

Effects of Applying High-Speed Congestion Control Algorithms in Satellite Network

Xiuchao Wu, Mun Choon Chan, and A. L. Ananda
 School of Computing, National University of Singapore
 Computing 1, Law Link, Singapore 117590
 Email: {wuxiucha, chanmc, ananda}@comp.nus.edu.sg

Extended version of the paper published in ICC 2008

Abstract—In recent years, many high-speed congestion control (HSCC) algorithms have been proposed for utilizing network pipes with huge bandwidth-delay product (BDP), and some of them have also been implemented in popular operating systems. Considering the extremely long round trip propagation delay (RTPD) of a satellite network, it is very likely that these algorithms are triggered when TCP flows pass through satellite network.

But the existing HSCC algorithms are normally evaluated on network pipes with high bandwidth and moderate RTPD (≤ 300 ms). This paper is an attempt to study these algorithms on a simulated satellite network with moderate bandwidth and extremely long RTPD. Their effects on the existing applications, especially World Wide Web (WWW) and the emerging streaming applications, are emphasized. Different queue sizes are also used in simulation with the aim of investigating how to provision satellite link's queue for well accommodating flows driven by HSCC algorithms. Through this study, we find that Compound-TCP, Cubic-TCP, and H-TCP are not suitable for satellite network. Currently, satellite link should use moderate queue size for accommodating flows driven by HSCC algorithms. As for end hosts, when they find that round trip time (RTT) is very long, HS-TCP should be adopted for high throughput while avoiding to hurt the existing applications as little as possible.

I. INTRODUCTION

TCP is the de-facto standard protocol of the Internet for uni-cast reliable data transmission. But it is also well known that TCP's congestion control algorithm[1], AIMD (Additive Increase and Multiplicative Decrease), can not work well on network pipes with huge BDP[2]. On these network pipes, TCP becomes the performance bottleneck and network bandwidth is under-utilized. Based on this observation, many HSCC algorithms, such as HS-TCP[2], BIC-TCP[3], Cubic-TCP[4], H-TCP[5], Fast-TCP[6], and Compound-TCP[7], have been proposed in recent years.

It is very attractive to apply these HSCC algorithms for providing higher throughput to bandwidth-greedy applications, such as FTP and Peer-to-Peer file sharing. Some of these algorithms have been implemented within popular operating systems. For example, Windows Vista of Microsoft adopts

This work is supported by National University of Singapore under research grant R-252-000-203-112.

Compound-TCP and Linux is distributed with several algorithms which can be easily selected through a socket option. These HSCC algorithms are normally triggered when a flow finds that its congestion window (*cwnd*) is larger than a threshold. In HS-TCP and Compound-TCP, this threshold is 38 and 41 segments respectively. It means that these algorithms may run over many network pipes of the Internet.

Today, on a Geostationary Orbit Satellite link, per-carrier bandwidth can be up to 155 Mbps and bit error rate can be 10^{-10} on a very clear sky [8]. When terrestrial network is not feasible, as shown in figure 1, satellite network is now a good choice for connecting two networks far apart. Satellites are being used by ISPs and corporations for network construction and situations during unexpected disruption of transoceanic optical fiber links.

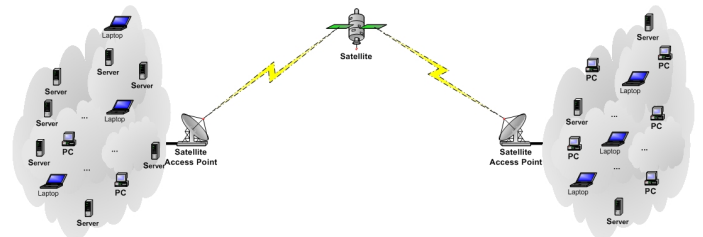


Fig. 1. Satellite Network Scenario

Considering the long RTPD of a satellite link (about 500ms), it is more likely that HSCC algorithms are triggered when flows pass through satellite network. But the existing HSCC algorithms are normally evaluated on highly reliable network pipes with high bandwidth and moderate RTPD (≤ 300 ms) with the focus on their efficiency, convergence, RTT fairness, and friendliness to long-lived TCP flows. Before applying these algorithms in satellite networks, we should make sure that they work well on network pipes with extremely long RTPD, moderate bandwidth, and some packet corruption.

