TCP Performance Enhancement over UMTS Network with RNC Feedback

A. Jayananthan, Harsha R. Sirisena
Electrical and Computer Engineering,
University of Canterbury, New Zealand

Krzysztof Pawlikowski
Computer Science and Software Engineering,
University of Canterbury, New Zealand

{a.nanthan@elec., h.sirisena @elec., krzysztof.pawlikowski@}canterbury.ac.nz

Abstract

TCP optimization for wireless networks to deal with packet losses due to fading, shadowing and contention should preferably maintain TCP end-to-end semantics with minimal dependence on intermediate nodes. The development of advanced 3G networks and services makes it necessary to find a way of improving TCP's efficiency and resource utilization. Previous research on this issue suggests that TCP needs radio network feedback to distinguish wireless related losses from congestion related losses. This paper presents such a mechanism that notifies the TCP sender of any non-congestion related losses by introducing a proxy at the RNC node of the UMTS network. Only a minimal change to the standard TCP is required to achieve this. OPNET is used in this study and the simulation results show that the proposed scheme significantly improves the TCP performance.

1. Introduction

Third Generation (3G) technology is revolutionizing the capabilities of mobile communications. 3G mobile networks are expected to provide more enhanced services than are possible over existing cellular systems, including higher bit rates services and greater capacity and service capability. Universal Mobile Telecommunications System (UMTS) is a 3G wireless technology offering data rates up to 2Mbps. The rapid development of wireless network technologies, such as UMTS, WiFi and WiMAX, is making it imperative to design and optimize wireless networks to get maximum benefit with minimal cost.

Being dominant in wired networks, and due to the desire for internetworking between wired and wireless networks, Transmission Control Protocol (TCP) has become the most commonly used transport protocol in mobile Internet technology built on wireless cellular networks [1]. TCP is a reliable connection-oriented protocol employing a window-based congestion control mechanism to avoid network congestion [2]. TCP uses a congestion control algorithm to control the amount of data that the sender can inject into the network [3].On opening a connection, the TCP sender enters the slow start phase in which congestion window (cwnd) is increased by one maximum size segment (MSS) per acknowledgement (ack) received. Exponential growth ensues until cwnd reaches the slow-start threshold (ssthresh) when it enters the congestion avoidance phase.

During congestion avoidance, cwnd is incremented by 1 MSS per RTT and this phase continues until congestion is detected. When a TCP sender detects segment loss using the retransmission timer, it will enter into slow start. Else fast retransmit is triggered if the sender receives a third duplicate acknowledgement (dupack) before timeout occurs. It then retransmits the lost segment, saves ssthresh as half of the current cwnd and increments cwnd by 1 MSS for each additional dupack. When the next ack arrives that acknowledges new data, cwnd is set to the ssthresh and the TCP fast recovery phase ends. Notice that the sender reduces its cwnd, or enters into slow start, even if the segment loss is due to a temporary wireless effect rather than congestion.

Solutions previously proposed to improve TCP performance over wireless networks fall into three broad categories: link layer solutions, split connection schemes and extensions to TCP itself [4]. In this paper, we present a new technique that improves TCP performance over a UMTS network by enabling TCP sender to distinguish non-congestion related packet losses from congestion related losses.

The rest of the paper is organized as follows. Section 2 gives an overview of UMTS. Previous solutions to improve TCP performance over wireless links are surveyed in Section 3. The new scheme and its OPNET model are given in Sections 4 and 5, respectively. Section 6 presents the simulation results. The conclusions drawn and ideas for future work are presented in Section 7.

2. UMTS protocol overview

A UMTS network consists of three interacting domains: Core Network (CN), UMTS Terrestrial Radio Access Network (UTRAN) and User Equipment (UE). Figure 1 shows the UTRAN architecture based on 3G TS 25.401 UTRAN Overall Description [5]. The CN includes the serving GPRS support node (SGSN) and the gateway

GPRS support node (GGSN). The GPRS support nodes (GSNs) include all GPRS functionality needed to support GSM (Global System for Mobile Communications) and UMTS packet services. The SGSN monitors user location and performs security functions and access control. The GGSN contains routing information for packet-switched (PS) attached users and provides internetworking with external PS networks such as packet data network (PDN).

