Congestion Control in Wireless Environments: Simulation and Experimental Assessments *

M. Casoni, *Member IEEE*, and M.L. Merani, *Member IEEE*
Department of Information Engineering - University of Modena and Reggio Emilia
Via Vignolese, 905 - 41100 - Modena - Italy
phone: +39-059-2056166; Fax.: +39-059-2056129
email: casoni.maurizio@unimore.it, merani.marialuisa@unimore.it

Keywords: TFRC, TCP-friendliness, wireless LAN, IEEE 802.11

Abstract

This work investigates the TCP-friendliness of a particular congestion control mechanism, TFRC, when it is employed to regulate the transmission rate of non-TCP sources competing for bandwidth with different TCP flavors (Reno, Sack and Westwood) flows in an IEEE 802.11b wireless LAN. Throughput of simultaneously active TFRC and TCP flows have been both experimentally measured and investigated throughout simulation, revealing that the wireless indoor radio channel plays a significant role on such parameter. Moreover, some useful insights have been derived on the behavior of TFRC and its degree of friendliness in an environment featuring a low degree of statistical multiplexing, such as the wireless context examined.

1 Introduction

Research today is swiftly flowing through a challenging era, where wireless networking solutions enjoy unprecedented popularity: their diffusion boosts so quickly to make market analysts claim that wireless LANs will represent a revolutionary upheaval, just like the Internet did, not so many years ago. New, multimedia content-aware services are unceasingly devised for the deployment in the radio environment, while the Wi-Fi IEEE 802.11 standard continues its unfailing race towards higher and higher transmission rates: at the time of this writing, its new, powerful amendment, IEEE 802.11g, is only one month old. However, many design and performance aspects of wireless LANs still remain obscure: they reveal the necessity to build a thorough, crosswise knowledge of the different transmission and networking views that such LANs require to become fully competitive in the long term. This article undertakes the charge to spotlight one research issue pertinent to the world of IEEE 802.11, as it critically addresses the combined performance of TFRC (TCP Friendly Rate Control Protocol) and TCP in a wireless LAN. Indeed, wired and wireless multimedia applications such as real-time audio/video streaming are becoming more and more popular. Yet, these applications rarely employ TCP, and consequently do not fairly compete for bandwidth with TCP-based protocols such ast HTTP, SMTP, FTP: their greedy behavior can lead to starvation of TCP traffic, or even to a congestion collapse [1], [2]. Hence the necessity to devise proper congestion control schemes for non-TCP real-time traffic: within this realm, TFRC has recently emerged as a good candidate for congestion control of unicast traffic. TFRC is an equation-based congestion control mechanism [11]-[5], designed to guarantee smoother throughput than TCP, while being ”reasonably fair” with TCP flows. Unlike TCP, it does not halve the sending rate in response to a single congestion indication, therefore avoiding the abrupt ”sawtooth” behavior of TCP congestion window, that so badly matches real-time applications.

As regards current literature, TCP behavior in a wireless LAN has been investigated in [6]; the achievable throughput of an IEEE 802.11b network is determined in [7], by simulation and analysis; some among the most recent contributions to the study of TFRC are [8] and [9], while [10] provides interesting guidelines to understand when an equation-based control is indeed TCP-friendly.

Although some tests to analyze how TFRC behaves in environments with a low level of statistical multiplexing have been performed before, see e.g. [11] for cable modems, to the authors' knowledge TFRC response in an 802.11 network has not been investigated before. The main goal of this paper is therefore to analyze TFRC in such a wireless context, where the unreliable radio channel plays a significant role, highlighting how fairly TFRC flows compete with TCP Reno, Sack and Westwood connections: friendliness and throughput will therefore be the main performance parameters. Our findings are that TFRC is, in the topologies so

*This work has been partially supported by MIUR within the framework of the National Project FIRB-TANGO.

far investigated, successful in providing smooth rate transmissions and competes fairly for bandwidth with the different flavors of TCP flows. Actually, in the considered radio environment TFRC behaves quite *timidly* and suffers the aggressiveness of the TCP slow-start mechanism. Moreover, our results evidence the moderate values of throughput that saturate the 802.11 network.