In this paper, several influential HSCC algorithms, HS-TCP, BIC-TCP, Cubic-TCP, H-TCP, and Compound-TCP, are evaluated on a simulated satellite network. Due to the long RTPD of satellite network, user experience of the existing

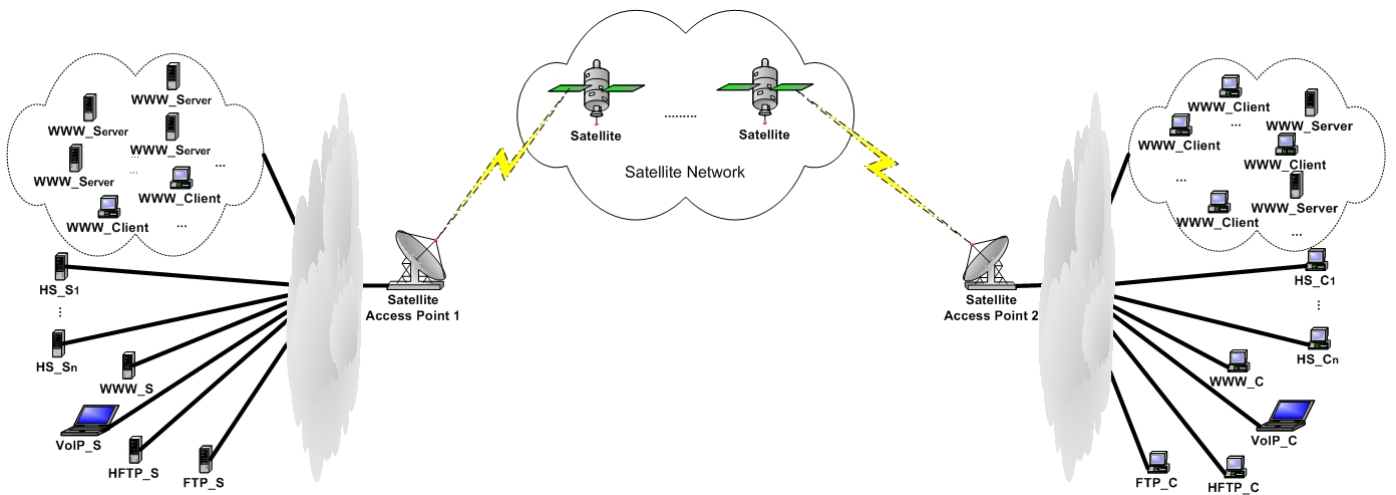


Fig. 2. Network Topology: Two High Speed Networks Connected Through Satellite Network

applications is low when satellite network is passed through. Flows driven by HSCC algorithms inevitably increase the load of satellite network and the existing applications may be hurt. Hence, their effects on the existing applications, especially WWW and the emerging streaming applications, are emphasized in this evaluation. Different queue sizes are also used in simulation with the aim of investigating how to provision queue for well accommodating flows driven by HSCC algorithms.

This paper is organized as follows. Section II presents the motivations and the methodology used in this paper. Simulation results are then presented and analyzed in section III. The paper is concluded in section IV.

II. MOTIVATIONS AND EVALUATION METHODOLOGY

A. Existing Evaluation and Motivations

When a HSCC algorithm is proposed, its authors also evaluate and compare their proposal with pre-existing algorithms. These algorithms are normally investigated in simulated and/or emulated networks whose bandwidth varies from 50Mbps to 1Gbps and RTPD varies from 20ms to 350ms. Efficiency (link utilization) and fairness (fairness, convergence, RTT fairness, etc.) are the main metrics for evaluating these algorithms. When their effects on TCP flows are considered, the throughput of concurrent long-lived TCP flow is used as the metric. It is normally assumed that long-lived TCP flow uses window scale option[9] and there is no buffer limitation. In [3], the authors suggested that it may be more realistic to evaluate their effects on long-lived TCP flows with buffer limitation and/or without support of window scale option.