The UTRAN consists of a set of Radio Network Subsystems (RNS) connected to CN through Iu interfaces. An RNS consists of a Radio Network Controller (RNC) and one or more Node-Bs. An RNC manages a set of base stations, referred as Node-Bs, through interfaces called Iub. The RNC exerts admission control for new mobiles or services attempting to use the Node-B. Admission control ensures that mobiles are only allocated radio resources up to what the network has available. The RNC is also responsible for the Handover decisions that require signaling to the UE. Each UE has exactly one Serving RNC and can have one or more Drift RNCs, where the mobile physical layer communications terminate. Drift RNCs communicate with the Serving RNC via the Iur interface. A Drift RNC may also be the Serving RNC where no soft handover activity is in progress.

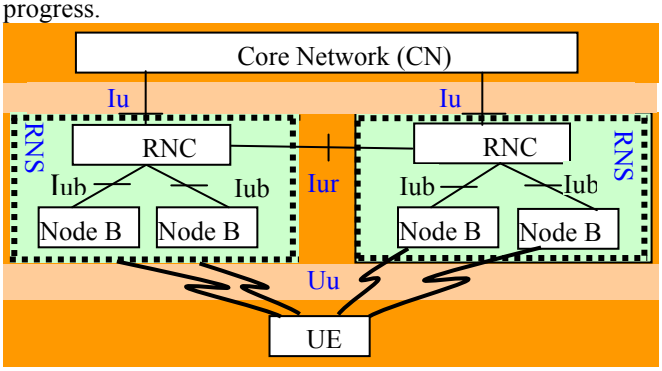

Figure 1 UTRAN architecture

UMTS uses Wideband Code Division Multiple Access (WCDMA) to carry the radio transmissions. WCDMA uses direct spread spectrum with a chip rate of 3.84 Mcps and employs a 5MHz channel bandwidth (BW). A Node-B can support Frequency Division Duplex mode (FDD), Time Division Duplex mode (TDD) or dual-mode operation. In FDD mode, the uplink (from Node-B to mobiles), and the downlink (from mobiles to Node-B) are on different frequencies. The radio frame has a length of 10 ms and is divided into 15 slots. Spreading factors vary from 256 to 4 for an FDD uplink and from 512 to 4 for an FDD downlink. With these spreading factors, data rates of up to 2 Mbps are attainable.

3. Related work

In this section, we survey some protocols proposed to improve TCP performance over wireless links.

Adaptive-TCP (A-TCP) [6] is a TCP-aware link layer protocol and performs local retransmission, sender freezing and flow control in order to hide the wireless environment from the TCP sender. A cross-layer congestion avoidance scheme in [7] gathers network capacity information such as BW and delay at the link layer and, based on these measurements, it adjusts the outgoing data stream. Link layer proposals [8] try to hide packet losses across the wireless link from the TCP sender by employing error correction using techniques such as forward error correction (FEC) and retransmitting the lost packets locally in response to the automatic repeat request (ARQ) message.

A study of Wireless TCP proposals with proxy servers in the GPRS network [9] analyses the Split mechanism [10] and the Snoop protocol [4] and concludes that the Snoop protocol does not perform well because of the high delays in the GPRS radio channel while the Split mechanism slightly improves the TCP throughput. Snoop is a link-layer protocol that retransmits the lost packets locally, trying to hide packet losses across the wireless link from the TCP sender, while the Split mechanism attempts to separate loss recovery over the wireless link from that over the wireline network.

The abovementioned protocols attempt to either hide or separate the wireless effects from that of traditional wireline network. However, they all fail to completely recover from the wireless effects [4, 11]. In this paper, we propose a new technique that utilizes the RNC feedback (RNC-FB) to completely distinguish the wireless packet losses form the congestion related losses.

4. Proposed RNC feedback mechanism

Previous studies show that minimizing the wireless environment effects could improve the TCP performance. In reality, they do optimize the link layer protocols and this in effect improves the TCP performance. The actual TCP improvements should be achieved by tuning the TCP itself to utilize the available network resources efficiently in both wireline and wireless environments. We propose a scheme that monitors the UMTS network radio interface and notifies the TCP sender of any effects caused by the wireless link. The TCP end-to-end semantic is maintained but it is modified in order to adapt to the characteristics of the wireless environment.

