A New Incremental Control Congestion Algorithm for High-Speed Networks

Firas Nakoul
Faculty of Mech. & Elec. Engineering
Airport Road – Damascus
SYRIA
f_nakoul@yahoo.com

Abstract: Several congestion control algorithms have been proposed for use in High-speed networks with large delays. In these networks the TCP may have problems utilizing the full bandwidth. A suitable protocol should not take away too much bandwidth from standard TCP flows while utilizing the full bandwidth of high-speed networks. In this paper, besides TCP friendliness and bandwidth scalability properties, RTT (round-trip time) unfairness is considered, that means the situation in which competing flows with different RTTs may consume vastly unfair bandwidth shares. RTT unfairness for high-speed networks occurs distinctly with drop tail routers for flows with large congestion windows where packet loss can be highly synchronized. A new congestion control scheme that alleviates RTT unfairness is proposed. The simulation results confirm the properties of the proposed protocol.

Keywords: congestion control, high-speed networks, RTT unfairness, protocol design.

Mr.Firas Nakoul was born born at Damascus, Syria on January 22, 1974. He received his MSc degree in Mechanical and Electrical Engineering from Damascus University in 1996. As a PhD student his current research interests include communication technologies, high speed networks and network applications

1. Introduction

In the context of communication networks, the term ,'congestion control' is generally used to refer to the action of regulating various flows within a network. Such a control action is needed due to scarcity of network resources, such as link capacities. There are two main objectives of congestion control. First, the efficient use of the network resources while keeping the loss rate, and average delay of all 'connections' at some reasonable level. Second, it is desirable to maintain some notion of 'fairness' between sources sharing a communication link [1].

In contrast to the traditional, low-bandwidth, voice-based telephone network, high-speed networks, such as B-ISDN (Broadband Integrated Service Digital Network), allow several types of network traffic (such as, voice and video) to coexist in the same transmission medium. The traffic in such networks can be broadly classified as guaranteed-service traffic and best-effort service traffic. Guaranteed service refers to a contract between the network service provider and the end user which requires the network to provide fixed QoS (Quality of Service) to the traffic. The QoS guarantees can be in the form of upper bounds on packet loss probability, delay, etc. In contrast, best-effort traffic is guaranteed very little a priori.

In the context of ATM (Asynchronous Transfer Mode) networks, the best effort traffic may be guaranteed a minimum rate, and a bound on the loss rate. Instead of guaranteeing fixed QoS parameters, the idea is to fairly allocate network resources to competing users. In the Internet today, most users can be thought of generating best-effort type traffic, too. For instance, no traffic related service guarantees are made in advance when browsing a web page, or sending an email (unless the connection uses a leased line). Thus, in general, best-effort sources can adjust their rates to the level of available service, making it possible to control the congestion in the network.

For a best effort source to adapt its rate to changing network conditions, there must be a mechanism through which information about the state of the network is conveyed to the source. This information can be in the form of bandwidth availability, state of congestion, or impending congestion. In ATM networks, this is achieved via explicit rate control messages. In the Internet, however, no explicit feedback from the routers is available. TCP/IP (the current Internet protocol) uses a window-based congestion control mechanism, where the sources dynamically adjust their window sizes using packet losses and timeouts as congestion indicators [2]. Although TCP has several nice features, such as being self-clocking, easy-to-implement, etc., it does not perform well under various scenarios. Several new protocols which address the TCP limitations have been put forward including XCP [10], SABUL [11], FAST [12], High Speed TCP (HSTCP) [7, 8], and Scalable TCP (STCP) [9]. Except XCP, these protocols adaptively adjust their increase rates based on the current window size.

The work presented here began as an attempt to assess the performance of high-speed congestion control protocols in under "realistic" network operation: i.e., to assess their impact on overall network performance in the presence of heterogeneous, competing flows. We focus on HSTCP and STCP, which are window-based, self-clocking protocols known for safer incremental deployment [14].

The rest of this paper is organized as follows. In the next section, the TCP performance problems in HS networks are addressed. A HSTCP protocol version is presented in Section 3. In Section 4 a new version of an incremented congestion control is presented. Section 5 contains simulation results, and the paper ends with the concluding remarks of Section 6, which identifies some challenges for future research.

