

Robust Lyapunov–Krasovskii based design for explicit control protocol against heterogeneous delays

Lina He^{1,2} · Hairui Zhou³

Published online: 1 March 2017
© Springer Science+Business Media New York 2017

Abstract TCP has been extensively credited for the stability of the Internet. However, as the product of bandwidth and latency increases, TCP becomes inefficient and prone to instability. The explicit control protocol (XCP) is a promising congestion control protocol that outperforms TCP in terms of efficiency, fairness, convergence speed, persistent queue length and packet loss rate. However, XCP is not globally stable in the presence of heterogeneous delays. When the ratio of maximum to average transmission latency is sufficiently large, XCP will become instability. In this paper, according to the robust control theory, with the help of a recently developed Lyapunov–Krasovskii functional, an improved version of XCP, named R-XCP, is proposed to solve the weakness of XCP under heterogeneous delays, which adjusts parameter α from an initial value of 0.4 to a reasonable value for improving system robustness. And then, the synthesis problem is reduced to a convex optimization scheme expressed in terms of linear matrix inequalities. Extensive simulations have shown that R-XCP significantly decreases the volatilities of the aggregate traffic rate and control time interval, and indeed achieves this stability goal. Compared with previous work, R-XCP has a better balance between robustness and responsiveness, and the computational complexity declines significantly at the same time. Besides, R-XCP makes the

system less sensitive to flows, which contribute little traffic but maliciously report their transmission delays.

Keywords Explicit control protocol · Heterogeneous delays · Lyapunov–Krasovskii functional

1 Introduction

Congestion control is very important to guarantee specific service quality in communication networks, especially in situations where the availability of resources and the set of competitors change over time unpredictably, yet efficient sharing is expected. Since Jacobson's work in 1988, congestion control has been associated with the transmission control protocol (TCP) in the Internet, and it has performed remarkably well in the past and was generally believed to have prevented severe congestion as the Internet scaled up by six orders of magnitude in size, speed, load and connectivity [1]. However, it is also well-known that TCP becomes inefficient and prone to instability when the product of bandwidth and latency increases [2,3].

The explicit control protocol (XCP) is a novel and promising congestion control protocol that outperforms TCP in terms of efficiency, fairness, convergence speed, persistent queue length and packet loss rate [4]. Then, a large amount of works have been made to further analyze and improve the XCP performances. Zhang and Henderson [5] first implemented the XCP stack in Linux, and revealed that the XCP performance can be adversely affected by the data type choice and TCP/IP configurations. In the incremental deployment, XCP behaves improperly, even worse than TCP, because XCP requires the collaboration of routers along the path, which is impossible in an actual scenario. To address this problem, XCP-i model is proposed to make XCP operable on

✉ Lina He
hlnyh@hhu.edu.cn
Hairui Zhou
zhou_hairui@126.com

¹ School of Earth Science and Engineering, Hohai University, Nanjing 210098, China

² State Key Laboratory of Geo-Information Engineering, Xi'an 710054, China

³ The 28th Institute of CETC, Nanjing 210007, China

an inter-network consisting of XCP routers and traditional IP routers without losing the benefit of the XCP control laws [6]. In a multi-bottleneck network, XCP may cause some bottleneck links to be under-utilized, and a flow may not receive its max–min fair allocation. Yang et al. [7] first pinpoint that it is caused by the bandwidth shuffling operation, and an improved version of XCP is proposed, which modifies XCP to shuffle bandwidth only among flows that are bottlenecked at current routers. This modification ensures that the deallocated bandwidth by the shuffling operation will be reused. Therefore, a link can be fully utilized. Hairui et al. [8] proposes another interesting algorithm based on the proportional integral controller to solve the link under-utilization problem, and it also effectively improves XCP performance in a multi-bottleneck topology, the capacity of each link can be optimally utilized, each flow can obtain its max–min fair rate in steady state. According to the operating principle, XCP needs to know the available bandwidth at the output link, which is a tricky business over wireless links, e.g., IEEE 802.11. XCP-b does a nice job of providing a first step toward running XCP on wireless links. It requires a few parameters and preserves the good properties of XCP, including high utilization, stable throughput, and reasonable fairness [9]. Barreto [10] proposes XCP-Winf, which relies on the MAC layer information gathered by a new method, rt-Winf, to accurately estimate the available bandwidth and the path capacity over a wireless network path. The evaluation results of XCP-Winf, obtained through simulations, show that the rt-Winf algorithm improves significantly XCP behavior, making it more efficient and stable. The satellite network, as a kind of high bandwidth–delay product network, has recently been paid more and more attention because of its large geographic coverage and cost effectiveness. Long link delay and high bit error rate (BER) are characteristics of satellite networks. Zhou et al. [11] and Sun et al. [12] show that high BER has a seriously negative impact on the XCP performance, and two modified variants of XCP are proposed to solve low throughput under high BER conditions. Liu et al. [13] investigates the Hopf bifurcations in the XCP system. These bifurcation behaviors may cause heavy oscillation of average queue length and induce network instability. Then, a time-delayed feedback control method was proposed for controlling Hopf bifurcations in the XCP system. Additionally, many other efforts have been done on different aspects of the XCP scheme, such as efficiency [14], stability [15, 16], security [17], as well as enhancements for optical packet switched (OPS) networks [18, 19].

Among these concerns, one critical defect in XCP was found. Lachlan et al. [20] reveal that XCP will be unstable, when the ratio of maximum to mean round trip time (RTT) is sufficiently large. Their experiments show that large RTT flows cannot obtain their max–min allocations, the aggregate traffic and control interval fluctuate dramatically, and big

buffers are required to absorb these oscillations. Therefore, XCP is not globally stable in the presence of heterogeneous delays. Taking into account that the RTT distribution exhibits heavy-tailed characteristics [21, 22], this problem is becoming even more serious owing to the fact that a small number of large RTT flows can significantly reduce the system performance. Based on their analyses, Lachlan et al. propose M-XCP, which reduces the control loop gain with setting the control interval d to be the maximum RTT observed during the last interval, instead of the mean RTT used in the original XCP specification, and shows a strong stability in the face of heterogeneous delays. However, M-XCP needs additional operations for each packet, and then the computing complexity of M-XCP increases dramatically. Most importantly, this modification reduces the system responsiveness significantly and makes the system vulnerable to erroneous RTT attacks.

In this paper, according to the robust control theory, with the help of a recently developed Lyapunov–Krasovskii functional [23], an improved version of XCP, named R-XCP, is proposed to solve the weakness of XCP under heterogeneous delays, and the synthesis problem is reduced to a convex optimization scheme expressed in terms of linear matrix inequalities (LMI) [24]. Extensive simulations have shown that R-XCP significantly decreases the volatilities of the aggregate traffic rate and control time interval, and effectively improves the system stability. Meanwhile, R-XCP sets the control interval d to be the mean RTT, rather than the maximum RTT used as in M-XCP, so it has the same control loop gain with XCP, and is more responsive than M-XCP. Additionally, R-XCP decreases the computational complexity dramatically, and makes the system less sensitive to flows, which contribute little traffic but maliciously report their delays, compared with M-XCP.

The rest of the paper is organized as follows. In Sect. 2, some related works are presented. Section 3 briefly summarizes how XCP works and its weakness in the presence of heterogeneous delays. The main motivations and design guidelines for R-XCP are presented in Sects. 4 and 5 respectively. In Sect. 6, extensive simulations are employed to evaluate the performance of R-XCP, and compare it with XCP and M-XCP. Finally, the conclusion is drawn in Sect. 7.

2 Related works

With the rapid advances in the deployment of high-speed links in the Internet, the requirements for a viable replacement of TCP has become increasingly crucial, and many protocols have been proposed to solve this problem, which can be broadly classified into two categories according to primal–dual modeling [25]. Primal congestion control approach refers to the algorithm executed by source end systems for regulating their sending rates, and is widely used in

the current Internet to prevent congestion collapse. TCP and many its variants such as HSTCP [26], STCP [27], HTCP [28], CUBIC [29] and FAST [30] fall into this category, and they are very attractive short-term solutions because the actual barriers to their deployment are very small. However, it is very difficult to achieve high link utilization and fair bandwidth allocation while maintaining small queue length and minimizing packet loss [31].

To address the limitations of pure end-to-end based protocols, active queue management (AQM) [32] was introduced to provide early congestion notification from router sides to end hosts by triggering packet dropping or marking before buffer overflows, which has departed from the strict end-to-end principle. In the past few years, a series of AQM methods, e.g., RED [33], PI [34], BLUE [35], CHOKe [36], AVQ [37], and REM [38] have been proposed, and these AQM schemes can significantly reduce packet loss and achieve some degree of trade-off between queuing delay and flow throughput. However, due to the challenging nature of non-linear and time-varying behaviors in TCP dynamics, these AQM schemes derived from heuristics ideas or linear models fail to provide satisfactory performance and are not able to guarantee stability over a wide range of network scenarios [39,40].

In the traditional primal–dual approach, TCP and AQM are designed independently, but should cooperate with each other, causing system performance is obviously restricted. Therefore, it is a natural choice for us to support joint design. Now, many joint design strategies between intermediate nodes and end systems, such as XCP [41], VCP [42], MLCP [43], RCP [44], JetMax [45] and MaxNet [46] have already been proposed. Unlike the primal-dual TCP/AQM distributed algorithm, a flow does not implicitly probe available bandwidth. Instead, a router fairly allocates the spare bandwidth among all flows that share the same link, and quantitatively informs senders how to adjust their rates. Experimental results have shown that joint design based approaches can achieve better network performance compared to pure end-to-end schemes with the addition of simple queue management algorithms.

