A Fair TCP-Based Congestion Avoidance Approach for One-to-Many Private Networks

Tarik Taleb[1,*], Hiroki Nishiyama[1,†], Abbas Jamalipour[2], Nei Kato[1,†], and Yoshiaki Nemoto[1,*]
[1] Graduate School of Information Sciences
Tohoku University, Japan
[*]{taleb, nemoto}@nemoto.ecei.tohoku.ac.jp
[†]{bigtree, kato}@it.ecei.tohoku.ac.jp
[2] School of Electrical & Information Engineering
University of Sydney, Australia
a.jamalipour@ieee.org

***Abstract*— Over the past few years, a number of private networks have emerged. In these private networks, a server provides its subscribed clients with Internet services, forming a one-to-many network topology. Given the fact that users are located at different distances from the server, usage of the Transmission Control Protocol (TCP) for communication results in drastically unfair bandwidth allocations among the users. In this regard, this paper addresses the fairness and efficiency issues of TCP in such one-to-many IP (Internet Protocol) networks. The efficiency of TCP is controlled by matching the aggregate traffic rate of all TCP connections to the sum of the link capacity and total buffer size. On the other hand, its unfairness issue is mitigated by allocating bandwidth among individual flows in relative proportion with their RTTs. Simulation results elucidate that the proposed method makes better utilization of the network resources, reduces the number of packet drops, and provides a fair service to users.**

I. Introduction

Along with the wide spread of the Internet, a potential number of new technologies have emerged. Asymmetric Digital Subscriber Line (ADSL) and Fiber To The Home (FTTH) are few examples. Based on these technologies, a number of private networks have been constructed. Such networks are awaiting a promising market in the wireless area as well given the recent advances in wireless technologies, such as Universal Mobile Telecommunications System (UMTS) and Wireless LAN (WLAN). In each private network, a set of identical servers provides a library of applications to a large population of users, forming a one-to-many network topology. The Transmission Control Protocol (TCP) is the backbone of most of these networks. TCP usually results in drastically unfair bandwidth allocations when multiple connections share a bottleneck link. This unfairness issue becomes more substantial when flows have different Round Trip Times (RTTs). Indeed, when a group of users, with considerably different RTTs, compete for the same link capacity, TCP exhibits undesirable bias against long RTT flows. As the window size is increased for each acknowledgment (ACK) packet, connections with shorter RTTs receive ACKs at a faster rate resulting in a larger window for shorter connections.

As an attempt to solve the unfairness issue of TCP, this paper proposes a scheme that allows TCP senders to send data at rates relatively proportional to their RTT values. To efficiently utilize network resources, the aggregate window sizes of all active TCP flows is set to the effective network bandwidth-delay product.

The remainder of this paper is structured as follows. Section II highlights some research works in the context of improving the performance of TCP in IP networks. The proposed scheme is described in Section III. Section IV portrays the simulation environment and discusses the simulation results. Concluding remarks are in Section V.

II. Related Work

Despite the widespread acceptance of TCP, its performance is still limited due to many factors. To improve the performance of TCP, several algorithms have been proposed. Traditional TCP-based algorithms infer the congestion state of a network from implicit signaling such as arrival of ACKs, timeouts, and receipt of duplicate ACKs. Other type of algorithms requires modifying network elements and adding new fields in IP packets. Other schemes use changes in network performance, such as increase in RTT or changes in throughput, to detect congestion.

Among protocols that detect congestion from comparison between measured throughput rates and estimated ones, TCP Westwood [1] is the most notable example. TCP Westwood is a sender-side modification of the TCP congestion window algorithm. It improves upon the performance of TCP Reno in wired as well as wireless networks. The key concept of TCP Westwood is to use an estimate of the available bandwidth to set the congestion window and slow start threshold after a congestion episode. A TCP Westwood source performs end-to-end estimate of the bandwidth available along a TCP connection by measuring and averaging the rate of returning ACKs. Whenever the sender perceives a packet loss, inferred by a timeout occurrence or reception of a certain number of duplicate ACKs, the sender uses the bandwidth estimate, an approximate of the effective bandwidth at the congestion time, to select the optimum values of the congestion window and the slow start threshold. By so doing, TCP Westwood ensures faster recovery, efficient utilization of network resources, and eventually an acceptable control of network congestion.

