On the Performance of TCP Vegas over UMTS/WCDMA Channels with Large Round-Trip Time Variations

Anthony Lo[1], Geert Heijenk[2], and Ignas Niemegeers[1]

[1] Delft University of Technology, P O Box 5031,
2600 GA Delft, The Netherlands
{A.Lo, I.Niemegeers}@ewi.tudelft.nl
http://www.wmc.ewi.tudelft.nl/

[2] University of Twente, P O Box 217,
7500 AE Enschede, The Netherlands
Geert.Heijenk@utwente.nl
http://wwwhome.cs.utwente.nl/∼heijenk

Abstract. Universal Mobile Telecommunications System (UMTS) is a third-generation cellular network that enables high-speed mobile Internet access. This paper evaluates and compares the performance of two well-known versions of Transmission Control Protocol (TCP), namely, Vegas and Reno, in a UMTS environment. Bulk data transfer was considered in the simulation with varying radio channel conditions. We assume that data losses are only due to the radio channel. Simulation results show that the performance of Vegas is worse than Reno even though data losses incurred by the radio channel are completely recovered by the UMTS radio link control layer. This has led us to conduct a thorough investigation on the behavior of Vegas in order to identify the cause of performance degradation in Vegas. The poor performance of Vegas is attributed to the UMTS radio interface characteristics which resulted in large and highly variable TCP round-trip times. Vegas would interpret the round-trip time variation as a sign of congestion, and consequently, shrink its window size which reduces the transmission rate. Furthermore, a sudden increase in the instantaneous round-trip time can trigger spurious timeouts at the TCP sender using Vegas which performs unnecessary retransmissions. Spurious timeouts can lead to significant throughput reduction. Reno, on the other hand, does not show any abnormality and delivers the expected performance.

1 Introduction

Universal Mobile Telecommunications System (UMTS) [1, 2] is a third-generation cellular mobile network where the radio interface is based on code division multiple access, known as Wideband Code Division Multiple Access (WCDMA). To date, a number of mobile operators in Europe and Asia have launched their UMTS commercial service and some plan to roll out their UMTS networks. In addition to the legacy voice service, UMTS enables mobile users access to the Internet in a seamless fashion (i.e., always on) at data rates up to 2 Mb/s in indoor or small-cell environments, and wide-area coverage of up to 384 kb/s.

Today, the Internet is the most popular and widely used packet-switched network that supports applications like File Transfer Protocol (FTP), Email, etc. These Internet applications rely on two commonly used protocols, namely, Transmission Control Protocol and Internet Protocol (TCP/IP) [3], to reliably transport data across heterogeneous networks. IP is concerned with routing data from source to destination host through one or more networks connected by routers, while TCP provides reliable end-to-end data transfer and congestion control. TCP Reno is the most extensively used variant of TCP, while TCP Vegas [4] is a newer variant with improved congestion avoidance and retransmission mechanisms. Unlike Reno, Vegas constantly tries to detect congestion in the network before packet loss occurs and lower the rate linearly when sign of congestion is detected. On the contrary, Reno only reacts when packet losses are detected.

The performance of TCP over wireless networks has been extensively studied [5–8]. All these studies show that TCP performance is significantly degraded since TCP interprets packet losses due to the radio channel as signs of network congestion, which resulting in sender throttling and causes significant throughput reduction. Various solutions were proposed in the literature to combat TCP performance degradation, and in general, can be classified into three major categories: link-layer [7–10], split-connection [11] and proxy [6]. However, all these studies including the proposed solutions were purely targeted at Reno rather than Vegas. The solution employed by UMTS falls under the link-layer category. Presently, it is not clear how the link-layer solution used by UMTS would adversely affect the performance of Vegas.

The paper aims at evaluating the performance of TCP Vegas and compare its performance with Reno in a UMTS environment, in particular, how the performance of Vegas is adversely affected by the UMTS radio interface. We employ a simulation-based approach to analyze the performance of TCP Vegas and Reno over UMTS for FTP traffic with varying channel conditions.

2 Universal Mobile Telecommunications System (UMTS)

2.1 System Architecture

Fig. 1 shows a simplified architecture of UMTS for packet-switched operation [2, 12], which consists of one or several User Equipments (UEs), the UMTS Terrestrial Radio Access Network (UTRAN) and the core network. The UTRAN is composed of several Node Bs connected to a Radio Network Controller (RNC). The core network, which is the backbone of UMTS, comprises the Serving GPRS Support Node (SGSN) and the Gateway GPRS Support Node (GGSN). The SGSNs route packets to and from UTRAN, while GGSNs interface with external IP networks. UE, which is a mobile station, is connected to Node B over the UMTS radio interface.

