Intraprotocol fairness and interprotocol friendliness of TFRC congestion control algorithm

L.A. Grieco and S. Mascolo

The TCP-friendly rate control (TFRC) algorithm has been proposed for supporting applications such as video streaming or telephony over the Internet, where a relative smooth sending rate is of importance. The fairness and friendliness of TFRC over scenarios with multiple congested routers is evaluated. Results obtained using the *ns*-2 simulator show that while TFRC improves network utilisation with respect to Reno TCP, it exhibits a significant degree of unfairness and unfriendliness towards Reno TCP.

Introduction: The TCP-friendly rate control (TFRC) algorithm is currently considered by the Internet Engineering Task Force (IETF) as an alternative to classic Reno TCP [1] for applications, such as video streaming or telephony, where a relative smoother sending rate is of importance [2–4]. TFRC aims at providing a smooth sending rate, with respect to TCP, while preserving a friendly behaviour towards TCP, which is the protocol that dominates the Internet. To achieve these requirements, the basic idea of TFRC is to set the transmission rate as dictated by the model of the long-term throughput of a Reno TCP source [5].

Mandatory requirements of any new congestion control protocol are interprotocol friendliness and intraprotocol fairness. The use of the long-term equation provides a slow-responsive behaviour which is in fact one of the goals for which TFRC was designed. However the slow responsive behaviour of the TFRC algorithm could not provide fairness (friendliness), since a new joining TFRC (Reno TCP) flow could experience starvation when competing with slow responsive rate based flows, such as TFRC flows.

Our aim in this Letter is to investigate the relevant properties of fairness and friendliness of the TFRC algorithm using the *ns*-2 simulator [6]. For that purpose, a network topology with multiple congested routers, in the presence of both homogeneous and heterogeneous traffic mixes, has been considered. Collected results show that TFRC is friendly and fair towards Reno TCP in a single bottleneck scenario, whereas it exhibits intraprotocol unfairness and interprotocol unfriendliness towards Reno TCP in a multi-bottleneck scenario.

Fairness and friendliness of TFRC algorithm: To investigate the TFRC algorithm in a realistic scenario, we consider the multihop topology depicted in Fig. 1. It is characterised by: (a) N hops; (b) one persistent connection C_1 going through all the N hops; (c) $2N$ persistent sources $C_2, C_3, C_4, \ldots, C_{2N+1}$ of cross traffic transmitting data over every single hop.

The simulation lasts 1000 s during which the cross traffic sources always send data. The connection C_1 starts data transmission at time $t = 10$ s when all network bandwidth has been grabbed by the cross traffic sources that have started at $t = 0$ s.

The behaviour of TFRC and Reno TCP has been evaluated using different buffer sizes; in particular, by considering a typical $RTT =$ 150 ms, buffer sizes of 12 packets, which corresponds to the bandwidth delay product, and 24 packets have been used.

The following three scenarios have been considered [7]:

Scenario 1. All traffic sources are controlled by the same control algorithm. We have measured the goodput of the C_1 connection and the overall network utilisation, which has been computed as the total goodput over the link capacity (i.e. 1 Mbit/s), where the total goodput is defined as the goodput of the C_1 connection plus the average goodput of the $(C_2, C_4, \ldots, C_{2N})$ connections. For both the considered buffer sizes, TFRC provides full utilisation, whereas Reno achieves a utilisation that ranges from 80 to 68%, when N increases from 1 to 10. Fig. 2 shows that, when the traffic sources are controlled by Reno TCP, the goodput achieved by the C_1 connection is not sensitive to the queue size and is much larger than the goodput achieved when the traffic sources are controlled by TFRC. In particular, for $N \geq 5$, TFRC provides a goodput that is more than one order of magnitude smaller than those provided by Reno TCP. This behaviour is due to the TFRC cross traffic, which get almost all the network bandwidth so that the C_1 source is not able to get its share of bandwidth when the number of hops it goes through is greater than 5. Moreover, for $N \geq 5$, a larger buffer size does not mitigate the TFRC unfairness. Finally it should be noted from Fig. 2 that, when $N = 1$, i.e. the scenario reduces to a single bottleneck topology, TFRC is fair. In fact the TFRC goodput is close to the fairshare, which is the grey curve at 0.5 Mbit/s reported in Fig. 2. This result confirms what has been also reported in [3].

Scenario 2. The $C_2, C_3, C_4, \ldots, C_{2N+1}$ sources of cross traffic are controlled by Reno TCP whereas the C_1 connection is controlled by TFRC. This scenario allows us to investigate the friendliness of Reno TCP towards TFRC. Fig. 3 shows that the C_1 source controlled by TFRC achieves a goodput that is slightly larger than the goodput achieved by the C_1 Reno connection in the homogeneous scenario (see Fig. 2). This means that the Reno TCP cross traffic sources allow the joining C_1 connection to get an acceptable share of the network capacity. The network utilisation monotonically decreases from 96 to 72% when N increases from 1 to 10.

