TCP Congestion Control over HSDPA: an Experimental Evaluation

Luca De Cicco and Saverio Mascolo

Abstract— **In this paper, we focus on the experimental evaluation of TCP over the High Speed Downlink Packet Access (HSDPA), an upgrade of UMTS that is getting worldwide deployment. Today, this is particularly important in view of the "liberalization" brought in by the Linux OS which offers several variants of TCP congestion control. In particular, we consider four TCP variants: 1) TCP NewReno, which is the only congestion control standardized by the IETF; 2) TCP BIC, that was, and 3) TCP Cubic that is the default algorithm in the Linux OS; 4) Westwood+ TCP that has been shown to be particularly effective over wireless links. Main results are that all the TCP variants provide comparable goodputs but with significant larger round trip times and number of retransmissions and timeouts in the case of TCP BIC/Cubic, which is a consequence of their more aggressive probing phases. On the other hand, TCP Westwood+ provides the shortest round trip delays, which is an effect of its unique way of setting control windows after congestion episode based on bandwidth measurements.**

I. Introduction

Wireless high-speed Internet is spreading worldwide thanks to the development of wireless technologies such as IEEE 802.11 for local access and 3G-4G for large area coverage. In a recent report published by Cisco it is stated that mobile traffic is doubling for the fourth year in a row and it is projected that more than 100 millions of smartphones will consume more than one gigabyte of traffic per month [4].

High Speed Downlink Packet Access (HSDPA) is an upgrade of UMTS that is getting worldwide deployment even in countries where the CDMA-EVDO networks had the early lead on performance. Today, HSDPA is present in 128 countries distributed over all the continents, with the most advanced deployment in Europe[1].

Currently available HSDPA commercial cards provide downlink peak rate of several Mbps, which is more than one order of magnitude improvement with respect to the 100kbps offered by GSM EDGE few years ago [19].

At the beginning of wireless access to the Internet, the Transmission Control Protocol (TCP) experienced very low throughput over wireless links due to the fact that losses due to unreliable wireless links were interpreted as due to congestion [2]. In [6] it has been shown that this problem can be overcome by making the wireless link reliable through link layer retransmissions. This is today well-known and implemented at link layer of different technologies such as 3G-4G systems through Automatic Repeat reQuest (ARQ) protocols [6], which guarantee error-free segment delivery to the transport layer. The use of ARQ mechanisms masks link layer losses at the expenses of increased transmission delays.

Regarding the issue of improving TCP performance over wireless links, a large amount of literature has been published which proposes to modify the link layer, the transport layer or both using a cross-layer approach [2]. Variants that have been proposed to improve the performance of the TCP over wireless networks include TCP Westwood+ [9] and TCP Veno [8].

Today, the Linux OS offers the choice of as many as twelve TCP congestion control algorithms of which TCP Cubic is selected by default. If this can be viewed as a "liberalization" with respect to the "old" BSD TCP style that used to offer only the TCP with the enhancements standardized by the IETF [1], it poses questions on the stability and efficiency from both the point of view of the users and the network.

If a large body of literature is available concerning the performance evaluation of congestion control variants in high-speed networks [10], the same cannot be said regarding the performance evaluation over new cellular networks, in spite of the fact that more than 300 million users are accessing the Internet using broadband cellular networks such as WCDMA/UMTS [19].

In this work we evaluate the TCP performance over HSDPA, an optimization of the UMTS radio interface, which can provide downlink throughputs up to 14 Mbps and round trip times (RTT) in the order of 100 ms [19].

The purpose of this work is twofold: on one hand we aim at evaluating how TCP performs on 3.5G mobile networks; on the other hand, we provide a comparison of relevant congestion control protocols over such networks. We have made extensive experimental measurement over downlink channel in static conditions of the User Equipment (UE). We focus on a static scenario in order to be able to provide and unbiased comparison among the considered TCP variants.

We have considered four TCP variants: TCP NewReno, which is the only TCP congestion control standardized by IETF, TCP BIC and TCP Cubic, which have been selected as default congestion control algorithms in the Linux OS, and TCP Westwood+ that is known to be particularly efficient over wireless networks [14].