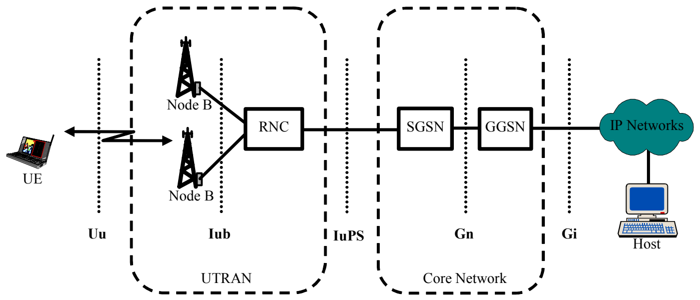


Fig. 1. UMTS Network Architecture

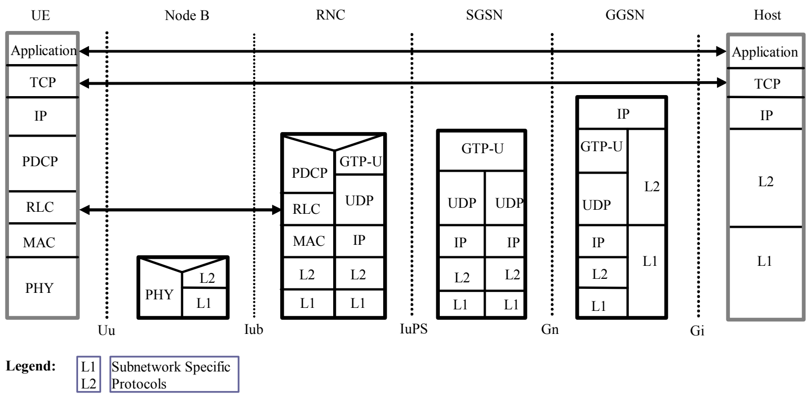


Fig. 2. UMTS Protocol Architecture for the User Plane

2.2 Protocol Architecture

Fig. 2 depicts the UMTS protocol architecture for the transmission of user data which is generated by TCP-based applications. The applications as well as the TCP/IP protocol suite are located at the end-nodes, namely, the UE and a host.

The Packet Data Convergence Protocol (PDCP) provides header compression functionality. The Radio Link Control (RLC) layer can operate in three different modes: acknowledged, unacknowledged and transparent. The acknowledged mode provides reliable data transfer over the error-prone radio interface. Both the unacknowledged and transparent modes do not guarantee data delivery.

The Medium Access Control (MAC) layer can operate in either dedicated or common mode. In the dedicated mode, dedicated physical channels are allocated and used exclusively by one user (or UE), whereas in the common mode, users share common physical channels for transmitting and receiving data.

The Physical (PHY) layer contains, besides all radio frequency functionality, spreading, and the signal processing including RAKE receiver, power control, forward error-correction, interleaving and rate matching.

3 TCP Vegas

Both TCP Reno and Vegas perceive packet losses as a sign of network congestion. Reno only reacts to network congestion when packet losses are detected via timeout or three duplicate acknowledgements. On the other hand, Vegas continuously monitors the state of the network and increment or decrement the current window size in order to prevent packet drops due to buffer overflowing in the intermediate routers. Vegas detects signs of incipient congestion by comparing the expected throughput to the measured throughput, which is given as follows [4]:

$$\Delta = (expected - actual) \times RTT_{base} \tag{1}$$

where $expected = windowSize/RTT_{base}$ and $actual = windowSize/RTT$.

$windowSize$ is the current window size which is the number of segments in transit; RTT_{base} is the minimum of all the instantaneous Round-Trip Times (RTTs); and RTT is the average round-trip time measured for each individual segment transmitted in the $windowSize$. Round-trip time is defined as the total time required by the TCP sender to transmit a segment through a network and receive an acknowledgement that the segment was received correctly.

Vegas defines two thresholds α and β, which are normally set to 1 and 3, respectively. When $\Delta < \alpha$, Vegas increases the congestion window linearly in the next round-trip time; and when $\Delta > \beta$, Vegas decreases the congestion window linearly in the next round-trip time. The congestion window is unchanged when $\alpha < \Delta < \beta$.

