Adaptive Logarithmic Increase Congestion Control Algorithm for Satellite Networks

Minsu Shin[1], Mankyu Park[1] , Deockgil Oh[1], Byungchul Kim[2]and Jaeyong Lee[2]
[1] Dept. of Satellite Wireless Convergence, Electronics and Telecommunications Research Institute
Daejeon, 305-700, S.KOREA
[e-mail: {msshin, neomkpark, dgoh}@etri.re.kr]
[2] Dept. of Information and Communications, Chungnam National University
Daejeon, 305-764, S.KOREA
[e-mail: {byckim, jyl}@cnu.ac.kr]
*Corresponding author: Minsu Shin

Received February 17, 2014; revised May 9, 2014; accepted June 28, 2014; published August 29, 2014

Abstract

This paper presents a new algorithm called the adaptive logarithmic increase and adaptive decrease algorithm (A-LIAD), which mainly addresses the Round-Trip Time (RTT) fairness problem in satellite networks with a very high propagation delay as an alternative to the current TCP congestion control algorithm. We defined a new increasing function in the fashion of a logarithm depending on the increasing factor α, which is different from the other logarithmic increase algorithm adopting a fixed value of $\alpha = 2$ leading to a binary increase. In A-LIAD, the α value is derived in the RTT function through the analysis. With the modification of the increasing function applied for the congestion avoidance phase, a hybrid scheme is also presented for the slow start phase. From this hybrid scheme, we can avoid an overshooting problem during a slow start phase even without a SACK option. To verify the feasibility of the algorithm for deployment in a high-speed and long-distance network, several aspects are evaluated through an NS-2 simulation. We performed simulations for intra- and inter-fairness as well as utilization in different conditions of varying RTT, bandwidth, and PER. From these simulations, we showed that although A-LIAD is not the best in all aspects, it provides a competitive performance in almost all aspects, especially in the start-up and packet loss impact, and thus can be an alternative TCP congestion control algorithm for high BDP networks including a satellite network.

Keywords: computer networks, wireless networks, wireless communications, congestion control, satellite network

http://dx.doi.org/10.3837/tiis.2014.08.013

1. Introduction

Satellite networks are becoming an important candidate in information and communication infrastructures as they provide several advantages to wireless Internet communications. They generally provide wider coverage and higher bandwidth, and can be deployed relatively faster than other terrestrial wireless networks. Moreover, continuous efforts have been made to incorporate satellite systems with terrestrial networks for the purpose of higher service availability [1], and new types of applications requiring a wideband transmission in a channel are emerging nowadays [2]. These trends make satellite networks more attractive than ever before. Nevertheless, satellite links pose some challenges to the congestion control operation of the Transmission Control Protocol (TCP) [3]. The main challenges to the TCP performance of satellite networks are the long Round-Trip Time (RTT), which is generally around 600 msec in bi-directional networks over a geostationary satellite, and the presence of a high packet loss rate by random wireless errors, which cause a spurious TCP congestion control.

TCP Reno [4], TCP NewReno [5], and SACK TCP [6] are the standard versions of TCP congestion control protocols currently deployed on the Internet, and they have achieved great success in performing congestion avoidance and control. The key feature of standard TCP is its congestion avoidance phase, which uses the additive increment multiplicative decrement (AIMD) algorithm [4]. Being a window-based algorithm, TCP controls its sending rate by maintaining a window size variable, *W*, which limits the number of unacknowledged packets in the network from a single user. When a packet loss is detected, the TCP sender decreases its sending window by half. On the other hand, the TCP sender increases its sending window by one when a packet is successfully delivered. Under this algorithm, senders gently probe the network for available bandwidth by cautiously increasing their sending rates, and sharply reduce their sending rates when congestion is detected.

Based on TCP feature analyses over the past many years, several performance issues faced by TCP/IP-based applications on satellite links have been reported. Their performance is limited by the delay and probability of bit errors inherent in geosynchronous satellite systems. These limitations are becoming more critical as new satellite systems offer much higher data transmission rates than those available in the past.

Motivated by the observations above, we are focusing more on the impact of RTT in the protocol design. After the “Slow Start (SS)” phase, standard TCP enters the “Congestion Avoidance (CA)” phase, where the congestion window value is increased approximately linearly by one for every RTT. As a result, TCP increases its sending rate proportionally to 1/RTT, making small RTT flows more aggressive than ones with a large RTT [5], [7]. Such a behavior leads to underutilization and RTT unfairness issues when flows with a large RTT are involved in a network.

Many contributions have been presented to enhance the TCP performance in high-speed networks with long distances. While these algorithms provide their own advantages in various performance metrics, most of them have an impact on the RTT of the flows. Even among the variants adopting a logarithmic increase, i.e., LIAD [8], LogWestwood+ [9], BIC [10], etc., no variants have presented this RTT fairness issue clearly. Herein, we present the dependence on the RTT and packet loss rate of the LIAD approach from its performance analysis in **Fig. 1**. As shown in **Fig. 1**, the average sending rate of the LIAD approach is decreased as the RTT of a flow increases almost linearly.

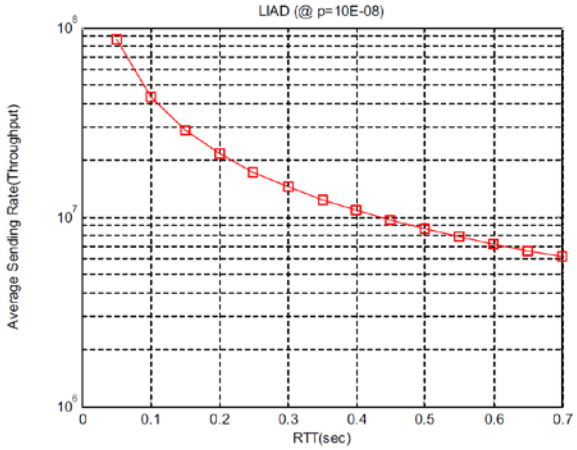


Fig. 1. Average sending rate of LIAD in terms of RTT (@PER=10E-08)

The paper is structured as follows. In section 2, we review the existing variants. In section 3, we present our proposed congestion control protocol. Next, we provide NS-2 simulation results using several different metrics in section 4. Finally, in section 5, we offer a summary conclusion of this paper.

2. Related Works

As we mentioned previously, several new protocols have been introduced to replace standard TCP in high-speed networks with a long RTT. Since conventional TCP mechanisms have been designed to be well operated in wired networks with moderate network capacity and very few link errors, in which most packet losses are due to network congestion, they consequently show inefficiencies and performance degradations in high bandwidth delay networks with a relatively high number of link errors. A lot of effort has been devoted for TCP to operate well even in these different environments, and we now briefly review the existing TCP modifications to see how they work for such networks.