The rest of the paper is organized as follows: Section 2 provides a brief description of the building blocks that the congestion control mechanism enforced by TFRC employs; Section 3 introduces the measured performance parameters; Section 4 reports the laboratory test bed and the simulation scenarios, as well as a discussion about the main results obtained. Section 5 summarizes the insigths that have been inferred from these results.

2 TFRC: an Overview

Let us start recalling that for multimedia real-time applications delay and jitter are the main performance parameters and TCP reveals not to be particularly suitable for them. The AIMD (Additive Increase Multiplicative Decrease) congestion control mechanism that TCP implements negatively impacts real-time applications, that usually rely on UDP (User Datagram Protocol) and/or RTP (Real Time Protocol). On the other hand, streaming data applications seldom incorporate rate adaptation mechanisms and consequently behave in an unfair manner with respect to TCP: it might indeed happen that, when congestion occurs, they steal all the available bandwidth to simultaneously active TCP flows, as these "obediently" back-off and reduce their rate to confine congestion, whereas non-TCP connections keep sending data at a significant rate. A host of TCP-friendly congestion control schemes have been so far proposed – and their behavior analyzed –, with the goal of forcing non-TCP traffic to share the available bandwidth with TCP connections as fairly as possible [1]. Among the unicast congestion control mechanisms, TFRC has recently become a proposed standard within the Internet Society [5]: it is an equation-based scheme, designed for applications that adjust their sending rate to the average long-term throughput of TCP. TFRC uses a slightly modified version of TCP Reno throughput equation [12], to describe TCP's sending rate as a function of the loss event rate p, the round trip time RTT and the packet size s, and accordingly determines the allowed sending rate of the controlled non-TCP source. In greater detail, the congestion control mechanism enforced by TFRC mandates that [5]:

- the receiver measures the loss event rate p, where the loss event is defined as one or more packets lost in a round trip time, and sends this information back to the source;
- the sender also uses these feedback messages to estimate the round trip time RTT;
- the loss event rate p and RTT are then inserted into the throughput equation, providing an acceptable rate;
- the sender adjusts its rate to match the calculated one.

The throughput equation TFRC employs is

$$X = \frac{s}{RTT\sqrt{2b\frac{p}{3}} + RTO(3\sqrt{3b\frac{p}{8}}p(1+32p^2)}\,, \tag{1}$$

X is the transmit rate in bytes/s, RTO is the TCP retransmission timeout value in seconds and b is the number of packets acknowledged by a single TCP acknowledgment. Last expression is further simplified by setting $RTO = 4 \cdot RTT$ and recommending $b = 1$, so that the throughput can be expressed as follows:

$$X = \frac{s}{RTT f(p)}\,, \tag{2}$$

with

$$f(p) = \sqrt{2\frac{p}{3}} + (12\sqrt{3\frac{p}{8}}p(1+32p^2))\,, \tag{3}$$

suitable for a table lookup [5].

Regarding the determination of the parameter s, the packet size, it is worth observing that its value is normally known to an application; if not, so that the packet size varies depending on the data, it is recommended to use an estimate of the mean packet size for s; finally, if the application modifies the packet size to perform congestion control, then TFRC with its inherent mechanism is deemed completely inappropriate.

In what follows, RTT and p determination, as prescribed by TFRC, will be briefly described. For RTT, its initial value is undefined until it is set as described below. The sender calculates a new round trip sample, RTT_{sample}, every time it receives a feedback packet from the data receiver and then updates the RTT estimate in the following manner:
If no feedback has been received before
$RTT = RTT_{sample}$
$Else$
$RTT = q \cdot RTT + (1-q) \cdot RTT_{sample}$,
where the recommended value for q depends on whether the sender modifies or not its sending rate,basing its decision on how the most recent sample of the RTT differs from its estimate of the long-term RTT [5]. In the former case, $q = 0.9$, in the latter, q should be close to or exactly zero. When the degree of statistical multiplexing in the network is low, it is however recommended that the sender modifies its instantaneous transmit rate.