An IP packet entering the UMTS network is double encapsulated, as shown in Figure 2, to cross the UMTS network. The GPRS Tunneling Protocol (GTP) sets up the GTP tunnels between the GGSN and SGSN. When this packet reaches the RNC node, the original IP datagram is obtained at the GTP layer of the RNC protocol stack and passed on to the RNC layer for delivery to the destination UE. Our proposed scheme modifies the GTP layer of RNC protocol stack in order to

be able to monitor the IP datagram flows and to extract TCP header information, assuming that the IP datagram is not encrypted. Extracted TCP header information is maintained in a cache table and used to observe wireless environment effects and prepare the RNC feedback to the TCP sender. The functionality of the GTP layer and the required modifications is briefly explained next.

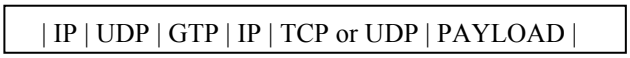

Figure 2 IP Datagram double encapsulation

When receiving a packet from the lower layer (UDP Layer), the GTP layer in the RNC protocol stack takes one of two possible actions, depending on the packet type. If the packet is a GTP signaling message, GTP processes it locally and acts accordingly, otherwise it decapsulates and forwards the packet to the higher layer. If a packet is delivered by the upper layer (RNC layer), it encapsulates and tunnels the packet or directly delivers the packet to the UDP layer if the Iur interface implementation is ready. We modified implementations of both GTP encapsulation and decapsulation states to generate RNC-FB.

The GTP encapsulation state is modified to monitor all the packet flows and to maintain a cache table with the TCP header information, including the TCP connection information to support multiple TCP connections. The GTP decapsulation state is modified to utilize the cached TCP header information to provide RNC feedback to the TCP sender as follows: it monitors all the packet flows and uses the cache table to detect wireless packet losses. If a wireless packet loss is detected, it notifies the TCP sender of this by utilizing the control bit next to the CWR flag in the reserved field of the TCP header, called the Radio Network Feedback (RNF) flag. The standard TCP should be tuned to accommodate this effect so as to avoid unnecessary congestion window reduction and spurious timeouts due to wireless environment effects.

Impact of transmission errors on TCP reduces the network resource utilization; it forces the TCP to reduce the cwnd by invoking unnecessary congestion control measures, resulting in poor throughput. TCP is modified to accept the RNF control flag and is tuned to perform wireless fast retransmit without reducing the congestion window if the RNF flag is set. Wireless fast retransmit is triggered when the TCP sender receives two dupacks with the RNF flag set to quickly recover from wireless packet losses. The sooner the fast retransmit occurs, the better TCP performs because the TCP recovery phase ends and the congestion avoidance phase is entered sooner. It also minimizes spurious TCP timeouts, resulting in further TCP performance improvement.

5. Implementation of the proposed scheme

There are eleven states in the OPNET implementation of the GTP process model [12]. Modifications of "decap" and "gtp encap" states are required to implement our proposed scheme and are briefly explained in Figures 3 and 4, respectively.

```
if (RNC feedback turned on)
{while (packets arrive)
 {/* Check the payload packet format.*/
 if (packet format is ip_datagram)
 { if (getTcpInfo success)
  {if (tcpInfo data length > 0)
   {if (NewInCache table)
    { if (this is a new connection)
     {if (AddTable success)
      { cacheTCPInfo;
        /* First packet for this connection */  }}}
    else /*found in connection table*/
    {cacheTCPInfo;}}/*end of NewInCache*/
 }}/*end of getTcpInfo success*/
 }}/*end of while*/
}/*end of RNC feedback turned on */
```

Figure 3 Pseudo code to modify GTP "decap" state

```
if (RNC feedback turned on)
{while (packets arrive)
 {/* Check the payload packet format.*/
 if (packet format is ip_datagram)
 { if (getTcpInfo success and tcpInfo ACK flag set)
   { if (tcpInfo FIN flag set){destroy cache table;}
    else if (this connection is already recorded)
     {if(tcpInfo ack number > last acked number)
      /*new ack*/
       { destroy cache table; /*Up to this ack*/
        update last acked number in this connection table;
        reset the dupack counter; /* end of new ack*/
       else if (dupack >= 0)
       {if (tcpInfo data length == 0) { dupack++;}
        if (dupack == 1){/*we do not send feedback
            because the acknowledged packet might have
            arrived out of order*/}
         else if (dupack >= 2)
         { if (this missing packet is found in cache table)
          {/*send RNC feedback in the form of RNF
flag*/
            Set TCP RNF bit;
       }}}}/*end of new ack*/
      }/*end of connection found*/
    }}/*end of tcpInfo ACK flag set*/
  }}/*end of while*/
```

