TCP Congestion Control over 3G Communication Systems: an Experimental Evaluation of New Reno, BIC and Westwood+[⋆]

Luca De Cicco and Saverio Mascolo
{ldecicco,mascolo}@poliba.it

Dipartimento di Elettrotecnica ed Elettronica, Politenico di Bari, Via Re David 200, Italy

Abstract. One of TCP's key tasks is to react and avoid network congestion episodes which normally arise in packet switched networks. A wide literature is available concerning the behaviour of congestion control algorithms in many different scenarios and several congestion control algorithms have been proposed in order to improve performances in specific scenarios. In this paper we focus on the UMTS wireless scenario and we report a campaign of measurements that involved around 3000 flows and more than 40 hours of measurements using three different TCP stacks: TCP NewReno, which is the congestion control algorithm standardized by IETF, TCP BIC which is the default congestion control algorithm adopted by the Linux operating system, and TCP Westwood+ also available in the Linux kernel. The experimental evaluation has been carried out by accessing the public Internet using an UMTS card. Measurements of goodputs, RTTs over time, packet loss ratios, number of timeouts and Jain Fairness Indices are reported through cumulative distribution functions. Moreover, the efficiency of each TCP version in transferring files has been evaluated by varying the file size in the range from $50\,KB$ up to $500\,KB$. The cumulative distribution functions reported in the paper show interesting results: 1) a single downlink flow is far from saturating the channel bandwidth; 2) considered TCP stacks provide similar results; 3) $90th$ ($50th$) percentile of the goodput of a single downlink flow is less or equal then $230\,kbps$ ($120\,kbps$) compared to a nominal $384\,kbps$ UMTS downlink channel.

TCP, congestion control, 3G, UMTS

1 Introduction

Since 2001, the year that the first commercial Universal Mobile Telecommunications System (UMTS) network was deployed by *NTT DoCoMo* in Japan,

[⋆] We would like to thank Fabio Ricciato for allowing us to use a server at FTW. This work has been partially supported by the MIUR-PRIN project no. 2005093971 "FAMOUS Fluid Analytical Models Of aUtonomic Systems" and by Financial Tradeware S.r.l..

many telecom operators have launched UMTS access to subscribers. As many 3G networks are emerging, it is important to evaluate how different Transmission Control Protocol (TCP) congestion control mechanisms behave in such networks. The UMTS network provides wide area Internet wireless access with downlink speeds up to 384 $kbps$ and round trip times in the order of 300 ms thus providing a viable solution for multimedia and for Voice over IP (VoIP) applications.

It is known that efficiency of TCP as a transport protocol degrades when lossy links are present in the routing path [6][4], such as in the case of wireless links. For this reason, the UMTS link layer implements the Radio Link Control (RLC) protocol that masks the lossy channel to upper layers through retransmissions. In this way, in-order packet delivery and loss probability less than 1% are provided [1]. The reliability of the link layer comes at the cost of an highly variable segments delay as seen at the transport layer when frames are retransmitted at the link layer, thus possibly leading to spurious timeouts [10]. The impact of spurious timeouts has been studied extensively so far and it has been often considered as one of the major causes of TCP throughput degradation. However, against this common belief authors in [5] have found that, in the case of a well-designed UMTS network and in the static scenario, the number of spurious timeouts is very low, thus having negligible impact on TCP throughput.

Another issue raised by the UMTS link layer is the variability of the available bandwidth that is caused by the channel state scheduling. In [1] authors address both rate and delay variabilities, showing the negative effects of delay variability on throughput achieved by TCP.

In this paper we present the results obtained through an extensive campaign of measurements obtained over a live UMTS network. Both downlink and uplink measurements of goodputs, round trip times (RTT), queuing times, packet loss ratios, number of timeouts and Jain Fairness Indices (JFI) have been collected. We have considered three different TCP stacks that are available in the Linux kernel: TCP NewReno [12], which is the congestion control algorithm standardized by the Internet Engineering Task Force (IETF), TCP BIC [16], which is the default congestion control algorithm adopted by the Linux operating system, and TCP Westwood+, which has been proposed in [13,11] and is also available in the Linux kernel.