Next, the estimate of the loss rate p. Being such calculation performed at the data receiver, we note that TFRC can be also classified as a receiver-based mechanism. As required by the TCP model, TFRC does not measure the packet drop rate, defined as the number of lost packets over the total number of transmitted packets; instead, it employs the loss event rate p. A stable and accurate measurement of

p is crucial for TFRC, as its behavior and fairness to TCP is significantly influenced by such parameter. Its calculation requires that all TFRC packets contain a sequence number, incremented by one for each packet that is sent. A method that recalls TCP's triple duplicate ACK's is devised, to distinguish lost from reordered packets: namely, the loss of a packet is detected by the arrival of three or more packets with a higher sequence number than the lost packet. The loss event rate is measured over a certain time interval, called the *loss history*: the *loss history* size currently specified is $n = 8$ loss intervals, deemed adequate to guarantee a good compromise between TFRC stability and speed in responding to changes in the level of congestion. In detail, the loss rate estimation is performed adopting the Average Loss Interval method, that computes the weighted average of n loss intervals, i.e., the weighted average of the number of packets between loss intervals, with equal weights on each of the most recent $n/2$ intervals. If we indicate by s_i the number of packets in the i-th most recent loss interval, and by s_0 the interval containing the packets that have arrived since the last loss, then the calculation of the average loss interval $E[s]$ requires [5]:

$$E[s] = max\left(\frac{\sum_{i=0}^{n-1} s_i w_i}{\sum_{i=0}^{n-1} w_i}, \frac{\sum_{i=1}^{n} s_i w_i}{\sum_{i=1}^{n} w_i}\right) \quad (4)$$

with the loss even rate, p, simply being

$$p = \frac{1}{E[s]}\,. \quad (5)$$

TFRC mechanism is even more sophisticated than it appears from the former overview and features more options, that we have intentionally omitted. The interested reader is referred to [5] for a comprehensive protocol description.

3 Performance Metrics

Let us start with the definition of the basic performance parameters we have employed. For the sake of simplicity, in perfect accordance with [11], we let $n_{f,\delta}(t)$ denote the number of packets of flow f during an interval δ that ends at time t. If the amount of time since the flow started is less than δ, then $n_{f,\delta}(t)$ is defined as the number of packets since the start time t_0:

$$n_{f,\delta}(t) = \begin{cases} packets\_sent\_in\_(t-\delta, t] & for\ t-\delta \geq t_0 \\ n_{f,\delta}(t-t_0) & for\ t-\delta < t_0 \end{cases} \quad (6)$$

Let then $B_{f,\delta(t)}$ be the throughput of flow f at time t with granularity δ and packet size s:

$$B_{f,\delta(t)} = \frac{n_{f,\delta}(t)\cdot s}{\delta} \quad (7)$$

and define the average throughput of flow f as the total amount of data being transmitted during the measurement interval $[t_0, T]$, i.e.,

$$B_f = \sum_{i=1}^{(t-t_0)/\delta} \frac{n_{f,\delta}(t_0+\delta i)\cdot s}{T} = B_{f,T-t_0}(T)\,. \quad (8)$$

In order to compare the throughput of flows using different protocols, let B_P be the average throughput of the set of flows that utilize protocol P:

$$B_P = \frac{\sum_{f\in P} B_f}{|P|} \quad (9)$$

being $|P|$ the number of such flows.

