Residual Capacity Estimator for TCP on wired/wireless links

Claudio E. Palazzi – *cpalazzi@cs.ucla.edu*

Computer Science Department, University of California Los Angeles
Boelter Hall, Los Angeles CA, 90095 USA

Dipartimento di Scienze dell'Informazione, Università di Bologna
Via Mura Anteo Zamboni 7, 40127 Bologna, Italy

Abstract

Having an inexpensive and accurate means to estimate capacity could be put in good use for several purposes: QoS schemes could utilize it to claim the feasible ratio at which video/audio streaming are allowed to transmit, bandwidth estimators could receive help in discriminating between their samples. Nowadays, the plethora of applications available to users and the widely adopted mobile technology require new features that only an deeper awareness of the accessed link characteristics could provide.

In this paper, we present RCE (Residual Capacity Estimator), a new scheme able to provide a simple and effective esteem of the bottleneck link capacity deducted the uniformly distributed traffic present. Our scheme is embedded in the normal TCP functionalities and achieves precise results even from the very beginning of a connection. We provide simulation results under various and complex situations in order to prove the efficacy of RCE and we conclude with future directions for this work.

Keywords: RCE, TCP Westwood, Wireless, Capacity Estimation, Packet Train.

1. Introduction

Even if, at present, packet switched technology can be considered mature, new applications as peer-to-peer, remote working, video/audio streaming, television distribution and interactive online gaming present new challenges that require at least an adaptation or, in some cases, a complete redesign of the currently used mechanisms. New applications usually necessitate of capabilities that where not included in the original design of the Internet, thus asking for solutions able to provide the efficiency needed in order make the new products widely adopted by the final users.

In order to really understand the complexity of this scenario we also have to mention the high number of people using wireless technology. Cellular phones, laptops and PDAs are devices owned by a very elevated and still increasing percentage of people and, with the introduction of the higher data rates of the third generation (3G) of mobile systems [10], we can easily foresee a raising request for new services able to push people in purchasing this new technology. The new "killer applications" able to alter the trajectory of the market will probably come from the interaction between the new devices and the Internet. Indeed, virtual libraries, video-telephony over IP, videoconferencing, games, remote-medicine, video and music on demand and locality based information are only few of the innumerable services that will be available in every place and at every time.

On the other hand, the interaction between mobile devices and the Internet, coupled with the need for an efficient and reliable data transfer have generated several unresolved problems [5], amongst the other, the high error rate present on the channel. Therefore, applications that rely on a reliable transmission protocol are especially affected by a wireless environment. Focusing on the TCP (Transmission Control Protocol), indeed the most popular transport protocol for reliable data delivery, we have also to notice that it was designed in a time when networks were exclusively based on wired technology. For this reason, the flow control and the congestion control functions fail when introduced into a wireless context [13]. To really understand the reasons of this failure we should remember that traditional TCP uses packet losses as a metric to evaluate the congestion level of the network. Consequently, when a packet is considered to be lost, TCP reduces the data sending rate. In presence of numerous losses related to non-congestion factors, as in a wireless environment, this behavior is not appropriate and causes a consistent underutilization of the available bandwidth on the link [4]. Having a means to precisely estimate the accessible bandwidth could help in discriminating between the congestion related losses and wireless error losses. In order to avoid potential overestimation of the available bandwidth it could be useful to have a tool able to provide a capacity measure of the bottleneck. Not only this value could be used as an upper bound for data sending rates, but it could also be put in good use in other applications as, for instance, to determine appropriate routes/trees for multicast overlay networks [12].

In Section 2 we briefly present related works which regards estimation either of bandwidth or capacity, in Section 3 we propose the reasoning that brought us to design our new link capacity estimator, in Section 4 we explain the simulated environment used to test the new mechanism, in Section 5 we show the results obtained in simulations by our scheme and Section 6 concludes this work and presents future directions.