With the growth of data volumes and variety of Internet applications, data centers have become an efficient and promising infrastructure, and mixing workloads require small predictable latency with others requiring large sustained throughput. In this environment, today's state-of-the-art TCP protocol fails to satisfy these requirements together within the time boundaries because of impairments such as TCP incast/outcast, buffer pressure and pseudo-congestion effect [47]. And it becomes a new research hotspot. Vasudevan et al. [48] present a practical solution that reducing the minimum retransmission timeout (RTO) from 200 ms to 200 μ s significantly alleviates the problems of TCP in data center networks and improves the overall throughput by

several orders of magnitude. Alizadehzy et al. [49] propose DCTCP, which employs a marking scheme at switches that sets the congestion experienced (CE) codepoint in packets as soon as the buffer occupancy exceeds a fixed pre-determined threshold. DCTCP obviously alleviates TCP incast and outcast problems in data center networks, and it is already implemented in latest versions of Microsoft Windows Server operating system. Unlike DCTCP, ICTCP does not require any modifications at the sender side or network elements such as routers, switches, etc. Instead, ICTCP adaptively adjusts the TCP receive window proactively before packet loss occurs. Experimental results demonstrate that ICTCP is effective in avoiding congestion by achieving almost zero timeouts for TCP incast, and it provides high performance and fairness among competing flows [50]. D2TCP uses a distributed and reactive approach for bandwidth allocation and employs a novel deadline-aware congestion avoidance algorithm to vary the sender's congestion window. D2TCP reduces the fraction of missed deadlines up to 75% as compared to DCTCP [51]. Timeouts lead to dramatic degradation in the network performance and affect the user perceived delay. TCP with guarantee important packets (GIP) can avoid almost all of timeouts and achieve higher goodput for applications with the incast communication pattern [52]. By detecting the unusual spikes, accurately measuring RTT and proper management of RTO, PVTCP is much more effective in addressing incast congestion in virtualized data centers than standard TCP, with requiring no modification to the hypervisor [53]. CONGA splits TCP flows into flowlets, estimates real-time congestion on fabric paths, and allocates flowlets to paths based on feedback from remote switches. This enables CONGA to efficiently balance load and seamlessly handle asymmetry, without requiring any TCP modifications in incast scenarios [54]. Chen et al. [55] builds up an interpretive model which emphasizes particularly on describing qualitatively how various factors affect network performances in incast traffic pattern. With this model, it gives plausible explanations why the various solutions for TCP incast problem can help, but do not solve it entirely.

3 XCP control laws and problem statement

3.1 XCP control laws

In the XCP algorithm, routers explicitly inform sources how to adjust their sending rates. To enable this communication between the network and sources, XCP introduces a congestion header, which allows every packet to carry information about the flow to which the packet belongs, that contains three fields, the sender's current congestion window size $cwnd$, the estimated round trip time rtt , and the router feedback field

feedback. Each sender fills its current $cwnd$ and rtt values into the congestion header on a packet departure, and initializes the $feedback$ field to its desired window size.

The core control laws of XCP are implemented at routers. XCP introduces the new concept of decoupling utilization control from fairness control [4]. Each router has two logical controllers, efficiency controller and fairness controller. In each control interval, the efficiency controller calculates the amount of spare bandwidth φ that to be distributed among the flows. The spare bandwidth φ is given by,

$$\varphi = \alpha(c - y) - \beta \frac{q}{d} \quad (1)$$

where c is the link capacity, y is the aggregate input traffic rate, and q is the persistent queue length. d is the control interval, α and β are 0.4 and 0.226 respectively, that are chosen to make the system stable [4]. To enable high efficiency, the efficiency controller follows the multiplicative increase multiplicative decrease (MIMD) rule, that allows fast adaptation of y to c .

After calculating φ , the fairness controller decides how to distribute φ among all the flows. The fairness controller employs an additive increase multiplicative decrease (AIMD) scheme for this bandwidth allocation, distributing φ equally among all flows when φ is positive, and proportionally to each flow rate when φ is negative. When a link is in the high utilization region, resulting in $\varphi \approx 0$, which would disable bandwidth redistribution for new data flows. To enable fair redistribution of bandwidth, even when $\varphi \approx 0$, bandwidth shuffling is performed at each control interval. This operation simultaneously allocates and deallocates the shuffled bandwidth among flows. The shuffled bandwidth is computed as follows,

$$h = \max(0, \gamma y - |\varphi|) \quad (2)$$

where γ is a control parameter with default value 0.1.

3.2 XCP stability problem under heterogeneous delays

Firstly, we intuitively describe XCP stability problem when the ratio of maximum to average RTT is too large, using the simple single bottleneck topology shown in Fig. 1. In this sce-

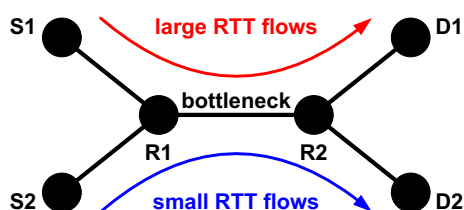


Fig. 1 A simple single bottleneck topology

nario, there are nine small RTT flows (flow i_1, \dots, i_9 , $RTT_i = 20$ ms) and only one large RTT flow (flow j_1 , $RTT_j = 1000$ ms) that go through a 100 Mbps bottleneck link. In an ideal case, with a max–min fair allocation, each of the flows gets a bandwidth allocation of 10 Mbps, the aggregate traffic arriving at the bottleneck link and the control interval d will stabilize at 100 Mbps and 118 ms respectively. However, actual simulation results in Fig. 2 show that the large RTT flow cannot obtain its max–min allocation as expected, the aggregate traffic and control interval fluctuate dramatically, therefore, a big buffer is required to absorb these oscillations.

Lachlan et al. first pinpoint that XCP is not globally stable but locally stable in the face of heterogeneous delays, and believe that the instability of XCP is largely due to the nonlinearity encountered when the traffic rate is far from the equilibrium value. Considering that the RTT distribution exhibits heavy-tailed characteristics, this problem is becoming even more serious owing to the fact that a small number of large RTT flows can significantly reduce the system performance. Therefore, this is an urgent problem to be solved or alleviated for making the XCP algorithm more practical. Lachlan et al. [20] further proposes M-XCP, an improved version of XCP, which sets the control interval d to be the maximum RTT observed during the last cycle, instead of the average RTT, and shows a strong stability under heterogeneous delays. However, M-XCP needs additional operations for each packet, the computational complexity of M-XCP increases dramatically, as shown in Sect. 5. Most importantly, M-XCP experiences sluggish system responsiveness. Figure 3 shows that when the control interval d increases, the instability of XCP is removed at the cost of becoming less responsive. Therefore, increasing the value of parameter d is not a good solution, and in-depth theoretical analysis will be given in Sect. 4.

4 Linear analysis and motivation

The dynamic model of XCP behavior was developed in [4]. This model is described by the following coupled, nonlinear differential equations,

$$\dot{y}(t) = -\frac{\alpha}{d}(y(t-d) - c) - \frac{\beta}{d^2}q(t-d) \quad (3)$$

$$\dot{q}(t) = y(t) - c \quad (4)$$

where c is the link capacity, $y(t)$ is the aggregate traffic rate, $q(t)$ is the persistent queue length, and d is the control interval. α and β are system parameters, that are chosen to make the system stable.

In order to analyze the stability of XCP, Katabi et al. [4] employ a variable change $z(t) = y(t) - c$ to yield a linear system independent of c , and the open-loop transfer function of

Fig. 2 XCP behaves in a noticeably unstable manner when the ratio of maximum to mean RTT is large. The large RTT flow cannot obtain its max–min allocation of 10Mbps, the aggregate traffic, queue length and control interval fluctuate dramatically, at the 100Mbps bottleneck

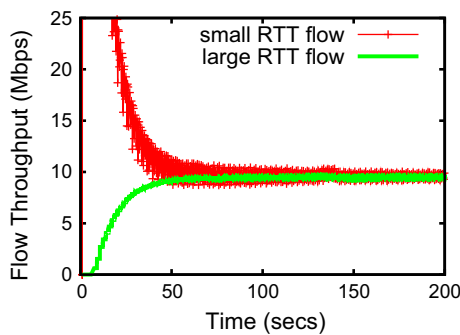
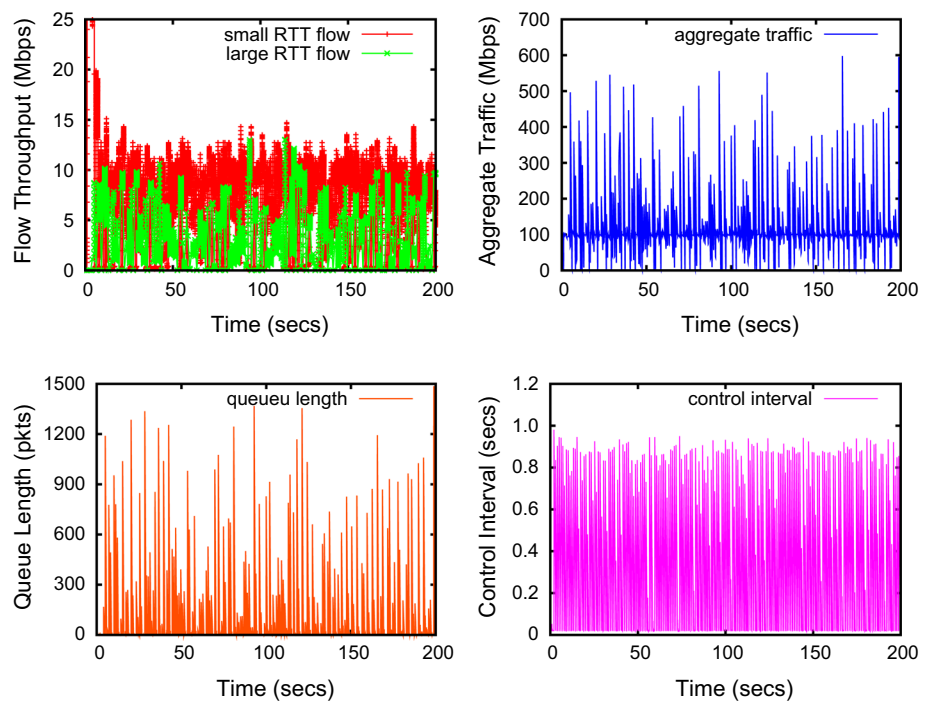


Fig. 3 M-XCP converges to the equilibrium rate at around 50s, which is a long convergence time

its corresponding linearized XCP model can be represented as follows, where $K_1 = \frac{\alpha}{d}$ and $K_2 = \frac{\beta}{d^2}$.

$$G(s) = \frac{K_1 \cdot s + K_2}{s^2} e^{-ds} \tag{5}$$

Further, we can easily get the XCP closed-loop sensitivity function,

$$S(s) = \frac{s^2}{s^2 + K_1 \cdot s \cdot e^{-ds} + K_2 \cdot e^{-ds}} \tag{6}$$

where $K_1 = \frac{\alpha}{d}$, $K_2 = \frac{\beta}{d^2}$, $\beta = \alpha^2 \sqrt{2}$, $0 < \alpha < \pi/(4\sqrt{2})$ and d is the control interval. With the help of the XCP closed-loop sensitivity function, robustness and responsiveness of the XCP system can be quantitatively analyzed, deduced from its frequency response. Roughly speaking,

robustness is inversely proportional to the peak frequency response $\|S\|_\infty = \sup_\omega |S(j\omega)|$, while responsiveness is proportional to the unity gain crossover frequency ω_g defined by $|S(j\omega_g)| = 1$, where S is the XCP closed-loop sensitivity function [56]. Since the linearized XCP system is independent of capacity c , robustness and responsiveness are not affected by the link capacity. Thereby, XCP is very suitable for high-speed networks, which has been proven by simulations and experiments [4,5].

