Modeling and Improving TCP Performance over Cellular Link with Variable Bandwidth

Fengyuan Ren, *Member*, *IEEE*, and Chuang Lin, *Senior Member*, *IEEE*

Abstract—To facilitate a viable evolution of cellular networks toward extensive packet data traffic, the High Speed Downlink Packet Access (HSDPA) technology is introduced. The various link adaptation techniques employed by HSDPA augment the bandwidth variation, which is identified as one of the most important factors resulting in the deterioration of TCP performance. In this paper, we firstly build an analytical model of TCP throughput to explain why the bandwidth variation degrades the TCP performance. Subsequently, a split-connection Window Adaptation TCP Proxy is proposed to improve the TCP throughput in HSDPA networks. To use the precious cellular link resources sufficiently, the length of the queue in Node-B is intentionally kept around the reference value through adaptively adjusting the sending window size of TCP proxy based on the dynamic values of varying bandwidth. Since both the disturbance caused by bandwidth variation and the feedback delay are prone to lead an unstable queue system, the robust sliding mode variable structure control theory is employed to design the proper control law to weaken the impact of noise and delay on the stability of the queue system in Node-B. The theoretical analysis and the enhanced scheme are verified through simulation experiments. The simulation results show that our TCP proxy is able to resist against bandwidth oscillation and improve the cellular link utilization.

Index Terms—TCP performance, link rate variation, variable structure control, HSDPA networks.

1 Introduction

Third generation (3G) wireless cellular networks are increasingly being deployed throughout the world and wireless data services are anticipated to grow rapidly in the coming years, which will probably become a dominant source of traffic load in the 3G cellular networks. Therefore, efficient provisioning of the expected diverse data services is considered a key factor for the success of 3G networks. Since the vast majority of data applications rely on the Transmission Control Protocol (TCP), optimizing TCP performance over 3G networks constitutes a significant research issue.

This issue was first studied through real experiments in [1] and the experimental results show that TCP suffers significant throughput degradation if a wireless link is included in the end-to-end connection. Recently, the impact of wireless link bandwidth variation on TCP performance is specifically examined in [2], [3], [4], [5], [6], and [35].

As is well known, bandwidth oscillation readily occurs in 3G cellular networks, such as UMTS and CDMA2000. On the one hand, the physical link rate always varies due to signal fading and other radio phenomenon in cellular networks. On the other hand, the High Speed Downlink Packet Access (HSDPA) technology is introduced for high speed data access and various link adaptation techniques employed by HSDPA augment the bandwidth variation. In addition, dynamic resource sharing among concurrent data users is required in 3G networks; originally designed for the maximal resource utilization among multiple users, the channel-state-based scheduling mechanisms also result in variations of the bandwidth available for TCP connections in the transport layer. This available bandwidth oscillation was identified as one of the most important factors of TCP throughput degradation [4], [6].

Since such many factors can lead to bandwidth variation, the TCP throughput degradation due to wireless link bandwidth variation is a complicated research issue. Two specific open problems related to this issue come into the focus of academic research: to what extent the performance of transport protocols can be optimized in the presence of bandwidth variation, and to what extent wireless link-level mechanisms for bandwidth adaptation can be designed to take into account the transport protocols running over these links [7]. A better understanding of these problems will certainly help in designing an optimized solution, which in turn will make significant contributions to the performance of user-perceived data applications.

In this paper, we focus on the aforementioned open issues. First, to gain insights into how bandwidth variation degrades the TCP performance, we build an analytical model to accurately estimate the TCP throughput over the wireless link with variable bandwidth. This theoretical analysis demonstrates that spurious timeout is one of the dominant factors in deteriorating TCP performance. Subsequently, based on the split-connection approach, we propose a new TCP enhancement solution called Window Adaptation TCP Proxy to maximize TCP throughput in UTMS-HSDPA networks. Our TCP proxy employs a dynamic window adjusting scheme to adapt effectively to bandwidth variations. The length of the queue associated with the cellular link is purposely kept around the reference value, and then both buffer overflow and empty queue can

- *The authors are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, P.R. China.*
E-mail: {renfy, chlin}@tsinghua.edu.cn.

Manuscript received 26 June 2009; revised 28 July 2010; accepted 1 Oct. 2010; published online 2 Dec. 2010.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2009-06-0253.
Digital Object Identifier no. 10.1109/TMC.2010.234.

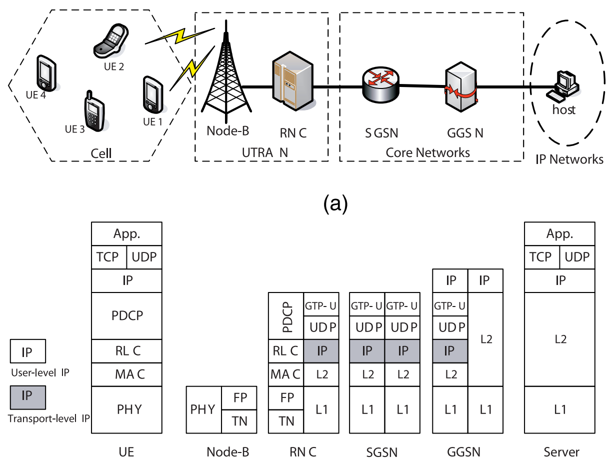

Fig. 1. A reference architecture of UMTS network. (a) An overview of UMTS network topology. (b) Protocol stacks.

be avoided. Thus, the cellular link is sufficiently utilized and TCP throughput is improved. In addition, different from most of the existing work, such as [2], [3], [4], and [33], which are based on the deterministic methods, our investigation emphasizes the random feature of bandwidth variations, where the analysis and design approaches in the stochastic robustness control theory are used.

The remainder of the paper is organized as follows: The simplified architecture of UMTS-HSDPA network, its technical features, and the sliding mode variable structure (SMVS) control theory are introduced in Section 2, along with the related work. In Section 3, the performance model is developed to explain why the bandwidth oscillation degrades the TCP throughput, and the motivation of our design scheme is presented. In Section 4, the architecture of window adaptation TCP proxy and its features are described. Subsequently, the control algorithm is designed to dynamically regulate the sending window size of TCP proxy, and the guidelines toward parameter setting are presented. In Section 5, our TCP Proxy is validated and its performance is compared with the standard TCP through simulation experiments. The impact of parameters on performance is discussed, and the suggestions for parameter configuration are summarized. Finally, the conclusions are drawn in Section 6.

2 Background and Related Works

2.1 UMTS-HSDPA Network

First, we provide an overview of UMTS-HSDPA networks, which is helpful for achieving a considerable comprehension of our investigation in this paper. An illustrative UMTS network topology is depicted in Fig. 1a. The functionality is divided into three main domains: User Equipment (UE), UMTS Terrestrial Radio Access Network (UTRAN), and Core Network (CN). The Core Network provides switching and routing for user traffic, and comprises two basic nodes: Serving GPRS Support Node (SGSN) and Gateway GPRS Support Node (GGSN). The latter provides access to external networks, such as the Internet. UTRAN consists of Node-B and RNC, and is responsible for providing the UEs with access to the CN, as well as managing the radio resources. After receiving IP packets, the SGSN will forward them to the appropriate Node-B through the intermediate RNC, where the IP packet is fragmented into a number of radio frames. The corresponding protocol stacks of UMTS are described in Fig. 1b. TCP/IP protocol suite and applications are located both at the UE and the end host. UDP/IP are used to transport traffic and signaling message among GGSN, SGSN, and RNC. The main function of Packet Data Convergence Protocol (PDCP) is header compression, which improves the spectral efficiency for transmitting IP packets. The RLC protocol runs in both RNC and UE, where it implements regular data link functionality and provides segmentation and retransmission. The technical details can be found in [18].

HSDPA is a new technology standardized by 3G Partnership Project (3GPP) in Release 5 to boost the support for packet switched services as an evolution of UMTS radio interface. The main idea behind the HSDPA channel is to share one single downlink data channel among all users, accompanying some modifications that make it especially effective for packet-based communication. To facilitate the high peak data rates, HSDPA introduces the variable-rate turbo encoding. The peak rate varies from 120 kbps up to 10.8 Mbps (up to 2 Mbps in practice) under different parameter configurations. To increase the link efficiency, the link adaptation is performed by continuously adjusting the modulation and coding parameters in every transmit time interval (TTI). Node-B maintains the per-user queue and schedules transmission on the wireless channel according to the channel states. On the one hand, the various link adaptation techniques employed by HSDPA unavoidably give rise to bandwidth variation. On the other hand, the sharing of the downlink bandwidth among all users also implies high bandwidth variation. As one user is assigned a determined amount of bandwidth depending on the number of users of the cell, the available bandwidth for a connection is affected by the number of users entering and leaving the cell as they all compete for the resources.

2.2 Sliding Mode Variable Structure Control

In this paper, we focus on the impact of bandwidth variations on TCP performance. In the formulation of control theory, the stochastic variation of state variables in a control system can be regarded as disturbance through definite transform. How to design a controller to resist these disturbances and meet performance requirements is one of research issues in robust control theory. In this work, we will apply the approach in Sliding Mode Variable Structure Control to develop a control algorithm to improve TCP performance. We first summarize its basic principle and main features. As evidenced by its name, the structure of SMVS control is not constant but is varying during the control process. The controllers are designed to drive and then constrain the system state within a neighborhood of the switching plane. This approach poses two main features: 1) The dynamic behavior of the system may be tailored by the particular choice of a switching plane. 2) The closed-loop response becomes insensitive to a particular class of uncertainty, including external

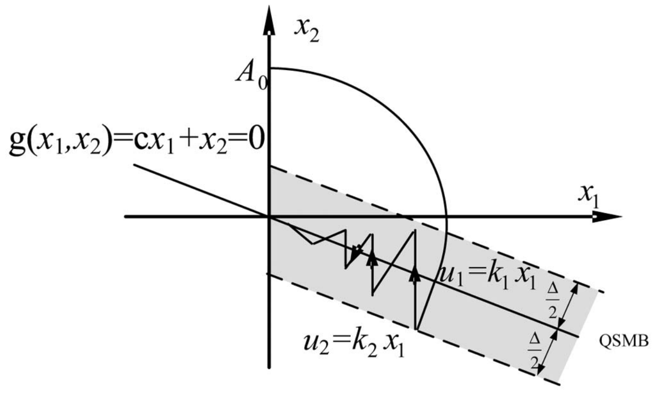

Fig. 2. SMVS control system.

disturbance. For the sake of clarity, we briefly illustrate these features by the system depicted in Fig. 2. Suppose that at some time t_0, the system lies on the initial point A_0, and then moves along one phase trajectory under the control law $u_1 = k_1x_1$. At time t_1, the system reaches and crosses the line $g(x_1, x_2) = cx_1 + x_2 = 0$, the control law is switched to $u_2 = k_2x_1$, and then the system will move along another phase trajectory; but it cannot remain there, since the system will immediately cross the line $g(x_1, x_2) = 0$ again. The control law $u_1 = k_1x_1$ intends to make the system cross the line $g(x_1, x_2) = 0$ at the third time, and so on. The control law is frequently switched back and forth; the system oscillations about the switching line (also called sliding mode line) eventually move toward to the origin. Because the sliding mode line is designed to be independent of the system itself, the SMVS control has an important property: the corresponding steady state of the system is independent of changes in the plant parameters and of external disturbance, which is very suitable for the time-varying bandwidth in this investigation. The design of SMVS controller consists of two key steps: 1) Determine a switching function $g(\vec{x})$ such that the sliding mode on the switching plane $g(\vec{x}) = 0$ is stable. 2) Determine a control law $u(\vec{x})$ such that a reaching condition is satisfied. The technical details of designing the SMVS controller can be found in [17].

