Enhancing Bluetooth TCP Throughput via Link Layer Packet Adaptation

Ling-Jyh Chen, Rohit Kapoor, M. Y. Sanadidi, Mario Gerla
Department of Computer Science, University of California at Los Angeles
Los Angeles, CA 90095, USA
{cclljj, rohitk, medy, gerla}@cs.ucla.edu

Abstract— **TCP throughput limitations over wireless links have received considerable attention in the last few years. One of the problems is that TCP congestion control interprets packet losses as an indication of congestion, whereas in wireless links, losses could be due to transient link quality degradations. In this paper, we propose and study a link layer solution and evaluate its effects on TCP in the context of Bluetooth. We enhance the Bluetooth link layer to make use of channel state information and accordingly adapt the Bluetooth packet type to enhance TCP throughput. We propose a simple analytical method to determine the optimal packet type for a given channel state by adding FEC support or changing packet size. Since wireless interfaces, such as 802.11 or Bluetooth, can provide information regarding the channel state using relevant APIs, this simple enhancement can easily be added to the link layer. We implemented this functionality in the Bluetooth link layer. Our simulation experiments show that the proposed adaptive packet type solution significantly improves TCP throughput. The throughput enhancement increases with the error rate. For high error rates close to 0.1%, the link layer enhanced with the adaptive scheme is able to maintain good TCP throughput, whereas throughput is almost zero when the adaptive scheme is not used.**

Keywords-component; TCP; Bluetooth; Adaptive Link Layer; FEC; Link Quality

I. INTRODUCTION

TCP [1] is the most widely used transport protocol on the Internet. Its congestion control scheme was designed for the Internet when most links were fiber optics with minimal or no error losses. With the huge growth in the popularity of wireless technologies such as 802.11b, Bluetooth and Cellular, the shortcomings in the operation of TCP over wireless links has become an important issue.

Wireless links are characterized by higher bit error rates and this causes inefficiencies in the operation of TCP [2]. Essentially, any perceived packet loss (occurring because of *error or buffer overflow*) is construed by a TCP sender as occurring due to buffer overflow. The response of TCP to all such events is to invoke its congestion control procedures, resulting in unnecessary window reduction, which causes a drop in the TCP throughput. Note, though, that some of the packet losses occur due to corrupted packets being dropped by the link layer and invoking the congestion avoidance procedures when these events occur is not desirable. Various enhancements to TCP have been proposed to make TCP behave more "intelligently" in the presence of such wireless link errors. These are discussed later in the section on related work.

Most wireless interfaces available today monitor the characteristics of the medium and can provide such information when appropriate APIs are invoked. As explained in [3], using this information, the 'optimal' link layer packet type for the prevailing channel conditions can be adaptively selected. By optimal, we mean the link layer packet type (or link layer packet size) that maximizes the throughput at the link layer. In this paper, we show the effects of adaptively selecting the optimal link layer packet type on the performance of TCP over a wireless link. Using Bluetooth [4] as the wireless technology, we present a simple analytical method to select the 'optimal' link layer packet type, given the channel conditions. We perform simulations under different conditions to show that TCP throughput can be improved significantly by the use of the 'optimal' packet type. One thing to be noted is that though we have demonstrated our results using Bluetooth, this technique could be applied to any other wireless technology, as long as it is possible to adapt the packet length or the use of Forward Error Correction according to the prevailing channel conditions.

In Section 2, we summarize relevant Bluetooth features, and in Section 3 review related work on TCP. In Section 4, we present the analytical method to select the appropriate packet type and discuss its implementation in Bluetooth. In Section 5 we present simulation results, and provide concluding remarks in Section 6.

II. BLUETOOTH OVERVIEW

Bluetooth is a short-range radio technology operating in the unlicensed 2.4GHz ISM (Industrial-Scientific-Medical) frequency band. Its original goal was to replace the numerous proprietary cables to provide a universal interface for devices to communicate with each other. But it soon became a good solution to interconnect devices to form a "personal area network", primarily due to the low cost, low power and small size of Bluetooth chips.