2. Related works

In recent years, many researchers have focused their studies on the obstacles present in a wireless environment proposing various alternative techniques [14]. Several solutions proposed to face those problems relies on the ability to estimate the factual capacity or the available bandwidth. Equipped with these information, protocols could be able to set the most appropriate sending parameters decoupled from the losses.

Tsaoussidis and Badr propose the use of special pairs of probe packets after each loss [17]. When one of these couples reaches the receiver, the measured delay of these probing packets, is used to understand if the network is congested and, only in this case, to diminish the data sending rate. If the wireless link is experiencing a disconnection or a fading, the probe cycle is extended, thus avoiding further data losses and consequently erroneous restrictions of the sending rate. Their purpose is also obtaining a better energy resource consumption. This protocol requires modification at both sender and receiver side in order to handle with probing packets: this lack of compatibility with the current implementation of the TCP seriously affects its real development possibility.

A new end-to-end transport protocol is also proposed by Sinha et al. [18]. They face the high number of errors and the variable latency of a wireless environment eliminating the timeout mechanism, using periodic SACK packets [7] to understand when a retransmission is appropriate, and estimating the channel capacity to set the data sending rate. In their protocol, the departure time is included in each packets, thus the receiver can use this information and the interarrival times to measure the bandwidth and communicate the sending rate back to the sender by SACK packets. If the sender doesn't receive any SACK for a long time, it suspends the data transmission and starts sending probe packets until an acknowledgment from the receiver makes it resume the communication. Since the bandwidth estimation is computed at the receiver, the packet data rate calculation is quite accurate. Despite this, Sinha's mechanism development in the TCP/IP stack is limited by the requirement of modifications on both sender and receiver hosts.

Mascolo et al. suggest a new transport protocol which uses a sender side end to end estimation of the available bandwidth [1]. While traditional TCPs blindly reduce the data sending rate every time a data packet is lost, the key innovative idea in their work is to use the rate of the returning acknowledgments of received data to measure the effective link availability. This bandwidth estimation is computed by sampling and exponential filtering methods that have been progressively refined in order to be effective and fair at the same time [2][3] and then used, after a loss or during slow start to appropriately set the slow start threshold. This protocol has shown great results on big pipes, even if affected by error losses, as for example satellite links; however, in some other wired-cum-wireless connection with a very narrow links, as with 802.11, 1xRTT and Bluetooth, the behavior seems to be too conservative if compared with TCP Newreno [11].

Dovrolis et al. present a capacity estimation scheme which starts making use of packet pairs but, in case of multimodal distribution, the number of packets used is increased until having a value N for the cardinality of the packet train that produces an unimodal distribution [15]. Their scheme results to be accurate but too slow.

Finally, Kapoor et al. suggest to use a packet pair scheme to estimate the capacity of a link [8]. In particular, between the various samples, they suggest to use that one obtained with the packet pair having the *minimum delay sum*. That value, in fact, has probably not suffered by cross traffic thus giving the most accurate measure. Their work shows maximum values close to the real channel capacity but, on the other hand, packet pair techniques tend to suffer when coupled with TCP because of the bulk nature of its transmissions and the use of delayed acks.

3. Rationale of the idea

A packet train is a set of packets which depart from the sender one close to the other. Their leaving time is beat by the transmission rate available, thus depending on the outgoing capacity. Along their path to reach the destination, the packets composing the train could encounter links with lower bandwidth than that present on the first one traversed, consequently, packets requires more time to be transmitted and the train becomes longer. This dispersion could be caused not only by narrow links, but also by time spent in queue due to other traffic sharing part of the same connection. The correspondent acks go back to the sender triggering new transmissions at a rate which hence depends on all these factors and causes packets coming out separated by gaps of idle time.

