Improving TCP Throughput over HSDPA Networks

Fengyuan Ren, *Member, IEEE,* Xiaomeng Huang, Feng Liu, and Chuang Lin, *Senior Member, IEEE*

***Abstract*—The various link adaptation techniques employed by High Speed Downlink Packet Access (HSDPA) in the third generation (3G) networks augment the bandwidth oscillation, which is identified as one of the most important factors resulting in the throughput deterioration of Transmission Control Protocol (TCP). In this paper, we firstly explain why the bandwidth oscillation degrades the TCP performance through a special simulation experiment. Subsequently, a split connection Window Adaptation TCP Proxy is proposed to improve the TCP throughput over HSDPA networks. In this solution, the built-in attributes of HSDAP system are sufficiently utilized. In order to effectively use the precious cellular link resources, the length of the queue connected with it is intentionally kept around the reference value through adjusting the sending window size of TCP proxy based on the dynamic values of varying bandwidth. A discrete-time stochastic state space model is formulated to analyze the system stability. The validity of enhanced scheme is verified through simulation experiments. The performance of TCP proxy is compared with the standard TCP protocol. The numerical results show that our TCP proxy is able to keep the cellular link utilization over 90%, and to improve TCP throughput by 100% under most conditions.**

***Index Terms*—TCP performance, bandwidth oscillation, proxy, stochastic control, HSDPA networks.**

I. Introduction

THE wireless data services are anticipated to have a rapid growth over the next years, and will likely become the dominating source of traffic load in 3G wireless cellular networks, which are increasingly being deployed throughout the world. Therefore, an efficient provision for 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 TCP in the Internet, particular attention must be paid to the performance of TCP over 3G wireless networks.

In fact, the performance of TCP over wireless networks has been extensively studied in last decade since the experiment results show that TCP suffers the significant throughput degradation when there is a wireless link in the end-to-end connection [1]. It has been found that wireless link losses have an adverse impact on performance as TCP is unable to distinguish corruption loss from congestion loss, so any kind of losses are perceived as congestion, which results in source throttling and very low throughout. These findings have been one of the main motivations for the use of extensive local retransmission mechanisms in 3G networks[2].

Manuscript received March 12, 2007; revised September 19, 2007; accepted November 9, 2007. The associate editor coordinating the review of this paper and approving it for publication was W. Zhuang.

F. Ren, X. Huang, and C. Lin are with the Computer Science and Technology Department, Tsinghua University, Beijing, P.R. China (e-mail: renfy@tsinghua.edu.cn, huangxiaomeng@tsinghua.edu.cn).

F. Liu is with the School of Electronics and Information Engineering, Beihang University, Beijing, P.R. China.

Digital Object Identifier 10.1109/TWC.2008.061007.

Recently, researchers have begun to investigate the impact of the wireless link bandwidth variation on the TCP performance [3][4][5][6]. 3G networks are required to support dynamic resource sharing among concurrent data users. If multiple users wish to transfer large numbers of data simultaneously, the scheduler may have to repeatedly allocate and deallocate resources for each user. Intuitively, the channel-state based scheduling mechanisms improve the overall throughput, while they also result in bandwidth oscillations, which were identified as one of the most important factors in deteriorating throughput [3][6].It is an open question to what extent transport protocols can be designed to optimize performance in the presence of bandwidth oscillation, and to what extent wireless link-level mechanisms for bandwidth variation can be designed to take into account the transport protocols that will run over these links [7].

In this paper, we will restrict ourselves to Universal Mobile Telecommunication System(UMTS) based 3G networks to investigate this open performance optimization issue. Based on the split-connection approach [8], we propose a new TCP enhancement solution called Window Adaptation TCP Proxy to maximize TCP throughput over HSDPA in UTMS networks. Our TCP proxy sufficiently utilizes the inherent attribute of HSDAP system to obtain the dynamic bandwidth value, and employs a dynamic window adjusting scheme to adapt to the variable bandwidth.

The remainder of the paper is organized as follows. The simplified HSDPA operation principle and its technical features are introduced, and the related works are discussed in Section 2. In Section 3, why the bandwidth oscillation degrades the TCP performance is analyzed and explained through a special simulation experiment. In Section 4, the architecture of window adaptation TCP proxy is described and its advantages are discussed. Subsequently, the control algorithm is designed to dynamically regulate the sending window size of TCP proxy, and then a discrete-time stochastic state space model is formulated to analyze the system stability. In Section 5, the validity of our TCP Proxy is verified and its performance is compared with the standard TCP through simulation experiments. Finally, the conclusions are drawn in Section 6.