Bluetooth employs FHSS (Frequency Hopping Spread Spectrum) to avoid interference. There are 79 hopping frequencies (23 in some countries), each having a bandwidth of 1MHz. Frequency hopping is combined with fast ARQ (Automatic Repeat Request), CRC (Cyclic Redundancy

Check), and FEC (Forward Error Correction) to achieve high reliability on the wireless links.

Bluetooth units can be connected to each other to form a piconet, which consists of up to eight active units. One of the units acts as a master and the others act as slaves. All the data/control packet transmissions are coordinated by the master. Slave units can only send in the slave-to-master slot after being addressed in the preceding master-to-slave slot. Each slot lasts for 625 microseconds.

For real-time data such as voice, Synchronous Connection Oriented (SCO) links are used, while for data transmission, Asynchronous Connectionless Link (ACL) links are used. There are several ACL packet types, differing in packet length (and consequently, data transmission rate) and whether they are FEC coded or not. The FEC coding scheme used in ACL DM mode is a (15, 10) shortened Hamming code, in which each block of 10 information bits is encoded into a 15 bit codeword, and it is capable of correcting single bit error in each block. Table 1 shows the different ACL packet types and their properties.

TABLE 1: PACKET TYPES FOR BLUETOOTH ACL CONNECTIONS

Mode	FEC	Packet Size	Symmetric	Asymmetric	
DM1	2/3	17 bytes	108.8 kbps	108.8 kbps	108.8 kbps
DM3	2/3	121 bytes	258.1 kbps	387.2 kbps	54.4 kbps
DM5	2/3	227 bytes	286.7 kbps	477.8 kbps	36.3 kbps
DH1	no	27 bytes	172.8 kbps	172.8 kbps	172.8 kbps
DH3	no	183 bytes	390.4 kbps	585.6 kbps	86.4 kbps
DH5	no	339 bytes	433.9 kbps	723.2 kbps	57.6 kbps

Note that, in the symmetric connection mode, both master and slave nodes will occupy the same amount of Bluetooth time slots (625 microseconds in each time slot); whereas in the asymmetric connection mode, the Bluetooth link will occupy 1/3/5 time slots (for DM1/DM3/DM5 or DH1/DH3/DH5 mode) in one direction of this link and only one time slot in the opposite direction.

III. RELATED WORK

The impact of wireless link quality on TCP has been studied extensively in the literature. One of the initial papers to focus on this issue was [2], in which the authors also presented the Snoop solution for TCP. The main reason for the degraded performance of TCP over high bit error rate links is that it treats all lost packets as having been dropped due to congestion in the network and consequently slows down its rate of sending data. This assumption is valid for wired networks where BERs on such links are extremely low (of the order of 10^{-9}) and most packet drops are due to congestion. On wireless links, on the other hand, the quality of the channel can have large variations due to interference and mobility, and BERs can be much higher (as high as 10^{-3}). Thus, the above TCP philosophy of treating all lost packets as congestion losses can be rendered invalid. This wrong interpretation of the cause of packet loss can lead to reduced performance. Such problems become even more serious when contiguous packet losses happen, in which case the TCP sender reduces its congestion window size and backs off its retransmission timeout (RTO) to a very large time interval. This can cause the TCP sender to be "idle" for a long period of time. In fact, a number of studies have shown severely degraded performance for TCP under extremely bad link conditions [2] [5].

Several studies have pointed out possible solutions to TCP problems in error-prone environments. Apart from the Snoop solution, which makes uses of a Snoop Agent to cache TCP segments and retransmits segments only on the wireless link, one of the first works to identify a solution for TCP's degraded performance over wireless links was the Indirect TCP (I-TCP) protocol [6]. I-TCP splits a connection between a fixed station and a mobile host into two connections: one between the fixed host and the base station, and the other between the base station and the mobile host. Split connections are able to separate flow and congestion control on the wired and wireless links and enhance performance, but the violation of the end-to-end semantics is its primary drawback. Some other prominent works have been the Explicit Loss Notification (ELN) scheme [7], and TCP Westwood (TCPW) [8]. ELN is capable of distinguishing packet loss due to congestion from the one by transmission errors, but like Snoop, there is state maintenance overhead at the base station. On the other hand, TCPW employs an end-to-end approach to adjust the congestion window by the estimated "Eligible Rate" of the connection, but it still suffers from low throughput if the bit error rate is really high.