2.3 Related Work

In fact, the performance of TCP over wireless networks has been extensively studied in the last decade since the experiment results show that TCP suffers significant throughput degradation when there is a wireless link in the end-to-end connection. Numerous mechanisms have been proposed for improving TCP performance over wireless links, including those in wireless cellular networks. These mechanisms fall broadly into four categories: end-to-end solutions, link layer approaches, split-connection schemes, and cross-layer solutions.

End-to-end solutions assume that it is not possible to rely on the network in order to improve the performance of the transport protocol. In such cases, the endpoints are responsible of performing the necessary changes to ensure a good adaptation. The main advantage is that these solutions can be used in any situation, as they do not depend on the underlying layers. However, the code of either the sender or the receiver (or even both) must be modified, which might be a shortcoming in many cases. Typical schemes belonging to this category are Freeze-TCP [20], Fast-Start [21], Adaptive Start [22], and TCP Veno [23]. In addition, some schemes recommended in RFC 3481 [24], such as *window scale* option, *timestamp* option, and *increase initial windows*, also follow this paradigm.

On the contrary, link layer solutions manage to improve TCP's performance from the network itself. These alternatives rely on determined network elements, which collaborate at the link level in order to reduce the effects of the wireless link. In this case, it is the network, but not the terminals, that is aware of the problems of TCP over wireless links. These solutions reduce and hide the problems from the transport layer, so the endpoints do not need to be aware of the problem. The main advantage is that the code in terminals and servers does not need to be modified. However, most of them likely introduce the relatively large delay variations, which makes more difficult to accurately estimate Round Trip Time (RTT) and results in the spurious timeout. Naturally, all kinds of link enhanced techniques in cellular networks, such as forward error correction and link layer retransmission [25], [26], fall in this category. The other representative schemes are TCP Snoop [27], ACK Regulator [3], Window Regulator [4], etc. TCP Snoop is a TCP-aware link layer protocol. The Snoop agent resides in the base station, and monitors every TCP packet that passes through the connection. It keeps a cache of TCP packets, and it also keeps track of which packets have been acknowledged by the receiver. It detects duplicated ACKs and timeouts, and performs local retransmissions of lost packets. Its main objective is to locally recover random losses, and hide the effects of the wireless link from the TCP. The ACK Regulator runs in an intermediate network element, such as the RNC in the case of UMTS networks. It determines the available buffer space at the bottleneck link, monitors the arrival rate of packets, and controls the release of ACKs to the TCP source so as to prevent buffer overflow. The Window Regulator is very similar to the ACK Regulator. It manages to maintain the queue length at the bottleneck link within the desired range through dynamically computing the adequate receive window value in the ACKs sent from the receiver back to the sender, as well as inserting an ACK buffer to absorb the bandwidth variation, and to prevent buffer underflow and overflow.

Split-connection schemes intend to completely hide the wireless link from the wired portion of the network. They achieve so by terminating the TCP connection at the base station, and establishing another connection from the base station to the wireless nodes. The transport protocol used in the latter can be TCP, a modification of TCP, or any other suitable protocol. These solutions are said to be more efficient than end-to-end solutions, and the endpoints do not need to be aware of the adaptation. However, there is a need to translate from one protocol to the other at the base station, with the resulting overhead. The split-connection approach is initially employed in I-TCP [28], many subsequent works, such as M-TCP [29], TRL-PEP [30], and TCP Proxy [31], follow its basic principle. The main feature of M-TCP is to allow freeze the TCP sender during a disconnection, and then resume the transmission at the rate before disconnection to when the mobile terminal reconnects. Both TCP Proxy and TRL-PEP focus on mitigation of wireless channel errors.

Finally, the cross-layer approach transports state information via the layer boundaries, so that control parameters can be dynamically and adaptively set. Although this approach violates the strict layered model of protocol design, it provides potential chances to contribute an improvement on system performance under various operational conditions. The cross-layer optimization is explored widely in wireless network research. Some investigations adopt this optimization method to improve TCP performance in cellular networks [32], [33], [34]. In [33], the cross-layer signaling is introduced to improve the TCP senders adaptation to varying radio conditions. A two-state TCP-aware scheduler is developed to adapt its channel rate in response to the TCP sending rate in [34].

Only recently, researchers have begun to investigate the impact of the wireless link rate variation on the TCP performance [2], [3], [4], [5], [6], [33]. In [6], Khafizov and Yavuz conducted experiments of TCP over IS-2000 networks with finite burst mode of operation, and showed that bandwidth oscillation severely degrades TCP throughput. In [3], Chan and Ramjee argued that TCP performance degradation caused by bandwidth and delay variation attributes to the difficulty in estimating the bandwidth-delay product (BDP) of the end-to-end path at the TCP source. They also developed two enhanced schemes successively, i.e., ACK Regulator [3] and Window Regulator [4]. Both of them perform well for long-lived TCP flows. However, it does not address the performance issue of short-lived flows, such as HTTP, and some rough estimation might lead to multipacket drops resulting in lower throughput. In [33], Fiorenzi et al. proposed an explicit cross-layer signaling (Radio Network Feedback (RNF)) generated by RNC, which knows the radio channel and link state. When the RNF message is fed back to the TCP sender, its sending window is adjusted according to the value of link rate carried by RNF message. Because the control law does not properly compensate the delay caused by the feedback message, the performance improvement is limited and uncertain. In [5], a rate-adaptive Snoop (RA Snoop) is designed to improve the TCP performance by incorporating a cache-based local recovery and window adaptation mechanism. TCP packets are secretively cached by RA Snoop based on the wireless channel condition, and then retransmitted locally over the rate-controlled lossy wireless links. To alleviate the negative effect of bandwidth variation, RA Snoop also adopts a window adaptation mechanism, which is analogous to the Window Regulator [4], but differs in the concrete implementation algorithms. Although many existing schemes, such as ACK Regulator, Window Regulator, and RA Snoop, can improve TCP performance over the wireless link with changeable rate, they are unsuitable for the architecture of UTMS/HSDPA network because they require the intermediate nodes, such as RNC or Node-B, to be aware of user-level IP packets, while the Node-B node in HSDPA network can only deal with radio frame as illustrated in Fig. 1. We need to develop a new optimization scheme for HSDPA network.

TABLE 1
Configuration of Parameters

Basic Parameters	BS Transmission power	38dBm
	Base station antenna gain	17dBi
	Intra cell interference	30
	Inter cell interference	6
	TTI value	2ms
	Time delay of CQI in TTI	3
Link (Bandwidth, Delay)	Node-B to RNC	(622Mbps, 10ms)
	RNC to SGSN	(622Mbps, 2ms)
	SGSN to GGSN	(622Mbps, 10ms)
	GGSN to host	(100Mbps, 5ms)
User Equipment (velocity, dist. to Node-B)	UE1 (Indoor)	(3km/h,100m)
	UE2 (Pedestrian)	(3km/h,300m)
	UE3 (Vehicular)	(30km/h,500m)
	UE4 (Rural)	(9km/h,700m)
Simulation	Time	150 seconds

3 Throughput Modeling of TCP in Network with Variable Bandwidth

3.1 Bandwidth Variation in HSDPA Networks

To verify the fact of bandwidth variation in HSDPA networks, we conduct some simulation experiment on *ns2* platform. The Enhanced UMTS Radio Access Network Extension [15] (EURANE for *ns2*) module developed by Ericsson Telecommunication is borrowed to simulate HSDPA networks. The network is set up as shown in Fig. 1a, and four user equipments UE1, UE2, UE3, and UE4 are configured in indoor, pedestrian, vehicular, and rural environments, respectively. The values of key parameters are listed in Table 1, and the other parameters are set to the default values. Three typical scheduling algorithms, Round-Robin (RR), Maximum Carrier to Interference ratio (MCI), and Fair Channel-Dependent Scheduling (FCDS), go into effect in turn. The detailed description of MCI and FCDS can be found in [16]. We count the number of packets forwarded by Node-B per second as the value of available bandwidth. The results are presented in Figs. 3, 4, and 5, respectively. The curves demonstrate that the available bandwidth of wireless link in HSDPA networks is variable except that UE1 can obtain a stable share under the support of RR and MCI since it is close to the base station and its channel condition is better.

3.2 Modeling TCP Throughput

In this section, we model the TCP throughput over a wireless link with variable bandwidth, and gain some insights into the real reasons why the bandwidth oscillation degrades the TCP performance; thus, we can design an appropriate scheme to deal with this issue. TCP throughput modeling has been extensively investigated [7], [8], [9], [10], [11]. Most of them assume constant link capacity and calculate TCP throughput in terms of packet loss probability. In [7], Mathis et al. estimate TCP throughput

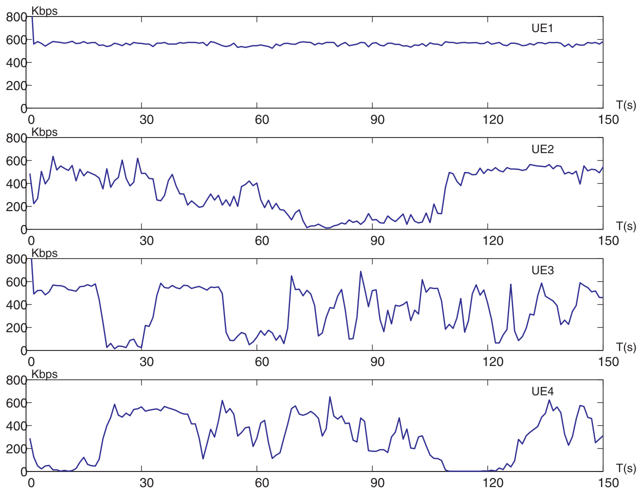

Fig. 3. RR.