In the case of Reno, packet losses are detected via the receipt of three duplicate acknowledgements or retransmission timeout expiration. The latter resets the congestion window size to one segment, while the former reduces the congestion window by one half of the current window size.

4 Simulation Models

In order to analyze the performance of TCP Vegas and Reno over UMTS, network-level simulations were carried out using *ns-2* [13], which is an event-driven simulator. Several extensions were made to this simulator for modeling UMTS. The extensions were developed within the framework of the IST SEACORN project [14]. With the extensions, instances of UMTS nodes, viz., UE, Node B and RNC can be created.

The model used for simulation analysis is illustrated in Fig. 3. The model is based on the system architecture discussed in the previous section (see Fig. 1). UE, Node B, RNC and host are modeled according to the aforementioned protocol stack illustrated in Fig 2. The TCP/IP protocol stack of *ns-2* was used.

Since the primary aim of the simulation was to investigate the impact of the radio interface on end-to-end TCP performance, we assume that no packet losses, errors or congestion on either the Internet or the UMTS core network. Hence, the TCP performance is solely attributed to the radio interface. The links between two nodes are labeled with their bit rate (in bits per second) and

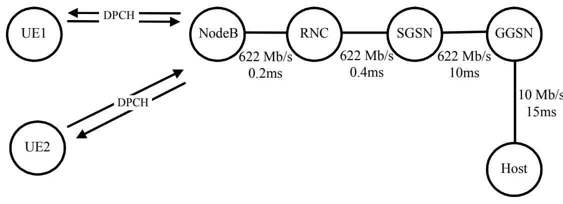


Fig. 3. Top Level Simulation Model

delay (in seconds). Each link capacity was chosen so that the radio channel was the connection bottleneck. Hence, the TCP performance is solely attributed to the UMTS radio interface. Consequently, the functionality of SGSN and GGSN was abstracted out and modeled as traditional *ns* nodes since generally they are wired nodes and, in many ways, mimic the behavior of IP router. Currently, no header compression technique is supported in the PDCP layer. In the following subsections, the UMTS model is described in detail.

4.1 RLC Model

The RLC model supports both the acknowledged and unacknowledged modes. For TCP-based applications, the acknowledged mode was used in the simulation since the acknowledged mode was designed to hide losses due to radio channels from TCP. The retransmission strategy adopted by the acknowledged mode is the Selective-Repeat ARQ (Automatic Repeat reQuest) scheme. With Selective-Repeat ARQ, the only RLC blocks retransmitted are those that receive a negative acknowledgement. An RLC block consists of a header and a payload which carries higher layer data.

A status message is used by the receiver for notifying loss or corruption of an RLC block. The status message is in bitmap format. That is, bit_j indicates whether the jth RLC block has been correctly received or not. The frequency of sending status messages is not specified in the standard [15]. However, several mechanisms are defined, which can trigger a status message. Either the sender or the receiver can trigger the status message. Table 1 and Table 2 list the triggering mechanisms for sender and receiver, respectively. It is important to note that not all the triggering mechanisms are needed for the Selective-Repeat ARQ to operate. However, a combination of triggering mechanisms, which deliver optimum performance, is sought.

The advantage of receiver-initiated mechanisms is that the receiver has direct information about missing blocks. For the sender-initiated mechanisms, the sender has first to request a status message by enabling the poll flag in the RLC block and wait for a reply, which has longer turn around time. Therefore, receiver-initiated mechanisms are preferred. Nevertheless, sender-initiated mechanisms are required to prevent deadlocks and stall conditions. Periodic mechanisms might be more robust compared to others but may result in too frequent status message. In addition, a timer is required at the sender and receiver for

Table 1. Sender-Initiated Mechanisms

Trigger	Explanation
Last Block in buffer or retransmission buffer	*status report is requested by enabling the poll flag in the RLC header*
Every m blocks	poll flag is enabled for every blocks
Every n service data units	poll flag is enabled for every n service data units
Utilization of Send Window	*poll flag is enabled when the Send Window is x% full*

Table 2. Receiver-Initiated Mechanisms

Trigger	Explanation
Detection of missing blocks	*status message is generated once a gap is detected in the RLC sequence number*
Estimated block counter	status message is generated if not all the retransmitted blocks are received within an estimated period

proper operation of the triggering mechanisms. At the sender, the timer is called poll timer, which is started when a request for status messages is sent to the receiver. If the status message from the receiver does not arrive before the timer expires, the sender repeats the same procedure again. The receiver is equipped with a timer called status prohibit timer, which controls the time interval between status messages if triggered consecutively. If the interval is too short, then bandwidth is wasted. On the other hand, if the interval is too long, bandwidth is preserved, but delay increases. The selected triggering mechanisms for the RLC model are the rows written in *italics*.