To overcome the effect of low and varying link quality on throughput, the selection of the optimal link layer packet size, given estimated channel conditions, has been proposed in [3], in which the authors analytically evaluate the 'optimal' link layer packet size based on the current bit error rate of the channel. In this paper, we extend this solution and apply it to the Bluetooth link layer. Moreover, we will study the benefit of enabling the link layer to select the 'optimal' packet size, as well as enabling/disabling FEC encoding, on the performance of TCP.

IV. LINK LAYER WITH OPTIMAL PACKET TYPE

Wireless link layers of popular technologies such as Bluetooth and 802.11 can provide information regarding the quality of the channel by the use of APIs called from upper layers. For example, in the Bluetooth specification, the *Get_Link_Quality* function call can be used to obtain link quality information. *Get_Link_Quality* returns the value of the Link Quality between 0 and 255, with the higher value representing a better channel. The BT standard allows vendors to determine how to measure Link Quality and map it to a number between 0 and 255.

As an example, for Bluetooth cards containing CSR (Cambridge Silicon Radio [9]) chipsets, the Link Quality is calculated from the Bit Error Rate in the following manner:

If BER (Bit Error Rate) = 0, LQ (Link Quality) = 255; perfect channel.

*If BER <= 40/40000, LQ = 255 – BER * 40000.*

*If 40/40000 < BER <= 4000/40000, LQ = 215 – ((BER / 32) * 40000).*

*If 4000/40000< BER <= 40000/40000, LQ = 105 – ((BER / 256) * 40000).*

The Link Quality is the average bit error rate on the link with a time-weighted average in units of 1/40000. Fig 1 shows the value of the Link Quality versus the BER for the CSR chipset, which we used in some of our experiments.

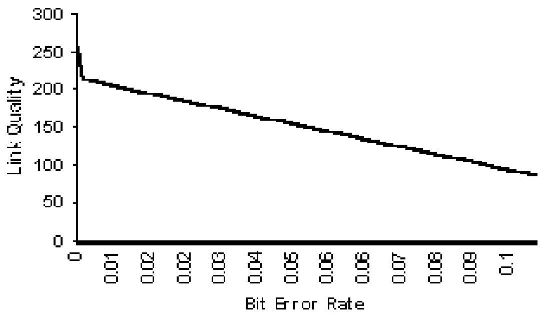

Fig 1: Link Quality vs BER for the CSR chipset

TABLE 2: PACKET SIZES AND LENGTHS OF ACL PACKET TYPES

Packet Type	Packet Size (bits)	Packet Length (Bluetooth Slots)
DH5	2712	5
DM5	1816	5
DH3	1464	3
DM3	968	3
DH1	216	1
DM1	136	1

A. Analytical Evaluation of Optimal Packet Type

We now describe a simple analysis to determine the relative performance of different packet types under different channel BERs (bit error rates). The analysis can then be used to determine the "threshold" values of the BER, i.e., values at which the packet type that gives the best performance changes.

Suppose the BER is b, the packet size (which is given in Table 1 for different packet types) is s bits and the number of Bluetooth slots occupied by a packet type (also given in Table 2) is n.