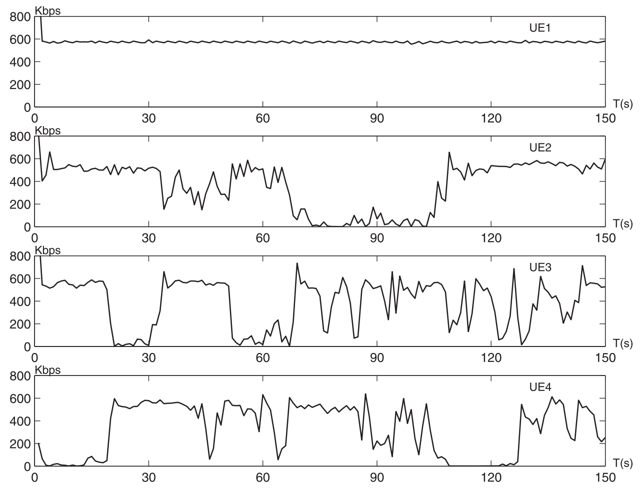

Fig. 4. MCI.

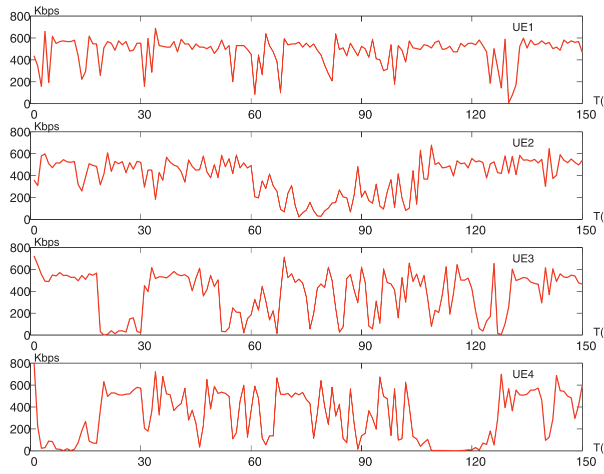

Fig. 5. FCDS.

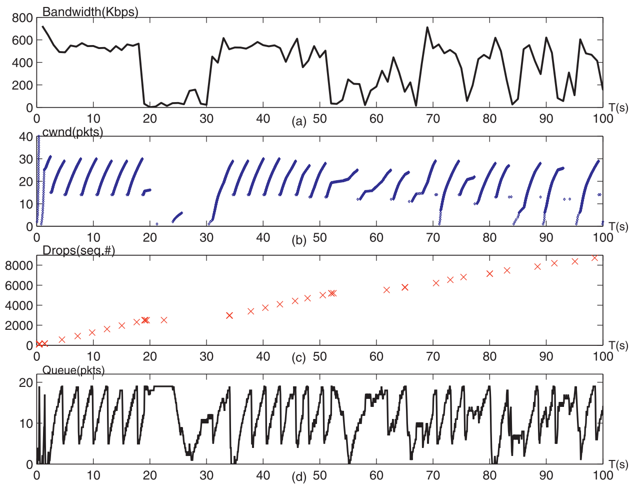

Fig. 6. TCP over the link with variable bandwidth: (a) Variable Bandwidth. (b) Congestion window. (c) Sequence number of dropped packets. (d) Queue evolution.

assuming that the congestion window traverses a perfect periodic sawtooth. In [8], Padhye et al. develop an analytical model of TCP throughput, which can account for the impacts of both fast retransmit and timeout mechanisms on TCP performance. Padhye's model is prevalent in the traditional wired network and has been widely used in many related investigations. In [9], Canton and Chahed propose an analytical model for TCP throughput over UTMS network by extending Padhye's model and inflating the round trip time value to account for the additional delay introduced by the link-level error detection and correction mechanism. However, these models do not capture the specific behavior of TCP over the wireless link with bandwidth oscillation so as to hardly predict the TCP throughput accurately. In [10], Ghaderi et al. develop a fluid model of the steady-state behavior of a TCP session and derive analytical expressions for TCP throughput that explicitly accounts for link rate variability. Although it can provide some insights about impact of the variable link rate on TCP throughput, the assumption of two-state channel restricts its applicability.

To obtain an accurate throughput model of TCP over the wireless link with arbitrarily variable bandwidth, the dynamic behavior of the TCP congestion window should be fully understood. Using the network topology shown in Fig. 1a, we conduct an experiment in which the TCP source using newReno at the host transfers a large file to UE3, Node-B employs the FCDS scheduler, and both radio frames and IP packets are of 500 bytes. To comprehensively understand the effect of bandwidth oscillation on TCP performance, we monitor the congestion window at TCP source, and trace the number of dropped packets and the queue length at Node-B. The results are presented in Figs. 6b, 6c, and 6d, respectively, and the bandwidth variation is redrawn in Fig. 6a. From Fig. 6b, the TCP congestion window follows a much more irregular sawtooth pattern when the bandwidth variation is frequent and drastic. The slow start caused by timeout repeatedly occurs. The models in [8] and [3] can capture the retransmission timeouts (RTOs) triggered by the burst of packet loss. Fig. 6c, however, shows that some slow starts are not caused by bursty loss. A successive two-packet loss only appears at about 20 s and 50 s, and the former results in slow start, but the latter does not. Otherwise, slow start occurs frequently after 70 s, but there is only one packet loss every time. Therefore, the existing analytic models, such as those in [8] and [3], hardly predict the accurate TCP throughput in this case. We infer that bandwidth oscillation probably results in spurious timeouts because when the link bandwidth changes from high to low, the RTT value increases, thereby a low RTO value leads to a spurious retransmission and forces TCP into slow start.

Observing Fig. 6b, the congestion window evolution includes two typical patterns. One is fast retransmission caused by one packet loss due to buffer overflow, and another is slow start caused by timeouts. Therefore, we build our throughput model with the parameters p_1 and p_2 representing the conditional probability of slow start and fast retransmission occurrences under the condition that the packets are dropped due to buffer overflow, respectively. By this definition, $p_1 + p_2 = 1$. Next, we first infer the parameter p_1. The RTT experienced by the incoming packet at time t can be calculated as

$$rtt(t) = \tau + \frac{q(t) + 1}{b(t)}, \tag{1}$$

where τ is the constant propagation delay, $b(t)$ and $q(t)$ denote the bandwidth of the bottleneck link and the queue length, respectively. When $rtt(t + \triangle t) > RTO(t) \approx 2rtt(t)$[1] (where $\triangle t$ denotes the time interval between packet leaving the source and arriving the queue associated with the bottleneck link), the retransmission timeout is triggered. From (1), this condition can be rewritten as

$$\tau + \frac{q(t + \triangle t) + 1}{b(t + \triangle t)} > 2\tau + 2\frac{q(t) + 1}{b(t)}. \tag{2}$$

Since $\triangle t \to 0, q(t + \triangle t) \approx q(t)$; also define $\xi(t) = \frac{\Delta b(t)}{b(t+\Delta t)} = \frac{[b(t)-b(t+\Delta t)]}{b(t+\Delta t)}$, the condition of spurious timeout resulting from the abrupt bandwidth change can be simplified as

$$\xi(t) > 1 + \frac{\tau b(t)}{q(t) + 1}. \tag{3}$$

In fact, the parameter $\xi(t)$ describes the relative decrement of link bandwidth. From (3), the necessary condition of a spurious timeout is $\xi(t) > 1$.

Monitoring the bandwidth and queue length, we can get $b(t)$ and $q(t)$. Using the inequality (3), we can infer the number of possible spurious timeouts N_{to}. In addition, the number of packet drops N_d can be obtained through tracing the queue. When the rule of thumb is used to mandate the buffer size equal to the delay-bandwidth product in practice, the abrupt bandwidth decrease always results in buffer overflow, and subsequently starts a new fast retransmission, then if the timeout timer expires, a slow start will be triggered immediately, which implies that N_{to} is covered in N_d, Therefore, the probability p_1 can be calculated as

$$p_1 = \frac{N_{to}}{N_d}. \tag{4}$$

Although most models, such as [3], [7], and [8], also infer the parameter statistical values from tracing, the inequality (3) needs the values of bandwidth and queue length at the same instant. It is rather troublesome to record them. At the instant when the abrupt decrease of bandwidth leads to a spurious timeout, the queue length $q(t)$ is a random value in section $[0, B]$ (B denotes buffer size). Assume that this random variable complies with a uniform distribution, we can substitute $B/2$ for $q(t)$ in the inequality (3), and yield

$$\xi(t) > 1 + \frac{2\tau b(t)}{B + 2}. \tag{5}$$

The condition (5) is very practicable since we can infer p_1 and p_2 only from the bandwidth profile.

Next, we build the throughput model of TCP over the wireless link with variable bandwidth through extending the models presented in [11], where the throughput of TCP Tahoe and TCP Reno is analyzed in an "ideal" environment, i.e., fixed link capacity and round trip time, and loss only due to buffer overflow. Before proceeding to our model, we make a brief summary of the results in [11].

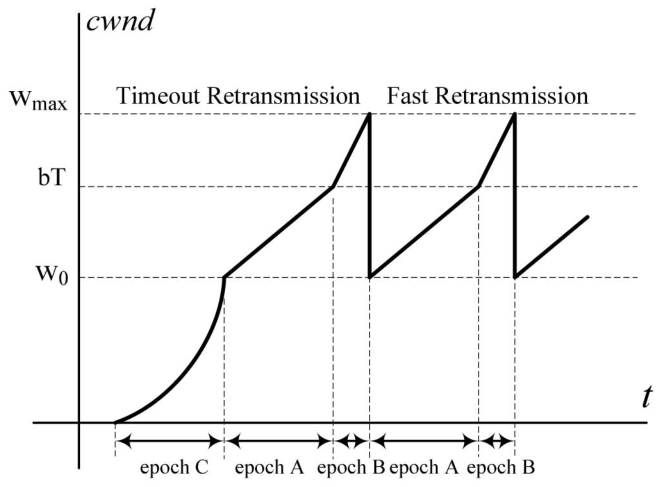

Fig. 7. Congestion window in timeout and fast retransmissions.