4.2 MAC Model

The MAC model implemented the dedicated mode. It requests the number of blocks buffered at the RLC layer, which are ready for transmission, and submits to the PHY layer as transport blocks. In this case, each transport block corresponds to an RLC block since no MAC header is required in the dedicated mode as depicted in Fig. 4. The frequency in which the PHY layer can accept transport blocks from MAC is defined by the Transmission Time Interval (TTI). In the UMTS standard, the values of TTI are 10 ms, 20 ms, 40 ms, and 80 ms, where TTI = 10 ms corresponds to the duration of one radio frame.

4.3 PHY Model

The PHY model is responsible for transmitting transport blocks over the physical channels using one or multiple radio frames. For the MAC dedicated mode, the transport blocks are sent over the Dedicated Physical Channel (DPCH). DPCH is a bi-directional channel dedicated to a single user only. The bit rates and

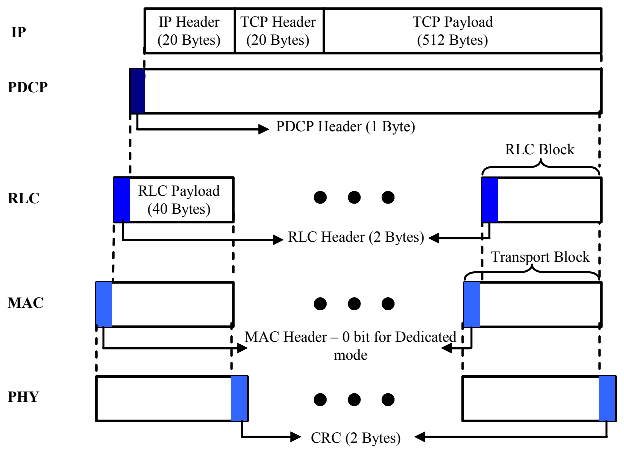


Fig. 4. IP Packet Data Transfer

the TTI associated with the DPCH channel used in the simulation are shown in Table 3. Note that the bit rates exclude RLC headers. Since the PHY layer passes the transport block to the MAC layer together with the error indication from the Cyclic Redundancy Check (CRC), the output of the PHY layer can be characterized by the overall probability of transport block error – also called Transport Block Error Rate (TBLER) in this paper. Thus, an error model based on uniform distribution of transport block errors, was used in the simulation. It is valid to assume that the erroneous transport blocks perceived by the RLC is independent and uniformly distributed as a result of interleaving and forward error-correction mechanisms used by the PHY layer. The TBLER, in the range from 0 to 30%, was considered in the simulation.

The transmission of an IP packet over the radio interface is illustrated in Fig. 4. The RLC entity receives a PDCP packet which comprises an IP packet of 552 bytes or an acknowledgement of 40 bytes, and additionally the PDCP header of 1 byte. This PDCP packet is segmented into multiple RLC blocks of fixed sizes. Each of these blocks fits into a transport block in which a CRC is attached at the PHY layer. In the simulation, the RLC header and the payload size was set to 2 bytes and 40 bytes, respectively. For this RLC payload size and a bit rate of 384 kb/s, twelve transport blocks can be transmitted within one TTI of 10 ms. The other simulation parameters are summarized in Table 3.