The packet error rate, *p*, for DH packets is:

$$p = 1 - (1 - b)^{s} \tag{1}$$

Recalling that DM packets are encoded with a 2/3 block FEC, i.e., in every block, 15 bits are used to encode 10 bits of data, and this block coding can correct 1 bit in every block of 15 bits. The packet error rate, *p*, for DM packets can be approximated as:

$$p = 1 - \left((1 - b)^{15} + 15\,b(1 - b)^{14}\right)^{s/15} \tag{2}$$

Assuming that the ARQ scheme retransmits a packet until the acknowledgement of a successful reception, the average number of attempts, $\overline{N}$, needed to successfully transmit one packet is given by:

$$\overline{N} = 1 \times (1 - p) + 2 \times p \times (1 - p) + 3 \times p \times p \times (1 - p) + \ldots$$
$$\Rightarrow \overline{N} = \frac{1}{1 - p} \tag{3}$$

The effective link layer throughput *T* is then given by:

$$T = \frac{s}{\overline{N} \times (n + 1) \times 625\ \mu s} = \frac{s(1 - p)}{0.000625\ \ (n + 1)} \tag{4}$$

where *s* (packet size in bits) and *n* (length of packet in Bluetooth slots) for different packets as given in Table 2, and 625µs is the length of a Bluetooth slot.

Using Eq. 4, we plot the effective throughput of Bluetooth packets versus the bit error rate in Fig 2. Assuming UDP traffic sources, figure 3 shows the relation between bit error rate and packet error rate for different ACL packet types. The robustness of the coded DM packet types is visible in the figure. As the bit error rate increases, the packet error rate of DM packet types increases at a much smaller rate than that of the DH packet types.

The thresholds for best packet type can be determined by comparing the values of throughput T for different BERs and selecting the packet type that gives the largest T as the best packet type for that BER. These thresholds are given in Table 3.

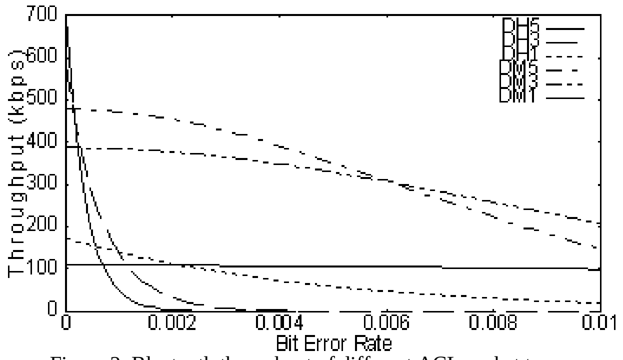

Figure 2: Bluetooth throughput of different ACL packet types

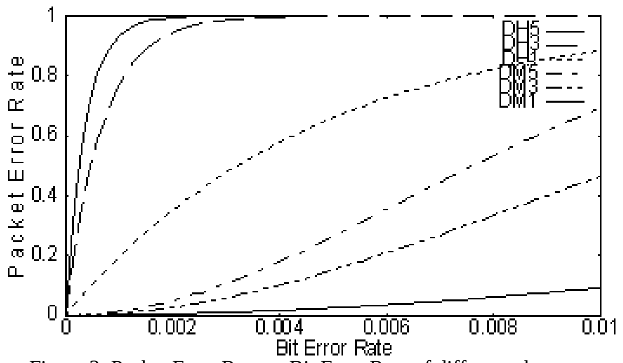

Figure 3: Packet Error Rate vs Bit Error Rate of different pkt types

Using the above analysis, we added a feature to the Bluetooth link layer to enable it to select the optimal packet type, using the thresholds in Table 3, dynamically based on link conditions. We implemented this functionality in our simulator. In later sections, we refer to this as the 'enhanced' adaptive packet type (or APT) link layer.

TABLE 3: ANALYTICALLY CALCULATED THRESHOLDS AT WHICH DIFFERENT PACKET TYPES SHOW BEST PERFORMANCE

Mode	BER range
DH5	<0.0001529
DM5	>0.0001529, and <0.0060795
DM3	>0.0060695, and <0.0157813
DM1	>0.0157813

V. SIMULATION RESULTS

In this section, we present simulation results showing the improvement in performance when using the enhanced Bluetooth link layer. The Bluetooth simulation model was developed as an extension to NS-2 (Network Simulator) [10]. The simulation model implements most of the features of the Bluetooth baseband layer, such as frequency hopping, time division duplexing, multi-slot packets, fragmentation and reassembly of packets. In addition, the simulator enables scatternets and inter-piconet communication by defining gateway nodes to forward packets between piconets. The channel model assumes independent bit errors.