In Fig. 3.1 time is divided into slots; using these slots as the measurement units, we indicate as *X* the part used to transmit the packets back to back and with *Y* the time needed to contain the dispersion of the correspondent acks. Considering the time slot equal to 1, it is easy to understand that the maximum portion of the slot usable to send packets corresponds to $\frac{X}{Y}$. The rate at which the sender could transmit is given by $\frac{Bits\_transmitted}{X}$, hence to obtain the maximum rate usable, given the bottleneck, we simply multiply for $\frac{X}{Y}$ obtaining $\frac{Bits\_transmitted}{Y}$.

In a situation with a single flow and having the packets leaving the sender back to back, the formulas given above easily determine the capacity of the connection. If we introduce in this scenario other traffic, we also have to take into account the possibility that packets could leave the source distributed on the whole slot. In fact, the effective transmitting time of the sender will probably be divided into several chunks of time having gaps in between them that remain unused. In those pauses, the sender just waits for new acks in order to be allowed to transmit new data. Consequently, the correspondent acks distribution will be characterized by gaps that does not depend on the bottleneck size.

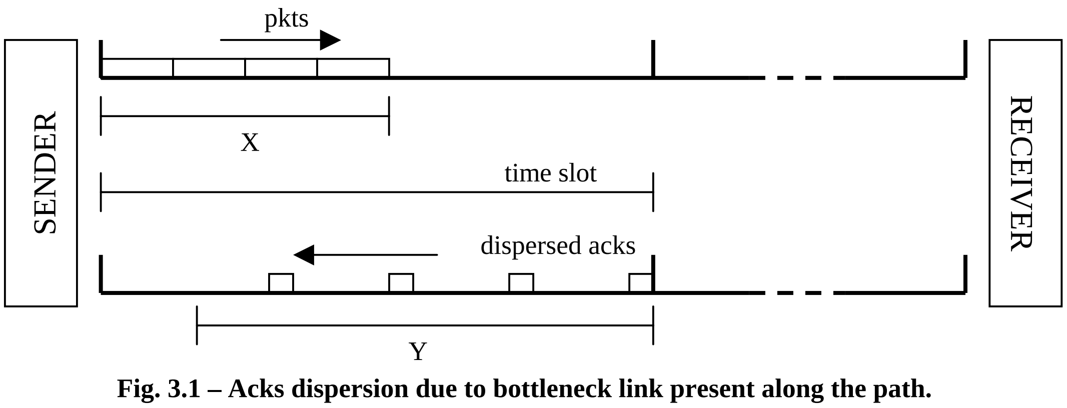

Fig. 3.1 – Acks dispersion due to bottleneck link present along the path.

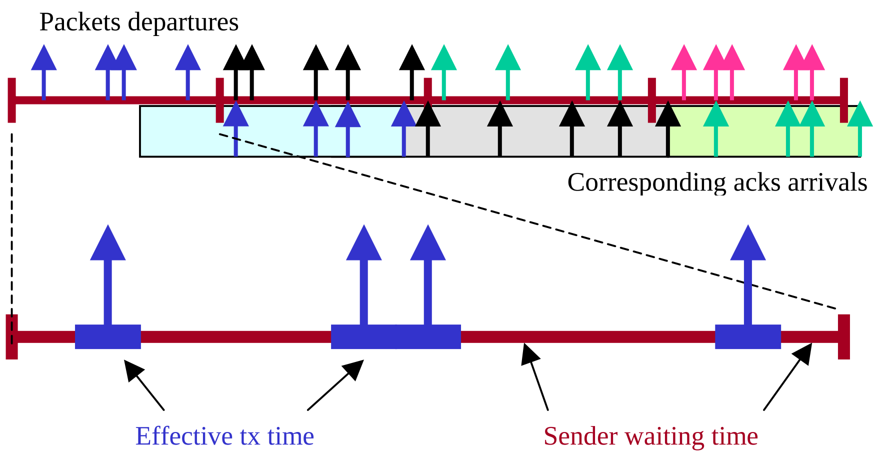

Fig. 3.2 – Packets-slots and acked-slots division.