5 Simulation Results

5.1 TCP Throughput

End-to-end TCP *throughput* is used as the performance measure. The throughput (in bits per second) is defined as the amount of successfully received TCP segments by the receiver within the simulation duration. The TCP throughput

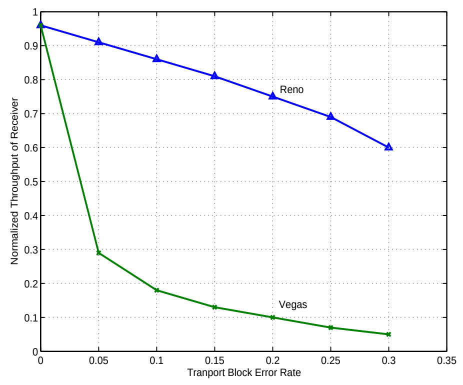
Table 3. Simulation Parameters

Application	File Transfer Protocol (FTP)		
TCP	TCP Variants	Vegas and Reno	
	Window Size (Segments)	64	
	Maximum Segment Size (Bytes)	512	
	TCP Header Size (Bytes)	20	
IP	IP Header Size (Bytes)	20	
	IP Packet Loss Rate in the Internet	0%	
PDCP	TCP/IP Header compression	No	
RLC	RLC Mode	Acknowledged Mode with In-sequence delivery	
	Window Size (Blocks)	4096	
	Payload Size (Bits)	320	
	RLC Header (Bits)	16	
	Max Bit Rate (kb/s)	Uplink	Downlink
		64	384
MAC	MAC Header (Bits)	0	
	MAC Multiplexing	Not required for DPCH	
PHY	Physical Channel Type	DPCH	
	Transport Block Size (Bits)	336	
	TTI (ms)	Uplink	Downlink
		20	10
	Transport BLER	0 – 30%	
	Error Model	Uniform Distribution	

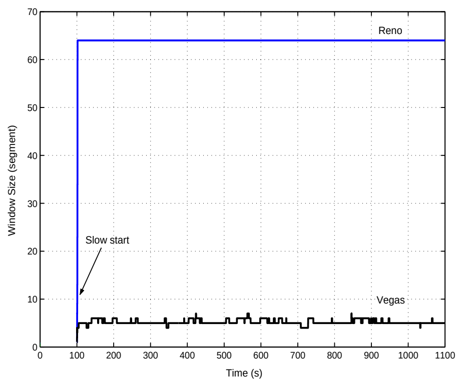
was obtained using a single FTP session between a UE and a host. Data is transferred from the host to the UE. That means, the only higher layer data going in the opposite direction (or uplink channel) are TCP acknowledgements, which justifies for using lower bit rate in the uplink channel.

The FTP session was run for 1000 s, which is equivalent to 100,000 radio frames. Firstly, the simulation was run using TCP Vegas, and then, the same set of simulation was repeated for TCP Reno. Fig. 5(a) depicts the plots of TCP throughput as a function of TBLER for Vegas and Reno. For both Reno and Vegas, the throughput is normalized to the maximum downlink channel bit rate, i.e., 384 kb/s.

Under ideal radio channel condition (i.e., 0% TBLER), both Reno and Vegas attain the maximum throughput of approximately 96% of the radio channel capacity, which is the maximum achievable throughput, after discounting the overhead of RLC control messages (namely, status messages), PDCP and IP headers. As observed in Fig. 5(a), the performance of Vegas is rapidly deteriorating when the TBLER increases even though the simulation traces showed that the RLC acknowledged mode successfully delivered every transmitted TCP segment to the receiver. When the TBLER is 30%, Vegas' throughput drops to 5% of the radio channel capacity, which is 60% lower than Reno. The cause of poor TCP Vegas performance is analyzed and explained in detail in the next section.



(a)



(b)

Fig. 5. (a) Throughput versus Transport Block Error Rate; (b) TCP Congestion Window versus Time for 5% TBLER

5.2 TCP Round-Trip Time Variability

Fig. 5(b) plots the Vegas and Reno congestion window sizes in segments over the simulation duration for 5% TBLER. The poor performance of Vegas is clearly evidenced by the small window size, while Reno's congestion window grows exponentially until the maximum window size is reached, and remains at this size throughout the simulation. The maximum window size of the TCP connection is 64 segments, which also corresponds to the bandwidth-delay product. In contrast, the congestion window size of Vegas only managed to reach a maximum window size of 7 segments for a short period (appeared as peaks in the Vegas' curve of Fig. 5(b)) and then dropped to 6 segments The first peak occurred at approximately 430 s. The Vegas' congestion window exhibits instability and oscillates about the mean window size of 5 segments. This mean window size is relatively small as compared with Reno, which explains for the low throughput achieved by Vegas. The peculiar behavior of Vegas is due to the congestion avoidance which uses the TCP round-trip times to adjust the window size. The round-trip time for each TCP segment was obtained from the simulation traces for 5% TBLER. In total, there were approximately 26,000 round-trip time samples produced from the simulation traces and plotted in Fig. 6(a). The y-axis shows the round-trip time expressed in seconds for each individual segment and the x-axis shows the time in seconds when each round-trip time was recorded. The round-trip time can be expressed as the sum of the end-to-end delay of the TCP segment and the corresponding acknowledgement,

$$RTT_{TCP} = t_{seg} + t_{ack} \tag{2}$$

where t_{seg} and t_{ack} are the end-to-end delay of the TCP segment and the acknowledgement, respectively. t_{seg} and t_{ack} consist of the delay components over