2. TCP Performance Problems in High-Speed Links

TCP was first designed in the early 1970s. Many research, development and standardization efforts have been devoted to the TCP/IP technology since then. It is widely used in the current Internet and it is the standard transport-layer protocol.

Congestion management allows the protocol to react to and recover from congestion and operate in a state of high throughput sharing the link fairly with other traffic. In [2] the original TCP congestion management algorithms were proposed. TCP's congestion management is composed of two important algorithms. The slow-start and congestion avoidance algorithms allow TCP to increase the data transmission rate without overwhelming the network. They use a variable called CWND (Congestion Window). TCP's congestion window is the size of the sliding window used by the sender. TCP cannot inject more than CWND segments of unacknowledged data into the network.

The general characteristics of the TCP algorithm are an initial relatively fast scan of the network capacity followed by a cyclic adaptive behavior that reacts quickly to congestion by reducing its sending rate, and then slowly increasing the sending rate in an attempt to stay within the area of maximal transfer efficiency. This general behavior is shown in Figure 1.

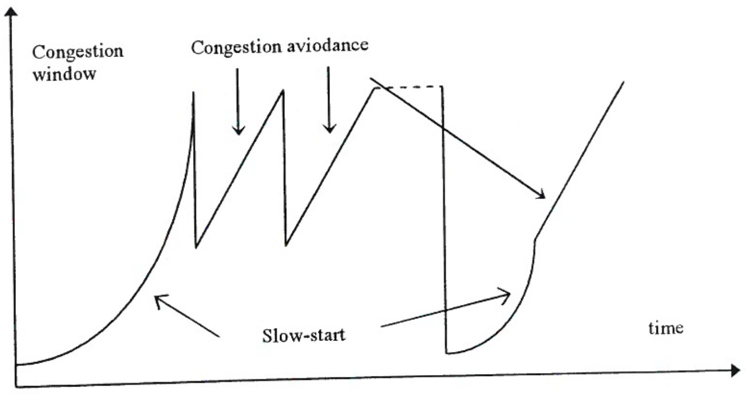

Figure 1: TCP Congestion Control.

TCP's algorithms are referred to as AIMD (Additive Increase Multiplicative Decrease) and are the basis for its steady-state Congestion Control. TCP increases the congestion window by one packet per window of data acknowledged, and halves the window for every window of data containing packet drop. The TCP congestion control can roughly be expressed in the following equations:

Congestion Avoidance

$$\text{ACK}: \quad cwnd \leftarrow cwnd + a / cwnd \tag{1}$$

$$\text{DROP}: \quad cwnd \leftarrow cwnd - b \times cwnd \tag{2}$$

Slow-Start

$$\text{ACK}: \quad cwnd \leftarrow cwnd + c \tag{3}$$

The terms cwnd, a, b and c are all defined in units of Maximum Segment Size (MSS). The canonical values for a, b, and c are: a = 1, b = 0.5 and c = 1.

The introduction of high-speed network technologies has opened the opportunity for a dramatic change in the achievable performance of TCP based applications. Unfortunately, this potential has not generally been realized.

The performance of a TCP connection is dependent on the network bandwidth, round-trip time (RTT), and packet loss rate. At present, TCP implementations can only reach the large congestion windows necessary to fill a pipe with a high bandwidth delay product when there is an exceedingly low packet loss rate. Otherwise, random losses lead to a significant throughput deterioration when the product of the loss probability and the square of the bandwidth delay is larger then one [3].

For example, a standard TCP connection with 1500-byte packets and a 100 ms round-trip time, would require an average congestion window of 83,333 segments and a packet drop rate of at most one congestion event every 5,000,000,000 packets to achieve a steady-state throughput of 10 Gbps (this translates to at most one congestion event every 1h:40m). This loss rate is well below what is possible today with the present optical fiber and router technology.

3. High-Speed TCP Fundamentals

The HSTCP for Large Congestion Windows was introduced in [16] as a modification of TCP's congestion control mechanism to improve performance of TCP connections with large congestion windows. HSTCP is designed to have a different response in environments of very low congestion event rate, and to have the standard TCP (referred to in this work as Regular TCP or REGTCP) response in environments with packet loss rates of at most 10^{-3}. Since, it leaves TCP's behavior unchanged in environments with mild to heavy congestion, it does not increase the risk of congestion collapse. In environments with very low packet loss rates (typically lower than 10^{-3}), HSTCP presents a more aggressive response function.