Figure 4 Pseudo code to modify GTP "gtp encap" state

Note that our modification to the GTP "decap" and "gtp encap" states looks only for the IP datagram flow to extract TCP header information because the information flow between SGSN and UE is not only data packets. There is also some UMTS management frames exchanged between these nodes. Actually, before any data flow can happen, first the UE has to complete its GMM GPRS attachment with the CN, and then go through PDP (Packet Data Protocol) Context Activation with the SGSN node for the QoS (Quality of Service) class of the data flow.

TCP Reno [3], the current de facto standard for TCP, is fine-tuned to minimize the wireless effects on its performance by adding the RNF flag into TCP header as;

#define TCPC_FLAG_RNF 0x100

When TCP sees an ack with the RNF flag set it can confirm that packet is lost and retransmit that packet. However, it is tuned to perform wireless fast retransmit on receiving two dupacks with RNF flag set because unnecessary TCP retransmissions will contribute to the waste of valuable network resources and significant degradation of TCP end-to-end throughput.

6. Simulation Results and discussion

To demonstrate the effectiveness of our proposed scheme, the UMTS network model shown in Figure 5 is implemented, in turn, with the standard RNC and the modified RNC process model in the RNC protocol stack. The FTP server is configured to generate files of sizes 100 Kbytes and 2 Mbytes. UEs are configured to download 100 Kbyte FTP files except for one UE (UE_0), which downloads 2 Mbyte FTP files and is used for analyzing our simulation results. Modified TCP Reno and UMTS with their default parameters [12] are used in all simulation scenarios. An extract of the TCP Reno parameter values are given in Table 1. Wireless packet drops are generated in mobile nodes (UEs) using a uniform probability distribution.

Receive Window Size at FTP server	8760 bytes
Receive Window Size at UEs	Default
Delay ACK Mechanism	Segment/Clock Based
Maximum ACK Delay	0.2 seconds
Maximum ACK Segments	2
Maximum Segment Size (MSS)	Auto-Assigned
Sack Option	Disabled
RTT Gain	0.125
RTT Deviation Gain	0.25
RTT Deviation Coefficient	4
Slow-Start Initial Count	1 MSS
Minimum RTO	1 seconds
Maximum RTO	64 seconds

Table 1 TCP Reno Parameters

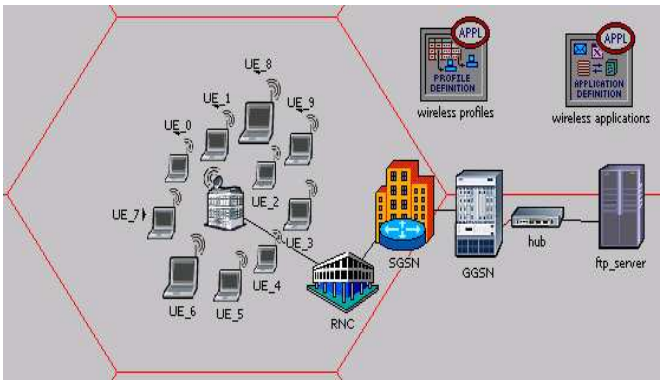

Figure 5 UMTS network model

6.1 Scenario 1 Results and Observations

Figures 6 and 7 show the number of dropped packets, number of cached TCP headers and the TCP congestion window size response of our proposed scheme and that of the standard TCP over UMTS model for packet drop rates of 2% and 5% respectively. Figure 8 compares the TCP sent segment sequence number performance of our proposed scheme with that of the standard UMTS for different packet drops rate. In Figures 6 and 7, it is seen that our proposed scheme, with the aid of RNC-FB, significantly increases the TCP congestion window size; it recovers most of the wireless packet losses and minimizes the number of spurious timeouts by early triggering the wireless fast retransmit algorithm.

