Improving TCP Performance over Mobile Wireless Environments using Cross Layer Feedback

Vijay T. Raisinghani[*]
Tata Infotech Ltd.(ATG) and
School of Information Technology
IIT Bombay
rvijay@it.iitb.ac.in

Ajay Kr. Singh
Dept. of Computer Sc. & Engg.
IIT Bombay
aksingh@cse.iitb.ac.in

Sridhar Iyer
School of Information Technology
IIT Bombay
sri@it.iitb.ac.in

ABSTRACT

Transmission Control Protocol (TCP) is known to suffer from performance degradation in mobile wireless environments. This is because such environments are prone to packet losses due to high bit error rates and mobility induced disconnections. TCP interprets packet losses as an indication of congestion and inappropriately invokes congestion control mechanisms, which leads to degraded performance. In this paper we show how *cross-layer feedback*, i.e. information from layers above and below, may be used to improve TCP performance over wireless networks.

To illustrate the power of cross layer feedback, we present two novel mechanisms. One, we incorporate *user feedback* into the protocol stack by which the TCP throughput of a desired set of applications running on the mobile host can be dynamically controlled. Other, we propose an approach to use the lower layer *connection* and *disconnection* information, for improving TCP performance. Both these mechanisms entail minimal modification to the mobile host and no modifications to the sender or other network entities.

We have implemented both these mechanisms in ns-2 and the simulation results demonstrate the effectiveness of cross layer feedback.

I. INTRODUCTION

Transmission Control Protocol (TCP) [18] is a reliable, connection-oriented, full-duplex, transport protocol widely used in wired networks. Mobile wireless environments are prone to packet losses, high bit error rates, and mobility induced disconnections. TCP interprets packet losses as an indication of congestion and inappropriately invokes congestion control mechanisms, which leads to degraded performance [20].

Much research has been done proposing various mechanisms for improving TCP performance [3], [9], [13], [14]. However, few approaches utilize the information available at other layers [6]. We believe that leveraging information at other layers may have significant impact on improving the performance of TCP. In this paper we show (i) the benefit of using user level feedback and (ii) using lower layer feedback for improving TCP performance.

Our first contribution is the idea of incorporating *user feedback* into the protocol stack. For example: A user could *dynamically* indicate application priorities and the system may in turn tune the receiver buffers of the various applications to control the throughput of the applications.

While some approaches [10], [11] have been proposed for automatic tuning of TCP buffers at application start, there are others which propose dynamic tuning of the receive buffers based on the varying bandwidth-delay product and application characteristics [8], [15]. However, all of these approaches focus on increasing the overall throughput of applications. Further, none of these approaches take into consideration the dynamic changing of application priorities. We believe that user feedback can be gainfully used by the system to increase the throughput of TCP, as per user specified application priorities.

We show using simulations in ns-2, that user feedback in terms of application priority, implemented as receiver window control (RWC), helps improve throughput of the desired applications.

Our second contribution is the approach of using *lower layer* feedback, in terms of connection and disconnection signals, to improve TCP performance.

While several approaches have been proposed for improving TCP performance under adverse channel conditions [3], [4], [9], [13], [14], there are a few approaches [5], [6], [16] which tackle mobility induced disconnection. Also, most of these schemes do not work well when the sender is the mobile host (MH) instead of the fixed host (FH). We use network layer feedback to appropriately manipulate the TCP congestion control mechanism, leading to enhanced TCP throughput in both directions of data transfer.

We show using simulations in ns-2 that network layer feedback leads to significant improvement in TCP performance.

The paper is organized as follows: section II presents an overview of cross layer feedback. In section III we present our user feedback scheme, section IV presents the receiver window control mechanism. In section V we present our lower layer feedback scheme. Section VI concludes the paper.

II. CROSS LAYER FEEDBACK

It is well known that layering is desirable since it helps in creation of standard modular software components. However, protocol stack *implementations* based on layering do not function efficiently in mobile wireless environments. This is due to the highly variable nature of wireless links and the resource-poor nature of mobile devices. We believe that cross layer feedback may be used, to improve the performance of layered protocol stacks, in wireless environments.

