Congestion Control with Deterministic Service Delay Guarantee

Xinglin Yang[†], Wei Wang[†*], Jiangping Han[‡], Kaiping Xue[‡], Zhaoyang Zhang[†]
[†]Zhejiang Provincial Key Laboratory of Information Processing, Communication and Networking,
College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China
[‡]School of Cyber Science and Technology, University of Science and Technology of China, Hefei 230027, China
[*]Email: wangw@zju.edu.cn

***Abstract*—To meet the ever-increasing demand for mission-critical applications, the deterministic service delay guarantee from the application prospective has become an important metric for congestion control in end-to-end communications. In this paper, we propose a two-timescale congestion window control (TCWC) mechanism with delay-aware priority based on TCP Vegas. Different from the existing works only considering the network delay, we formulate the network utility maximization problem for congestion control by adding additional flow queuing delay at the source node to guarantee the deterministic service delay constraints. By designing a virtual queue, we transform the delay constraint to the time-averaged queue stability, and solve it in each time slot according to the Lyapunov drift-plus-penalty method. Then we obtain the adjustment strategy of congestion window according to Lagrangian duality theory. For further guarantee the service delay, we apply extreme value theory (EVT) to evaluate the priorities of different flows, which determines the update rate of congestion window. Finally, simulation results show that our algorithm can significantly reduce the average service delay and achieve better delay guarantee compared to traditional TCP Vegas.**

I. Introduction

With the rapid development of the Internet of Things (IoT) and the rise of various new applications, such as augmented/virtual reality (AR/VR), autonomous vehicles, and intelligent manufacturing, etc., deterministic low-latency service has become one of the most important requirements of user equipment [1], [2].

The transmission control protocol (TCP) is a widely used protocol in the network, which can achieve reliable end-to-end transmission. Congestion control algorithms are the main algorithms used in TCP to control sending rate for avoiding congestion in the network. Based on the packet loss or delay signal, TCP congestion control mechanism can increase or decrease the congestion window to limit the number of unacknowledged data segments allowed in the network. The goal is to minimize congestion while maximizing utilization of all flows in the network. On the contrary, user datagram protocol (UDP) provides a simple connectionless solution without the guarantee of reliable data transmission. UDP does not use congestion control, so it will cause a lot of packet loss and reduce performance when network congestion occurs. For provide low delay and high reliability, quick UDP internet connection (QUIC) is developed by Google [3], [4], in which many mechanisms, including congestion control, are inspired by TCP to provide flexible and reliable data transfer. QUIC improves congestion control by using faster connection establishment, multiplexing, forward error correction, etc. Especially, encrypted QUIC connections can be established within 0 round-trip time (RTT).

This work is supported in part by National Natural Science Foundation of China (No. U20A20158), National Key R&D Program of China (No. 2018YFB1801104), Ningbo S&T Major Project (No. 2019B10079), and Key R&D Program of Zhejiang (No. 2022C03078).

Most of the existing congestion control algorithms focus on utilizing the network capability efficiently with fairness. For deterministic low delay services, rather than fairness, different priorities should be provided for guaranteeing the performance on the service delay by congestion control. For application services, the delay is considered since the application data are generated. Thus, the end-to-end delay from the application layer perspective, referred to as *the service delay*, includes not only the delay in the network but also the delay at source nodes before entering the network since the data are generated in the application layer. Different flows adjust their congestion control strategies distributively according to their source queue lengths and delay constraints. It is quite challenging since there is an inherent conflict between the service delay guarantee and the network congestion avoidance.

In this paper, to address the congestion control problem with deterministic service delay guarantee, we propose a two-timescale congestion window control (TCWC) mechanism with delay-aware priority based on TCP Vegas. Specifically, we construct the virtual queue to equivalently replace the deterministic service delay constraint. In this way, we transform the original problem to a time-averaged queue stability optimization problem, and solve it by optimizing the drift-plus-penalty in each time slot based on the Lyapunov framework. Then we derive the approximate optimal solution and give the update strategy of the congestion window according to the Lagrangian dual method. To further guarantee the service delay, we use the extreme value theory (EVT) to analyze the tail distribution of the real-time service delay samples. For the flows that violate the deterministic constraints, higher priority is provided by increasing the step size of congestion window adjustment.

The remainder of the paper is structured as follows. In Section II, we summarize related works on congestion control. The system model is presented and the network utility maximization problem for congestion control is formulated in Section III. In Section IV, we propose a two-timescale congestion control mechanism with deterministic service delay guarantee. The proposed algorithm is evaluated by simulation in Section V. Finally, we conclude the paper in Section VI.

II. Related Works

Due to the continuous change of Internet application requirements, these are many researches have enhanced the traditional TCP congestion control for low delay in both academic and industrial. In [5], the authors proposed a new delay-based congestion control algorithm that supported low queuing delay and high network utilization. It is especially useful for traffic that mixes delay-sensitive and bandwidth-demanding applications. An new end-to-end congestion control protocol was introduced to achieve high throughput, low delay and fair rate allocation in [6]. It modeled the bottleneck queue as a Markov chains explicitly and adjusted aggressiveness to compete fairly with loss-based protocols. At the same time, the sending rate was determined by using the delay profile. In [7], another delay-based congestion control scheme was presented and designed to work with multi-packet segments offloading for high performance. Instead of building the queue to a fixed RTT threshold, it adjusted the congestion window through using the gradient of the RTT to predict the onset of congestion.

Besides, there are a few deadline-aware congestion control algorithms in recent years. In [8], a deadline-award data center TCP was proposed for bandwidth allocation based on a distributed and reactive approach. It controlled the delay and improve user experience under soft real-time constraints. A software defined network based explicit deadline-aware TCP was proposed for cloud data center networks in [9]. The non-deadline flows were assigned with a base rate, while the deadline flows were allocated spare bandwidth as much as possible. The packet-loss timeout problem was solved by introducing a retransmission-enhanced algorithm. In [10], the authors proposed a deadline-awareness congestion control mechanism by parameterizing the traditional TCP New Reno congestion control strategy. The modulation of the congestion window was dynamically adjusted to minimize deadline-missing flows according to delivery requirements.

Recently, the Lyapunov optimization theory has been widely used in congestion control analysis, mainly in the network stability analysis. In [11], the authors addressed the redesigning of congestion control for TCP applications on networks with coupled wireless links. Based on the Lyapunov method, they rigorously established the global stability of the proposed QUIC-TCP to achieve optimal equilibrium in the network fluid model. A novel stochastic optimal scheduler was proposed for multipath TCP in software defined wireless network [12]. The control decisions can be made through Lyapunov optimization technique to maximize the throughput while minimizing the cost for users. In [13], a model that trades off energy efficiency and throughput was established by using the Lyapunov method. The split ratio of each link was obtained to directly control the change of the congestion window and queue stability ensured that all data left the buffer in a finite time.