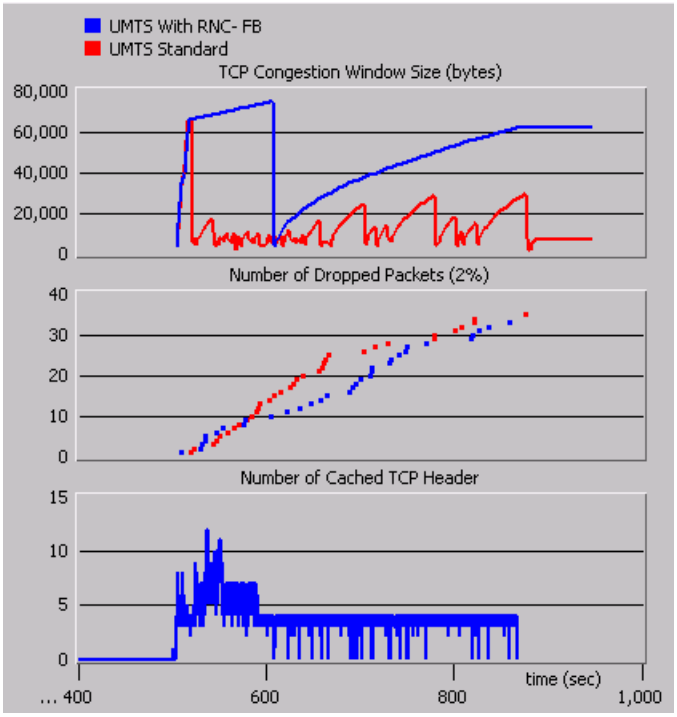

Figure 6 TCP cwnd, dropped and cached packets

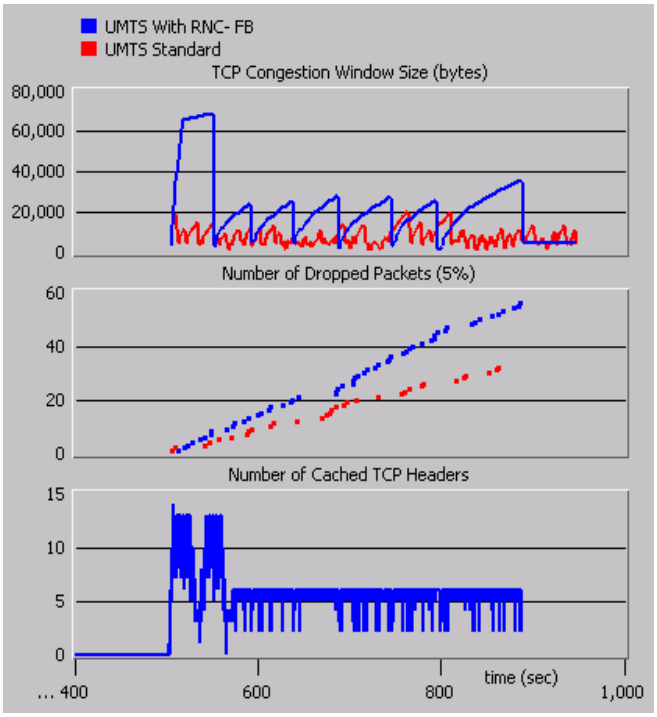

Figure 7 TCP cwnd, dropped and cached packets

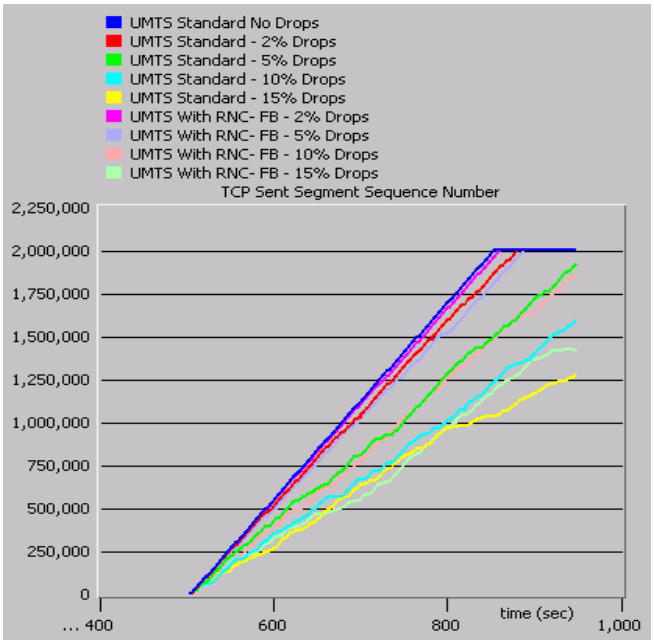

Figure 8 TCP sent segment sequence number

From the TCP performance summary, shown in Table 2, it is also observed that our proposed scheme significantly improves the TCP performance with high packet error rates. The number of cached TCP headers in Figures 6 and 7 shows that our proposed scheme adds very little overhead to the RNC process model. Since the TCP header information is cached at the RNC, it can also be used to provide additional performance enhancement by freezing the TCP to handle TCP timeouts caused by either handoffs or by temporary wireless disconnections.

Packet Drop Rates (%)	2	5	10	15
UMTS Standard TCP Throughput (Kbps)	42.49	24.55	28.48	22.59
UMTS With RNC-FB TCP Throughput(Kbps)	44.68	41.55	33.56	25.73
TCP Improvement (%)	***5.14***	***20.26***	***17.81***	***13.88***

Table 2 Summary of TCP performance

6.2 Scenario 2 Results and Observations

This scenario is designed to see the effects of using TCP Reno with the SACK option enabled on our proposed scheme. Figure 9 compares the number of dropped packets, number of cached TCP headers and the TCP congestion window size response of our proposed scheme with and without the SACK option enabled for a 10 % packet drop rate. Figure 10 compares the TCP sent segment sequence number response of TCP Reno with and without the SACK option enabled for different packet drop rates. In Figure 9, it is seen that TCP Reno with SACK option enabled performs much better than does TCP Reno with the SACK option disabled. However, with the packet drop rates of 2% and 10 %, its performance seems to be similar to that with the SACK option disabled. This effect is shown in Figure 10

