A DCCP Variant for High Speed Networks

Carlos A. Froldi, Nelson L. S. da Fonseca
Institute of Computing, State University of Campinas
Campinas, Brazil
Email: {carlos,nfonseca}@ic.unicamp.br

Abstract **The Datagram Congestion Control Protocol (DCCP) protocol was proposed to provide a great variety of functionality to real-time applications. Its congestion control mechanisms were de ned to be friendly to the TCP-Reno, which makes its operation inappropriate to high speed networks. In this paper, a variant of the DCCP, called FAST DCCP, is proposed to overcome such de ciency. As evidenced by experimental evaluation as well as by simulation, the new protocol scales with the increase of network capacity.**

I. Introduction

Recently, there has been a considerable increase of the use of multimedia applications in the Internet. These applications are delay sensitive and have minimum transmission rates requirements. Although the User Data Protocol (UDP) is largely used for the transport of delay sensitive media, UDP does not support several functionalities desirable for multimedia applications.

Moreover, the wide use of UDP by multimedia applications has the potential of jeopardizing the bandwidth acquisition by TCP senders, since UDP does not implement any kind of congestion control mechanism. Besides that, some TCP features such as connection establishment are valuable to multimedia applications. Motivated by these issues, the IETF has proposed a novel transport layer protocol, called Datagram Congestion Control Protocol (DCCP) [1]. It provides connection establishment for unreliable data transfer as well as congestion control mechanisms. Two different TCP-friendly congestion control mechanism were proposed: the TCPLike mechanism [2], which is similar to the window-based congestion control mechanism implemented by TCP-Reno, and the *TCP-Friendly Rate Control* (TFRC) [3], which implements a rate-based congestion control type of mechanism.

Preliminary work evaluated the performance of DCCP in wireless networks as well as in low speed networks [4] [7]. Recently, we analyzed its performance in high speed networks and we made several recommendations for adapting it to this type of networks [8]. Results derived both via simulation and via experimental evaluation con rmed the lack of scalability of its congestion control mechanism. For a link capacity of 1 Gbps, DCCP utilizes at most 60% of the available bandwidth. This was expected since its congestion control mechanisms are friendly to TCP-Reno, which is inef cient to operate in high speed networks, motivating the creation of several TCP variants [9], [10] for high speed networks. In line with that and with the aim of making DCCP the choice of transport protocol for multimedia applications, it is of paramount importance to make its operation ef cient in high speed networks.

978-1-4673-0279-1/11/$26.00

This paper introduces a DCCP variant for high speed networks called FAST DCCP. Modi cations in the TFRC mechanism are proposed to promote scalability. This new mechanism is called FAST TFRC. Both simulation using the NS-2 network simulator and measurements of the FAST DCCP protocol operation in the Linux operating system were employed in the evaluation. Different scenarios were used for the assessment of the scalability, convergence, fairness and compatibility with the TCP-Reno. The evaluation conducted follows the recommendations in [11] for the analysis of new congestion control mechanisms for Internet protocols. It is the authors' best knowledge that this is the rst proposal of a DCCP variant for high-speed networks. It is shown that the FAST DCCP is scalable and also has several other desired properties.

The remainder of this paper is organized as follows. In Section II, the DCCP protocol is describe. In Section III, the FAST DCCP protocol is introduced. In Section IV, the properties evaluated in the analysis conducted are described. In Section V, results are discussed and in Section VI conclusions are drawn.

II. The DCCP Protocol

The DCCP protocol was designed to support multimedia applications in the Internet, such as video streaming and voice over IP. It provides congestion control mechanisms and packet arrival acknowledgments (ACKs) for applications which do not tolerate the overhead imposed by the TCP protocol. The main features of DCCP are connection-oriented without reliable delivery, acknowledgment for packet arrival, handshake for reliable connection establishment and termination and TCP-Friendly congestion control mechanisms (wich can use Explicit Congestion Noti cation).

The congestion control mechanisms supported by DCCP are de ned by a Congestion Control Identi er (CCID), which varies between 0 and 255. The CCIDs de ned initially for DCCP are: 0 Reserved; 1 Unspeci ed and based on the user; 2 TCP-Like; and 3 TFRC. The TFRC congestion control (CCID 3) is recommended for applications that need to transfer data at constant rates, for instance, voice over IP.