According to Eq. (6), robustness and responsiveness are primarily affected by system parameter α ($\beta = \alpha^2 \sqrt{2}$) and control interval d . In order to insure the system stability, Katabi et al. [4] propose a constraint, $0 < \alpha < \pi/(4\sqrt{2}) \approx 0.55$. Due to control interval d is defined as average round trip time of flows passing through the bottleneck link, and vast majority of flows (over 90%) have RTT less than 200 ms [57], we choose $0 < d < 1.0$ s. On the basis of these constraints, we quantitatively study their impacts, and plot the robustness and responsiveness as a function of parameter $\alpha \in (0, 0.55)$ and control interval $d \in (0, 1.0)$ in Figs. 4 and 5, respectively.

Figure 4 shows that XCP responsiveness increases while robustness decreases, when parameter α increases. In original XCP design, α is set to 0.4, which is an empirical choice. According to Fig. 4, we can see that $\alpha = 0.4$ is a good performance trade-off between robustness and responsiveness. Figure 5 shows that XCP responsiveness decreases dramatically, when control interval d increases. The robustness is almost completely unaffected by control interval d , which is a surprising result. However, Lachlan et al. [20] believe

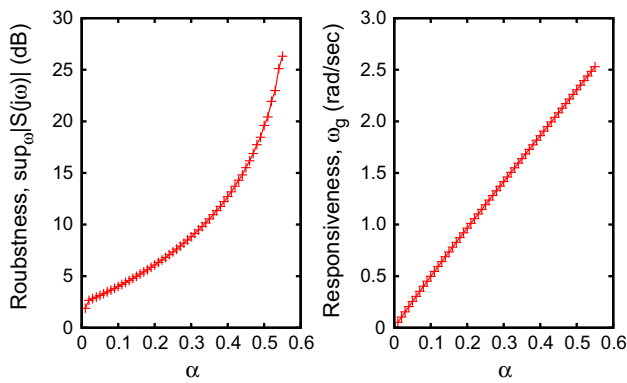


Fig. 4 When system parameter α increases, XCP becomes less robust (robustness is inversely proportional to the vertical axis in the *left* figure), but more responsive (responsiveness is proportional to the vertical axis in the *right* figure)

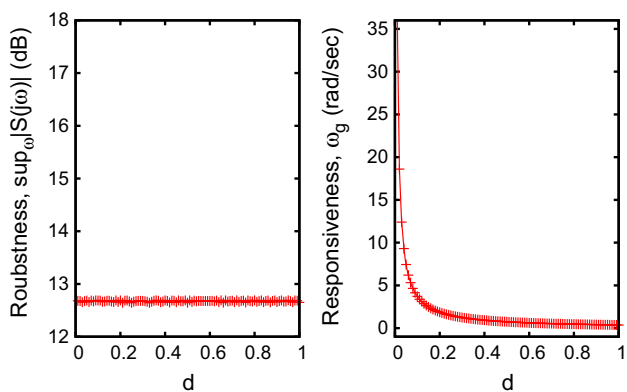


Fig. 5 When control interval d increases, XCP becomes less responsive, but robustness is almost completely unaffected by control interval d

that increasing control interval d can improve the system stability, and demonstrate that XCP is locally stable but globally unstable using NS-2 simulations. This is because XCP is essentially a nonlinear system [15], the linear model is inadequate, and a discrete-time nonlinear system model is required. However, such a nonlinear model is not amenable to straightforward theoretical analysis that is possible with a linear system.

Comparing Figs. 4 and 5, we find that system robustness is sensitive to parameter α but not sensitive to control interval d , and in contrast system responsiveness is sensitive to control interval d but not sensitive to parameter α . Therefore, it is natural for us to decrease parameter α for improving system robustness against heterogeneous delays. However, the Internet is a highly dynamic and evolving system of large scale and complex, it is a challenge to choose a reasonable value of α for each link in such environments. In this paper, according to the robust control theory and Lyapunov–Krasovskii functional, we propose an effective mechanism to adjust parameter α from an initial value of 0.4 to a reasonable value to improve

system robustness against heterogeneous delays, without significantly affecting system responsiveness at the same time.

5 R-XCP protocol

5.1 A modification on XCP system

Experience tells us that congestion control algorithms should be more robust against unpredictable scenarios when aggregate traffic rate $y(t)$ exceeds link capacity c . Corresponding to the XCP system, parameter α should be decreased when aggregate traffic rate $y(t)$ fluctuates dramatically. More importantly, the control objective is to make $y(t) = c$, therefore, it is a good choice to choose $z(t) = y(t) - c$ as the controlled variable. Additionally, Zhang et al. [15] believe that capacity c in XCP efficiency controller, expressed by Eq. (1), is a configurable variable. Therefore, we modify the XCP algorithm by introducing control variable $u(t)$, and expressed via the following nonlinear differential equations,

$$\dot{y}(t) = -\frac{\alpha}{d}(y(t-d) - c - u(t)) - \frac{\beta}{d^2}q(t-d) \tag{7}$$

$$\dot{q}(t) = y(t) - c \tag{8}$$

Hence, we can dynamically change parameter α for improving system robustness on the basis of $z(t) = y(t) - c$. The revised linearized XCP model is depicted in Fig. 6. Please note that, it is not reasonable to disconnect control path and introduce control variable $u(t)$ at these places like place **a** or **b** in Fig. 6. At place **a**, the rate of queue change will become $\dot{q}(t) = y(t) - c - u(t)$, not being consistent with actual physical system, which is $\dot{q}(t) = y(t) - c$. In order to analyze the stability of XCP, Katabi et al. [4] ignore the boundary conditions on the queue length $q(t)$, and assumes that $q(t)$ can range from $-\infty$ to $+\infty$. In practice, $q(t)$ always has an upper bound of Q , which is the router buffer size, and never goes negative [15]. When the feedback is bounded, the instability caused by this non-linearity is apparent [58]. So it is not reasonable to disconnect control path at place **b** too.

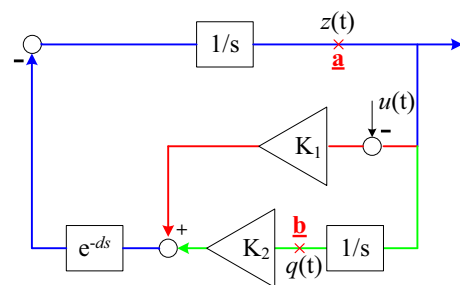


Fig. 6 Block diagram of the revised XCP system

5.2 Lyapunov–Krasovskii based controller design

The revised XCP system can be rewritten as follows,

$$\dot{x}(t) = Ax(t) + A_d x(t - d) + Bu(t) \tag{9}$$

where $z(t) = y(t) - c$, $A = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$, $A_d = \begin{bmatrix} -\frac{\alpha}{d} & -\frac{\beta}{d^2} \\ 0 & 0 \end{bmatrix}$, $B = \begin{bmatrix} \frac{\alpha}{d} \\ 0 \end{bmatrix}$. $x(t) = \begin{bmatrix} z(t) \\ q(t) \end{bmatrix}$ is the state vector. $x_0(\theta) = \varphi(\theta)$, $\theta \in [-d, 0]$ is the initial condition.

Since all states are available, we construct the following memoryless state feedback,

$$u(t) = Kx(t - d) \tag{10}$$

to control system (9). K is the designed feedback gain matrix, corresponding to our controller.

In order to guarantee the asymptotic stability of system (9) under controller (10) for any delay $d > 0$, we applied Lyapunov–Krasovskii functional approach and LMI method to design a suitable feedback gain matrix K .