The rest of the paper is organized as follows: in Section 2 we summarize the prior work on live UMTS measurements and we briefly describe the considered congestion control algorithms; in Section 3 we describe the considered experimental testbed; Section 4 reports experimental results whereas a discussion of the results is reported in Section 5. Finally, Section 6, draws the conclusions.

2 Related Work

2.1 Live UMTS network performance evaluation

The academic literature contains a plethora of simulation studies about TCP performances over UMTS and GPRS, but very few papers have addressed the performance evaluation of a live UMTS network.

In [9] authors report results obtained by an experimental investigation carried out in an early deployment of two 3G networks in near-ideal conditions, i.e. no other user was accessing the network. In the paper it is noticed that the UMTS link is affected by very few spurious timeouts and that the employment of the Eifel algorithm did not provide any throughput improvement.

In [7] authors report goodput measurements obtained by accessing the public 3G/UMTS network to transfer files with different size; the paper focuses only on the downlink and does not report any data regarding RTT variability, typical number of timeouts and fairness indices.

Authors of [8] provide an IP and TCP level measurement of the UMTS downlink and uplink in both static and mobile scenario. The paper focuses on finding the optimal settings for the MSS and the initial receiver window; throughput measurements are given for a limited number of considered scenarios.

2.2 TCP Congestion control algorithms

One of the most important tasks that the TCP addresses is regulating the sending rate in order to avoid the network congestion. In [14] Van Jacobson has proposed a solution to the network congestion control problem that mainly consists of two distinct phases: a probing phase and a decreasing phase (the well-known Additive Increase Multiplicative Decrease - AIMD - paradigm [2]). During the probing phase the link capacity is probed using an exponential growth law which is called slow start, or a linear growth law that is called congestion avoidance. The congestion control algorithm switches from the probing phase to the decrease phase when a three duplicate acknowledgment (3DUPACK) or a timeout is experienced indicating that a congestion event has taken place. During the decreasing phase the congestion window *cwnd* is multiplicatively decreased in reply to the congestion episode.

During the congestion avoidance phase of TCP NewReno the congestion window is increased by one packet for each RTT, whereas in the slow start phase the *cwnd* is doubled each RTT. The congestion window is halved after a congestion episode, whereas when a timeout is experienced the *cwnd* is set to 1 segment and the slow start phase takes place.

TCP Binary Increase Congestion Control (BIC) [16] is made of two parts: the binary search increase phase and the additive increase phase. In the binary search phase the congestion window setting is performed as a binary search problem. After a packet loss, the congestion window is reduced by a constant factor b, $cwnd_{max}$ is set to the window size before the lost and $cwnd_{min}$ is set to the value of congestion window after the loss ($cwnd_{min} = b \cdot cwnd_{max}$). If the difference between the congestion window middle point $(cwnd_{max} + cwnd_{min})/2$ and the minimum congestion window is lower than a threshold S_{max} the protocol starts a binary search algorithm increasing the congestion window to the middle point, otherwise the protocol enters the “linear increase” phase and increments the congestion window by one for each received ACK. If BIC does not get a loss indication at this window size, then the actual window size becomes the new minimum window; otherwise, if it gets a packet loss, the actual window size becomes the new maximum. The process goes on until the window increment becomes lower than the S_{min} threshold and the congestion window is set to $cwnd_{max}$. If the window grows more than $cwnd_{max}$, the protocol enters into a new phase (max probing) that is specular to the previous phase; that is, it uses the inverse of the binary search phase first and then the additive increase.

