TCP (Transmission Control Protocol) over Wireless Links *

Aldar C. F. Chan†, Danny H. K. Tsang†, Sanjay Gupta§

†Department of Electrical and Electronic Engineering,
The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, HONG KONG
{eealdar, eetsang}@ee.ust.hk

§Department of Electrical and Computer Engineering,
Illinois Institute of Technology, Chicago, IL60616, U.S.A.
gupta@ece.iit.edu

Abstract— The congestion control policy of Transmission Control Protocol (TCP) works well today over a wide range of networks. However, if a TCP connection consists of erroneous links, for example, in wireless environment, degradation in throughput and delay performance can be significant. In this paper, we use simple analysis on the TCP window dynamics to determine the end-to-end throughput of a TCP connection with a wireless link and to demonstrate the impact of high error rate in wireless environment. To improve the performance of TCP in wireless environment, a simple modification to TCP, which uses negative acknowledgment as an explicit notification for packet corruption is proposed. The performance of the proposed “NACK”-based scheme is compared with an existing modified version, LHACK, for both binary and Rayleigh fading channels.

I. INTRODUCTION

Since its first introduction, Transmission Control Protocol (TCP) has been designed and tuned for networks composed of wired links and stationary hosts. It adapts to the changing end-to-end delay conditions [1], [2] and assumes that packet¹ loss and unexpected increase in delay are largely due to network congestion. In response, the source promptly slows down its transmission, by changing the window size, to allow the network to recover from congestion. The retransmission timer is also reset with backoff to avoid unnecessary retransmissions. These congestion control policies have been shown to perform well over a wide range of data networks.

However, in data networks that consist of wireless links and mobile hosts, the high bit error rate (BER of 10^{-2} to 10^{-4} over wireless links compared to 10^{-9} over fiber link) incurred can cause packet errors to become false alarms which trigger unnecessary congestion control and retransmission timer backoff at the transmitting host. Consequently, the end-to-end throughput and latency of detecting packet loss are severely degraded. It is thus necessary to modify the existing protocol to take the erroneous link characteristics into account so that congestion controls are triggered only in the case of a genuine network congestion.

This paper studies the dynamic behavior of TCP over a network consisting of wired and wireless parts with stationary and mobile hosts, with an effort to identify the influence of the high error rate on the performance of TCP over wireless links. In order to make TCP more robust in wireless environment, a simple end-to-end modification using negative acknowledgement (NACK) is proposed to allow the hosts to distinguish corruption from congestion. The performance of the proposed scheme will be compared with an existing modified version (LHACK),

proposed by Cobb [3]. Due to the scope of this paper, discussions to alleviate the effects due to handoff phenomena will not be addressed in this paper.

In Section 2, an analysis of a simplified TCP is used to demonstrate how the throughput is affected by the relatively high error rate of wireless channels. Two modified TCP schemes, one using last-hop acknowledgment and the other using negative acknowledgment, are also discussed in details. In Section 3, simulation results are presented. Finally, Section 4 concludes the paper and discusses handoff-related issues.

II. TCP IN MOBILE WIRELESS ENVIRONMENT

A. Influences of high error rate on TCP performance

Fig. 1. A typical TCP operation cycle

Detailed descriptions of TCP can be found in [7], [8]. In short, the operation of a TCP session consists of a number of periodic cycles (as shown in Fig. 1) each having two distinctive phases, namely the *slow start phase* and the *congestion avoidance phase* [1], [2]. They respectively correspond to an exponential and an linear increase in window size. The transition from the *slow start phase* to the *congestion avoidance phase* is marked by the *slow start threshold*.

In wireless networks, the high bit-error-rate causes significant corruption loss. Whenever a packet corruption occurs, TCP misinterprets this as congestion and undergoes congestion control mechanisms. First, TCP reduces the window size and hence the optimal window size can never be achieved if errors occur often. Second, the slow start algorithm is activated with a reduced threshold (half of the previous window size) which restricts the rate at which the window size grows to the optimal value. The retransmission timer is reset with backoff, leading to slow response to congestion and causing more idle time, particularly when there are multiple losses in a single window. As a result, the performance of TCP is significantly degraded.

To demonstrate this, an analysis on a single TCP connection, made up of one wired link and one wireless link, is considered. An error free feedback path and fixed sized packets are assumed. Although some features of TCP (for examples, the exponential

*Supported in part by Hongkong Telecom Institute of Information Technology under grant HKTIIT93/94.EG01.