We can distinguish between several causes that insert gaps between returning acks:

- Different capacity between outgoing link and bottleneck link
- Queuing time caused by congestion
- Different size, and thus channel occupancy, between data packets and acks
- Wasted time due to a low sending window

RCE eliminates the last type of waiting time from the above list maintaining, at the same time, the dispersion of the packets due to links with different total bandwidth along the path. In this way, we are able to obtain an accurate estimation of the capacity of the bottleneck link.

The scalability of the mechanism is assured by the very easy set of calculations and by the very few information we need to store at sender side about the TCP flow. Time is divided into slots: we set *packets-slots* as large as an RTO [6] and we count packets leaving in that period, we then wait for correspondent returning acks to determine the *acks-slots* as shown in Fig. 3.2. Since the number of packets-slots and of acks-slots is the same, on average they will have the same length.

When the acks corresponding to the sent packets come back, we compute the bottleneck capacity as $\frac{Bits\_acked}{Acks\_slot\_time - Wasted\_time}$. With RCE, we utilize the dispersion of the acknowledgments, the source ratio and the information regarding the time wasted with no data sent to the destination due to a low sending window. In particular, the last element is the introduced enhancement in our scheme that allows to have a correct estimation since the very first period of a connection making use of a mechanism which is perfectly embedded into the usual TCP operations. This wasted time at sender, in fact, potentially affects the effectiveness of all the bandwidth or capacity estimators that relies on returning acknowledgements. Following the aim of considering just the wasted time due to a little sending window while maintaining the dispersion induced by the narrower links, we have to calculate the Wasted_time from the acks_slot standpoint. The proposed scheme measures the average of the interarrival time between the acks of the slot (the first one is counted from the beginning of the slot, the slot ends receiving the last ack belonging to it); the Wasted_time is then computed as the sum of the time exceeding this average in each interarrival time of the acks_slot. This formula is justified by the fact that all the included packets will experience the same channel conditions in terms of transmission time: consequently, the exceeding gap times between acks is most likely a result of having periods of no transmissions due to the sending window size. Focusing on queuing time, we have to notice that, since the bulk nature of TCP transmission, this element will not be endured equally by all packets in a slot.

- At sender side, time is divided into slots
- In each slot, N packets are sent to destination
- The corresponding N acks will return back in Acks_slot_time
- Sender_waiting_time is calculated as:

```
Sender_waiting_time = 0;
calculate AVG_acks_interarrival_time;
for each acks_interarrival_time of the slot {
      if acks_interarrival_time > AVG_acks_interarrival_time {
            Diff = acks_interarrival_time – AVG_acks_interarrival_time;
            Sender_waiting_time = Sender_waiting_time + Diff;
      };
};
```

- The sample is:

```
Cap_sample = Bits_acked / (Acks_slot_time – Sender_waiting_time)
```

- This is averaged as:

```
Cap_est = 0.5 * Cap_sample + 0.5 * Cap_est
```

Tab. 3.1 – The proposed scheme.

This difference in time spent in queue will produce acks with higher interarrival time than the average. Consequently, queuing time is removed by our mechanism in case of contemporary presence of other TCP flows and the final result is the capacity of the bottleneck. Conversely, if the considered TCP connection competes for the channel with other CBR flows, the outcome has a different meaning. The distribution shape of a CBR transmission, in fact, uniformly occupies a portion of space both in the channel and in the queues. Since all the interarrival times of the acks are in this case affected by the same amount of queuing time, our mechanism leaves it in the average of the interarrival time. Thereby, the final estimation will count also queuing time caused by the CBR traffic thus computing the shared bandwidth. Summarizing what we are going to demonstrate with simulations in Section 5, we can say that RCE returns the bottleneck capacity, detracted the portion of channel occupied by the uniformly distributed traffic.

