aug. 2003.

[32] K. Fall and K. Varadhan, "The *ns* manual (formerly *ns* notes and documentation)," *The VINT Project*, Nov. 2011.

[33] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida, "The NewReno modification to TCP's fast recovery algorithm," *RFC 6582, IETF*, Apr. 2012.

[34] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *ACM SIGOPS Operating Sys. Review*, Vol. 42, No. 5, pp. 64-74, Jul. 2008.

[35] L.F. Shampine and S. Thompson, "Solving DDEs in MATLAB," *Applied Numerical Mathematics*, Vol. 37, No. 4, pp. 441–458, Jun. 2001.

[36] K. Nagaraj, D. Bharadia, H. Mao, S. Chinchali, M. Alizadeh, and S. Katti, "NUMFabric: Fast and flexible bandwidth allocation in datacenters," *Proc. of ACM SIGCOMM*, Aug. 2016.

[37] I. R. Shafarevich and A. O. Remizov, *Linear Algebra and Geometry*. Berlin/Heidelberg, Germany: Springer-Verlag, 2013.

[38] K. Gu, V. L. Kharitonov, and J. Chen, *Stability of Time-Delay Systems*. Boston, MA: Birkhäuser, 2003.