In the TFRC mechanism, the receiver measures the packet loss and returns this information to the sender, which uses these feedback to calculate the round-trip times (RTTs). The packet loss information as well as the estimated round trip

times are used to adjust the transmission rate at the sender. For that, the following ow equation, which is a simpli ed version of the throughput equation of TCP-Reno, is employed.

$$X = \frac{s}{R * \sqrt{2 * b * \frac{p}{3}} + tRTO * 3 * \sqrt{3 * b * \frac{p}{8}} * p * (1 + 32 * p^2)}, \quad (1)$$

where X is the transmission rate in bytes/second; s is the packet size in bytes; R is the estimated RTT in seconds; b is the maximum number of packets acknowledged by a single packet and set to 1; p is a value between 0 and 1 which is de ned by the loss ratio, and $tRTO$ is the timeout of the TCP retransmission in seconds.

The transmission rate of the senders are adjusted as a function of the reception of acknowledgment packets. An acknowledgment packet carries the following information: the time stamp of the last data packet received, the time interval between the arrival of the last packet and the sending of its acknowledgment, and the estimated rate at which the receiver has received packets after sending the last acknowledgment packet.

A precise estimation of the loss ratio p is of paramount importance to the TFRC; the receiver makes such estimation by accounting the number of packets received. Each packet transmitted has a sequence number, which is incremented by one for each packet sent. The receiver maintains a data structure for keeping the history of the packets received and those lost. A lost packet is detected when at least three packets with higher sequence numbers are received. To perform the calculation of the loss ratio, it is necessary to determine the mean interval between losses, which is done by considering the n most recent loss events and assigning weights to them. The computation of the weights of these events is de ned in [3].

III. FAST DCCP

As reported in [8], the major de ciency of the DCCP protocol is the lack of scalability. To overcome such limitation, the rate should be increased in a more aggressive fashion as well as it should be decreased in a smoother way than as in DCCP. The proposed changes aim to improve these two rates and they were based in the operation of the FAST TCP protocol [9].

In DCCP, the update on the transmission rate considers the connection mean delay, (R), and the sample delay, ($Rsample$), as follows [3]:

$$R_{i+1} = (0.9 * R_i) + (1–0.9) * Rsample \quad (2)$$

$$Rsample = (tnow–trecvdata)–tdelay \quad (3)$$

where $tnow$ is the time at which the packet is received at the sender; $trecvdata$ is the timestamp of the packet last received and $tdelay$ is the elapsed time between the reception of the last packet and the sending of its acknowledgement.

FAST TRFC introduces a multiplicative factor, α, for the increase of the transmission rate when no loss occurs. This multiplicative factor considers the ratio between the mean delay and the sampled delay. If this ratio is greater than 1, it means that there is available bandwidth in the network which allows the increase of the transmission rate. A new value for the transmission rate ($Xupdate$) is computed as follows:

$$Xupdate = ((X * \alpha) + X)); \quad (4)$$

where X is the current transmission rate. Moreover, the new value of X is compute as follows:

$$X = max(Xupdate, 2 * X, 2 * Xrecv, s/R); \quad (5)$$

where $Xrecv$ is the transmission rate informed by the receiver and s is the packet size de ned for the FAST DCCP connection. Whenever $\alpha > 1$, $X = Xupdate$

which is quite more aggressive than traditional TRFC that updates the transmission rate according to [3]:

$$X = max(min(2 * X, 2 * Xrecv)), s/R); \quad (6)$$

The update of the transmission rate when loss occurs impacts the capacity of acquisition of bandwidth in the long run. FAST TRFC updates the rate according to:

$$X = max(Xcalc, 2 * Xrecv, s/t_mbi); \quad (7)$$

where $Xcalc$ is the rate given by Equation 1 and it is the rate given by the periodic sending of a packet at intervals of duration t_mbi, set to 64 seconds.

This is in major contrast to TRFC which updates the rate in case of loss, as folllows [3]:

$$X = max(min(Xcalc, 2 * Xrecv), s/t_mbi); \quad (8)$$

So, rather than choosing the minimum value among those computed by TRFC for the update of the transmission rate ($Xcalc$, $2 * Xrecv$), the new FAST TRFC takes the maximum value, making it less conservative than traditional TRFC.