We can summarize the concepts above noticing that RCE recalls the packet train technique. The main difference is the fact that no special packets are utilized: the scheme is perfectly embedded in the usual TCP operations. Packets are sent as usual to destination causing acks response from receiver which generates new transmissions. The outcome is depurated from the wasted time at sender side, virtually reproducing the results that we would have obtained from an initial configuration of a compact packet train. Since in this way we have a new usable sample every RTO, this also solve the problem of having few natural packet pairs on a TCP connection seen in [9]. The scheme here proposed is recapitulate in Tab. 3.1.

4. Simulation environment

NS-2 is a widely used network simulator [16], many scientific articles regarding various aspects of networking are based on this tool. In particular, since the widely accepted reliability of simulations results obtained from modifications of transport layer protocols, NS-2 is the standard for simulations of TCP. In this work we have utilized the seventh version.

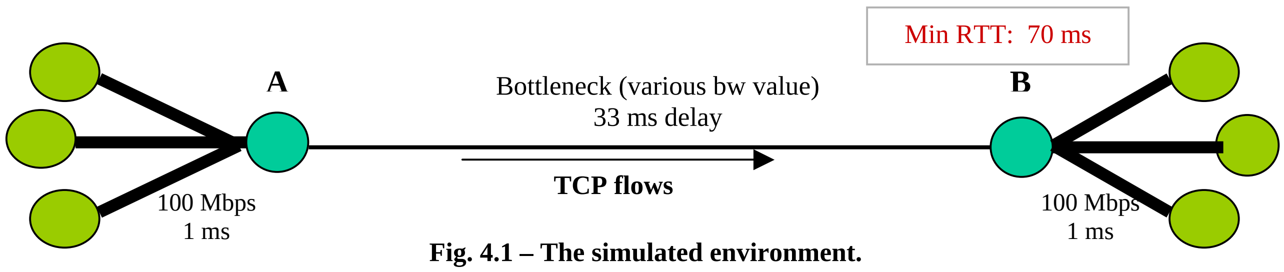

Fig. 4.1 – The simulated environment.

The configuration of the simulations is intuitively understandable from Fig. 4.1. One or more connections share a link which represents the bottleneck. In particular, nodes have access to the common link coming from very high bandwidth connections having in B an almost infinite queue. This is a simplification that allows us to have congestion only before the bottleneck link which is desirable in order to properly test the ability of RCE to detect the capacity in that point. Besides the data cited in the picture, we have used in our simulations different bottleneck bandwidths having the queue in entrance set to the pipe size. Our scheme has been tested with or without errors on the link, the former in order to simulate a wireless connection and the latter to simulate a normal wired connection, and with or without the concurrent presence of CBR (Constant Bit Rate) flow both in the same or in the opposite direction of the TCP flows. The functionalities of the transport protocol utilized have not been modified: our estimator makes calculations simply observing the normal execution of the TCP flows.

A summary of the main characteristics of the various flows follows:

- Single or multiple TCP flows (bulk transfer):
    - Type: Westwood Agile
    - Packet size: 1000 bytes

- Straight and/or reverse UDP/CBR flows:
    - Packet size: 125 Byte
    - Transmission interval: 1 ms

In order to simulate the RCE mechanism under various conditions, we have modified some NS-2 modules and written new ones in order to implement the new capacity estimation scheme. Moreover, we have prepared a tcl script that allows with minimal commands to run the various simulations having as results trace files and graphs.

5. Results of the simulations

We have verified the correctness of RCE running several simulations under various conditions. It is obviously not possible to cover here the whole plethora of all possible cases that we could encounter in the Internet but we believe that the set of situations recreated for this work allows us to state some important conclusions. Since the good performance demonstrated by the Westwood protocol in estimating the correct eligible rate under various conditions [1][2][3], we have decided to run this version of the transport protocol. Besides, this gives us an immediate comparison between an accurate shared bandwidth value and our capacity estimator. In all the graphs presented, the red line represents the congestion window (which is in our simulations always corresponding to the sending window), the green one is the slow start threshold, in yellow we have the Westwood Agile ERE (Eligible Rate Estimate) and in blue there is the capacity estimation provided by our mechanism.