In addition, the delay sensitivity of applications has also been extensively studied in ultra-reliable low-latency communication (URLLC). In [14], a two-timescale task offloading and resource allocation mechanism was proposed in a mobile edge computing (MEC) network. The deterministic and statistical constraints can be equivalently solved by introducing virtual queue and extreme value theory. In [15], [16], the multi-destination MEC was studied to fulfill the delay constraint and minimize the energy consumption. In [17], a jointly content caching and cooperative transmission algorithm was designed to minimize the average delay, where the problem is decomposed into a short time-scale transmission and a long time-scale caching. The random access problem for massive devices with heterogeneous capabilities was studied in [18], where a two-dimensional Markov decision process was adopted to obtain an optimal policy to minimize the delay. In [19], a joint power and resource allocation problem for URLLC was studied by estimating and characterize the tail distribution of the queue lengths using extreme value theory (EVT).

In summary, all the above existing works considered only the average-based delay constraints and ignored flows queuing delay at source nodes in the congestion control design. Different to existing works, for deterministic low delay services, we consider additional *deterministic constraints* of the service delay to gradually adjust the congestion window. In addition, the Lyapunov method is adopted to guarantee not only the network queuing stability but also the deterministic service delay.

III. System Model

We consider a wireless network with a set $\mathcal{N}$ of N flows sharing a set $\mathcal{L}$ of L communication links with finite capacities $\boldsymbol{c} = (c_l, l \in \mathcal{L})$ in packets per second, where flow i consists of a source node s_i and a destination node d_i, i.e., the node pair $(s_i, d_i), i \in \mathcal{N}$. Each flow i transmits data from the source node to the connected destination node and uses a set $L_i \subseteq \mathcal{L}$ of links. An $L \times N$ routing matrix is defined with entries as

$$R_{li} = \begin{cases} 1, & l \in L_i \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

For simplicity, the timeline is divided into time slots in seconds and indexed by $t \in \mathbb{N}$. Each flow maintains a data queue buffer at the source node where tasks can wait before sending. For the queue buffer of flow i at source node $s_i, i \in \mathcal{N}$, we denote the queue length in time slot t as $Q_i(t)$. The transmission queue dynamics is expressed as

$$Q_i(t+1) = \max\left\{Q_i(t) + a_i(t) - x_i(t), 0\right\}. \tag{2}$$

where $a_i(t)$ is the amount of data generated by flow i at source node s_i in time slot t, and $x_i(t)$ is the sending rate of flow i

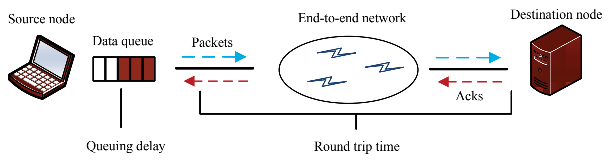

Fig. 1. Schematic diagram of the service delay.

at the source node s_i. Flow i adjusts its congestion window, denoted by $w_i(t)$, to change the sending rate distributively. For the communication link $l \in \mathcal{L}$, the backlog in time slot t is denoted by $b_l(t)$ which evolves as

$$b_l(t+1) = \max\left\{b_l(t) + \sum_{i\in\mathcal{N}} R_{li}x_i(t) - c_l, 0\right\}. \quad (3)$$

Since the network structure from the source node s_i to the destination node d_i is unknown, we consider it as a whole system and denote the RTT as $D_i(t)$. When given a sending rate $x_i(t)$, we can get $Q_i(t+1)$ according to (2) and $D_i(t)$ fed back from the network. Therefore, the total service delay $T_i(t)$ is the sum of the flow i's queuing delay at the source node s_i and the RRT fed from the network, i.e.,

$$T_i(t) = \frac{Q_i(t+1)}{\lambda_i} + D_i(t), \quad (4)$$

where λ_i is the average data generation rate of flow i in the source node s_i, i.e., $\lambda_i = \mathbb{E}[a_i(t)]$. Based on *Little's Law* [20], the average queuing delay is proportional to the ratio of the average queue length to the average arrival rate. In order to achieve congestion control and meet the deterministic delay in end-to-end communication, we impose the deterministic constraint on the service delay of each flow $i \in \mathcal{N}$ as follows:

$$\lim_{T\to+\infty} \frac{1}{T}\sum_{t=0}^{T-1} \Pr\left(T_i(t) \geq L_{i,\max}\right) \leq \epsilon_i. \quad (5)$$

where $L_{i,\max}$ is the maximum end-to-end service delay and $\epsilon_i \ll 1$ is the corresponding tolerant probability of delay violation.

The channel transmission resources can be fully utilized when we increase the size of the congestion window. However, the congestion even packet loss can occur for overly large windows. Based on TCP Vegas, we define the utility function as $U(x_i(t)) = \alpha_i p_i \log x_i(t)$, where α_i is a constant and p_i is the propagation delay of flow i. Considering the deterministic constraint of the whole system delay, the network utility maximization problem is formulated as follows:

$$\begin{aligned} \textbf{P:} \quad & \max_{\boldsymbol{x}(t)\geq 0} \sum_{i\in\mathcal{N}} \bar{U}_i \\ & \text{s.t.} \sum_{i\in\mathcal{N}} R_{li}x_i(t) \leqslant c_l, l \in L \\ & \lim_{T\to+\infty} \frac{1}{T}\sum_{t=0}^{T-1} \Pr\left(T_i(t) \geq L_{i,\max}\right) \leq \epsilon_i, \end{aligned} \quad (6)$$

where $\bar{U}_i = \lim_{T\to+\infty} \frac{1}{T}\sum_{t=0}^{T-1} U(x_i(t))$ is the long-term time-averaged utility function of flow i, and $\boldsymbol{x}(t) = (x_i(t), i \in \mathcal{N})$ is the sending rate vector containing all flows.

IV. Two-Timescale Congestion Window Control (TCWC)

For flows that violate the deterministic delay constraints, we should give a higher priority to ensure the performance of the service. In this section, we consider a two-timescale flow priority assignment and congestion window control mechanism. Specifically, every successive T_0 time slots are grouped as a time frame and denoted by $\mathcal{T}(m) = [(m-1)T_0, \cdots, mT_0 - 1]$, in which $m \in \mathbb{Z}^+$ is the index of the time frame. In the ending of each time frame (i.e., the long timescale), each flow is assigned unique priority which determines the step size of the congestion window adjustment. Subsequently, in each time slot within the mth frame, (i.e., the short timescale), each flow adjusts the size of the congestion window based on the assigned priority distributively.

In this section, we first transform the network utility maximization into the drift-plus-penalty minimization, via Lyapunov optimization, in each time slot, taking into account the queue backlog of flows at source nodes and links, and deterministic delay constraints. Then we obtain the adjustment strategy of the congestion window in each time frame according to the Lagrangian dual method and the KKT condition. In the ending of each time frame, we use extreme value theory to process and analyze the service delay data in the frame, and update the priority of all flows.