TCP Westwood+ [11,13] is a sender side modification of TCP NewReno in which the multiplicative decreasing phase is replaced with an adaptive decreasing phase. In particular, after a congestion episode *cwnd* is set such that the bandwidth which is available at the time of congestion is exactly matched. The available bandwidth is estimated by counting and averaging the stream of ACK packets. In particular, when three DUPACKs are received, the congestion window *cwnd* is set equal to the estimated bandwidth (BWE) times the minimum measured round trip time (RTT_{min}). After a timeout the slow start threshold is set equal to $BWE \cdot RTT_{min}$ and the *cwnd* is set to one.

3 Experimental Testbed

In order to carry out our experiments we have set up a machine at the FTW research center in Vienna, installing a Linux Kernel with Web100 support. The TCP flows have been generated and received using a modified version of `iperf` [15] which uses the `libnetmeas` [3] library that we have developed in order to automatically get instantaneous values of internal kernel variables such as cwnd, RTT, ssthresh, timeouts.

We have found out that the telecom operator uses a transparent proxy that probably implement some sort of Split Stream solution, so that when the user downloads the file for the first time, the file is cached on a proxy that is located in the telecom operator network. This way the second download provides better results, since the path results shorter than that of the first download. For this reason we have not used an HTTP server as the sender and an HTTP client (such as `wget`) for the receiver.

The scenarios and the testbed that we have used in this investigation are aimed at characterizing both the uplink and downlink UMTS channels in the most common user scenarios. The user equipment (UE) is a Nokia 6630 mobile phone connected to a laptop via USB 2.0 that was located at the C3Lab, Politecnico di Bari (Italy) accessing the public UMTS network using a commercial card provided by a local mobile operator (Figure 1).

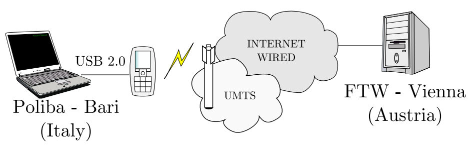


Fig. 1. Experimental Testbed

The UE was static and was accessing the UMTS network in an indoor environment so that handovers could not occur during measurements. The nominal value declared by the telecom operator for the downlink (uplink) channel is $384\,kbps$ ($64\,kbps$). Based on the results obtained in [8] we have fixed the maximum segment size to 1500 bytes. Moreover we have set the initial receiver advertisement window to the default value of $64\,Kb$ which is well above the bandwidth delay product so that we are sure that the bottleneck is located in the UMTS network.

For each connection we have collected a very rich set of measurement including goodputs, RTTs, queuing times, number of timeouts, retransmission ratios. We have evaluated the Jain Fairness Index as follows [2]:

$$JFI = \frac{\left(\sum_{i=1}^{N} x_i\right)^2}{N\sum_{i=1}^{N} x_i^2}$$

where x_i is the mean goodput obtained by the i-th flow accessing the downlink or the uplink.

We have evaluated TCP congestion control algorithms in the following scenarios: i) one flow over the UMTS downlink or uplink; ii) two or four homogeneous flows sharing the UMTS downlink or uplink; iii) short file transfers of 50KB, 100KB, 200KB, 500KB, files on both UMTS downlink and uplink.

For each scenario we have run experiments using the Linux Kernel implementation of TCP Bic, TCP NewReno and TCP Westwood+[11,13]. All tests, except those for the short file transfer scenario, lasted approximately 100 seconds each, thus counting for more than 40 hours of active measurements of downlink and uplink UMTS channel. In order to perform a fair comparison, we have run tests by rotating TCP congestion control algorithms and scenarios and repeating tests in different days and in different hours of the day.

4 Experimental Results

We have performed an extensive experimental evaluation of the three congestion control algorithms by collecting measurements of around three thousand flows for more than 40 hours of active measurements. In this section we report the results collected for both downlink and uplink.

4.1 Goodput, link utilization and fairness

The case of downlink flows
Figures 2 (a), (b) and (c) report the cumulative distribution functions for the results obtained for the cases of one, two and four flows respectively, sharing the UMTS downlink, whereas mean goodputs and standard deviations are summarized in Table 1.

