Front. Comput. Sci., 2014, 8(3): 426–439
DOI 10.1007/s11704-014-3443-9

RESEARCH ARTICLE

Achieving high throughput and TCP Reno fairness in delay-based TCP over large networks

Jingyuan WANG (✉)[1], Jiangtao WEN[2], Yuxing HAN[4], Jun ZHANG[2], Chao LI[1,3], Zhang XIONG[1]

1 School of Computer Science and Engineering, Beihang University, Beijing 100191, China
2 Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
3 Research Institute of Beihang University in Shengzhen, Shenzhen 518057, China
4 Flora Production Inc., Santa Clara CA 95054, USA

Abstract The transport control protocol (TCP) has been widely used in wired and wireless Internet applications such as FTP, email and HTTP. Numerous congestion avoidance algorithms have been proposed to improve the performance of TCP in various scenarios, especially for high speed and wireless networks. Although different algorithms may achieve different performance improvements under different network conditions, designing a congestion algorithm that can perform well across a wide spectrum of network conditions remains a great challenge. Delay-based TCP has a potential to overcome above challenges. However, the unfairness problem of delay-based TCP with TCP Reno blocks widely the deployment of delay-based TCP over wide area networks. In this paper, we proposed a novel delay-based congestion control algorithm, named FAST-FIT, which could perform gracefully in both ultra high speed networks and wide area networks, as well as keep graceful fairness with widely deployed TCP Reno hosts. FAST-FIT uses queuing delay as a primary input for controlling TCP congestion window. Packet loss is used as a secondary signal to adaptively adjust parameters of primary control process. Theoretical analysis and experimental results show that the performance of the algorithm is significantly improved as compared to other state-of-the-art algorithms, while maintaining good fairness.

Keywords TCP congestion control, congestion avoidance, fairness

Received November 6, 2013; accepted April 7, 2014

E-mail: jywang@buaa.edu.cn

1 Introduction

The Transport control protocol (TCP) is a reliable connection-oriented transport layer protocol that is widely used on the Internet. Congestion control is an integral module of TCP that directly impacts the performance of the protocol. TCP implements congestion control using a sliding window called the congestion window. Standard TCP congestion control algorithms such as TCP Reno [2], which make use of the slow start (SS) and congestion avoidance (CA) algorithms to adjust the size of the congestion window, have enjoyed tremendous success to date. However, both high-speed networks channels such as optical-fiber links and lossy channels like wireless networks could cause significant performance degradation for TCP Reno. On one hand, TCP Reno assumes all packet losses are *only* caused by over-driving the network; therefore any packet losses caused by random physical layer artifacts (e.g., multi-path, interferences) that have nothing to do with congestion will also cause the CA algorithm of Reno to aggressively slow down the transmission rate. On the other hand, in a high-speed network, TCP Reno requires a very low

This paper is an extended version of the 2014 International Conference on Computing, Networking and Communications paper [1].

bit error rate (in the order 10^{-7} or lower [3]) to fully utilize network capacity. This requirement is far from the reality of network condition.

Many TCP variants have been proposed to improve the CA algorithm of TCP Reno over wireless or high speed networks, including TCP Westwood [4, 5], TCP Veno [6] for wireless applications and highspeed TCP [7], scalable TCP [8] compound TCP [9], TCP CUBIC [10] for high speed networks, as well as some variants and related technololgies for datacenter networks [11, 12]. Some literatures reported the delay based TCPs, like TCP Vegas [13] and FAST TCP [3], had a good scalability over wireless and highspeed network [14], but the unfairness problem between delay based TCP and widely deployed TCP Reno limited their deployment in the Internet [15]. In other words, although these existing TCP variants have achieved success in their respective target applications, designing a TCP CA algorithm that performs gracefully in **both** wireless and high speed links to deal with heterogeneity of the Internet remains a great challenge.

With network queuing delay as the algorithms inputs, FAST TCP also has great potential to overcome above challenges. However, the well-know shortcoming of delay-based TCP that it is unable to fairly co-exist with standard TCP Reno seriously limits FAST TCP wide deployment over wide area networks. Although many improved TCP variants, such as compound TCP [9], Cx-TCP [16] etc., are proposed to enable the coexistence between delay-based TCP and TCP Reno, none of them can achieve similar performance to FAST TCP over ultra high speed networks. Designing a TCP variant that can achieve the stat-of-the-art performance over **both** ultra high-speed networks and wide area networks, in other words, enabling FAST TCP over WAN, still remains a great challenge.

In this paper, we first analyze the theoretical advantages and disadvantages for existing TCP CA algorithms from the throughput model perspective, and demonstrate the advantage of pure delay based TCP algorithms over other CA algorithms when used over wireless and high speed networks. Based on the analysis, we show that the unfairness between delay based TCP and TCP Reno is a major barrier of obtaining efficient and fair congestion avoidance for wireless and high speed heterogeneous network. To solve this problem, we propose a novel TCP CA algorithm, named FAST-FIT, which is a continuation and improvement over the TCP-FIT [17–19] algorithm that we proposed. Similar to the Reno based TCP-FIT, FAST-FIT is capable of utilizing more fully the capacity of wireless and high-speed networks while maintaining good fairness characteristic. In contrast to the TCP-FIT algorithm in [18] which uses packet loss as the primary congestion window control input, FAST-FIT primarily uses queuing delay, and thereby improves the performance in high packet loss environments. Experiments over live networks with emulators demonstrated the throughput and fairness characteristic of the proposed algorithm and improvements over other approaches.

The rest of this paper is organized as follows. An analysis of the inherent performance limitation based on the throughput model of existing loss-based and hybrid TCP algorithms is provided in Section 2. An analysis on advantages and disadvantages of delay based TCP algorithms is provided in Section 3. In Section 4, a novel TCP algorithm, FAST-FIT, is proposed on the basis of recent proposed TCP-FIT. In Section 5, three sets of experimental results, Emulation experiments, PlanetLab experiments and 3G wireless experiments are shown. Section 6 concludes the paper.

2 Throughput model based analysis of existing loss based and hybrid CA algorithms

In this section, we use the throughput models to analyze the inherent performance limitations of existing loss based and hybrid Congestion Avoidance algorithms. We will use TCP Reno as an example, although the same analysis also applies to other loss based or hybrid CA algorithms.

As illustrated in Fig. 1, we assume that the network only contains one bottleneck link with a bandwidth capacity limit C. We also assume that there are S TCP sessions indexed by $s \in \{1, 2, \ldots, S\}$ sharing this bottleneck link, and different sessions may have different non-bottleneck routing paths. For each TCP session s, the corresponding end-to-end round-trip propagation delay and inherent random packet loss rate are denoted as D_s and P_s respectively. D_s and P_s reflect the aggregated impacts of both the bottleneck link and the non-bottleneck links traversed by session s. After the combined throughput of all TCP sessions reaches the bottleneck capacity C, TCP packets begin to queue up in various buffers,

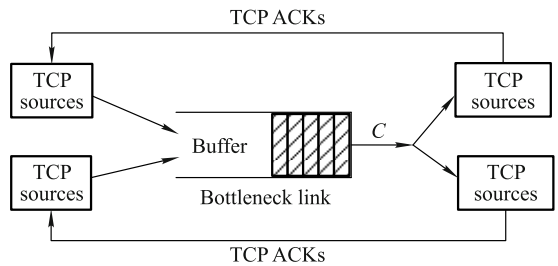


Fig. 1 A network with single bottleneck link

resulting in a queueing delay. When the length of a queue becomes longer than a threshold, routers using mainstream queue management algorithms will start dropping packets, producing congestion packet loss.

For TCP session s, the throughput $T'_s(t)$ is adjusted in each period according to a function F_s:

$$T'_s(t+1) = F_s(T'_s(t), v'(t)), \tag{1}$$

where $v'(t)$ is the congestion measure price of the bottleneck link at period t. Following the notation of [20], $v'(t) = m(p'(t), q'(t))$, where $v'(t)$ is a function of the congestion packet loss rate $p'(t)$ and the queuing delay $q'(t)$ of the bottleneck link at time t. To investigate the intrinsic properties of Congestion Avoidance algorithms, we follow the convention of [20] and [21], and ignore the impacts of factors such as slow start, time out, and retransmission to the performance of the system, and assume that the throughput of a TCP session is determined solely by its Congestion Avoidance algorithm.