II. Background and related work

A. TCP congestion control algorithms

1) TCP NewReno : The TCP congestion control [11] is made of a *probing phase* and a *decreasing phase*, the well-known Additive Increase and Multiplicative Decrease

[1]http://www.gsmworld.com/our-work/mobile_broadband/networks.aspx

(AIMD) phases introduced by Jain [3]. Congestion window (*cwnd*) and slow-start threshold (*ssthresh*) are the two variables employed by the TCP to implement the AIMD paradigm. In particular, *cwnd* is the number of outstanding packets, whereas *ssthresh* is a threshold that determines two different laws for increasing the *cwnd*: 1) an exponential growth, i.e. the *slow-start phase*, in which the *cwnd* is increased by one packet every ACK reception to quickly probe for extra available bandwidth and which lasts until *cwnd* reaches *ssthresh*; 2) a linear growth when $cwnd \geq ssthresh$, i.e. the *congestion avoidance* phase, during which *cwnd* is increased by $1/cwnd$ packets on ACK reception.

The probing phase lasts until a congestion episode is detected by TCP in the form of 3 duplicate acknowledgments (3DUPACK) or timeout events. Following a 3DUPACK episode, TCP NewReno [7] triggers the multiplicative decrease phase and the *cwnd* is halved, whereas when a timeout occurs *cwnd* is set to one segment. The algorithm can be generalized as follows:

1) On ACK: $cwnd \leftarrow cwnd + a$
2) On 3DUPACK:

$$cwnd \leftarrow b \cdot cwnd \quad (1)$$

$$ssthresh \leftarrow cwnd \quad (2)$$

3) On timeout: $cwnd \leftarrow 1;\ ssthresh \leftarrow b \cdot cwnd$

In the case of TCP NewReno a is equal to 1, when in slow-start phase, or to $1/cwnd$ when in congestion avoidance, and b is equal to 0.5.

2) TCP Westwood+ : TCP Westwood+ [9] is a sender-side modification of TCP NewReno that employs an estimate of the available bandwidth BWE obtained by counting and averaging the stream of returning ACKs to properly reduce the congestion window when congestion occurs. In particular, when a 3DUPACK event occurs, TCP Westwood+ sets the *cwnd* equal to the available bandwidth BWE times the minimum measured round trip time RTT_{min}, which is equivalent to set $b = BWE \cdot RTT_{min}/cwnd$ in (1). When a timeout occurs, $ssthresh$ is set to $BWE \cdot RTT_{min}$ and $cwnd$ is set equal to one segment.

The unique feature of TCP Westwood+ is that the setting of *cwnd* in response to congestion is able to clear out the bottleneck queue, thus increasing statistical multiplexing and fairness [9].

3) TCP BIC : TCP Binary Increase Congestion Control (BIC) [20] consists of two phases: the binary search increase and the additive increase. In the binary search phase the setting of $cwnd$ is performed as a binary search problem. After a packet loss, $cwnd$ is reduced by a constant multiplicative factor b as in (1), $cwnd_{max}$ is set to the *cwnd* size before the loss event and $cwnd_{min}$ is set to the value of $cwnd$ after the multiplicative decrease phase ($cwnd_{min} = b \cdot cwnd_{max}$). If the difference between the value of congestion window after the loss and the middle point $(cwnd_{min} + cwnd_{max})/2$ is lower than a threshold S_{max} , the protocol starts a binary search algorithm increasing $cwnd$ to the middle point, otherwise the protocol enters the linear increase phase. 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 threshold S_{min} and the congestion window is set to $cwnd_{max}$. When $cwnd$ is greater 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.

4) TCP Cubic : TCP Cubic [18] simplifies the dynamics of the congestion window employed by TCP BIC and improves its TCP-friendliness and RTT-fairness. When in the probing phase, the congestion window is set according to the following equation:

$$cwnd \leftarrow C(t - K)^3 + max\_win \quad (3)$$

where C is a scaling factor, t is the time elapsed since the last $cwnd$ reduction, max_win is the $cwnd$ reached before the last window reduction, and K is equal to $\sqrt[3]{max\_win \cdot b/C}$, where b is the multiplicative factor employed in the decreasing phase triggered by a loss event.

According to (3), after a reduction the congestion window grows up very fast, but it slows down as it gets closer to max_win. At this point, the window increment is almost zero. After that, $cwnd$ again starts to grow fast until a new loss event occurs.