In [10], HS-TCP is evaluated on a simulated transocean link (bandwidth: 1Gbps, RTPD: 100ms). Simulation results indicate that HS-TCP can utilize this link efficiently, fairly, and friendly to concurrent long-lived TCP flows when there is no large amount of background traffic. In [11], Cubic-TCP is evaluated

and analyzed on a testbed (bandwidth: 1-500Mbps, RTPD: 16-200ms) with the focus on its convergence speed and fairness properties. In [12], Compound-TCP is evaluated and compared with HS-TCP on several Internet paths (RTT: 8.5-151ms).

In [13], BIC-TCP, FAST-TCP, HS-TCP, and H-TCP are evaluated and compared on a testbed. The bandwidth of the emulated bottleneck link varies from 10 to 250Mbps and the emulated RTPD varies from 16 to 324ms. These algorithms are also evaluated when background web sessions consume 1.5% bandwidth of bottleneck link. In [14], BIC-TCP, FAST-TCP, HS-TCP, H-TCP, and Cubic-TCP are evaluated and compared on a emulated bottleneck link (bandwidth: 400Mbps, RTPD: 16-324ms). The authors mainly evaluate the effects of different kinds of background traffic on flows driven by HSCC algorithms.

Hence, these HSCC algorithms have not been evaluated on satellite-like network whose RTPD can be much larger than 500ms. Except [14] and [12], these algorithms are evaluated without large amount HTTP sessions as background traffic. In addition, these evaluations focus on the performance of flows driven by HSCC algorithms. Their effects on TCP flows are only measured by the throughput of long-lived TCP flows and the total throughput acquired by short-lived flows.

In this paper, we try to fill the gap by evaluating these HSCC algorithms on a simulated satellite network with large amount of HTTP sessions as background traffic. Their effects on the existing applications, such as WWW and streaming applications, are put on the first place and are investigated from user experience point of view. The details of our evaluation methodology will be given in the following section.

B. Evaluation Methodology

1) *Network Topology and the Simulated Satellite Network:* In order to evaluate the effects of applying HSCC algorithms in satellite network, the dumbbell network configuration in figure 2 is used for the simulation with NS-2[15]. A satellite network

which is composed by two satellites is used to connect two high speed networks. This scenario is quite realistic since one satellite is not enough to connect two sites far apart, such as Singapore and USA. The bandwidth of the simulated satellite network is 155Mbps and its RTPD is set to 1000ms (two satellites). Considering that satellite network is liable to bad weather and packets are quite large (1500 bytes), packet error rate of the satellite network is set to 10^{-5} . The other links are all highly reliable and very fast (bandwidth: 1Gbps, packet error rate: 0). Hence, the simulated satellite network is the only bottleneck.

2) *Traffic Generated in Simulation:* Between the two computer clouds of figure 2, HTTP sessions are generated according to the connection-based web traffic model [16]. The propagation delay between web nodes and satellite access points are randomly selected between 5ms and 15ms. In the forward path (from Satellite Access Point 1 to Satellite Access Point 2), 800 HTTP sessions are generated per second. In the reverse direction, 200 HTTP sessions are generated per second. Hence, web background traffic consumes quite large amount bandwidth of the simulated satellite network. This scenario is very similar to the running satellite network.

We adopt TCP-Linux patch[17] with latest bug-fixes for comparing these HSCC algorithms in a common TCP implementation. N flows driven by HSCC algorithms are generated between HS_S_i and HS_C_i . Unless specified, N equals to 4. *window_*, NS parameter for socket buffer, is set to 10000 packets so that the sending rate of these flows is only affected by congestion control. In order to mitigate the presence of phase effects, as was done in [3], we set *overhead_*, NS parameter for node processing delay, to 8×10^{-6} . The propagation delay between HS_S_i/HS_C_i and satellite access points are also randomly selected between 5ms and 15ms.

In order to evaluate a HSCC algorithm's effects on existing applications from user experience point of view, a client (WWW_C) continuously accesses a web server (WWW_S) which sends back responses with fixed sizes (1KB, 2KB, 4KB, 8KB, 16KB, 32KB, 64KB, 128KB). The response time of these HTTP transactions is used to evaluate the HSCC algorithm's effects on WWW. A VoIP connection is established between VoIP_S and VoIP_C. We use the ITU G.711 PCM VoIP traffic as our voice source. The data rate during talk spurt is 87.2 Kbps, including the relative protocol headers. Each packet size is 218 bytes. The average burst time is 0.4s, average idle time is 0.6s, and the distribution follows an exponential ON/OFF model. The one way delay from VoIP_S to VoIP_C, its jitter, and packet loss rate are used to evaluate the HSCC algorithm's effect on streaming applications. A FTP connection (HFTP) is established between HFTP_S and HFTP_C. It uses a TCP Agent whose *window_* is set to 100000. Its throughput is used to evaluate the HSCC algorithm's effect on long-lived TCP flows with support of window scale option and without buffer-limitation. Another FTP connection (FTP) is established between FTP_S and FTP_C. It uses a TCP agent whose *window_* is set to 64. Its throughput is used to evaluate the