According to the conclusion of [20], the network topology of Fig. 1 will reach a equilibrium state when

$$T_s = F_s(T_s, v(p,q)), \tag{2}$$

where p and q are the average congestion packet loss rate and queueing delay of bottleneck at equilibrium state. Assuming F_s is continuously differentiable and $\partial F_s/\partial v \neq 0$ in the open set $\{(T_s, v)|T_s > 0, v > 0\}$, by implicit function theorem, there exists a unique function f_s from $\{T_s > 0\}$ to $\{v > 0\}$ that

$$v = f_s(T_s).$$

Following the conclusion of [21] and [22], the congestion control process of such network can be modeled as an optimization problem that maximizes the aggregate utility of TCP sessions passed by the bottleneck:

$$\max_{T \geqslant 0} \sum_s U_s(T_s), \qquad \text{subject to} \quad \sum_s T_s \leqslant C, \tag{3}$$

where U_s is the utility function of each session s:

$$U_s(T_s) = \int f_s(T_s) dT_s, \quad T_s \geqslant 0,$$

Since the throughput property of a CA algorithm could be described by its throughput model, F_s and f_s of a TCP session s are determined by the throughput model of the Congestion Avoidance algorithm adopted by s:

$$T_s = T_{ca}\left(v(p,q), D_s, P_s\right).$$

The throughput models of several popular TCP CA algorithms are summarized in Table 1 [3,8,9,23–27]. All of these models follow the same assumptions and only consider the CA part of a TCP CA algorithm.

Taking TCP Reno as a example, since $p \geqslant 0$ and $q \geqslant 0$, there is an upper bound for the throughput of a Reno session s for any given P_s and D_s:

$$T_s = \frac{1}{D_s + q}\sqrt{\frac{3}{2(P_s+p)}} \leqslant \frac{1}{D_s}\sqrt{\frac{3}{2P_s}} = T_s^{\max}. \tag{4}$$

For a bottleneck link only contains TCP Reno sessions, when the optimal solution of the optimization problem defined in Eq. (3) is achieved, the aggregated throughput of all TCP sessions over the bottleneck link is upper bounded by

$$\sum_s T_s \leqslant \min\left\{C, \sum_s T_s^{\max}\right\}.$$

It can been seen that when $C > \sum_s T_s^{\max}$, no matter what the solution of the optimization problem is, the bottleneck always stays in an unsaturated state. Because $T_s^{\max}$ is inversely proportional to D_s and $\sqrt{P_s}$ as shown in Eq. (4), TCP Reno sessions are unable to fully utilize network capacity when the bottleneck link state of the network is extremely bad (i.e., having a high random packet loss rate and/or a very long propagation delay), or if the bandwidth of bottleneck is

Table 1 TCP CA throughput models

	Throughput models	**Upper bounds $T^{\max}$**
Reno	$T = \frac{1}{D+q}\sqrt{\frac{3}{2(P+p)}}$ [23]	$T^{\max} = \frac{1}{D}\sqrt{\frac{3}{2P}}$
CUBIC	$T = 1.17\sqrt[4]{\frac{1}{(D+q)(P+p)^3}}$ [24]	$T^{\max} = 1.17\sqrt[4]{\frac{1}{DP^3}}$
CTCP	$T = \frac{1}{D+q}\frac{\Lambda}{(P+p)^{\frac{1}{2-k}}}$, $\Lambda = \frac{\left(\frac{1-(1-\beta')^{2-k}}{2-k}\right)^{\frac{1-k}{2-k}}}{\alpha'^{\frac{1}{2-k}}(1-(1-\beta')^{1-k})}$ [9]	$T^{\max} = \frac{1}{D}\frac{\Lambda}{P^{\frac{1}{2-k}}}$
Scalable TCP	$T = \frac{\alpha(1+\beta)}{2(P+p)(1-\beta)(D+q)}$ [8]	$T^{\max} = \frac{\alpha(1+\beta)}{2PD(1-\beta)}$
	Defined $\alpha = 0.01, \beta = 0.875$	
Veno	$T = \frac{1}{D+q}\sqrt{\frac{1+\gamma}{2(1-\gamma)(P+p)}}$, where $\frac{1}{2} \leqslant \gamma \leqslant \frac{4}{5}$ [25]	$T^{\max} = \frac{1}{D}\sqrt{\frac{1+\gamma}{2(1-\gamma)P}}$
Westwood	$T = \frac{1}{\sqrt{(D+q)((P+p)D+q)}}\sqrt{\frac{(1-(P+p))}{P+p}}$ [26]	$T^{\max} = \frac{1}{DP}\sqrt{1-P}$
Vegas/FAST	$T = \alpha'/q$, where α'' is preset parameter [3,27]	×

sufficiently high. This can be seen as an inherent reason why TCP Reno cannot achieve optimal performance in wireless and/or high speed networks. Similar conclusions can be drawn from other loss based or hybrid TCP Congestion Avoidance algorithms in Table 1. As shown in last column of Table 1, all these algorithms have a corresponding throughput upper bounder $T^{\max}$, which limits network utilization.

3 Delay based CA algorithms

Based on the results in [27] and [3], in this section we show that delay based TCP CA algorithms, TCP Vegas and FAST TCP, have the potential of fully utilizing network bandwidth capacity, but they have the problem of unfairness.

In delay based TCP algorithms such as TCP Vegas, the congestion window w is controlled by the following method [13]:

$$w \leftarrow \begin{cases} w+1, & \text{if} \quad diff < \gamma_v, \\ w, & \text{if} \quad diff = \gamma_v, \\ w-1, & \text{if} \quad diff > \gamma_v. \end{cases} \tag{5}$$

In TCP Vegas, $diff$ is calculated as:

$$diff = \left(\frac{w}{rtt_{\min}} - \frac{w}{rtt_{\text{avg}}}\right) \cdot rtt_{\min} = \frac{rtt_{\text{avg}} - rtt_{\min}}{rtt_{\text{avg}}} w,$$

where rtt_{avg} is the average RTT of a congestion window update period, and $rtt_{\min}$ is the minimum RTT sample observed. $rtt_{\min}$ is in general used as a reasonable estimate of the propagation delay of the network. γ_v is a parameter of TCP Vegas.

In delay based TCP, $q = rtt_{\text{avg}} - rtt_{\min}$ is an approximation for the queuing delay in bottleneck router, $T = w/rtt_{\text{avg}}$ could be considered as an estimate of the average throughput for a TCP session, since TCP sends w packets in a RTT. Therefore, the average queue length Q of the in-flight packets in the bottleneck buffer of a TCP session can be expressed as:

$$Q = q \cdot T = (rtt_{\text{avg}} - rtt_{\min}) \cdot \frac{w}{rtt_{\text{avg}}}. \tag{6}$$

Therefore, $diff$ is equal to the length of the in-flight packet queued in the bottleneck buffer. When TCP Vegas reaches the steady state, the stable queue length is $Q^* = \gamma_v$ according to Eq. (5).

In FAST TCP, the congestion window w is controlled by [3]:

$$w \leftarrow \min\left\{2w, \frac{1}{2}w + \frac{1}{2}\left(\frac{rtt_{\min}}{rtt_{\text{avg}}}w + \gamma_f\right)\right\},$$

where γ_f is a parameter. When the congestion window of FAST TCP reaches the steady state value w^*, γ_f can be calculated as

$$\gamma_f = \left(1 - \frac{rtt_{\min}}{rtt_{\text{avg}}}\right) w^* = \frac{rtt_{\text{avg}} - rtt_{\min}}{rtt_{\text{avg}}} w^* = Q^*.$$

It can be seen from the above analysis that, the queue length Q equals to γ_v and γ_f for TCP Vegas and FAST TCP respectively. From Eq. (6), $T = Q/q$ in Vegas and FAST. Their steady state throughput of delay based CA can then be calculated as [3, 13]:

$$T_{\text{delay}} = \frac{\gamma}{q}. \tag{7}$$

It is easy to prove that if there exists a Vegas/FAST session s' in the network defined in the last section, optimization problem Eq. (3) will reach a optimal solution that satisfies $\sum_{s \in S} T_s = C$.

Proof We assume that the optimal solution of Eq. (3) satisfies $\sum_{s \in S} T_s < C$, which means that the bottleneck is not saturated. According to the network model in Fig. 1, there are no packets queued in bottleneck buffer, leading to queuing delay $q = 0$. From Eq. (7), $T_{s'} \longrightarrow \infty$. This contradicts with $\sum_{s \in S} T_s < C$. □

From the above analysis, theoretically, except for pure delay based TCP CA algorithms such as Vegas or FAST, loss based or hybrid CA algorithms can only mitigate but can not fully solve the bandwidth utilization problem in wireless and/or high speed networks.