HSTCP introduces a new relation between the average congestion window w and the steady-state packet drop rate p. For simplicity, this new HSTCP response function maintains the property that the response function gives a straight line on a log-log scale. (as does the response function for Regular TCP, for low to moderate congestion).

The HSTCP response function is specified using three parameters: Low Window, High Window, and High P. Low Window is used to establish a point of transition and ensure compatibility. The HSTCP response function uses the same response function as Regular TCP when the current congestion window is at most Low Window, and uses the HSTCP response function when the current congestion window is greater than Low Window. High Window and High P are used to specify the upper end of the HSTCP response function. It is set as the specific packet drop rate High P, needed in the HSTCP response function to achieve an average congestion window of High Window. The HSTCP response function is represented by new additive increase and multiplicative decrease parameters. These parameters modify both the increase and decrease parameters according to CWND. In congestion avoidance phase, its behavior can be expressed in the same manner as in (1) and (2) with a and b act like variables.

4. Incremental Congestion Control Algorithm (ICC)

The pseudo-code of ICC implemented as a modification of BIC from [4]. Preset parameters used: *low_window*: if the window size is less than this threshold, normal TCP increase/decrease, otherwise ICC; *Smax*: the maximum increment ; *Smin*: the minimum increment ; b : multiplicative window decrease factor.

The following variables are used: *Wmax*: the maximum window size ; *cwnd*: congestion window size;

ICC_inc: window increment per RTT.

```
if (cwnd < low_window){              //normal TCP
cwnd = cwnd *0.5; return; }
if (cwnd < Wmax)
  Wmax = cwnd * (a- b ) / a;
else
  Wmax = cwnd;
cwnd = cwnd * (1- b );               //multiplicative decrease
```

An acknowledgment for a new packet arrives:

```
if (cwnd < low_window) {             //normal TCP
cwnd = cwnd + 1/cwnd; return;  }
if  (cwnd < Wmax)
```

```
        $ICC\_inc = (Wmax - cwnd)/a$;
else                                    //slow-start
        $ICC\_inc = cwnd - Wmax$;
if ($ICC\_inc > Smax$)
        $ICC\_inc = Smax$;
else if ($ICC\_inc < Smin$)             //slow-start
        $ICC\_inc = Smin$;
$cwnd = cwnd + ICC\_inc/cwnd$;
```

Two special cases are considered as in [4]. First it is assumed that $W_{\max} b >> 2S_{\max}$. The sending rate of ICC mainly depends on the linear increase part, and for small values of p, it becomes independent of Smin and can be approximated as follows:

$$R \approx \frac{1}{RTT}\sqrt{\frac{S_{\max}}{2}\frac{a-b}{b}\frac{1}{p}} \tag{4}$$

Second, in the case $W_{\max} b \le 2S_{\max}$ and assuming 1/p >> Smin then

$$R \approx \frac{W_{\max}}{RTT}\left(1-\frac{2b}{\log_2\left(\frac{W_{\max} b}{S_{\min}}\right)+2}\right) \approx \frac{W_{\max}}{RTT} \tag{5}$$

the last approximation is valid when $2b << \log_2(W_{\max} b / S_{\min}) + 2$. In the case when $W_{\max} b << 2S_{\max}$ the sending rate becomes independent of *Smax*.

The sending rate of ICC is proportional to $1/p^d$, with $1/2 < d < 1$. As the window size increases, d decreases from 1 to 1/2. For a fixed b, when the window size is small, the sending rate is a function of *Smin* and when the window size is large, a function of *Smax*. When window is small, the protocol is TCP-friendly; and when the window is large, it is more RTT fair and gives a higher sending rate than TCP. This can be realized by adjusting *Smin* and *Smax*.

It is considered the RTT unfairness of a protocol under the synchronized loss model. The RTT unfairness of a protocol with a response function $\frac{1}{RTT}\frac{c}{p^d}$ is $(RTT_2 / RTT_1)^{1/(1-d)}$.