IV. EVALUATED METRICS

To evaluate the performance of the FAST DCCP protocol, the following properties were evaluated in the experiments: scalability, fairness, convergence and compatibility with TCP-Reno, which is in accordance with the recommendation in [11]. In this section, these metrics are described, as well as their relevance to the protocol performance.

Scalability is the ability of the protocol to adapt itself to the increasing availability of network resources. The experiments assessed the scalability as a function of the link capacity. The aim was to investigate whether the protocol can use ef ciently the available bandwidth when the link capacity increases. Scalability was also evaluated as a function of the number of connections.

Fairness is the ability of the protocol to ensure an even share of bandwidth among all connections. Fairness in this

paper is measured according to the Jain Fairness Index [12], which varies between 0 and 1. Values closer to 0 indicate lack of fairness and values closer to 1 represent a fair system.

Convergence is the duration required to reach fairness, i.e., an even share of bandwidth among all connections. Such property is important to ensure that recently created connections can fairly compete for resources with older ones. Moreover, whenever existing connections are terminated, the bandwidth released must be equally distributed among all other active connections.

Compatibility with TCP-Reno is the ability to ensure that the connections of a protocol behave fairly with existing TCP connections, so that the connections of both protocols share equally the available bandwidth.

V. Performance Evaluation

The scenarios and metrics employed to evaluate the FAST DCCP protocol in high-speed networks were derived in [11]. The experiments employed the *Dumbbell* topology, illustred in Figure 1, which contains a single bottleneck link comprised by two routers. The number of connections varied according to the scenario considered. The network simulator NS-2 [13], version 2.31, was used for the simulation experiments. The existing implementation of the DCCP protocol in the Linux operating system, kernel version 2.6.20, was utilized for incorporating the changes that turns it into the FAST DCCP variant.

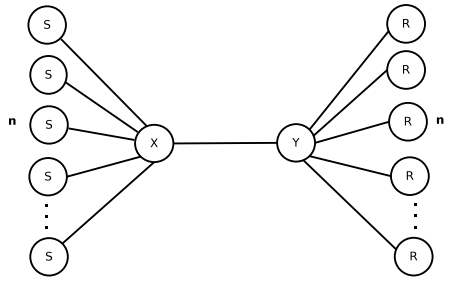

Fig. 1: The Dumbbell topology.

In the simulation experiments with NS-2, the two routers were established with the droptail queue management and a buffer size of 2500 packets, which is similar to those of real routers [14]. To avoid synchronization, the transmission start time and the propagation delay for the connections were randomized. The transmission start times varied in an interval between 1 and 10s, while propagation delays varied in an interval between 5 and 15ms. Background traf c was introduced in order to reproduce aggregated traf c, as in real network scenarios. The reverse direction background traf c used 20%, 50% or 80% of the capacity of the bottleneck link, and it was composed by 20% UDP traf c, 56% web traf c and 24% FTP traf c. The traf c generated by the FAST DCCP connections was of type CBR. Each connection had its transmission rate equivalent to the bottleneck link capacity.

For the measurement experiments, two computers with 1Gbps Ethernet interface were interconnected by a switch router of 1Gbps capacity per port. The ports were set up with a static route in order to emulate two different routers, since each port has a independent queue. The background traf c distribution and the capacity constraints were obtained by the use of the Linux module Traf c Control (tc) [15]. The generation and transmission of traf c for the FAST DCCP and for the TCP connections were accomplished by the tool *Iperf* [16], which was con gured to reproduce CBR traf c and to have transmission start times varying between 1 and 10s.

The graphics show average values obtained from the execution of several replications for each experiment. To calculate the con dence interval with a con dence level of 95% the method of independent replication was employed. The simulation time for all the experiments was 600s. Results are compared to those derived in [8] to highlight the bene ts of the adoption of the FAST DCCP protocol.

A. Scalability

Two different scenarios were created to evaluate the scalability of the FAST DCCP protocol: one to assess the scalability as a function of the link capacity and the other to assess the scalability as a function of the number of connections.