B. Live performance evaluations of HSDPA networks

In [12], authors report goodput and one-way delay measurements obtained over both HSDPA and WCDMA networks from the end-user perspective, focusing in particular on VoIP and web applications. Regarding TCP, the paper reports that HSDPA provides better results with respect to WCDMA. In particular, the maximum value measured for the goodput is close to the advertised downlink capacity that was 1Mbps, whereas concerning the one-way delay, the reported measured average value is around 50ms. In the case of the HSDPA network, the number of spurious timeouts due to link layer retransmission is also lower than in the case of WCDMA due to the employment of the ARQ mechanism in the Node-B rather than in the RNC.

In [13], authors perform measurements related to physical, data-link and transport layer, in order to evaluate the interactions between these levels when variations in the wireless channel conditions occur. Regarding TCP performances, authors report measurements of goodput, retransmission percentage and excess one-way delay by using TCP NewReno, TCP Westwood+, TCP Vegas and TCP Cubic. Experiments were conducted in both static and dynamic scenarios, in the case of WCDMA2000 considering one flow or four flows sharing the downlink channel.

In the single flow case, experiments in both static and mobile scenarios provide similar results; authors have found that TCP Vegas achieves a much lower goodput than the other variants, with the lowest packet loss. The other variants

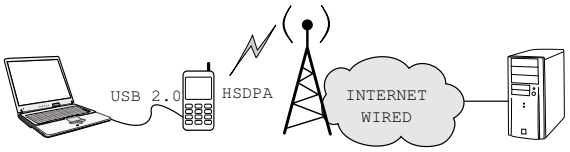

Fig. 1: Experimental testbed

generally achieve higher goodput at the expense of higher packet delays, with TCP Cubic exhibiting the largest latency.

In this paper we have not considered TCP Vegas because of its known problems in the presence of reverse traffic [9], [16].

In [5] we carried out an experimental evaluation of TCP NewReno, TCP BIC, and TCP Westwood+ when accessing UMTS downlink and uplink channels. We found that the three considered TCP variants performed similarly on the downlink. In particular we found: 1) a low channel utilization, less than 40%, in the case of a single flow accessing the downlink; 2) a high packet retransmission percentage that was in the range $[7, 11]\%$; 3) a high number of timeouts, quantified in 6 timeouts over a 100s connection, that was not dependent on the number of flows accessing the downlink; 4) RTTs in the range $[1440, 2300]$ms increasing with the number of concurrent flows.

III. Experimental Testbed

Figure 1 shows the employed testbed which is made of two workstations equipped with the Linux Kernel 2.6.24 patched with Web100 [17]. TCP flows have been generated and received using `iperf`[2], which was instrumented to log instantaneous values of internal kernel variables, such as *cwnd*, *RTT*, *ssthresh* by using `libweb100`.

A laptop is connected via USB 2.0 to the User Equipment (UE), which is a mobile phone equipped with a commercial HSDPA card provided by a local mobile operator. The UE has been tested in a static scenario so that handovers could not occur during measurements. The other workstation, instead, was connected to the Internet using an Ethernet card.

The considered TCP variants have been evaluated over the downlink channel in the cases of single, 2, 3 or 4 concurrent connections.

For each experiment run, we have injected TCP flows by rotating the four considered TCP variants, repeating this cycle many times, resulting in 55 hours of active measurements involving 2500 flows. The experiments have been executed in different hours of the day and over many days.

For each flow we have logged the most relevant TCP variables and we have computed a rich set of TCP metrics, such as goodput, throughput, round trip time, number of timeouts, packet loss ratio. In the case of N concurrent flows, the fairness has been evaluated using the Jain Fairness Index [3] defined as $JFI = (\sum_{i=1}^{N} g_i)^2/(N \sum_{i=1}^{N} g_i^2)$ where g_i is the average goodput obtained by the i-th concurrent flow.

[2]http://dast.nlanr.net/Projects/Iperf/

#	NewReno	Westwood+	BIC	Cubic
1	383 (+1.6%)	377 (0%)	519 (+37%)	582 (+54%)
2	463 (+11%)	415 (0%)	537 (+39%)	571 (+37%)
3	550 (+5%)	521 (0%)	606 (+16%)	637 (+22%)
4	609 (+11%)	549 (0%)	665 (+22%)	647 (+18%)

TABLE I: Average values (in ms) of RTT over the HSDPA downlink

IV. Experimental Results

In this Section, we report the main measurements obtained over the downlink channel. Cumulative distribution functions (CDF), along with average values of each metrics are shown. In the box-and-whisker diagrams shown in this Section the bottom of each box represents the 25-th percentile, the middle line is the median value, whereas the top of each box represents the 75-th percentile. The length of the whiskers is 1.5 times the interquartile range. The average value is represented with a cross and the outliers are not shown.