Although having the advantage of potentially fully utilizing network capacity, delay based TCP CA algorithms also have severe problems. Because of buffers, increasing queuing delay may not necessarily lead to immediate packet loss. As a result, when delay based TCP shares the same bottleneck with other TCP algorithms, between the time delay starts to increase and congestion-caused packet loss starts to occur, the transmission rate for the delay based TCP will decrease; while that for other TCPs will not, resulting in unfairness. This disadvantage limits the deployment of delay based TCP CA algorithms in real networks. As pointed out in [15], "given any target fairness between TCP Reno and FAST/Vegas, there exists a protocol parameter γ that achieves it. It is however an open problem how to compute γ in practice dynamically with only local information."

4 An Inter-protocol fair delay based CA algorithm

In this section, we describe an inter-protocol fair CA algorithm called FAST-FIT. We show how the proposed delay

based TCP CA algorithm overcomes the problems described in Sections 2 and 3. The new algorithm has the potential of fully utilizing network bottleneck capacity, and is able to coexist with TCP Reno in a fair way. The design of proposed algorithm is motivated by throughput model of the TCP-FIT [18] algorithm.

4.1 TCP-FIT

TCP-FIT is a novel TCP Congestion Avoidance algorithm for heterogenous networks. It was first proposed by Wang et al. in [18]. The algorithm is a hybrid TCP CA algorithm and uses both packet loss and queuing delay as inputs to congestion window control.

Similar to TCP Reno, TCP-FIT uses the additive increase multiple decrease (AIMD) mechanism to adjust the congestion window w. In AIMD based Congestion Avoidance algorithms,

$$\begin{aligned} &\text{Each } RTT: \ w \leftarrow w + a, \\ &\text{Each } Loss: \ w \leftarrow w - bw, \end{aligned} \tag{8}$$

where a and b are two parameters that are usually fixed. For example in TCP Reno, $a = 1$ and $b = 1/2$. By contrast, in TCP-FIT,

$$\begin{aligned} &\text{Each } RTT: \ w \leftarrow w + N_t, \\ &\text{Each } Loss: \ w \leftarrow w - \frac{2}{3N_t + 1}w, \end{aligned}$$

where N_t is a dynamic parameter updated periodically:

$$\begin{cases} N_{t+1} = N_t + 1, & Q < \alpha\dfrac{\bar{w}}{N_t}, \\ N_{t+1} = N_t, & Q = \alpha\dfrac{\bar{w}}{N_t}, \\ N_{t+1} = \max\{1, N_t - 1\}, & Q > \alpha\dfrac{\bar{w}}{N_t}, \end{cases} \tag{9}$$

where Q is calculated as in Eq. (6), and similar to the value of $diff$ in TCP Vegas, is the estimated average queue length of the in-flight packets in the bottleneck buffer. $\bar{w}$ is the average window size of a N_t update period and $\alpha \in (0, 1)$ is a preset parameter.

The average throughput model of Eq. (8) is found in [28] as:

$$T = \frac{1}{RTT}\sqrt{\frac{2b}{a(2-b)PLR}}. \tag{10}$$

Using the notations in Sections 2 and 3, and let D, P, q, p be the end-to-end propagation RTT, the inherent packet loss rate, the queuing delay and the congestion packet loss rate of a TCP session, then $RTT = D + q$ and $PLR = P + p$. Let $a = 1$ and $b = 1/2$, the throughput model of TCP Reno

$$T_{\text{reno}} = \frac{1}{D+q}\sqrt{\frac{3}{2(P+p)}}. \tag{11}$$

For TCP-FIT, in the steady state, $a = N^*$ and $b = 2/(3N^* + 1)$, where N^* is the steady state of N_t, we get:

$$T_{\text{fit}} = \frac{N^*}{D+q}\sqrt{\frac{3}{2(P+p)}}. \tag{12}$$

Comparison between Eqs. (12) and (11) shows that the throughput of TCP-FIT is N^* times of the throughput of TCP Reno under the same network conditions.

From Eq. (9) we also know that N^* satisfies:

$$Q^*_{\text{fit}} = \alpha \cdot \frac{\bar{w^*}}{N^*}, \quad (N^* \geqslant 1), \tag{13}$$

where Q^*_{fit} and $\bar{w^*}$ are average queue length and window size at the steady state. On the other hand, from Eq. (6) we have

$$Q^*_{\text{fit}} = q \cdot \frac{\bar{w^*}}{D+q}. \tag{14}$$

Combining Eqs. (13) and (14) gives

$$N^* = \max\left\{\alpha\frac{D+q}{q}, 1\right\}. \tag{15}$$

Plugging Eq. (15) into Eq. (12), we find the throughput model of TCP-FIT:

$$T_{\text{fit}} = \max\left\{\frac{\alpha}{q}\sqrt{\frac{3}{2(P+p)}}, \ \frac{1}{D+q}\sqrt{\frac{3}{2(P+p)}}\right\}. \tag{16}$$

It is easy to show following the same proof as for Vegas and FAST in Section 3 that TCP-FIT can fully utilize network bandwidth.

4.2 The FAST-FIT algorithm

Comparing the throughput models of TCP-FIT with Vegas/FAST, it is observed that if let $\gamma = \alpha\sqrt{3/2(P+p)}$, Vegas/FAST would have a similar throughput expression as TCP-FIT, i.e.,

$$T = \frac{\alpha}{q}\sqrt{\frac{3}{2(P+p)}}. \tag{17}$$

Therefore solely from the throughput model perspective, TCP-FIT can be considered as a delay based TCP having an average queue length of $Q^*_{\text{fit}} = \alpha\sqrt{3/2(P+p)}$. We know TCP sends w packets during a RTT, and the average window size of TCP Reno should be

$$\bar{w}_{\text{reno}} = T \cdot (D+q) = \sqrt{\frac{3}{2(P+p)}} = \frac{1}{\alpha}Q^*_{\text{fit}}. \tag{18}$$

Therefore, from Eq. (18), we know that the queue length of a TCP-FIT session is α times of the window size of a TCP Reno session under the same network condition.

To guarantee that the throughput of TCP-FIT is not lower than TCP Reno, N is limited to a value of at least 1. In addition, we can show that a well selected α will ensure that the value of N of a TCP-FIT session will decrease to 1 when a TCP-FIT and some TCP Reno sessions share a bottleneck without random packet loss.

For a network scenario where a group of TCP Reno sessions share a lose-free bottleneck link with a TCP-FIT session, if $\alpha \leqslant \bar{Q}_{\text{reno}}/\bar{w}_{\text{reno}}$, the TCP-FIT session will degenerate to a Reno session, i.e., $N^* = 1$. $\bar{Q}_{\text{reno}}$ and $\bar{w}_{\text{reno}}$ are the average queue length and window size of the TCP Reno sessions.

Proof Define $\bar{T}_{\text{reno}}$ as the average throughput of Reno sessions, T_{fit} as the throughput of the TCP-FIT session. From Eq. (16) we know $T_{\text{fit}} \geqslant \bar{T}_{\text{reno}}$. We assume that $T_{\text{fit}} > \bar{T}_{\text{reno}}$. Since they share the same bottleneck and have the same queueing delay q, we have the relation of queue length of TCP-FIT session Q_{fit}:

$$Q_{\text{fit}} = T_{\text{fit}} \cdot q > \bar{T}_{\text{reno}} \cdot q = \bar{Q}_{\text{reno}}. \tag{19}$$

But from Eq. (18) and $\alpha \leqslant \bar{Q}_{\text{reno}}/\bar{w}_{\text{reno}}$ we have:

$$Q_{\text{fit}} = \alpha \cdot \bar{w}_{\text{reno}} \leqslant \frac{\bar{Q}_{\text{reno}}}{\bar{w}_{\text{reno}}} \cdot \bar{w}_{\text{reno}} = \bar{Q}_{\text{reno}}.$$

This contradicts with Eq. (19), therefore $T_{\text{fit}} > T_{\text{reno}}$ cannot be true, so $T_{\text{fit}} = T_{\text{reno}}$ and $N^* = 1$. □

Through the proof we found the throughput model of TCP-FIT can keep a fairness with TCP Reno over a no random packet loss link if we set α lower than a threshold. It is easier to implement since we donot need to keep α equal to an exact value. In practical implementations, $\bar{Q}_{\text{reno}}/\bar{w}_{\text{reno}}$ can be approximated by the ratio of buffer size to the bandwidth-delay product (BDP) of the link. It is approximated by $(rtt_{\max} - rtt_{\min})/rtt_{\max}$ in TCP-FIT, where $rtt_{\max}$ and $rtt_{\min}$ are the maximal and the minimal sample values of a TCP session. α is set to $(rtt_{\max} - rtt_{\min})/2rtt_{\max}$. Experimental results show that both the throughput and fairness of the implementation are good when adopting this approximation [18].