Finally, be $\mathcal{P}$ be the set of all protocols of an experiment. To study the behavior of flows belonging to protocol $P \subseteq \mathcal{P}$, the *inter-protocol fairness* index is introduced:

$$F_P^{inter} = \frac{B_{\mathcal{P}-\{P\}}}{B_{\mathcal{P}}} \quad (10)$$

where $B_{\mathcal{P}-\{P\}}$ represents the average throughput experienced by flows competing with P flows and $B_{\mathcal{P}}$ the average throughput of all flows in $\mathcal{P}$; $F_P^{inter} = 0.5$ means ideal fairness, i.e., all flows experience the same average throughput whatever protocol they belong to. Values greater than 0.5 indicate that protocol P is too conservative and its flows exploit less bandwidth than that they should. On the contrary, values lower than 0.5 reflect an aggressive behavior of protocol P.

When protocol P only is examined, we choose to assess its *intra-protocol fairness* determining the corresponding max-min fairness index, defined in [13] as

$$F_P^{max-min} = \frac{min_{f\in P} B_f}{max_{f\in P} B_f}. \quad (11)$$

The corresponding values lie in the $[0, 1]$ range: the closer to 1, the more equally distributed the bandwidth is.

4 Numerical Results

The results presented in this Section refer to what is here called *uplink* configuration. This means that wireless nodes are always the senders while the receivers may be either wireless or wired.

4.1 Laboratory Test Bed

TFRC behavior has been previously investigated in several contexts, showing good friendliness and trade-off between stability and reactivity [11]. TFRC is here tested in a radio environment, namely, over a wireless IEEE 802.11b LAN, with the aim of highlighting the influence that the radio channel has on its performance. The test-bed LAN is reported in Figure 1. PCs are equipped with standard IEEE 802.11b

cards, operating at 11 Mbit/s in the $(2.4 - 2.4835)$ GHz range, connected through an access point located within the same room. The two PCs are Pentium IV desktop at 1.7 Ghz and Pentium IIIm laptop at 1.2 Ghz, with 256 MB RAM. Both PCs are equipped with Suse Linux 8.0 operating system. Incoming and outgoing datagrams are monitored with *tcpdump*.

We next present some of the results referring to the network described above. We have conducted several tests with different combinations of TCP/TFRC flows. For each combination we have performed 8 experiments, 3 minutes long each. The average throughput values have been computed excluding the first 60 seconds data, in order not to affect the comparison by different initial behaviors, e.g., slow start.

Figure 2 reports the inter-protocol fairness index for different combinations of TCP Reno and TFRC flows. Note that equally increasing the number of flows the index gets close to 0.6, thus revealing not only TFRC friendliness, but also its conservative behavior. As a matter of fact, time outs for TCP Reno keep increasing so that it reacts with slow start which is, actually, a quite aggressive behavior that eventually penalizes TFRC flows. In this WLAN scenario, the radio channel is one of the main reasons for the so many time out events which make the slow start phase last ever longer then allowing a TFRC flow perceive low loss rate and RTT values which, to its turn, make the transmission rate increase. This is shown in the $10/1$ case. Thus, compared to test in wired LANs a greater number of time-outs make TFRC get less and less bandwidth then increasing the fairness up to 0.6.

Figure 3 shows the inter-protocol fairness index for different combinations of TCP Sack and TFRC flows. When TCP Sack is employed, time-out events are much less frequent than with TCP Reno; on the other hand, the number of triple duplicate ACKs is much higher. This translates into a more aggressive behavior of TCP Sack towards TFRC. The very first effect is the increasing of RTT which, together with higher values of loss, makes TFRC evaluate an available bandwidth lower than the fair one. This is confirmed in this Figure by fairness values almost always bigger than 0.6. When the number of flows is very high, i.e. $10/10$, time out events limit the aggressiveness of TCP Sack, thus improving the fairness.