It is noteworthy that the three congestion control protocols provide similar results in all the cases, the only remarkable difference being the case of the single flow (Figure 2 (a)) where TCP NewReno obtain 12% and 8% goodput less than TCP Westwood+ and TCP Bic. The median values in the case of one, two and four flows are in the ranges $[101,121]$, $[76,87]$ and $[71,78]$ $kbps$ respectively, whereas the upper $10th$ percentile experiences bandwidth in the ranges $[220,249]$, $[128,146]$ and $[121,139] kbps$, respectively.

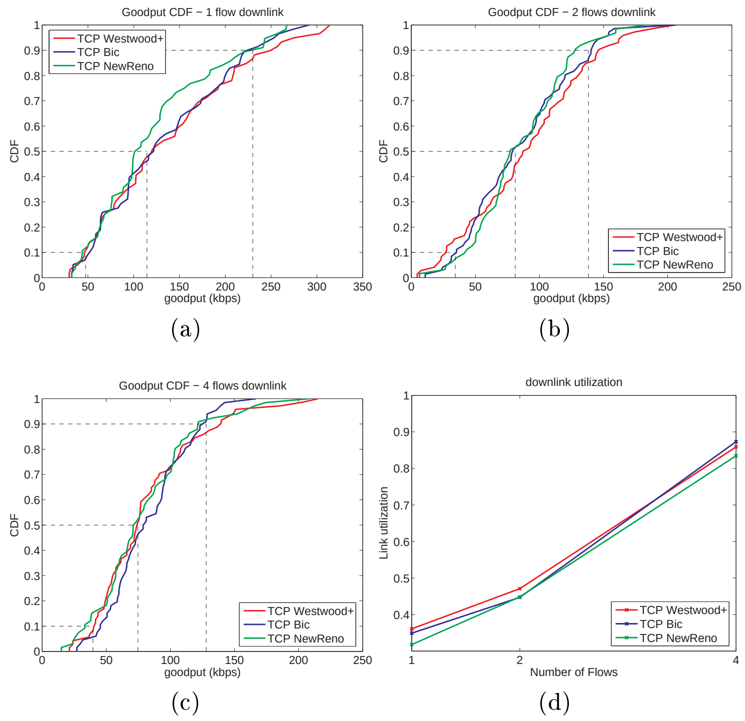


Fig. 2. Cumulative distribution function of the goodput measured in the case of 1 flow (a), 2 flows (b), 4 flows (c) sharing the UMTS downlink; bandwidth utilization (d)

	New Reno			BIC			Westwood+		
#Flows	$E[x]$	$\sigma(x)$	Ch. Utiliz.	$E[x]$	$\sigma(x)$	Ch. Utiliz.	$E[x]$	$\sigma(x)$	Ch. Utiliz.
1	122.03	65.38	31.8%	133.98	69.41	34.9%	138.58	77.10	36.1%
2	86.08	36.20	44.8%	85.82	40.51	44.7%	90.41	46.12	47.1%
4	80.13	38.53	83.5%	83.83	30.68	87.3%	82.46	40.60	85.9%

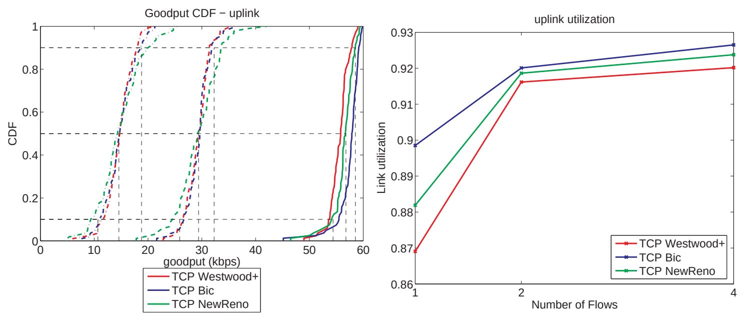
Table 1. Average and standard deviation values (in kbps) of goodput for the UMTS downlink channel

Figure 2 (d) shows the downlink utilization that is computed by averaging goodputs of all the experiments performed for each considered protocol when the number of flows sharing the link are one, two or four. It is worth noting that all the three tested TCP variants provide less than 40% of bandwidth utilization in the single flow case. By increasing the number of flows sharing the bottleneck the utilization reaches about 90% in the four flows case. We will return later on this topic in Section 5 where we will discuss possible reasons of the very low link utilization in the one flow case. Finally, it is worth to notice that goodputs reported by Catalan et al. [8] in the single flow case averaged over three experiments are similar to our results.