According to the above analysis, the throughput model of TCP-FIT has the potential of fully utilizing the network capacity over wireless and high speed network while preserving fairness with TCP Reno. The algorithm is also closely related to delay based TCP algorithms, and we can implement the following delay based TCP achieving the same steady state throughput model as in Eq. (17).

In this proposed algorithm that we will name FAST-FIT, the congestion window is adjusted by:

$$w \leftarrow \min\left\{2w, \frac{1}{2}w + \frac{1}{2}\left(\frac{rtt_{\min}}{rtt_{\text{avg}}}w + \alpha \cdot \beta\right)\right\},$$

where β is a parament which is updated in a manner similar to the congestion window in a Reno session, i.e.,

$$\text{Each } RTT: \quad \beta \leftarrow \beta + 1,$$
$$\text{Each } Loss: \quad \beta \leftarrow \beta/2,$$

and in the steady state $\beta^* = \sqrt{3/2(P+p)}$. Therefore the throughput of FAST-FIT becomes:

$$T_{\text{FAST-FIT}} = \frac{\alpha \cdot \beta^*}{q} = \frac{\alpha}{q}\sqrt{\frac{3}{2(P+p)}},$$

which is identical to Eq. (17). To guarantee that the congestion window of FAST-FIT is no less than that for TCP Reno, w is set to β if $w < \beta$. Therefore, FAST-FIT has the same throughput model Eq. (16) with TCP-FIT.

FAST-FIT uses queuing delay as the primary congestion window control signal and packet loss as the secondary signal to adaptively adjust the parameter β of primary control, and oppositely, TCP-FIT uses packet loss as the primary control signal, and queuing delay as secondary signal to adjust N. FAST-FIT could be considered as a dual implementation of TCP-FIT. Although FAST-FIT and TCP-FIT have identical steady state throughput model, their behavior in the transient state are quite different. As can be seen from the experimental results, FAST-FIT inherits the inherent random packet loss immunity of delay based TCP, which has better performance over high random packet loss networks, whereas TCP-FIT performs better over low random packet loss environments.

5 Experiments

In this section, we show a group of experimental results from academic labs to real world deployment. First, we compare the performance of FAST-FIT with other variants over an emulator based testbed. Secondly, we show the results performed on 254 nodes of PlanetLab test bed distributed over 192 cities in 43 countries. PlanetLab is a group of computers available as a testbed for computer networking and distributed system research. Thirdly, we show performance results over commercial 3G wireless networks: WCDMA of ChinaUnicom, CDMA 2000 of ChinaTelecom, and TD-SCDMA of ChinaMobile.

5.1 Emulation experiments

We first compare the performance of FAST-FIT with other

TCP variants over an emulator based testbed. The setup of the experiments is shown in Fig. 2. In the setup, Linux servers and a client are connected via a Linktropy emulator[1)]. The Linktropy emulator provides an interface of adjusting bandwidth, packet loss rate, delay, delay distribution, and other parameters of the network. In our experiments, FAST-FIT and TCP-FIT are implemented and embedded into Linux kernel (v2.6.31). Other TCP Congestion Avoidance algorithms are available on the Linux kernel v2.6.31.

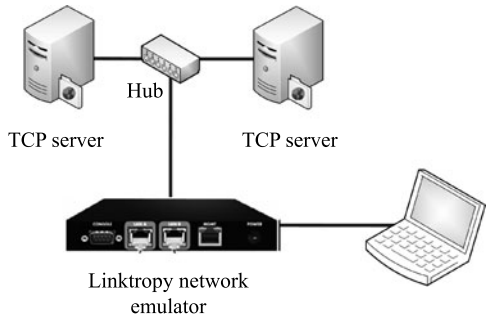


Fig. 2 The setup of emulator testbed

We first investigate the throughput performance for FAST-FIT, TCP-FIT, Reno, CUBIC and Vegas. In our experiment, the link bandwidth of Lintropy was set to 4 Mbps, and the packet loss rate was set from 0.5% to 4%. In each experiment, the propagation delay of the link was varied from 150 ms to 400 ms. As can be seen in Fig. 3, FAST-FIT and TCP-FIT both achieved higher throughput than other TCP CA algorithms. When the random packet loss is at a lower level,

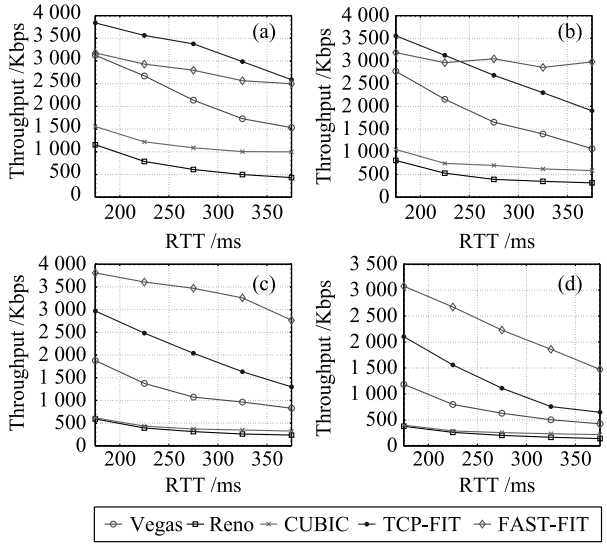


Fig. 3 Bandwidth occupation among competing TCP sessions. (a) Packet loss rate 0.5%; (b) Packet loss rate 1%; (c) Packet loss rate 2%; (d) Packet loss rate 4%

[1)] Linktropy mini2 wan emulator

0.5%, the throughput of TCP-FIT was higher than FAST-FIT. With increasing packet loss rate and RTT, the throughput of FAST-FIT becomes higher than TCP-FIT. In order to further understand this performance difference, we define a network status factor $S = RTT\sqrt{PLR}$ as a quantitative metric of network condition in the paper. As shown in Fig. 4, compared to TCP-FIT, FAST-FIT has a better performance in worse network environments.

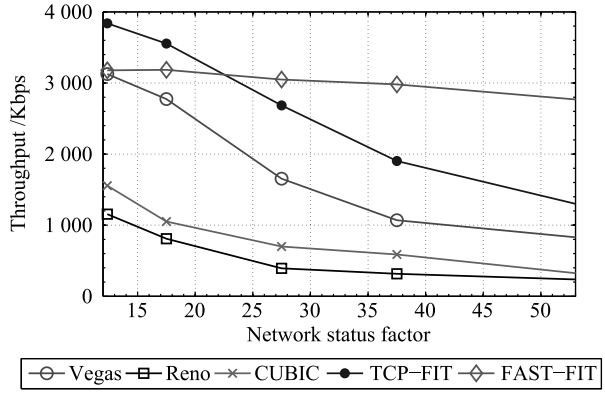


Fig. 4 Throughput performance comparison of TCP variants with different network status factor over emulator testbed

The throughput improvements of FAST-FIT and TCP-FIT to other TCP variants are summarized in Table 2. In the present paper, we define throughput improvement of algorithms A over B as $(T_A - T_B)/T_B$. As shown in Table 2, FAST-FIT achieved a remarkable performance gain compared with other TCP CA algorithms.