A. Round Trip Time measurements

Figure 2 shows the cumulative distribution functions (CDF) of the average round trip time (RTT) experienced by a flow for each considered TCP variant, and in the case of one, two, three and four flows sharing the HSDPA downlink, respectively.

In all the cases, there is a remarkable difference between the pair of algorithms formed by TCP NewReno and TCP Westwood+, and the pair formed by TCP BIC and TCP Cubic, with the latter producing higher delays.

It is worth noting that TCP Westwood+ provides the lower round trip times in all considered scenarios. Figure 2 shows that the 85th percentile RTT_{85} of TCP Westwood+ is around 530ms whereas the RTT_{85} of TCP Cubic is around 760ms, in all the considered scenarios.

Table I summarizes the average RTTs for each considered algorithm and scenario: in parenthesis we report the relative RTT percentage increase with respect to the lowest average value. In particular, in the case of the single flow, TCP Cubic provides an average RTT that is 54% higher than that of TCP Westwood+.

It is interesting to compare average values measured over the HSDPA downlink with those obtained over UMTS access links and reported in [5]. For the HSDPA network, measured values were in the range [377,665]ms, whereas for the UMTS network they were in the range [1102,1550]ms.

B. Timeouts

Figure 3 shows a box-and-whisker plot of measured number of timeouts per connection when one, two, three or four flows shared the HSDPA downlink.

Again, there is a remarkable difference between the NewReno-Westwood+ TCP pair, and the BIC-Cubic TCP pair. In fact, in all the cases the 50% of the connections that use NewReno or Westwood+ experience around a single timeout in 180s; on the other hand, BIC or Cubic flows experience three timeouts during the same connection duration.