Previously an intra-fairness index has been defined (11). Tests have been done with $2, 4, 10$ and 20 TFRC flows and, separately, with TCP Reno. Each test has been repeated 8 times. Figure 4 reports the max-min fairness as a function of the number of simultaneous flows. With the increase of the number of flows TFRC reveals to be more fair than TCP. When the overall load increases the model on which TFRC is based can predict quite well the actual throughput so that the bandwidth can be equally shared; on the other hand, TCP is very reactive to losses and time outs so that if a flow has many time outs in a row, very likely in radio environment, TCP drastically reduces the throughput and it can take some time to regain a fair value.

Figure 5 reports the throughput of one TFRC flow and three simultaneous TCP Reno connections that saturate the wireless channel during the 3 minutes test. As expected, the TFRC flow achieves quite a constant throughput if compared to TCP. It is also worth highlighting that the average throughput values of each TCP flow is equal to 67.06 Kbyte/s, while TFRC experiences a thorughput of 53.48 Kbyte/s. Roughly, 2 Mbit/s ($67 \times 3 \times 8 + 53 \times 8$) is the actual overall throughput, out of the nominal 11 Mbit/s, available to applications of the wireless LAN.

Finally, Figure 6 reports the throughput of one TFRC flow and three TCP Sack flows during the 3 minutes test. As noted before, the TFRC flow now behaves more timidly, with TCP Sack exhibiting a smoother behavior with respect to TCP Reno. This means that TFRC perceives an almost always congested channel which never allows TFRC to increase the rate.

Radio channel impacts the performance by causing an increase of time out events which make TCP Reno show large oscillations which, however, let TFRC to exploit the bandwidth during the narrowing of TCP Reno window. When TCP Sack is used, on the other hand, oscillations are smoother because time out events are much less but this make it much more aggressive so that TFRC gets even less bandwidth.

4.2 Simulation

TFRC friendliness has been investigated by means of simulation as well. In particular the *ns* simulator has been employed to analyze several wireless-wired scenarios. Figure 7 refers to a multi-hop topology where the wireless nodes are the senders and the receivers are wired. The simulation settings are as follows:

- five sources ($S0$-$S4$);
- five receivers ($R0$-$R4$);
- four intermediate nodes ($BS0$, $H0$-$H2$) with buffer size equal to 20 packets and drop tail policy;
- all links, wired and wireless, are 1 Mbit/s and delay equal to 10 ms;
- sources are CBR and generate 500 bytes packets;
- $S0$ employs TCP NewReno, $S1$ Westwood, $S2$ Sack, $S3$ and $S4$ TFRC;
- source $S4$ transmits with a 35 s delay;

Figure 8 shows the average throughput obtained by each flow as a function of time. It is shown that NewReno and Sack sources at the beginning get more bandwidth than the others; then, in particular when $S4$ starts transmitting, Sack

reduces its amount of bandwidth while NewReno keeps to gain more than the others. TFRC provides a pretty stable data pipe to $S3$ and it seems to be fair towards TCP flows; as a matter of fact, it gets roughly the same bandwidth as Westwood and Sack but half the bandwidth obtained by NewReno.

Figure 9 reports the second multi-hop topology where the bottleneck has been placed in $H0$ by increasing the bandwidth of the link between BS and $H0$ to 4 Mbit/s. Figure 10 shows the related average throughput obtained by each flow as a function of time. In this case, TCP Sack seems to be the most penalized while TFRC provides again a stable and fair data connection to $S3$.

Our simulations seem to confirm the friendliness of TFRC in wireless scenarios as well. However, many more topologies have still to be investigated to draw a final conclusion. In particular, the downlink scenario where the receivers, instead of the senders, are always wireless is current work in progress and it has to be done in order to complete the study of end-to-end control over hybrid wireless and wired networks.

5 Conclusions

In this paper, TFRC has been evaluated together with TCP Reno and TCP Sack over a wireless IEEE 802.11b LAN. Friendliness, fairness and throughput have been the main performance parameters in the framework of an indoor radio channel. Also, TFRC has been investigated through simulation in competition with other TCP flavors such as NewReno, Sack and Westwood in heterogeneous wireless-wired multi-hop topologies. In particular, here we have shown the uplink scenario, where the senders are always wireless sources.