As illustrated in Fig. 7, the congestion window of TCP Reno mostly follows the regular sawtooth pattern, going from w_0 to w_{max}, where $w_0 = \frac{w_{max}}{2}$ and $w_{max} = \tau b + B$ (b is the fixed link bandwidth, B is the buffer size, and τ denotes a constant propagation delay). Due to the regularity of the sawtooth, only one such sawtooth is considered; moreover, it is further divided into two epochs, i.e., epochs A and B. In epoch A, the congestion window increases from w_0 to bT (T is the minimum round trip time, namely, $T = \tau + 1/b$), in time t_A with the number of packets delivered n_A. In the epoch B, the congestion window expands from bT to w_{max}, in time t_B with the number of packets sent n_B. Finally, TCP throughput is given by $\frac{n_A+n_B}{t_A+t_B}$, where

$$t_A = T(bT - w_0), \tag{6}$$

$$n_A = \frac{w_0 t_A + \frac{t_A^2}{2T}}{T}, \tag{7}$$

$$t_B = \frac{w_{max}^2 - (bT)^2}{2b}, \tag{8}$$

$$n_B = bt_B. \tag{9}$$

1. TCP uses an exponentially weighted moving average algorithm to estimate an average RTT and the variation of RTT. The RTO value is then obtained from these two statistics. Here, we approximate RTO $\approx$ 2RTT, which is a conservative estimation just like that in the initial implementation of TCP.

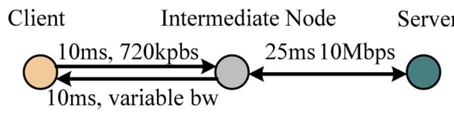

Fig. 8. Simple simulation topology.

TCP Tahoe has an additional epoch C where the congestion window grows exponentially from 1 to w_0, whose value is half of the window size when the last packet loss. Thus, the TCP throughput is calculated by $\frac{n_A+n_B+n_C}{t_A+t_B+t_C}$, where

$$t_C = [log_2(w_0+1)]T, \tag{10}$$

$$n_C = w_0. \tag{11}$$

In fact, the congestion window evolutions of TCP Tahoe and TCP Reno in [11] correspond to a typical slow start and fast retransmission shown in Fig. 6b, respectively. Moreover, a slow start phase in Fig. 6b is always born of a fast retransmission, which implies $w_0 = \frac{w_{max}}{2}$. To capture the highly variable congestion window behavior of a TCP source under bandwidth variation, we make definitions as follows:

$$\tilde{b} = \sum_i^N b_i/N, \tag{12}$$

$$\tilde{T} = \tau + 1/\tilde{b}, \tag{13}$$

$$\tilde{w}_{max} = \sqrt{\sum_i^N w_{imax}^2/N} = \sqrt{\sum_i^N (b_i\tau + B + 1)^2/N}, \tag{14}$$

where N and b_i are the total number of bandwidth change and the values of each cycle, they can be inferred from the bandwidth profile.

Replacing b , T, and w_{max} in (6-11) by $\tilde{b}$, $\tilde{T}$, and $\tilde{w_{max}}$, the average throughput of TCP over the wireless link with variable bandwidth can be approximated by a weighted combination of the typical slow start and fast retransmission to be

$$T_h = \frac{p_1(n_C+n_A) + (1-p_1)n_A + n_B}{p_1(t_C+t_A) + (1-p_1)t_A + t_B}. \tag{15}$$

To validate our model, we use a simplified network topology shown in Fig. 8 to conduct an experiment on the ns2 simulation platform. The link between client and intermediate node consists of two simplex links. One from client to intermediate node has fixed bandwidth, i.e., 720 bpks, and the bandwidth of the simplex link from client to intermediate node is driven by the trace files obtained in Section 3.1. Table 2 lists the simulation results and the estimated throughput from Padhye's and our models. We use the absolute value of relative error to evaluate the accuracy of model, namely,

$$Accuracy = \frac{|T_a - T_e|}{T_a}, \tag{16}$$

where T_a is the actual throughput, and T_e is the estimated throughput.

From Table 2, we can clearly see that both Padhye's and our analytical models can accurately predict TCP throughput when the bandwidth varies in a small range, or even can be approximated as a constant, such as in items $UE1_{rr}$ and $UE1_{mci}$. However, in most conditions where the bandwidth varies drastically, Padhye's model overestimates TCP throughput but our model can estimate relatively accurate results and improve the estimation accuracy by one order of magnitude, which implies that the spurious timeout caused by the abrupt bandwidth decrease is one of the dominant factors degrading TCP performance. A solution to improve TCP performance should reduce or avoid the occurrence of spurious timeouts. Based on this understanding, in the next section, we will present a solution that disables the slow start and the classical AIMD mechanism in the standard TCP to achieve this goal.

TABLE 2
Actual and Estimated Throughput

Item	Actual Throughput (pkts/s)	Estimated Throughput (pkts/s)		Accuracy	
		Padhye's	Our	Padhye's	Our
$UE1_{rr}$	138.6	135.6	137.8	0.018	0.002
$UE1_{mci}$	141.6	142.1	139.4	0.003	0.016
$UE1_{fcds}$	109.6	112.2	109.1	0.024	0.004
$UE2_{rr}$	77.74	86.80	76.1	0.117	0.021
$UE2_{mci}$	85.85	98.58	82.94	0.148	0.034
$UE2_{fcds}$	92.80	95.88	91.47	0.033	0.014
$UE3_{rr}$	86.36	96.74	82.90	0.120	0.040
$UE3_{mci}$	89.00	96.47	87.57	0.084	0.016
$UE3_{fcds}$	79.66	86.54	80.12	0.086	0.005
$UE4_{rr}$	69.79	79.51	68.26	0.139	0.028
$UE4_{mci}$	75.59	84.26	76.63	0.115	0.026
$UE4_{fcds}$	72.72	76.47	72.00	0.052	0.010

4 Improving TCP Performance

IETF has published RFC 3135 [12] to discuss various aspects about performance enhancement proxies (PEPs). The consensus is that a PEP should be employed to mitigate link-related performance degradations. There have been many types of PEPs, and different PEPs aim at optimizing different performance metrics. The technical objective in this work is to adapt the dynamic bandwidth through adjusting the sending window in the proxy, thereby to maximize TCP throughput. Therefore, our performance enhancement proxy is called Window Adaptation TCP Proxy. As the split-connection scheme invented in [13] has been widely used in commercial cellular networks [14], we will inherit this basic mechanism to develop our Window Adaptation TCP Proxy.

4.1 Architecture

In HSDPA, per-flow queue is moved to Node-B from RNC. Referring to the protocol stacks in Fig. 1b, we can find that Node-B only deals with radio frame and is unaware of user-level IP packets; therefore, most existing enhanced schemes, such as ACK Regulator [3] and Window Regulator [4], can not work normally. In fact, it should be considerably

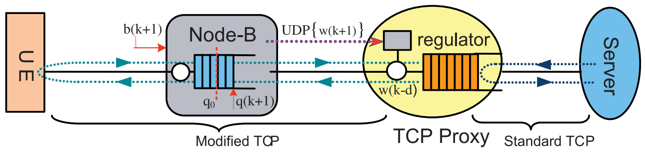

Fig. 9. Window adaptation TCP proxy.

prudent to maintain ACK queue in RNC just like ACK/Window Regulator even without HSDPA evolution because the user-level IP packets are not transparent to RNC and the overhead of drawing/reassembling them from/into transport-level packets is considerable. To avoid overhead and strictly comply with the layered structure of the protocol stack, our split-connection TCP proxy is located between GGSN and the external networks. For the convenience of discussion, its framework is illustrated in Fig. 9. Two different TCP connections are established when a UE attempts to establish a connection to one of the Internet servers: one between the UE and the proxy and the other between the proxy and the server. Upon receipt of data request, the server starts to send the packets under the control of the standard TCP congestion control algorithm. All transferred data packets are received by the proxy, where they are buffered to be forwarded to the UE as soon as possible. The transport protocol between the proxy and the UE needs to be customized to adapt to the dynamic changes of the wireless link bandwidth. Because a modified TCP protocol requiring changes at the mobile terminals might be less attractive in many deployment situations, here we make some modifications only on the proxy. For the TCP connection on the wireless link, the proxy is a sender. Some schemes and algorithms related to congestion window adjustment in the standard TCP can be disabled in the modified TCP protocol. The sending window size is dynamically regulated according to the available wireless bandwidth and the queue length in the Node-B which can be obtained because Node-B is responsible for link adaptation function and maintains a dedicated queue for each flow. It is notable that the calculation of sending window size is performed in Node-B, but the actual regulator resides in the proxy; thus, a UDP connection needs to be established between Node-B and the proxy to periodically convey the update values of sending window size to the proxy as shown in Fig. 9. Intuitively, if the available bandwidth of the bottleneck link is known, it is straightforward to set the window size of the proxy using a back-of-envelope calculation of delay-bandwidth product, but the extensive local retransmission mechanisms employed by Node-B to correct or recover most of corruption loss caused by the unreliable wireless link introduce the relatively large delay variations, which impede to accurately estimate the end-to-end delay between UE and the proxy. Therefore, it makes sense to design an algorithm independent of the end-to-end delay to determine the window size of the proxy. The observable and controllable state variable in the Node-B queue system, i.e., the instantaneous queue length, should be helpful to achieve this goal.

If the instantaneous queue length in Node-B is always kept around the small reference value through dynamic adjustment of sending window size, the perfect utilization of wireless link would be reached because the link is not wasted due to empty queue. In addition, the system can accommodate more spontaneous bursts without dropping packets, and the queuing delay experienced by flow will be reduced. To achieve this goal, we need to design a stable and effective algorithm to control the sending window size, which will be discussed in detail in the next section. Since the congestion control algorithms in the standard TCP, including slow start, fast retransmission, fast recovery, and timeout retransmission, are disabled in our modified TCP running on the proxy, most possible factors resulting in performance degradation over wireless cellular links, such as spurious timeouts, etc., are naturally eliminated. In addition, the reasons that the standard TCP is ineffective in dynamic wireless link can also be partly attributed to the sluggish adjustment of congestion window, which does not timely catch up the varying bandwidth so as to waste the precious wireless bandwidth. Our modified TCP can promptly adjust the sending window to match the instantaneous available bandwidth so that the wireless link would be considerably utilized. Notice that the recovery strategy in the standard TCP will be retained, namely, three duplicate ACKs trigger the retransmission of the corresponding packet, but it does not affect the adjustment of the sending window.