In [2], Balakrishnan et. al. envisioned a case where two different communication scenarios coexisted; one involving a single connection between two end-terminals and another simulating a user having multiple connections at the same time. To cope with the unfairness issue of TCP in such scenario, TCP-INT is proposed. The basic concept behind TCP-INT consists in integrating the multiple connections of

1-4244-0355-3/06/$20.00 (c) 2006 IEEE

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.

a single user into a virtual single congestion window. By so doing, the TCP-INT scheme guarantees a fair allocation of the network resources among all competing users. In [3], a network topology where users have different possible paths for communication is considered. In this multi-path environment, the most appropriate path is selected for transmitting data packets. This selection is based on an estimate of path bandwidth and the minimum measured RTT. While this selection helps to alleviate network congestion, it becomes inefficient when the selected paths share the same bottleneck link.

Network congestion can be controlled also by employing scheduling mechanisms, fair queuing, and intelligent packet-discard policies such as Random Early Marking (REM) [4], Random Early Discard (RED) [5] combined with Explicit Congestion Notification (ECN) [6], BLUE [7], Stabilized RED (SRED) [8], and Fast Adaptive Fuzzy Controller(FAFC) [9]. These Active Queue Management (AQM) schemes require a packet loss to signal early network congestion to TCP sources. The main drawback of these policies is that they drop packets from many connections and cause them to decrease their windows at the same time resulting in a significant degradation of throughput. Furthermore, these AQM schemes are inefficient in fairly allocating bandwidth among multiple TCP connections sharing the bandwidth of the same link and/or the buffer of the same network element. This unfairness issue becomes further more significant in the case of TCP connections with highly varied RTTs. This is attributable to the fact that the packet discarding probability in most AQM approaches does not take into account flows RTT.

While a number of improvements have been made to the TCP implementation at the end-terminals, the greatest improvement to congestion control are achieved by adding new mechanisms to network elements (e.g. routers, gateways) to complement the endpoint congestion avoidance policy. These mechanisms should allow network elements between a TCP source and a TCP destination to acknowledge the source with its optimal sending rate. By so doing, the whole system becomes self-adaptive to traffic demands and more active in controlling congestion and buffer overflows.

To cope with TCP limitations in high bandwidth-delay product networks, several studies have been conducted providing valuable insight into TCP dynamics in such environments. Katabi *et al.* [10] proposed a new congestion control scheme, eXplicit Control Protocol (XCP). The scheme substantially outperforms TCP in terms of efficiency in high bandwidth-delay product environments. However, the main drawback of this protocol is that it assumes a pure XCP network and requires significant modifications at the end-system. The likelihood of XCP to be implemented is low due to the problem related to converting existing equipments. Explicit Window Adaptation (EWA) [11] and WINdow TRAcking and Computation (WINTRAC) [12] suggest an explicit congestion control scheme of the window size of TCP connections as a function of the free buffer value. While the two schemes make efficient use of network resources, they remain grossly unfair towards connections with high variance in their RTTs distribution.

To cope with both the fairness and efficiency issues of TCP, the authors have recently proposed a scheme, dubbed Recursive, Explicit, and Fair Window Adjustment (REFWA) [13]. The scheme achieves high efficiency by matching the sum of window sizes of all active TCP connections sharing a bottleneck link to the effective bandwidth-delay product of the network. The system fairness is improved by assigning for each flow a feedback proportional to its RTT. The feedback value is the optimum window size a TCP sender should be sending data at so as not to overload the network with packets. The REFWA scheme is specifically designed for broadband satellite networks. In REFWA, information on flows RTT is handled by a simple monitoring of hops count in the backward and forward traffic of each flow. Hops count of each flow is computed from Time to Live (TTL) field in the IP header of both ACK and data packets. Given the fact that prior knowledge of RTTs is usually not available at network elements in terrestrial networks, the main objective of the research work outlined in this paper is the implementation of REFWA in terrestrial networks, particularly in one-to-many private networks, to guarantee both an efficient and fair utilization of network resources. The proposed scheme is referred to as Terrestrial-REFWA (T-REFWA).