A. Per-slot Optimization via Lyapunov Optimization

Firstly, we know that $\Pr(T_i \geq L_{i,\max}) \leq \epsilon_i$ can be recast as $\mathbb{E}[T_i] \leq \epsilon_i L_{i,\max}$ for each flow i by using the upper bound condition $\Pr(T_i \geq L_{i,\max}) \leq \mathbb{E}[T_i]/L_{i,\max}$ based on the Markov's inequality [21]. So the deterministic delay constraints in (6) can be equivalently rewritten as

$$\lim_{T\to+\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left[T_i(t)\right] \leq \epsilon_i L_{i,\max}. \quad (7)$$

The long-term average constraints on queue or delay can be transformed to queue stability constraints based on the concept of virtual queue [22]. We introduce the virtual queue Q_i^V for the constraint (7), and give the queue dynamics as

$$Q_i^V(t+1) = \max\{Q_i^V(t) + T_i(t) - \epsilon_i L_{i,\max}, 0\}. \quad (8)$$

So we only need to ensure that the virtual queue is stable, which is equivalent to satisfying the time-averaged constraints. The problem $\mathbf{P}$ is equivalently transferred to

$$\begin{aligned} \textbf{P1:} \quad & \max_{\boldsymbol{x}(t)\geq 0} \sum_{i\in\mathcal{N}} \bar{U}_i \\ & \text{s.t.} \sum_{i\in\mathcal{N}} R_{li}x_i(t) \leqslant c_l, l \in L \\ & \text{Stability of (2), (8).} \end{aligned} \quad (9)$$

Let $\mathbf{Q}(t) = \big(Q_i(t), Q_i^V(t) : i \in \mathcal{N}\big)$ denote the physical and virtual queues vector and define the conditional expected Lyapunov drift-plus-penalty in time slot t as

$$\mathbb{E}\Big[V\big(\mathcal{L}(\mathbf{Q}(t+1)) - \mathcal{L}(\mathbf{Q}(t))\big) - \sum_{i\in\mathcal{N}} U(x_i(t))|\mathbf{Q}(t)\Big], \quad (10)$$

where $\mathcal{L}\left(\mathbf{Q}(t)\right)$ is the Lyapunov function, described as

$$\mathcal{L}\left(\mathbf{Q}(t)\right) = \frac{1}{2}\sum_{i\in\mathcal{N}} \Big(\big[Q_i(t)\big]^2 + \big[Q_i^V(t)\big]^2\Big). \quad (11)$$

The weight parameter $V > 0$ controls the tradeoff between the queue length stability and the accuracy of the optimal solution of (9).

Theorem 1: For the optimization problem

$$\begin{aligned}\textbf{P2:} \quad \min_{\boldsymbol{x}(t)\geq 0} \quad & \sum_{i\in\mathcal{N}} \bigg(V(a_i(t) - x_i(t))\Big(\Big(1 + \frac{1}{\lambda_i^2}\Big)Q_i(t) \\ & + \frac{1}{\lambda_i}(Q_i^V(t) + D_i(t) - \epsilon_i L_{i,\max})\Big) - U(x_i(t))\bigg) \\ \text{s.t.} \quad & \sum_{i\in\mathcal{N}} R_{li} x_i(t) \leqslant c_l, l \in L. \end{aligned} \quad (12)$$

The optimal solution to the problem will asymptotically approach to the problem **P1** as V decreases.

Proof: According to (2) and $\{\max\{(\cdot), 0\}\}^2 \leq (\cdot)^2$, we can get

$$\begin{aligned} & \Delta Q_i(t) \\ = \; & Q_i^2(t+1) - Q_i^2(t) \\ = \; & \{\max\{Q_i(t) + a_i(t) - x_i(t), 0\}\}^2 - Q_i^2(t) \\ \leq \; & (Q_i(t) + a_i(t) - x_i(t))^2 - Q_i^2(t) \\ = \; & (a_i(t) - x_i(t))^2 + 2Q_i(t)(a_i(t) - x_i(t)). \end{aligned} \quad (13)$$

Similarly, according to (8), we have

$$\begin{aligned} & \Delta Q_i^V(t) \\ = \; & [Q_i^V(t+1)]^2 - [Q_i^V(t)]^2 \\ = \; & \{\max\{Q_i^V(t) + T_i(t+1) - \epsilon_i L_{i,\max}, 0\}\}^2 - [Q_i^V(t)]^2 \\ \leq \; & (Q_i^V(t) + T_i(t+1) - \epsilon_i L_{i,\max})^2 - [Q_i^V(t)]^2 \\ = \; & (T_i(t+1) - \epsilon_i L_{i,\max})^2 + 2Q_i^V(t)(T_i(t+1) - \epsilon_i L_{i,\max}) \\ = \; & \Big(\frac{Q_i(t+1)}{\lambda_i} + D_i(t) - \epsilon_i L_{i,\max}\Big)^2 + 2Q_i^V(t) \\ & * \Big(\frac{Q_i(t+1)}{\lambda_i} + D_i(t) - \epsilon_i L_{i,\max}\Big) \\ = \; & \frac{(a_i(t) - x_i(t))^2 + Q_i^2(t)}{\lambda_i^2} + (D_i(t) - \epsilon_i L_{i,\max})^2 \\ & + \frac{2(a_i(t) - x_i(t))}{\lambda_i}\Big(\frac{Q_i(t)}{\lambda_i} + Q_i^V(t) + D_i(t) - \epsilon_i L_{i,\max}\Big) \\ & + 2Q_i^V(t)\frac{Q_i(t)}{\lambda_i} + 2\Big(Q_i^V(t) + \frac{Q_i(t)}{\lambda_i}\Big)\Big(D_i(t) - \epsilon_i L_{i,\max}\Big). \end{aligned} \quad (14)$$

The upper bound of the Lyapunov drift $\Delta\mathcal{L}_t = \mathcal{L}\left(\mathbf{Q}(t+1)\right) - \mathcal{L}\left(\mathbf{Q}(t)\right)$ in time slot t can be derived as

$$\begin{aligned} \Delta\mathcal{L}_t = & \frac{1}{2}\sum_{i\in\mathcal{N}} \Big([\Delta Q_i(t)]^2 + [\,\Delta Q_i^V(t)]^2\Big) \\ \leq & \sum_{i\in\mathcal{N}} (a_i(t) - x_i(t))\Big(\Big(1 + \frac{1}{\lambda_i^2}\Big)Q_i(t) + \frac{1}{\lambda_i}(Q_i^V(t) \\ & + D_i(t) - \epsilon_i L_{i,\max})\Big) + C_0 + C_1(t), \end{aligned} \quad (15)$$

where

$$\begin{aligned} C_0(t) = & \sum_{i\in\mathcal{N}} \frac{1}{2}\Big(1 + \frac{1}{\lambda_i^2}\Big)(a_i(t) - x_i(t))^2, \\ C_1(t) = & \sum_{i\in\mathcal{N}} \bigg(\frac{1}{2}\Big(\frac{Q_i^2(t)}{\lambda_i^2} + (D_i(t) - \epsilon_i L_{i,\max})^2\Big) \\ & + Q_i^V(t)\frac{Q_i(t)}{\lambda_i} + \Big(Q_i^V(t) + \frac{Q_i(t)}{\lambda_i}\Big)\Big(D_i(t) - \epsilon_i L_{i,\max}\Big)\bigg). \end{aligned} \quad (16)$$