We find that TFRC is successful in providing stable rate transmissions while being friendly with respect to both TCP Reno, TCP Sack and TCP Westwood. Actually, on the considered radio environment it shows to be "too" friendly which means it suffers the aggressiveness of all TCP versions, in particular when large TCP buffers are set.

Radio channel, compared to cables, causes many time out events which eventually are "normal"events in this environment basically due to frequent degradations of the channel rather than to congestions. However, both TCP Reno and TFRC decrease their rate but TFRC is far slower to regain higher rate thus being penalized as the inter-protocol fairness index discussion as underlined. TCP Sack can quite effectively neutralize the time out events but remarkably penalizes TFRC which now always perceives a congested channel.

In conclusion, TFRC has also in the investigated WLAN scenario a TCP friendly behavior and often does not even manage to exploit its fair amount of bandwidth.

References

[1] J. Widmer, R. Denda, M. Mauve, "A Survey of TCP-Friendly Congestion Control," IEEE Network, Vol.15, no.3, May/June 2001, pp.28-37.

[2] S. Floyd, K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet", *IEEE/ACM Transactions on Networking*, vol.7, no.4, pp. 458-472, August 1999.

[3] J. Widmer, "Equation-Based Congestion Control", Diploma Thesis, February 2000, University of Manheim, http://www.icir.org/tfrc/.

[4] S. Floyd, M. Handley, J. Padhye, J. Widmer, "Equation-Based Congestion Control for Unicast Applications", in Proc. of ACM SIGCOMM 2000.

[5] M. Handley, S. Floyd, J. Padhye, J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", IETF RFC 3448, *ftp://ftp.rfc-editor.org/in-notes/rfc3448.txt*.

[6] S. Pilosof, R. Ramjee, D. Raz, Y. Shavitt, P. Sinha, "Understanding TCP Fairness over Wireless LAN", Proc. of IEEE INFOCOM 2003.

[7] M. Heusse, F. Rousseau, G. Berger-Sabbatel, A. Duda, "Performance Anomaly of 802.11b", Proc. of IEEE INFOCOM 2003, March 30 - April 3, 2003, San Francisco, California, USA.

[8] J. Tian, S. Xiangzhi, W. Wenjun, "The effect on the inter-fairness of TCP and TFRC by the phase of TCP traffics", in Proc. of the 2001 International Conference on Computer Networks and Mobile Computing, 16-19 Oct. 2001, pp. 131 -136.

[9] Ekstrom, H.; Ludwig, R."Queue management for TFRC-based traffic in 3G networks", in Proc. of the 23^{rd} International Conference on Distributed Computing Systems Workshops, 2003, pp.870-876.

[10] M. Vojnovic, J.-Y. Le Boudec, "On the Long-Run Behavior of Equation-Based Rate Control", in Proc. of SIGCOMM'02, August 19-23, 2002, Pittsburgh, Pennsylvania, USA.

[11] J. Widmer, "Equation-Based Congestion Control", Diploma Thesis, February 2000, University of Manheim, http://www.icir.org/tfrc/.

[12] J. Padhye, V. Firoiu, D. F. Towsley, J. F. Kurose, "Modeling TCP Reno Performance: A Simple Model and its Empirical Validation", *IEEE/ACM Transactions on Networking*, vol.8, no.2, pp.133-145, April 2000.

[13] D. Bertsekas, R. Gallager, "Data Networks", 2nd Ed., Prentice Hall.

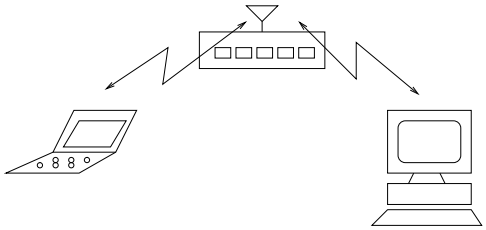

Figure 1: The reference IEEE 802.11b network.