Cross layer feedback can be categorized as follows:

• *Upper to lower layers*: This feedback may be application QoS requirements to lower layers, user feedback, and TCP timer information to lower layers.
One example of upper to lower layer feedback is [7], which presents a model to adapt the maximum number of link-layer retransmissions based on the QoS desired at the transport layer.

[*]This author is a Ph.D student at IIT Bombay, sponsored by Tata Infotech Ltd.

• *Lower to upper layers*: This feedback may be link characteristics and network connectivity information to upper layers. A few of the schemes that use lower layer feedback to improve the performance at upper layers like TCP are as follows: [6] propose using Mobile-IP layer feedback for inducing *fast retransmit* at TCP. This helps in reducing the multiple timeouts in TCP due to cellular handoffs. Along similar lines, [19] propose the use of layer2 hand-off information at the Mobile-IP layer to reduce Mobile-IP hand-off latency.

In the next section, we present our idea of upper to lower layer feedback i.e. user feedback.

III. USER FEEDBACK

While some aspects of cross layer feedback have been explored [6], [7], [19], to the best of our knowledge none of these schemes employ user inputs for dynamically controlling application priority.

While the system, based on heuristics, can take certain actions, a user may be able to take *better* decisions, which may be contrary to the decision of the system. For example: a user may see an approaching tunnel and know by experience that a disconnection will occur inside the tunnel. This information if given to the system, to adapt pro-actively, rather than react to signal deterioration or disconnection, can enable it to prioritize user specified applications.

Another example of user feedback is dynamic application prioritisation. A user may be running multiple applications on his MH, such as a *ftp* application and a video conference. By default, the system may assume the real-time application (video conference) to be of higher priority than the *ftp*. However, in view of impending disconnection a user may want the *ftp* to take higher priority. This is contrary to the usual assumption that the video-conference is of higher priority. Further, these priority requirements may change over time. In the next section we discuss one method, for controlling dynamic application priority, by modifying the receiver window.

A. *Implementing user feedback*

User feedback can be communicated to the appropriate layers of the protocol stack by having a module which captures user inputs, like application priority, and conveys it through a separate control path to lower layers like TCP. The module may interpret the user inputs into layer specific information. For example: application priority information may be mapped to receiver window control.

The details of the mechanisms for communicating user feedback is beyond the scope of this paper.

In the next section, we describe one method of mapping user feedback about application priorities to lower layer specific information i.e. the receiver window control mechanism.

IV. RECEIVER WINDOW CONTROL (RWC)

TCP uses congestion and flow control mechanisms to avoid swamping the network or the receiver [12]. When the sender is not congestion window limited, the receiver can control the transmission rate of the sender by advertising a window, which reflects the buffer state at the receiver. The current TCP implementations have fixed receive buffer sizes for all applications. Application level APIs are available, that allow an application to set its receiver buffer at the start of a connection [17]. However, once set it cannot be modified to reflect changes in application priorities.

We note that throughput for a TCP connection is is decided by the receiver window setting and the corresponding bandwidth-delay product [18]. In case of multiple flows, each having a different bandwidth-delay product, each of the flows will have a different optimum receiver window(awnd). This property of awnd's relation to the bandwidth-delay product can be exploited to *intentionally* make some of the TCP sessions get lower throughput, and thus dynamically control the application priorities. We call this approach Receiver Window Control(RWC). This assumes that total *actual* receiver buffer space is large enough to allow manipulation(increase or decrease) of the awnd *values* for the different sessions.

The intuitive benefits of using RWC can be seen from the following examples: Consider a user running multiple downloads on a wireless device. Now, the user increases the priority of a particular download. Through RWC, the advertised window and thus the throughput can be increased for the higher priority download and by decreasing the advertised window, the throughput can be decreased for the lower priority downloads. This control is dynamic and is invoked as and when the user changes application priorities.