¹For TCP, the term “segment” is more appropriate than “packet”. For simplicity, herein, the term “packet” will be used throughout the paper.

RTO timer backoff and fast retransmission) are ignored, the analysis serves to demonstrate the effects of transmission errors in wireless environment on the end-to-end TCP throughput.

The notations used are as follows.

p :	packet corruption rate (PCR)
μ :	service rate of the bottleneck link (wireless link), packets/s
τ :	round-trip propagation delay, s
T :	round-trip delay, $T = \tau + \frac{1}{\mu}$
B :	buffer size at the basestation
C :	capacity of the bottleneck link in unit time T, $C = \mu T$
s :	slow-start threshold
W_{max} :	maximum window size that can be achieved, $W_{max} = \mu T + B$

Using the fluid flow approximation, the evolution of the window size for a cycle starting at $t = 0$ is given as

$$w(s, t) = \begin{cases} 2^{\frac{t}{T}}, & t < t_s \\ s + \frac{t-t_s}{T}, & t_s \leq t < t_c \\ [C^2 + 2\mu(t-t_c)]^{\frac{1}{2}}, & t_c \leq t \leq t_{max} \end{cases} \quad (1)$$

where $t_s = T \log_2 s$, $t_c = t_s + T(C - s)$, and $t_{max} = t_c + \frac{W_{max}^2 - C^2}{2\mu}$.

After integrating Equation (1), the number of successfully transmitted packets up to time t is given as

$$n(s, t) = \begin{cases} 2^{\frac{t}{T}} - 1, & t < t_s \\ \frac{1}{T} \left[s(t-t_s) + \frac{(t-t_s)^2}{2T} \right] + (s-1), & t_s \leq t < t_c \\ (s-1) + n_{CA} + \mu(t-t_c), & t_c \leq t \leq t_{max} \end{cases} \quad (2)$$

where $n_{CA} = \frac{1}{T} \left[s(t_c - t_s) + \frac{(t_c - t_s)^2}{2T} \right]$.

By inverting Equation (2) and letting $n(s, t) = n$ and $t(s, n) = t$, we have

$$t(s, n) = \begin{cases} T \log_2(n+1), & n < s \\ t_s + T \left[\sqrt{s^2 + 2(n-s+1)} - s \right], & s \leq n < (s-1) + n_{CA} \\ t_c + \frac{1}{\mu} [n - (s-1) - n_{CA}], & (s-1) + n_{CA} \leq n \leq n_{max} \end{cases} \quad (3)$$

where

$$n_{max} = (s-1) + n_{CA} + \frac{1}{2} (W_{max}^2 - C^2) \quad (4)$$

If S_m is the slow-start threshold of an arbitrary cycle m and M_m is the maximum window size achieved in cycle m , the conditional probability $P[M_m = j | S_m = i]$, denoted by $q_{i,j}$ with $i \in [1, \lceil \frac{W_{max}}{2} \rceil]$ and $j \in [1, W_{max}]$, is given by

$$q_{i,j} = \begin{cases} (1-p)^{j-1} p, & j < i \\ (1-p)^{\frac{1}{2}(i+j-2)(j-i+1)} p \left(\sum_{a=0}^{j-1} (1-p)^a \right), & i \leq j < W_{max} \\ (1-p)^{\frac{1}{2}(W_{max}+i-3)(W_{max}-i)} \times [p \left(\sum_{a=0}^{W_{max}-1} (1-p)^a \right) + (1-p)^{W_{max}}], & j = W_{max} \end{cases} \quad (5)$$

The evolution of the slow-start threshold S_m for cycle m can now be formulated as a discrete time Markov chain with transition matrix $[p_{i,j}]$ whose elements are given by

$$p_{i,j} = q_{i,(2j-1)} + q_{i,2j}, \quad i, j \in [1, \lceil \frac{W_{max}}{2} \rceil]$$

With the 1-step transition matrix, $[p_{i,j}]$, the steady state probabilities π_s , $s = 1, 2, \dots, \lceil \frac{W_{max}}{2} \rceil$, can be found by solving $\pi = [p_{i,j}] \pi$, where π is the steady state probability vector.

For a given slow-start threshold, s , the probability mass function for the total number of packets successfully transmitted in a cycle, denoted by N , is given by

$$P[N = n | s] = \begin{cases} (1-p)^n p, & n < n_{max}(s) \\ (1-p)^{n_{max}(s)}, & n = n_{max}(s) \end{cases} \quad (6)$$

where $n_{max}(s)$ is given in Equation (4).