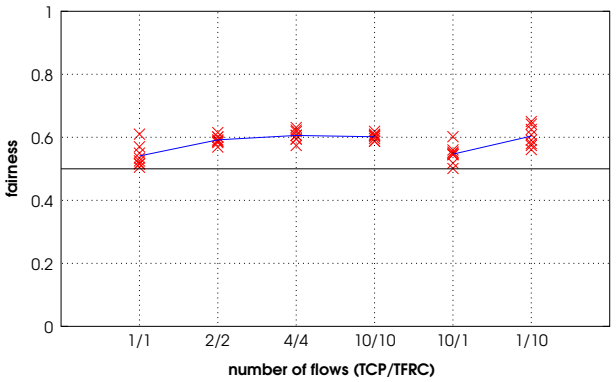

Figure 2: Inter-protocol fairness index for different combinations of TCP Reno and TFRC flows.

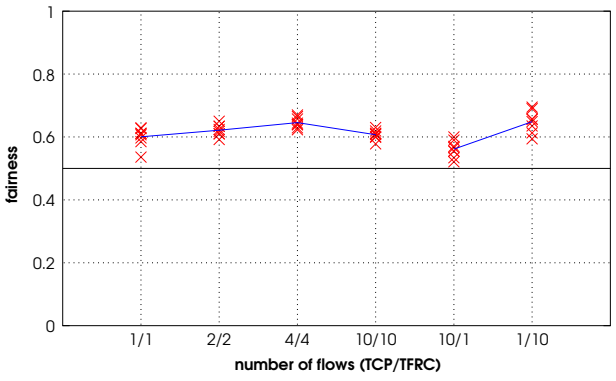

Figure 3: Inter-protocol fairness index for different combinations of TCP Sack and TFRC flows.

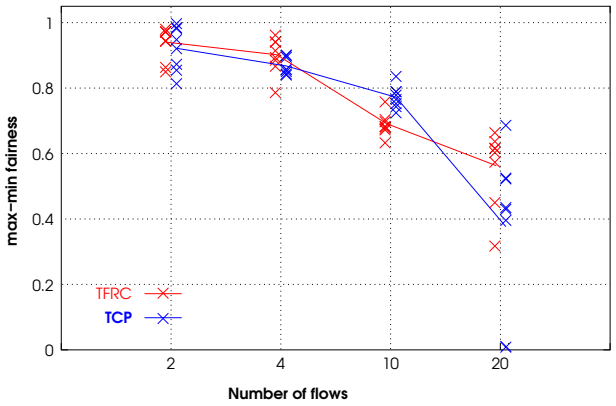

Figure 4: Max-min fairness index as a function of the number of simultaneous TCP Reno and TFRC flows.

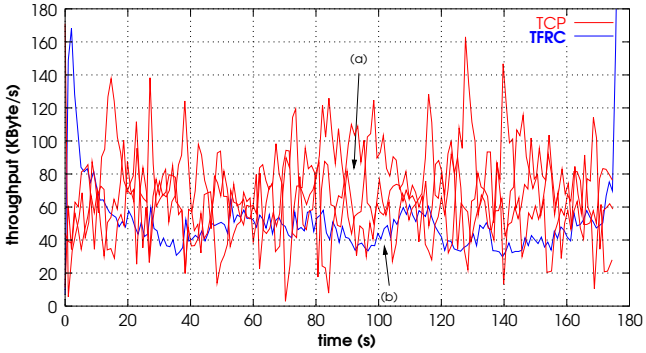

Figure 5: Throughput of three TCP Reno flows and one TFRC during a 3 minutes test.