Table 2 Throughput improvements on emulator testbed

PLR 0.5%	to Vegas/%	to Reno/%	to CUBIC/%
FAST-FIT	24	300	138
TCP-FIT	460	360	179
PLR 1.0%			
FAST-FIT	66	527	310
TCP-FIT	50	466	266
PLR 2.0%			
FAST-FIT	176	842	707
TCP-FIT	70	480	397
PLR 4.0%			
FAST-FIT	220	890	714
TCP-FIT	75	441	345

We next tested the fairness of FAST-FIT, TCP-FIT, CUBIC, Reno and Vegas when they share the same bottleneck link with Reno. In our experiment, one connection of each of the above TCP algorithms was set up to compete with four TCP Reno sessions. The combined bandwidth was set at 10 Mbps, with a propagation delay of 100 ms and 0% to 2% ran-

dom packet loss rate. In Fig. 5, the gray part of each bar represents the bandwidth occupied by the four TCP Reno sessions, and the black part represents the fifth TCP session. Under perfect network conditions with no packet loss, as shown in Fig. 5(a), both TCP-FIT and FAST-FIT occupied about 20% of the total bandwidth, i.e., it seems as if the Reno sessions were replaced by a fifth Reno session, or the theoretical "fair share". TCP CUBIC, however, occupies up to 25% of the bandwidth, i.e., TCP CUBIC might be overly aggressive and therefore is not TCP Reno friendly. As can be seen from the figure , TCP Vegas only occupied less than 10% of the bandwidth . When the packet loss rate was set to 2%, as illustrated in Fig. 5(b), the combined throughput of all five TCP sessions were only able to occupy up to 70% of the total network bandwidth. TCP-FIT in this case was able to pick up some of the capacity that other Reno sessions left unused. It is important to note that the percentages of the bandwidth that the other Reno sessions used were still comparable to their respective numbers when five Reno sessions were used, indicating that the additional throughput that TCP-FIT was able to "grab" did not come at the expense of the Reno sessions.

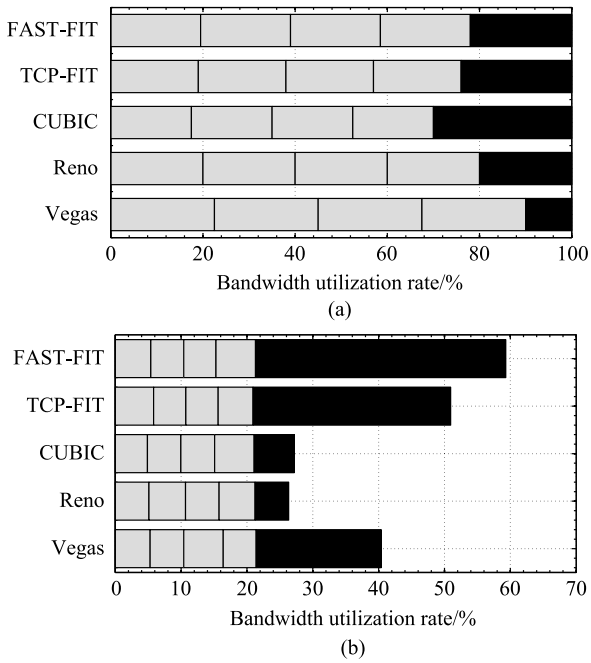


Fig. 5 Bandwidth occupation among competing TCP sessions. (a) Packet loss rate 0%; (b) packet loss rate 2%

5.2 PlanetLab experiments

PlanetLab is a group of computers available as a testbed for computer networking and distributed systems research. We tested and compared the performance of TCP-FIT and FAST-FIT on PlanetLab along with TCP Reno and CUBIC, as well as TCP Vegas. In our experiments, 245 PlanetLab nodes located in 192 cities of 43 countries were used to download video clips from HTTP servers located in San Diego, California, USA. The geographical distribution of these nodes is shown in Fig. 6. These nodes covered 233 ISPs, representing the condition of the current Internet.

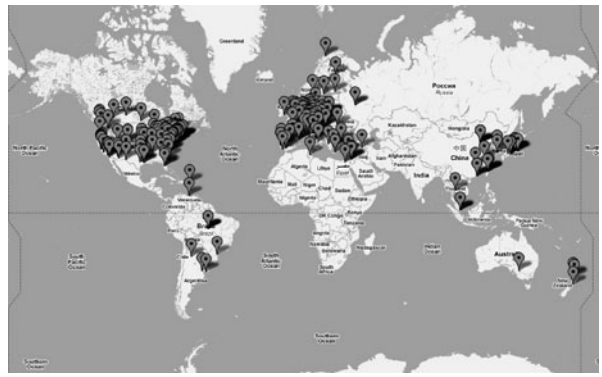


Fig. 6 Geographical distribution of nodes in PlanetLab experiments

As described in Section 2, in TCP-FIT and FAST-FIT, α should be set smaller than the ratio of link buffer size and BDP. This ratio can be approximated by $(rtt_{\max} - rtt_{\min})/rtt_{\max}$ average of each link from the server to planetlab nodes. We used TCP Reno to measure the ratio and its distribution for each planetlab node (Fig. 7). The average value of $rtt_{\max}/(rtt_{\max} - rtt_{\min})$ was 2.25, for the sake of safety, we set $\alpha = 1/5$.

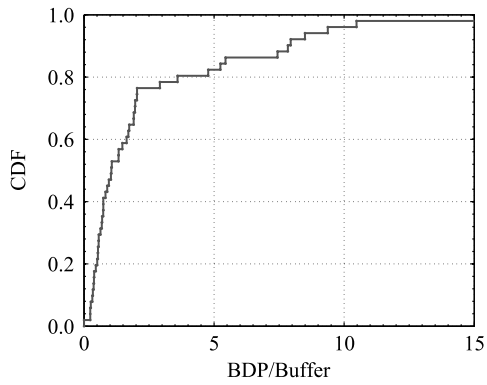


Fig. 7 CDF of $rtt_{\max}/(rtt_{\max} - rtt_{\min})$ on PlanetLab

5.2.1 Throughputs

The throughputs of different TCP CA algorithms on the PlanetLab testbed was studied first. In the experiments, PlanetLab nodes downloaded seven 10-minute video clips encoded at 4 Mbps from an HTTP server located in Orange, California, USA. The average size of these video clips was 300 MB. 245 planetlab nodes simultaneously and repeatedly downloaded these clips for six hours. The cumulative distribution function

(CDF) for the download throughput of these nodes is shown in Fig. 8(a), while the average throughput of these nodes is shown in Fig. 8(b). As shown in the figures, the throughput of FAST-FIT was higher than Vegas, Reno and CBUIC but lower than TCP-FIT. We speculated that this was because many nodes on PlanetLab were connected using high speed networks. To justify this speculation, we applied a modified TCP Vegas with a limited window size smaller than 5 to measure the inherent packet loss rate and propagation delay on each planetlab node. The CDF of the measured packet loss rate and RTT are shown in Fig. 9. As can be seen from the figure, 90% of the PlanetLab nodes experienced less than 1% packet loss rate; 80% of the PlanetLab nodes experienced RTT shorter than 100 ms. It can be calculated from the figure that the average packet loss rate and RTT are 0.7% and 82 ms respectively and the network status factor is 6.8. The network condition of planetlab testbed is relatively good compared to linktropy experiments to the benefit of TCP-FIT according to Fig. (4). This confirmed our speculation.

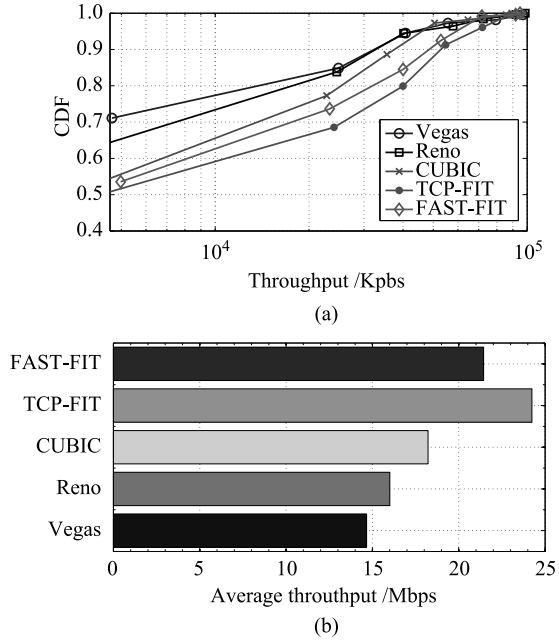


Fig. 8 CDF and average throughput of different TCP variants on planetLab. (a) Throughput CDF; (b) average throughput

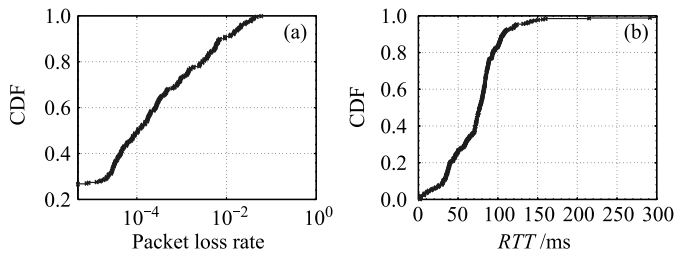


Fig. 9 CDF of (a) packet loss rate and (b) *RTT* of PlanetLab nodess

The average improvement of FAST-FIT and TCP-FIT to TCP Vegas, Reno and CUBIC on planetlab are summarized in Table 3. As can be seen from the table, both FAST-FIT and TCP-FIT achieve remarkable throughput gain compared to other TCP variants.