Several contributions are proposed to address TCP protocol issues with wireless links. TCP-Westwood [12] tried to identify the cause of packet loss through an effective end-to-end bandwidth estimation, such that it can avoid an unnecessary decrease of the congestion window size and thus achieve better fairness and friendliness than Reno over the lossy link. However, its dependency on the accuracy of bandwidth estimation should be verified in various network scenarios, and the estimation performance can be improved by adopting more sophisticated delay measurement algorithms [13]. TCP Veno [14] utilizes the buffer estimation scheme of TCP Vegas for network congestion detection to differentiate the cause of packet loss over a wireless access network. It modifies the increase and decrease policies of TCP Vegas using this congestion state information, and thus it increases the congestion window more conservatively during a congestion state, and decreases its congestion window to 80% of the current size even in the event of a loss if the buffer utilization is not excessive. Although TCP Veno improves the problem of proactive approaches in terms of fairness, there is no significant effect over the standard algorithms [17]. TCP Jersey [15] and its enhancement, TCP-NJ [16], have also been proposed as new TCP schemes capable of distinguishing a wireless packet loss from congestion losses. TCP Jersey computes the available bandwidth once every RTT using time-sliding window estimation, and this available bandwidth is used to

set the optimum congestion window. In addition, TCP-NJ enhances the performance of this bandwidth estimation such that it can be immune to the reverse path conditions over which ACK packets are delivered. In addition to the bandwidth estimation, they have considered a congestion warning signal from the network as another criterion for loss differentiation. This feature imposes a critical limitation in their wide deployment in real networks, as it requires all routers in the network to be configured. Another type of TCP variant has been proposed to enhance TCP performance in a satellite network, which typically has a very large bandwidth and delay product. TCP-Peach [18] is a new congestion control scheme for improving the goodput performance and fairness in satellite networks by substituting Slow Start with Sudden Start and Fast Recovery with Rapid Recovery in the traditional TCP protocol. These two new algorithms are based on the use of dummy segments, which are low-priority segments that do not affect the network traffic, to probe the availability of network resources. TCP-Peach+ [19] enhanced its previous version by introducing NIL segments instead of dummy segments. Since they carry unacknowledged information, NIL segments can be used for error recovery as well as probing the unused capacity of the network in two new algorithms, called Jump Start and Quick Recovery. TCP-Peach and TCP-Peach+ use redundant segments with low priority and require all the routers to support a priority mechanism for their intended improvements. Moreover, for the generation and identification of the redundant segments, the sender and receiver modifications are necessary, and can lead to a deployment problem in a real network. To remove the performance dependence on RTT in heterogeneous networks including a long RTT link such as a satellite connection, TCP Hybla [20] was presented. The basic idea of modifications to the standard congestion control rules is that long RTT connections have the same instantaneous transmission rate with a comparatively fast reference TCP connection with a short RTT, such as wired connections. To this end, TCP Hybla introduces the normalized factor, defined as the ratio between the actual RTT and the round trip time of the reference connection, to make the congestion window of a long RTT connection increase exponentially. TCP-Cherry [21] was proposed to improve TCP performance over satellite IP networks under increased link errors by introducing a different type of low-priority probing packet, called a supplement segment, which carry data that are not yet transmitted. Using this low-priority probing packet, TCP-Cherry proposed two new algorithms, Fast-Forward Start and First-Aid Recovery, replacing Slow Start and Fast Recovery in Reno to increase the congestion window quickly from the connection start and differentiate the cause of packet loss, respectively. Even though it was shown to have a better performance in terms of goodput and fairness, the same limitations as described in TCP-Peach and TCP-Peach+ can be expected.

To address the under-utilization and RTT unfairness problems in high-speed and long-delay networks, there are other approaches used to make their congestion windows increase more rapidly when the current window size is small. For these purposes, BIC [10] adopted binary searching for finding the optimal window size, in its Rapid Convergence phase, by computing repeatedly the midpoint between two boundary points, W_{min} and W_{max}, and setting it to either W_{min} or W_{max} according to the occurrence of the loss event. This technique allows bandwidth probing to be more aggressive initially, and becomes less aggressive as the current window size becomes closer to W_{max}. As a result, the increase function becomes logarithmic. CUBIC [11] is an enhanced version of the TCP BIC, and is less aggressive at startup, avoiding the additive increase by adopting a cubic function of the elapsed time since the last packet loss event for its congestion window update rules. This was later supported in terms of RTT fairness and utilization through various experimental studies, although several concerns were raised [22]. LogWestwood+ [9] proposed a logarithmic increase mechanism, in addition to the adaptive decrease of Westwood+, for less sensitivity of RTT and high utilization of network

capacity. Since it increases the congestion window in a similar way as BIC, the congestion window increases rapidly when the current value is small, and gently increases when approaching an estimated maximum value. However, LogWestwood+ has not been sufficiently verified for RTT fairness, while having good friendliness characteristics with standard NewReno. Another protocol adopting a logarithmic increase was presented in LIAD [8], which mainly addresses the impact of high bit errors in wireless networks. LIAD proposes an accurate prediction of the initial ssthresh value and a change in the adaptive decrease factor according to the congestion level, while maintaining the same increase function of LogWestwood+. As a result of involving two new algorithms, LIAD can provide better goodput performance and friendliness behavior with a high wireless error rate than other logarithmic increase-type protocols, such as LogWestwood+, BIC, and CUBIC. However, it is not evaluated in detail for RTT fairness.

And another interesting studies to allow self-organization for individual nodes by adjusting their control parameters under the varying channel environments [24]. Although it presents modelling wireless networks with CSMA, its control conept could be applied for TCP friendliness which is one of main considerations to address in the paper.

3. The Proposed A-LIAD Algorithm

As mentioned above, since the TCP protocol dates back to the early and rather low-speed wired networks, it requires an adaptation taking into account the network characteristic of a large BDP in order to maintain high bandwidth utilization while being fair with existing widely deployed TCP solutions.

Thus, we propose an adaptive logarithmic increase and adaptive decrease (A-LIAD) algorithm. The conceptual idea of an adaptive logarithmic increase function is that each congestion epoch should have the same time duration independent of the RTT. To this end, during each congestion epoch, the adaptive logarithmic increase algorithm adjusts the increasing parameter, α, in the function of the RTT so that a long RTT flow increases its congestion window faster than a short RTT flow. During a slow start phase, A-LIAD adopts the same exponential increase mechanism during the first half period of the slow start, and the logarithmic increase during the next half period of the start. With this hybrid scheme, we can avoid the overshooting problem, which is known to result in multiple packet losses and thus severe performance degradation during a slow start phase. During the congestion avoidance phase, delay-based max-probing is performed as a first step to find an appropriate W_{max} value that the network can hold without any expected packet losses. The process then enters the adaptive logarithmic increase step until a packet loss occurs. When a packet loss is detected by three duplicated ACKs, the protocol reduces its window through a decrease parameter, β, which is adaptively determined based on the current RTT in the final step. A fast retransmit and fast recovery are performed in the same way as in the standard TCP. The general congestion window dynamics of A-LIAD are shown in **Fig. 2**.