HSCC algorithm's effects on long-lived TCP flows with buffer limitation and/or without support of window scale option. The propagation delay between the above nodes and satellite access points is 10ms.

Table I summarizes the traffic generated by these nodes and the delay of side links that are used to connect these nodes to satellite access points.

| Nodes | Delay of Side Link | Traffic Type | Traffic Load |
|--|--------------------------|-------------------------|---|
| WWW_Server and WWW_Client nodes | Random Number [5ms-15ms] | Background Web Traffic | 800 Sessions / s (forward path) 200 Sessions / s (backward path) |
| HSCC_S _i HSCC_C _i | Random Number [5ms-15ms] | Long-Lived FTP Flows | 4 Flows (forward path) |
| WWW_S WWW_C | 10ms | HTTP Sessions | 10 Sessions / s (forward path) |
| VoIP_S VoIP_C | 10ms | ITU G.711 PCM Traffic | 1 Connection (forward path) |
| HFTP_S HFTP_C | 10ms | Long-Lived FTP Flow | 1 Flow (forward path) |
| FTP_S FTP_C | 10ms | FTP Flow (small buffer) | 1 Flow (forward path) |

TABLE I
PARAMETERS OF GENERATED TRAFFIC

3) *Experiments:* In order to investigate how to provision queue for well accommodating flows driven by HSCC algorithms, satellite access point 1 uses DropTail queue and queue size is set to 0.02, 0.05, 0.1, 0.2, 0.5, or 1 BDP of the simulated satellite network. For each queue size, six experiments are carried out. HS-TCP, BIC-TCP, Cubic-TCP, H-TCP, and Compound-TCP are used by flows between HS_S_i and HS_C_i in different experiments. TCP (Newreno) is also used by these flows for comparison in the remaining one experiment. For each experiment, simulation runs for 830 seconds and figure 3 shows the start and end sequence of different kinds of flows.

III. SIMULATION RESULTS AND ANALYSIS

In this section, simulation results are presented and analyzed. Instead of the performance of flows driven by HSCC algorithm, we focus on their effects on the existing applications. More specifically, user experiences of WWW, VoIP, and FTP are first presented and compared when satellite network uses different queue sizes and different congestion control algorithms are used by flows between HS_S_i and HS_C_i .

A. WWW, VoIP, and FTP User Experience

1) *WWW User Experience:* For WWW, response time of HTTP transaction is used to measure its user experience. Irrespective of the size of response data, response time of HTTP transaction follows the same pattern. Due to space limitation, figure 5 only present average response time of HTTP transactions whose response data size is 4KB or 64KB.