III. Basic Operations of T-REFWA

In current TCP implementations, RTT is computed to make an estimate of the Retransmission Time Out (RTO). The average value of RTT is noted as Smoothed RTT (SRTT). In T-REFWA, senders write down the value of SRTT in the Type Of Service (TOS) field of IP headers and transmit it to routers. If T-REFWA is to be used along with some Differentiated Services (DiffServ) schemes that use the TOS field, another possible alternative to the latter is the option field of the IP packet header. T-REFWA needs only six bits of the eight bits in the TOS field. The remaining two bits can be used for other schemes, such as ECN [6]. Given the required six bits, the value of SRTT should be an integer value from within the range [0:63]. It is thus transformed in a discrete manner to an integer value, referred to as RTT Code (RTTC) throughout this paper, within the range $[0:63]$ as shown in the following function:

$$RTTC = \min\big(\lfloor K \cdot \log_2(SRTT) \rfloor, 63\big) \qquad K = 6.3 \qquad (1)$$

It should be noted that as most of Internet connections have RTTs smaller than one second [14][15], K is set to 6.3 so that when $(RTT = 2^{10} = 1024\text{ms})$, $RTTC$ will be equal to 63. It should be emphasized that errors in transforming the original SRTT may yield unfair bandwidth allocation among competing TCP flows. To minimize such impact, most importantly over any flows RTT distribution, we use the Logarithmic function as the transformation function. On the other hand, the average RTT value of flows with a given RTTC equal to α is computed as follows:

$$RTT_\alpha = \frac{2^{\alpha/K} + 2^{(\alpha+1)/K}}{2} \qquad K = 6.3 \qquad (2)$$

TABLE I
FLOWS GROUP STATE TABLE

RTTC	Flow Count	Flow ID	Last Packet Transmission Time
0	n_0	0.1 : $0.n_0$	$t_{0.1}$: $t_{0.n_0}$
1	n_1	1.1 : $1.n_1$	$t_{1.1}$: $t_{1.n_1}$
:	:	:	:
63	n_{63}	63.1 : $63.n_{63}$	$t_{63.1}$: $t_{63.n_{63}}$

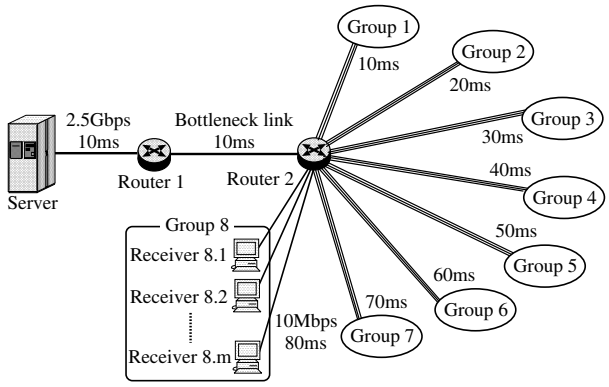


Fig. 1. A simple network topology with connections having different RTTs.

It should be admitted that the implementation of T-REFWA requires some modification at the sender side for the computation of RTTC. To reduce the overhead that may be incurred because of this operation, we consider the application of T-REFWA to only "one-to-many" networks. In such networks, T-REFWA will be implemented at only servers and the required overhead will be negligible if compared to when T-REFWA is implemented at all senders. Another reason behind the choice of one-to-many networks consists also in the fact that fairness among users is important in such networks. It should be also stressed out that the RTTC computation is not performed on a periodic basis or per packets, but only when the sender observes a change in the SRTT value. As verified by simulation results, T-REFWA senders experience almost no packet drops and their SRTT values remain relatively stable. The load due to the computation of RTTC is accordingly minimal.

At routers, flows are grouped according to their RTTC values, as shown in Table I. Each group of flows is identified by a RTTC. Note that the maximum size of the table is 63. Scalability is thus not an issue. At each router, a group G is defined as the set of flows that have the same RTTC value. Flows are identified also by a flow ID and are defined as streams of packets sharing the quintuple: source and destination addresses, source and destination port numbers, and protocol field. A flow is considered to be in progress if the time elapsed since its last packet transmission time is inferior than the most recent estimate of the average RTT of all active flows traversing the router, RTT_{avg}.