With $t(s, n)$, π_s , and $P[N = n | s]$, the average utilization u can be given approximately by

$$u = \frac{1}{\mu} \sum_{s=1}^{\lceil \frac{W_{max}}{2} \rceil} \sum_{n=0}^{n_{max}(s)} \pi_s P[N = n | s] \frac{n}{t(s, n)} \quad (7)$$

If it is further assumed that the average transmission attempts per packet (including the successful trial) can be regarded as independent of the dynamics of the TCP congestion control, the throughput γ can be given approximately by

$$\gamma \approx (1-p)u \quad (8)$$

Fig. 2 and Fig. 3 show the numerical results for cases with wireless link capacity μ equal to 20 packets/s and 100 packets/s respectively. Performance of the TCP connection concerned degrades significantly as the packet error rate increases, particularly for those connections with a large bandwidth-delay product.

Fig. 2. Throughput of a single TCP connection with $\mu=20$ packets/s.

Fig. 3. Throughput of a single TCP connection with $\mu=100$ packets/s.

B. Modified versions of TCP

In order to improve the performance of TCP in the wireless environment, it is necessary to distinguish corruption from congestion or to improve the quality of the TCP connections by partial error recovery in the layer underneath. In [10], the authors discuss three ways of placing the error recovery functions, and they are depicted in Fig. 4. Detailed comparisons of the three approaches can be found in [12].

Fig. 4. Placement of error recovery function for a path consisting of both wired and wireless links

The two modified versions of TCP investigated in this paper can be used in the first approach as well as in the wireless portion of the second approach.

B.1 Negative acknowledgment scheme (NACK)

In order to have the advantage of “time diversity”, cumulative acknowledgment is used to acknowledge the last in-sequence and correctly received packet in NACK scheme. An additional negative acknowledgement (“NACK”) is added in the option field (Fig. 5) to explicitly indicate which packet is received in error so that retransmission of that packet can be initiated quickly, particularly in the case of multiple corruptions in a single window. Under the assumptions that a corrupted packet can still reach the destination and the source address of a corrupted packet is still known, whenever a corrupted packet is received, a “NACK” is sent. Upon the detection of “NACK”, only the corrupted packet is retransmitted by the source and no window size adjustments are performed. After the retransmission, the source resumes normal packet transmission. To avoid inflation of the round-trip time estimate, the round-trip time measurements from all the packets which have been sent before the retransmission of the corrupted packet are ignored.

Fig. 5. Option field for negative acknowledgment in TCP header

B.2 Last-hop acknowledgment scheme (LHACK)

For details of LHACK, [3] can be referred to. As a summary, in LHACK, the wireless router/base station (radio interface) sends a “LHACK” (“FHACK”, first-hop acknowledgment) to the stationary (mobile) source for each packet it receives. For the connection from a fixed source to a mobile destination, if “LHACK” is received at the source but the end-to-end acknowledgment from the destination, “DACK”, is missing, this infers a corruption and no flow control mechanisms are triggered. In

contrast, missing both “LHACK” and “DACK” implies congestion and the standard slow-start congestion control is taken.

The major drawbacks of the LHACK scheme are that two acknowledgments are sent for each message and this results in extra load to the return path, and that the scheme must rely on timeout and fast retransmission to detect corrupted packets and this results in significant throughput degradation when multiple corruptions occur in a single window. For a connection consisting of a mobile and a fixed host, loss of “DACK” does not cause false trigger to congestion control in LHACK. However, for a connection made up of two mobile hosts, in addition to the large processing load due to the 3 types of acknowledgments (“LHACK”, “FHACK”, and “DACK”), loss of “LHACK” on the wireless link to the source can be misinterpreted as network congestion.

III. SIMULATION RESULTS

Fig. 6. Simulation Scenario

The simulation scenario is shown in Fig. 6. It is assumed that the source always has data to send. Packets from both the source concerned and the cross-traffic source are assumed fixed size. Here, only a connection from a fixed source to a mobile destination is considered, in fact, simulation results reveal that the same general conclusions can be drawn for a connection from mobile to fixed and a connection connecting two mobiles. It is also assumed that the receiver can always send out acknowledgment immediately for each data received without delay other than processing delay. The TCP used in the simulation is based on the REAL simulator [11].