In the rst scenario, the capacity of the bottleneck link took values in the set [155Mbps, 622Mbps, 1Gbps] for the measurement experiments. Moreover, in the simulation experiments links with capacity of 2.5Gbps were also used. The propagation delay was set to 100ms, and the number of FAST DCCP connections was 16. Figures 2 and 3 present the link utilization as a function of the link capacity, obtained from simulation and measurement experiments, respectively. These two gures show a similar behavior: FAST DCCP utilized ef ciently the available bandwidth, occupying more than 90% of different capacity values. Moreover, Figures 4 and 5 present the results for the same experiment using DCCP [8]. As the link capacity increases the utilization decreases to values lower than 60% of the link capacity. These results demonstrates that FAST DCCP is scalable as a function of the link capacity, i.e., the new mechanism FAST TRFC is capable of ef ciently utilizing high capacity channels.

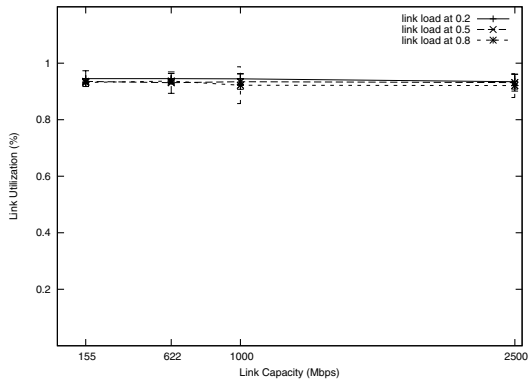

Fig. 2: Link utilization as a function of capacity (simulation).

In the second scenario, scalability is assessed as a function of the number of connections. To accomplish that, the number

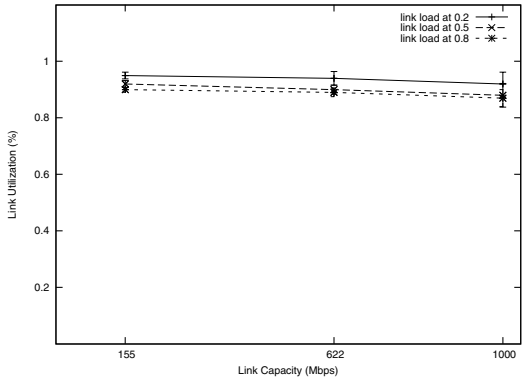

Fig. 3: Link utilization as a function of capacity (measurement).

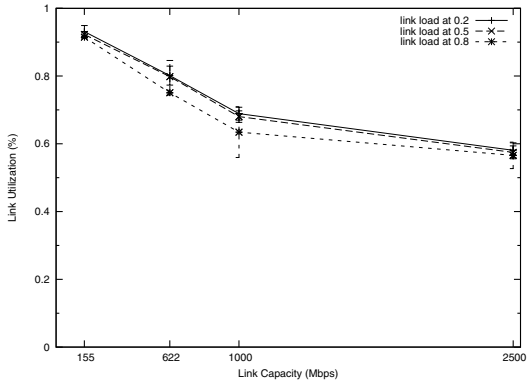

Fig. 4: Link utilization as a function of capacity (simulation).

of FAST DCCP connections varied in the set $[2^1, 2^2, ..., 2^5]$ (2 32) for the measurement experiments whereas it varied in the set $[2^1, 2^2, ..., 2^8]$ (2 256) for the simulation experiments. The former employed a smaller number of connections due to hardware limitations. The capacity of the bottleneck link was set to 1Gbps and the propagation delay was set to 100ms. Figures 6 and 7 show the link utilization as a function of the number of connections, obtained, respectively, from simulation and from measurement experiments. Similarly to DCCP [8], FAST DCCP is scalable as a function of the number of connections. Moreover, in this paper, we show that FAST DCCP maintains the scalability as a function of the number of connections for capacities higher than those DCCP is capable to maintain.