Similarly to the original REFWA scheme, the feedback computation is performed periodically every RTT_{avg} time interval. The feedback computation load is thus not so heavy. At time $(t = n \cdot RTT_{avg})$, the feedback value of flows that have a given RTTC$=\alpha$, $F_\alpha(n)$, is computed as in Equation 3 (shown at the bottom of the page). In Equation 3, B and Bw are the router's buffer size and the link bandwidth, respectively. n_j denotes the size of the group of flows with RTT Code equal to j and RTT_j denotes the RTT value of its flows. $\Upsilon(n)$ and $Q(n)$ denote the aggregate TCP window size and the router's queue occupancy at time $(t = n \cdot RTT_{avg})$. ϕ and ψ are constant parameters. It should be recognized that ϕ and ψ play a significant role in exploiting well the system spare bandwidth and free buffer size, respectively. In other words, in case of optimum values of ϕ and ψ, when some connections are not making full use of their allocated bandwidths during the interval time $[(n-1)RTT_{avg}, nRTT_{avg}]$, the spare bandwidth, $(Bw \cdot RTT_{avg} - \Upsilon(n-1))$, will increase and the buffer occupancy, $Q(n-1)$, will go down causing an increase in the computed feedbacks of the other connections at time $(t = n \cdot RTT_{avg})$. This will help to fully utilize the link capacity while maintaining small buffer sizes. Using the same stability analysis as in [16], ϕ and ψ can be set to 1.5 and 0.5 so the system will be always stable regardless of the flows count and their RTTs. Stability refers to the fact that under any conditions (e.g. any number of flows and any distribution of their RTTs), the system should always converge to its point of equilibrium; transmission of the highest number of packets without causing network congestion.

The window feedback is computed every RTT_{avg} time and is written in the receiver's advertised window (RWND) field carried by the TCP header of acknowledgment packets. The RWND value can only be downgraded. If the original value of the receiver's advertised window, which is set by the TCP receiver, exceeds the feedback value computed by a downstream node, the receiver's advertised window is reduced then to the computed feedback value. A more congested router later in the path can further mark down the feedback by overwriting the receiver's advertised window field. Ultimately, the RWND field will contain the optimal feedback from the bottleneck along the path. Therefore, given a specific router where the traffic concentrates at all time, we can get enough effects along with less implementation load by setting the proposed method at only that router. When the feedback

$$
\begin{aligned}
F_\alpha(n) &= \frac{RTT_\alpha}{\sum_{j=0}^{63} n_j \cdot RTT_j} \cdot \Upsilon(n) \qquad (3)\\
\Upsilon(n) &= \Upsilon(n-1) + \phi\Big(Bw \cdot RTT_{avg} - \Upsilon(n-1)\Big) + \psi\Big(B - Q(n-1)\Big)
\end{aligned}
$$

reaches the sender, the sender should react to the message and accordingly updates its current window.

So far, we have considered the case of only TCP flows. However, our studies can be easily extended to more general scenarios where nonresponsive traffic such as User Datagram Protocol (UDP) coexists with TCP flows. In such case, by using classification based on port number and per-class queuing, application of the proposed scheme is relevant to only the TCP queuing class.

IV. Performance Evaluation

In this section, we evaluate the performance of the T-REFWA scheme. Performance evaluation is based on simulations using the Network Simulator (NS) [17]. While the proposed scheme can be implemented on any TCP variant, we consider the case of T-REFWA over NewReno. In all simulations, TCP Reno, TCP NewReno, and TCP Westwood are used as comparison terms. To investigate the fairness of the schemes, we use the following fairness index:

$$F(x) = \frac{(\sum_{i=1}^{N} x_i)^2}{N \cdot \sum_{i=1}^{N} (x_i)^2}$$

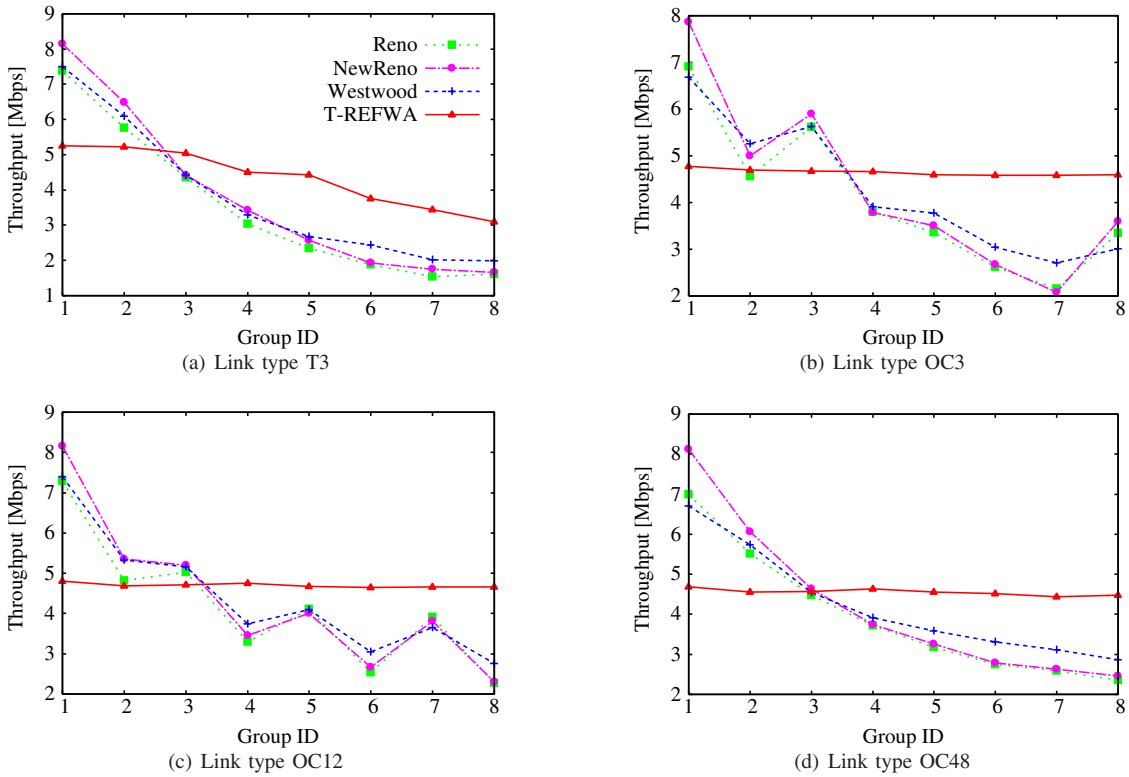
where N and x_i denote the total number of flows and the actual throughput of the i^{th} flow, respectively. The data packet size is fixed to 1kB. Buffers equal to the bandwidth-delay product of the bottleneck link are used. Drop-Tail is used as a packet-discarding policy. Simulations were all run for 60s.

First, to illustrate the basic working of the T-REFWA, we consider a simple network topology with a single bottleneck link, as shown in Fig. 1. The illustrated scenario depicts the case of a server providing a particular application (File Transfer Protocol (FTP) in the simulation) to a set of users grouped in eight equally-sized groups. This forms a single "one-to-many" network topology. Connections between the server and users from different groups have different RTTs. The figure indicates the propagation delay of links connected to end-terminals of each group. A number of test scenarios were created by setting the bottleneck bandwidth to different typical link speeds. For each link type, a maximum number of flows MAX is fixed. Table II lists the simulation parameters in case of each link type.

TABLE II
Simulation parameters: Link type and maximum number of flows.

Link Type	Link Bandwidth [Mbps]	Buffer Size [packets]	Flow Count (MAX)
T3	44	50	8
OC3	155	190	32
OC12	622	760	128
OC48	2400	2930	512

The average throughput of each group in case of the four link speeds is plotted in Fig. 2. In case of the T-REFWA scheme, the figure indicates that all flows could send nearly the same amount of packets. The figure demonstrates also the greediness of other TCP variants. Indeed, short RTT connections sent significantly larger number of packets compared to the long RTT connections. In Figs. 2-(b) and 2-(c), the throughputs achieved by Reno, NewReno, and Westwood



(a) Link type T3
(b) Link type OC3
(c) Link type OC12
(d) Link type OC48

Fig. 2. Average throughput of each group in case of each link type (Drop Tail).