As the window size increases, the value of d of ICC decreases from 1 to 1/2. That is, with a high bandwidth, ICC has the same RTT unfairness as the TCP; while with a low bandwidth, ICC has the same RTT unfairness as STCP.

5. Simulation Results

The experiments were conducted using the NS-2 simulator [6]. The simulation network topology used was a *dumbbell* with a single bottleneck, as shown in Figure 2. All traffic passed through the bottleneck link. The bottleneck link bandwidth was 1 Gbits/s, the link delay was 50 ms. The simulations used two types of router queue management, DT (DropTail) and RED (Random Early Detection).

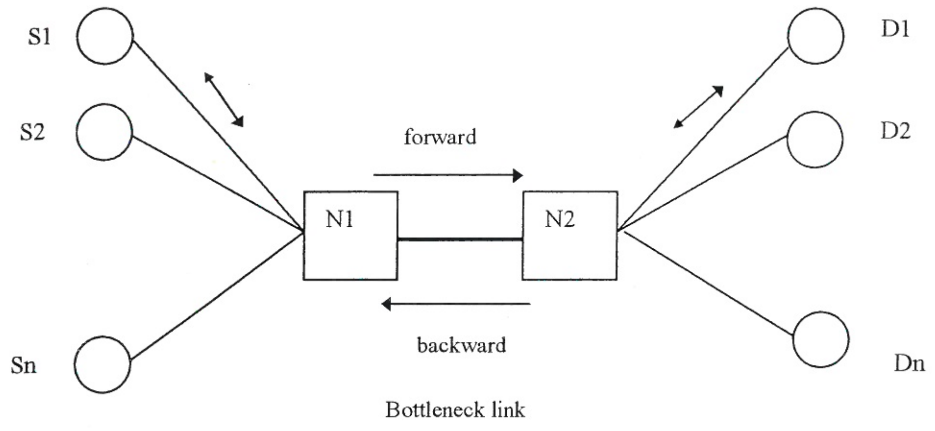

Figure 2: Simulation Network Topology.

The TCP flows had the ECN bit set; the packet size was 1500 bytes; the maximum window size was large enough to not impose limits; random times between sends were set to avoid phase effects; the flows used a modified version of the Limited Slow-Start algorithm for large congestion windows[5]. The TCP agent used for the sender and the receiver was SACK. FTP was the application used to transmit data through the TCP connections. All the HSTCP flows used the forward direction. The HSTCP parameters used are

Low_Window = 31;High_Window = 83000; High_ P = 0.0000001.

For comparison, the HSTCP flows were run against TCP SACK implementations. A set of web-like flows and a set of small TCP flows were also used as *background noise* for all the simulation. Other TCP flows were used to represent bursty traffic. They were short-lived flows that lasted for a few seconds.

The first experiment allowed to observe the basic behavior of a single REGTCP or a single HSTCP flow run in isolation. The experiment ran only one time, without external interference. As shown by Figures 3 and 4, a REGTCP flow has a slower growth compared to the HSTCP flow. The REGTCP flow takes around 300 seconds to reach the bandwidth limit in congestion avoidance. In comparison, the HSTCP flow reaches this point before 50 seconds (RED case). HSTCP has an oscillatory behavior with a very short period. With DT queuing, drops did not occur until the router buffer overflowed. With RED queuing, the router sends the congestion signal earlier. In the case of an empty network this can lead to lower utilization with RED.