II. Background and Related Works

A. HSDPA

Firstly, we provide a general overview of the HSDPA technology since it may be helpful to achieve a full comprehension of the investigation in this paper. HSDPA is a new technology standardized by 3GPP (3G Partnership Project) in Release 5 to boost the support for packet switched services as an evolution of UMTS radio interface. The main idea behind the HSDPA is to share one single downlink data channel among

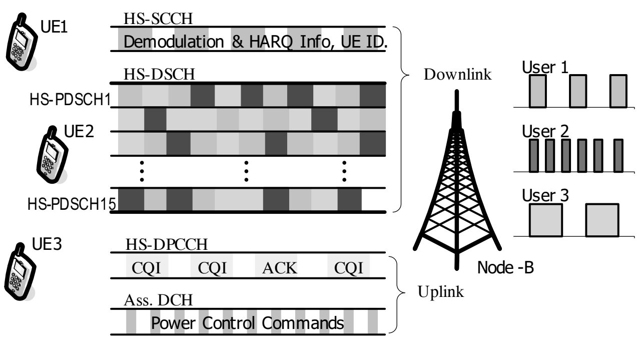

Fig. 1. HSDPA operation principle.

all the users, along with many modifications that make it especially effective for packet-based communication. HSDPA utilizes Adaptive Modulation and Coding (AMC) to substitute fast power control and variable spreading factor in traditional 3G technology. To facilitate the high peak data rates, HSDPA introduces the variable-rate turbo encoding and employs the multi-code transmission. The peak rate varies from 120 kbps up to 10.8 Mbps (in practice up to 2 Mbps) under different parameter configurations.

The simplified HSDPA operation principle is illustrated in Fig.1. Depending on packet prioritization and resource availability, Node-B schedules user data on the High Speed Downlink Shared Channel (HS-DSCH), where a large amount of power and code resources are assigned to a single user at a certain TTI(Transmit Time Interval) in time and/or code multiplex manner. Prior to sending data on the HS-DSCH, Node-B sends a detailed demodulation message to the active users via the High-Speed Shared Control Channel (HS-SCCH). The uplink High Speed Dedicated Physical Control Channel (HS-DPCCH) carries the necessary control information, such as ARQ acknowledgements and Channel Quality Indicators (CQIs). CQI is an indicator of AMC and multi-code number currently supported by the user equipment (UE), namely the supported data rate under the current channel conditions. The feedback cycle of the HS-DPCCH CQI can be set as a network parameter in predefined steps, such as 2ms and 20ms etc.

On the viewpoint of optimizing TCP performance, there are two obvious features in HSDPA technology. First, the various link adaptation techniques employed by HSDPA give rise to the bandwidth variation. Otherwise, the fact that the bandwidth of the downlink data channel is shared among all users also augments high bandwidth variation. The available bandwidth for a connection is affected by the number of users entering and leaving the cell since they all compete for the resources. Second, some wireless link-level enhanced mechanisms, such as AMC and scheduling schemes based on CQI, make NodeB accurately know the bandwidth assigned to each user. Although it is not purposely designed for optimizing TPC performance, we will adequately exploit this potential in our TCP proxy.

B. Related work

In [6], F. Khafizov and M. Yavuz conducted experiments of TCP over IS-2000 networks and concluded that bandwidth oscillation severely degrades TCP throughput. The reasons

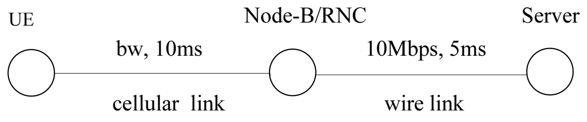

Fig. 2. Simulation topology.