Moreover, concerning the fairness indices we have found that the considered TCP variants provide similar average values of the JFI both in the two and four flows cases. We have obtained values around 0.94 for the two flows case and around 0.86 for the four flows case.

The case of uplink flows

The evaluation of the UMTS uplink channel have led to the results depicted in Figure 3 (a). Also in the uplink scenarios the goodputs obtained by the considered TCP congestion control algorithms are very similar, TCP Bic performing slightly better in the single flow case.



(a) (b)

Fig. 3. Goodput cumulative distribution function for the uplink scenario (a): 1 flow in solid lines , 2 flows dashed lines and 4 flows in dot-dashed lines. Link utilization (b)

In the case of uplink flows Table 2 enlights that the values of the standard deviation of goodput are very low for all the considered TCP stacks and in all cases. The median values in the case of one, two and four flows are respectively in the ranges $[53, 55]$, $[24, 26]$ and $[14, 15]$ $kbps$, whereas the upper $10th$ percentile experiences bandwidths that are respectively in the ranges $[57, 59]$, $[31, 33]$ and $[17, 20] kbps$.

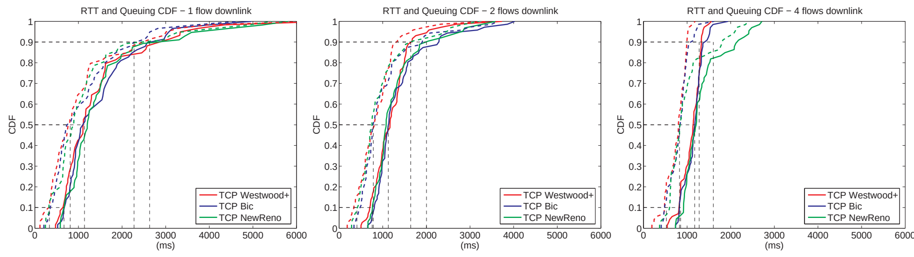
Differently from the case of downlink flows, the protocols that we have tested have provided nearly full uplink utilization even in the single flow case as it is shown in Figure 3 (d). In the uplink scenario Jain Fairness Index is very high for the considered TCP stacks, being around 0.99 for the two flows case and around 0.95 for the four flows case.

	New Reno			BIC			Westwood+		
#Flows	$E[x]$	$\sigma(x)$	Ch. Utiliz.	$E[x]$	$\sigma(x)$	Ch. Utiliz.	$E[x]$	$\sigma(x)$	Ch. Utiliz.
1	56.44	2.09	88.2%	57.51	2.11	89.9%	55.62	1.74	86.9%
2	29.40	3.95	91.9%	29.44	2.31	92.0%	29.32	2.19	91.6%
4	14.78	4.08	92.4%	14.82	2.81	92.6%	14.72	2.51	92.02%

Table 2. Average and standard deviation values (in kbps) of goodput for the UMTS uplink channel

4.2 RTT and queuing time

The case of downlink flows.
Figures 4 (a), (b) and (c) show the cumulative distribution functions of the round trip times (RTT) and the queuing times, hereinafter named t_q[1], in the case of the downlink channel, while Table 3 collects mean values, RTT standard deviations and average queuing times for the considered scenarios.



(a) (b) (c)

Fig. 4. RTT (solid lines) and queuing time (dashed lines) cumulative distribution function in the case of 1 flow (a), 2 flows (b) and 4 flows (c) sharing the UMTS downlink.