A. *RWC details*

The actions for RWC are summarized below:
Let, n be the number of applications,
$B =$ Bandwidth available to MH on the bottleneck link, which is shared among the applications running on MH. B is assumed to be constant.
$R = rtt$ for all the applications on the MH and is assumed to be constant
For the i^{th} application, let – *before* user feedback $awnd_i =$ initial advertised window, $x_i =$ initial user defined priority, and *after* user feedback $awnd'_i =$ new advertised window (computed using RWC), $x'_i =$ new user defined priority. $awnd_i$, $awnd'_i$, x_i and x'_i are normalized integer values.
We note that the following condition holds:

$$\sum_{i=1}^{n} awnd_i = \sum_{i=1}^{n} awnd'_i = A, \; where \; A = B * R \qquad (1)$$

When user changes priority x_k of application k to x'_k, change advertised window of the applications as follows:

$$awnd'_i \quad = \quad round(\frac{x'_i}{\sum_{j=1}^{n} x'_j} * A), \; \forall i \neq k \qquad (2)$$

$$awnd'_k \quad = \quad A - \sum_{i=1}^{n} awnd'_i, \; i \neq k \qquad (3)$$

The priorities are used for relative ordering of the applications and actual numbers have no significance.

For example, consider three applications. Let, $A = 30$, $x_1 = 1$, $x_2 = 1$, $x_3 = 1$. From equation 2: $awnd_1 = 10$, $awnd_2 = 10$ and $awnd_3 = 10$. Now, after user feedback, let $x'_1 = 2$ ($x'_2 = 1$ and $x'_3 = 1$). Now, from equation 2: $awnd'_2 = round(\frac{1}{4} * 30) = 8$, $awnd'_3 = 8$ and from equation 3: $awnd'_1 = 30 - 16 = 14$. Thus, it can be seen that RWC increases awnd for the higher priority application and decreases awnd for the lower priority applications.

In the next section, we discuss our simulation setup and results for RWC.

B. RWC Simulation Setup

To validate our idea we have conducted some preliminary simulations, using ns-2. The simulation setup is shown in figure 1.

Two ftp flows, f_1 and f_2, were run from the two FHs, N1 and N2 respectively. Each simulation run was of 10s duration. The user feedback was simulated by setting the *window_* parameter of the TCP sessions, at a time of 5 seconds. The initial priorities were: $x_1 = 1$ and $x_2 = 1$. This leads to $A = 32$ packets. Thus from equation 2, $awnd_1 = 16$ and $awnd_2 = 16$. After feedback, $x'_1 = 2$ and $x'_2 = 1$. Thus from equation 2, $awnd'_1 = 11$ and from equation 3, $awnd'_2 = 21$.

One set of simulations were done assuming no losses on the links. A second set was done, assuming a loss of 0.1% on the link N3-N4, before and after feedback.

C. Observations

- *Scenario 1: No loss with no RWC* – (figure 2). As expected the throughput of both f_1 and f_2 is found to be equal, each ≈ 1 Mbps.
- *Scenario 2: No loss with RWC* – (figure 3) the throughput for f_1 increases to ≈ 1.31 Mbps (increase by 31%) and that for f_2 decreases to ≈ 0.69 Mbps (decrease by 31%).
- *Scenario 3: 0.1% loss with RWC* – (figure 4) user feedback has the effect of increasing the throughput for f_1, even if there are some packet losses.

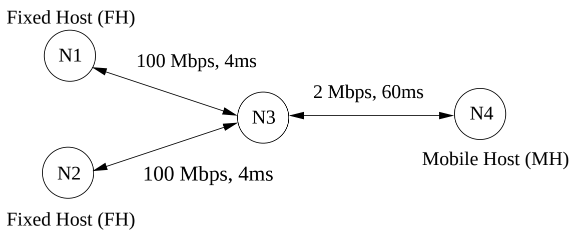

Fig. 1. RWC: Simulation setup

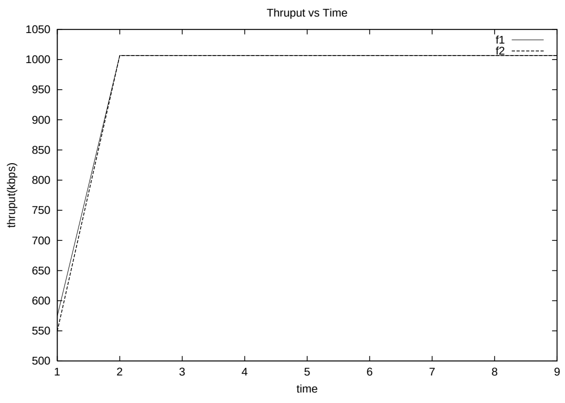

Fig. 2. Scenario 1: No loss with no RWC

These simulation results validate our intuition about RWC being useful for application prioritisation. Further detailed experiments are needed before we can draw stronger conclusions about RWC.