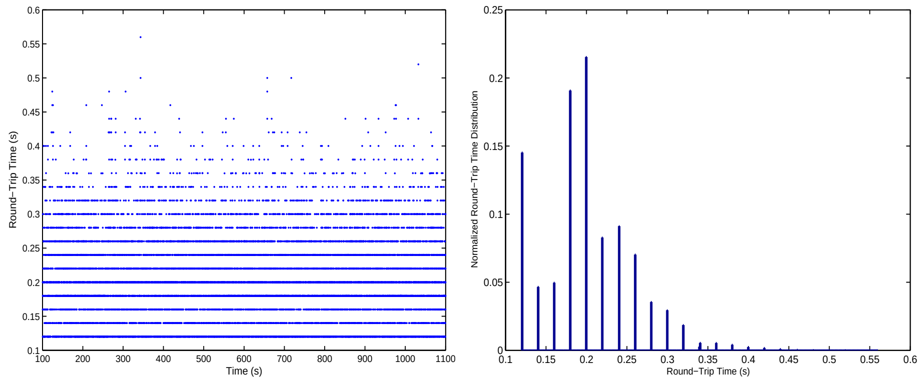


Fig. 6. (a) TCP Vegas Round-Trip Time for 5% TBLER; (b) TCP Vegas Round-Trip Time Distribution for 5% TBLER

the interfaces from U_u to G_i as shown in Fig. 1

$$t_{seg} \quad or \quad t_{ack} = t_{U_u} + t_{I_{ub}} + t_{I_{uPS}} + t_{G_n} + t_{G_i} \tag{3}$$

Each delay component in Eq. (3) is composed of the propagation delay and the transmission delay. The latter is the time required to transmit a TCP segment or an acknowledgement, that is the number of bits in a segment or acknowledgement divided by the channel bit rate in bits per second. The transmission delay over all the interfaces except U_u remains constant throughout the simulation since the TCP segment and the acknowledgement are of fixed length. The propagation delay is the time required by the signal to traverse the physical distance of the channel, which is limited by the speed of light in the tranmission medium. The propagation delay for all the interfaces except U_u is also fixed. The one-way propagation delay for each interface is marked in Fig. 3. The plot in Fig. 6(a) indicates that the TCP round-trip time exhibits relatively large variation between consecutive round-trip times with the maximum and the minimum values of 0.56 s and 0.12 s, respectively. The minimum round-trip time is the basic time to send a TCP segment and receive the corresponding acknowledgement in the absence of errors. Fig. 6(b) presents the round-trip time distribution of Fig. 6(a) in a histogram. The mean round-trip time is approximately 0.2 s.

Since the transmission delay and the propagation delay over the interfaces between G_i and I_{ub} are fixed in all the simulations, the delay component, t_{U_u} in Eq. (3), is the only contributor to the round-trip time variation. On the U_u interface, the propagation delay is negligible as compared with the transmission delay. Hence, the round-trip time variation is mainly due to the transmission delay on the U_u interface, which in turn, caused by the in-sequence delivery and retransmission strategy at the RLC layer. Without in-sequence delivery, the TCP receiver can generate spurious duplicate acknowledgements due to out of order segments. The duplicate acknowledgement triggers the fast retransmit of the TCP sender, which leads to redundant retransmission. As mentioned, the retransmission strategy of RLC is selective-repeat ARQ which is used to retransmit erroneous RLC blocks that composed a TCP segment or an acknowledgement. Note that, the uplink and downlink channels are subject to equal block error rates. Hence, the RLC blocks that composed the TCP segment or the acknowledgement might be retransmitted. The transmission delay can be expressed in terms of the number of TTI. Eq. (4) gives the transmission delay for an error-free transmission[3]