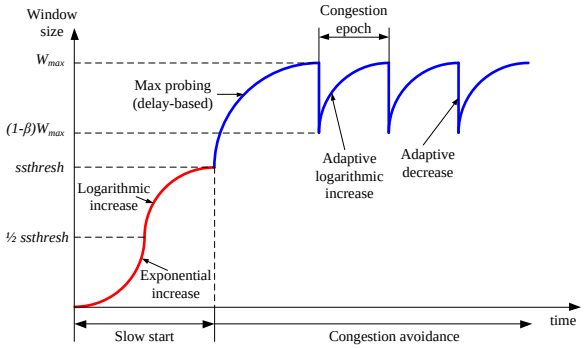


Fig. 2. Window dynamics of the proposed protocol

3.1 Slow start behavior

A-LIAD adopts a hybrid increase function for a slow start phase. When a flow starts, it estimates the network bandwidth delay product (BDP) using Hoe's estimation, and uses this estimate as the initial ssthresh. By repeating this estimation every RTT during a slow start, A-LIAD can update its adequate level of ssthresh such that it can adapt to the network variation even if a new flow sharing the same bottleneck link enters. This process continues until the end of the slow start phase. A-LIAD adopts the same exponential increase mechanism until the first half period of the slow start (cwnd< (1/2)ssthresh), while the logarithmic increase is used during the next half period of the slow start ((1/2)ssthresh<cwnd<ssthresh). With this hybrid scheme, we can avoid the overshooting problem, which is known to result in multiple packet losses and thus a severe performance degradation during a slow start phase.

Denoted by $W(t)$, the congestion window is expressed in segments, and through t_γ, the time at which half of the ssthresh value γ is reached, we use the following rule for a slow start phase:

$$W(t) = \begin{cases} 2^{t/RTT}, & 0 \le t < t_\gamma \quad ,SS \\ \gamma + \gamma\{1 - \gamma(1/2)^{t/RTT}\}, & t_\gamma \le t < t_{2\gamma} \quad ,SS. \end{cases} \tag{1}$$

By expressing the value W of the congestion window in MSS units, the cwnd update rules for each ACK reception are given by

$$W_{i+1} = \begin{cases} W_i + 1, & 0 \le t < t_\gamma, \quad SS \\ W_i + \left(\frac{\gamma}{W_i} - \frac{1}{2}\right), & t_\gamma \le t < t_{2\gamma}, \quad SS. \end{cases} \tag{2}$$

We will show in the simulation that this kind of increase function, called an S-function, can avoid multiple packet losses owing to the known problem of a slow start overshoot by reducing its increasing window rate around the network BDP value.

3.2 Congestion avoidance behavior

During the congestion avoidance phase, the A-LIAD protocol performs a logarithmic increase step, adjusting its increasing rate adaptively according to its RTT after a delay-based max-probing step, followed by an adaptive decrease step when a packet loss is detected based on the reception of three consecutive duplicate ACKs.

For a logarithmic increase as the first step in the congestion avoidance phase, the maximum window size, W_{max}, should be determined. The only information we can use to predict an impending congestion and the resulting occurrence of a packet loss is a delay. Packet delays increase abruptly just before a loss, although they change very slowly even when network congestion increases [23]. When the ratio of the current RTT to the minimum RTT is smaller than the threshold δ, the congestion window increases rapidly following a logarithmic increase with an increasing factor α = 2 since this region can be considered far from the severe congestion level. On the other hand, the congestion window increases slowly with a linear increase when the delay ratio becomes larger than the threshold since the congestion loss is impending. From the simulation, δ = 1.93 gives the best results, and thus this value is used throughout the simulation. The window update rule for each ACK reception is presented in (3).

$$W_{i+1} = \begin{cases} W_i + (ssthresh/W_i) - \left(\frac{1}{2}\right), & \frac{RTT}{RTT_{min}} < \delta \\ W_i + 1, & otherwise. \end{cases} \tag{3}$$

After determining the maximum window size, W_{max}, through the delay-based max-probing step, the protocol enters an adaptive logarithmic increase step after reducing the window through the decreasing parameter, β.

Thus, the protocol starts the adaptive logarithmic increase step from $(1-\beta)W_{max}$ and increases the window until a packet loss occurs. We call the time period between the last packet loss and current packet loss the congestion epoch. As shown in **Fig. 2**, during each congestion epoch, the congestion window increases from $(1-\beta)W_{max}$ to W_{max}.

Denoted by $W(t)$ with the elapsed time since the last packet loss occurred, the congestion window is expressed in segments as

$$W(t) = W_{max}\left(1-\beta(1/\alpha)^{t/RTT}\right), \quad t_c \le t, \quad CA. \tag{4}$$

where t_c denotes the time when the last packet loss takes place. Note that the increasing rate is governed by the parameter α. Different from the existing protocols adopting a logarithmic increase with a fixed increasing rate of $\alpha = 2$, which leads to a binary increase [8]-[10], we adaptively change the increasing parameter α in the RTT function. Increasing parameter α should be higher, and thus the protocol should be able to increase the window faster as the RTT becomes longer such that consequently the protocol can provide good RTT fairness when flows with different RTTs are competing in the same bottleneck link. The method for determining the value of α in terms of a flow's RTT is detailed in section 3.

We need to give the per ACK increment rules according to the window increasing function in (4) to react to each ACK reception. We derive the total number of packets sent in the k-th RTT, W_k, and then convert this number into the per-ACK increment.

$$W_{k+1} = W_k + \left(1-\frac{1}{\alpha}\right)(W_{max} - W_k). \tag{5}$$

Equation (5) shows the per-RTT increment rule of cwnd. Now, we can then derive the per-ACK increment rules for the i-th ACK reception as follows

$$W_{i+1} = W_i + \left(1-\frac{1}{\alpha}\right)(W_{max} - W_i)/W_i. \tag{6}$$

For an adaptive decrease of the congestion window upon a packet loss event happening after entering an adaptive logarithmic increase phase, A-LIAD takes the network congestion level into account in a similar way as H-TCP. Since it is difficult to assume that the bottleneck buffer size is equal to the bandwidth-delay product in a high-speed network, setting the decrease parameter as $\beta = 0.5$ is not practical. Therefore, we adopt an adaptive decrease mechanism such that the throughput is matched before and after a decrease, and decrease parameter β can be calculated as in (7).

$$\beta = 1 - \frac{RTT_{min}}{RTT_{max}}, \tag{7}$$

where RTT_{min} and RTT_{max} are the minimum and maximum RTTs experienced by the flow, respectively.

3.3 Analysis of the growth function in CA

This section analyzes A-LIAD's growth function during the congestion avoidance phase, as indicated in **Fig. 3**. If the increasing parameter α of all flows is the same regardless of their RTTs, the duration of the congestion epoch for a short RTT flow will be shorter than that for a long RTT flow. For this reason, the window of a long RTT flow will increase slower than that of a short RTT flow, and therefore the RTT fairness characteristic worsens.