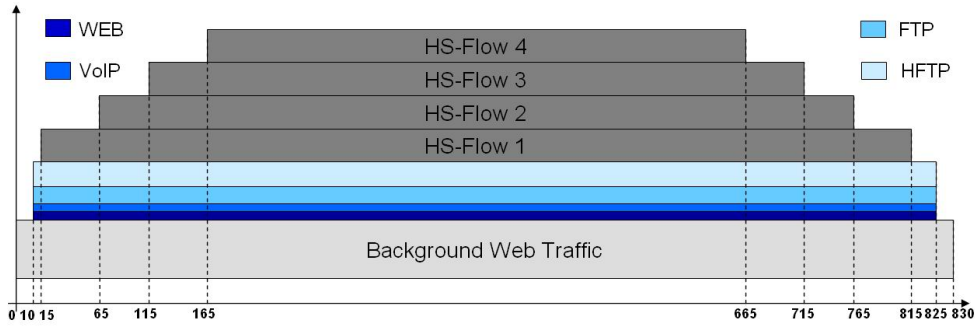


Fig. 3. The start and end sequence of different flows

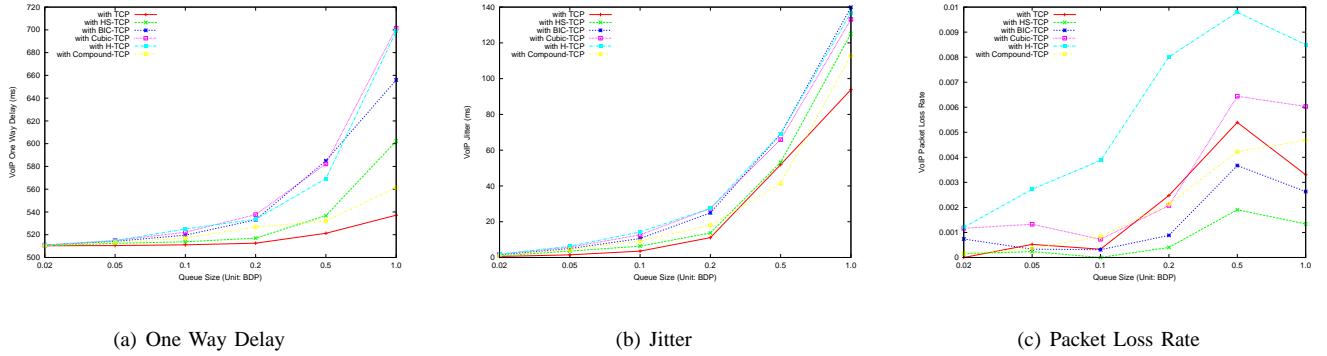


Fig. 4. VoIP User Experience

Figure 5 shows that H-TCP and Cubic TCP always cause much longer response time. When queue is not larger than 0.5 BDP, BIC-TCP and Compound TCP does not obviously increases the response time of HTTP transactions. *Surprisingly, HS-TCP is even better than TCP when queue is between 0.1 and 0.5 BDP*.

2) *VoIP User Experience*: Figure 4 shows the average one-way delay, jitter, and packet loss rate suffered by VoIP packets. Figure 4(a) indicates that compared with TCP, HSCC algorithms always cause longer delay. When queue is large and H-TCP, Cubic-TCP, or BIC-TCP is used, VoIP packets suffer much longer delay. *In this metric, Compound-TCP is better than HS-TCP when queue is large and HS-TCP is better than Compound-TCP when queue is small*. Figure 4(b) indicates that in the metrics of jitter, Compound-TCP is better than other HSCC algorithms when queue is large. It is even better than TCP when queue is 0.5 BDP. Figure 4(c) indicates that when H-TCP are used, the loss rate of VoIP packets is always much higher than other algorithms. Cubic-TCP also brings higher packet loss rate. There may not be enough VoIP packets for accurately measuring packet loss rate. But figure 6(b) does accord with the lines in figure 4(c).

3) *FTP User Experience*: Figure 7 shows the average throughput of FTP and HFTP connections. Figure 7(a) indicates that when H-TCP and Cubic-TCP are used, FTP connections with small socket buffer can be starved. Figure 7(b)

indicates that for HFTP connection, when queue is between 0.05 and 0.5BDP, HS-TCP is the best, H-TCP is the worst, and Cubic-TCP also adversely affects HFTP connection a lot.

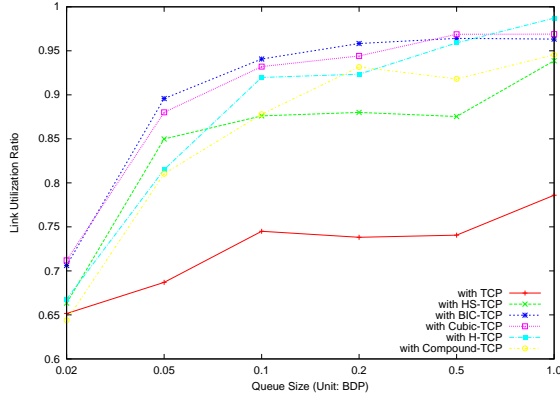
B. Analysis and Discussion

1) *Compound-TCP*: Simulation results indicate that when considering the effects on existing applications, Compound-TCP does not perform obviously better than other HSCC algorithms. The results contradict with the claim that Compound-TCP is very friendly to TCP since it is a delay-based congestion control algorithm and can drive the network to work at the knee[18]. The following three issues may cause this problem.