Here, $C_0(t)$ is bounded based on the queue stability assumption. and $C_1(t)$ is independent of the optimization variable $x_i(t)$ except $D_i(t)$. These two items do not affect system performance in Lyapunov optimization, so we can omit details for simplified analysis. Therefore, the long-term time-averaged problem **P1** can be transformed into the minimizing the drift-plus-penalty upper bound in each time slot t. Due to the stability of queues, we have $\mathbb{E}(Q_i(t)) = \mathbb{E}(Q_i^V(t)) = 0$. Besides, the model parameters are all average bounded. Therefore, according to (10) and (15), we can get

$$\begin{aligned} \mathbb{E}\Big[V\Delta\mathcal{L}_t - \sum_{i\in\mathcal{N}} U(x_i(t))\Big] \leq & \mathbb{E}\Big[V\Big(\sum_{i\in\mathcal{N}}(a_i(t) - x_i(t)) \\ & * (D_i(t) - \epsilon_i L_{i,\max}) + C_0 + C_1(t)\Big) - \sum_{i\in\mathcal{N}} U(x_i(t))\Big] \\ \leq & V * C - \mathbb{E}\Big[\sum_{i\in\mathcal{N}} U(x_i^*(t))\Big] = V * C - \sum_{i\in\mathcal{N}} U(x_i^*), \end{aligned} \quad (17)$$

where C is a constant and x_i^* is the optimal solution to **P1**. Accumulate both sides of the above inequality from time slot 0 to $t-1$ and let $\mathcal{L}\left(\mathbf{Q}(0)\right) = 0$, we can derive

$$\frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\Big[-\sum_{i\in\mathcal{N}} U(x_i(\tau))\Big] \leq V * C - \sum_{i\in\mathcal{N}} U(x_i^*). \quad (18)$$

Obviously, the time average expected of negative utility function can be arbitrarily close to the optimal x_i^* as V decreases. ■

In fact, there is an implicit relationship between $D_i(t)$ and $\boldsymbol{x}(t)$ due to the unknown of multi-flow transmission coupling. Therefore, we cannot directly solve the optimal solution to this problem, which is also in line with the mechanism that the congestion window should be gradually adjusted according to the feedback RTT.

B. Congestion Window Adjustment Based on Service Delay

Since the sending rates $x_i(t)$ of all flows should be independent of each other, we consider using the Lagrangian dual method to decouple the constraints. The Lagrangian of problem **P2** can be obtained as

$$\begin{aligned} h(\boldsymbol{x}(t), \boldsymbol{\mu}) = & \sum_{i \in \mathcal{N}} \Bigg(V(a_i(t) - x_i(t)) \Big(\Big(1 + \frac{1}{\lambda_i^2}\Big) Q_i(t) \\ & + \frac{1}{\lambda_i}(Q_i^V(t) + D_i(t) - \epsilon_i L_{i,\max}) \Big) \\ & - U(x_i(t)) \Bigg) + \sum_{l \in L} \mu_l \Big(\sum_{i \in \mathcal{N}} R_{li} x_i(t) - c_l \Big) \\ = & \sum_{i \in \mathcal{N}} \Bigg(V(a_i(t) - x_i(t)) \Big(\Big(1 + \frac{1}{\lambda_i^2}\Big) Q_i(t) \\ & + \frac{1}{\lambda_i}(Q_i^V(t) + D_i(t) - \epsilon_i L_{i,\max}) \Big) \\ & - U(x_i(t)) + x_i(t) \sum_{l \in L} R_{li} \mu_l \Bigg) - \sum_{l \in L} c_l \mu_l, \end{aligned} \tag{19}$$

where $\boldsymbol{\mu} = (\mu_l \geq 0, l \in L)$ is the Lagrange multiplier vector associated with the inequality constraint. The Lagrange dual function $d(\boldsymbol{\mu})$ is the lower bound of the Lagrangian, i.e., $\inf_{\boldsymbol{x}(t)} h(\boldsymbol{x}(t), \mu)$. Therefore, the Lagrange dual problem is formulated as

$$\max_{\boldsymbol{\mu} \geq 0} \ d(\boldsymbol{\mu}) = \max_{\boldsymbol{\mu} \geq 0} \ \min_{\boldsymbol{x(t)} \geq 0} \ h(\boldsymbol{x}(t), \boldsymbol{\mu}) \tag{20}$$

Because the upper bound of Lyapunov-drift-penalty is a convex function of x_i and the constraints are linear, **P2** is a convex optimization problem. The optimal solution $\boldsymbol{x}^*(t) = (x_i^*(t), i \in \mathcal{N})$ of **P2** and the optimal solution $\boldsymbol{\mu}^* = (\mu_l^* \geq 0, l \in L)$ of the dual problem can be obtained according to the KKT conditions, i.e.,

$$\begin{cases} \sum_{i \in \mathcal{N}} R_{li} x_i^*(t) \leqslant c_l, \\ \mu_l^* \geq 0, \\ \dfrac{\partial h(\boldsymbol{x}^*(t), \boldsymbol{\mu}^*)}{\partial x_i(t)} = 0, \\ \mu_l^* \Big(\sum_{i \in \mathcal{N}} R_{li} x_i^*(t) - c_l \Big) = 0. \end{cases} \tag{21}$$

According to (19), we can get

$$\frac{\partial h(\boldsymbol{x}^*(t), \boldsymbol{\mu}^*)}{\partial x_i(t)} = \sum_{l \in L} R_{li} \mu_l^* - \frac{\alpha_i p_i}{x_i^*(t)} - V B_i^*(t) \tag{22}$$

where

$$B_i^*(t) = \Big(1 + \frac{1}{\lambda_i^2}\Big) Q_i(t) + \frac{1}{\lambda_i}(Q_i^V(t) + D_i^*(t) - \epsilon_i L_{i,\max}) \tag{23}$$

When the network meets the determined requirements and reaches the optimum, the number of packets queued in routers of the flow i's path can be easily expressed as

$$w_i^*(t) - p_i x_i^*(t) = (D_i^*(t) - p_i) x_i^*(t) = \sum_{l \in L} R_{li} b_l^* \frac{x_i^*(t)}{c_l}, \tag{24}$$

where $w_i^*(t)$ is the congestion window of flow i and b_l^* is the backlog of the communication link l when the whole system reaches equilibrium. Let $u_l^* = \frac{b_l^*}{c_l}$ and according to (21), (22) and (24), we can derive

$$x_i^*(t) = \frac{\alpha_i p_i}{\sum_{l \in L} R_{li} \frac{b_l^*}{c_l} - V B_i(t)} = \frac{\alpha_i p_i}{D_i^*(t) - p_i - V B_i^*(t)} \tag{25}$$