Summarized in Table I are some parameters used for the simulations. The assumptions and parameters used seem specific, but general conclusions drawn in the following remain valid for different values of the parameters tested.

Wired link capacity	0.5Mbps
Round-trip wired link propagation delay	60ms
Raw wireless link capacity	50kbps
Round-trip wireless link propagation delay + processing delay	20ms
TCP segment/packet size	500 bit
Wireless router buffer size	40
Maximum window size	100
Minimum slow-start threshold	1
Mean arrival rate of the cross traffic	10 packets/s
Maximum Doppler frequency	1.34 Hz, 80 Hz
Number of diversity branches	2

TABLE I
SIMULATION PARAMETERS

A. Binary channel

In a binary channel, the packets are corrupted randomly and independently. The throughput performance of the three TCP

versions (normal, LHACK, NACK) are shown in Fig. 7 and Fig. 8. Here throughput refers to the effective traffic passed to the application layer on top of TCP and is normalized relative to the average link capacity available for the connection (i.e. link capacity minus cross traffic mean rate). The NACK scheme outperforms the other two — normal TCP and LHACK scheme when ACR (acknowledgment corruption rate) is less than 0.3, which is quite high and no practical networks operate in that range of loss rates. For example, in the wireless LAN standard, IEEE 802.11, the maximum frame error rate specified is 0.08. That is, NACK is superior than LHACK in most of the realistic situations. Throughput of the normal TCP drops abruptly even at small PCR. At small PCR (< 0.02), the performance of both NACK and LHACK are comparable. However, as PCR increases, the throughput of LHACK drops more rapidly than NACK due to the long idle time and window shut down by the consecutive losses at high PCR.

Fig. 9 depicts the acknowledgment traffic as a function of PCR. LHACK has the largest amount of acknowledgment traffic relative to the other two. This is mainly because in LHACK, for each packet received correctly, two acknowledgments are sent. This acknowledgment traffic drops significantly at high PCR because of longer idle time.

Fig. 10 shows the delay (relative to the end-to-end propagation delay) performances of the three schemes. The delay is measured from the time a packet is first transmitted to the instance it is passed to the application layer at the receiver. At $PCR > 0.17$, delay for normal TCP is unreasonably high. NACK has the best performance over the other two.

B. Rayleigh fading channel

Jake's model with time correlation [9] and constant envelop assumption [5] are employed in the simulation for Rayleigh fading channel. Shadow fading effect is not considered as in [5] because it is in general mitigated by other means such as handoff in practical applications.

Fig. 11(a) and Fig. 11(b) show the relationship between throughput and average SNR (Signal-to-Noise Ratio) for slow fading ($f_D = 1.34$ Hz) and fast fading ($f_D = 80$ Hz) respectively. A diversity gain of more than 5dB is seen for all the three schemes at both slow and fast fading. In the practical range of $\bar{E}_b/N_o$ (12 - 17 dB), NACK outperforms the other two in both fast and slow fading.

With slow fading, error occurs in burst leading to consecutive packet losses. LHACK is thus affected significantly ($\sim 4 - 10\%$ loss compared with normal TCP) due to long idle time in cases with consecutive losses. On the contrary, normal TCP is less affected because the window adjustment is triggered less frequently with slow fading. The gain of NACK over normal TCP is thus not very significant ($< 5\%$). However, at small $\bar{E}_b/N_o$, the higher acknowledgment corruption for NACK and the more frequent window adjustment (due to the much larger amount of errors) for normal TCP, make the throughput drop significantly, even smaller than that of LHACK. The above observations are also found in the fast fading case. However, NACK and LHACK have much better throughput performance than normal TCP ($> 25\%$ for NACK and $> 10\%$ for LHACK for most cases) in the fast fading case. This is consistent with the results from binary channel since fast fading creates errors almost randomly with little correlation.

Fig. 12 shows the throughput performance with respect to maximum Doppler frequency at $\bar{E}_b/N_o = 12$ dB. If 2-branch selection diversity is used, NACK has a much better performance than LHACK and normal TCP. If no diversity is used, NACK

has similar performance compared with LHACK. The drop in throughput for NACK is much larger than that of LHACK without diversity. This demonstrates that NACK is very sensitive to acknowledgment loss or corruption relative to LHACK.