are finally attributed to the spurious retransmission. However, it has been mentioned that no performance benefit has been obtained with modifications in the TCP retransmission timer in [9]. M.C. Chan and R. Ramjee believed that TCP performance degradation caused by bandwidth and delay variation is due to the difficulty in estimation of *bandwidth-delay product*(BDP) of the end-to-end path at the TCP source [3]. They developed two improving schemes successively, i.e. ACK Regulator [3] and Window Regulator [4]. The ACK Regulator runs in an intermediate bottleneck network element and monitors the arrival rate of packets, and controls the release of ACKs to the TCP source so as to prevent buffer overflow. The main advantage of the ACK Regulator is that no modification is needed at end systems. However, it does not address the performance of short-lived flows. The Window Regulator is very similar with 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. However, the algorithms in [3] and [4] have a common features, namely they infer the bandwidth variation through observing the evolution of queue length. This approach is effective for slow and moderate variation in bandwidth, but does not work for rapid and drastic bandwidth oscillations since the buffer may become empty or full in this situation. Naturally, it is hard to calculate the appropriate receive window value in ACKs or the release rate of ACKs using the heuristic algorithms with the incomplete bandwidth information. In addition, the determinate analysis techniques used in these work can not provide comprehensive evaluation on system performance since the bandwidth variation is random and indeterminate.

III. Why Bandwidth Oscillation Degrades TCP Performance

In order to focus on the effects of bandwidth oscillation on TCP performance, we use a rather simple network topology shown in Fig.2 to conduct an experiment on the *ns2* simulation platform since it may neglect the effects caused by other possible factors. The system consisted of an integrated NodeB/RNC, a server connected to the RNC by a 10 Mbps wired link, and a mobile device connected to the NodeB by a cellular link with 96 kbps uplink and varying bandwidth downlink, whose capacity varies according to the rectangular pattern shown in Fig.3(a), i.e. the interval is 10s, and the peak bandwidth and the valley bandwidth are 384 kbps and 38.4 kbps respectively. The TCP source at the server using newReno transfers a large file to the mobile device. The maximum transmission unit (MTU) is 1500bytes and the buffer at the RNC is 20 packets. We monitor the evolutions of the congestion window at TCP source and the queue length at RNC, and trace TCP sequences (including

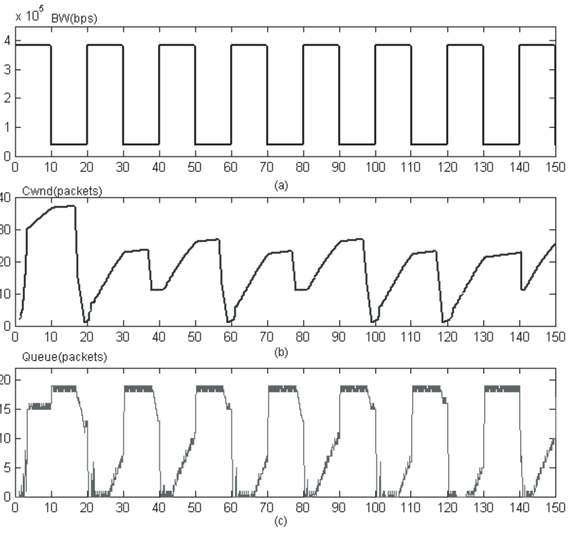

Fig. 3. Simulation Experiment plot (a) shows the bandwidth variation, plot (b) shows the evolution of congestion window, and plot (c) shows the evolution of queue length in the intermediate node.

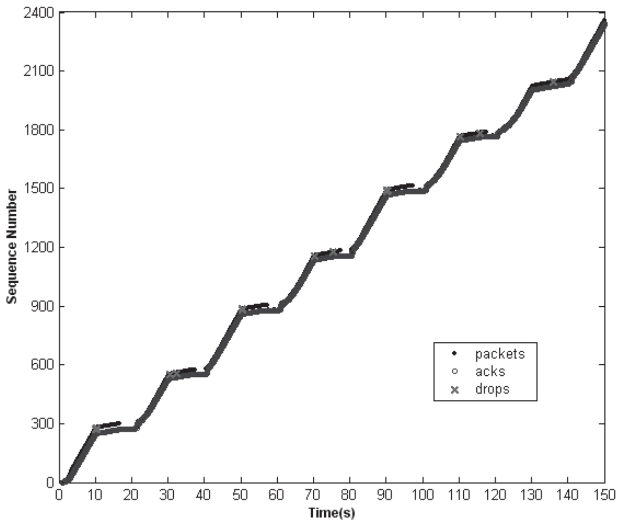

Fig. 4. TCP sequence trace.