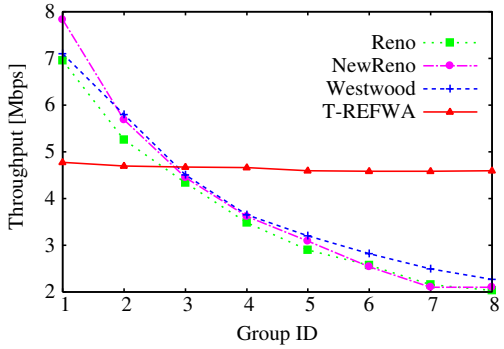


Fig. 3. Average throughput over RED in case of OC3 link type.

exhibit some oscillations, most probably due to the well-known TCP synchronization issue. AQM mechanisms can help to mitigate such phenomenon. To illustrate the idea with more clarity, we plot the throughput of the eight groups when the three TCP variants are used in a network with RED as packet discarding policy (Fig. 3). Due to the paper length limitation, we present only the results when OC3 is used. Identical results are obtained in case of other link types. Fig. 3 demonstrates a smooth decrease in the throughput when RTT increases. The figure demonstrates also that while T-REFWA is implemented over Drop Tail, it still shows the best performance compared to the other three schemes despite their use of an intelligent discarding policy such as RED. The obtained results confirm as well that the T-REFWA estimate of the average RTT of the system operates correctly in environments with high variance in the RTT distribution. The obtained results are in accordance with the results of Table III. Indeed, the table indicates that the T-REFWA scheme exhibits the highest value of fairness index while achieving the highest utilization of network resources.

To investigate the performance of the T-REFWA scheme in a more general scenario, we consider a network topology with two bottlenecks as shown in Fig. 4. Three groups of flows are considered. Group 1 consists of five flows and shares the bandwidth of the first bottleneck (link $L1$) with the four flows of group 2. Group 3 is formed of six flows and shares bandwidth of the second bottleneck (link $L3$) with flows of

TABLE III

FAIRNESS AND LINK UTILIZATION FOR DIFFERENT LINK SPEEDS.

Link Type	Protocol	Fairness	Link Utilization [%]
T3	Reno	0.74	63.5
	NewReno	0.74	69.1
	Westwood	0.80	69.1
	T-REFWA	0.96	79.0
OC3	Reno	0.86	83.5
	NewReno	0.84	88.8
	Westwood	0.90	87.8
	T-REFWA	1.00	95.9
OC12	Reno	0.86	85.6
	NewReno	0.85	89.9
	Westwood	0.90	90.5
	T-REFWA	1.00	96.7
OC48	Reno	0.86	84.2
	NewReno	0.83	89.9
	Westwood	0.91	90.1
	T-REFWA	1.00	97.2