4.2 Algorithm Design

As aforementioned, the technical goal in our solution is to dynamically adjust the sending window in the TCP proxy according to the changeable available bandwidth to keep the queue length in Node-B around the reference value, which is equivalent to designing a classical regulation system in control field. The main task is to design a proper control law, which regulates the control variable (i.e., the sending window) to drive the controlled system (i.e., the queue in Node-B) to move toward the fixed equilibrium point (i.e., the reference queue length or the origin in Fig. 2).

The SMVS controller consists of two parts, namely, the switching function and the corresponding control algorithm. Before determining them, we first build a discrete-time stochastic model to describe the dynamic behavior of the system. The time is divided into steps, which correspond to the intervals in which the update messages are transmitted on the UDP connection. Assume that congestion always occurs on the wireless cellular links. Let $w(k)$ denote the sending window size in step k, and $b(k)$ and $q(k)$ denote the bandwidth of radio channel and the instantaneous queue length in Node-B at the end of the kth step, respectively. Defining the duration of each step as T_s, we have

$$q(k+1) = q(k) + \gamma w(k-2d) - b(k)T_s, \tag{17}$$

where $d = \lceil \frac{\tau_p}{T_s} \rceil$, and τ_p is the propagation delay between Node-B and the proxy. The coefficient γ reflects the difference in length between radio frames and IP packets. The variable $q(k)$ can be readily obtained through sampling the queue on Node-B, but it is hard to get $b(k)$ straightforwardly. Since Node-B knows the status of different components, including scheduler and coder, it is likely to record the related state information, and then calculate the value of $b(k)$. The second approach is to infer the available bandwidth $b(k)$ by the queue evolution equation $b(k) = \{q(k-1) - q(k) + a(k)\}/T_s$, where $a(k)$ is the number of arriving frames in the sampling interval. In our implementation, the latter is employed. Note that it takes d steps to convey the update values of sending window size to the proxy, and the packets sent by the proxy take another d steps to arrive at Node-B; therefore, $w(k-2d)$ is used in (17). Define

$$y(k) = q(k) - q_0, \tag{18}$$

where q_0 is the reference queue length in Node-B, and

$$b(k) = \bar{b} + \eta(k)/T_s, \tag{19}$$

where $\bar{b} \leftarrow \alpha\bar{b} + (1-\alpha)b(k)(0 < \alpha < 1)$, namely, $\bar{b}$ is a weighted moving average of bandwidth $b(k)$, and $\eta(k)$ is a zero-meaned random variable. Let

$$x(k) = \gamma w(k) - \bar{b}T_s. \tag{20}$$

Then, the discrete system (17) can be transformed to

$$y(k+1) = y(k) + x(k-2d) - \eta(k). \tag{21}$$

In the equivalent system (21), $\eta(k)$ can be regarded as an external disturbance. Next, we design a robust SMVS controller to keep the queue length $q(k)$ around the reference value through adjusting the sending window size $w(k)$ in the TCP proxy, and eventually improve the utilization of the precious cellular link and maximize TCP throughput. Let the switching function $s(k)$ be

$$s(k) = y(k+2d). \tag{22}$$

In [17], a reaching law in a continuous system is introduced as follows:

$$s(\dot{k}) = -\delta s(t) - \varepsilon \operatorname{sgn}(s(k)) \qquad \delta > 0, \varepsilon > 0. \tag{23}$$

Using $s(\dot{k}) = \frac{s(k+1)-s(k)}{T_s}$ (where $T_s > 0$ is the sampling period), we can directly obtain an equivalent form of the reaching law in a discrete system

$$s(k+1) - s(k) = -\delta T_s s(k) - \varepsilon T_s \operatorname{sgn}(s(k)), \tag{24}$$

where the parameter $\delta > 0$ and $\varepsilon > 0$. Moreover, the inequality $1 - \delta T_s > 0$ must be held to guarantee that the system trajectory will move monotonically toward the switching line $s(k) = 0$ and cross it in finite time from any initial state. Once the trajectory has crossed the switching line for the first time, it will cross this line again in each successive sampling period, resulting in a zigzag motion about the switching line. Subsequently, the system motion will be constrained in the limited region. From (21) and (22), the incremental change in $s(k)$ can also be expressed as

$$s(k+1) - s(k) = x(k) - \eta(k+2d). \tag{25}$$

Substituting it into the reaching law (24) gives

$$\begin{aligned} s(k+1) - s(k) &= x(k) - \eta(k+2d) \\ &= -\delta T_s s(k) - \varepsilon T_s \operatorname{sgn}(s(k)). \end{aligned} \tag{26}$$

Solving for $x(k)$ gives the control law

$$x(k) = -\delta T_s s(k) - \varepsilon T_s \operatorname{sgn}(s(k)) + \eta(k+2d). \tag{27}$$

Since $\eta(k+2d)$ is uncertain, the control law in this form cannot be implemented. To solve this problem, $\eta(k+2d)$ can be replaced by a constant η_c as long as it is sufficiently conservative to maintain the reaching condition. Then, the control law becomes

$$x(k) = -\delta T_s s(k) - \varepsilon T_s \operatorname{sgn}(s(k)) + \eta_c. \tag{28}$$

Next, we determine η_c to meet the requirement of the reaching condition. It is reasonable to assume that the bounds of $\eta(k+2d)$ are known, and given by

$$\eta_l < \eta(k+2d) < \eta_u. \tag{29}$$

Substituting (28) into (26), we have

$$s(k+1) - s(k) = -\delta T_s s(k) - \varepsilon T_s \operatorname{sgn}(s(k)) + \eta_c - \eta(k+2d). \tag{30}$$

To meet the reaching condition $[s(k+1) - s(k)]\operatorname{sgn}[s(k)] < 0$ (see in [17]), the choice of η_c is done to ensure that the sign of the increment of $s(k)$ is opposite to the sign of $s(k)$. A practical choice is to set $\eta_c = \eta_l$ as $s(k) > 0$ and $\eta_c = \eta_u$ as $s(k) < 0$. Further define

$$\eta_1 = \frac{\eta_u + \eta_l}{2}, \tag{31}$$

$$\eta_2 = \frac{\eta_u - \eta_l}{2}. \tag{32}$$

η_c can be written in the following compact form:

$$\eta_c = \eta_1 - \eta_2 \operatorname{sgn}[s(k)]. \tag{33}$$

In addition, $s(k)$ in control law (28) should also be determined, but $s(k) = y(k+2d) = q(k+2d) - q_0$, and $q(k+2d)$ is unknown in the kth step. From (21), we can use its estimation $\hat{y}(k+2d)$ to predict its value as follows:

$$\begin{aligned} y(k+2d) \simeq \hat{y}(k+2d) &= \hat{y}(k+2d-1) + x(k-1) \\ &= y(k) + \sum_{i=1}^{2d} x(k-i). \end{aligned} \tag{34}$$

Using (22), (28), (33), and (34), a practicable control law can be completely expressed as

$$\begin{aligned} x(k) = \eta_1 &- \delta T_s \left[y(k) + \sum_{i=1}^{2d} x(k-i) \right] \\ &- [\eta_2 + \varepsilon T_s] \operatorname{sgn} \left[y(k) + \sum_{i=1}^{2d} x(k-i) \right]. \end{aligned} \tag{35}$$

Substituting (18), (19), and (20) into (35), the adjusting algorithm of the sending window size in the proxy is obtained as

$$\begin{aligned} w(k) = \Bigg\{ \bar{b}T_s + \eta_1 &- \delta T_s \left[q(k) - q_0 - 2d\bar{b}T_s + \gamma \sum_{i=1}^{2d} w(k-i) \right] \\ &- [\eta_2 + \varepsilon T_s] \operatorname{sgn} \left[q(k) - q_0 - 2d\bar{b}T_s + \gamma \sum_{i=1}^{2d} w(k-i) \right] \Bigg\} \Bigg/ \gamma, \end{aligned} \tag{36}$$

where $\eta_l = \min_k\{[b(k) - \bar{b}]T_s\}$ and $\eta_u = \max_k\{[b(k) - \bar{b}]T_s\}$.

For the TCP connection on the wired link, the TCP proxy is a receiver. Without any control, the server will utilize the abundant bandwidth to rapidly send packets, and a lot of packets occupy the buffer of the TCP proxy, so that the end-to-end delay increases and some packets for real-time interactive applications may become stale. In order to avoid this phenomenon, we employ the advertised window, which is inserted into the corresponding field of ACKs, to limit the sending rate of the server. The objective is also to keep the queue length around the reference value. Here, we assume the time-varying sending window of the TCP proxy as a stochastic process. Following the same way, we can readily keep a stable TCP proxy queue system.

4.3 Parameter Setting

To determine the proper parameter settings, we first introduce a proposition.

Proposition 1. *After the queue system driven by the SMVS controller begins the zigzag motion around the switching line, the queue length in Node-B will be constrained in the following range:*

$$\left\{ q(k) \middle| |q(k) - q_0| < \frac{\varepsilon T_s}{(1 - \delta T_s)} \right\}. \tag{37}$$

Proof. Once the system enters the quasisliding mode (QSM), its trajectory will stay within the state region, where every state x satisfies the following condition:

$$\left\{ x \middle| -\frac{\Delta}{2} < s(x) < \frac{\Delta}{2} \right\},$$

where Δ is the width of the region called QSM Band (QSMB) as shown in Fig. 2.