sending packets, receiving ACKs, and dropping packets). The results are presented in Fig.3(b), Fig.3(c) and Fig.4 respectively. Theoretically, the average throughput should be equal to $\int_0^{150} bw(t)dt/150 = 222.7$kbps, but the actual average throughput is 190.4 kbps, and the link utilization is about 85%. Observing Fig.3 and Fig.4, we can find some hints that the bandwidth oscillation deteriorates TCP performance. At time t=50 and 90, the link bandwidth decreases from 384 kbps to 38.4 kbps, the buffer overflows, and the congestion window is shrunk, which indicates that the *Fast Retransmission* mechanism is triggered. Subsequently, the congestion window is reset to 1 but without any dropping packets, which means that the spurious timeouts occurs. This phenomenon agrees with the explanation presented in [6]. However, not all bandwidth attenuations certainly result in the spurious timeouts, such as at time t=30 and 70, since the queue length is relatively small at that time, the variation of queuing delay caused by the abrupt decrease bandwidth is not large enough to lead the spurious timeouts. It is noticeable that the bandwidth is decreasing at time t = 30, but the congestion window is increasing, finally the buffer overflows so that the window size is halved. Next, the bandwidth jumps, but the congestion window is too small to fill the link, moreover, it climbs slowly, so that the scarce wireless link resource is wasted. Therefore, it is incomplete if TCP throughput deterioration caused by bandwidth oscillation is merely attributed to the spurious timeouts. We believe that the essential reason should be that the congestion window does not timely catch up the varying bandwidth. If they could harmoniously match, the link would be fully utilized, and there should not be so many packets to accumulate in the queue, then the spurious timeouts would be naturally avoided. The necessary condition that TCP sending window timely keeps up with the bandwidth changes is that the accurate bandwidth information available. It is hard to estimate the available bandwidth in Internet. Fortunately, it is readily obtained in HSDPA network because Node-B is responsible for allocate bandwidth for each UE based on channel state and maintains a dedicated queue for each flow. We will sufficiently utilize the inherent attribute to improving TCP performance over HSDAP networks with bandwidth oscillation.

IV. Optimizing TCP Performance in HSDPA

The split connection scheme proposed in [8] has been widely used in commercial cellular networks [11]. In this work, combining with the architecture of UTMS network and the inherent attribute of HSDPA system, we propose a split connection Window Adaptation TCP proxy to optimize TCP performance.

A. Architecture

A simplified architecture of UMTS network is depicted in Fig.5, only the elements that play an important role on the TCP connection are presented. The UMTS functionality is divided into three main domains: 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. Node-B maintains the per-user queue and performs the link adaptation operation, and then schedules transmission on the wireless channel based on the channel state. Our split connection TCP proxy is located between GGSN and the external networks. For the convenience of discussion, the basic structure of our window adaptation TCP proxy is illustrated in Fig.6. One TCP connection between the UE and the server is split into two ones: one between the UE and the proxy and another between the proxy and the server. The connection between the server and proxy can employ any standard TCP protocols, while the transport protocol on the connection between the proxy and the UE needs to be

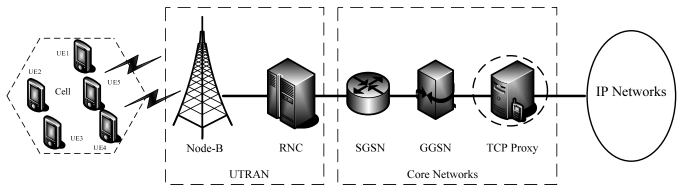

Fig. 5. Simplified UTMS network architecture.

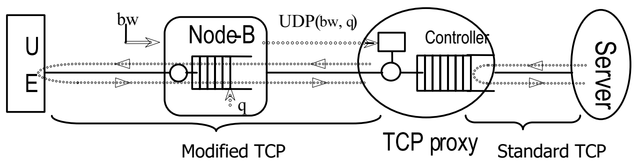

Fig. 6. Basic structure of window adaptation TCP proxy.

customized in order to adapt to the dynamic changes of the wireless link bandwidth. For the TCP connection on the wireless link, the proxy is a sender. All the schemes and algorithms related with congestion window adjustment in the standard TCP will be disabled in the modified TCP protocol. The sending window size is dynamically regulated depending on the available wireless bandwidth and the queue length in the Node-B. They are periodically conveyed to the TCP proxy through the UDP connection established between Node-B and the proxy. If the queue length in Node-B could be kept around the desired small value all along, the perfect performance would be reached since both buffer overflow and empty queue can be avoided so that the cellular link will be fully utilized. To achieve this target, 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 flow control algorithms in the standard TCP, including slow start, fast retransmission, fast recovery and timeout retransmission, are disabled in our modified TCP version, most possible factors resulting in performance degradation over wireless cellular links, such as superior timeouts and small initial window size etc., are naturally eliminated. However, the recovery strategy in the standard TCP will be retained, namely 3 duplicate ACKs trigger the retransmission of the corresponding packet, but it does not affect the sending window adjustment.