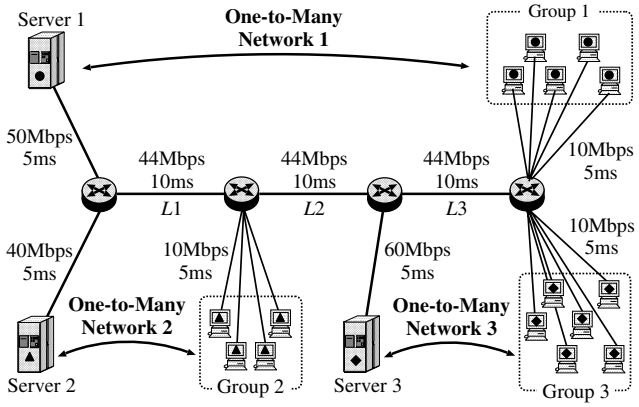


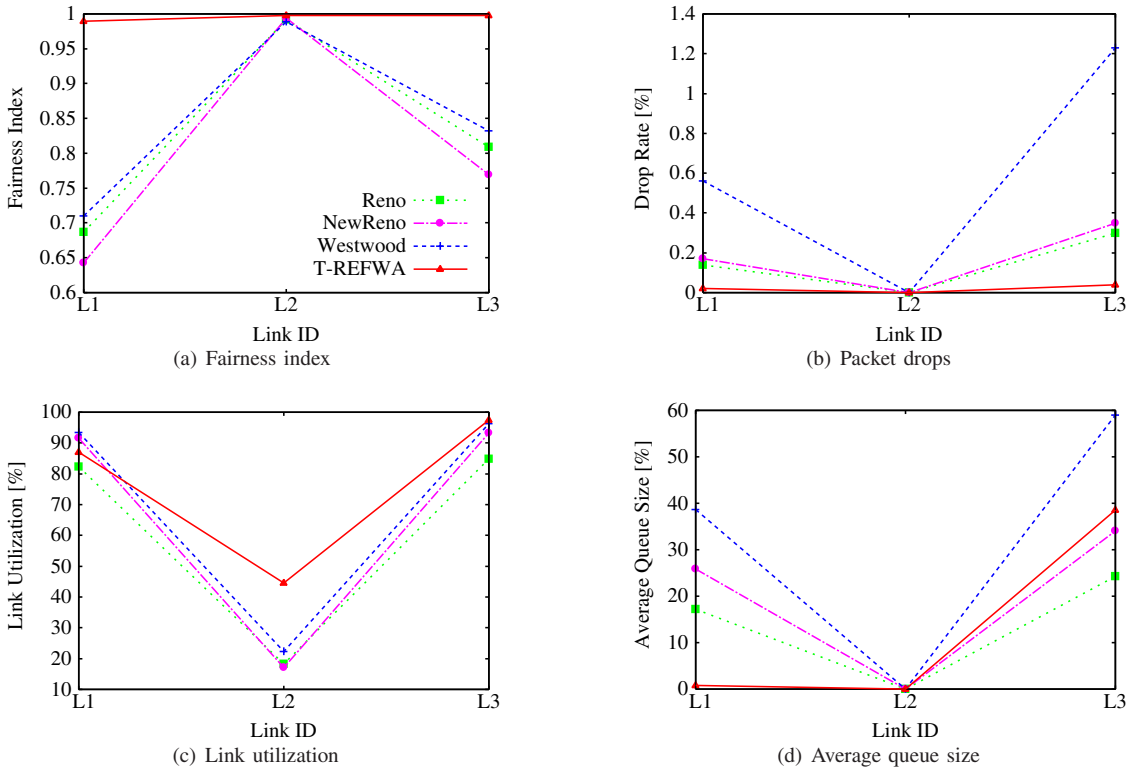
Fig. 4. A network topology composed of two bottlenecks.

group 1. The bandwidth and delays of the used links are as shown in Fig. 4. The three simulated groups have different servers and form three different "one-to-many" networks.

Fig. 5 graphs the link utilization, drops rate, fairness index, and average queue size experienced by the three simulated links. As explained earlier, the feedback value of a connection can be only downgraded along its path. If the feedback value set by a node exceeds the value computed by its downstream counterpart, the feedback value is marked down to the smaller value. In case of the considered network topology, TCP connections of group 1 are assumed to send data with rates fed back by link $L3$ that are smaller than the windows allocated by link $L1$. This compels flows of group 1 not to fully utilize their allocated bandwidths at link $L1$ causing the spare bandwidth to increase and the buffer occupancy to go down. This eventually leads to an increase in the computed feedbacks at link $L1$. Flows of group 2 will transmit data at rates equal to the computed feedback and this will help to fully utilize the $L1$ link capacity while maintaining small buffer sizes. Results of Fig. 5 demonstrate the high utilization of both links $L1$ and $L3$ and confirm this observation. It is noticed that without T-REFWA, link $L1$ is also highly utilized but this is at the price of lower system fairness. This is likely because flows of group 2 conquer most of the link bandwidth given their short RTTs. Observe also the high number of losses and large queue sizes experienced when T-REFWA is not used.

V. CONCLUSION

In this paper, we proposed a fair TCP-based congestion avoidance method based on a recently proposed scheme, REFWA, that was specifically designed for broadband satellite networks. Given the fact that prior knowledge of RTT is not available in terrestrial networks, information on RTT is stored in a discrete manner in the TOS field of IP packets and sent to congested routers. The latter use this information to compute optimal sending rates for each active TCP connection. Similarly to the original REFWA, the scheme matches the sum of window sizes of all active TCP connections sharing a bottleneck link to the effective bandwidth-delay product of the network. The proposed scheme improves the system fairness by assigning for each flow a weight relatively proportional to its RTT.



(a) Fairness index (b) Packet drops

(c) Link utilization (d) Average queue size

Fig. 5. Overall performance of the three links (Drop Tail).