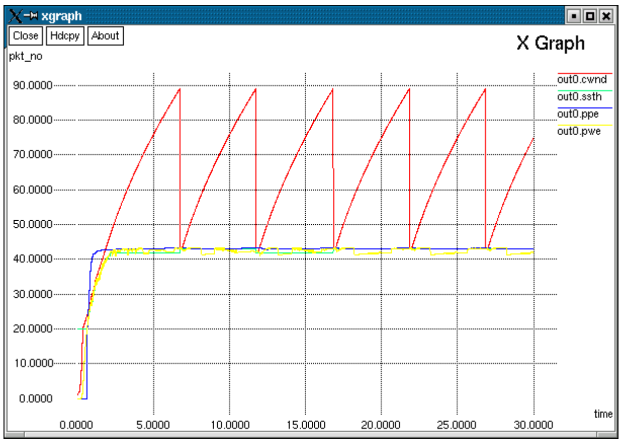

Fig. 5.1 – Single TCP flow on a bottleneck link of 5Mb, with no errors

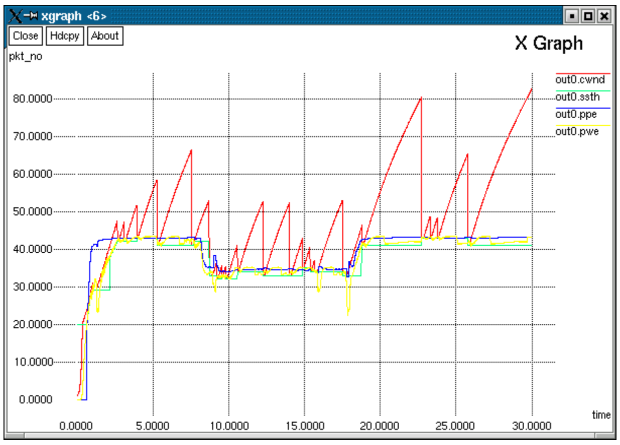

Fig. 5.2 – Single TCP flow on a bottleneck link of 5Mb, PER of 0.1% and a period of CBR traffic

From now, every time we say TCP we mean TCP Westwood Agile and for all the simulations run we have set the condition (CBR transmission interval, CBR and TCP packet size) as presented in Section 4.

In Fig. 5.1 we present the outcome of a very simple scenario with a single TCP flow on a connection with a bottleneck bandwidth of 5Mb. Since the presence of a single connection on the channel, the eligible rate and the capacity of the link are the same. As we can see, our estimator (blue line) reaches the correct value almost immediately, in particular the maximum value given by our estimator correspond to 43.25 packets while the pipe size is 44 packets. Maximum value is chosen as a parameter in order to allow comparisons with CapProbe papers [8][9].

In Fig. 5.2 we can see the outcome of a simulation run in order to test the reactivity and effectiveness of RCE, even in situation characterized by errors as those faced in a wireless environment. In particular, a single TCP flow operates from second 0 to second 30 while a CBR flow starts at second 8 of the simulation and ends at second 18. The capacity of the channel and the introduction of a uniformly distributed traffic are perfectly detected by the estimator: the latter is perceived as a decrease of the actual capacity. We have also introduced a PER (Packet Error Rate) of 0.1% to simulate the error prone conditions of a typical wireless environment: our mechanism is not affected at all by this.