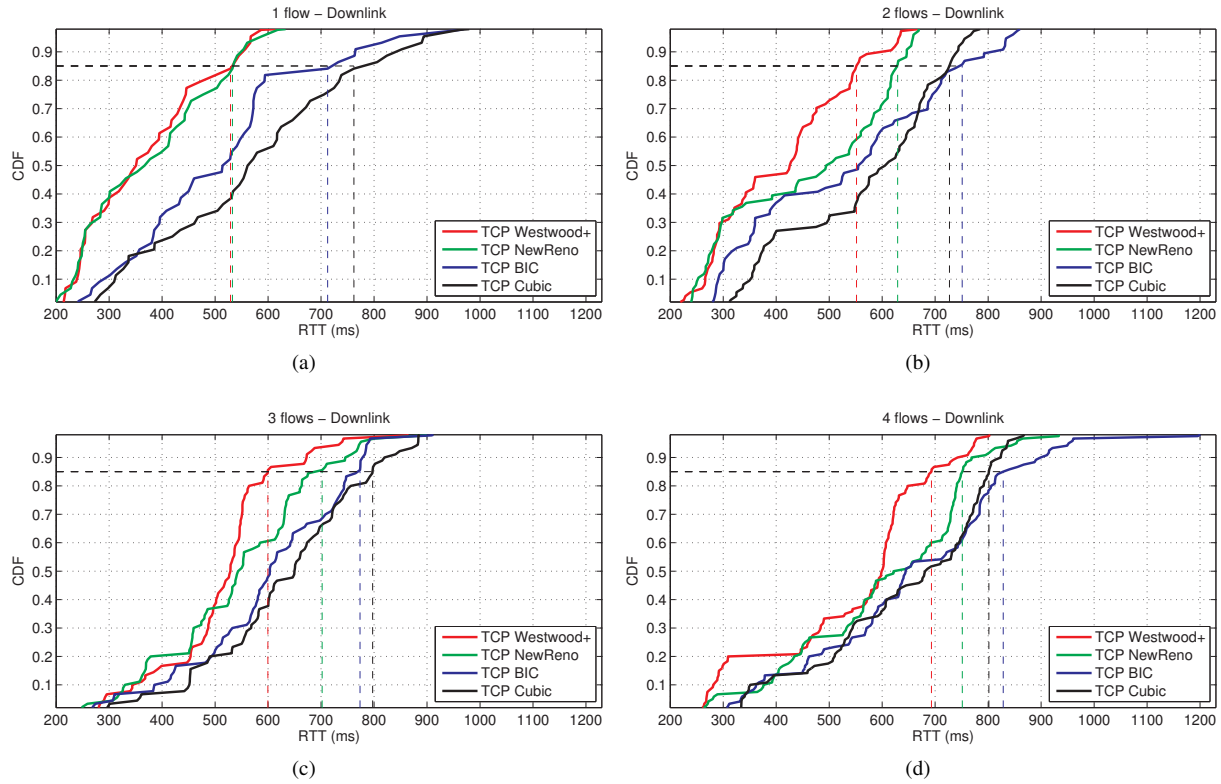

Fig. 2: CDFs of the RTT in the case of one (a), two (b), three (c) and four (d) flows sharing the HSDPA downlink. RTT_{85} are shown in dashed lines

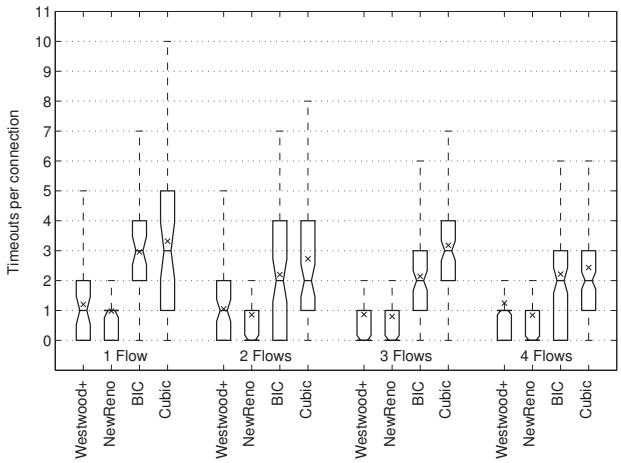

Fig. 3: Box-and-whisker plot of number of timeouts per connection

Finally, it is worth noting that the average number of timeouts obtained over HSDPA is remarkably lower with respect to the case of the UMTS downlink, where the average was around 6 in 100s [5].

C. Packet Retransmissions

Figure 4 shows the cumulative distribution functions of the packet retransmission percentage in the case of HSDPA downlink, whereas Table II reports the average values. Also in this case, TCP BIC and TCP Cubic provoke higher packets

#	NewReno	Westwood+	BIC	Cubic
1	0.052 (0%)	0.053 (+2%)	0.10 (+92%)	0.16 (+207%)
2	0.11 (0%)	0.12 (+9%)	0.21 (+90%)	0.33 (+200%)
3	0.17 (+6%)	0.16 (0%)	0.31 (+93%)	0.56 (+250%)
4	0.29 (+3%)	0.28 (0%)	0.46 (+64%)	0.80 (+185%)

TABLE II: Average values (in %) of packet retransmissions over the HSDPA downlink