Theorem 1 System (9) is stabilized by controller (10) for any delay $d > 0$, if there are symmetric positive definite matrices U , V and a matrix X , which satisfy the following matrix inequality,

$$\begin{bmatrix} UA^T + AU + V & A_d U + BX \\ UA_d^T + X^T B^T & -V \end{bmatrix} < 0 \tag{11}$$

and meanwhile the state feedback gain is given by $K = XU^{-1}$. □

Proof we propose to employ the well-known Lyapunov–Krasovskii functional,

$$V(x_t) = x^T(t)Px(t) + \int_{t-d}^t x^T(\theta)Qx(\theta)d\theta$$

where matrices P and Q are symmetric and positive definite. The time derivative of this function along trajectory of system (9) under controller (10) is given by,

$$\begin{aligned} \dot{V}(x_t) &= \dot{x}^T(t)Px(t) + x^T(t)P\dot{x}(t) + x^T(t)Qx(t) - x^T(t-d)Qx(t-d) \\ &= x^T(t)[A^T P + PA + Q]x(t) + x^T(t)[PA_d + PBK]x(t-d) \\ &\quad + x^T(t-d)[A_d^T P + K^T B^T P]x(t) - x^T(t-d)Qx(t-d) \\ &= \begin{bmatrix} x(t) \\ x(t-d) \end{bmatrix}^T \begin{bmatrix} A^T P + PA + Q & PA_d + PBK \\ A_d^T P + K^T B^T P & -Q \end{bmatrix} \begin{bmatrix} x(t) \\ x(t-d) \end{bmatrix} \end{aligned}$$

According to the Lyapunov–Krasovskii stability theorem [23], system (9) is stabilized by controller (10) for any delay

$d > 0$ if the following inequality holds,

$$\begin{bmatrix} A^T P + PA + Q & PA_d + PBK \\ A_d^T P + K^T B^T P & -Q \end{bmatrix} < 0$$

Further, the left side of above matrix inequality left and right multiplied by $diag\{P^{-1}, P^{-1}\}$ respectively makes,

$$\begin{bmatrix} P^{-1}A^T + AP^{-1} + P^{-1}QP^{-1} & A_d P^{-1} + BK P^{-1} \\ P^{-1}A_d^T + P^{-1}K^T B^T & -P^{-1}QP^{-1} \end{bmatrix} < 0$$

Let $U = P^{-1}$, $V = P^{-1}QP^{-1}$, $X = KU$, we can obtain,

$$\begin{bmatrix} UA^T + AU + V & A_d U + BX \\ UA_d^T + X^T B^T & -V \end{bmatrix} < 0$$

and meanwhile the state feedback gain is given by $K = XU^{-1}$. □

5.3 A better controller

In this subsection, our goal is to design a better controller which takes into account the upperbound of parameter d , to make the system get better performance.

Let us suppose that d has the lowerbound $h_1 \in R^+$, namely, $h_1 \leq d$, yields $\frac{1}{d} \leq \frac{1}{h_1}$. Let $H_1 = \frac{1}{h_1}$, matrix A_d can be written as follows,

$$A_d = DFE_a \tag{12}$$

where $D = \begin{bmatrix} -\alpha H_1 & 0 & 0 & -\beta H_1^2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$, $F = \begin{bmatrix} \frac{1}{dH_1} I_2 & 0 \\ 0 & \frac{1}{dH_1^2} I_2 \end{bmatrix}$,

$E_1 = \begin{bmatrix} I_2 \\ I_2 \end{bmatrix}$, I_n denotes a unit matrix of n order.

Then, the following inequality is clearly established,

$$F^T F < I_4 \tag{13}$$

Since all states are measurable, we construct memoryless state feedback $u(t) = Kx(t - d)$ to control system (9). Then, we have,

$$\dot{x}(t) = Ax(t) + \bar{A}_d x(t - d) \tag{14}$$

where $\bar{A}_d = A_d + BK$.

Lemma 1 [24] Let define proper matrices D , E and F , if $F^T(t)F(t) \leq I$ is correct, then, for any scalar $\varepsilon > 0$, there exists,

$$DEF + E^T F^T D^T \leq \varepsilon DD^T + \varepsilon^{-1} E^T E \tag{15}$$

□

Using an information on parameter d , we expect a better controller, and propose Theorem 2.

Theorem 2 For $d \in [h_1, h_2]$, $h_2 > h_1 > 0$, and scalars $\mu_i (i = 2, 3)$, if there exists $\varepsilon > 0$, matrices $\tilde{P} > 0$, $\tilde{R} > 0$, Y , $\tilde{N}_i (i = 1, 2, 3)$, and a nonsingular matrix X , to make LMI (16) is established (* denotes a matrix block given by matrix symmetry),

$$\begin{bmatrix} \tilde{\mathcal{E}}_{11} & * \\ \tilde{\mathcal{E}}_{21} & \tilde{\mathcal{E}}_{22} \end{bmatrix} < 0 \tag{16}$$

where

$$\begin{aligned} \tilde{\mathcal{E}}_{11} &= \begin{bmatrix} \Gamma_{11} & * & * \\ \Gamma_{21} & \Gamma_{22} & * \\ \Gamma_{31} & \Gamma_{32} & \Gamma_{33} \end{bmatrix} \\ \tilde{\mathcal{E}}_{21} &= \begin{bmatrix} h_2 \tilde{N}_1^T & h_2 \tilde{N}_2^T & h_2 \tilde{N}_3^T \\ 0 & \mu E_a \tilde{X}^T & 0 \end{bmatrix} \\ \tilde{\mathcal{E}}_{22} &= \text{diag}\{-h_2 \tilde{R}, -\mu \varepsilon I\} \\ \Gamma_{11} &= \tilde{N}_1 + \tilde{N}_1^T + AX^T + XA^T + \varepsilon DD^T \\ \Gamma_{21} &= \tilde{N}_2 - \tilde{N}_1^T + \mu_2 AX^T + Y^T B^T \\ \Gamma_{31} &= \tilde{N}_3 + \mu_3 AX^T - X + \tilde{P}^T \\ \Gamma_{22} &= -\tilde{N}_2 - \tilde{N}_2^T + \mu_2 BY + \mu_2 Y^T B^T + \mu_2 \varepsilon DD^T \\ \Gamma_{32} &= -\tilde{N}_3 - \mu_2 X + \mu_3 BY \\ \Gamma_{33} &= -\mu_3 X - \mu_3 X^T + h_2 \tilde{R} + \mu_3 \varepsilon DD^T \\ \mu &= 1 + \mu_2 + \mu_3 \end{aligned}$$

Then, system (9) will be stabilized with a state feedback $u(t) = Kx(t - d)$ for $d \in [h_1, h_2]$. Meanwhile, the state feedback gain is obtained by $K = YX^{-T}$. □

Proof we propose to employ a novel Lyapunov–Krasovskii functional,

$$V(x_t) = x^T(t)Px(t) + \int_{t-h_2}^t \int_s^t \dot{x}^T(v)R\dot{x}(v)dv ds$$

where $P > 0$ and $R > 0$. By taking the derivative of $V(x_t)$, system (14) and $x(t) - x(t - d) - \int_{t-d}^t \dot{x}(s)ds = 0$ (Newton–Leibniz Equation [24]) are used, we have,

$$\begin{aligned} \dot{V}(x_t) &= 2x^T(t)P\dot{x}(t) \\ &\quad + 2[x^T(t)N_1 + x^T(t - d)N_2 + \dot{x}^T(t)N_3] \cdot \\ &\quad [x(t) - x(t - d) - \int_{t-d}^t \dot{x}(s)ds] \\ &\quad + 2[x^T(t)M_1 + x^T(t - d)M_2 + \dot{x}^T(t)M_3] \cdot \\ &\quad [Ax(t) + \bar{A}_d x(t - d) - \dot{x}(t)] \\ &\quad + \lambda \dot{x}^T(t)R\dot{x}(t) - \int_{t-\lambda}^t \dot{x}^T(s)R\dot{x}(s)ds \end{aligned}$$

where N_i and $M_i (i = 1, 2, 3)$ are the free matrixes with corresponding dimensions.

Obviously, the following inequalities are established, where $\zeta^T(t) = [x^T(t), x^T(t - d), \dot{x}^T(t)]$ and $N^T = [N_1^T, N_2^T, N_3^T]$.

$$\begin{aligned} & - \int_{t-h_2}^t \dot{x}^T(s)R\dot{x}(s)ds \leq - \int_{t-d}^t \dot{x}^T(s)R\dot{x}(s)ds \\ & - 2[x^T(t)N_1 + x^T(t - d)N_2 + \dot{x}^T(t)N_3] \int_{t-h_2}^t \dot{x}(s)ds \\ & \leq h_2 \zeta^T(t)NR^{-1}N^T \zeta(t) + \int_{t-h_2}^t \dot{x}^T(s)R\dot{x}(s)ds \end{aligned}$$

According to the above two inequalities, the following inequality is obtained,

$$\dot{V}(x_t) \leq \zeta(t)^T \left\{ \begin{bmatrix} \Pi_{11} & * & * \\ \Pi_{21} & \Pi_{22} & * \\ \Pi_{31} & \Pi_{32} & \Pi_{33} \end{bmatrix} + \Theta \right\} \zeta(t)$$

where

$$\begin{aligned} \Pi_{11} &= N_1 + N_1^T + M_1 A + A^T M_1^T \\ \Pi_{21} &= N_2 - N_1^T + M_2 A + K^T \bar{A}_d^T M_1^T \\ \Pi_{31} &= N_3 + M_3 A - M_1^T + P^T \\ \Pi_{22} &= -N_2 - N_2^T + M_2 \bar{A}_d + K^T \bar{A}_d^T M_2^T \\ \Pi_{32} &= -N_3 - M_2^T + M_3 \bar{A}_d \\ \Pi_{33} &= -M_3 - M_3^T + h_2 R \\ \Theta &= h_2 N R^{-1} N^T \end{aligned}$$

By setting $\Omega = \begin{bmatrix} \Pi_{11} & * & * \\ \Pi_{21} & \Pi_{22} & * \\ \Pi_{31} & \Pi_{32} & \Pi_{33} \end{bmatrix} + \Theta$, if inequality $\Omega < 0$ exists, system (14) will be stabilized based on Lyapunov–Krasovskii stability theorem.