IV. CONCLUSIONS

TCP is found to suffer significant throughput loss and delay increase when used over wireless links without necessary protocol modifications. It is found that both NACK and LHACK offer improvements over normal TCP when used in wireless environment. In the practical range of ACR, NACK has the best throughput performance and the smallest interactive delay without much processing burden added to the wireless router or base station, compared to LHACK which requires the cooperation of the wireless router or base station to return a last-hop-acknowledgment. In addition, NACK produces less acknowledgment traffic load in the return path.

Similar to link errors, when a handoff from one cell to the other occurs, there may be extra delay and packets loss. If these are mistaken as signals of congestion, long pause (due to timer backoff) and small window size after handoff can also cause significant throughput reduction. There are some readily available solutions to this (for examples, [4] and [6] make use of the information from the IP layer). Although the NACK scheme cannot cope with the degradation caused by handoff, the available solutions can be used in conjunction with the NACK scheme without problems, thus possibly solving the handoff-related performance degradation problem.

Besides, some ways to reduce the overhead due to the TCP header should also be studied as this is too expensive to carry the long TCP header over the scarce radio bandwidth. Finally, adaptive FEC using information of the channel condition needs to be studied further so that the channel quality can be improved with minimum overheads added.

REFERENCES

- [1] V. Jacobson, "Congestion Avoidance and Control", ACM SIGCOMM Computer Communication Review, Vol.18, No.4, p.314-29, August1988.
- [2] P. Karn, and C. Partridge, "Improving Round-Trip Time Estimates in Reliable Transport Protocols", ACM Transactions on Computer Systems, Vol.99, No.4, p.364-373, November1991.
- [3] J. A. Cobb, P. Agrawal, "Congestion or Corruption? A Strategy for Efficient Wireless TCP Sessions", in Proc. IEEE Symposium on Computers and Communications, p.262-8, 1995.
- [4] R. Cáceres, and L. Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments", IEEE JSAC, Vol.13, No.5, p.850-7, June1995.
- [5] J. C. I. Chuang, "Comparison of Two ARQ Protocols in a Rayleigh Fading Channel", IEEE Transactions on Vehicular Technology, Vol.39, No.4, p.367-73, November1990.
- [6] P. Manzoni, D. Ghosal, and G. Serazzi, "A Simulation Study of the Impact of Mobility on TCP/IP", IEEE JSAC, Vol.13, No.5, p.858-67, June1995.
- [7] D. Comer, Internetworking with TCP/IP, Vol.1, 3rd edition, Englewood Cliffs, N. J.: Prentice Hall, 1995.
- [8] J. B. Postel, "Transmission Control Protocol — DAPRA Internet Program Protocol Specification", RFC793, September1981.
- [9] W. C. Jakes, Microwave Mobile Communication, 3rd edition, Piscataway, N. J.: IEEE Press, 1994.
- [10] M. Naghshineh, M. Schwartz, and A. S. Acampora, "Issues in Wireless Access Broadband Networks", Wireless Information Network — Architecture, Resource Management, and Data Services, Kluwer Academic Publisher, p.1-20, 1996.
- [11] S. Keshav, "REAL: a network simulator", Report 88/472, Computer Science Department, University of California at Berkeley, CA, 1988.
- [12] A. C. F. Chan, D. H. K. Tsang, and S. Gupta, "A modified TCP (Transmission Control Protocol) for Mobile Wireless Environment", available from eeraldar@ee.ust.hk.

Fig. 7. Normalized throughput as a function of packet corruption rate in the forward path for different acknowledgment corruption rates (ACR) in the return path.

Fig. 8. Normalized throughput as a function of acknowledgment corruption rate (ACR) in the return path, at forward path packet corruption rates of 0.01 (solid line) and 0.05 (dotted line).

Fig. 9. Acknowledgment traffic load as a function of forward path packet corruption rate for different acknowledgment corruption rates (ACR) in the return path.

Fig. 10. Average delay (relative to the round-trip propagation delay) as a function of forward path packet corruption rate for different acknowledgment corruption rates (ACR).

Fig. 11. Normalized throughput as a function of $\overline{E_b/N_o}$ with 2-Branch Selection Diversity (2BSD) and no diversity used at (a) $f_D = 1.34$ Hz (slow fading), (b) $f_D = 80$ Hz (fast fading).

Fig. 12. Normalized throughput as a function of f_d at $\overline{E_b/N_o} = 12$ dB, with 2-Branch Selection Diversity (2BSD) and no diversity used.