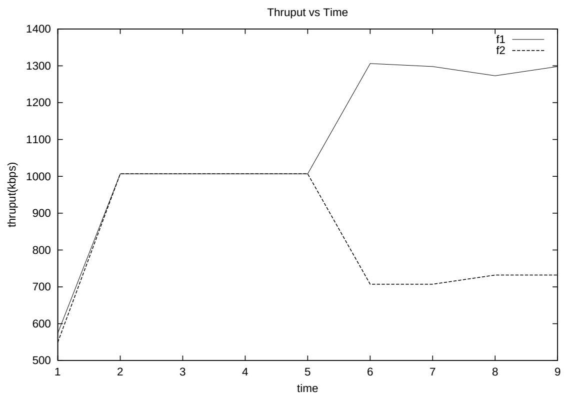

Fig. 3. Scenario 2: No loss with RWC

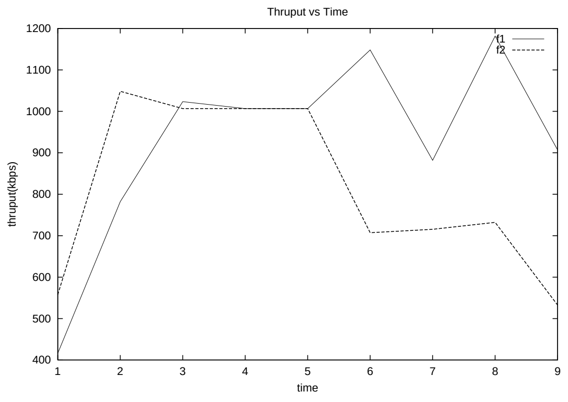

Fig. 4. Scenario 3: 0.1% loss with RWC

V. LOWER LAYER FEEDBACK

We now describe our approach, ATCP, which utilizes lower layer feedback regarding status of connectivity. ATCP assumes that the network layer sends a connection event signal to TCP when MH gets connected to the network and a disconnection event signal when the MH gets disconnected from the network. It uses this information along with the RTO information to enhance TCP throughput.

The working of ATCP is summarized in tables I and II.

For MH to FH transfer, the corresponding actions enable ATCP to quickly regain the cwnd value prior to disconnection, thus reducing under utilization of the available link capacity.

For FH to MH transfer, the corresponding actions prevent TCP at FH from taking congestion control measures when packets are lost due to MH being disconnected.

A. ATCP Simulations and Observations

Figure 5 shows the simulation setup. Only RTT $\approx$ 5ms graphs are being presented for illustration. Detailed simulation results for other values may be found in [1], [2]

For MH to FH data transfer (figure 6) having short RTT connections, ATCP shows a throughput improvement of upto 40% as compared to TCP Reno. TCP Reno performance degrades due to its halving of *ssthresh* and exponentially increasing the retransmission timer, at each RTO. 3DA [6] and Freeze-TCP [16] do not

Event	**State**	**ATCP action**
Disconnection	Sending window open	Do not wait for ACK for packets sent, cancel retransmission timer
	Sending window closed and waiting for acks	Do not cancel RTX; wait for RTO event
Connection	Sending window open	Send data and set new RTX. ACK for new data acknowledges data sent before disconnection
	Sending window closed and RTO	Retransmit
	Sending window closed and no RTO	Wait for RTO
RTO	Disconnected	set *ssthresh* = *cwnd* at disconnection and set *cwnd* = 1
	Connected and DisconnectionOccurred = true	Retransmit lost packet without modifying *ssthresh* or *cwnd*

TABLE I
ATCP: MH IS TCP SENDER

Event	**ATCP action**
ATCP delays the ACK for the last two bytes by *d* milliseconds (at most 500 ms)	
Disconnection	Update network connectivity status
Connection	Send ACK for first pending byte with zero window advertisement
	Send ACK for second pending byte with full window advertisement

TABLE II
ATCP: MH IS TCP RECEIVER

mention any actions when the MH is the sender, hence ATCP not compared with these approaches in this case.