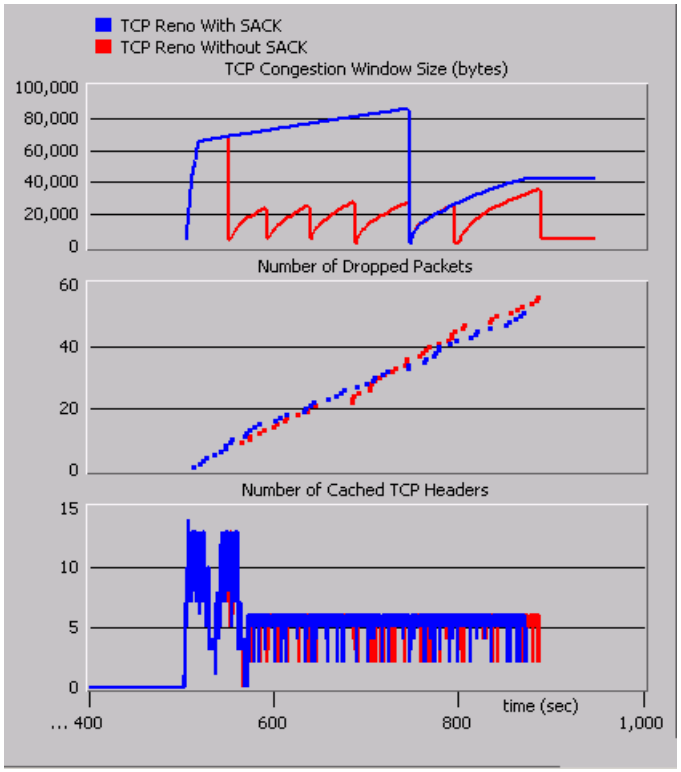

Figure 9 TCP cwnd, dropped and cached packets

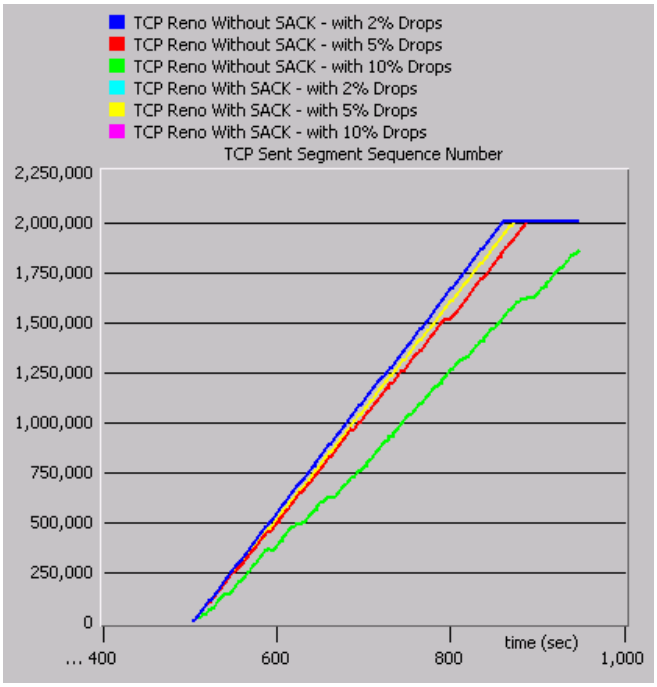

Figure 10 TCP Sent segment sequence number

7. Conclusions and Future work

We have proposed and implemented a radio network transport layer feedback system in the UMTS network model and run simulation studies to validate the model by comparing its performance with that of the standard TCP over UMTS model. The simulation results showed that the new RNC-FB scheme significantly improved the TCP performance compared to that of standard TCP over UMTS. Specifically, the new scheme recovered most of the wireless packet losses and minimized the number of spurious timeouts by early triggering of the wireless fast retransmit algorithm, introduced in TCP Reno as a modification to accommodate this effect. Utilizing one of the reserved control flags (RNF) enabled the TCP sender to successfully distinguish wireless packet losses from losses due to congestion, thereby avoiding unnecessarily invocation of the congestion control mechanism, resulting in a higher TCP performance. A minimal modification to the standard TCP is required while maintaining its end-to-end semantics.

The effect of using TCP Reno with the SACK option was also investigated. It was found that our proposed scheme with the TCP Reno SACK option enabled performs better than with the SACK option disabled when the packet drops rate is moderate, otherwise the performances are quite similar.

The size of the cache table required to record TCP headers is quite small and so would not add significant overhead to the RNC process. The TCP header information cached at the RNC can also be used to provide additional performance enhancement by freezing the TCP sender to handle timeouts caused by either handoffs or by temporary wireless disconnections. We are currently investigating the effect of freezing TCP during handoffs on UMTS network performance and also implementing the RNC-FB scheme with other TCP flavors to investigate whether they, too, would benefit from it.