Obviously, $u_l^* \geq 0$ and $\frac{\partial h(\boldsymbol{x}^*(t), \boldsymbol{\mu}^*)}{\partial x_i(t)} = 0$. When $u_l^* = 0$, $b_l^* = 0$ and $\mu_l^* \Big(\sum_{i \in \mathcal{N}} R_{li} x_i^*(t) - c_l \Big) = 0$. In this case, the backlog of communication link l is empty, which means the total arrival rate does not exceed its capacity, i.e., $\sum_{i \in \mathcal{N}} R_{li} x_i^*(t) \leqslant c_l$. When $u_l^* > 0$, $\sum_{i \in \mathcal{N}} R_{li} x_i^*(t) = c_l$. In summary, $(x_i^*(t), \mu_l^*)$ satisfies the KKT conditions in (21). Therefore, $x_i^*(t)$ is the optimal solution of problem **P2**. In the fluid model, the sending rate x_i can be approximated by the ratio of the congestion window to the RTT, i.e., $x_i^*(t) = \frac{w_i^*(t)}{D_i^*(t)}$, so we can get

$$\frac{w_i^*(t)}{p_i} - \Big(1 + \frac{V B_i^*(t)}{p_i}\Big) \frac{w_i^*(t)}{D_i^*(t)} = \alpha_i. \tag{26}$$

The above equation represents that the entire network has reached the optimal solution of the original utility function maximization problem, otherwise we should adjust the congestion window according to the current network conditions. Similar to TCP Vegas, the congestion window is only increased or decreased by 1 per RTT, which depends on the size of both sizes in (26). For flow i at the source node s_i, when the congestion window is $w_i(t)$ and the network end-to-end delay is $D_i(t)$, we design the update strategy of the congestion window as

$$w_i(t+1) = w_i(t) - \frac{1}{D_i(t)} \mathrm{sgn}(\nabla_i(t) - \alpha_i), \tag{27}$$

where

$$\begin{aligned} \nabla_i(t) &= \frac{w_i(t)}{p_i} - \frac{w_i(t)}{D_i(t)} - V \frac{B_i(t) w_i(t)}{p_i D_i(t)}, \\ B_i(t) &= \Big(1 + \frac{1}{\lambda_i^2}\Big) Q_i(t) + \frac{1}{\lambda_i}(Q_i^V(t) + D_i(t) - \epsilon_i L_{i,\max}), \end{aligned} \tag{28}$$

where $\mathrm{sgn}(\cdot)$ is the sign function. Obviously, the difference between the first two terms of $\nabla_i(t)$ is consistent with traditional TCP Vegas, where is represents the number of packets in the entire network. Besides, we additionally consider the impact of physical and virtual queues.

C. Delay-Aware Priority Based on Extreme Value Analysis

To judge whether the probability constraints is satisfied, we consider the case of service delay violation $T_i(t) > L_{i,\max}$, which is an extreme event with low probability when $\epsilon_i \ll 1$. The extreme value theory can help us characterize the tail distribution and statistics of extreme events.

Based on the *Pickands-Balkema-de Haan Theorem* [23], the conditional complementary cumulative distribution function (CCDF) of the excess value can be estimated as follows. For a random variable T with the cumulative distribution function (CDF) $F_T(t)$, and a threshold value t_u, when the threshold $t_u \to F_T^{-1}(1) = \sup\{t : F_T(t) < 1\}$, the CCDF of the excess value $X|_{T>t_u} = T - t_u > 0$, i.e., $\bar{F}_{X|T>t_u}(x) = \Pr(T - t_u > x | T > t_u)$, can be approximately expressed as

$$G(x;\sigma,\xi) = \left(1 + \frac{\xi x}{\sigma}\right)^{-1/\xi}; \sigma > 0, \xi \in \mathbb{R}. \quad (29)$$

Here, $G(x;\sigma,\xi)$ is called the generalized Pareto distribution (GPD), which contains a scale parameter σ and a shape parameter ξ. The probability density function can be obtained easily as

$$g(x;\sigma,\xi) = \frac{d[1 - G(x;\sigma,\xi)]}{dx} = \frac{1}{\sigma}\left(1 + \frac{\xi x}{\sigma}\right)^{-1-1/\xi}. \quad (30)$$

In other words, regardless of the distribution of T, the cumulative distribution function of the excess value X can be approximated as a GPD distribution and is equivalent when the threshold t_u satisfied $F_T(t_u) = 1$, i.e., the threshold takes the upper bound of T. Obviously, the parameters $\theta = [\sigma,\xi]$ determine the tail distribution characteristics of extreme events and need to be estimated based on the actual data.

In the mth time frame, we can obtain service delay data set $T_{i,m} = \{T_i(t), t \in \mathcal{T}(m)\}$ of each flows i by getting the real-time queue buffer at the source node s_i and the RTT fed from the destination node d_i. The selected threshold value $t_{i,u}$ should be less than $L_{i,\max}$, otherwise it will task more observation time to get enough excess value data. Besides, threshold value $t_{i,u}$ should also be large enough to approximately satisfy GPD distribution. For a given threshold $t_{i,u} < L_{i,\max}$, the excess value data set can be denoted as $\mathcal{T}_{i,m} = \{T_i(t) - t_{i,u} | T_i(t) > t_{i,u}, t \in \mathcal{T}(m)\}$. In order to fit GPD distribution with $\mathcal{T}_{i,m}$, we use the maximum likelihood estimation method to find the optimal parameters by minimizing the negative log-likelihood function, which can be expressed as

$$\min_{\theta \in \Theta(\mathcal{T}_{i,m})} \quad -\frac{1}{|\mathcal{T}_{i,m}|}\sum_{T \in \mathcal{T}_{i,m}} \ln\ g(T;\theta) = \frac{1}{|\mathcal{T}_{i,m}|}\sum_{T \in \mathcal{T}_{i,m}} I_\theta(T), \quad (31)$$

where

$$I_\theta(T) = \ln\ \sigma + \left(1 + \frac{1}{\xi}\right)\ln\left(1 + \frac{\xi T}{\sigma}\right), \quad (32)$$

Algorithm 1 Two-Timescale Congestion Window Control

```
1: Input:   $V$, $\lambda_i$, $t_{i,u}$, $\forall i \in \mathcal{N}$.
2: Initialization:
       $w_i(0) = \gamma_i = 1$, $Q_i(0) = Q_i^V(0) = 0$, $T_i = \emptyset$, $\forall i \in \mathcal{N}$.
3: for $m = 0, 1, 2, ...$ do
4:     for $t = mT_0, ..., (m+1)T_0 - 1$ do
5:         Get $D_i(t)$ and update $p_i$ based on current RTT data.
6:         Adjust the congestion window to $w_i(t+1)$ in (39).
7:         Update the queue length according to (2) and (8).
8:         Add $T_i(t)$ to service delay data set $T_i$.
9:     end for
10:    Obtain excess value data $\mathcal{T}_i$ to estimate parameter
       $(\sigma_i^*, \xi_i^*)$ according to (34).
11:    Update $\gamma_i$ according to (40) and set $T_i = \emptyset$.
12: end for
```

and $\theta \in \Theta(\mathcal{T}_{i,m}) = \{(\sigma,\xi) | \sigma > 0, 1 + \xi T/\sigma \geq 0, \forall T \in \mathcal{T}_{i,m}\}$. The partial derivative of the function $I_\theta(T)$ with respect to parameter $\theta = (\sigma,\xi)$ can be obtained as

$$\nabla I_\theta(T) = \begin{bmatrix} \frac{\partial I_\theta(T)}{\partial \sigma} \\ \frac{\partial I_\theta(T)}{\partial \xi} \end{bmatrix} = \begin{bmatrix} \frac{\sigma - T}{\sigma(\sigma + \xi T)} \\ \frac{(1+\xi)T}{\xi(\sigma+\xi T)} - \frac{1}{\xi^2}\ln\left(1 + \frac{\xi T}{\sigma}\right) \end{bmatrix}. \quad (33)$$