From the reaching law (24), we have

$$s(k+1) = (1 - \delta T_s)s(k) - \varepsilon T_s \mathrm{sgn}(s(k)). \tag{38}$$

Since the reaching condition requires $(1 - \delta T_s) > 0$, the sign of the first right-hand term in (38) is the same as the sign of $s(k)$, and the sign of the second right-hand term is opposite to that of $s(k)$. By the definition of quasisliding mode (see in [17]), the sign of $s(k+1)$ must be opposite to that of $s(k)$. Thus, when $s(k+1) > 0$, we have

$$s(k+1) = (1 - \delta T_s)s(k) + \varepsilon T_s > 0, \tag{39}$$

namely,

$$s(k) > -\frac{\varepsilon T_s}{(1 - \delta T_s)}. \tag{40}$$

While $s(k+1) < 0$, we also have

$$s(k) < \frac{\varepsilon T_s}{(1 - \delta T_s)}. \tag{41}$$

From (18), (22), (40), and (41), we can readily obtain

$$q_0 - \frac{\varepsilon T_s}{(1 - \delta T_s)} < q(k) < q_0 + \frac{\varepsilon T_s}{(1 - \delta T_s)}. \tag{42}$$

□

Remark. Inequality (42) shows that the instantaneous queue length $q(k)$ on Node-B will fluctuate around the reference value q_0 with the amplitude $\frac{\varepsilon T_s}{(1-\delta T_s)}$ after the queue system under the control (36) enters the stable quasisliding mode motion. As we know, the actual buffer size B is limited, i.e., $0 \le B \le B_{max}$. When $q_0 < \frac{\varepsilon T_s}{(1-\delta T_s)}$, the queue may become empty, and then the precious wireless channel is wasted; otherwise, when $q_0 + \frac{\varepsilon T_s}{(1-\delta T_s)} > B_{max}$, the buffer overflows, and then the incoming packets will be dropped. These two unsatisfactory cases can be avoided through configuring the parameters properly as follows:

$$\left\{ (T_s, \varepsilon, \delta) \middle| \begin{array}{c} \frac{\varepsilon T_s}{(1-\delta T_s)} < \min[q_0, B_{max} - q_0] \\ \delta > 0, \varepsilon > 0, 1 - \delta T_s > 0 \end{array} \right\}. \tag{43}$$

Actually, the width of QSMB defined by the system (17) and (36) is $2\frac{\varepsilon T_s}{(1-\delta T_s)}$, i.e., $\Delta = 2\frac{\varepsilon T_s}{(1-\delta T_s)}$. For a certain q_0, it is better that the width of QSMB (Δ) is narrower. It is seen that Δ decreases with the parameters (ε, δ, and T_s) decreasing. To reduce Δ, these three parameters can be fixed at the values as small as possible. However, from control law (36), we can see that adjustment in each step is rather trivial if δ is too small. Then, the system becomes unresponsive so that it will take long time to reach the stable state. In addition, if the sampling period T_s is too short, the frequent message exchange will result in too much communication overhead. It is necessary to make a reasonable trade-off under the constraint (43). In the next section, we will discuss the impact of parameters through experiments. Moreover, combined with the restrictions of actual systems, the practicable guidelines toward parameters setting will be summarized.

5 Performance Evaluation and Impact of Parameters

5.1 Performance Evaluation

We have implemented the Window Adaptation TCP Proxy on the ns2.29 simulator and conducted the simulation experiments to verify the proposed algorithm and evaluate its performance. Referring to Fig. 1a, our TCP Proxy can only be inserted between GGSN and IP networks. The propagation delay between GGSN and TCP proxy is fixed at 3 ms and the other parameters are configured according to the values listed in Table 1, then the propagation delay between Node-B and TCP proxy is 25 ms. Assume that the cellular link is only a bottleneck, the queuing delay is not introduced in other components. Let $T_s = 25$ ms, i.e., $d = 1$. The size of radio frames and IP packets is 40 bytes and 512 bytes, respectively. (i.e., the parameter $\gamma = 13.0$), and let $\alpha = 0.85$, $\delta = 35$, $\varepsilon = 20$, and $q_0 = 100$, the buffer size of per-flow queues in Node-B is 200 radio frames, which is approximately equivalent to 15 IP packets. As mentioned in Section 4.3, the queue system may fluctuate around the reference value with the amplitude $\frac{\varepsilon T_s}{(1-\delta T_s)}$. Since $\varepsilon T_s/(1 - \delta T_s) = 20 \times 0.025/(1 - 35 \times 0.025) = 40$ and $\min[q_0, B_{max} - q_0] = 100$, in other words, the constraint (43) is satisfied; therefore, such parameter settings are proper. As aforementioned reasons, most of the existing schemes, such as ACK Regulator and Window Regulator,

TABLE 3
Statistic Results of Throughput of Various Schemes

Item	TH of Proxy(Kbps)		TH of TCP(Kbps/s)		TH of I-TCP(Kbps/s)	
	Average	Std. dev.	Average	Std. dev.	Average	Std. dev.
$UE1_{rr}$	560.3	3.31	543.9	5.82	551.2	7.93
$UE1_{mci}$	561.2	6.52	549.1	10.96	549.0	6.03
$UE1_{fcds}$	427.2	4.62	415.9	4.66	416.5	6.10
$UE2_{rr}$	340.9	3.10	311.8	6.15	316.3	4.55
$UE2_{mci}$	310.0	5.26	294.0	4.27	291.1	5.15
$UE2_{fcds}$	333.3	3.50	290.0	5.58	292.3	6.14
$UE3_{rr}$	336.2	5.68	315.6	4.26	310.8	6.97
$UE3_{mci}$	349.2	4.23	316.9	15.88	318.9	8.08
$UE3_{fcds}$	321.7	3.45	296.5	7.18	291.1	6.94
$UE4_{rr}$	301.4	3.67	276.3	6.90	268.8	7.43
$UE4_{mci}$	291.9	4.55	261.3	26.41	264.5	25.5
$UE4_{fcds}$	263.7	5.18	228.5	9.61	228.4	7.29

cannot work normally in HSDPA. To the best of our knowledge, there is few work on improving TCP performance over HSDPA network, we use TCP and I-TCP [13], namely, the pure split-connection scheme, as the benchmark. Each UE randomly starts its session in first second, and all simulations last 300 s.

The average throughput of various schemes and theirs standard deviations are calculated, and the results are listed in Table 3. In most situations, our TCP Proxy improves throughput by 10 percent compared with both the standard TCP and I-TCP. In addition, we can find that the performance of I-TCP, the pure split-connection proxy without our window adaption mechanism, is nearly identical with that of TCP, which implies that the pure split-connection proxy could not be employed to exert sufficiently the precious cellular link.

To make a vivid comparison, taking example for the cellular link driven by $UE3_{fcds}$, we draw the evolutions of various key parameters in Fig. 10. Fig. 10a describes the variable bandwidth, Figs. 10b and 10d present the sending window size of the proxy and the queue length in Node-B controlled by Window Adaptation TCP Proxy, respectively. When the proxy employs I-TCP instead of our TCP Proxy, the evolutions of the congestion window of TCP connection between UE and the proxy and the queue length in Node-B are shown in Figs. 10c and 10e, respectively. The congestion window of TCP newReno displays the irregular sawtooth shape, which does not timely catch up the dynamic change of the bandwidth and results in queue oscillation with a large amplitude. Many packets are dropped due to buffer overflow, and the end-to-end goodput is deteriorated. On the other hand, the empty queue occurs frequently, and the wireless link is not sufficiently utilized. On the contrary, our TCP proxy dynamically adjusts the sending window based on the queue length in Node-B and the cellular link bandwidth. The queue length fluctuates around the reference value. The packet dropping and empty queue seldom appear. The end-to-end performance, including goodput and delay jitter, is improved, and the precious cellular link is sufficiently utilized.

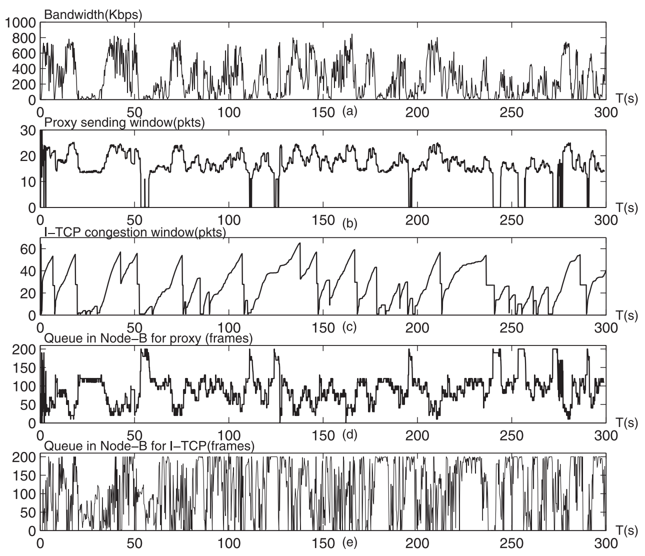

Fig. 10. Representative simulation results: (a) Variable bandwidth. (b) Proxy sending window. (c) I-TCP congestion window. (d) Queue length in Node-B for TCP proxy. (e) Queue length in Node-B for I-TCP.

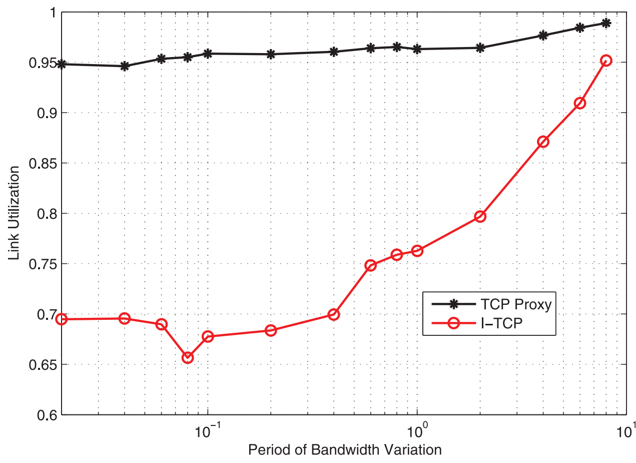

Fig. 11. Link utilization with different bandwidth variation.

The above experiments verify that the TCP Proxy scheme and our SMVS control algorithm can improve the utilization of cellular link with variable bandwidth. However, these variations are relatively slow and mild. When UE stays in the more complex channel environment, TTI is fixed at relatively small values (such as 1 ms in LTE), and various opportunistic scheduling schemes are employed for high spectrum efficiency [19], the available bandwidth may vary quickly and drastically. To test the performance of our TCP proxy in these harsh environments, we conduct a series of simulations using the simple network topology shown in Fig. 8. TCP proxy is inserted between Node-B/RNC and server, the propagation delay between RNC and the proxy is 20 ms, and $\gamma = 1$ which implies that both radio frames and IP packets have the same size (500 bytes). We generate several random sequences whose periods are 0.02, 0.04, 0.06, 0.08, 0.1, 0.2, 0.4, 0.6, 0.8, 1.0, 2.0, 4.0, 6.0, and 8.0 s, respectively, and then use them to dynamically configure the cellular link in Fig. 8. Recording data and calculating the link utilization, we draw the results in Fig. 11.