Table 3 Throughput improvements over the PlanetLab testbed

Improvements	to Vegas/%	to Reno/%	to CUBIC/%
FAST-FIT	46	34	17.7
TCP-FIT	65	51	33

5.2.2 Fairness

Next, the fairness properties of FAST-FIT were investigated on the planetlab testbed. In the experiments, another HTTP server using TCP Reno was used to compete with a competitor server. Both servers had the same set of video clips. During the experiments, the CA algorithm on the competitor server cycled through different algorithm, while nodes on Planetlab nodes simultaneously downloaded video clip files from the two servers. The Reno server and its competitor were located close-by so as to avoid impacts from differences in routing. We used the "tracepath" tool to get the router list from the servers to each planetlab nodes to calculate the router overlap rates, the number of routers shared by the two server divides the total number of routers they used. This rate was about 60% on average and its CDF of paths from the servers to each planetlab nodes is shown in Fig. 10.

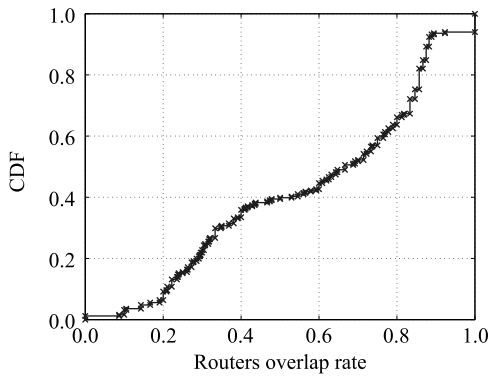


Fig. 10 CDF of throughput of routers overlap rate

The download speed distribution and average of planetlab nodes for the competitor server is shown in Fig. 11. The result of the figure is the same as that of the throughput experiments. The download speed distribution and mean from Reno server is shown Fig. 12. From the Fig. 12 we found, competing with different TCP algorithms, sessions of TCP Reno server have different throughputs. As shown in Figs. 12(a)

and 12(b), the Reno session competing with FAST-FIT had an average throughput similar to TCP CUBIC. It was higher than the session competing with TCP-FIT but lower than with Reno and Vegas, indicating that the fairness of the proposed algorithm was comparable to the widely used CUBIC algorithm.

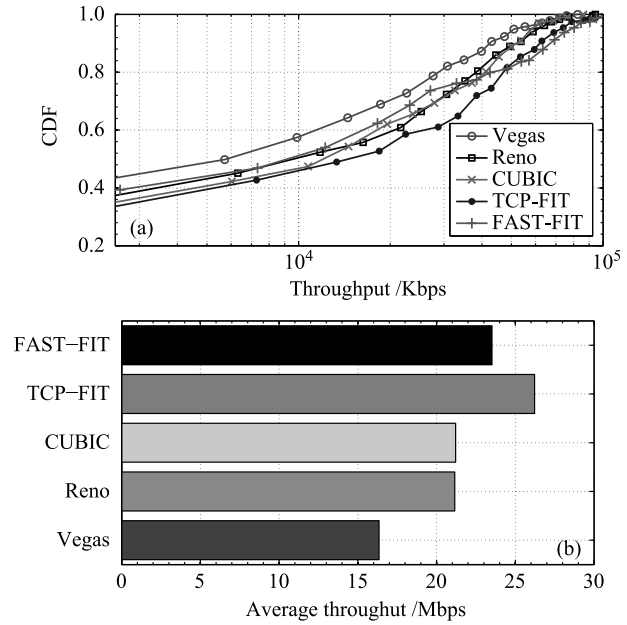


Fig. 11 CDF and average throughput of competitor server. (a) Throughput CDF; (b) average throughput

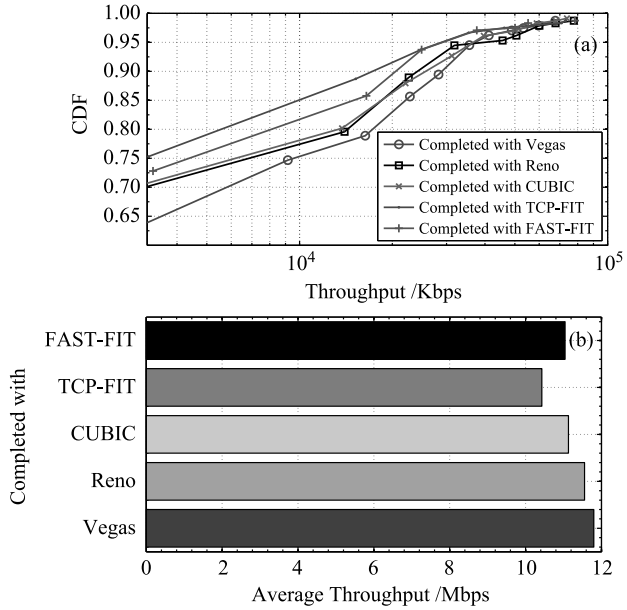


Fig. 12 CDF and average throughput of Reno server. (a) Throughput CDF; (b) average throughput

The average aggregative throughputs for the Reno and competitor servers are plotted in Fig. 13. As shown in the figure, the aggregative throughputs of FAST-FIT and TCP-FIT were still higher than other TCP variants, i.e., the gain in performance was not due to stealing bandwidth from the Reno server. The improvement to the aggregative throughput is summarized in Table 4. It is worth mentioning that, in both the throughput and fairness experiments, TCP Vegas showed poor performance on the planetlab testbed. Although TCP Vegas is robust in random packet loss environments, it cannot work well in wired wide area networks.

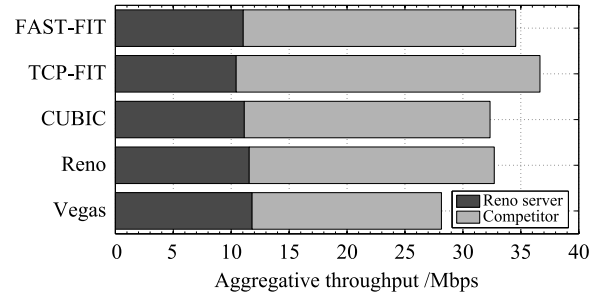


Fig. 13 Aggregative throughput of different TCP variants of Reno and competitor servers on planetLab

Table 4 Improvements of aggregative throughput of Reno Server and Competitor

Improvements	to Vegas/%	to Reno/%	to CUBIC/%
FAST-FIT	22	7	8
TCP-FIT	30	12	13

5.3 Wireless experiments

Finally, we test the algorithm on three different live commercial 3G wireless networks, namely the WCDMA network of ChinaUnicom, the CDMA 2000 network of ChinaTelecom and the TD-SCDMA network of ChinaMobile. The experiments covered all 3G standards operating in the Chinese mobile telecommunication market.

In each experiment, Windows XP laptops were connected to TCP servers using 3G wireless cards. A script was used to automatically and periodically cycle through different TCP algorithms on the server over long durations of time (e.g., seven hours), while the client collected throughput information and other useful data. The period for changing the TCP algorithms was set to about 5–10 minutes, so that 1) the algorithms tested were able to reach close to steady-state performances; 2) the period was consistent with the durations of a reasonably large percentage of the TCP based sessions on the Internet (e.g., YouTube streaming of a single piece of content, synchronizing emails, refreshing one web page, etc.).

The throughput variation and average comparison of different TCP variants over ChinaUnicom WCDMA network are shown in Fig. 14. As shown in the figures, the throughput of FAST-FIT was always higher than the other TCP algorithms

during the 7-hour test period. TCP-FIT had a similar throughput as TCP Vegas which was higher than Reno and CUBIC. The variation of packet loss rate and RTT is plotted in Fig. 15. The average RTT and packet loss rate were 508 ms and 5.4%. Such poor network condition resulted in larger FAST-FIT performance gains.