Figg. 5.3, 5.4 and 5.5 show the results of a simulation in which three TCP flows compete for a common bottleneck of 10Mb over a total period of time of 80 seconds; the RTT is 70ms, thus the pipe size results to be 88 packets. Since RCE relies on the dispersion of the acks, we have inserted a CBR flow sharing the same bottleneck but in the opposite direction in order to study the impact of reverse traffic. In this configuration, the first TCP source starts sending packets at second 0 and ends at 40, the second one transmits from second 10 to 50 and the third one starts at 20 and finish at 80. As it is easy to see, in all the three cases the capacity is estimated promptly and correctly. More in detail, the maximum value estimated for the capacity is respectively 87.87, 86.71 and 86.84 which is very close to the factual pipe size of 88 packets.

Finally, we have also tested RCE with delayed acks. Obviously, delayed acks employment impacts on the acknowledgments distribution, thus affecting the detected capacity of the connection. Despite of this, results where not bad in a scenario with two TCP competing for a 10Mb bottleneck: over 30 seconds of simulation, the achieved maximum capacity estimate were respectively 83.17 and 82.22 packets over a truthful capacity of 88. Our mechanism is hence able to give an appropriate value even in a case where traditional packet pair techniques fail.

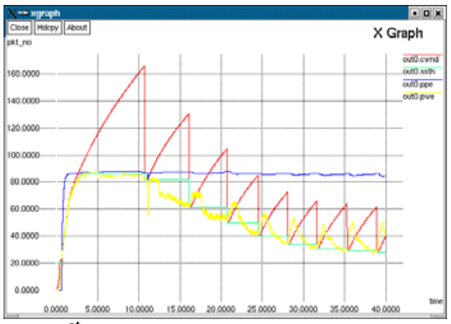

Fig. 5.3 – 1st over 3 TCP flow on a bottleneck link of 10Mb, no errors, with reverse CBR flow

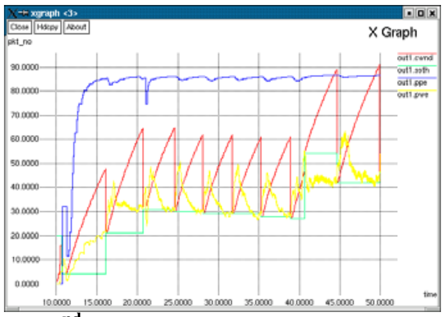

Fig. 5.4 – 2nd over 3 TCP flow on a bottleneck link of 10Mb, no errors, with reverse CBR flow

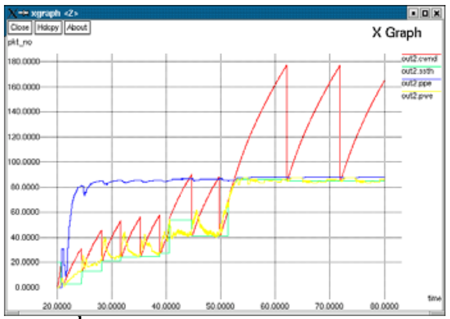

Fig. 5.5 – 3rd over 3 TCP flow on a bottleneck link of 10Mb, no errors, with reverse CBR flow

6. Conclusions and future directions.

In this work, we have studied the problem of properly estimate the capacity of a channel. In particular, we have defined the various components of the waiting time at sender side, enlightening the impact of one of its component: the wasted time due to a current low sending window. Aware of this, we have then proposed RCE, a simple and non expensive algorithm able to provide accurate estimation of the channel capacity detracted the uniformly distributed concurrent traffic, since the very beginning of a connection. Our scheme recalls the packet train mechanism but is perfectly embedded with the normal functionalities of traditional TCPs; in fact, it uses normal packets and acks and does not require the employment of apposite new packets.

The simulations results showed that RCE is highly precise and provides an appropriate value also at the very beginning of a connection. Moreover, it works properly even if plunged in an error prone environment or with preexisting or successively starting concurrent flows. Finally, despite of relying on returning acks distribution, our scheme has been demonstrated robust also against reverse flows and able to provide an acceptable esteem in case of delayed acks utilization.