B. Algorithms design

To design a stable and effective flow control algorithm running on the TCP proxy, we firstly 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 at which the feedback information is sent on the UDP connection. At the end of each step, the TCP proxy updates its sending window size using the feedback information, i.e. the available bandwidth and the queue length. Let $w(n)$ denote the sending window size during step n. Let $b(n)$ and $q(n)$ denote the bandwidth of radio channel and the instantaneous queue length in Node-B at the end of the nth step respectively. Defining the duration of each step as a unit time, we have:

$$q(n+1) = q(n) + w(n) - b(n) \tag{1}$$

Since the available bandwidth $b(n)$ may change over time randomly, we model it as an ARMA(p, q) stochastic process.

$$b(n) = b + \varphi(n) = b + \sum_{i=1}^{p} \alpha_i \varphi(n-i) + \sum_{j=1}^{q} \beta_j \phi(n-j) \tag{2}$$

where b denotes the nominal value, $\phi(n)$ is independent and identically distribution(i.i.d.) standard Gaussian random variables, whose mean and variance are equal to 0 and 1 respectively. Both $\alpha_i (i = 1 \ldots p)$ and $\beta_j (j = 1 \ldots q)$ are constant. ARMA models can adequately model any stationary time series, at the same time, they are analytically tractable [12]. Although Node-B explicitly feeds back the available bandwidth and the queue length to the TCP proxy, the proxy can only obtain the noisy estimates of these two random variables due to the interferences introduced by all kinds of sources, such as transfer delay and loss of signaling packet. Let $\hat{b}(n)$ denote the estimate for the available bandwidth, and $\hat{q}(n)$ denote the estimate for the queue length. Assume that these estimates can be expressed as:

$$\hat{b}(n) = b(n) + \gamma(n+1) \tag{3}$$

$$\hat{q}(n) = q(n) + \eta(n+1) \tag{4}$$

where $\gamma(n+1)$ and $\eta(n+1)$ are Gaussian white noise, thus the TCP proxy can use these estimates to update its sending window size.

Substituting (3) and (4) into (1), we have:

$$q(n+1) = \hat{q}(n) + w(n) - \hat{b}(n) + \pi(n+1) \tag{5}$$

where $\pi(n+1) = \eta(n+1) - \gamma(n+1)$, it is another Gaussian white noise. In order to solve the optimal window adjusting algorithm, we define the cost function J as:

$$J = E[q(n+1) - q_0]^2 \tag{6}$$

where q_0 is the desired queue length. Through minimizing the cost function J, we can get the adjusting algorithm of sending window size on the TCP proxy as follows:

$$w(n) = \hat{b}(n) - \hat{q}(n) + q_0 \tag{7}$$

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 there will be a lot of packets in the buffer on the TCP proxy, so that the end-to-end delay increases and some packets in real-time interactive applications may become stale. In order to avoid this possible phenomenon, we employ the advertised window inserted into the corresponding field of ACKs to limit the sending rate of the server. The objective is still 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 obtain the control law to regulate the advertised window size:

$$aw(n) = w(n) - q_2(n) + q_{20} \tag{8}$$

where $aw(n)$ denotes the advertised window size, $q_2(n)$ is the instantaneous queue length on the TCP proxy, and q_{20} is its reference value.

C. Stability analysis

The control law (7) and (8) will be effective as long as the system is stable. In this section, we will proof a theorem relating to the system stability.