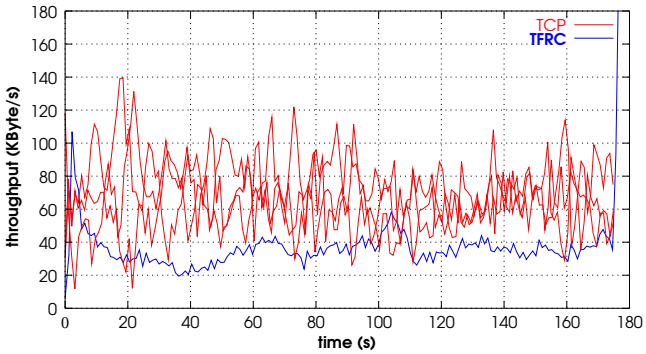

Figure 6: Throughput of three TCP Sack flows and one TFRC during a 3 minutes test.

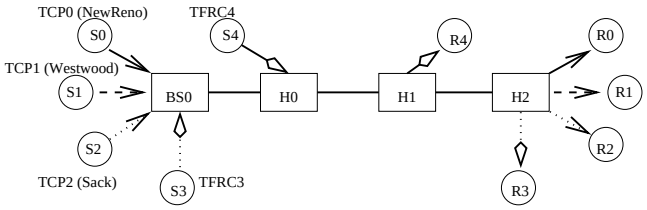

Figure 7: Wireless to wired: scenario n.1.

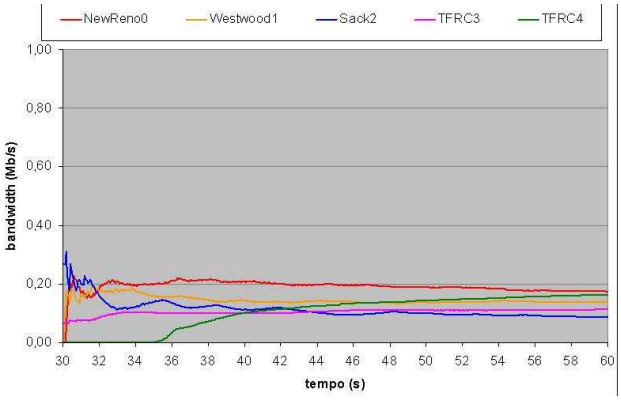

Figure 8: Average throughput of five flows: NewReno, Sack, Westwood and TFRC. TFRC4 starts with a 35 s delay.

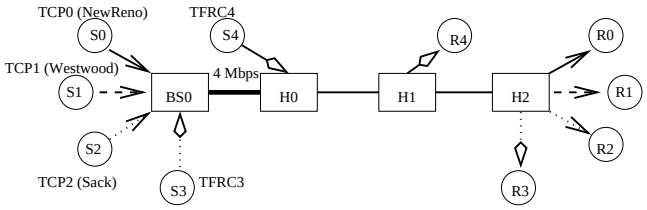

Figure 9: Wireless to wired: scenario n.2.

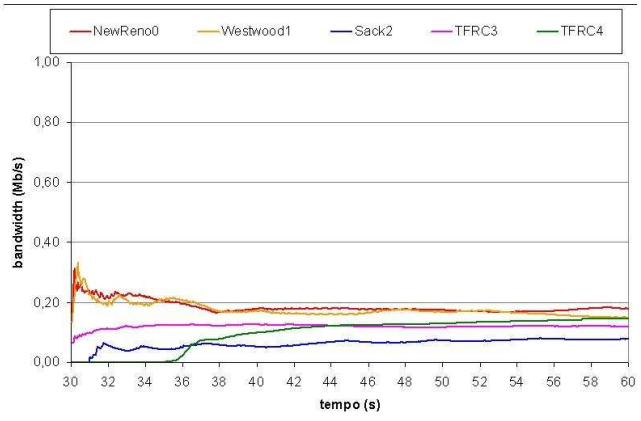

Figure 10: Average throughput of five flows: NewReno, Sack, Westwood and TFRC. TFRC4 starts with a 35 s delay. The link between the nodes BS and H0 is 4 Mbit/s.