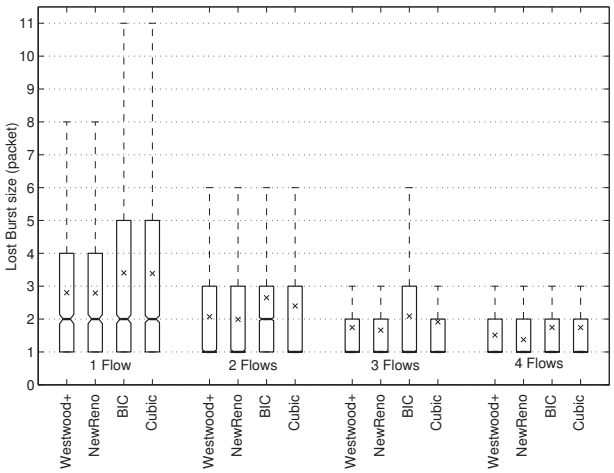

Fig. 5: Box-and-whisker plot of retransmission burst size

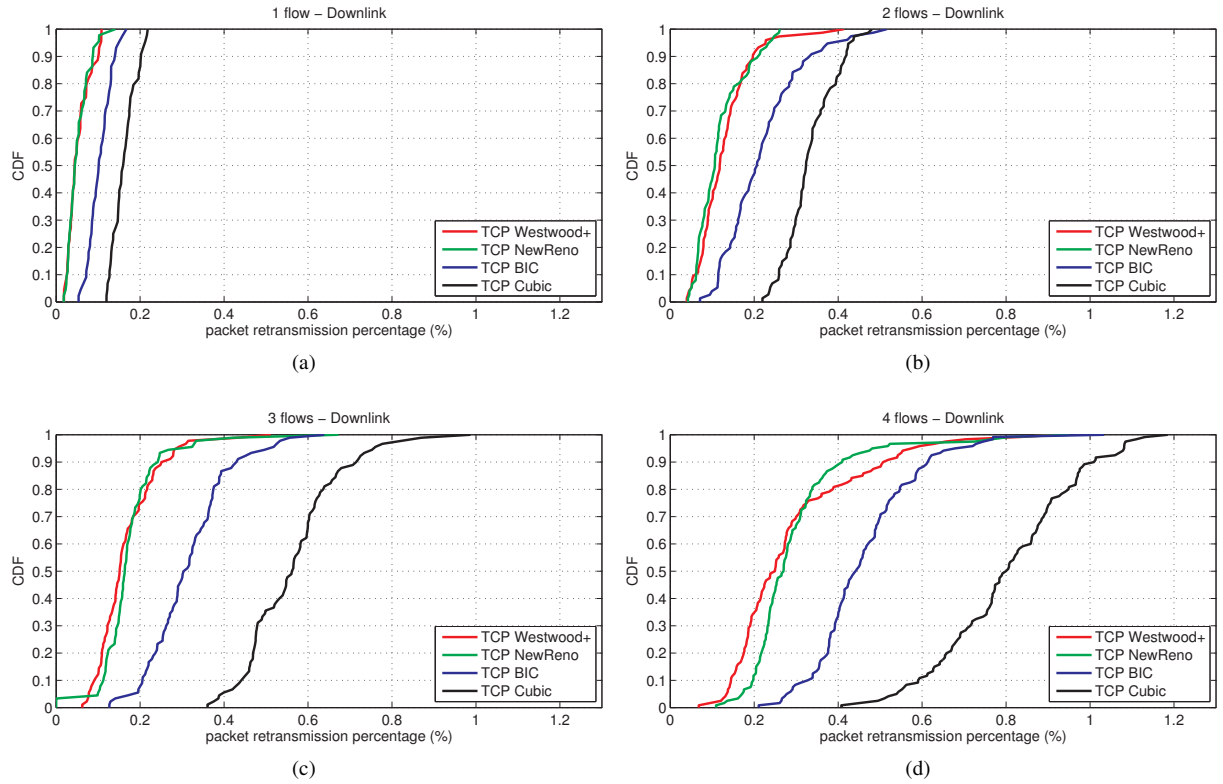

Fig. 4: CDFs of the packet retransmission percentage in the case of one (a), two (b), three (c) and four (d) flows sharing the HSDPA downlink

retransmission percentages than those of TCP NewReno or Westwood+, with TCP Cubic generating retransmissions percentages three times larger than TCP NewReno or TCP Westwood+.

From Table II, the average packet retransmission percentages belong to the range [0.052,0.80]%; these values are negligible with respect to those found in the UMTS downlink channel [5], where percentages in the range from 7% to 11% were reported.

Another important aspect to consider is the burst size of the loss events, which is the number of packets that have to be retransmitted when a loss event occurs. Figure 5 shows a box-and-whisker diagram of the retransmission burst size for the considered protocols in all the considered scenarios. It shows that the retransmission burst sizes decrease when the number of concurrent flows increases and that TCP Westwood+/NewReno pair tends to produce shorter retransmission burts with respect to TCP BIC/Cubic pair in the case of a single flow accessing the downlink.

D. Goodput, Aggregated Goodput and Fairness

Table III reports the average per-connection goodput measured over the HSDPA downlink. All the algorithms provide a similar average goodput per-connection, which are around 1400Kbps in the single flow case. By increasing the number of connections N, the goodput decreases roughly as $1/N$.

Figure 6 shows the aggregate goodput, which is the sum of the goodput of each connection when more concurrent flows share the downlink. In all the considered cases, each TCP variant provide similar values for the aggregated goodput that is around 1400 Kbps.