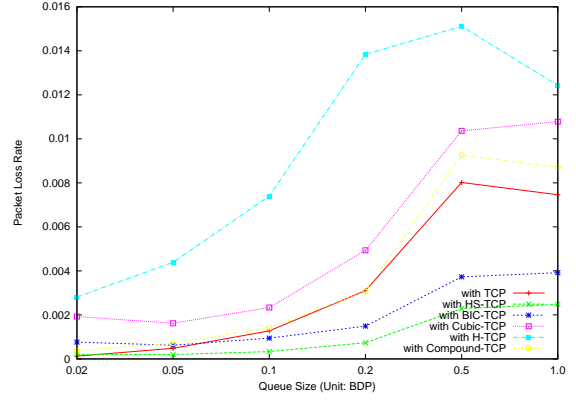
First, Compound-TCP uses equation 1 to detect congestion by queue delay. On a satellite network with extremely long RTPD and moderate bandwidth, RTT_{min} is very long, per flow's $cwnd$ ($cwnd + dwnd$ in CTCP algorithm) is not very large, and $cwnd$ decreases with the increase of flow number. Very large queue is needed to support a small number of Compound-TCP flows. When queue is not large enough, Compound TCP flows increase sending rate binomially and can not drive network to work at the knee.

$$\left(\frac{cwnd}{RTT_{min}} - \frac{cwnd}{RTT_{min} + qdelay} \right) * RTT_{min} > \gamma$$

$$\gamma = 30 \Leftrightarrow qdelay > \frac{30 * RTT_{min}}{cwnd - 30} \quad (1)$$

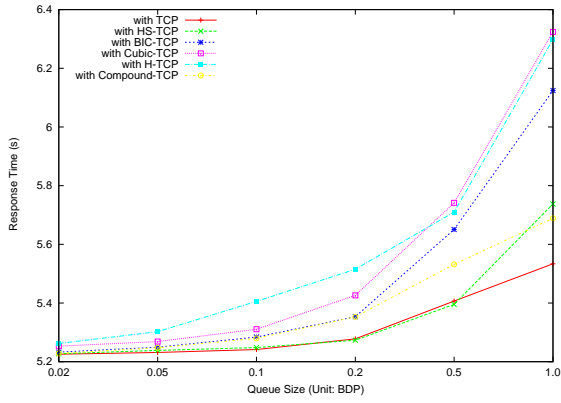


(a) Link Utilization Ratio

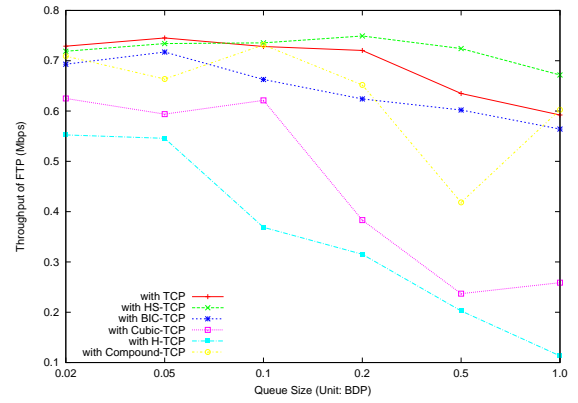


(b) Packet Loss Rate

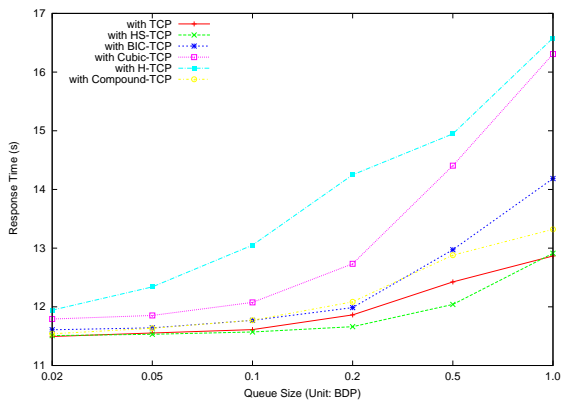
Fig. 6. Utilization Ratio and Packet Loss Rate of Satellite Network