	New Reno			BIC			Westwood+		
#Flows	$E[RTT]$	$\sigma(RTT)$	$E[t_q]$	$E[RTT]$	$\sigma(RTT)$	$E[t_q]$	$E[RTT]$	$\sigma(RTT)$	$E[t_q]$
1	1550	1096	1248	1457	897	1137	1469	1110	1125
2	1297	624	953	1369	691	1024	1219	508	873
4	1338	488	995	1159	230	825	1102	221	765

Table 3. Average and standard deviation values (in ms) of RTT for the UMTS downlink channel. Average values (in ms) of queuing.

Even though the minimum RTT experienced in our evaluation is around $300\,ms$, the measured queuing times are very high and they do not depend on the number of flows, thus suggesting that in the single flow case we have

[1] The round trip time is defined as sum of a propagation time, which can be evaluated as the minimum RTT, and the queuing time t_q.

an excessive queuing. TCP NewReno exhibits a slightly inflated value of RTT quantified in $100\,ms$ in the one flow case, due to larger queuing time. In all evaluated scenarios, TCP Westwood+ provides less queuing time with respect to the other two algorithms. This result is coherent with the Westwood+ unique feature of clearing all queuing after a congestion.

The case of uplink flows

Figures 5 (a), (b) and (c) show the round trip time and the queuing time cumulative distribution functions in the case of uplink flows while Table 4 collects the values of mean RTT, RTT standard deviation and average queuing time. Similarly to the downlink scenario, TCP Westwood+ exhibits less queuing time except the case of four flows where TCP NewReno provides slightly less queuing time. Differently from the case of downlink flows, the RTT and queuing time tends to increase with the number of concurrent flows.

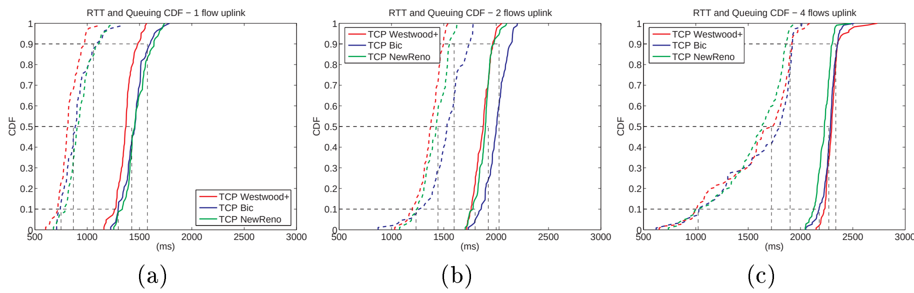


Fig. 5. RTT (solid lines) and queuing time (dashed lines) cumulative distribution function in the case of 1 flow (a), 2 flows (b) and 4 flows (c) sharing the UMTS uplink.

	New Reno			BIC			Westwood+		
#Flows	$E[RTT]$	$\sigma(RTT)$	$E[t_q]$	$E[RTT]$	$\sigma(RTT)$	$E[t_q]$	$E[RTT]$	$\sigma(RTT)$	$E[t_q]$
1	1471	113.2	927	1452	113.2	904	1368	87.1	828
2	1887	80.2	1416	1998	104.3	1521	1869	79.5	1364
4	2222	72.7	1556	2276	70.7	1621	2301	78.7	1587

Table 4. Average and standard deviation values (in ms) of RTT for the UMTS uplink channel. Average values (in ms) of queuing.

4.3 Timeouts and packet retransmission percentage

Table 5 summarizes the average number of timeouts obtained on the UMTS downlink and uplink in the considered scenarios. The results obtained in the downlink and uplink scenarios are very different and can be the source of the very different link utilization that we have reported in Section 4.1. In the downlink case the flows suffer around five timeouts throughout the $100\,s$ duration of the connection regardless of the number of flows and the congestion control algorithm used. On the other hand, in the uplink flows case, the number of timeouts is negligible.