Obviously, I-TCP is ill suited to rapid and drastic bandwidth change. In the extreme case, I-TCP only reaches

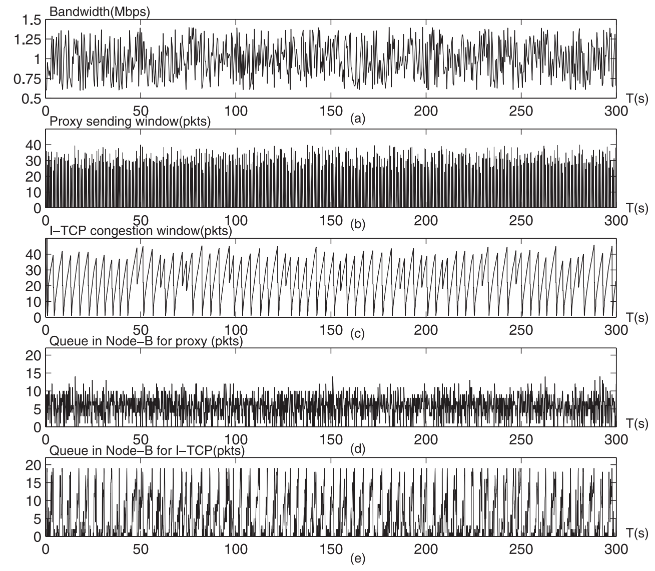

Fig. 12. Simulation Results of fast varying bandwidth: (a) Variable bandwidth. (b) Proxy sending window. (c) I-TCP congestion window. (d) Queue length in Node-B for TCP proxy. (e) Queue length in Node-B for I-TCP.

about 70 percent link utilization, but our TCP Proxy keeps over 95 percent under most conditions. Naturally, the frequency of bandwidth variation also affects the performance of our TCP Proxy, but the negative impact on the link utilization can be limited in the acceptable scope. Otherwise, Fig. 11 also tells that both I-TCP and our TCP proxy perform similarly as the bandwidth varies slowly. The detailed simulation results are presented in Fig. 12 where the period of bandwidth variation is 0.2 s. As the bandwidth is modulated by the random sequences in Fig. 12a, through observing Fig. 12c which describes the evolution of congestion window of I-TCP, we can find that slow start led by spurious timeouts occurs frequently and Fast Retransmission and Fast Recovery (FR/FR) built in TCP newReno is nearly disabled. On the other hand, the queue length in Node-B shown in Fig. 12e readily inclines to be empty, which indicates the precious cellular link is likely wasted. However, as shown in Fig. 12d, the queue controlled by the SMVS regulator vibrates around the reference value. The empty or full queue does not often appear, which assures that the wireless link is fully utilized.

5.2 Impact of Parameters

Theoretically, if only a parameter setting meets condition (43), the queue in Node-B will be stable. The system defined by (17) and (36) is unconstrained, namely, the state variable $y(k)$ and the control variable $w(k)$ can be any values in real domain, but both of them are constrained in an actual system, such as $0 \leq q(k) \leq B_{max}$ and $w(k) \geq 0$. Under these hard restrictions, if the sampling period T_s is too large, the dynamic behavior of the queue system may be hardly observed in time. For example, when $q(k) = B_{max}$, from control law (36), $w(k+1)$ should be equal to zero in most conditions, namely, no packets will enter the queue in the next step. If $T_s > B_{max}/\bar{b}$, the queue will be drained out before the update message is sent; then $q(k+1) = 0$, the empty queue results in a relatively large $w(k+2)$, which may cause buffer overflow again, i.e., $q(k+2) = B_{max}$. This

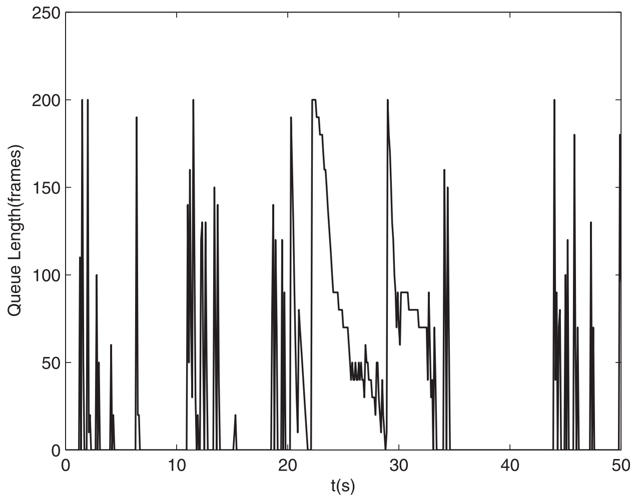

Fig. 13. Unstable queue system ($T_s = 0.2$).

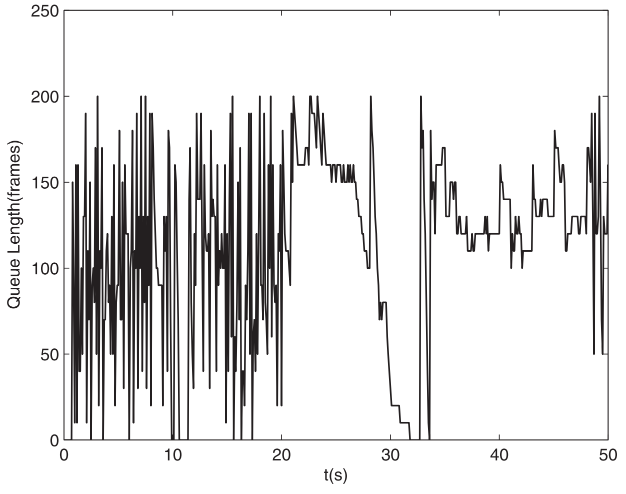

Fig. 14. Stable queue system ($T_s = 0.1$).

oscillation needs to be avoided since it certainly deteriorates the performance; thus, $T_s < B_{max}/\bar{b}$ should be included into the conditions for parameter setting in practice. To verify this rule, we redo the foregoing experiment. The cellular link is driven by $UE3_{fcds}$, then $\bar{b} \simeq 1{,}160$ frames/s ($1{,}160 \text{ frames/s} = 13 \times 89.26$ pkts/s, see Table 3). Let $T_s = 0.2$, $\delta = 4$, $\varepsilon = 20$, $q_0 = 100$, and $B_{max} = 200$ frames, Although this configuration satisfies condition (43), the queue shown in Fig. 13 is unstable and the link utilization only reaches 23.76 percent. The fact that the queue oscillates between 0 and the buffer size accords with our analysis because $T_s = 0.2 > 0.17 = B_{max}/\bar{b}$. Let $T_s = 0.1$ and other parameter be kept unchangeable, the stabilized queue presented in Fig. 14 confirms our judgement. Notice that $B_{max}/\bar{b}$ is a very loose upper bound of T_s, but is still meaningful for actual parameter configuration. On the other hand, to impose a more effective control on the queue, the Nyquist-Shannon sampling theorem needs to be followed to capture the details of bandwidth variation, i.e., $T_s < \frac{T_b}{2}$, where T_b is the period of bandwidth variation.

After the value of T_s is determined, since the reaching law requires $(1 - \delta T_s) > 0$, we can fix the parameter δ according to $0 < \delta < \frac{1}{T_s}$. Otherwise, the control law (36) tells that δ dominates the adjustment of the sending window size

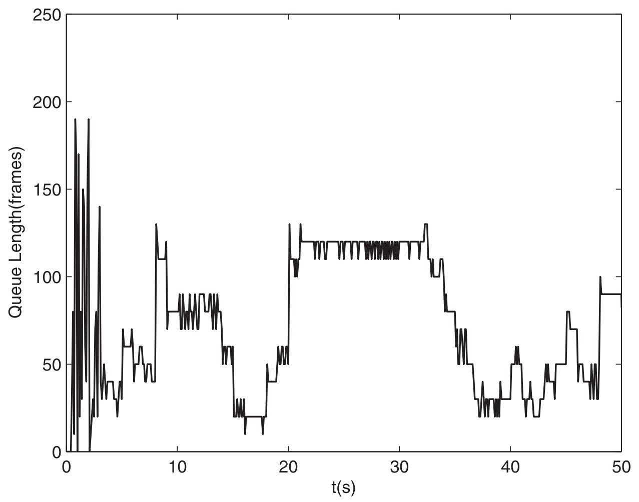

Fig. 15. Short transient adjusting ($T_s = 0.025$ and $\delta = 35$).

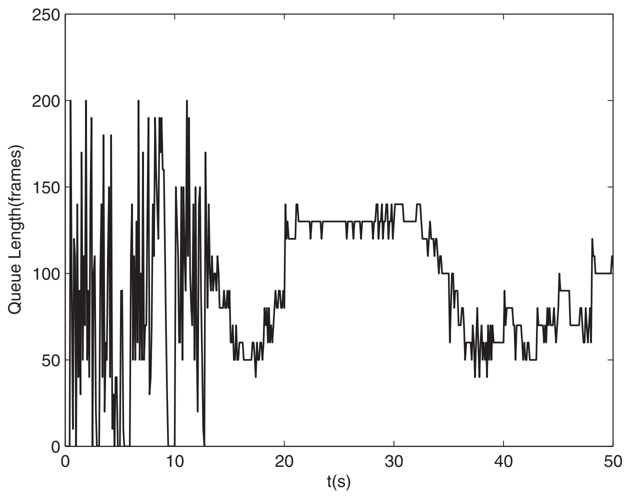

Fig. 16. Long transient adjusting ($T_s = 0.025$ and $\delta = 20$).

in the proxy. If its value is too small, the transient adjusting may take relatively long time. Let $T_s = 0.025$, $\delta = 20$, and $\varepsilon = 20$; we redo the foregoing experiment. The adjusting time of the queue system nearly approaches 20 s as shown in Fig. 16, which definitely matches our prediction. Therefore, to optimize the system performance, δ should be fixed at a value as large as possible, namely, suggesting the product δT_s should approach 1. Comparing the queue evolutions in Fig. 15 ($T_s = 0.025, \delta = 20$) and Fig. 16 ($T_s = 0.025, \delta = 35$), we can confirm this adjustment.

If both T_s and δ are fixed properly, it is straightforward to configure the parameter ε according to the constraint $\frac{\varepsilon T_s}{(1-\delta T_s)} < \min[q_0, B_{max} - q_0]$.