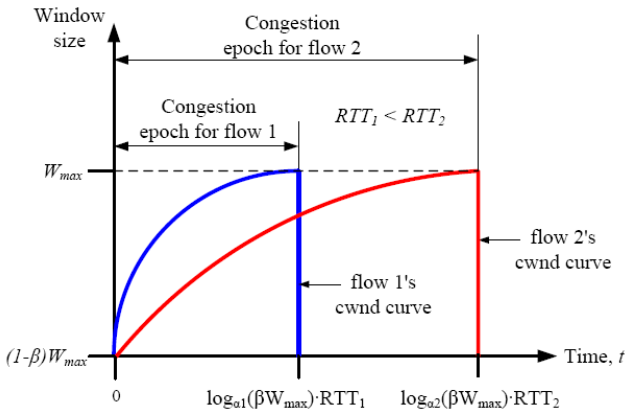


Fig. 3. Window dynamics of two flows with different RTTs

The basic concept of A-LIAD is to adjust the increasing parameter α in terms of the RTT. The window growth function of A-LIAD increases more aggressively with a higher α, yielding a faster increase. Therefore, if α can be adjusted to be proportional to the RTT, it can guarantee RTT fairness between flows with different RTTs, and provide a significant advantage for the protocol to be used in recent high BDP networks with wireless links. To the best of our knowledge, this property is unique, and the existing logarithmic increase protocols cannot provide this characteristic since they adopt a fixed increasing parameter α= 2, leading to the same movement as a binary increase. This is the main point of using A-LIAD.
The objective here is to determine α in the RTT function, and to this end, the window dynamics of A-LIAD are analyzed.

When the congestion window increases from $(1-\beta)W_{max}$ to W_{max}, the total number of RTTs, N, within a congestion epoch is

$$N = log_{\alpha}(\beta \cdot W_{max}), \tag{8}$$

and the throughput R of the increasing period, $N \cdot RTT$, can be computed by

$$\begin{aligned} R &= \frac{Y}{N \cdot RTT} \\ &= \frac{Y}{N \cdot RTT}\left\{W_{max} - \left(\frac{\alpha}{\alpha-1}\right)\beta \cdot W_{max}\right\} \\ &= \frac{W_{max}}{N \cdot RTT}\left\{(N+1) - \left(\frac{\alpha}{\alpha-1}\right)\beta\right\}. \end{aligned} \tag{9}$$

where Y denotes the total number of TCP segments sent in the period, $N \cdot RTT$.

In (9), the response function of the protocol, which represents the average sending rate during a congestion epoch, needs to be independent of the round-trip time. For this purpose, we have to select the value of α that can absorb the effect of the RTT so that each congestion epoch has the same duration of time regardless of its RTT. In other words, we have to be able to reduce the duration to a certain level by adjusting the α value according to the RTT when the RTT of a flow is long. This condition derives the following equation.

$$\frac{log_{\alpha}(\beta \cdot W_{max}(\alpha))}{log_{\alpha_{ref}}(\beta \cdot W_{max}(\alpha_{ref}))} = \frac{RTT_{ref}}{RTT}. \tag{10}$$

where $W_{max}(\alpha)$ and $W_{max}(\alpha_{ref})$denote the W_{max}values when their increasing factors are α and α_{ref}, respectively.

We need to decide how much we should reduce the congestion epoch duration according to the RTT, and therefore determine the reference levels to which we want to reduce the congestion epoch duration. In A-LIAD, we try to keep the performance of LIAD when the RTT is 75 ms, and thus α_{ref} = 2 and RTT_{ref} = 0.075. With the reference values, we can reduce (10) into the following form:

$$log_{\alpha}(\beta \cdot W_{max}(\alpha)) = 1.5787/RTT. \tag{11}$$

However, W_{max} is the function of α, and thus this equation is difficult to solve in a closed form. To construct a mathematical function for α, we use the curve fitting method, and as a consequence, derive α as the function of the RTT.

4. Performance Evaluation

4.1 Network model

This section first defines the network model for an NS-2 simulation. Several performance metrics, including goodput, fairness, and friendliness, are then defined for evaluations under different topologies in heterogeneous wireless networks.

A network topology is shown in **Fig. 4**. The satellite TCP connections consist of wired links followed by a satellite link, while the wired background traffic uses entirely wired paths. All connections share an R1-R2 bottleneck link, whose bandwidth was deliberately limited to 10 or 50 Mbps according to the metrics.

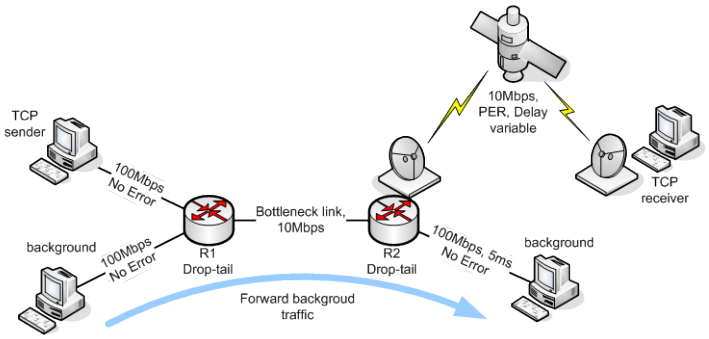


Fig. 4. Network model used for the simulation

4.2 Start-up performance

In this section, we evaluate the ability of each TCP variant to exploit the available bandwidth of a GEO satellite network under the most favorable conditions (i.e., in the absence of congestion owing to wired cross traffic and wireless link losses). To evaluate the start-up dynamics, the throughput in **Fig. 5** is measured at different elapsed times.

As a general comment, all protocols proposed for a high-speed and long-distance network present a fast exploitation of bandwidth during a start-up duration of 30 sec. CUBIC, BIC, STCP, and HSTCP have the highest throughput during the start-up phase. While the proposed A-LIAD has the second highest throughput during the start-up, it also has about 80% of the bandwidth, which is an acceptable level since the difference between CUBIC and A-LIAD is about 3% of the bandwidth.