$\Omega < 0$ is equivalent to the following inequality,

$$\Omega = \tilde{\Omega} + M_d^T F M_e + M_e^T F^T M_d < 0 \tag{17}$$

where

$$\begin{aligned} \tilde{\Omega} &= \begin{bmatrix} \Omega_{11} & * & * & * \\ \Omega_{21} & \Omega_{22} & * & * \\ \Omega_{31} & \Omega_{32} & \Omega_{33} & * \\ h_2 N_1^T & h_2 N_2^T & h_2 N_3^T & -h_2 R \end{bmatrix} \\ \Omega_{11} &= N_1 + N_1^T + M_1 A + A^T M_1^T \\ \Omega_{21} &= N_2 - N_1^T + M_2 A + K^T B^T M_1^T \\ \Omega_{31} &= N_3 + M_3 A - M_1^T + P^T \\ \Omega_{22} &= -N_2 - N_2^T + M_2 B K + K^T B^T M_2^T \\ \Omega_{32} &= -N_3 - M_2^T + M_3 B K \end{aligned}$$

$$\begin{aligned}\Omega_{33} &= -M_3 - M_3^T + h_2 R \\ M_d &= [D^T M_1^T \quad D^T M_2^T \quad D^T M_3^T \quad 0] \\ M_e &= [0 \quad E_a \quad 0 \quad 0]\end{aligned}$$

Let $M = M_1$, $M_2 = \mu_2 M$, $M_3 = \mu_3 M$, $X = M^{-1}$, $Y = KX^T$, $\tilde{P} = XPX^T$, $\tilde{N}_i = XN_iX^T$, $\tilde{M}_i = XM_iX^T$, $\tilde{R} = XRX^T$, and the left side of inequality (17) left and right multiplied by $\text{diag}\{X, X, X, X\}$ respectively, inequality (16) can be get based on Lemma 1 and Schur Complement Lemma. Thereby, system (9) is asymptotic stable under the control law (10) for $d \in [h_1, h_2]$, with the feedback gain $K = YX^{-T}$. \square

As a numerical example, using Matlab LMI Toolbox [59], when $d \in [0.1, 1.0]$, the state feedback gain $K = [-6.3707, -0.0001]$ was obtained from Theorem 2.

5.4 Implementation

Like XCP, each R-XCP packet carries a congestion header, which is used to communicate a flow's state to routers and feedback from the routers on to the receiver. The field H_cwnd is the sender's current congestion window, whereas H_rtt is the sender's current RTT estimated value. These are filled in by the sender and never modified in transit. The remaining field, $H_feedback$, takes positive or negative values and is initialized by the sender. Routers along the path modify this field to directly control the sender's congestion windows. pkt_size is the length of a packet.

Then, implementing an R-XCP router is fairly simple and can be described using the following pseudo code. There are three relevant blocks of code. The first block, which is the same as XCP, is executed at the arrival of a packet and involves updating the estimates (e.g., input_bytes is the number of input bytes in a control cycle. sum_rtt_by_cwnd and $\text{sum_rtt_square_by_cwnd}$ are two parameters, and their detailed meanings can be found in [4]) maintained by the router.

Algorithm 1: R-XCP Pseudocode 1 [4]

```
// on packet arrival do
1 input_bytes+ = pkt_size
2 sum_rtt_by_cwnd+ = H_rtt * pkt_size / H_cwnd
3 sum_rtt_square_by_cwnd+ =
  H_rtt * H_rtt * pkt_size / H_cwnd
```

The second block is executed when the control timer fires. It involves updating our control variables, reinitializing the estimation variables, and rescheduling the timer. Code lines 1–4 describe the efficiency controller, of which purpose is to maximize link utilization while minimizing drop rate and

persistent queues. They are the core of the R-XCP algorithm, where $y = \text{input_traffic}$ is the aggregate input traffic rate, $d = \text{avg_rtt}$ is the control interval, $c = \text{capacity}$ is the link capacity, and q is the persistent queue length. u is R-XCP's control variable, ε_1 and ε_2 are the feedback gains, derived from theoretical result $K = [\varepsilon_1, \varepsilon_2]$, and they both have negative values. φ is the amount of spare bandwidth in a control cycle that to be distributed among the flows. α and β are 0.4 and 0.226 respectively, that are chosen to make the system stable [4].

As can be seen from line 3 and 4, when the control timer expires, R-XCP dynamically adjusts parameter α online, based on spare bandwidth and persistent queue length. When the link is under-utilized, we have $(\text{input_traffic} - \text{capacity}) < 0$ and $q = 0$, due to $\varepsilon_1 < 0$ and $\varepsilon_2 < 0$, we have control variable $u > 0$, R-XCP indirectly increases parameter α , in turn increasing system responsiveness with the cost of robustness. When the link is congestion, we have $(\text{input_traffic} - \text{capacity}) > 0$ and $q > 0$, due to $\varepsilon_1 < 0$ and $\varepsilon_2 < 0$, we have control variable $u < 0$, R-XCP indirectly decreases parameter α , in turn increasing system robustness.

Code lines 5–9 calculate the control variables of the fairness controller. With Algorithm 3, they jointly implement a complete fairness controller, apportioning the feedback to individual packets to achieve fairness. Since R-XCP does not modify the XCP's fairness controller, the detailed explanation of the course of fairness computing and involved parameters' meanings are omitted from this paper, and can be found in [4].

Algorithm 2: R-XCP Pseudocode 2

```
// on estimation-control timeout do
1 input_traffic = input_bytes / avg_rtt
2 avg_rtt = sum_rtt_square_by_cwnd / sum_rtt_by_cwnd
3 u = \varepsilon_1 * (input_traffic - capacity) + \varepsilon_2 * q
4 \varphi = \alpha * avg_rtt * (capacity + u - input_traffic) - \beta * q
5 shuffled_traffic = 0.1 * input_traffic
6 \xi_p = ((max(\varphi, 0) + shuffled_traffic) / (avg_rtt *
  sum_rtt_by_cwnd))
7 \xi_n = ((max(-\varphi, 0) + shuffled_traffic) / (avg_rtt *
  input_traffic))
8 residue_pos_fbk =
  (max(\varphi, 0) + shuffled_traffic) / avg_rtt
9 residue_neg_fbk =
  (max(-\varphi, 0) + shuffled_traffic) / avg_rtt
10 input_bytes = 0
11 sum_rtt_by_cwnd = 0
12 sum_rtt_square_by_cwnd = 0
13 timer.schedule(avg_rtt)
```

The third block of code involves computing the feedback and is executed at packets' departure.

Algorithm 3: R-XCP Pseudocode 3 [4]

```

// on packet departure do
1  $pos\_fbk = \xi_p \times H\_rtt \times H\_rtt \times pkt\_size / H\_cwnd$ 
2  $neg\_fbk = \xi_n \times H\_rtt \times pkt\_size$ 
3  $feedback = pos\_fbk - neg\_fbk$ 
4 if ( $H\_feedback \geq feedback$ )
5    $H\_feedback = feedback$ 
6    $residue\_pos\_fbk- = pos\_fbk / H\_rtt$ 
7    $residue\_neg\_fbk- = neg\_fbk / H\_rtt$ 
8 else
9    $residue\_pos\_fbk- = H\_feedback / H\_rtt$ 
10   $residue\_neg\_fbk- = (feedback - H\_feedback) / H\_rtt$ 

```

5.5 Complexity analysis

Like M-XCP, an R-XCP router implementation also has three relevant code blocks, which are executed when a packet arrives, when the control timer expires, and when a packet departs. The statements in each code block are all executed in sequence order, without loop operations. Therefore, each code block has constant time complexity of $O(1)$, and the specific time consumption depends upon the number of statements in each code block. Meanwhile, the R-XCP algorithm uses a small and fixed amount of space which doesn't depend on the input. Specifically, the input is not taken into account and R-XCP takes the same and constant amount of space for global variables. Therefore, the R-XCP algorithm also has constant space complexity of $O(1)$.

However a detailed analysis found that R-XCP only modifies the second block, i.e., control timer block. It is executed after each control interval d , which is the average RTT of all flows passing through the router. Internet measurement reports that average RTT is roughly 200 ms [57], so the increase of operations is negligible. In contrast, M-XCP requires to operate each packet to observe the maximum RTT during a control interval. Table 1 gives two examples, considering the mean packet size is 400 bytes [60], there are two bottleneck links, each with capacity of 1 and 10 Gbps, the increased operations of M-XCP are at least 312,500 and 3,125,000 ops/s comparing operations respectively (XCP is selected as a common basis for comparison). Under the same scenario, the increased operations of R-XCP are independent of link capacity, and maintains the same level, 5 ops/s. Compared with M-XCP, the increased operations of R-XCP are negligible. Therefore, R-XCP is more suitable for high-speed networks.

Table 1 Additional operations compared with XCP

Protocols (Gbps)	R-XCP (ops/s)	M-XCP (ops/s)
1	5	312,500
10	5	3,125,000

6 Performance evaluation**6.1 Effectiveness validation**

First of all, we validate that R-XCP is effective in solving the weakness of XCP when the ratio of maximum to mean RTT is too large, using the same scenario described in Sect. 3. There are nine small RTT flows (flow i_1, \dots, i_9 , $RTT_i = 20$ ms) and only one large RTT flow (flow j_1 , $RTT_j = 1000$ ms) that cross an 100 Mbps bottleneck link. Simulation results in Fig. 7 confirm that R-XCP effectively stabilizes the system, each flow obtains its max-min bandwidth allocation of 10 Mbps, aggregate traffic rate and control interval stabilize at 100 Mbps and 205 ms respectively. Even though fluctuations of per-flow throughput, aggregate traffic rate and control interval are larger than that of M-XCP, however, the convergence time of R-XCP decreases significantly, see Figs. 3 and 7. Further simulations display that R-XCP maintains the same convergence speed ($< 1.0s$) and M-XCP's convergence time increases continuously, when the bandwidth of a bottleneck link increases. In other words, R-XCP achieves a better balance between robustness and responsiveness, compared with M-XCP. Additionally, control interval d in R-XCP is slightly larger than the average RTT, this is because when parameter α decreases, system responsiveness decreases slightly, and router buffer is cleared slowly, eventually leading to increased queuing delays and increased cycle length.

The reasons of simulation results in Figs. 2 and 7 can be further understood by looking in more detail at the congestion window ($cwnd$) behavior of each flow. The enlarged view of the flow's $cwnd$ in Fig. 8 shows that the source of dramatical rate fluctuation is the rapid changes of instantaneous congestion window size, which may significantly differ from the average congestion window over one round trip time, computed by the XCP controller.

In the XCP algorithm, the large RTT flow's $cwnd$ is highly peaked, and the small RTT flow's $cwnd$ also changes violently. The reasons can be explained as follows, when the bottleneck congestion occurs, the XCP controller reduces each flow's rate. Since a large RTT flow has larger congestion window, with more packets in transit, it obtains greater negative feedback, decreasing $cwnd$ significantly. Moreover, when the window size drops below the number of packets outstanding, large RTT flow's $cwnd$ also becomes zero and stays zero, based on the sliding window method [4]. It is annoying that this peaky behavior is self-sustaining. When the XCP controller finds the link inefficiency, it enlarges each flow's $cwnd$ at once, leading to a large peak in the queue size. Subsequently, feedbacks become very negative.