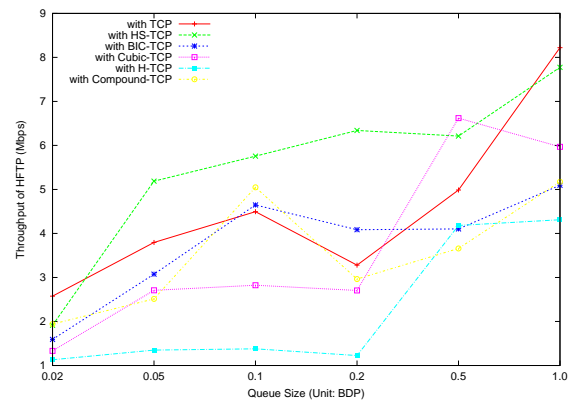
(a) Response Data Size=4KB



(a) FTP



(b) Response Data Size=64KB



(b) HFTP

Fig. 5. WWW User Experience (average response time of HTTP transactions)

Fig. 7. FTP User Experience

Second, Compound-TCP adjusts its sending rate once per RTT. Considering the extremely long RTT (> 1 second) and many background WWW traffic, even when queue is large enough to detect congestion through queue delay, Compound-TCP may not react quickly enough to drive network to work at the knee.

Third, Compound-TCP switches back to TCP for avoiding to be starved by TCP flows. When queue is large, it stays at the knee shortly, switches to TCP, and acts as TCP for a long time.

In a word, Compound-TCP is not a good selection for satellite networks since it monitors queue delay and acquires no benefit.

2) *Loss-Based HSCC Algorithms:* Figure 5 and 4 indicate that when loss-based HSCC algorithms are used, queue should not be larger than 0.2 BDP. Otherwise, user experience of WWW and VoIP will be much worse. The reason is that when segment loss is detected, loss-based HSCC algorithms normally reduce $cwnd$ slightly ($cwnd = cwnd * (1 - \beta)$, and β is normally not larger than 0.2) and drive network to work at persistent congested state. According to figure 6(a), we find that queue should not be too small for accommodating data burst. When queue is 0.02 BDP, all HSCC algorithms can not significantly increase the utilization ratio of satellite network.

When queue is moderate (0.05-0.2 BDP), H-TCP is much worse than other HSCC algorithms. HTTP transaction response time is longer and the loss rate of VoIP packets is higher. H-TCP is also the worst algorithm for cross FTP and HFTP connections. In addition, according to figure 6(a), when H-TCP is used, satellite network utilization ratio is not much higher than other HSCC algorithms. Hence, flows with H-TCP hurt the existing applications without getting much benefit for themselves. The reason is that H-TCP is too aggressive on satellite network. As shown in equation 2, its $cwnd$ additive increase parameter, $\bar{\alpha}(\Delta)$, is designed to be totally independent on RTT for fast convergence and good RTT fairness.

$$\bar{\alpha}(\Delta) = \begin{cases} 1 & \Delta \leq \Delta^L, \\ 1 + 10(\Delta - \Delta^L) + (\frac{\Delta - \Delta^L}{2})^2 & \Delta > \Delta^L. \end{cases} \quad (2)$$

Here, Δ is the elapsed time since last congestion event.

$\Delta^L = 1$ second;

On a link with a 50ms RTPD, it may be acceptable to switch to fast increase mode when one second has passed since the last congestion event. It is too soon for flows that pass through satellite network. On satellite network, $\bar{\alpha}(\Delta)$ is also increased too quickly from RTT point of view. When congestion approaches, $\bar{\alpha}(\Delta)$ of H-TCP tends to be large and many packets will be dropped. Hence, flows driven by H-TCP are more likely to hurt both themselves and other cross traffic.