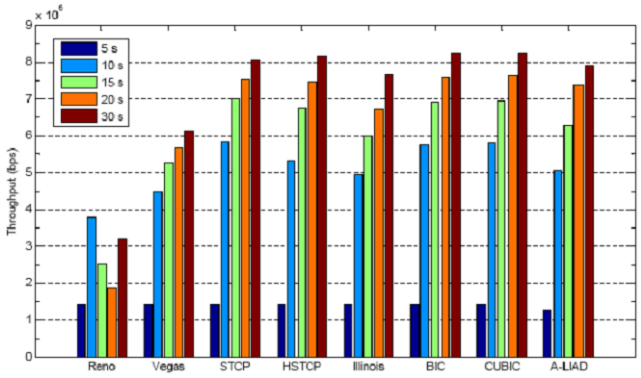


Fig. 5. Throughput performance at the start-up of a GEO satellite connection

We also show the congestion window dynamics of several TCP protocols in **Fig. 6** to demonstrate how to respond at the start-up. In particular, CUBIC operates at the network capacity level, including the link capacity and network buffer, since its decrease factor β takes a smaller value than other variants. Moreover, note that the SACK option is enabled for all variants such that multiple losses during a slow-start phase can be nearly recovered within a few rounds. Without the SACK option, which was not originally included in the variants, the start-up performance would suffer from multiple losses during a slow-start phase.

To further investigate the start-up performance in terms of the overshooting problem during a slow-start phase, let us examine the cwnd dynamics given in **Fig. 7**. It is known that standard TCP has an overshooting problem during a slow-start phase since its exponential increase is so

aggressive that cwnd becomes almost twice the BDP, causing multiple losses and thus many retransmissions and timeouts. The hybrid scheme adopted by A-LIAD tries to reduce the increasing rate at around ssthresh estimated in the initial stage of the connection by introducing a logarithmic increase triggered after the second half of ssthresh. From the simulation evaluation, A-LIAD does not experience an overshooting during the SS phase and enters the CA phase at the proper level, as shown in **Fig. 7(a)**, while TCP-W and CUBIC are shown to experience multiple window reductions after the end of the SS phase in **Fig. 7(b)**. If the SACK option is not enabled for the simulation, the performance of TCP-W and CUBIC during the start-up phase will be affected by these multiple losses more severely than the results in **Fig. 7(b)**. This property of the hybrid scheme leads to the avoidance of multiple losses even without the SACK option. This becomes important for short transfer flows since they make up more than 70% of Internet traffic.

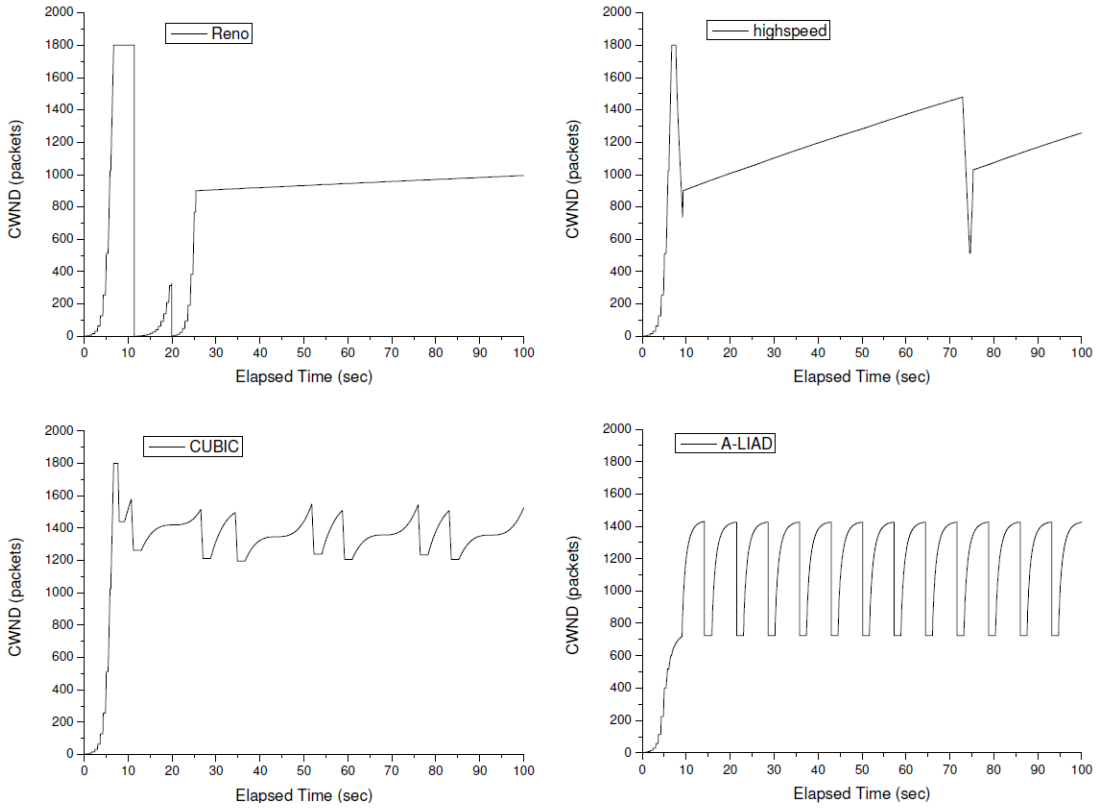
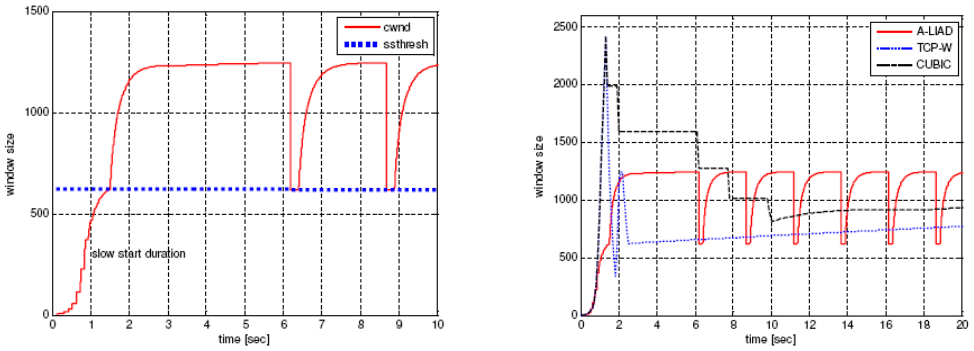


Fig. 6. Comparison of cwnd dynamics at a start-up



(a) window dynamics of A-LIAD (b) comparison of window dynamics with other variants

Fig. 7. Start-up performance of A-LIAD (a) and the comparison with other variants (b)

4.3 Fairness

In this section, we conduct simulations in the same multiple flow environment with C = 50 Mbps RTT = 100 ms, and varying PERs of 0%, 0.1%, and 1% in order to ensure the intra-fairness property in the presence of packet losses. Even if some flows take more than max fair rate, as shown in **Fig. 8**, most flows share almost an equal amount of bandwidth on average over each PER level, and Jain's fairness index of each case is around 99.5%.