The Bluetooth topologies we used in the simulations is shown in Fig 4. Black circles represent master nodes, and white circles represent slave (or gateway slave) nodes. Each simulation was run for 600 seconds. We varied the bit error rate to obtain performance results under different conditions. We also ran connections over a different number of hops and tried different versions of TCP (Tahoe [11], Reno [11], and NewReno [12]). The TCP packet size was 500 bytes, and the buffer size of each Bluetooth node was set to 9000 bytes.

Note that in the 2-hop case, the capacity of the master (black node) will be shared between the two slaves. Thus, the maximum throughput in this case is half of the maximum throughput in the 1-hop case. For the 4-hop case, the maximum throughput is the same as that of the 2-hop case.

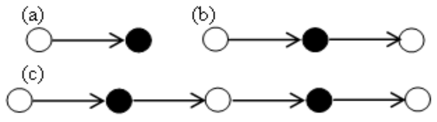

Figure 4: (a) 1 hop (b) 2 hop (c) 4 hop situation

A. *Fixed bit error rate*

In the first set of simulations, we ran a TCP NewReno connection over 1, 2 and 4 Bluetooth hops. We also considered various values of the bit error rate on the link.

Figures 5 (a), (b) and (c) show the throughput obtained by TCP NewReno when running over the enhanced link layer and compare these results with those of a regular link layer. The enhanced link layer clearly outperforms the regular link layer. In fact, for high bit error rates, the throughput of NewReno over a regular link layer goes to zero, while that over the enhanced link layer falls only slightly.

B. *Varying bit error rate*

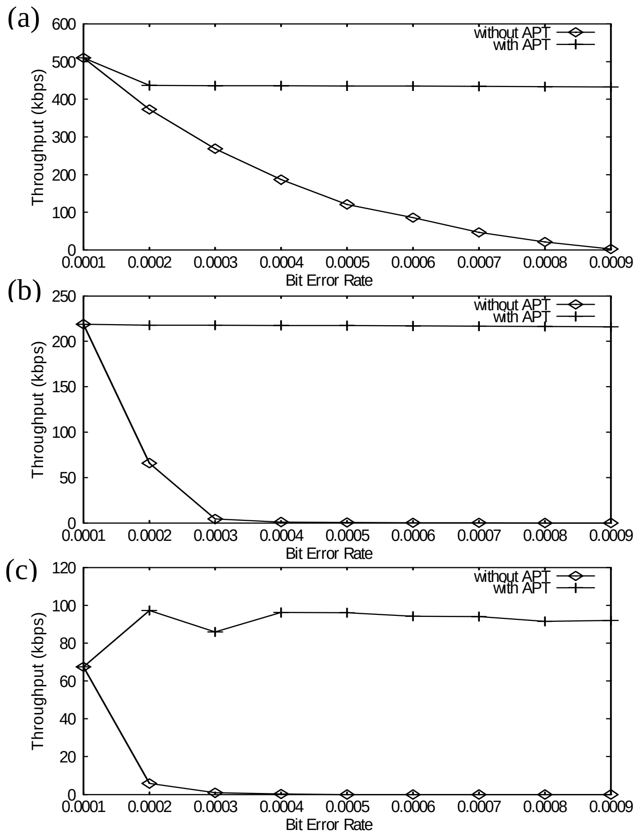

Figure 5: TCP Newreno throughput with/without the APT link layer for (a) 1-hop (b) 2-hops (c) 4-hops