	New Reno		BIC		Westwood+	
#Flows	Downlink	Uplink	Downlink	Uplink	Downlink	Uplink
1	5.84	0.15	4.82	0.13	5.27	0.15
2	6.44	0.04	5.43	0.03	6.20	0.06
4	5.35	0.45	5.24	0.35	5.80	0.43

Table 5. Average number of timeouts for downlink and uplink (the duration of the connection is $100\,s$)

Due to space limitation we can't show the cumulative distribution function of packet loss ratios, but we report the average value and the standard deviation in the Table 6. By considering the values reported in the table we can observe that there is no remarkable difference in the considered TCP congestion control algorithms. In the case of downlink flows, the retransmission percentage increases with the number of flows and it is below 11%, whereas in the case of uplink flows the fraction of retransmitted packets is less than 1%.

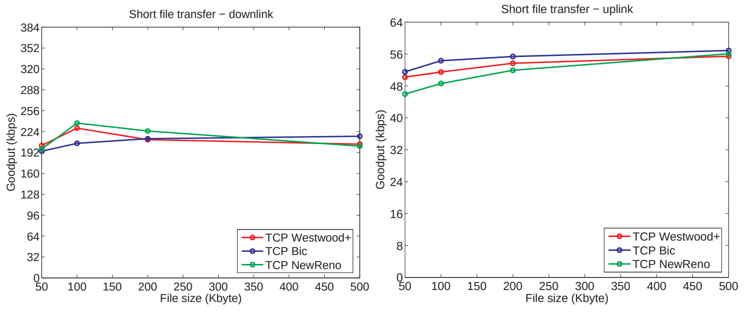
	New Reno				BIC				Westwood+			
	Downlink		Uplink		Downlink		Uplink		Downlink		Uplink	
#Flows	$E[p]$	$\sigma(p)$	$E[p]$	$\sigma(p)$	$E[p]$	$\sigma(p)$	$E[p]$	$\sigma(p)$	$E[p]$	$\sigma(p)$	$E[p]$	$\sigma(p)$
1	7.13	4.89	0.04	0.12	6.75	2.97	0.03	0.10	7.79	6.05	0.03	0.10
2	7.96	5.28	0.02	0.12	7.97	4.41	0.02	0.12	10.12	6.86	0.02	0.10
4	10.95	5.85	0.40	0.98	10.05	3.77	0.23	0.52	11.69	4.57	0.27	0.55

Table 6. Average retransmission percentage (%) and standard deviation in the case of uplink and downlink.

4.4 Goodput versus file size

In this section we investigate the impact of the file size on the goodput of TCP. We have collected goodput measurements for file size in the range from $50\,KB$ to $500\,KB$, in order to find out if the slow start phase degrades goodput in the case of small size file transfers. Figures 6 (a) and (b) show results respectively for the downlink and uplink cases. In the case of downlink flows, all considered TCP variants provide similar results and the goodput is essentially constant when the file size increases, showing a maximum at $100\,KB$ in the case of TCP NewReno and TCP Westwood+. Also in the case of uplink flows the goodput is constant, slightly increasing of $10\,kbps$ when the file size grows from $50\,KB$

to 500 KB. Thus, we can conclude that the slow start phase does not cause remarkable effects on the goodput in both the downlink and uplink channels.



(a) (b)

Fig. 6. Goodput vs file size for (a) downlink and (b) uplink channel

5 Discussion of results

In the previous sections we have reported the results of our extensive UMTS evaluation and we have found that the performance of the considered TCP congestion control algorithms are comparable in all the scenarios we have evaluated.