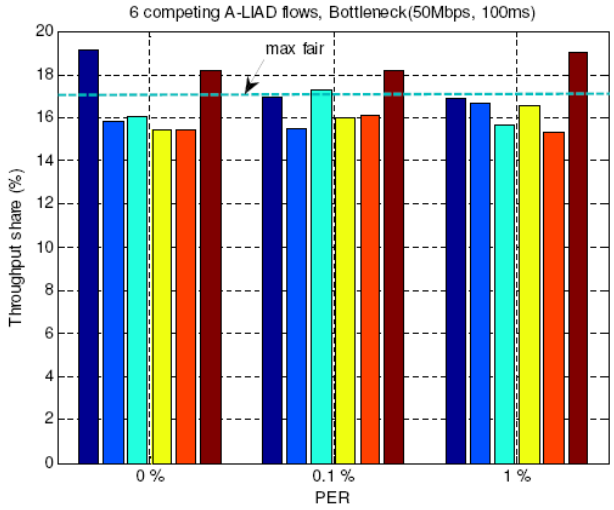


Fig. 8. Intra-fairness: Throughput of 6 competing A-LIAD flows in terms of PER

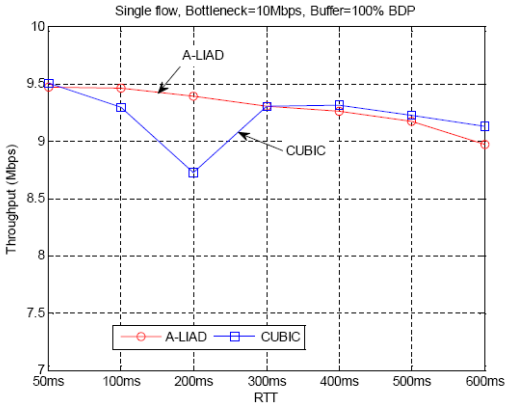


Fig. 9. RTT fairness: throughput performance in terms of RTT.

Next, we evaluate the RTT fairness of the proposed A-LIAD. One of the design goals is for A-LIAD to have less sensitivity to a long RTT by adapting increasing factor α proportionally to the RTT. A simulation was performed with a single flow at C = 10 Mbps for various RTTs, and the results are compared with CUBIC in **Fig. 9**. Within the range of 50 to 600 msec, the results show that A-LIAD is almost unaffected by the RTT variations.

4.4 Impact of packet loss

In this section, the effects of wireless channel errors are investigated. In **Fig. 10**, variable PERs are considered for a GEO satellite connection with RTT = 600 msec in the absence of competing flows. A PER value of 0.1% significantly affects all variants, including those designed to be error resilient, such as Westwood. This negative effect is dramatically increased for a very high PER (e.g., 1%). This is mainly due to the slow reopening of cwnd, which is caused by a very long RTT, after the loss recovery phases. Although the utilization of A-LIAD is also affected by the wireless packet loss, resulting in around a 60% bandwidth utilization, A-LIAD maintains the highest throughput in both PER = 0.1% and 1%.

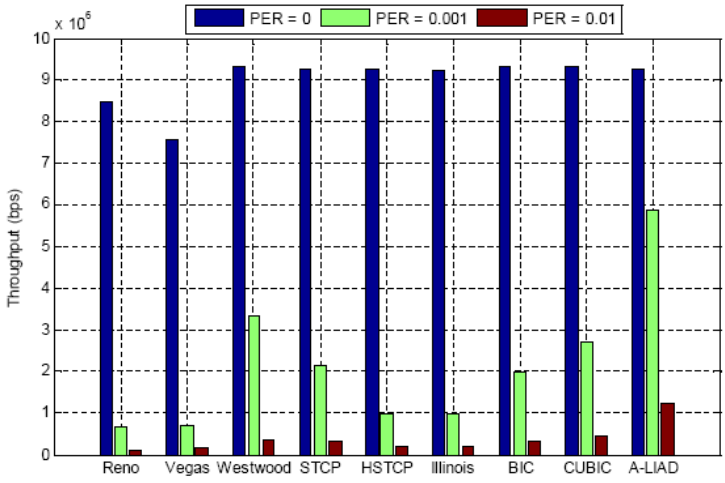


Fig. 10. Wireless packet loss impact on throughput

To better highlight this point, we investigated the window size dynamics of the variants in **Fig. 11**. While the other variants operate at much below BDP (in this case, 700 packets),

A-LIAD works at around a comparatively higher window leading to less sensitivity to a wireless packet loss. This property is mainly due to the logarithmic increase during a lossless period resulting in fast recovery of losses.

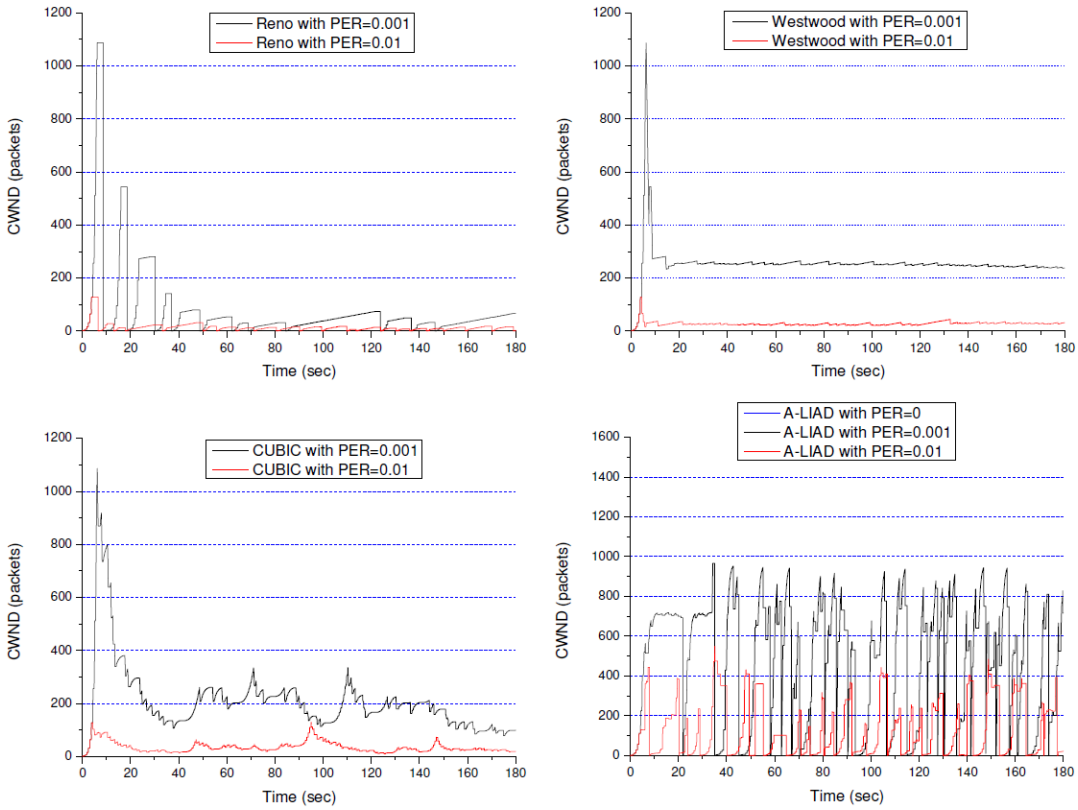


Fig. 11. Window dynamics of variants in terms of packet loss

4.5 Friendliness

To evaluate the friendliness (sometimes called inter-protocol fairness in other papers), we configured five Group B flows to always run the NewReno protocol, while one flow of Group A runs different TCP variants. As stated in section 2, the main reason behind a friendliness evaluation is to ensure the possibility for incremental deployment of the proposed protocol. This means that the proposed protocol should not necessarily be perfectly fair with the most widely implemented TCP, New-Reno. However, its deployment should not degrade the NewReno performance greatly.