B. Fairness

Two other scenarios were employed to evaluate the fairness promoted by the FAST DCCP protocol. In both scenarios, there are four FAST DCCP connections exchanging data through the bottleneck link. The propagation delay is 100ms, and the link capacity is 1Gbps. In the rst scenario, the FAST DCCP connections are established simultaneously ($t = 1s$), while in the second scenario, the FAST DCCP connections start at random times ($1s <= t <= 10s$). Figures 8 and 9 present fairness results for the rst and the second scenarios,

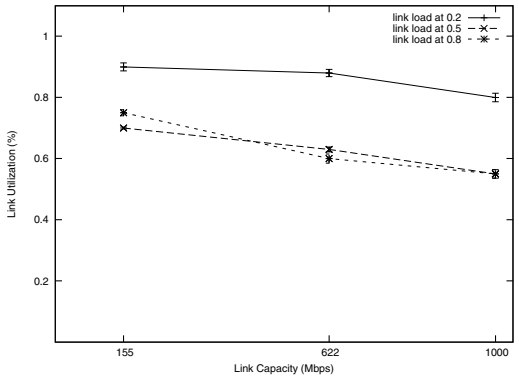

Fig. 5: Link utilization as a function of capacity (measurement).

respectively. Results from the simulation and the measurement experiment are shown in both graphs. It can be noticed that the values of Jain of Fairness Index are close to 0.9 in both scenarios, regardless of the intensity of the background traf c, which evinces that fairness was promoted, i.e., the FAST DCCP protocol ensured an even share of bandwidth among all connections. FAST DCCP promotes fairness similar to that of traditional DCCP [8].

C. Convergence

Another scenario was employed to evaluate the convergence to fairness. The bottleneck link was set to 1Gbps, two FAST DCCP connections were employed, and the propagation delay was set to 10ms. The second FAST DCCP connection transmitted traf c only at $t = 50s$, which is long enough for the rst connection to allocate most of the available bandwidth of the bottleneck link. Figure 10 shows the time taken to reach equal share of bandwidth for both simulation and measurement experiments, with different background traf c. It can be seen that the convergence time was long in both cases, with values varying between 100s and 135s. Such duration is much longer than the average duration of typical Internet connections. Thus, it can be said that, in despite of the high levels of fairness

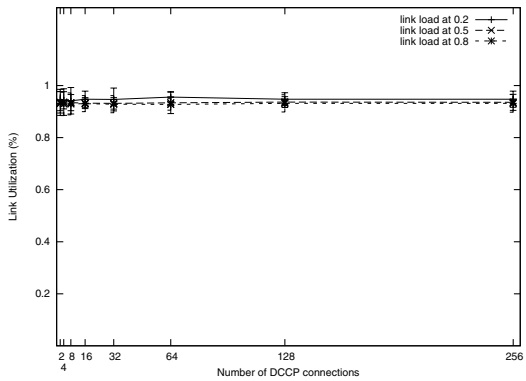

Fig. 6: Link utilization as a function of the number of DCCP connections (simulation).

reached, TFRC cannot provide a rapid response for balancing the shares of bandwidth among the existing connections. Indeed, the convergence time of FAST TRFC is 7 % higher than that of traditional TRFC. This nding is in accordance with previous work [17], [18], which also indicated the high response time of TFRC, being considerably slower than that of TCP-Reno.

D. Compatibility with TCP-Reno

To evaluate the compatibility of the FAST DCCP protocol with the TCP-Reno protocol, two other scenarios were created. In the rst, two TCP-Reno connections were set, both competing for the bottleneck link bandwidth and transmitting FTP traf c. In the second scenario, a FAST DCCP connection and a TCP-Reno connection were set, with the latter transmitting the same FTP traf c as in the previous scenario and both competing for the bottleneck link bandwidth. The link capacity was set to 1Gbps and the propagation delay was set to 10ms. The aim of these scenarios is to evaluate whether the TCP-Reno connection obtains the same link bandwidth in the presence of FAST DCCP ow that it would obtain if it were competing with another TCP-Reno ow.

Tables I and II present results derived in these experiments. The link utilization, called, TCP-1 shows values when the TCP ow competes with another TCP ow, whereas the link utilization TCP-2 shows values of the TCP ow when it competes with a FAST DCCP ow. The difference in the values observed suggests that the FAST DCCP protocol is not compatible with TCP-Reno when it employs the FAST TFRC congestion control mechanism, since the TCP-2 values indicate a lower link utilization than that of TCP-1. When compared to DCCP, FAST DCCP has a higher degree of incompatibility with TCP-Reno. This is due to the fact that FAST TRFC is more aggressive than traditional TRFC. Moreover, such incompatibility is in line with those of TCP variants for high speed networks such as [19] [21].