Future studies include the real implementation of the proposed mechanism in order to verify the simulation results. Moreover, we would like to find a practical employment for our capacity estimator and experiment it. As first step, we could use RCE as an upper bound provider for the bandwidth sample obtained by the TCP Westwood. This will let us know if it is possible to still achieve the same accuracy in bandwidth estimation substituting the heavier filter currently used with a less computationally expensive one. As another real application, we could utilize our scheme to compute the most appropriate transmission rate for video/audio streaming in presence of QoS policies requirements. Having an accurate estimation of the capacity can help the system to claim the correct feasible rate for high priority applications regardless of low priority traffic. Furthermore, it would be interesting to evaluate our capacity estimator coupled with routing algorithms to create efficient multicast overlay networks. Finally, we are looking for enhancing our scheme in order to take into account also the non uniformly distributed traffic present on the channel, in order to construct a simple and effective available bandwidth estimator. This feature could be obtained determining the queuing wasted time due to congestion and integrating this value in our scheme.

7. References

[1] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links", *MOBICOM 2001*, July 2001.

[2] R. Wang, M. Valla, M.Y. Sanadidi, and M. Gerla, "Adaptive Bandwidth Share Estimation in TCP Westwood", *Globecom 2002*.

[3] R. Wang, G. Pau, K. Yamada, M. Y. Sanadidi, M. Gerla, " TCP Startup Performance in Large Bandwidth Delay Netwroks ", *to appear in INFOCOM 2004*, March 2004.

[4] H. Balakrishnan, V. N. Padmanabhan, S. Sehan, and R. H. Katz, "A comparison of mechanism for improving TCP performance over wireless links", *IEEE/ACM Trans. Networking*, vol. 5, no. 6, pp. 756 - 769, december 1997.

[5] C. E. Palazzi, "Protocolli di Trasporto in Ambiente Wireless", Bachelor Degree Thesis, University of Bologna, July 2002, Cesena, Italy.

[6] W. R. Stevens, TCP/IP Illustrated, Volume 1: The Protocols, Addison Wesley, 1994.

[7] K. Fall, S, Floyd, "Simulation-based Comparisons of Tahoe, Reno, and Sack TCP", *ACM Computer Communication Review*, July 1996.

[8] “CapProbe: A Simple and Accurate Technique to Measure Path Capacity”, *submnitted for publication (authors’ name withheld of double-blind reviewing)*.

[9] "Accuracy of Link Estimates using Passive and Active Approaches with CapProbe", *submnitted for publication (authors’ name withheld of double-blind reviewing)*.

[10] UMTS forum, http://*www.umts-forum.org*

[11] V. Ghini, G. Pau, M. Roccetti, P. Salomoni, M. Gerla, "For Here or To Go? Downloading Music on the Move with an Ultra Reliable Wireless Internet Application", *IEEE ICC’2004*, Paris, 2004.

[12] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, " Scalable Application Layer Multicast," *Proc. ACM Sigcomm 2002*, Pittsburgh, Pennsylvania, August 2002.

[13] G. Huston, "The future for TCP", *The Internet Protocol Journal*, vol. 3, n. 3, September 2000.

[14] G. Huston, "TCP in a Wireless World", *IEEE Internet Computing*, pp. 82 – 84, March – April 2001.

[15] C. Dovrolis, P. Ramathan and D. Moore "What do packet dispersion techniques measure?", *in Proc. Of IEEE Infocom'01*, Anchorage, Alaska, April 2001.

[16] http://www.isi.edu/nsnam/ns/

[17] V. Tsaoussidis, H. Badr, "TCP-Probing: Towards an Error Control Schema with Energy and Throughput Performance Gains", *The 8th IEEE Conference on Network Protocols*, November 2000.

[18] P. Sinha, N. Venkitaraman, R. Sivakumar, V. Bharghavan, "WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks", *ACM Mobicom '99*, August 1999.