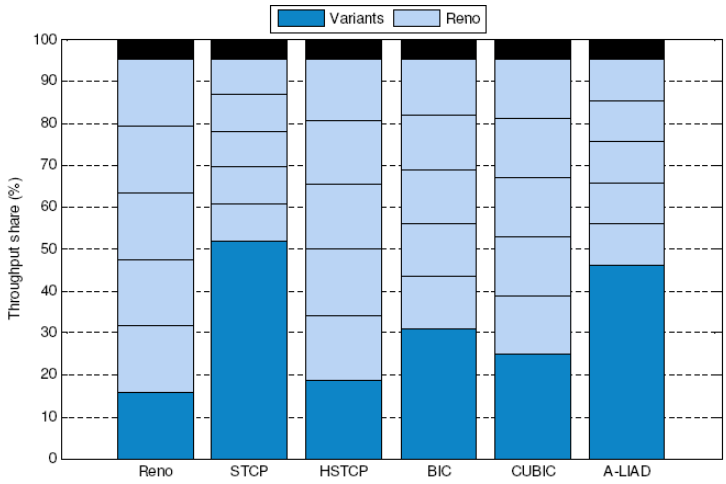


Fig. 12. Friendliness with Reno: RTT = 25 ms, PER = 0

Following this observation, we measured the throughput share of Group B flows running the NewReno protocol, and Group A running different protocols, including NewReno and other variants. According to the results presented in **Fig. 12**, which are obtained from a simulation with C = 10 Mbps and RTT = 25 msec for 300 sec, HSTCP and CUBIC demonstrate the best friendliness characteristic owing to their TCP mode. On the other hand, STCP and A-LIAD show the worst friendliness property for a short RTT, and they reduce the throughput of the NewReno flows in Group B by 45% and 38%, respectively, while CUBIC reduces the share of NewReno flows by 11%. However, as the delay increases, CUBIC also steals significant bandwidth from the NewReno flows since its aggressiveness, shown in **Fig. 13**, has one of the highest values, along with STCP and A-LIAD, in the case of RTT = 600 msec. This is not in the favor of incremental deployment for the current Internet, where most flows are based on the NewReno protocol. However, taking into account that CUBIC is already a part of the OS Linux kernel, and its friendliness is the lowest among all the evaluated protocols, A-LIAD will remain an attractive alternative for high-speed and long-distance networks.

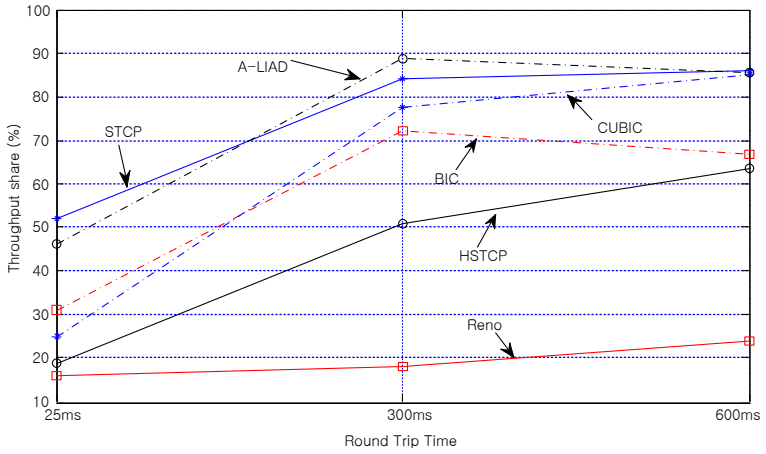


Fig. 13. Aggressiveness of A-LIAD against Reno in terms of RTT

5. Conclusion

In this paper, we presented the adaptive logarithmic increase and adaptive decrease algorithm, or A-LIAD, as an alternative to the current TCP congestion control algorithm. We described that the convex curve is not appropriate for the increasing function of the congestion window, and thus proposed a logarithmic increasing function that adaptively adjusts its increasing rate in the RTT function. We defined a new increasing function in the fashion of a logarithm depending on the increasing factor α, which is different from other logarithmic increase algorithms adopting a fixed value of $\alpha = 2$, leading to a binary increase. A mathematical analysis for the throughput of A-LIAD is represented, and the α value is derived for an RTT function through this analysis. With a modification of the increasing function applied for the CA phase, a hybrid scheme was also presented for a slow-start phase. We used the logarithmic increase function for the second half of a slow-start period, while for the first half period of a slow start, an exponential increase function is used in the same way as standard TCP. From this hybrid scheme, we can avoid an overshooting problem during a slow-start phase even without a SACK option. To verify the feasibility of the algorithm for deployment in a high-speed and long-distance network, several aspects were evaluated through an NS-2 simulation. We performed simulations for the start-up performance, intra-fairness, and inter-fairness as well as the packet loss impacts under different conditions of varying RTT, bandwidth, and PER. From these simulations, we showed that although A-LIAD is not the best option in every aspect, it provides a competitive performance in almost all aspects, especially during a start-up and for a packet loss impact, and thus can be an alternative TCP congestion control algorithm for high BDP networks including satellite networks.

References

[1] J. M. Park, D. S. Ahn, H. J. Lee, *et al.*, "Feasibility of Coexistence of Mobile-Satellite Service and Mobile Service in Cofrequency Bands," *ETRI Journal*, vol. 32, no. 2, pp. 255-264, Apr. 2010. Article(CrossRef Link)

[2] B. Y. Kim, M. S. Bang, S. H. Kim, *et al.*, "A Study on Feasibility of Dual-Channel 3DTV Service via ATSC-M/H," *ETRI Journal*, vol. 34, no. 1, pp. 17-23, Feb. 2012. Article(CrossRef Link)

[3] Y. Hu and V. O. H. Li, "Satellite-based internet: a tutorial," *IEEE Communications Magazine*, vol. 39, no. 3, pp. 164–171, Mar. 2001. Article(CrossRef Link)

[4] V. Jacobson, "Congestion avoidance and control," *ACM Computer Communication Review*, vol. 18, pp. 314–329, Aug. 1988. Article(CrossRef Link)

[5] Internet RFC 6582, *The NewReno Modification to TCP's Fast Recovery algorithm*, IETF, Apr. 2012.