Theorem If $\sum_{i=1}^{p}|\alpha_i| < 1$ and $E[q(0)] = q_0$ then the following system is neutrally stable.

$$q(n+1) = q(n) + w(n) - b(n) \qquad (9)$$

$$b(n) = b + \sum_{i=1}^{p}\alpha_i\varphi(n-i) + \sum_{j=1}^{q}\beta_j\phi(n-j) \qquad (10)$$

$$w(n) = b(n) - q(n) + q_0 + \pi(n+1) \qquad (11)$$

Proof For the convenience of analysis, the system can be rewritten as the state space model[10]. Let

$$\mathbf{X}\text{(n)}=\begin{pmatrix} w(n)-b \\ q(n)-q_0 \\ \varphi(n) \\ \vdots \\ \varphi(n-p+1) \end{pmatrix} \quad \mathbf{Y}\text{(n)}=\begin{pmatrix} \pi(n+1) \\ 0 \\ \sum_{j=0}^{q}\beta_j\phi(n-j) \\ \vdots \\ 0 \end{pmatrix}$$

$$\mathbf{A}=\begin{pmatrix} 0 & -1 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & -1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \alpha_1 & \alpha_2 & \cdots & \alpha_{p-1} & \alpha_p \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix}$$

then it is easy to verify that the system can be rewritten as:

$$\mathbf{X}(n+1) = \mathbf{A}\mathbf{X}(n) + \mathbf{Y}(n) \qquad (12)$$

The general solution of equation (12) is:

$$\mathbf{X}(n+1) = \mathbf{A}^n\mathbf{X}(0) + \sum_{i=1}^{n-1}\mathbf{A}^{n-i-1}\mathbf{Y}(i) \qquad (13)$$

Taking expected values on both sides of equation (13), we yield:

$$E[\mathbf{X}(n+1)] = \mathbf{A}^nE[\mathbf{X}(0)] + \sum_{i=1}^{n-1}\mathbf{A}^{n-i-1}E[\mathbf{Y}(i)] \qquad (14)$$

Since the random variables $\pi(n)$ and $\phi(n)$ are zero-mean, equation (14) becomes:

$$E[\mathbf{X}(n+1)] = \mathbf{A}^nE[\mathbf{X}(0)] \qquad (15)$$

If $\mathbf{A}$ can be diagonalized, i.e. $\mathbf{A} = \mathbf{S}\Lambda\mathbf{S}^{-1}$,then

$$E[\mathbf{X}(n+1)] = \mathbf{S}\Lambda\mathbf{S}^{-1}E[\mathbf{X}(0)] = \mathbf{k}_1\lambda_1^nf_1 + \cdots + \mathbf{k}_{p+2}\lambda_{p+2}^nf_{p+2} \qquad (16)$$

where $[\lambda_1, \cdots, \lambda_{p+2}]$ is eigenvalues of the matrix $\mathbf{A}$, and the coefficients $\mathbf{f} = [f_1, \cdots, f_{p+2}]^T$ that match the initial condition $E[\mathbf{X}(0)]$ are $\mathbf{f} = \mathbf{S}^{-1}E[\mathbf{X}(0)]$.

The general solution (16) of equation (12) indicates that the stability of the system (9) (11) can be analyzed using the eigenvalues of the matrix $\mathbf{A}$, Observing the structure of $\mathbf{A}$, some characteristics are beneficial to analyzing its eigenvalues. We can decompose the matrix $\mathbf{A}$ into blocks matrices. The upper left 2×2 block matrix of $\mathbf{A}$ is denoted by $\mathbf{B}$, and the lower right $p \times p$ block matrix of $\mathbf{A}$ is denoted by $\mathbf{C}$, namely:

$$\mathbf{B} = \begin{pmatrix} 0 & -1 \\ 1 & 1 \end{pmatrix} \mathbf{C} = \begin{pmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_{p-1} & \alpha_p \\ 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}$$

Matrix $\mathbf{A}$has $p+2$ eigenvalues. The two eigenvalues should be determined by matrix $\mathbf{B}$, and the other p eigenvalues are fixed by the matrix $\mathbf{C}$. The eigenvalues of the matrix $\mathbf{C}$ are the roots of a complicated polynomial of degree p, and it is hard to obtain the explicit solutions. However, using Gerschgorin's theorem[13], we can find that the modulus of any root of this polynomial satisfies the following inequality:

$$|\lambda - \alpha_1| \leq \sum_{i=2}^{p}\alpha_i \qquad (17)$$

Thus $\sum_{i=1}^{p}|\alpha_i| < 1$ is a sufficient condition that the system is stable. If this condition holds, as $n \gg 1$, the corresponding terms in (16) approaches zero, the value of $E[\mathbf{X}(n)]$ is only determined by the eigenvalues of $\mathbf{B}$.