Cubic-TCP also performs quite bad when considering its effect on existing applications. The reason is that window growth function of Cubic-TCP (equation 3) is also a function of real time. But this function keeps the good property of BIC-TCP. *The closer $cwnd$ is to W_{max} , the slower $cwnd$ is increased.* This function is also related with W_{max} , the product

of RTT and bandwidth acquired when the last congestion event occurred. Hence, Cubic-TCP is a little better than H-TCP, but its performance is still not acceptable.

$$W_{cubic} = C(t - \sqrt[3]{W_{max} * \beta / C})^3 + W_{max} \quad (3)$$

When queue is moderate, both BIC-TCP and HS-TCP are acceptable in the metrics of efficiency (link utilization ratio) and friendliness to the existing applications. In the metric of efficiency, BIC-TCP is better than HS-TCP. In the metrics of friendliness, *HS-TCP is much better than BIC-TCP, and is even better than TCP in some cases.* In satellite networks, we should first make sure that user experience of the existing applications, especially WWW and streaming applications, is not much worse. The throughput of bandwidth-greedy flows should have a much lower priority. Hence, *when end points find that RTT is very long (a signal that satellite network is passed through), they should use HS-TCP for their bandwidth-greedy applications so that they can acquire high throughput while hurting the existing applications as little as possible.*

IV. CONCLUSION

In this paper, we first study several influential HSCC algorithms on a simulated satellite network with a large amount of background web traffic for investigating their effects on existing applications, especially WWW and streaming applications. Through this study, we find that Compound-TCP, Cubic-TCP, and H-TCP are not suitable for satellite networks. Satellite link should maintain a moderate queue (0.05-0.2 BDP). And when end points find that RTT is very long (a signal that satellite network may be passed through), they should use HS-TCP for their bandwidth-greedy applications. In the next step, we plan to compare these HSCC algorithms with congestion control algorithms specially designed for satellite network, such as TCP-Peach[19], etc.

REFERENCES

- [1] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM*, 1988.
- [2] S. Floyd, "Highspeed tcp for large congestion windows," RFC 3649, Dec. 2003.
- [3] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control for fast long distance networks," in *INFOCOM*, 2004.
- [4] I. Rhee and L. Xu, "Cubic: A new tcp-friendly high-speed tcp variant," in *PFLDnet Workshop*, 2005.
- [5] D. Leith, R. Shorten, and Y. Lee, "H-tcp: A framework for congestion control in high-speed and long-distance networks," in *PFLDnet Workshop*, 2005.
- [6] C. Jin, D. X. Wei, and S. H. Low, "Fast tcp: motivation, architecture, algorithms, performance," in *INFOCOM*, 2004.
- [7] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound tcp approach for high-speed and long distance networks," in *INFOCOM*, 2006.
- [8] "Characteristics of satellite link," available online at <http://ai3.asti.dost.gov.ph/sat/chara.html>.
- [9] V. Jacobson, R. Braden, and D. Borman, "Tcp extensions for high performance," RFC 1323, May 1992.
- [10] E. de Souza, "A simulation-based study of highspeed tcp and its deployment," Tech. Report LBNL-52549, UC, Berkeley, Tech. Rep., May 2003.
- [11] D. Leith, R.N.Shorten, and G.McCullagh, "Experimental evaluation of cubic-tcp," in *PFLDnet Workshop*, 2007.

- [12] Y.-T. Li, "Evaluation of tcp congestion control algorithms on the windows vista platform," Tech. Report SLAC-TN-06-005, Stanford University, Tech. Rep., Jun. 2006.
- [13] Y.-T. Li, D. Leigh, and R. N. Shorten, "Experimental evaluation of tcp protocols for high-speed networks," *IEEE/ACM TON*, vol. 15, Oct. 2007.
- [14] S. Ha, L. Le, I. Rhee, and L. Xu, "Impact of background traffic on performance of high-speed tcp variant protocols," *Computer Networks*, vol. 51, pp. 1748–1762, 2007.
- [15] "Ns2 network simulator," <http://www.isi.edu/nsnam/ns/>.
- [16] J. Cao, W. S. Cleveland, Y. Gao, K. Jeffay, F. D. Smith, and M. Weigle, "Stochastic models for generating synthetic http source traffic," in *IEEE INFOCOM*, 2004.
- [17] D. X. Wei and P. Cao, "Ns-2 tcp-linux: An ns-2 tcp implementation with congestion control algorithms from linux," in *ACM ValueTools - Workshop of NS-2*, 2006.
- [18] R. Jain, "A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks," *ACM SIGCOMM CCR*, vol. 19, pp. 56–71, Oct. 1989.
- [19] I. Akyildiz, G. Morabito, and S. Palazzo, "Tcp-peach: A new congestion control scheme for satellite ip networks," *IEEE/ACM TON*, vol. 9, 2001.