#	NewReno	Westwood+	BIC	Cubic
1	1443 (-1%)	1406 (-3%)	1456 (0%)	1439 (-1%)
2	790 (-2%)	777 (-4%)	809 (0%)	806 (~0%)
3	500 (-1%)	488 (-3%)	505 (0%)	503 (~0%)
4	366 (-4%)	374 (-3%)	374 (-3%)	386 (0%)

TABLE III: Average per-connection goodput (in Kbps) over the HSDPA downlink

Also the measured Jain fairness indices, obtained when many TCP flows share the HSDPA downlink channel, are all close to 0.98, which is a high value.

V. Conclusions

In this paper we have tested four relevant TCP congestion control algorithms over a commercial HSDPA network. Key performance variables such as goodputs, retransmissions percentage, number of timeouts and round trip times have been measured.

All the TCP variants provide comparable goodputs but with a larger number of retransmissions and timeouts in the case of BIC/Cubic TCP, which is a consequence of their more aggressive probing phases. On the other hand, TCP Westwood+ provides the shorter round trip delays.

The experiments have shown that the HSDPA downlink channel does not exhibit any remarkable issues, achieving good goodput, low number of timeouts and retransmission

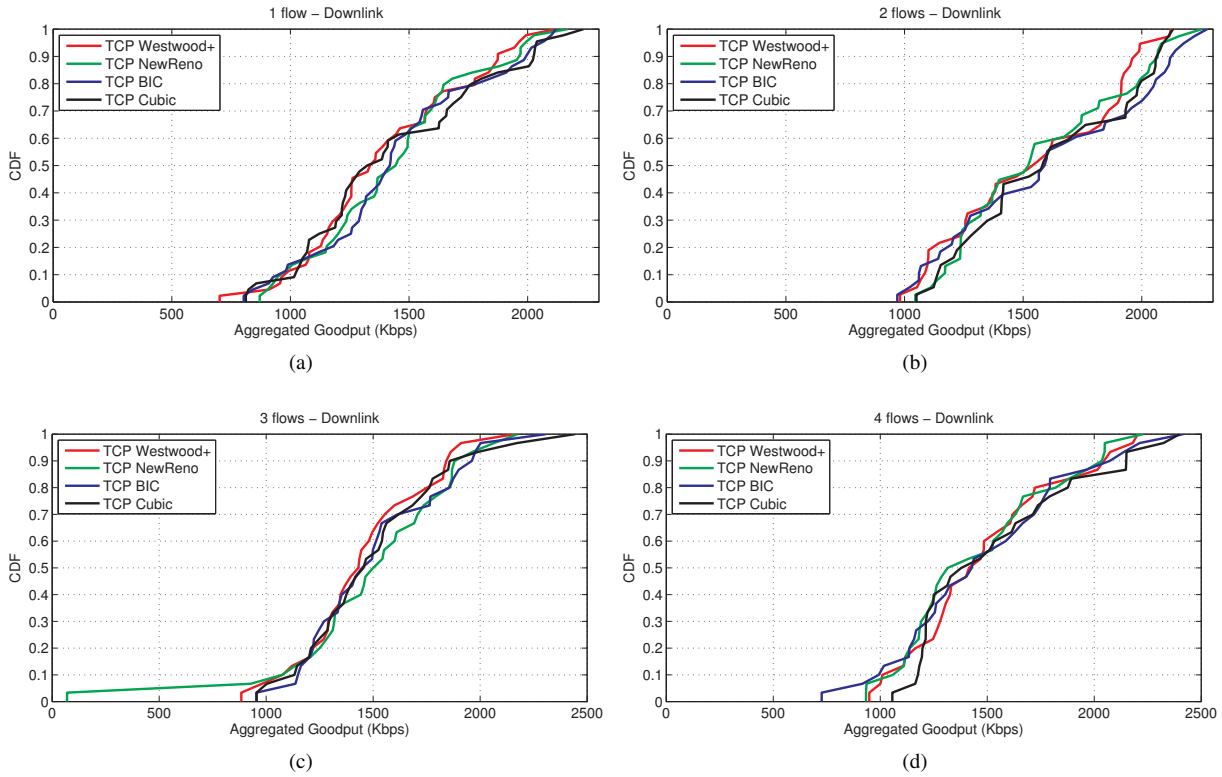

Fig. 6: CDFs of the aggregated goodput in the case of one (a), two (b), three (c) and four (d) flows sharing the HSDPA downlink