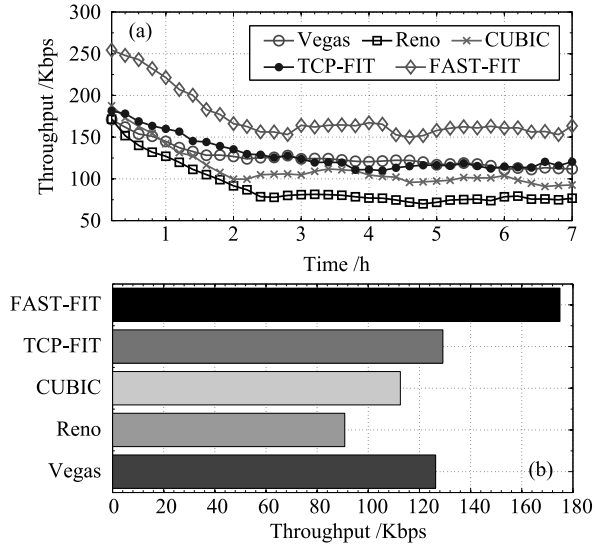


Fig. 14 CDF and average throughput of ChinaUnicom WCDMA network. (a) Throughput variation; (b) average throughput

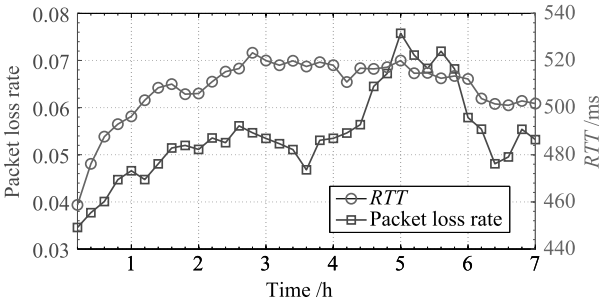


Fig. 15 PLR and RTT variation of ChinaUnicom WCDMA network

The throughput variation and average comparison of different TCP variants over ChinaMobile TD-SCDMA network are shown in Fig. 16. Similar to the experiment results of WCDMA networks, FAST-FIT achieved higher throughput than other TCP variants. The variation of packet loss rate and RTT is plotted in Fig. 17. The average RTT and packet loss rate are 345 ms and 3.4%.

The results for ChinaMobile were similar. However, the results for ChinaTelecom CDMA 2000 were different, and are shown in Fig. 18. The variations of packet loss rate and RTT are plotted in Fig. 19. The average packet loss and RTT were 0.07% and 360 ms respectively. This was better than both China Mobile and China Unicom, especially the random packet loss rate (Fig. 18). In this case, TCP-FIT achieved

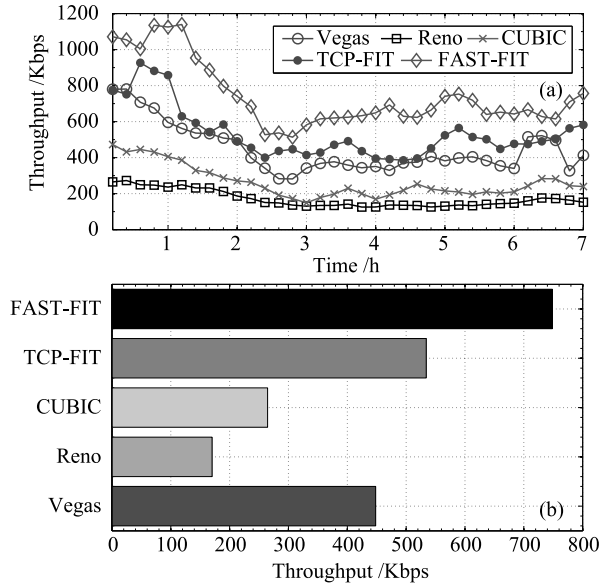


Fig. 16 CDF and average throughput of ChinaMobile TD-SCDMA network. (a) Throughput CDF; (b) average throughput

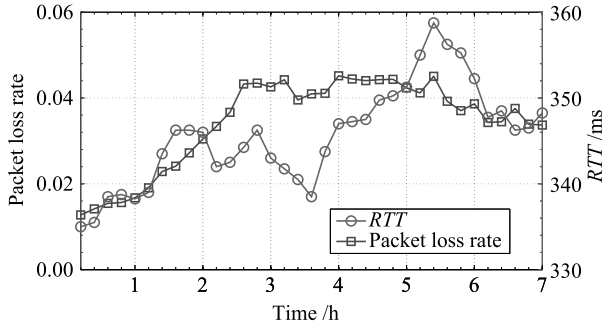


Fig. 17 PLR and RTT variation of ChinaMobile TD-SCDMA network

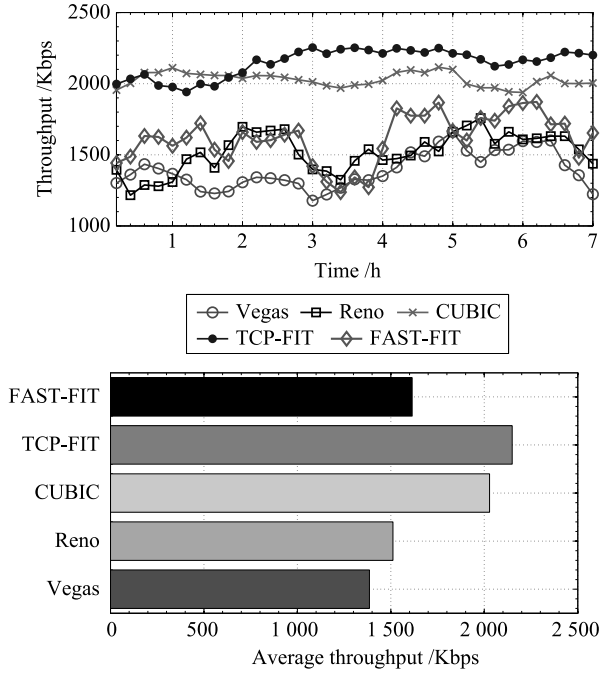


Fig. 18 CDF and average throughput of ChinaTelecom CDMA2000 network. (a) Throughput CDF; (b) average throughput

higher throughput than other TCP algorithms, whereas FAST-FIT was similar to TCP Reno but lower than CUBIC. The relations of TCP throughput and Network Status Factor over the 3G networks are plotted in Fig. 20, which has similar variation trend to the Fig. 4 of emulator experiments.

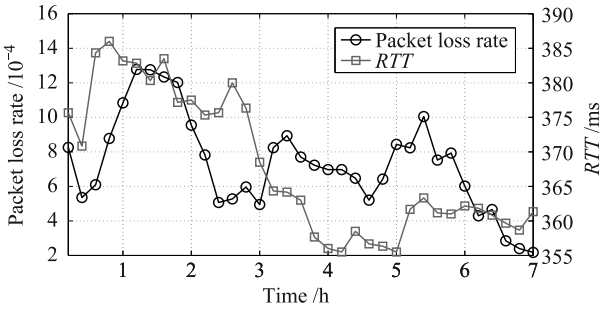


Fig. 19 PLR and RTT variation of ChinaTelecom CDMA2000 network

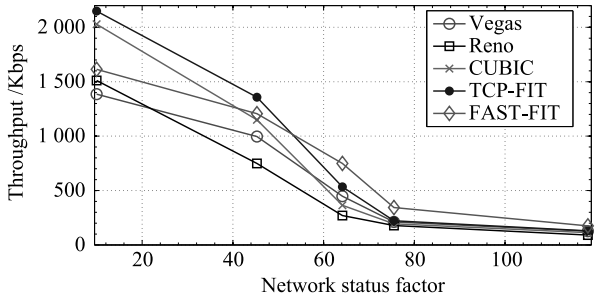


Fig. 20 Throughput performance comparison of TCP variants with different network status factor over 3G networks

The throughput improvement of FAST-FIT and TCP-FIT are summarized in Table 5. The gain for FAST-FIT was up to 3x.