For FH to MH data transfer (figure 7), ATCP shows improvement in throughput over TCP Reno and 3DA [6]. In WLAN environments, the performance of ATCP is similar to that of Freeze-TCP and both show an improvement of upto 40% over TCP Reno. This is because both ATCP and Freeze-TCP reduce the idle period on reconnection which occurs after a mobility induced disconnection.

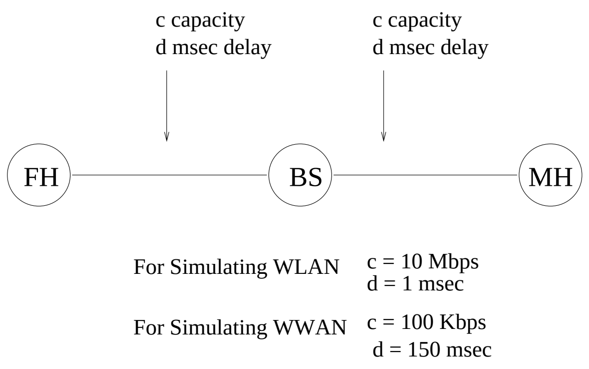

Fig. 5. ATCP simulation setup

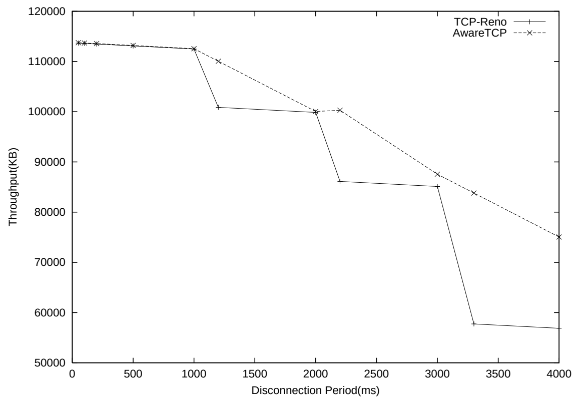

Fig. 6. MH to FH data transfer: RTT ≈ 5ms

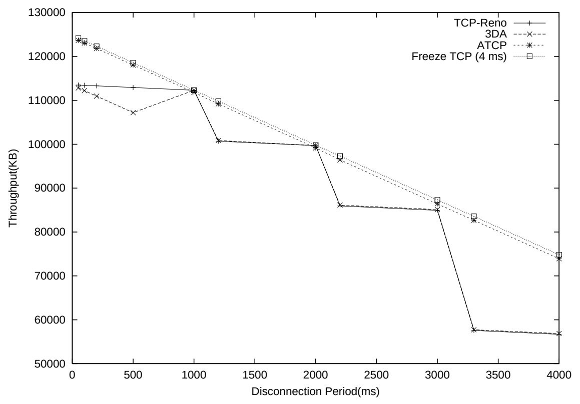

Fig. 7. FH to MH data transfer: RTT ≈ 5ms

We have done additional simulations, details of which are available in [1], [2]. These simulations show ATCP throughput improvement of upto 150% as compared to TCP Reno, in WLAN environments.

VI. CONCLUSION

We have presented cross layer feedback and discussed its benefits. Based on the mechanism of cross layer feedback, we proposed two new approaches RWC and ATCP. RWC manipulates the advertised window of the applications running on a MH. There are other schemes which propose using advertised window to increase the overall throughput of the applications[8], [10], [11], [15]. However RWC, differs from these schemes by dynamically incorporating user specified application priorities. Based on the user input, RWC increases the advertised window and thus throughput for higher priority applications and decreases it for the lower priority applications. Our other scheme, ATCP uses connection and disconnection feedback from the network layer at the MH, to improve TCP performance. There have been proposals in the past to improve TCP performance in mobile wireless environments, but most focus on improving the FH to MH transfer. ATCP enhances TCP throughput in both directions of data transfer. Further ATCP, is not dependent on the prediction of the disconnections as in [16].