This phenomenon can be alleviated by making XCP respond less rapidly to periods of high or low throughput, such as R-XCP by indirectly decreasing α for improving system robustness against heterogeneous delays. As shown

Fig. 7 R-XCP is effective in stabilizing XCP when the ratio of maximum to mean RTT is too large

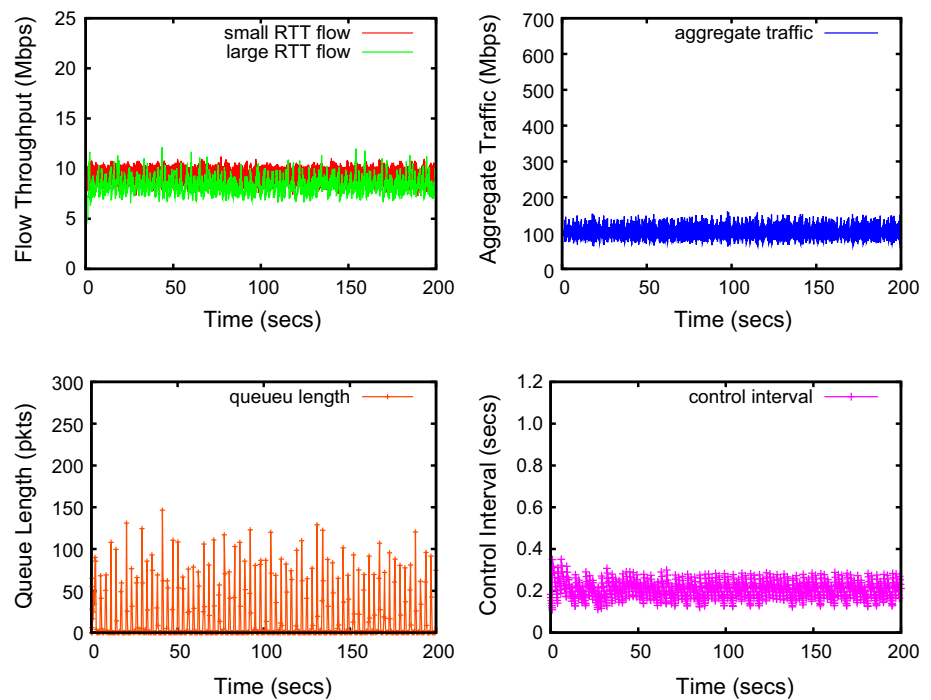
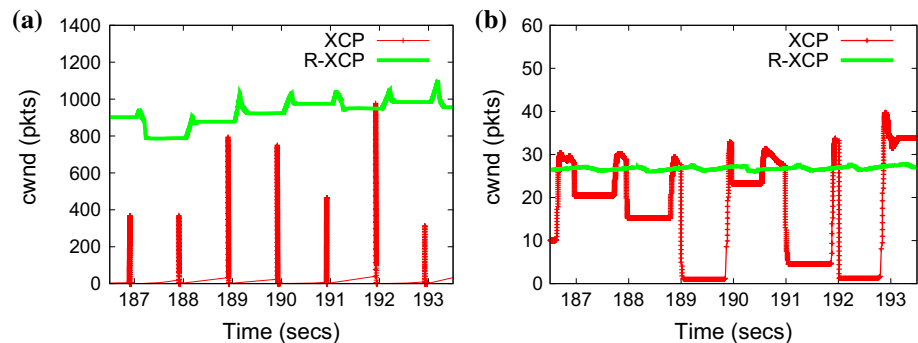


Fig. 8 The enlarged view of the congestion windows of large RTT flow and small RTT flow, **a** large RTT flow, **b** small RTT flow



in Fig. 8, R-XCP flow's *cwnd* is quite smooth, and generally maintains at steady state value, whether it is large RTT flow or small RTT flow.

Next, extensive NS-2 simulations are used to study the performance of R-XCP for a wide range of network scenarios including varying round trip times of large RTT flows in range [20, 1000ms], share of large RTT flows in range [0.1, 0.9], and link capacities in range [10, 1000Mbps]. We evaluate the impact of each network parameter in isolation while retaining the others as basic settings, which are as follows, the round trip times of small RTT flows are fixed at 20ms. The bottleneck buffer size is set to one bandwidth-delay product. The data packet size is 1000 bytes, while the ACK packet is 40 bytes. To investigate R-XCP's performance in the presence of variability and burstiness caused by the short-lived, web-like flows arrivals, short-lived traffic are added into the network. These flows arrive according to a Poisson process with an average arrival rate 500/s. The

transfer size of these flows is derived from a Pareto distribution with an average of 30 packets, which is consistent with real-world web traffic [61]. Another important issue is seed selection. In order to fully reflect the uncertain behavior of data flows and packets, the parameter defaultRNG value is set to 0. According to NS-2 implementation, the seed value will be changed based on clock and counter, which is a complex problem and is beyond the scope of this paper. Basically, the simulation results tend to be different for every run [62]. However, this non-deterministic behavior has no impact on the statistical characteristics of test items in this paper, this conclusion can be deduced from Sects. 6.2–6.4. Simulation parameters are summarized in Table 2.

In terms of congestion control algorithms, stability is frequently associated with oscillation of per-flow throughput, and measuring rate variations of flows is often used to measure the stability of transport protocols [63–65]. In this paper, we specifically use variances of rates and delays observed

Table 2 Simulation parameters

Parameters	Descriptions
Data packet size	1000 bytes
ACK packet size	40 bytes
Number of small RTT flows	45
Round trip times of small RTT flows	20 ms
Round trip times of large RTT flows	20–1000 ms
Share of large RTT flows	0.1–0.9
Link capacity	10–1000 Mbps
Web traffic average arrival rate	500/s
Transfer size of short-lived flows	30 packets
defaultRNG	0 ^a

^a The seed value will be changed based on clock and counter

on the bottleneck link to measure system stability. In order to make results scientific and credible, each experiment is repeated 100 times, and all simulations are run for at least 200 s to ensure that the system has reached its steady state. Meanwhile, the statistics neglect the first 20% of simulation time. Then on the base of these, we present the confidence interval of simulation results. Additionally, please note that only the results of XCP and R-XCP are listed here, because M-XCP is extremely stable, its variances are the smallest in all simulations. However, M-XCP also experiences sluggish system responsiveness as shown in Sects. 3 and 4.

6.2 Varying round trip times

In this experiment, there are 45 small RTT flows as background traffic and five flows with long RTT values. Therefore, the share of large RTT flows is 10%. We fix the bottleneck capacity at 100 Mbps and vary two-way propagation delay of large RTT flows from 10 to 1000 ms. Meanwhile, the web-like traffic is also injected into the network to investigate the protocol's robustness. Figure 9 gives the average variances of

per-flow's rate, aggregate traffic rate and control cycle time, with 95% confidence interval. As shown in Fig. 9, when the large RTT value is smaller than 100 ms, simulation results of XCP and R-XCP are basically the same. This is because the ratio of maximum RTT to mean RTT is not sufficiently large, XCP remains stable, the stabilizing effects of R-XCP is not clear. When the large RTT value is greater than 100 ms, we find that an order of magnitude decrease in R-XCP, compared with XCP, in terms of aggregate traffic rate variance. We also observe that R-XCP reduces the variance of control interval by at least an order of magnitude, when aggregate traffic rate variance is decreased. That is to say, R-XCP significantly decreases the volatilities of aggregate traffic rate and control interval, and effectively improves the system stability under heterogeneous delays. Additionally, per-flow rate variances of large RTT flows also decrease at the same time. *Please note the logarithmic scale of the figures in this and next subsection.*

6.3 Varying the share of large RTT flows

With an increase in the share of large RTT flows, aggregate traffic rate and control interval as well as the other performance indicator get more smooth, regardless of which mechanism is applied. But the absolute values of these indicators are significantly reduced, when R-XCP is employed. Figure 10 shows that control interval variance is smaller than 1 ms^2 , far less than its mean, when the share of large RTT flows is >0.7 . However, when the share of large RTT flows is >0.4 , the variance of aggregate traffic rate is less than 20 Mbps^2 , the stabilizing effect of R-XCP is not significant. Therefore, we have the following result, when delay uncertainty is within a certain range, XCP can stabilize aggregate traffic rate at the bottleneck link, which shows that it is too conservative to set control interval to maximum RTT, e.g., M-XCP, resulting in low responsiveness.