Furthermore, we have found that in the case of uplink flows there are no remarkable issues, since all TCPs provide satisfactory channel utilization and exhibit very low number of timeouts. On the contrary, Section 4.1 has shown that in the case of the downlink scenario, all TCPs provided a very low utilization of UMTS links in the single flow case. To gain an insight into the reason of the poor link utilization let's consider the number of timeouts in the case of the downlink flows as summarized in Table 5. It can be seen that the number of timeouts is roughly constant and does not depend on the number of flows. This observation suggests that timeouts are not due to congestion, otherwise we should expect to measure more timeouts in the case of multiple flows over the downlink. Thus, we argue that the reason of the poor downlink utilization in the case of single flow is mostly due to the high number of timeouts that impose an upper bound to the achievable goodput. In fact, the inflated RTT values due to large buffering, which has been already discussed in Section 4.2, make the retransmission timeout longer, thus implying a very long time spent in recovering timeout events [10]. Using trivial arguments, the average time t_{out} spent resolving timeouts is the product of the average number of timeouts N_{tout} that affect the connection and the average retransmission timeout value T_0:

$$E[t_{tout}] = E[N_{tout}] \cdot E[T_0] \cong 35\,s$$

which in our case, by considering that the connections last 100 s, results in 35% of the connection time. It is worth to notice that this value matches the link utilization shown in Section 4.1 (see Figure 2 (d)).

6 Conclusions

In this paper an extensive TCP performance evaluation over a live UMTS network by measuring both downlink and uplink performance indices for three different TCP congestion control algorithms is reported. The main findings can be summarized as follows: (i) the considered TCP congestion control algorithms performed similarly both in downlink and uplink scenarios; (ii) the UMTS uplink channel did not exhibit any remarkable issues, providing good channel utilization and very low number of timeouts and packet retransmissions; (iii) a very high number of timeouts has been observed in our measurements in the case of downlink channel that does not seem to be caused by congestion; (iv) the UMTS downlink channel utilization is poor in the single flow case because of the joint effect of the very high number of timeouts and the inflated RTT due to queuing.

References

1. M.C. Chan and R. Ramjee. TCP/IP Performance over 3G Wireless Links with Rate and Delay Variation. *Wireless Networks*, 11(1):81–97, 2005.
2. D.M. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17(1):1–14, 1989.
3. L. De Cicco. Libnetmeas, 2006.
4. D.A. Eckhardt and P. Steenkiste. Improving wireless lan performance via adaptive local error control. In *ICNP*, pages 327–338, 1998.
5. F. Vacirca et al. An algorithm to detect TCP spurious timeouts and its application to operational UMTS/GPRS networks. *Computer Networks*, 50(16):2981–3001, 2006.
6. H. Balakrishnan et al. A comparison of mechanisms for improving TCP performance overwireless links. *Networking, IEEE/ACM Trans. on*, 5(6):756–769, 1997.
7. K. Pentikousis et al. Active goodput measurements from a public 3G/UMTS network. *Communications Letters, IEEE*, 9(9):802–804, 2005.
8. M. Catalan et al. TCP/IP analysis and optimization over a precommercial live UMTS network. *Proc. IEEE WCNC'05*, 3, 2005.
9. M. Kohlwes et al. Measurements of TCP performance over UMTS networks in near-ideal conditions. *Proc. VTC 2005-Spring*, 2005.
10. R. Ludwig et al. Multi-layer tracing of TCP over a reliable wireless link. *Proc. ACM SIGMETRICS 1999*, pages 144–154, 1999.
11. S. Mascolo et al. TCP westwood: Bandwidth estimation for enhanced transport over wireless links. *Proc. ACM MOBICOM*, pages 287–297, 2001.
12. S. Floyd and T. Henderson. RFC2582: The NewReno Modification to TCP's Fast Recovery Algorithm. *Internet RFCs*, 1999.
13. L.A. Grieco and S. Mascolo. Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control. *ACM SIGCOMM Computer Communication Review*, 34(2):25–38, 2004.
14. V. Jacobson. Congestion avoidance and control. In *ACM SIGCOMM '88*, pages 314–329, Stanford, CA, August 1988.
15. A. Tirumala and J. Ferguson. Iperf, 2001.
16. L. Xu, K. Harfoush, and I. Rhee. Binary increase congestion control (BIC) for fast long-distance networks. In *Proc. INFOCOM 2004*, pages 2514–2524, 2004.