VI. Conclusion

The DCCP protocol was proposed to support real-time applications currently supported by UDP but it incorporates

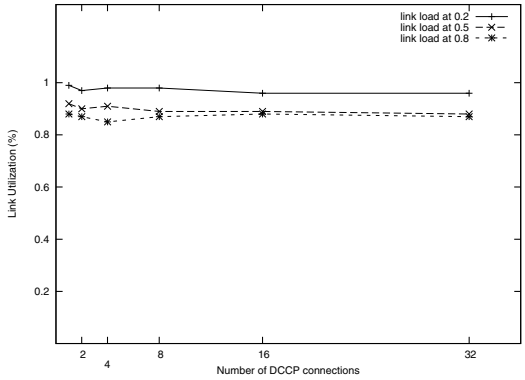

Fig. 7: Link utilization as a function of the number of DCCP connections (measurement).

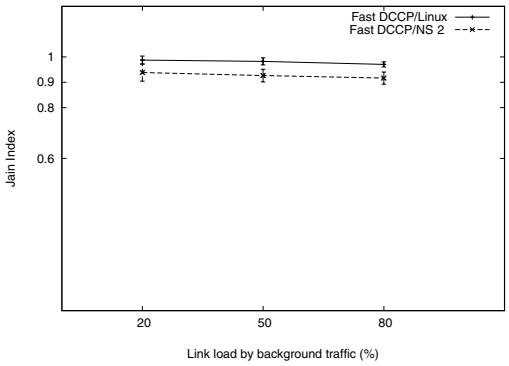

Fig. 8: Fairness: rst scenario (simulation and measurement).

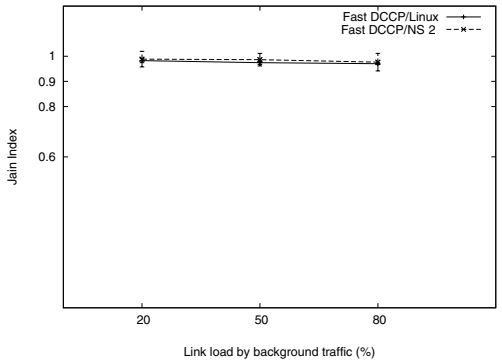

Fig. 9: Fairness: second scenario (simulation and measurement).

TABLE I: Link utilization of the TCP connections (simulation).

		Background Traf c (%)		
		0.2	0.5	0.8
Link utilization (%)	TCP-1	0.0152	0.0124	0.0061
	TCP-2	0.0043	0.0031	0.0012

TABLE II: Link utilization of the TCP connections (measurement).

		Background Traf c (%)		
		0.2	0.5	0.8
Link utilization (%)	TCP-1	0.3120	0.2210	0.1830
	TCP-2	0.1225	0.0885	0.0523

a greater variety of non-existing functionality in the latter. As any other protocol that adopts TCP-Reno friendly congestion control mechanism, the DCCP does not scale with the increase of available bandwidth, as reported in [8]. This paper introduced a variant of DCCP called FAST DCCP which congestion control mechanism is much more agressive than that of TRFC in the absence of loss and smoother than that of TRFC when loss occurs. As a result, the new protocol scales with the increase of the available bandwidth. As TCP variants for high speed networks, FAST DCCP is less friendly to TCP-Reno than the DCCP protocol.

To speed up convergence to fairnes, recommendations made

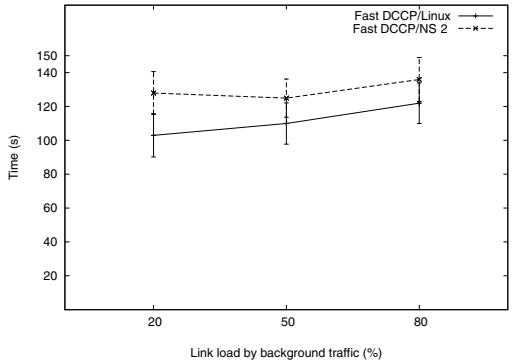

Fig. 10: Scenario of convergence (simulation and measurement).

to DCCP in [17] and [18] should also be considered to the FAST DCCP. Moreover, given such encouraging results found in the present evaluation, the FAST DCCP is recommended for the transport of real-time applications in high speed networks.