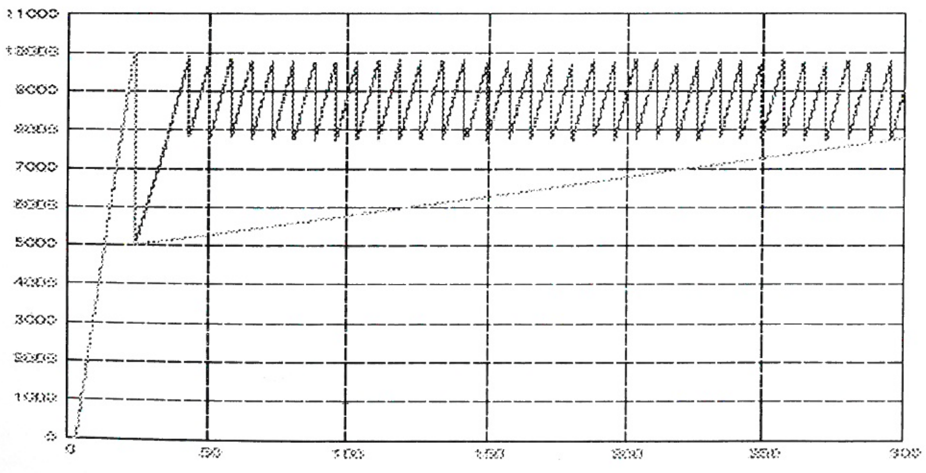

Figure 3: Evolution of Congestion Window for a Single Flow. RED.

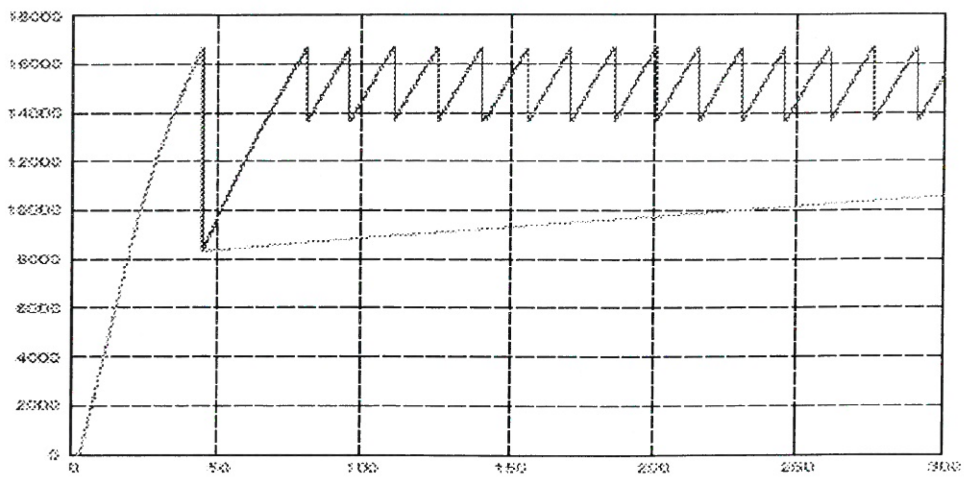

Figure 4: Evolution of Congestion Window for a Single Flow. Drop Tail.

In the following, the performance of ICC is compared with that of HSTCP and STCP. Every experiment uses the same simulation setup described earlier. In order to reduce noise in data sampling, we take measurement only in the second half of each run. We evaluate ICC, HSTCP, and STCP for the following properties: bandwidth utilization, RTT unfairness, and convergence to fairness. For ICC, we use *Smax* = 32, *Smin* = 0.01, and b = 0.125. All high-speed protocols are implemented by modifying TCP/SACK.

Utilization: In order to determine whether a high-speed protocol uses available bandwidth effectively under high-bandwidth environments, we measure bandwidth utilization for 2.5 Gbps bottleneck. Each test consists of two high-speed flows of the same type and two long-lived TCP flows. In each test, the total utilization of all the flows including background traffic is measured. In drop tail, all protocols give approximately 97% utilization, and in RED, ICC consume about 95% of the total bandwidth while HSTCP and STCP consume about 92%. The drop tail experiment consumes more bandwidth because drop tail allows flows to fill up the network buffers.

RTT Fairness: In this experiment, two high-speed flows with a different RTT share the bottleneck. The RTT of flow 1 is 40 ms. The values for the RTT of flow 2 were 40ms, 120ms, and 240ms. The bottleneck link delay is 10ms. Two groups of simulation, each with different values of bottleneck bandwidth: 2.5 Gbps, and 100 Mbps. This setup allows the protocols to be tested for RTT fairness for different window sizes. Around small window sizes, ICC shows the worst RTT unfairness. ICC has window sizes about 7000 (loss rate 0.0003) for 2.5Gbps and 300 (loss rate 0.004) for 100Mbps.

Table 1 shows the results of drop tail only, for the runs in 100Mbps and 2.5Gbps, respectively. The RTT unfairness under RED is close to the inverse of RTT ratio for every protocol.