$$t_{U_u} = \left\lceil \frac{l}{C \times TTI} \right\rceil \times TTI \tag{4}$$

where l is the length of a TCP segment or an acknowledgement in bits; and C is the channel bit rate. For example, the transmisson delay of an IP packet of 552 bytes or an acknowledgement of 40 bytes, is 2 TTIs and 1 TTI, respectively. Retransmission of any RLC blocks requires additional TTIs with a minimum of 2 TTIs, i.e., one TTI for the status message to inform the sender of erroneous RLC blocks and the other for retransmitting the RLC blocks. The plot in Fig. 6(a) shows that the round-trip time values occurred at discrete levels, which is due to the fact that the transmission delay is characterized by the number of TTI. The interval between consecutive round-trip times is 20 ms which corresponds to two TTIs in the downlink or one TTI in the uplink. A large variation in round-trip time causes instability and can result in a destructive effect to Vegas. Vegas becomes unstable once the RTT_{base} in Eq. (1) is locked to the minimum round-trip time. For the 5% TBLER case, the RTT_{base} is 0.12 s. Therefore, if $windowSize$ is equal to 6 segments, and if the average round-trip time, $RTT \geq$ 0.3 s, the criterion $\Delta > \beta$ is satisfied, which linearly decreases the congestion window as observed in Fig. 5(a) For $windowSize = 5$ and RTT at the mean value of 0.2 s, the criterion, $\alpha < \Delta < \beta$, is satisfied and the congestion window is unchanged, which correlates to the mean window size of 5 segments.

A similar phenomenon was also observed for higher TBLERs. The round-trip time variation, however, is even greater as the TBLER increases, which means, the likelihood of Δ exceeding β is higher than small TBLERs. In addition, spurious timeouts were observed for TBLERs equal to or greater than 20%, which causes significant throughput degradation. Fig. 7(a) presents the Vegas and Reno's congestion window size over simulation duration for 30% TBLER. Note that, the behavior of Reno is similar to the 5% TBLER case, where the maximum window size is reached. As for Vegas, the congestion window size is relatively smaller than the 5% TBLER case. The congestion window opened up to 4 segments during slow start, and then dropped to 3 segments at around 103.6 s as depicted in Fig. 7(a), which is due to the congestion avoidance. Unlike Reno, Vegas' retransmission timeout is accurately reflecting the measured round-trip time of TCP segments. As a result, a sudden increase of the instantaneous round-trip delay beyond the sender's retransmission timeout value causes spurious timeouts. As shown in Fig. 7(a), several spurious timeouts occurred (marked by arrows) throughout the simulation, which prevents the congestion

[3] For a detailed derivation of an analytical model for TCP, see [16].

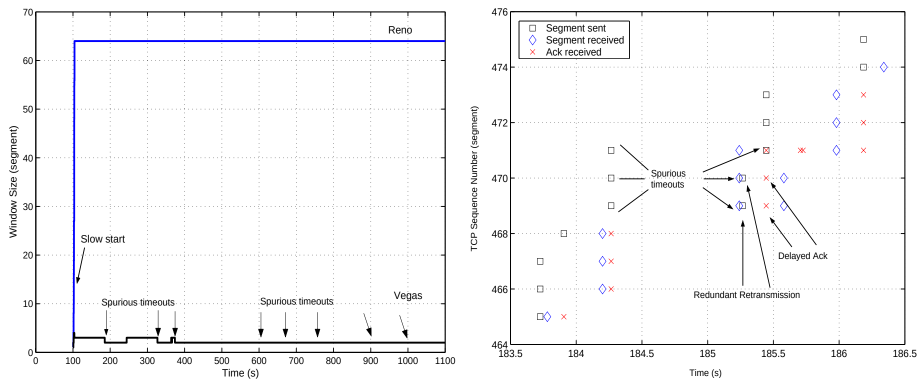


Fig. 7. (a) TCP Congestion Window versus Time for 30% TBLER; (b) TCP Vegas Trace shows Spurious Timeouts and Redundant Retransmissions for 30% TBLER

window from inflating. As a result, the congestion window size remained at 2 segments for more than fifty percent of the TCP connection time. Spurious timeouts can also cause redundant retransmissions which have contributed to the poor performance of Vegas. Fig. 7(b) shows the TCP trace for a spurious timeout event for 30% TBLER. The figure illustrates that the retransmission timeout value used by Vegas is vulnerable to the increased round-trip time and triggering unnecessary retransmissions.