From $det(\mathbf{B} - \lambda\mathbf{I}) = \lambda^2 - \lambda + 1 = 0$, we obtain $\lambda_{1,2} = (1 \pm \sqrt{3}i)/2$. Since $|\lambda_{1,2}| = 1$, the system is neutrally stable according to the conclusion in [13].

V. Simulation Results

We have implemented the *Window Adaptation TCP Proxy* on *ns2.29* simulator and conducted the simulation experiments to validate its performance and made comparison with the standard TCP. Our TCP Proxy is inserted between Node-B and the server depicted in Fig.2. We use a random variable, whose peak value is 2Mbps and the varying period is $T = 200$ ms, to describe the actual bandwidth oscillation shown in Fig.7(a). Without special statement, the parameters in foregoing simulation are kept unchangeable. We assume the updating step as the unit time in the previous analysis and design. If the sampling interval is τ, the corresponding control law will become:

$$w(n) = \hat{b}(n)\tau - \hat{q}(n) + q_0 \qquad (18)$$

Let $\tau = 10$ms, and fix q_0 and q_{20} at 15 packets and 50 packets respectively. The whole simulation lasts 150s. The related parameters are traced and presented in Fig.7 together. Fig.7(b) and Fig7 (d) describes the sending window size of the proxy and the queue length in Node-B controlled by our *Window Adaptation TCP Proxy*. When the server employs TCP NewReno without our TCP Proxy, the evolutions of the congestion window and the queue length in Node-B are shown in Fig.7 (c) and Fig.7(e) respectively. The congestion window of TCP newReno displays the regular saw-tooth shape, which does not timely catch up the dynamic change of the bandwidth and results in queue oscillation with large amplitude. Plenty of packets are dropped due to buffer overflow, and the end-to-end goodput is deteriorated. On the other hand, the empty queue occurs frequently, the wireless link is in underutilization. On the contrary, our TCP proxy dynamically adjusts the sending window based on the queue length on the Node-B and the cellular link bandwidth. The queue length is always kept around the constant value. Both packet dropping and empty queue seldom appear. The end-to-end performance, including goodput and delay jitter, is improved. The upper limit of

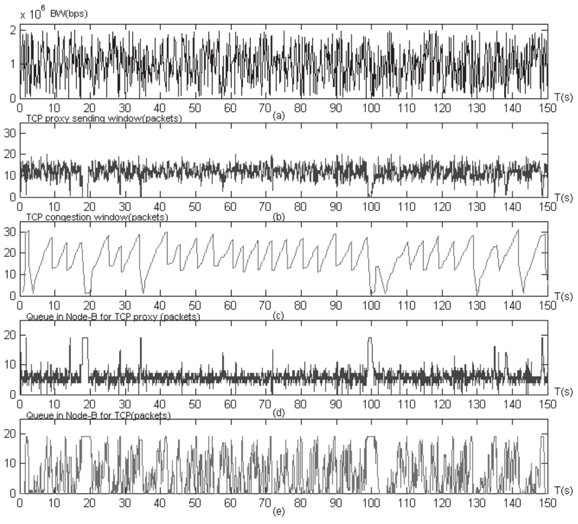

Fig. 7. Evolutions of various key parameters.

TABLE I
STATISTIC RESULTS

T	Throughput upper bound (pkts/s)	Avg. Throughput (pkts/s)		Link utilization (%)	
		TCP	Proxy	TCP	Proxy
$2s$	119.2	69.5	118.8	58.26	99.70
$200ms$	124.0	60.0	115.1	48.32	92.82
$20ms$	123.82	18.6	115.5	15.04	93.29

average throughput should be $\int_0^{150} bw(t)dt/150 = 124.0$ packets/s, and the actual throughput is 115.1 packets/s in our TCP Proxy, however, it is 60.0 packets/s in the standard TCP. Thus the link utilization is 92.82% and 48.32% respectively. The former is about twice as the latter. To demonstrate the impact of the bandwidth oscillation frequency on TCP performance, we change the period of random sequences modulating the bandwidth oscillation. Setting T=2s and 20ms, we repeat the above simulations, and record data and calculate the link utilization. The results are presented in Table 1. In the standard TCP protocol, the speed of bandwidth variation severely affects the TCP performance. As T=20ms, the link utilization is only 15.04%. For our *TCP Proxy*, the link utilization also decreases as the bandwidth oscillation frequency increases; however, it is always over 90% and acceptable. Generally, our *TCP proxy* can improve TCP performance by close to 100% (115-60/60=0.918, T=200ms), and even 500% (115.5-18.6/18.6=5.20, T=20ms) in the extreme case.