percentages when using classic TCP NewReno or TCP Westwood+, both implementing standard congestion avoidance phase. The more aggressive probing phase of BIC/Cubic TCP does not improve the goodput and it increases the number of timeouts and retransmissions, which is bad for the network. Finally, RTT is also higher with respect to NewReno/Westwood+ due to higher queuing time. This may be an important result to be considered since TCP Cubic is currently the default congestion control algorithm in the Linux OS. Moreover, TCP Westwood+ provides the shorter round trip times due to the cwnd setting after congestion that clears out the queue backlog along the connection path [15].

REFERENCES

[1] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. *RFC 2581, Standard*, April 1999.

[2] H. Balakrishnan, VN Padmanabhan, S. Seshan, and RH Katz. A comparison of mechanisms for improving TCP performance overwireless links. *IEEE/ACM Trans. on Networking*, 5(6):756–769, 1997.

[3] D. 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.

[4] Cisco. Cisco Visual Networking Index:Forecast and Methodology 2009-2014. *White Paper*, June 2010.

[5] L. De Cicco and S. Mascolo. TCP Congestion Control over 3G Communication Systems: An Experimental Evaluation of New Reno, BIC and Westwood+. In *Proc. of NEW2AN '07*, September 2007.

[6] D.A. Eckhardt and P. Steenkiste. Improving wireless LAN performance via adaptive local error control . In *Proc. of IEEE ICNP '98*, pages 327–338, Oct 1998.

[7] S. Floyd, T. Henderson, and A. Gurtov. NewReno modification to TCP's fast recovery. *RFC 3782, Standard*, April 2004.

[8] C.P. Fu and SC Liew. TCP Veno: TCP enhancement for transmission over wireless access networks. *IEEE Journal on selected areas in communications*, 21(2):216–228, 2003.

[9] L.A. Grieco and S. Mascolo. Performance evaluation and comparison of westwood+, new reno, and vegas tcp congestion control. *ACM Comput. Commun. Rev.*, 34(2):25–38, 2004.

[10] S. Ha, Y. Kim, L. Le, I. Rhee, and L. Xu. A step toward realistic performance evaluation of high-speed TCP variants. In *Proc. of Protocols for Fast Long-distance Networks*, February 2006.

[11] Van Jacobson. Congestion avoidance and control. *ACM SIGCOMM Comput. Commun. Rev.*, 18(4):314–329, 1988.

[12] M. Jurvansuu, J. Prokkola, M. Hanski, and P. Perala. HSDPA Performance in Live Networks. *Proc of IEEE ICC'07*, pages 467–471, 2007.

[13] X. Liu, A. Sridharan, S. Machiraju, M. Seshadri, and H. Zang. Experiences in a 3G network: interplay between the wireless channel and applications. In *Proc. of ACM MOBICOM '08*, pages 211–222, 2008.

[14] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang. Tcp westwood: Bandwidth estimation for enhanced transport over wireless links. In *Proc. of ACM MOBICOM '01*, pages 287–297, 2001.

[15] S. Mascolo and F. Vacirca. Congestion Control and Sizing Router Buffers in the Internet. In *Proc. of 44th IEEE Conference on Decision and Control*, pages 6750–6755, Dec. 2005.

[16] S. Mascolo and F. Vacirca. The effect of reverse traffic on TCP congestion control algorithms. In *Proc. of Protocols for Fast Long-distance Networks*, February 2006.

[17] M. Mathis, J. Heffner, and R. Reddy. Web100: extended tcp instrumentation for research, education and diagnosis. *SIGCOMM Comput. Commun. Rev.*, 33(3):69–79, 2003.

[18] I. Rhee and L. Xu. CUBIC: A new TCP-friendly high-speed TCP variant. In *Proc. of Protocols for Fast Long-distance Networks*, 2005.

[19] M. Sauter. *Beyond 3G - Bringing Networks, Terminals and the Web Together*. John Wiley & Sons, 2008.

[20] L. Xu, K. Harfoush, and I. Rhee. Binary increase congestion control (BIC) for fast long-distance networks. In *Proc. IEEE INFOCOM 2004*, 2004.