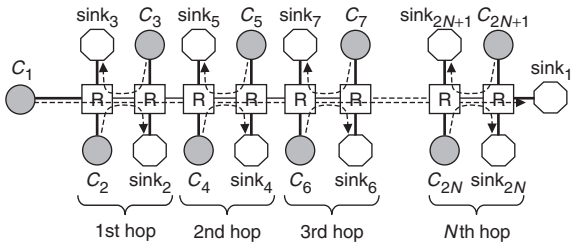

Fig. 1 *Multihop scenario*

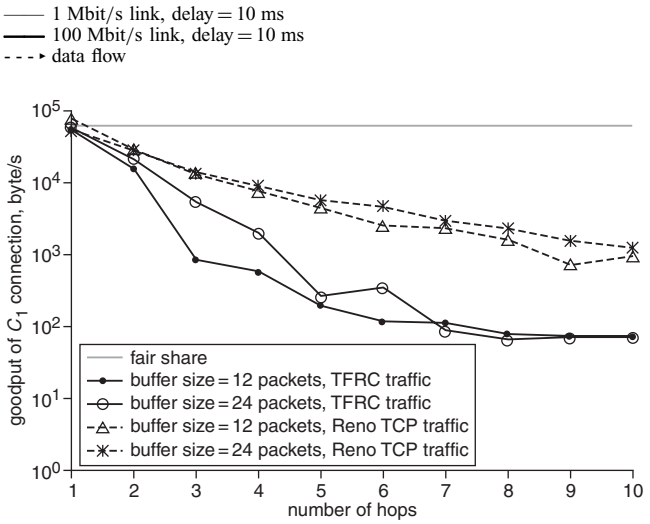

Fig. 2 *Goodput of C_1 connection in homogeneous N-hop scenario with N varying from 1 to 10*

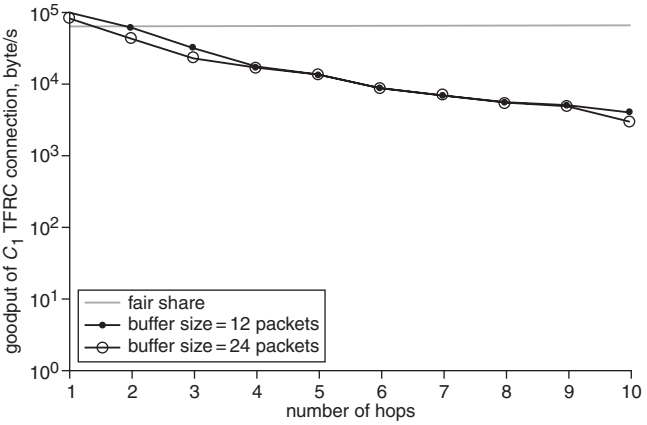

Fig. 3 *Goodput of C_1 connection controlled by TFRC in N-hop scenario with Reno cross traffic*

Scenario 3. The cross traffic sources $C_2, C_3, C_4, \ldots, C_{2N+1}$ are TFRC, whereas the C_1 connection is Reno. This is the most significant scenario, since it investigates the interprotocol friendliness of TFRC towards Reno TCP. Fig. 4 shows that the C_1 Reno connection achieves a

very poor goodput in the presence of TFRC cross traffic, i.e. TFRC reveals not to be friendly towards Reno in a multiple bottleneck scenario. This result sets a warning on the wide deployment of TFRC. The network utilisations measured in this scenario are similar to those obtained in Scenario 1.

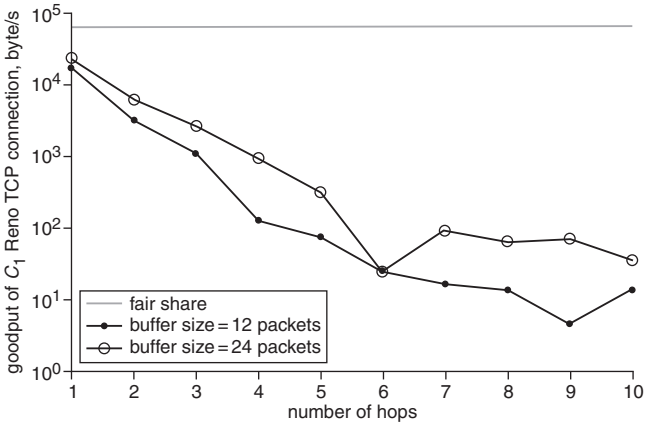

Fig. 4 *Goodput of C_1 connection controlled by Reno TCP in N-hop scenario with TFRC cross traffic*

Conclusion: Intraprotocol fairness and interprotocol friendliness of TFRC in network scenarios with multiple congested routers have been investigated. Results indicate that TFRC improves network utilisation with respect to Reno TCP, at the expense of a high degree of unfairness and unfriendliness towards Reno TCP in multi-bottleneck scenarios.

4 December 2003
Electronics Letters online no: 20040224
doi: 10.1049/el:20040224

L.A. Grieco and S. Mascolo (*Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, Via Orabona, Bari 4 – 70125, Italy*)

E-mail: mascolo@poliba.it

References

1 Allman, M., Paxson, V., and Stevens, W.R.: TCP congestion control. RFC 2581, April 1999
2 Floyd, S., Handley, M., Padhye, J., and Widmer, J.: 'Equation-based congestion control for unicast application'. ACM SIGCOMM 2000, Stockholm, Sweden, August 2000, pp. 43–56
3 Bansal, D., Balakrishnan, H., Floyd, S., and Shenker, S.: 'Dynamic behavior of slowly-responsive congestion control algorithms'. ACM SIGCOMM 2001, UC San Diego, CA, USA, August 2001, pp. 263–273
4 Handley, M., Floyd, S., Padhye, J., and Widmer, J.: TCP Friendly Rate Control (TFRC): Protocol specification. RFC 3448, January 2003
5 Padhye, J., Firoiu, V., Towsley, D., and Kurose, J.: 'Modeling TCP throughput: a simple model and its empirical validation'. ACM Sigcomm'98, Vancouver, BC, Canada, 1998, pp. 303–314
6 Ns-2. network simulator 2002
7 Tcl scripts of the simulated scenarios are available at http://www-ictserv.poliba.it/grieco/scripts.htm