Table 1. The throughput ratio of protocols under 100Mbps and 2.5 Gbps

Protocol	Bottleneck Bandwidth [bps]	Inverse RTT Ratio		
		1	3	6
ICC	100 M	0.98	11.69	26.94
	2.5 G	0.87	11.92	39.54
STCP	100 M	0.93	18.95	65.12
	2.5 G	0.99	126.89	389.21
HSTCP	100 M	1.02	8.85	38.87
	2.5 G	1.01	29.25	108.2

The performance of ICC does not deteriorate much while HSTCP and STCP have improved RTT unfairness. This is because in 100 Mbps, the window size of all the flows is much smaller than in 2.5Gbps run. Therefore, the degree of synchronized loss is very low. Although the RTT unfairness of ICC gets worse around this window size by the analysis, this deficiency gets compensated by lack of synchronized loss so that it does not have much performance degradation. Overall, its RTT unfairness is much better than HSTCP and STCP. HSTCP and STCP tend to starve long RTT flows under high bandwidth environments.

Fairness: Synchronized loss has impact also on bandwidth fairness and convergence time to the fair bandwidth share. In this experiment, we run 4 high-speed flows with RTT 100ms. Two flows start earlier randomly in [0:60] seconds, and the other two flows start later randomly in [100:160] seconds. The total simulation time is 600 seconds. The bottleneck link bandwidth is 2.5Gbps. For this experiment, the fairness index [13] is measured at each 50-second interval, after the first 100 seconds. This result gives an indication on (1) how fast it converges to a fair share, and (2) even after convergence to a fair share, how much it oscillates around the fair share.

Figure 5 shows the result for drop tail routers. At time around 150 seconds, the second set of two connections just start while the first two connections consuming almost all the bandwidth. Therefore, the difference between these four connections is the largest, and hence the fairness index is the lowest. As the latter two connections get more bandwidth, the fairness index increases. The protocol whose fairness index reaches 1 the fastest has the fastest convergence speed. ICC give the best results, and it quickly converge to the fair share (where the fairness index is close to 1). STCP is the worst, and even after 600 seconds, the fairness index is still below 0.95. The fairness index of all protocols first increases fast, and then slowly reaches to 1. The reason is that the congestion windows of the latter two connections first increase exponentially in the slow-start state; after the first loss event, they enter the congestion avoidance state, and increase slowly as specified by each protocol.

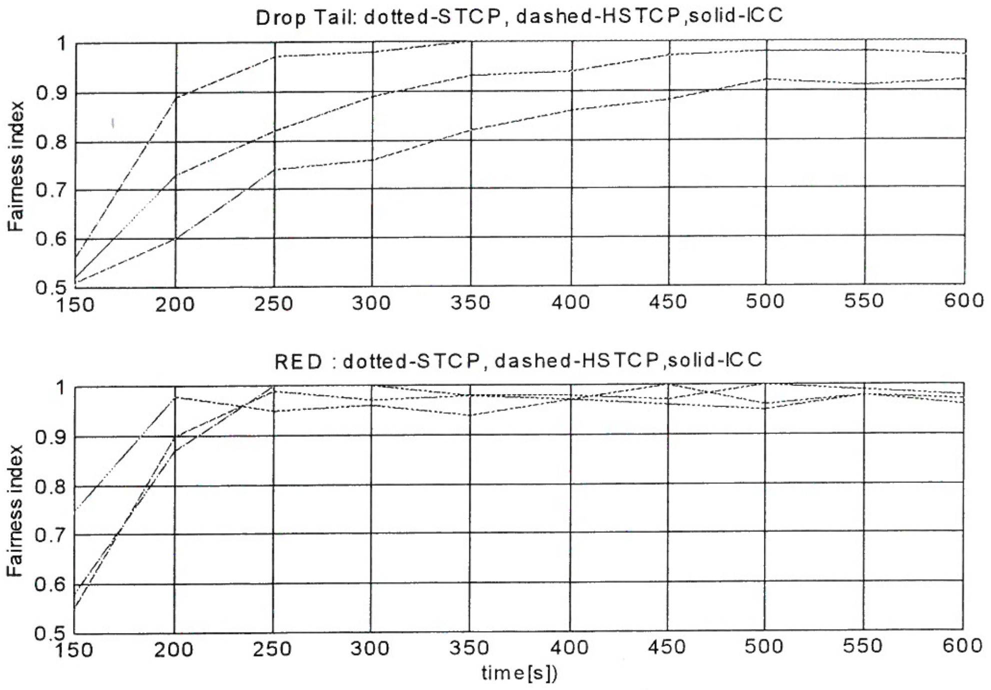

Figure 5: Fairness Index Over Various Time Scales for Drop Tail and for RED.