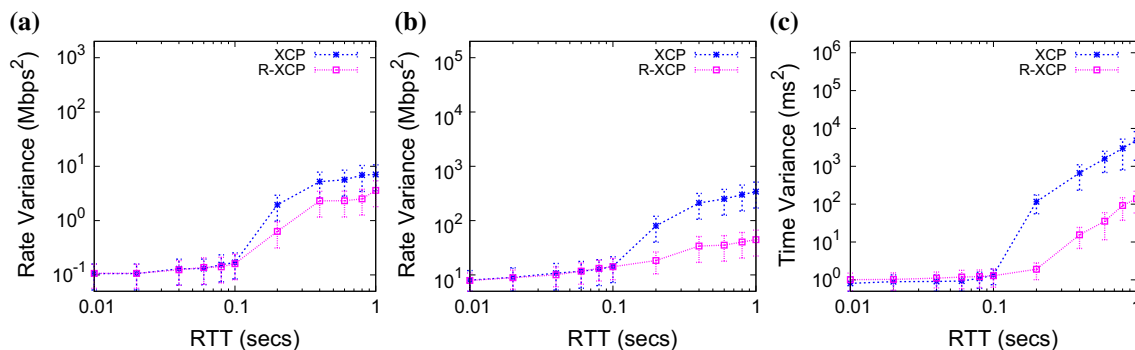


Fig. 9 The average variances (95% confidence interval) of per-flow's rate, aggregate traffic rate and control cycle time, with round trip times of large RTT flows ranging from 10 to 1000 ms, **a** per-flow's rate, **b** aggregate traffic rate, **c** control cycle time

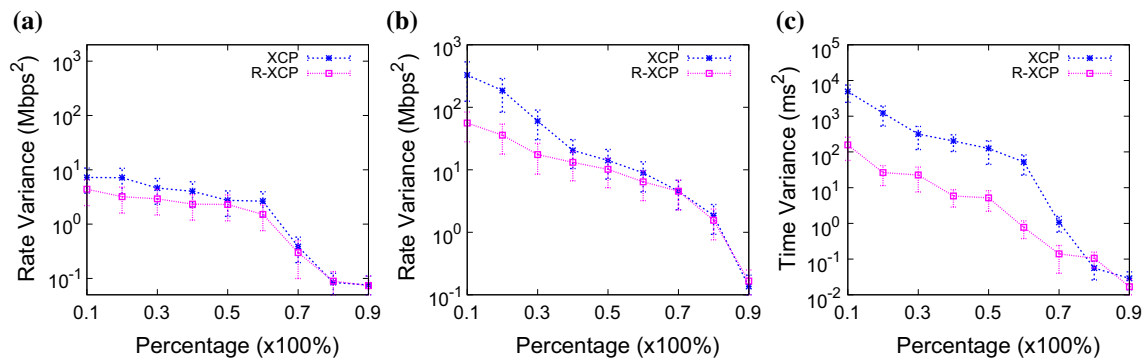


Fig. 10 The average variances (95% confidence interval) of per-flow’s rate, aggregate traffic rate and control cycle time, with share of large RTT flows varying from 10 to 90%. **a** per-flow’s rate, **b** aggregate traffic rate, **c** control cycle time

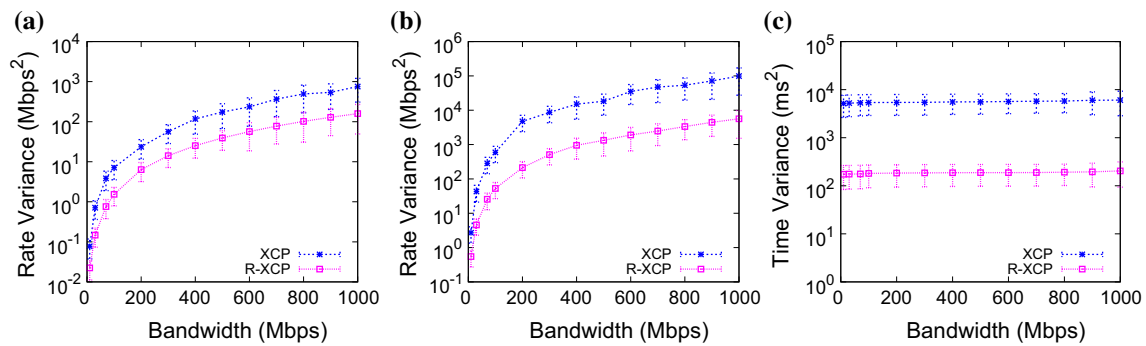


Fig. 11 The average variances (95% confidence interval) of per-flow’s rate, aggregate traffic rate and control cycle time, with bottleneck capacity varying from 10 to 1000Mbps. **a** per-flow’s rate, **b** aggregate traffic rate, **c** control cycle time

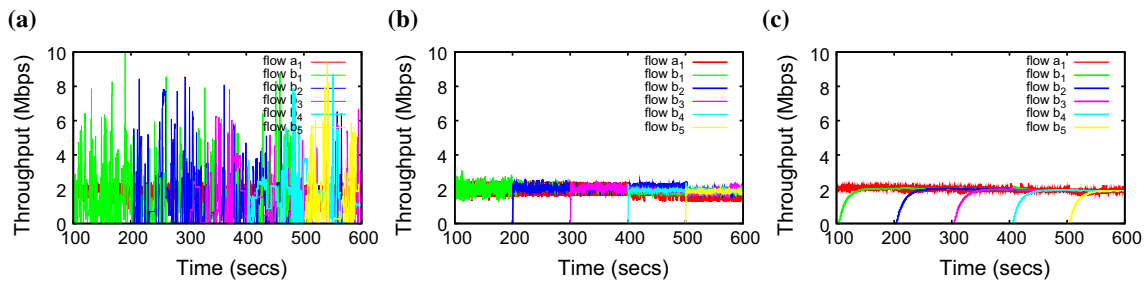


Fig. 12 The convergence dynamics of XCP, R-XCP and M-XCP, from *left to right*

6.4 Varying bottleneck capacity

As shown in Fig. 11, the bottleneck capacity has an amplification effect on the rate variance. When the bottleneck capacity increases, the fluctuations of aggregate traffic rate and per-flow throughput increase significantly, no matter which mechanism is employed. However, R-XCP reduces the variances of aggregate traffic rate and per-flow throughput by about an order of magnitude, compared with XCP, the stabilizing effect of R-XCP is obvious. Meanwhile, R-XCP reduces the variance of control interval significantly, but in each particular mechanism, control interval variance maintains the same across the whole range of bottleneck capacities varying from 10 to 1000Mbps.

6.5 Convergence dynamics

In this experiment, there are 45 small RTT flows (flow $a_1 \sim a_{45}$, $RTT_a = 20$ ms) as background traffic and 5 large RTT flows (flow $b_1 \sim b_5$, $RTT_b = 1000$ ms) that share a 100Mbps bottleneck link. We introduce 5 large RTT flows into the system one after another, with starting times separated by 100s. The left graph in Fig. 12 illustrates that XCP behaves in a noticeably unstable manner when the ratio of maximum to mean RTT is too large. The middle and right subfigures show that R-XCP and M-XCP both stabilize the system, and fairly reallocate bandwidth to new flows whenever they join the network. Although M-XCP is more smooth than R-XCP, however, M-XCP takes a much longer time than

R-XCP to converge to the fair allocation. Additionally, in R-XCP, whenever a new large RTT flow comes in, per-flow throughput becomes smoother, this is because the control cycle becomes larger in this case, leading to a smaller cutoff frequency and more robust levels.

6.6 More security

Additionally, M-XCP makes the system sensitive to erroneous flows which contribute little traffic but report large RTT values. These flows may actually have very large RTTs, overestimate their RTTs due to system jitters, or maliciously overstate their RTTs. Obviously, M-XCP is not an appropriate method under these scenarios, which slows down the entire network dynamics to accommodate a few flows, since it leads to long periods of under-utilization, alternating with periods of overload causing excessive delay and loss [20]. R-XCP can effectively alleviate, but not solve, this problem in M-XCP, due to the fact that it requires the amount of flows with large RTT values is large enough to significantly increase control interval d . To completely solve this problem, access control and other strategies are essentially needed.

7 Conclusions

In this paper, with the aid of Lyapunov–Krasovskii functional, R-XCP is proposed to solve the weakness of XCP under heterogeneous delays based on the robust control theory. R-XCP significantly decreases the volatilities of aggregate traffic rate and per-flow throughput, and effectively improves the system stability when the ratio of maximum to mean RTTs is sufficiently large. Meanwhile, the fluctuation of control interval also decreases dramatically. Compared with M-XCP, R-XCP has a better balance between robustness and responsiveness, and the computing complexity declines significantly at the same time. Besides, R-XCP also makes the system less sensitive to erroneous RTT attacks.

Acknowledgements This work is partially supported by the Program of National Natural Science Foundation of China (No. 41404025), and State Key Laboratory of Geo-Information Engineering (No. SKLGIE2014-M-2-2). It is also funded by the Fundamental Research Funds for the Central Universities (No. 2014B03314).

References

- Ott, T. J. (2005). Transport protocols in the TCP paradigm and their performance. *Telecommunication Systems*, 30(4), 351–385.
- Qazi, I. A., Znati, T., & Andrew, L. L. H. (2009). Congestion control using efficient explicit feedback. In *Proceedings of IEEE INFOCOM*.
- Rodriguez-Colina, E., Gil-Leyva, D., Marzo, J. L., Vctor, M., & Ramos, R. (2014). A bit error rate analysis for TCP traffic over parallel free space photonics. *Telecommunication Systems*, 56(4), 455–466.
- Katabi, D., Handley, & M., Rohrs, C. (2002). Congestion control for high bandwidth-delay-product networks. In *Proceedings of IEEE/ACM SIGCOMM*.
- Zhang, Y., & Henderson, T. (2005). An implementation and experimental study of the explicit control protocol (XCP). In *Proceedings of IEEE INFOCOM*.
- Lopez Pacheco, D. M., Pham, C., & Lefevre, L. (2006). XCP-i: explicit control protocol for heterogeneous inter-networking of high speed networks. In *Proceedings of IEEE GLOBECOM*.
- Yang, X., Lu, Y., & Zan, L. (2010). Improving XCP to achieve max-min fair bandwidth allocation. *Computer Networks*, 54(3), 442–461.
- Zhou, H., Hu, C., & He, L. (2013). Improving the efficiency and fairness of eXplicit control protocol in multi-bottleneck networks. *Computer Communications*, 36(10), 1193–1208.
- Abrantes, F., & Ricardo, M. (2006). XCP for shared-access multi-rate media. *ACM SIGCOMM Computer Communication Review*, 36(11), 29–38.
- Barreto, L. (2015). XCP-WinF and RCP-WinF: Improving explicit wireless congestion control. *Journal of Computer Networks and Communications*, 2015, 1–18.
- Zhou, K., Yeung, K. L., & Li, V. O. K. (2004). P-XCP: A transport layer protocol for satellite IP networks. In *Proceedings of IEEE GLOBECOM*.
- Sun, Y., Ji, Z., & Wang, H. (2012). A modified variant of explicit control protocol in satellite networks. *Journal of Computational Information Systems*, 8(10), 4355–4362.
- Liu, F., Wang, H. O., & Guan, Z. (2012). Hopf bifurcation control in the XCP for the internet congestion control system. *Nonlinear Analysis: Real World Applications*, 13(x), 1466–1479.
- Cheng, S., Li, J., Zhu, L., & Guo, C. (2010). Time-domain sending rate and response function of eXplicit control protocol. *Telecommunication Systems*, 45(4), 323–328.
- Zhang, Y., & Ahmed, M. (2005). A control theoretic analysis of XCP. In *Proceedings of IEEE INFOCOM*.
- Lu, Z., & Zhang, S. (2009). Stability analysis of XCP congestion control systems. In *Proceedings of IEEE WCNC*.
- Wilson, C., Coakley, C., & Zhao, B. Y. (2007). Fairness attacks in the explicit control protocol. In *Proceedings of IEEE IWQoS*.
- Alparslan, O., Arakawa, S., & Murata, M. (2009). XCP-based transmission control mechanism for optical packet switched networks with very small optical RAM. *Photonic Network Communications*, 18(2), 237–243.
- Alparslan, O., Arakawa, S., & Murata, M. (2011). Buffer scaling for optical packet switching networks with shared RAM. *Optical Switching and Networking*, 8(1), 12–22.
- Andrew, L. L., Wydrowski, B. P., & Low, S. H. An example of instability in XCP. http://netlab.caltech.edu/maxnet/XCP_instability.
- Jiang, H., & Dovrolis, C. (2002). Passive estimation of TCP round trip times. *ACM SIGCOMM Computer Communications Review*, 32(3), 75–88.
- Shakkottai, S., Srikant, R., Brownlee, N., Broido, A., & Claffy, K. (2004). The RTT distribution of TCP flows in the internet and its impact on TCP-based flow control. CAIDA technical report.
- Gu, K., Kharitonov, V. L., & Chen, J. (2003). *Stability of time-delay systems*. Boston: Birkhauser.
- Boyd, S., El Ghaoui, L., Feron, E., & Balakrishnan, V. (1997). *Linear matrix inequalities in system and control theory*. Philadelphia: Society for Industrial Mathematics.
- Kelly, F. P., Maulloo, A. K., & Tan, D. K. H. (1998). Rate control in communication networks: Shadow prices, proportional fairness,