The scheme is seen as an attractive solution to solve the unfairness issue of TCP in terrestrial networks. The scheme can be efficiently used for one-to-many private networks where a server serves a population of users located at different distances from the server. In such networks, guaranteeing fairness among users is mandatory. Simulation results demonstrated that T-REFWA has the potential of substantially improving the system fairness, reducing the number of losses and making higher utilization of the link. Experiments with multiple-bottlenecks network confirmed the good performance of T-REFWA in more complicated networks where multiple bottlenecks coexist. Finally, it should be noted that whilst more simulations with varying parameters should prove further the effectiveness of the scheme in varying network conditions, the presented results suffice at this stage. The authors are currently investigating the implementation of T-REFWA in hybrid wireless/wired networks. The actual implementation of the system forms also the focus of their future research work.

REFERENCES

[1] 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*, Rome, Italy, Jul. 2001, pp. 287–297.

[2] H. Balakrishnan, V.N. Padmanabhan, S. Seshan, M. Stemm, and R.H. Katz, "TCP Behavior of a Busy Internet Server: Analysis and Improvements," in *Proc. of IEEE INFOCOM*, San Francisco, CA, Mar. 1998, pp. 252–262.

[3] M. Fiore and C. Casetti, "An Adaptive Transport Protocol for Balanced Multihoming of Real-Time Traffic," in *Proc. of IEEE GLOBECOM*, St. Louis, MO, Nov./Dec. 2005, pp. 1091–1096.

[4] S. Athuraliya, V.H. Li, S.H. Low, and Q. Yin, "REM: Active Queue Management," *IEEE Network*, vol. 15, no.3, pp. 48–53, May/Jun. 2001.

[5] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Trans. on Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[6] K.K. Ramakrishnan and S. Floyd, "A Proposal to add Explicit Congestion Notification (ECN) to IP," RFC 2481, Jan. 1999.

[7] W. Feng, K.G. Shin, D.D.Kandlur, and D. Saha, "The BLUE Active Queue Management Algorithms," *IEEE/ACM Trans. on Networking*, vol. 10, no. 4, pp. 513–528, Aug. 2002.

[8] T.J. Ott, T.V. Lakshman, and L.H. Wong, "SRED: Stabilized RED," in *Proc. of IEEE INFOCOM*, New York, Mar. 1999, pp. 1346–1355.

[9] Y.H. Aoul, A.Naffa, D. Negru, and A. Mehaoua, "FAFC: Fast Adaptive Fuzzy AQM Controller For TCP/IP Networks," in *Proc. of IEEE GLOBECOM*, Dallas, TX, Nov./Dec. 2004, pp. 1319–1323.

[10] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks," in *Proc. of ACM SIGCOMM*, Pittsburgh, PA, Aug. 2002, pp. 89–102.

[11] L. Kalampoukas, A. Varma, and K.K. Ramakrishnan, "Explicit Window Adaptation: A Method to Enhance TCP Performance," *IEEE/ACM Trans. on Networking*, vol. 10, no. 3, pp. 338–350, Jun. 2002.

[12] J. Aweya, M. Ouellette, D.Y. Montuno, and Z. Yao, "WINTRAC: a TCP window adjustment scheme for bandwidth management," *Performance Evaluation*, vol. 46, no. 1, pp. 1–44, Sep. 2001.

[13] T. Taleb, N. Kato, and Y. Nemoto, "A Recursive, Explicit and Fair Method to Efficiently and Fairly Adjust TCP Windows in Satellite Networks," in *Proc. of IEEE ICC*, Paris, France, Jun. 2004, pp. 4268–4274.

[14] N. Brownlee and K.C. Claffy, "Understanding Internet Traffic Streams: Dragonflies and Tortoises," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 110–117, Oct. 2002.

[15] B. Huffaker, M. Fomenkov, D. Moore, and k. Claffy, "Macroscopic analyses of the infrastructure: measurement and visualization of Internet connectivity and performance," in *Proc. of PAM*, Amsterdam, Netherlands, Apr. 2001.

[16] T. Taleb, N. Kato, and Y. Nemoto, "REFWA: An Efficient and Fair Congestion Control Scheme for LEO Satellite Networks," to appear in *IEEE/ACM Trans. on Networking*, Oct. 2006.

[17] Network Simulator - ns (version 2) [Online]. Available: http://www.isi.edu/nsnam/ns/