VI. Conclusion

The various link adaptation techniques employed by HSDPA provide higher peak data rates, but they also aggravate bandwidth variations, which will severely deteriorate TCP throughput. Comparing with other network technology, HSDPA network has a unique advantage, namely NodeB determinately knows the value of bandwidth assigned to each user, which is very helpful for optimizing TCP performance. Motivated by these understanding, we propose a performance enhancement proxy to optimize TCP throughput over HSDPA network with bandwidth oscillation. The sending window size of our TCP proxy is regulated based on the dynamic values of varying bandwidth. Buffer overflow and empty queue are effectively prevented through holding the length of the queue connected with the cellular link around a fixed value, so that the bottleneck wireless link is fully utilized and TCP throughput is improved. Since TCP proxy adopts a completely new flow control mechanism, and the traditional algorithms in the standard TCP are disabled, many possible factors resulting in performance degradation over wireless cellular links also naturally disappear. In most situations, the link utilization is over 90%, and the TCP throughput is improve by 100%. We have also noticed that some factors and events likely impose the negative impact on performance of our improved scheme under particular environment and configuration, such as the large propagation delay between NodeB and our proxy, and the loss of feedback signaling caused by unreliable UDP. In future work, we will try to design the robust control algorithm against these disturbances. In addition, it is challenging and attractive for us how to build an analytical model to evaluate the throughput of TCP and our TCP proxy over the wireless link with bandwidth oscillation.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China (NSFC) under Grant No.60573122 and 60773138; the National Grand Fundamental Research Program of China (973) under Grant No.2003CB314804.

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, pp. 756-769, 1997,

[2] A. Chockalinggam and M. Zorzi, "Wireless TCP performance with link layer FEC/ARQ," in *Proc. IEEE ICC'99*, pp. 1212-1216, June 1999.

[3] M. Yavuz and F. Khafizov, "TCP over wireless links with variable bandwidth," in *Proc. IEEE Vehicular Technology Conference (VTC'02 Fall)*, Sept. 2002.

[4] M. C. Chan and R. Ramjee, "TCP/IP performance over 3G wireless links with rate and delay variation," in *Proc. ACM Mobicom*, pp. 71-82, 2002.

[5] M. C. Chan and R. Ramjee, "Improving TCP/IP performance over third generation wireless networks," in *Proc. IEEE INFOCOM 2004*.

[6] F. Khafizov and M. Yavuz, "Running TCP over IS-2000," in *Proc. IEEE ICC 2002*.

[7] A. Gurtov and S. Floyd,"Modeling wireless links for transport protocols," *ACM Computer Commun. Rev.*, vol. 34, no. 2, pp. 85-96, Apr. 2004.

[8] A. Baker and B. R. Badrinath, "I-TCP: indirect TCP for mobile networks," in *Proc. 15th International Conference on Distribution Computing Systems 1995.*

[9] M. Kohlwes, J. Riihijarvi, and P. Mahnen, "Measurement of TCP performance over UMTS networks in near-ideal conditions," in *Proc. IEEE Vehicular Technology Conference (VTC'05)*, vol. 4 pp. 2235-2238, May 2005.

[10] E. Altman, F. Baccelli, and J. C. Bolot, "Discrete-time analysis of adaptive rate control mechanisms," *High Speed Networks and Their Performance*, Perros and Viniotis, eds., North Holland, 1994, pp. 121-140.

[11] W. Wei, C. Zhang, H. Zang, J. Kurose, and D. Towsley, "Inference and evaluation of split-connection approaches in cellular data networks," in *Proc. Passive and Active Measurement Conference 2006*, Adelaide, Australia, Mar. 2006.

[12] C. Chatfield, *The Analysis of Time Series: An Introduction, 6th edition.* CRC Press LLC, 2004.

[13] G. Strang, *Linear Algebra and its Applications, 2nd edition.* Academic Press, 1980.