- and stability. *Journal of the Operational Research Society*, 49(x), 237–252.
26. Floyd, S. (2003). HighSpeed TCP for large congestion windows. RFC3649.
 27. Kelly, T. (2003). Scalable TCP: Improving performance in high-speed wide area networks. *ACM SIGCOMM Computer Communication Review*, 33(2), 83–91.
 28. Leith, D., & Shorten, R. (2008). H-TCP: TCP congestion control for high bandwidth-delay product paths. Internet Draft.
 29. Ha, S., Rhee, I., & Xu, L. (2008). CUBIC: A new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating Systems Review*, 42(5), 64–74.
 30. Wei, D. X., Jin, C., Low, S. H., & Hegde, S. (2006). FAST TCP: Motivation, architecture, algorithms performance. *IEEE/ACM Transactions on Networking*, 14(6), 1246–1259.
 31. Bullot, H., Les Cottrell, R., & Hughes-Jones, R. (2003). Evaluation of advanced TCP stacks on fast long-distance production networks. *Journal of Grid Computing*, 1(4), 345–359.
 32. Ryu, S., Rump, C., & Qiao, C. (2004). Advances in active queue management (AQM) based TCP congestion control. *Telecommunication Systems*, 25(3), 317–351.
 33. Floyd, S., & Jacobson, V. (1993). Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4), 397–413.
 34. Hollot, C. V., Misra, V., Towsley, D., & Gong, W. (2001). On designing improved controllers for AQM routers supporting TCP flows. In *Proceedings of IEEE INFOCOM*.
 35. Feng, W., Shin, K. G., Kandlur, D. D., & Saha, D. (2002). The BLUE active queue management algorithms. *IEEE/ACM Transactions on Networking*, 10(4), 513–528.
 36. Pan, R., Prabhakar, B., & Psounis, K. (2000). CHOKe: A stateless AQM scheme for approximating fair bandwidth allocation. In *Proceedings of IEEE INFOCOM*.
 37. Kunniyur, S., & Srikant, R. (2001). Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management. In *Proceedings of IEEE/ACM SIGCOMM*.
 38. Athuraliya, S., Li, V. H., Low, S. H., & Yin, Q. (2001). REM: Active queue management. *IEEE Network*, 15(3), 48–53.
 39. Kahe, G., Jahangir, A. H., & Ebrahimi, B. (2014). AQM controller design for TCP networks based on a new control strategy. *Telecommunication Systems*, 57(4), 295–311.
 40. Chrost, L., & Chydzinski, A. (2016). On the deterministic approach to active queue management. *Telecommunication Systems*, 63(1), 27–44.
 41. Falk, A., Katabi, D., & Pryadkin, Y. (2007). Specification for the eXplicit control protocol (XCP). draft-falk-xcp-03.txt (work in progress).
 42. Xia, Y., Subramanian, L., Stoica, I., & Kalyanaraman, S. (2005). One more bit is enough. In *Proceedings of IEEE/ACM SIGCOMM*.
 43. Qzai, I. A., & Znati, T. (2008) On the design of load factor based congestion control protocols for next-generation networks. In *Proceedings of IEEE INFOCOM*.
 44. Dukkupati, N., Kobayashi, M., Zhang-Shen, R., & McKeown, N. (2005) Processor sharing flows in the internet. In *Proceedings of IEEE IWQoS*.
 45. Zhang, Y., Leonard, D., & Loguinov, D. (2006). JetMax: Scalable max–min congestion control high-speed heterogeneous networks. In *Proceedings of IEEE INFOCOM*.
 46. Wydrowski, B. P., Andrew, L. L. H., & Mareels, I. M. Y. (2004). MaxNet: Faster flow control convergence. In *Proceedings of IFIP/TC6 Networking*.
 47. Tahiliani, R. P., Tahiliani, M. P., & Sekaran, K. C. (2015). *TCP congestion control in data center networks. Handbook on data centers*. New York: Springer.
 48. Vasudevan, V., Phanishayee, A., Shah, H., Krevat, E., Andersen, David G., Ganger, Gregory R., et al. (2009). Safe and effective fine-grained TCP retransmissions for datacenter communication. *ACM SIGCOMM Computer Communication Review*, 39(4), 303–314.
 49. Alizadehy, M., Greenberg, A., Maltzy, D. A., Padhye, J., Patley, P., Prabhakar, B., et al. (2010). Data center TCP (DCTCP). *ACM SIGCOMM Computer Communication Review*, 40(4), 63–74.
 50. Haitao, W., Feng, Z., Guo, C., & Zhang, Y. (2013). ICTCP: Incast congestion control for TCP in data-center networks. *IEEE/ACM Transactions on Networking*, 21(2), 345–358.
 51. Vamanan, B., Hasan, J., & Vijaykumar, T. N. (2012). Deadline-aware datacenter TCP (D2TCP). *ACM SIGCOMM Computer Communication Review*, 42(4), 115–126.
 52. Zhang, J., Ren, F., Tang, L., & Lin, C. (2013). Taming TCP incast throughput collapse in data center networks. In *Proceedings of 21st IEEE international conference on network protocols (ICNP)*.
 53. Cheng, L., Wang, C. L., & Lau, F. C. M. (2013). PVTCP: Towards practical and effective congestion control in virtualized datacenters. In *Proceedings of 21st IEEE international conference on network protocols (ICNP)*.
 54. Alizadeh, M., Edsall, T., Dharmapurikar, S., Vaidyanathan, R., Chu, Kevin, Fingerhut, A., et al. (2014). CONGA: Distributed congestion-aware load balancing for datacenters. *ACM SIGCOMM Computer Communication Review*, 44(4), 503–514.
 55. Chen, W., Ren, F., Xie, J., Lin, C., Yiny, K., & Baker, F. (2015). Comprehensive understanding of TCP incast problem. In *Proceedings of IEEE conference on computer communications (INFOCOM)*.
 56. Hollot, C. V., Liu, Y., Misra, V., & Towsley, D. (2003). Unresponsive flows and AQM performance. In *Proceedings of IEEE INFOCOM*.
 57. Paxson, V. (1997). End-to-end internet packet dynamics. In *Proceedings of IEEE/ACM SIGCOMM*.
 58. Balakrishnan, H., Dukkupati, N., McKeown, N., & Tomlin, C. J. (2007). Stability analysis of explicit congestion control protocols. *IEEE communications magazine*.
 59. Robust control toolbox. <http://www.mathworks.com/products/robust/>.
 60. Traffic analysis research. <http://www.caida.org/research/traffic-analysis/>.
 61. Crovella, M. E., & Bestavros, A. (1997). Self-similarity in world wide web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6), 835–846.
 62. Issariyakul, T., & Hossain, E. (2011). *Introduction to network simulator NS2*. Berlin: Springer.
 63. Floyd, S., Handley, M., Padhye, J., & Widmer, J. (2000). Equation-based congestion control for unicast applications. In *Proceedings of IEEE/ACM SIGCOMM*.
 64. Wei, D. X., Cao, P., & Low, S. H. (2005). Time for a TCP benchmark suite? Technical report, Caltech CS, Stanford EAS, Caltech.
 65. Ha, S., Kim, Y., Le, L., Rhee, I., & Xu, L. (2006). A step toward realistic evaluation of high-speed TCP protocols. Technical report, North Carolina State University.



Lina He received her MS and PhD degrees in Geodesy and Surveying Engineering from Tongji University, China, in 2010 and 2013, respectively. Her current research interest is satellite orbit determination and data processing. Dr. He authored and coauthored 4 journal papers, including articles in Computer Communications as well as Sensors. She has worked on 2 projects funded by Chinese National Natural Science Foundation.



Hairui Zhou received his MS and PhD degrees in Control Theory and Engineering from Northwestern Polytechnical University, China, in 2007 and 2010, respectively. He worked at Shanghai Jiao Tong University as post-doctoral from 2010 to 2012. His research interest includes congestion control in data center networks, and wireless networks as well as computer physical systems. He authored and coauthored 4 journal papers, including articles in Computer Communications as well as Proc IMechE Part G: Journal of Aerospace Engineering.