8. References

[1] F. Xin and A. Jamalipour, "TCP performance in wireless networks with delay spike and different initial congestion window sizes," *Computer Communications*, vol. In Press, Corrected Proof.

[2] V. Jacobson and M.J. Karels, "Congestion avoidance and control", *Proc. ACM SIGCOMM,* pp. 314-329, November 1988

[3] W. R. Stevens, "TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," *IETF RFC 2001,* 2001.

[4] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *Networking, IEEE/ACM Transactions on*, vol. 5, pp. 756-769, 1997.

[5] 3GPP, "3G TS 25.401: UTRAN Overall Description (Release 1999),"

[6] S.-J. Seok, S.-K. Youm, S.-W. kim, and C.-H. Kang, "A modification of TCP flow control for improving end-to-end TCP performance over networks with wireless links," *Computer Communications*, vol. 26, pp. 1998-2010, 2003.

[7] D. Kliazovich and F. Granelli, "Cross-layer congestion control in ad hoc wireless networks," *Ad Hoc Networks*, vol. In Press, Corrected Proof.

[8] J. W. K. Wong and V. C. M. Leung, "Improving end-to-end performance of TCP using link-layer retransmissions over mobile internetworks," *Proc. IEEE ICC'99,* pp 324-328 1999.

[9] J. Rendon, F. Casadevall, and J. Carrasco, "Wireless TCP proposals with proxy servers in the GPRS network," *Proc. IEEE PIMRC*, pp 0-1 2002.

[10] A. Bakre and B. R. Badrinath, "I-TCP: indirect TCP for mobile hosts", *Proc. IEEE ICDCS,* pp 136-143, May 1995.

[11] T. N. Prasun Sinha, Narayanan Venkitaraman, Raghupathy Sivakumar, Vaduvur Bharghavan *"WTCP: a reliable transport protocol for wireless wide-area networks,"* Kluwer Academic Publishers, March 2002.

[12] OPNET, "Making Networks and Applications Perform," OPNET Technologies Inc, http://www.opnet.com/.