REFERENCES

[1] S. Floyd, E. Kohler, and M. Handley, Designing dccp: Congestion control without reliability, in *Proceedings of ACM SIGCOMM'06*, Pisa, Italia, 2006, pp. 27 38.

[2] S. Floyd and E. Kohler, Pro le for datagram congestion control protocol (dccp) - congestion control id 2: Tcp-like congestion control, in *RFC 4341*, 2006.

[3] S. Floyd, M. Handley, J. Padhye, and J. Widmer, Tcp friendly rate control (tfrc): Protocol speci cation, in *RFC 3448*, 2006.

[4] P. Navaratnam, N. Akhtar, and R. Tafazolli, On the performance of dccp in wireless mesh networks, in *Proceedings of ACM MobiWac'06*, 2006, pp. 144 147.

[5] S. Takeuchi, H. Koga, K. Iida, Y. Kadobayashi, and S. Yamaguchi, Performance evaluations of dccp for bursty traf c in real-time applications, in *Proceedings of IEEE SAINT'05*, 2005, pp. 142 149.

[6] X. Gu, P. Di, and L. Wolf, Performance evaluation of dccp: A focus on smoothness and tcp-friendliness, in *Annals of Telecommunications Journal, 61*, 2006, pp. 191 216.

[7] G.Sawar, E. Lochin, and R. Boreli, Experimental performance of dccp over live satellite and long range wireless links, in *Proceedings of IEEE ICSIT'07*, Sydney, Australia, 2007.

[8] C. Froldi, N. Fonseca, C. Papotti, and D. Manzato, Performance evaluation of the dccp protocol in high-speed networks, in *Proceedings of IEEE CAMAD 2010*, Miami, USA, 2010, pp. 41 46.

[9] C. Jin, D. Wei, and S. Low, Fast tcp: Motivation, architecture, algorithms, performance, in *Proceedings of IEEE/ACM Transactions on Network*, 2006, pp. 1246 1259.

[10] S. Ha, I. Rhee, and L. Xu, Cubic:a new tcp-friendly high-speed tcp variant, in *Proceedings of ACM SIGOPS*, 2008, pp. 64 74.

[11] S. Floyd and E. Kohler, Tools for the evaluation of simulation and testbed scenarios, in *Internet Draft: draft-irtf-tmrg-tools-05*, 2006.

[12] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling*. John Wiley & Sons, 1991.

[13] http://www.isi.edu/nsnam/ns, 2010.

[14] D. X. Wei, P. Cao, and S. H. Low, Time for a tcp benchmark suite? http://www.cs.caltech.edu/ weixl/research/technicals/benchmark/summary.ps, 2005.

[15] http://tldp.org/howto/traf c-control-howto/index.html, 2010.

[16] http://www.noc.ucf.edu/tools/iperf, 2010.

[17] D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker, Dynamic behavior of slowly-responsive congestion control algorithms, in *Proceedings of ACM SIGCOMM'01*, San Diego,CA, 2001, pp. 263 274.

[18] M. Vojnovic and J. L. Boudec, On the long run behavior of equation-based rate control, in *Proceedings of ACM SIGCOMM'02*, 2003, pp. 103 116.

[19] H. Jamal and K. Sultan, Performance analysis of tcp congestion control algorithms, in *INTERNATIONAL JOURNAL OF COMPUTERS AND COMMUNICATIONS*, 2008, pp. 30 38.

[20] Y. Li, D. Leith, and R. Shorten, Experimental evaluation of tcp protocols for high-speed networks, in *ACM TRANSACTIONS ON NETWORKING, Vol. 15*, 2007, pp. 1109 1122.

[21] T. Hatano, H. Shigeno, and K. Okada, Tcp-friendly congestion control for highspeed network, in *Proceedings of IEEE SAINT'07*, 2007, pp. 1 10.