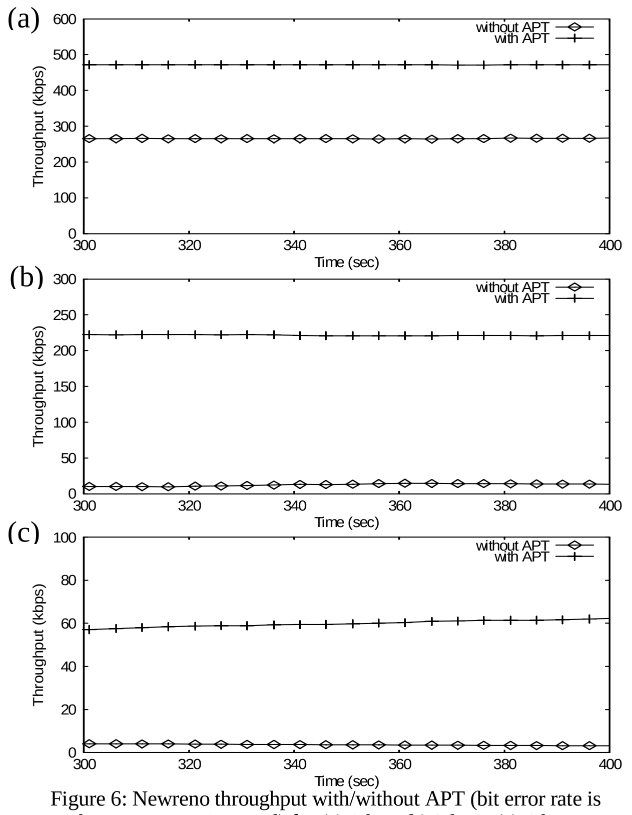

Figure 6: Newreno throughput with/without APT (bit error rate is changing every 1 second) for (a) 1-hop (b) 2-hops (c) 4-hops

We then considered a changing bit error rate environment, in which the bit error rate switches from 0.0001 to 0.0005 and back every 1 second. The aim was to evaluate the adaptation of the APT link layer under changing channel conditions. The TCP version used was NewReno.

Fig 6 (a), (b) and (c) show the throughput results over regular and enhanced link layers. In these results, the TCP throughput is sampled every 10 seconds. The improvement in throughput is again very significant, in some cases almost an order of magnitude.

C. APT with real bit error rate measurement

In order to simulate the APT enabled link layer with more realistic error behavior, we took traces of the channel behavior of a real Bluetooth environment over a 5-minute period and plugged these into our simulator. The measurements environment consisted of two Xircom Bluetooth cards (containing the CSR chipset) at a 5-meter distance, with 802.11b devices, placed near one of the Bluetooth devices, generating interference.

We used the *Get_Link_Quality* function (defined earlier) to obtain the Link Quality at every 100 msec from the Bluetooth device close to the 802.11 devices and converted this into the bit error rate, using the technique explained in Section 4. Fig 7 shows the measured bit error rate over the 10-minute period.

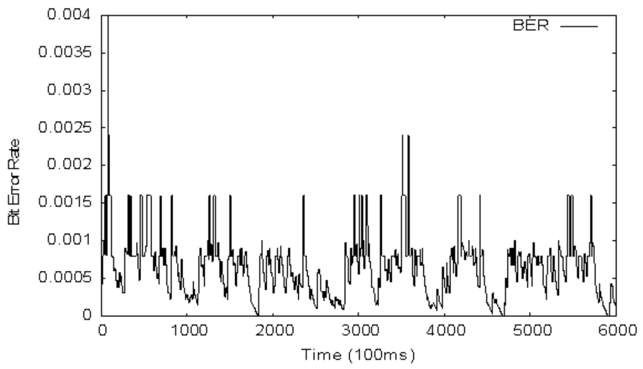

Figure 7: Measured Bit Error Rate in 5 minutes

We plugged this time varying BER trace into our simulator and ran TCP NewReno over 1, 2 and 4-hops. The results are shown in Fig 8. Clearly, the enhanced link layer outperforms the regular link layer.