[6] Internet RFC 2018, *TCP Selective Acknowledgment Options*, IETF, Oct. 1996

[7] J. Padhye, V. Firoiu, D. Towsley, *et al.*, "Modeling TCP throughput: A simple model and its empirical validation," in *Proc. of ACM SIGCOMM*, 1998. Article(CrossRef Link)

[8] B. J. Chang, S. Y. Lin, and J. Y. Jin, "LIAD: Adaptive bandwidth prediction based Logarithmic Increase Adaptive Decrease for TCP congestioncontrol in heterogeneous wireless networks," *Computer Networks*, vol. 53, issue 14, pp. 2566–2585, Sep. 2009. Article(CrossRef Link)

[9] D. Kliazovich, F. Granelli, and D. Miorandi, "Logarithmic window increase for TCP Westwood+ for improvement in high speed, longdistance networks," *Computer Networks*, vol. 52, no. 12, pp. 2395–2410, Aug. 2008. Article(CrossRef Link)

[10] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control for fast long-distance networks," in *Proc. of IEEE INFOCOM*, 2004. Article(CrossRef Link)

[11] S. Ha, I. Rhee and L. Xu, "CUBIC : A new TCP-friendly high-speed TCPvariant," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, pp 64-74. Jul. 2008. Article(CrossRef Link)

[12] C. Casetti, M. Gerla, S. Mascolo, *et al.*, "TCP Westwood : end-to-end congestion control for wired/wireless networks," *Wireless Networks*, vol. 8, pp. 467-479, 2002. Article(CrossRef Link)

[13] M. Aoki, E. Oki, and R. R. Cessa, "Measurement Scheme for One-Way Delay Variation with Detection and Removal of Clock Skew," *ETRI Journal*, vol. 32, no. 6, pp. 854-862, Dec. 2010. Article(CrossRef Link)

[14] C. P. Fu and S. C. Liew, "TCP Veno : TCP enhancement for transmission over wireless access network," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 2, pp. 216-228, Feb. 2003. Article(CrossRef Link)

[15] K. Xu, Y. Tian, and N. Ansari,"TCP-Jersey for wireless IP communications," *IEEE J. Sel. Areas Commun.*, vol. 22, pp. 747-756, 2004. Article(CrossRef Link)

[16] K. Xu, Y. Tian, and N. Ansari, "Improving TCP performance in integrated wireless communications networks," *Computer Networks*, vol. 47, pp. 219-237, 2005. Article(CrossRef Link)

[17] A. Afanasyev, N. Tilley, P. Reiher, *et al.*, "Host-to-Host Control for TCP," *IEEE Communications Survey & Tutorials*, vol. 12, no. 3, pp. 547-566, Third Quarter, 2010. Article(CrossRef Link)

[18] I. F. Akyildiz, G. Morabito, and S. Palazzo, "TCP peach : A new congestion control scheme for satellite IP networks," *IEEE/ACM Trans. Networking*, vol. 9, no. 3, pp. 307-321, Jun. 2001. Article(CrossRef Link)

[19] I. F. Akuildiz, X. Zhang, and J. Fang, "TCP Peach+ : enhancement of TCP Peach for satellite IP networks," *IEEE Communications Letters*, vol. 6, no. 7, pp. 303-305, Jul. 2002. Article(CrossRef Link)

[20] C. Caini and R. Firrincieli, "TCP Hybla: a TCP Enhancement for Heterogeneous Networks," *International Journal of Satellite Communications and Networking*, vol. 22, no. 5, pp. 547–566, Sep. 2004. Article(CrossRef Link)

[21] S. Utsumi, S. M. S. Zabir, and N. Shiratori, "TCP-Cherry : A new approach for TCP congestion control over satellite IP networks," *Computer Communications*, vol. 31, issue 10, pp. 2541-2561, Jun. 2008. Article(CrossRef Link)

[22] D. J. Leith, R. N. Shorten, and G. McCullagh, "Experimental Evaluation of Cubic-TCP," in *Proc. of Protocols for Fast Long Distance Networks (PFLDnet)*, Los Angeles, 2007.

[23] S. B. Moon, *Measurement and Analysis of End-to-End Delay and Loss in the Internet*, doctorial dissertation, University of Massachusetts Amherst, Massachusetts, Jan. 2000.

[24] Z. Shi, C. Beard and K. Mitchell, "Analytical Models for Understanding Space, Backoff and Flow Correlation in CSMA Wireless Networks," *Wireless Networks*, vol. 19, issue 3, pp. 393-409, Apr. 2013. Article(CrossRef Link)

Minsu Shin received his BS and MS degrees in electrical engineering from Korea Aerospace University, Seoul, Korea in 1998 and 2000, respectively, and Ph.D degrees in Computer Networks from Chungnam National University in 2011. Since he joined the Electronics and Telecommunications Research Institute (ETRI) in 2000, he has worked for satellite communication systems and broadcasting systems until now. Now his research interests are satellite communication network design, wireless resource management, wireless TCP and cross-layer enhancement.

Mankyu Park received the B.S. and M.S. degrees, from Kongju National University in 1999 and 2001, respectively. He received the Ph.D. degree in Computer Networks from Chungnam National University in 2011. In 2009, he joined the Electronics and Telecommunications Research Institute (ETRI), where he is a Senior Member of Engineering Staff in Satellite Broadcasting and Telecommunications Convergence Research Team. His research interests include Internet protocols, traffic control, TCP congestion control, performance analysis and mobile communication.

Deockgil Oh received his B.S. degree in Electronics Engineering Department from Seoul National University (SNU) in 1980. He received the M.S. and Ph.D degrees in Electronics Engineering from SNU, Korea, in 1984 and 1996, respectively. He joined ETRI in 1982, where he is currently working as a team leader at Team of Satellite Broadcasting and Communication Convergence. His research interests include wireless access technology, mobile communication and broadcasting system and future generation satellite broadcasting architectures.

Byungchul Kim received the B.S. degree in electronics engineering from Seoul National University in 1988, and M.S. and Ph.D degrees in electronic engineering from Korea Advanced Institute of Science and Technology (KAIST), Korea, in 1990 and 1996, respectively. He is currently a professor at the Department of Information and Communication Engineering of Chungnam National University, Korea since 1999. Also, from 1993 to 1999, he worked as a Research Engineer at the Samsung Electronics. His research interests include computer networks, wireless internet, sensor networks and mobile communications.

Jaeyong Lee received the B.S. degree in Electronics engineering from Seoul National University in 1988, and M.S. and Ph.D degrees in electronic engineering from Korea Advanced Institute of Science and Technology (KAIST), Korea, in 1990 and 1995. He is currently a professor at the Department of Information and Communication Engineering of Chungnam National University, Korea since 1995. Also, from 1990 to 1995, he worked as a research engineer at the Digicom Institute of Information and Communications. His research interests include Internet protocols, traffic control, performance analysis and mobile internet.