In order to solve the large scale optimization problem, we use the stochastic variance reduced gradient (SVRG) method to obtain the optimal θ^* and the iterative method has a faster convergence rate [24]. The specific estimation of θ using SVRG method at iteration n are described as follows:

$$\begin{cases} y_n = & \theta_{n-1} - \eta[\nabla I_{\theta_{n-1}}(T) - \nabla I_{\widetilde{\theta}}(T) + \nabla R_{\widetilde{\theta}}(\mathcal{T}_{i,m})], \\ \theta_n = & \operatorname{argmin}_{\theta \in \Theta(\mathcal{T}_{i,m})} ||y_n - \theta||, \end{cases} \quad (34)$$

where η is the learning rate, T is a randomly selected sample from $\mathcal{T}_{i,m}$, $\widetilde{\theta} = \frac{1}{n}\sum_{k=0}^{n-1}\theta_k$ is an average estimate of parameters θ for the previous n iterations and $\nabla R_{\widetilde{\theta}}(\mathcal{T}_{i,m}) = \frac{1}{|\mathcal{T}_{i,m}|}\sum_{T \in \mathcal{T}_{i,m}} \nabla I_{\widetilde{\theta}}(T)$ is an average estimate of the gradient. Through multiple update iterations, we can get the optimal estimates $\theta^* = (\sigma_i^*, \xi_i^*)$ for the sample set $\mathcal{T}_{i,m}$, i.e.,

$$\bar{F}_{X|T_i > t_{i,u}}(t) = \Pr(T_i - t_{i,u} \geq t | T_i > t_{i,u}) \approx G(t,\sigma_i^*, \xi_i^*). \quad (35)$$

Moreover, the conditional probability can be expressed as

$$\bar{F}_{X|T_i > t_{i,u}}(t) = \frac{1 - F_{T_i}(t + t_{i,u})}{1 - F_{T_i}(t_{i,u})}, \quad (36)$$

where $1 - F_{T_i}(t_{i,u})$ is the probability of exceeding the latency threshold $t_{i,u}$ which can be estimated by the sample set as $\frac{|\mathcal{T}_{i,m}|}{|T_{i,m}|}$. Combining (29), (35) and (36), we can get

$$F_{T_i}(t + t_{i,u}) = 1 - \frac{|\mathcal{T}_{i,m}|}{|T_{i,m}|}\left(1 + \frac{\xi_i^* t}{\sigma_i^*}\right)^{-1/\xi_i^*}. \quad (37)$$

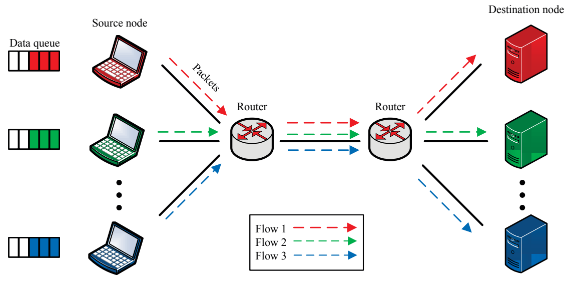

Fig. 2. Link sharing network scenario.

Let $t + t_{i,u} = L_{i,\max}$, we can get

$$\begin{aligned}\Pr(T_i \geq L_{i,\max}) &= 1 - F_{T_i}(L_{i,\max}) \\ &= \frac{|\mathcal{T}_{i,m}|}{|T_{i,m}|}\left[1 + \frac{\xi_i^*(L_{i,\max} - t_{i,u})}{\sigma_i^*}\right]^{-1/\xi_i^*},\end{aligned} \quad (38)$$

and adjust congestion window by comparing with ϵ_i. When $\Pr(T_i \geq L_{i,\max}) > \epsilon_i$, flow i is more urgently compared to those flows that satisfy the deterministic delay constraints. We introduce γ_i to change the adjustment speed of the congestion window w_i. The update strategy in (27) can be modified as

$$w_i(t+1) = \begin{cases} w_i(t) - \frac{1}{D_i(t)}, & \operatorname{sgn}(\nabla_i(t)) \geq 0 \\ w_i(t) + \frac{\gamma_i(m)}{D_i(t)}, & \operatorname{sgn}(\nabla_i(t)) < 0 \end{cases}, \quad (39)$$

where

$$\gamma_i(m+1) = \begin{cases} \beta_i, & \Pr(T_i \geq L_{i,\max}) < \epsilon_i \\ \gamma_i(m) + 1, & \Pr(T_i \geq L_{i,\max}) \geq \epsilon_i \end{cases}, \quad (40)$$

is a quantified form of the flow i's priority. For the flow whose service delay does not satisfy deterministic constraints, we should increase $\gamma_i(m)$ to give the higher priority. For other flows, we set the priority as β_i close to 1, which means that all flows whose current delay situation is not bad allocate a small amount of communication resources to more urgent flows. The steps of TCWC mechanism are detailed in Algorithm 1.

V. Simulation Results

In the simulation, we consider a link sharing network scenario where three source nodes are connected with three destination nodes sharing a communication link, as shown in Fig. 2. Three flows transmit data separately from different source nodes to their respective connected destination nodes. All links from source nodes to routers, between routers, and from routers to destination nodes are set as the channel with 5Mbps bandwidth and 2ms propagation delay. Three nodes share a common communication channel and the size of the data packets generated by each device is uniformly 1000 bits. The rest of simulation parameter values are set as: $\boldsymbol{\lambda} = [2.25, 1.5, 0.75]$ Mbps, $\alpha = 4$, $L_{\max} = 0.12$s, $\epsilon = 0.01$, $V = 10^{-6}$.

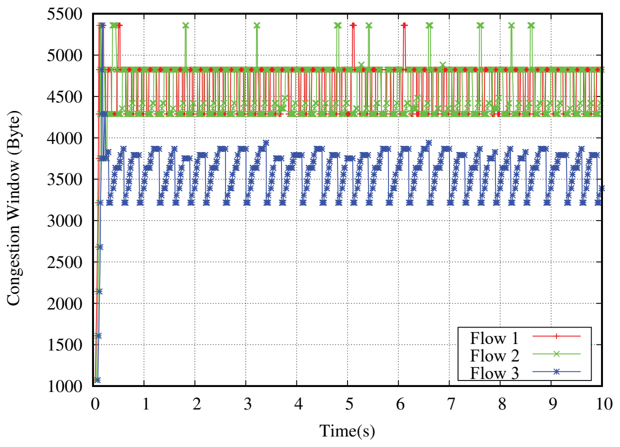

Fig. 3. Congestion window size for TCP Vegas.