VI. CONCLUSION

In this paper, we evaluated the effect of selecting the optimal packet type (size) at the link layer on the performance of TCP. We described a simple analytical model to select the optimal packet type when using Bluetooth. We presented an “enhanced” link layer, which selects the optimal packet type based on channel conditions. Using simulation experiments, we showed that the throughput of TCP is improved significantly when the “enhanced” link layer is used, particularly when the bit error rate on the wireless link is high. In a sense, these results show that a well-designed and optimized link layer can make a big difference in performance in a wireless environment.

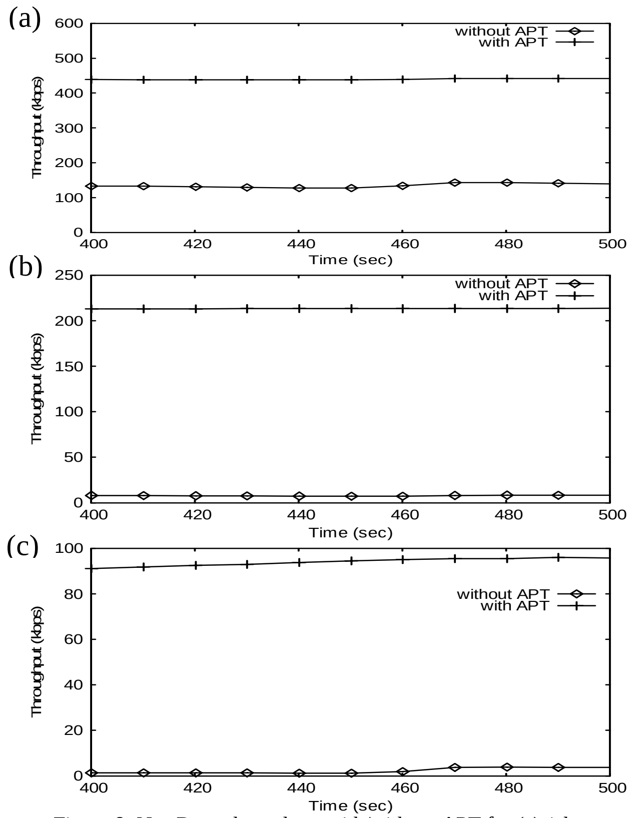

Figure 8: NewReno throughput with/without APT for (a) 1-hop (b) 2-hops (c) 4-hops (with measured bit error rate)

REFERENCES

[1] J. Postel, “Transmission Control Protocol”, RFC 793, Sept. 1981

[2] H. Balakrishnan, S. Seshan, E. Amir, R. Katz, “Improving TCP/IP Performance over Wireless Networks”, Proc. of 1st ACM Conf. on Mobile Computing and Networking, Berkeley, CA, November 1995.

[3] E. Modiano, "A Simple Algorithm for Optimizing the Packet Size Used in ARQ Protocols Based on Retransmission History", Proc. of the Thirty-First Conference on Information Science and Systems, Baltimore, MD, March 1997.

[4] Specification of the Bluetooth System – Core vol.1 v1.1, www.bluetooth.com

[5] H. Balakrishnan, V. Padmanabhan, S. Seshan, R. Katz., ”A Comparison of Mechanisms for Improving TCP Performance over Wireless Links”, IEEE/ACM Transactions on Networking, December 1997.

[6] A. Bakre, B.R. Badrinath, “I-TCP: Indirect TCP for Mobile Hosts”, 15th International Conference on Distributed Computing Systems, 1994.

[7] H. Balakrishnan, R. Katz., “Explicit loss notification and wireless web performance”, In Proceedings of the IEEE Globecom Internet Mini-Conference, November 1998.

[8] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links", In Proceedings of ACM Mobicom 2001, pp 287-297, Rome, Italy, July 16-21 2001.

[9] Cambridge Silicon Radio, www.csr.com

[10] Network Simulator (NS-2), www.mash.cs.berkeley.edu/ns/

[11] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", RFC 2001, January 1997.

[12] S. Floyd, T. Henderson, “The NewReno Modification to TCPs Fast Recovery Algorithm”, RFC 2582.