Our future research shall focus on exploring new mechanisms for cross layer feedback and other aspects of user feedback.

REFERENCES

[1] Ajay Kr. Singh, "ATCP: Adapted TCP for Mobile Wireless Environments," Master's thesis, Department of Computer Science and Engineering, IIT Bombay, January 2002.

[2] Ajay Kr. Singh and Sridhar Iyer, "ATCP: Improving TCP Performance over Mobile Wireless Environments," In *Fourth IEEE Conference on Mobile and Wireless Communications Networks*, Stockholm, Sweden, September 2002.

[3] Ajay Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," *15th International Conference on Distributed Computing Systems*, 1994.

[4] Hari Balakrishnan, Srinivasan Seshan, Elan Amir, and Randy H. Katz, "Improving TCP/IP Performance over Wireless Networks," In *Proceedings of the First Annual International Conference on Mobile Computing and Networking*, pages 2–11, ACM Press, 1995.

[5] Kevin Brown and Suresh Singh, "M-TCP: TCP for Mobile Cellular Networks," *ACM SIGCOMM Computer Communication Review*, 27(5):19–43, 1997.

[6] Rámon Cáceres and Liviu Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments," *IEEE Journal on Selected Areas in Communications*, 13(5):850–857, June 1995.

[7] Carla-Fabiana Chiasserini and Michela Meo, "Improving TCP over Wireless through Adaptive Link Layer Setting," In *IEEE GLOBECOM, Symposium on Internet Performance (IPS 2001)*, San Antonio, TX, 22, November 2001.

[8] Mike Fisk and Wu-chun Feng, "Dynamic Right-Sizing in TCP," In *Proceedings of LACSI Symposium 2001*, October 2001.

[9] Hari Balakrishnan, V. N. Padmanabhan, Srinivasan Seshan, and Randy H. Katz, "A Comparison of Mechanisms for Improving TCP/IP Performance over Wireless Networks," *IEEE/ACM Transactions on Networking*, 1997.

[10] H. T. Kung and Kolin Chang, "Receiver-Oriented Adaptive Buffer Allocation in Credit-Based Flow Control for ATM Networks," In *INFOCOM*, 1995.

[11] Jeffrey Semke, Jamshid Mahdavi, and Mathew Mathis, "Automatic TCP Buffer Tuning," In *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, Vancouver, British Columbia, Canada, October 1998, ACM Press.

[12] M. Allman and V. Paxson and W. Stevens, "RFC2581: TCP Congestion Control," April 1999.

[13] Saverio Mascolo, Claudio Casetti, Mario Gerla, M. Y. Sanadidi, and Ren Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links," In *Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking*, pages 287–297, ACM Press, 2001.

[14] Prasun Sinha, Narayanan Venkitaraman, Raghupathy Sivakumar, and Vaduvur Bharghavan, "WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks," In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 231–241, ACM Press, 1999.

[15] N. T. Spring, M. Chesire, M. Berryman, V. Sahasranaman, T. Anderson, and B. Bershad, "Receiver Based Management of Low Bandwidth Access Links," In *INFOCOM*, 2000.

[16] T. Goff and J. Moronski and D. S. Pathak and V. Gupta, "Freeze-TCP: A true end-to-end Enhancement Mechanism for Mobile Environments," Israel, 2000.

[17] Von Welch, "A User's Guide to TCP Windows," `http://archive.ncsa.uiuc.edu/People/vwelch/net_perf/tcp_windows.html`, 1996.

[18] W. Richard Stevens, *TCP/IP Illustrated, Volume I: The Protocols*, AWL, 1994.

[19] Jon Chiung-Shien Wu, Chieh-Wen Cheng, Nen-Fu Huang, and Gin-Kou Ma, "Intelligent Handoff for Mobile Wireless Internet," *Mobile Networks and Applications*, 6(1):67–79, 2001.

[20] George Xylomenos and George C. Polyzos, "Internet Protocol Performance over Networks with Wireless Links," *IEEE Network*, 13(4):55 – 63, July/August 1999.