Finally, we summarize the above discussion as the following guidelines toward parameter setting in practice.

$$\left\{ (T_s, \varepsilon, \delta) \middle| \begin{array}{l} \frac{\varepsilon T_s}{(1-\delta T_s)} < \min[q_0, B_{max} - q_0] \\ T_s < \min[B_{max}/\bar{b}, T_b/2], \varepsilon > 0 \\ (1 - \delta T_s) > 0 \quad \text{and} \quad \delta T_s \to 1 \end{array} \right\}.$$

6 Conclusion

Because 3G networks employ code-division and time-division multiplexing, bandwidth variations sensed by the mobile terminal are unavoidable. Various link adaptation techniques employed by HSDPA provide higher peak data rates, but they also aggravate bandwidth variations. How to improve TCP performance over wireless links with bandwidth oscillation is still an open problem. In this paper, we first build a performance model for the standard TCP over links with variable bandwidth to provide an accurate estimation of throughput. The analysis results based on this model convince us that the spurious timeout caused by abrupt bandwidth decrease is one of the most important factors which degrades TCP performance. In addition, the loss of synchronization (even conflict) between the congestion window adjustment and the bandwidth oscillation results in bandwidth waste and buffer overflow which leads bursty packet drops. Motivated by these insights, we propose an enhanced solution based on the split-connection scheme. The sending window size of a proxy is regulated based on the dynamic values of varying bandwidth. Buffer overflow and empty queue are effectively avoided through keeping the length of the queue associated with the cellular link around a fixed value, so that the bottleneck wireless link is fully utilized and transport performance is improved. Since some traditional mechanisms in the standard TCP are disabled by the new control algorithm in the TCP proxy, many possible factors which result in performance degradation over wireless cellular links, such as spurious timeouts, naturally disappear. In addition, the design approaches in the robust sliding mode variable structure control theory are suitable for random bandwidth variation. Therefore, the precious wireless links are sufficiently utilized and the TCP throughput is improved.

Acknowledgments

The authors gratefully acknowledge the anonymous reviewers for their constructive comments. This work is supported in part by the National Natural Science Foundation of China (NSFC) under Grant No.60773138, 60971102, the National Grand Fundamental Research Program of China (973) under Grant No. 2009CB320504, and the National Science and Technology Major Project of China (NSTMP) under Grant No. 2009ZX03006-001-003 and 2009ZX03006-003-01.

References

[1] H. Balakrishnan, V. Padmanabhan, S. Seshan, M. Stemm, and R.H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," *IEEE/ACM Trans. Networking,* vol. 5, no. 6, pp. 756-769, Dec. 1997.

[2] M. Yavuz and F. Khafizov, "TCP over Wireless Links with Variable Bandwidth," *Proc. IEEE Vehicular Technology Conf. (VTC '02 Fall),* Sept. 2002.

[3] M.C. Chan and R. Ramjee, "TCP/IP Performance over 3G Wireless Links with Rate and Delay Variation," *Proc. ACM MobiCom,* pp. 71-82, 2002.

[4] M.C. Chan and R. Ramjee, "Improving TCP/IP Performance over Third Generation Wireless Networks," *Proc. IEEE INFOCOM,* 2004.

[5] J.C. Moon and B.G. Lee, "Rate-Adaptive Snoop: A TCP Enhancement Scheme over Rate-Controlled Lossy Links," *IEEE/ACM Trans. Networking,* vol. 13, no. 3, pp. 603-615, June 2006.

[6] F. Khafizov and M. Yavuz, "Running TCP over IS-2000," *Proc. IEEE Int'l Conf. Comm. (ICC '02),* 2002.

[7] M. Mathis, J. Semke, and J. Mahdavi, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," *ACM Computer Comm. Rev.,* vol. 27, no. 3, pp. 76-82, July 1997.

[8] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation," *IEEE/ACM Trans. Networking,* vol. 8, no. 2, pp. 133-145, Apr. 2000.

[9] A. Canton and T. Chahed, "End-to-End Reliability in UMTS: TCP over ARQ," *Proc. IEEE Global Telecomm. Conf. (GLOBECOM '01),* 2001.

[10] M. Ghaderi, A. Sridharan, H. Zang, D. Towsley, and R. Cruz, "Modeling TCP in a Multi-Rate Multi-User CDMA System," *Proc. IFIP Networking,* May 2007.

[11] T.V. Lakshman and U. Madhow, "The Performance of Networks with High Bandwidth-Delay Products and Random Loss," *IEEE/ACM Trans. Networking,* vol. 4, no. 3, pp. 336-350, June 1997.

[12] J. Border et al., "Performance Enhancement Proxies Intended to Mitigate Link-Related Degradations," IETF RFC 3135, June 2001.

[13] A. Baker and B.R. Badrinath, "I-TCP: Indirect TCP for Mobile Networks," *Proc. 15th Int'l Conf. Distribution Computing Systems,* 1995.

[14] W. Wei, C. Zhang, H. Zang, J. Kurose, and D. Towsley, "Inference and Evaluation of Split-Connection Approaches in Cellular Data Networks," *Proc. Passive and Active Measurement Conf.,* Mar. 2006.

[15] EURANE, User Guide (rel.1.6), http://www.tl-wmc.nl/eurane, 2009.

[16] I.C.C. Bruin, G. Heijenk, M.E. Zarki, and J.L. Zan, "Fair Channel-Dependent Scheduling in CDMA Systems," *Proc. 12th IST Mobile and Wireless Comm. Summit,* pp. 737-741, June 2003.

[17] U. Itkis, *Control System of Variable Structure.* Wiley, 1976.

[18] H. Kaaranen et al., *UMTS Networks: Architecture, Mobile and Services.* John Wiley & Sons, 2001.

[19] R. Racic, D. Ma, H. Chen, and X. Liu, "Exploiting Opportunistic Scheduling in Cellular Data Networks," *Proc. 15th Ann. Network and Distributed System Security Symp. (NDSS '08),* Feb. 2008.

[20] T. Gff, J. Moronski, D.S. Phatak, and V. Gupta, "Freeze-TCP: A True End-to-End TCP Enhancement Mechanism for Mobile Environments," *Proc. IEEE INFOCOM,* pp. 1537-1545, Mar. 2000.

[21] V. Padmanabhan and R. Katz, "TCP Fast Start: A Technique for Speeding Up Web Transfers," *Proc. IEEE Global Telecomm. (GLOBECOM '98) Internet Mini-Conf.,* 1998.

[22] R. Wang, G. Pau, K. Yamada, M.Y. Sanadidi, and M. Gerla, "TCP Startup Performance in Large Bandwidth Delay Networks," *Proc. IEEE INFOCOM,* 2004.

[23] C.P. Fu and S.C. Liew, "TCP Veno: TCP Enhancement for Transmission over Wireless Access Networks," *IEEE J. Selected Areas on Comm.,* vol. 21, no. 2, pp. 216-228, Feb. 2003.

[24] H. Inamura et al., "TCP over Second (2.5G) and Third (3G) Generation Wireless Networks," IETF RFC 3481, Feb. 2003.

[25] A. Chockalinggam and M. Zorzi, "Wireless TCP Performance with Link Layer FEC/ARQ," *Proc. IEEE Int'l Conf. Comm. (ICC '99),* pp. 1212-1216, June 1999.

[26] N.M. Chaskar, T.V. Lakshman, and U. Mahonen, "TCP over Wireless with Link Error Control: Analysis and Design Methodology," *IEEE/ACM Trans. Networking,* vol. 7, no. 5, pp. 605-615, Oct. 1999.

[27] H. Balakrishnan, S. Seshan, E. Amir, and R.H. Katz, "Improving TCP/IP Performance over Wireless Networks," *Proc. ACM MobiCom,* 1995.

[28] A. Baker and B.R. Badrinath, "I-TCP: Indirect TCP for Mobile Networks," *Proc. 15th Int'l Conf. Distribution Computing Systems,* 1995.

[29] K. Brown and S. Singh, "M-TCP: TCP for Mobile Cellular Networks," *Proc. ACM Computer Comm. Rev.,* 1997.

[30] M. Ivanovich and P.W. Bickerdike, and J.C. Li, "On TCP Performance Enhancing Proxies in a Wireless Environment," *IEEE Comm. Magazine,* vol. 46, no. 9, pp. 76-83, Sept. 2008.

[31] M. Meyer, J. Sachs, and M. Holzke, "Performance Evaluation of a TCP Proxy in WCDMA Networks," *IEEE Comm. Magazine,* vol. 10, no. 5, pp. 70-79, Oct. 2003.

[32] K. Mattar, A. Sridharan, H. Zang, I. Matta, and A. Bestavros, "TCP over CDMA2000 Networks: A Cross-Layer Measurement Study," *Proc. Passive and Active Measurement Conf. (PAM '07).* Apr. 2007.

[33] M. Fiorenzi, D. Girella, N. Moller, A. Arvidsson, R. Skog, J. Petersson, P. Karlsson, C. Fischione, and K.H. Johansson, "Enhancing TCP over HSDPA by Cross-Layer Signalling," *Proc. IEEE Global Telecomm. Conf. (GLOBECOM '07),* 2007.

[34] M. Ghaderi, A. Sridharan, H. Zang, D. Towsley, and R. Cruz, "TCP-Aware Resource Allocation in CDMA Networks," *Proc. ACM MOBICOM '06.* Sept. 2006.

[35] M. Assaad and D. Zeghlache, *TCP Performance over UMTS-HSDPA.* CRC Press, 2008.

Fengyuan Ren received the BA and MSc degrees in automatic control from Northwestern Polytechnic University, China, in 1993 and 1996, respectively. In December 1999, he received the PhD degree in computer science from Northwestern Polytechnic University. He is a professor in the Department of Computer Science and Technology, Tsinghua University, Beijing, China. From 2000 to 2001, he worked in the Electronic Engineering Department of Tsinghua University as a postdoctoral researcher. In January 2002, he moved to the Computer Science and Technology Department of Tsinghua University. His research interests include network traffic management and control, control in/over computer networks, wireless networks, and wireless sensor networks. He has authored or coauthored more than 80 international journal and conference papers. He is a member of the IEEE and has served as a technical program committee member and local arrangement chair for various IEEE and ACM international conferences.

Chuang Lin received the PhD degree in computer science from Tsinghua University, Beijing, China, in 1994. He is a professor in the Department of Computer Science and Technology, Tsinghua University. He is an honorary visiting professor at the University of Bradford, United Kingdom. His current research interests include computer networks, performance evaluation, network security analysis, and Petri net theory and its applications. He has published more than 300 papers in research journals and IEEE conference proceedings in these areas and has published four books. He is a senior member of the IEEE and the chinese delegate in TC6 of IFIP. He serves as the technical program vice chair of the 10th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS 2004), the general chair of the ACM SIGCOMM Asia Workshop 2005 and the 2010 IEEE International Workshop on Quality of Service (IWQoS 2010), the associate editor of the *IEEE Transactions on Vehicular Technology*, the area editor of the *Journal of Computer Networks*, and the area editor of the *Journal of Parallel and Distributed Computing*.