In summary, Vegas would interpret an increase in round-trip delay as a sign of congestion in the network, and consequently, decrease the window size. However, this is not the desire behavior, but sender should perform the opposite.

6 Conclusion and Future Work

The paper has evaluated and compared the performance of TCP Vegas with TCP Reno over UMTS dedicated channels for bulk data transfers. Throughput was used as a performance measure. Throughput simulation results show that the performance of Vegas is worst than Reno under various transport block error rates. As a matter of fact, Vegas' throughput collapses for transport block error rates greater than 30% even though the RLC acknowledged mode successfully delivered every transmitted TCP segment. Conversely, Reno' behavior does not show any abnormality and it achieves the expected throughput over the different transport block error rates. The interaction between Vegas and the UMTS radio interface protocols, in particular, the radio link control layer was examined. The UMTS radio interface exhibits large and highly variable delay, which in turn, resulted in fluctuating TCP round-trip times. Vegas differs from Reno in the sense that it uses TCP round-trip times to detect congestion. Hence, the round-trip time variation is perceived as a sign of congestion and Vegas shrinks its window size, which has a detrimental effect on the performance. Moreover, a sudden increase in instantaneous round-trip time can trigger spurious timeouts at the sender running Vegas which performs unnecessary retransmissions. Spurious timeouts occur at high transport block error rates. The UMTS radio interface characteristics pose performance issues to Vegas due to the new congestion avoidance and retransmission features. On the other hand, Reno is less intelligent but its congestion avoidance and retransmission mechanisms are more robust as compared with Vegas.

In the near future, we anticipated that the use of TCP Vegas will be widely spread since it has been supported by some operating system, e.g. [17], in addition to Reno. Consequently, the poor performance of Vegas in wireless networks with large round-trip time variation such as UMTS needs to be improved. The proposed solutions, in particular, the split-connection and proxy approaches can be adapted for Vegas. Further work is required on adapting and to investigate the performance of Vegas using these two solutions.

References

1. 3GPP: available from `http://www.3gpp.org`
2. Kaaranen, H., et al.: UMTS Networks: Architecture, Mobility and Services, New York, John Wiley & Sons (2001)
3. Steven, W.: TCP/IP Illustrated,Vol. 1, Reading, Addison Wesley (1994)
4. Brakmo, L.S., Petersen, L.L.: TCP Vegas: End to End Congestion Avoidance on a Global Internet. IEEE Journal on Selected Areas in Communications, vol. 13, no. 8 (1995)
5. Caceres, R., Oftode, L.: Improving the Performance of Reliable Transport Protocols in Mobile Computing Environment. IEEE Journal on Selected Areas in Communications, vol. 13, no. 5 (1995)
6. Balakrishnan, H., et al.: Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks. ACM Wireless Networks, vol. 1, no. 4 (1995)
7. Chaskar, N.M., et al.: TCP over Wireless with Link Level Error Control: Analysis and Design Methology. IEEE/ACM Transaction on Networking, vol. 7, no. 5 (1999)
8. Bai, Y., et al.: Interactions of TCP and Radio Link ARQ Protocol. Proceedings of the IEEE Vehicular Technology Conference (VTC'99 - Fall), Amsterdam, The Netherlands (1999)
9. Parsa, C., Garcia-Luna-Aceves, J.J.: Improving TCP Performance over Wireless Networks at the Link Layer. Mobile Networks and Applications, vol. 5, no. 1 (2000)
10. Ayanoglu, E., et al.: AIRMAIL: A Link-layer Protocol for Wireless Networks. Wireless Networks, vol. 1, no. 1 (1995)
11. Bakre, A., Badrinath, B.R.: Implementation and Performance Evaluation of Indirect-TCP. IEEE Transaction on Computers, vol. 46, no. 3 (1997)
12. 3GPP: Network Architecture. TS 23.002 (2000)
13. Fall, K., Varadhan, K.: The *ns* Manual. `http://www.isi.edu/nsnam/ns/ns-documentation.html`
14. SEACORN Home page: `http://seacorn.ptinovacao.pt/`
15. 3GPP: Radio Link Control (RLC) Protocol Specification. TS 25.322 (2002)
16. Peisa, J., Meyer, M.: Analytical Model for TCP File Transfer over UMTS. 3G Wireless 2001, San Francisco, CA (2001)
17. Cardwell, N.: A TCP Vegas Implementation for Linux. `http://flophouse.com/~neal/uw/linux-vegas`