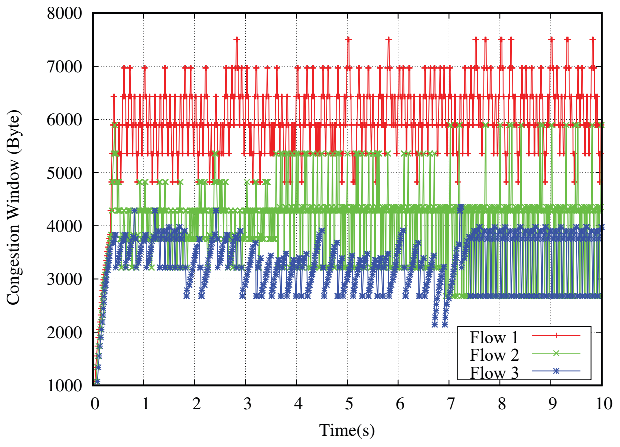

Fig. 4. Congestion window size for TCWC.

We compare the proposed TCWC algorithm with classic TCP Vegas in Figs. 3 and 4. At the same time, we consider the effect of different data generation rates in flows on congestion window adjustment. The generation rates of application data in the three node devices are same as parameter $\boldsymbol{\lambda}$. On the whole, TCP Vegas allocates the congestion window size equally to flows 1 and 2, excluding flow 3 that has not reached the average but is already optimal. In contrast, TCWC gives flows of higher data generation rates larger congestion windows to reduce the length of queue buffers at the source node. In the initial stage, both TCP Vegas and TCWC adopt the default scheme to increase the congestion window due to less RTT samples and underutilized data queue buffers. For the flows of the higher data generation, it can receive ACK and RTT earlier and adjust the congestion window faster. During slow start, the local queue buffers will increase rapidly, which play a major role in the service delay and affect the window adjustment strategy. Subsequently, different flows update the congestion window distributively according to (27) and (28). In order to facilitate the experiment, we choose the appropriate weight

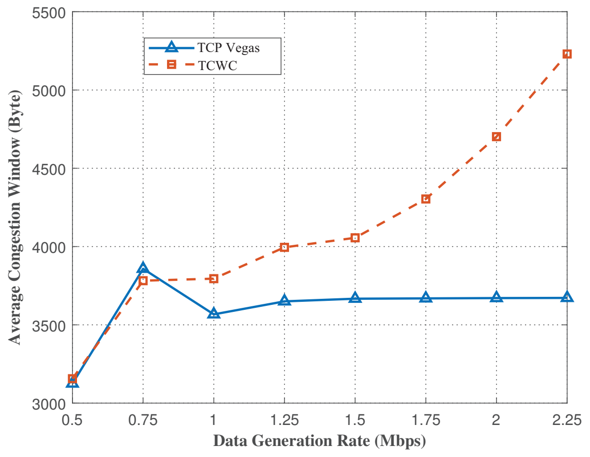

Fig. 5. Average congestion window v.s. data generation rate.

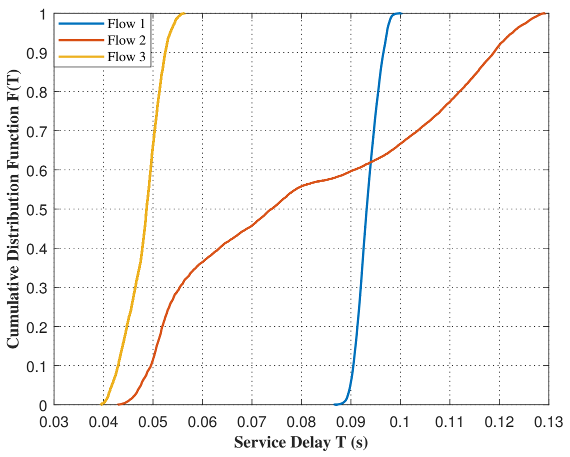

Fig. 7. CDF of service delay for TCP Vegas.

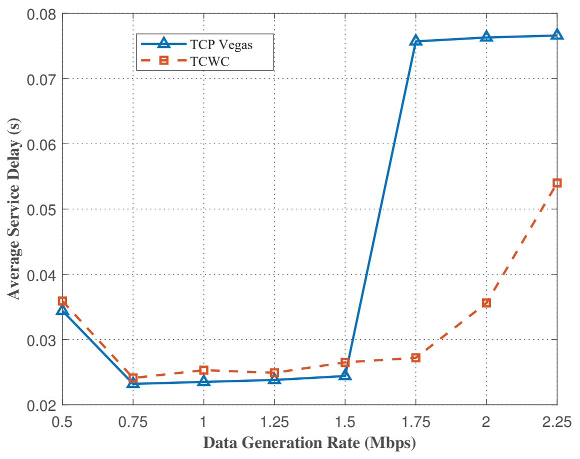

Fig. 6. Average delay v.s. data generation rate.

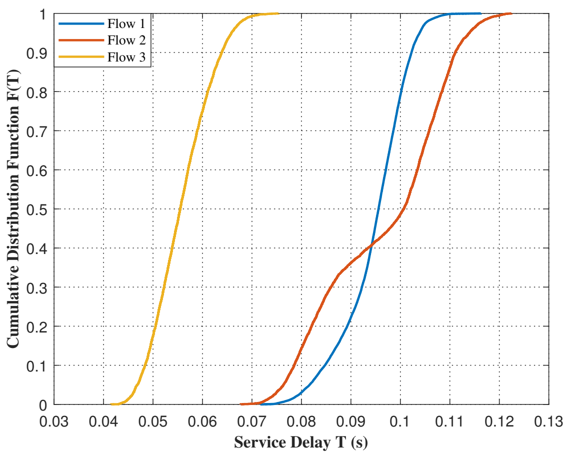

Fig. 8. CDF of service delay for TCWC.

parameter V to make $\nabla_i(t)$ and α_i on the same order of magnitude. After about 1 seconds, the congestion window has stabilized in TCP Vegas. However, considering flow queuing delay at source nodes, TCWC takes about 8 seconds for the entire network utility to reach optimal. Obviously, instead of fairness, at the cost of reducing the number of windows for other nodes, TCWC assigns a large congestion window to flow 1 for better performance.

In Figs. 5 and 6, the average-based congestion window and service delay are compared between TCP Vegas and TCWC algorithm respectively. Compared with the fairness of TCP Vegas, our congestion control mechanism considers flows queue at source nodes and gives higher priority to the flows that do not satisfy delay constraints. When the data generation rate is small relative to the link capacity, the buffers of source nodes and communication links are almost empty, which means low queuing delay. The service delay does not violate the probability constraint, so the results of the two methods are almost the same. With the data generation rate increases, the queue at the source node plays an important role in service delay, prompting us to increase the congestion window as much as possible to speed data transmission. When the data rate is in the interval $[0.75, 1.5]$Mbps, the resource utilization rate of the entire network reaches the maximum, and the utility function and system performance reach the optimum. As common link capacity is fully utilized, network congestion will occur and cause the longer flow queuing delay. Traditional TCP Vegas focuses on the end-to-end delay and adjusts the congestion window fairly, resulting in higher service delay for flows with larger data generation rates. In contrast, TCWC will sense the queuing delay at source nodes and increase the congestion window to decrease the service delay, thereby maximize the total network utility.