Table 5 Throughput improvements of FAST-FIT and TCP-FIT on 3G Networks

ChinaUnicom	to Vegas/%	to Reno/%	to CUBIC/%
FAST-FIT	38	93	55
TCP-FIT	2	42	15
ChinaMobile			
FAST-FIT	67	340	183
TCP-FIT	19	214	101
ChinaTelecom			
FAST-FIT	16	6.8	–20
TCP-FIT	55	42	6

6 Conclusions

In this paper, we analyze existing TCP CA algorithms from a TCP throughput model perspective. The analysis shows that delay based TCP CA algorithms such as TCP Vegas and FAST TCP have the potential of fully utilizing the bandwidth for both wireless and high speed networks, but they tend not to be fair when sharing bandwidth with other TCP algorithms such as Reno. To overcome the fairness problem of delay based CA algorithm, a novel TCP CA algorithm, named FAST-FIT, is proposed. Experiments results using the Linktropy emulator, the planetlab testbed and commercial 3G mobile wireless network show that the FAST-FIT achieves good performance in both throughput and fairness. Compared with TCP-FIT, the performance of FAST-FIT is better for high packet loss and/or long delay network environments.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Grant Nos. 61202426, 61272350), the National Science Fund for Distinguished Young Scholars of China (61125102), the State Key Program of National Natural Science of China (61133008), the National High Technology Research and Development Program of China (2011AA010502), the National key Technology R&D Program (2012BAK26B02).

References

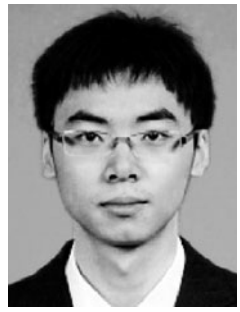
1. Wang J Y, Gao F, Wen J T, Li C, Xiong Z, Han Y X. Achieving TCP reno friendliness in fast TCP over wide area networks. In: Proceedings of the 2014 International Conference on Computing, Networking and Communications. 2014, 100–109
2. Allman M, Paxson V, Stevens W. TCP congestion control. RFC 2581, 1999
3. Wei D X, Jin C, Low S H, Hegde S. FAST TCP: motivation, architecture, algorithms, performance. IEEE/ACM Transactions on Networking, 2006, 14(6): 1246–1259
4. Mascolo S, Casetti C, Gerla M, Sanadidi M Y, Wang R. TCP westwood: Bandwidth estimation for enhanced transport over wireless links. In: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking. 2001, 287–297
5. Mascolo S, Grieco L A, Ferorelli R, Camarda P, Piscitelli G. Performance evaluation of westwood+ TCP congestion control. Performance Evaluation, 2004, 55(1-2): 93–111
6. Fu C P, Liew S C. TCP veno: TCP enhancement for transmission over wireless access networks. IEEE Journal on Selected Areas in Communications, 2003, 21(2): 216–228
7. Floyd S. Highspeed TCP for large congestion windows. RFC 3649, 2003
8. Kelly T. Scalable TCP: improving performance in highspeed wide area networks. SIGCOMM Computer Communication Review, 2003, 33(2): 83–91
9. Tan K, Song J, Zhang Q, Sridharan M. A compound TCP approach for high-speed and long distance networks. In: Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies. APRIL 2006
10. Ha S, Rhee I, Xu L. Cubic: a new TCP-friendly high-speed TCP variant. ACM SIGOPS Operating Systems Review, 2008, 42(5): 64–74
11. Wang J Y, Jiang Y X, Ouyang Y, Li C, Xiong Z, Cai J X. TCP congestion control for wireless datacenters. IEICE Electronics Express, 2013, 10(12): 1–11

12. Fang W, Liang X, Li S, Chiaraviglio L, Xiong N. Vmplanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers. Computer Networks, 2013, 57(1): 179–196
13. Brakmo L S, O'Malley S W, Peterson L L. TCP Vegas: new techniques for congestion detection and avoidance. SIGCOMM Computer Communication Review, 1994, 24(4): 24–35
14. Low S H, Peterson L L, Wang L. Understanding TCP vegas: a duality model. Journal of the ACM, 2002, 49(2): 207–235
15. Tang A, Wang J, Hegde S, Low S. Equilibrium and fairness of networks shared by TCP Reno and Vegas/FAST. Telecommunication Systems, 2005, 30(4): 417–439
16. Budzisz Ł, Stanojević R, Schlote A, Baker F, Shorten R. On the fair coexistence of loss-and delay-based TCP. IEEE/ACM Transactions on Networking (TON), 2011, 19(6): 1811–1824
17. Wang J, Wen J, Zhang J, Han Y. TCP-FIT — a novel TCP congestion control algorithm for wireless networks. In: Proceedings of the 2010 IEEE Globecom Workshop on Advances in Communications and Networks. 2010, 2133–2137
18. Wang J, Wen J, Zhang J, Han Y. TCP-FIT: an improved TCP congestion control algorithm and its performance. In: Proceedings of the 30th IEEE International Conference on Computer Communications, 2010, 2065–2069
19. Wang J, Wen J, Han Y, Zhang J, Li C, Xiong Z. CUBIC-FIT: A high performance and TCP CUBIC friendly congestion control algorithm. IEEE Communications Letters, 2013, 17(8): 1664–1667
20. Tang A, Wang J, Low S H, Chiang M. Equilibrium of heterogeneous congestion control: existence and uniqueness. IEEE/ACM Transactions on Networking, 2007, 15: 824–837
21. Low S. A duality model of TCP and queue management algorithms. IEEE/ACM Transactions on Networking, 2003, 11(4): 525–536
22. Kelly F, Maulloo A, Tan D. Rate control for communication networks: shadow prices, proportional fairness and stability. Journal of the Operational Research Society, 1998, 49(3): 237–252
23. Padhye J, Firoiu V, Towsley D, Kurose J. Modeling TCP throughput: A simple model and its empirical validation. In: Proceedings of the 1998 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication. 1998, 303–314
24. Ha S, Rhee I, Xu L. CUBIC: a new TCP-friendly high-speed TCP variant. ACM SIGOPS Operating Systems Review, 2008, 42(5): 64–74
25. Zhou B, Fu C P, Chiu D M, Lau C T, Ngoh L H. A simple throughput model for TCP veno. In: Proceedings of the 2006 IEEE International Conference on Communications. 2006, 5395–5400
26. Grieco L, Mascolo S. Mathematical analysis of Westwood+ TCP congestion control. IEEE Proceedings: Control Theory and Applications, 2005, 152(1): 35–42
27. Samios C, Vernon M. Modeling the throughput of TCP Vegas. In: Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems. 2003, 31(1): 71–81
28. Xu L, Harfoush K, Rhee I. Binary increase congestion control (BIC) for fast long-distance networks. In: Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies. 2004, 2514–2524

Jingyuan Wang received the PhD degree in 2011 from the Department of computer science and technology, Tsinghua University, China. He is currently an Assistant Professor of School of Computer Science and Engineering, Beihang University, China. His research interest is multimedia communication, datacenter networks, and TCP congestion control.

Jiangtao Wen received the BS, MS, and PhD degrees(with honors), all in electrical engineering, from Tsinghua University, Beijing, China in 1992, 1994, and 1996, respectively. From 1996 to 1998, he was a Staff Research Fellow at the University of California, Los Angeles (UCLA), USA, where he conducted cutting-edge research on multimedia coding and communications. Many of his inventions there were later adopted by international standards such as H.263, MPEG, and H.264. Since 2009, he has been a Professor at the Department of Computer Science and Technology, Tsinghua University. He is a fellow of IEEE.

Yuxing Han received a BE in electrical engineering at Hong Kong University of Science and Technology (HKUST), China in 2006 and obtained her PhD degree at University of California, Los Angeles, USA, in 2011 with research interests in next generation cellular systems, cognitive radio systems, network modeling and compressive sensing algorithms. She is currently at Flora Production Inc., where she is working on next generation network optimization products.



Jun Zhang received the BS degree in computer science and technology from Tsinghua University, China in 2010. He is currently working toward the MS and PhD degrees in computer science and technology in Tsinghua University.

Chao Li received his BS and PhD degrees in computer science and technology from Beihang University, China in 1996 and 2005, respectively. Now he is an associate professor and MS supervisor in the School of Computer Science and Engineering, Beihang University. Currently, he is working on data vitalization and computer vision.

Zhang Xiong received his BS degree from Harbin Engineering University in 1982. He received his MS degree from Beihang University, China in 1985. He is a professor and PhD supervisor in the School of Computer Science and Engineering, Beihang University. He is working on computer vision, wireless sensor networks, information security, and data vitalization.