6. Conclusions

TCP has difficulty in fully utilizing network links with a high bandwidth delay product. HSTCP outperforms TCP in this condition, and has an adaptability that makes an incremental adaption approach easy. HSTCP is easy to deploy avoiding changes in routers and programs.

In this paper RTT fairness is presented as an important safety condition for high-speed congestion control and raise an issue that existing protocols may have a severe problem in deployment due to lack of RTT fairness under drop tail. RTT fairness has been largely ignored in designing high-speed congestion control. This paper presents a new protocol that can support RTT fairness, TCP friendliness, and scalability. Simulation results indicates that it gives good performance on these metrics.

The response function of ICC may not be the only one that can satisfy these constraints. It is possible that there exists a better function that utilizes the tradeoff among these conditions. This is an area of further research.

REFERENCES

1. OMER, I. and T., BASAR. **Control of Congestion in High-Speed Networks**. European Journal of Control, vol.7, 2001, pp.132-144.

2. JACOBSON, V., **Congestion avoidance and control**, Proc. of the ACM SIGCOMM '88 Conference on Communications Architectures and Protocols, vol. 18, 1988, pp. 314–329.

3. LAKSHMAN, T. and U., MADHOW. **Performance analysis of window-based flow control using TCP/IP: Effect of high bandwidth-delay products and random loss**, Fifth Int. Conference on High PerformanceNetworking, 1994, pp. 135–149.

4. XU, L., K., HARFOUSH and I., RHEE. **Binary increase congestion control for fast long-distance networks**, Technical Report, Computer Science Department, NC State University, 2004.

5. SOUZA, E., **A simulation-based study of HSTCP and its deployment**, Lawrence Berkeley National Laboratory, Tech. Rep. LBNL-52549, April 2003.

6. Project, **NS-2 network simulator**, URL http://www.isi.edu/nsnam/ns.

7. FLOYD, S., A., RATNASAMY and S., SHENKER. **Modifying TCP's congestyion control for high speeds**", http://www.icir.org/floyd/hstcp.html, May 2002.

8. FLOYD, S.,**HighSpeed TCP for large congestion windows**, IETF, draft-floyd-tcp-highspeed-01.txt, 2003.

9. KELLY, T., **Scalable TCP: Improving performance in highspeed wide area networks**, submitted for publication, Dec. 2002.

10. KATABI, D., M., HANDLEY.and C., ROHRS. **Internet congestion control for high bandwidth-delay product networks**. ACM SIGCOMM 2002, Pittsburgh, Aug., 2002.

11. 11. GU, Y., HONG, X., MAZZUCCO, M., and GROSSMAN, R. L., **SABUL: A high performance data transport protocol,** submitted for publication, 2002.

12. JIN,C., D., WEI, H., LOW, G. BUHRMASTER, J. BUNN, D. H. CHOE, R. L. A. COTTRELL, J. C. DOYLE, H. NEWMAN, F. PAGANINI, S. RAVOT, and S. SINGH. **FAST Kernel: Background theory and experimental results**. First Int. Workshop on Protocols for Fast Long-Distance Networks (PFLDnet 2003), Feb.3-4, CERN, Geneva, 2003.

13. CHIU, D., and R., JAIN. **Analysis of the increase and decrease algorithms for congestion avoidance in computer networks**, Journal of Computer Networks and ISDN, Vol. 17, No. 1, June 1989, pp. 1-14

14. BANSAL, D., H., BALAKRISHNAN, S., FLOYD, and S., SHENKER. **Dynamic behavior of slowly responsive congestion controls**, Proc.of SIGCOMM, San Diego, California,2001.