Figs. 7 and 8 compare the CDF of service delay for different flows between TCP Vegas and TCWC algorithm. In order to obtain sample data whose service delay exceeds the threshold

$L_{\max}$, we set the delay of the common communication link to 8ms and the other links to 4ms. In TCP Vegas, except for flow 2 where there is a small amount of sample data exceeds the threshold, the service delay of other flow's data packet is stable within a cell range of about 0.02, or even smaller. Although the packet service delays of flow 1 and 3 are both low and meet the deterministic constraints, about 10 percent of the packets at flow 2 exceed the delay threshold, which is a heavy-tailed distribution. It is unacceptable for delay sensitive applications. Since TCP Vegas is relatively fair to all flows in end-to-end congestion control, the large difference is mainly reflected in flow queuing delay at source nodes. In TCWC, we take into account the deterministic requirements in (5) and give higher priority to flow 1 that violates the constraint. In order to make each flow meet the deterministic delay constraint as much as possible in the limited communication resources, we increase the sending rate to reduce the queue length generated by flow 2 at source node 2, and try to avoid the occurrence of congestion. Correspondingly, the sending rate of flow 3 decreases, and the service delay also increases. Therefore, we make the urgent flow 2 satisfy the delay constraint by sacrificing the average performance of flow 3 with less delay. Besides, we can see that the CDFs of the service delay for all flows dynamically approach to the light-tailed distribution.

VI. Conclusion and Future Works

In this paper, we propose a two-timescale congestion window control (TCWC) mechanism with delay-aware priority based on TCP Vegas. We consider the service delay from the application perspective, including not only the network delay but also the queueing delay at source nodes. By designing a virtual queue, we transform the deterministic delay constraint to the time-averaged queue stability, and solve it in each time slot according to the Lyapunov drift-plus-penalty method. Then we obtain the adjustment strategy of congestion window according to Lagrangian duality theory. For further guarantee the service delay, we apply extreme value theory to evaluate the priorities of different flows, which determines the update rate of congestion window. The simulation results show that the proposed TCWC can adjust the congestion window considering the queueing delay at the source nodes, which can significantly reduce the average service delay and provide better service delay guarantee than traditional TCP Vegas.

Although this work is based on TCP Vegas, the superiority on service delay guarantee is verified to be effective and general. In future works, our proposed two-timescale framework can be extended to various existing optimization-based congestion control algorithms for providing additional capability of deterministic service delay guarantee.

References

[1] N. Finn, "Introduction to time-sensitive networking," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 22–28, 2018.

[2] M. Khoshnevisan, V. Joseph, P. Gupta, F. Meshkati, R. Prakash, and P. Tinnakornsrisuphap, "5G industrial networks with CoMP for URLLC and time sensitive network architecture," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 4, pp. 947–959, 2019.

[3] F. Gratzer, S. Gallenmüller, and Q. Scheitle, "QUIC-quick UDP internet connections," *Future Internet and Innovative Internet Technologies and Mobile Communications*, 2016.

[4] J. Iyengar, M. Thomson *et al.*, "QUIC: A UDP-based multiplexed and secure transport," *Internet Engineering Task Force, Internet-Draft draft-ietf-quic-transport-27*, 2020.

[5] M. Hock, F. Neumeister, M. Zitterbart, and R. Bless, "TCP lola: Congestion control for low latencies and high throughput," in *IEEE Conference on Local Computer Networks (LCN)*, Oct. 2017.

[6] V. Arun and H. Balakrishnan, "Copa: Practical delay-based congestion control for the internet," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Apr. 2018, pp. 329–342.

[7] R. Mittal, V. T. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats, "Timely: RTT-based congestion control for the datacenter," *ACM Conference on Special Interest Group on Data Communication (SIGCOMM)*, vol. 45, no. 4, pp. 537–550, Aug. 2015.

[8] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter TCP (D2TCP)," *ACM SIGCOMM Conference Applications, Technologies, Architectures, and Protocols for Computer Communication*, vol. 42, no. 4, pp. 115–126, Aug. 2012.

[9] Y. Lu, "Sed: An SDN-based explicit-deadline-aware TCP for cloud data center networks," *Tsinghua Science and Technology*, vol. 21, no. 5, pp. 491–499, 2016.

[10] M. Claeys, N. Bouten, D. De Vleeschauwer, K. De Schepper, W. Van Leekwijck, S. Latré, and F. De Turck, "Deadline-aware TCP congestion control for video streaming services," in *IEEE International Conference on Network and Service Management (CNSM)*, 2016, pp. 100–108.

[11] H. Huang, Z. Sun, and X. Wang, "End-to-end TCP congestion control for mobile applications," *IEEE Access*, vol. 8, pp. 171 628–171 642, 2020.

[12] K. Gao, C. Xu, J. Qin, L. Zhong, and G.-M. Muntean, "A stochastic optimal scheduler for multipath TCP in software defined wireless network," in *IEEE International Conference on Communications (ICC)*, May 2019, pp. 1–6.

[13] H. Cui, X. Liao, S. Xu, and B. Liu, "Lyapunov optimization based energy efficient congestion control for MPTCP in hetnets," in *IEEE International Conference on Communication Technology (ICCT)*, Oct. 2018, pp. 440–445.

[14] C.-F. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4132–4150, 2019.

[15] X. Meng, W. Wang, and Z. Zhang, "Delay-constrained hybrid computation offloading with cloud and fog computing," *IEEE Access*, pp. 21 355–21 367, 2017.

[16] Y. Chen, X. Zhou, W. Wang, H. Wang, and Z. Zhang, "Delay-optimal closed-form scheduling for multi-destination computation offloading," *IEEE Wireless Communication Letters*, vol. 10, no. 9, pp. 1904–1908, 2021.

[17] N. Zhang, W. Wang, R. Lan, P. Zhou, and A. Huang, "Delay-aware cache-enabled cooperative D2D transmission in mobile cellular networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, Mar. 2021, pp. 1–6.

[18] D. Wang, W. Wang, Z. Han, and Z. Zhang, "Delay optimal random access with heterogeneous device capabilities in energy harvesting networks using mean field game," *IEEE Transactions on Wireless Communications*, vol. 20, no. 9, pp. 5543–5557, 2021.

[19] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Distributed federated learning for ultra-reliable low-latency vehicular communications," *IEEE Transactions on Communications*, vol. 68, no. 2, pp. 1146–1159, 2019.

[20] J. D. Little and S. C. Graves, "Little's law," in *Building intuition*. Springer, 2008, pp. 81–100.

[21] B. Ghosh, "Probability inequalities related to markov's theorem," *The American Statistician*, vol. 56, no. 3, pp. 186–190, 2002.

[22] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[23] S. Coles, J. Bawa, L. Trenner, and P. Dorazio, *An introduction to statistical modeling of extreme values*. Springer, 2001, vol. 208.

[24] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," *Advances in neural information processing systems (NIPS)*, vol. 26, Dec. 2013.