UTILIZATION-BASED CONGESTION CONTROL

Satoshi Utsumi[1] and Salahuddin Muhammad Salim Zabir[2]

[1] Department of Control and Information System Engineering, Tsuruoka National College of Technology, Yamagata, Japan

u-satoshi@tsuruoka-nct.ac.jp

[2] Orange Labs, France Telecom Japan, Tokyo, Japan

szabir@ieee.org

ABSTRACT

Traditional connection oriented protocols like TCP NewReno perform poorly over wireless links. The problem lies in their design assumptions based on loss based congestion control. Various modifications to loss based congestion control schemes have so far been proposed to overcome the issue. In addition, the comparatively newer family of delay based congestion control mechanisms like Caia-Hamilton Delay (CHD), offer effective solutions for wireless link loss. All these approaches aim at improving the performance of the target connection without considering the overall network condition. Therefore, when co-existing with the conventional loss-based congestion control, the performances suffer severely. Furthermore, for delay based schemes, it is not possible to determine the optimum value of the queuing threshold intuitively. In this paper, we propose a new end-to-end congestion control mechanism that we name as Utilization-based Congestion Control (UCC) in order to overcome the above problems. UCC provides a solution based on the utilization at the bottleneck link. It is free from critical parameters like queuing threshold and is friendly to other flows over wireless links when deployed together. Simulation results show that over wireless links, (i) UCC can yield a performance improvement of 150% or more compared to conventional loss-based schemes and (ii) it is friendly to conventional loss-based congestion controls.

KEYWORDS

Congestion Control, Queuing Theory, Link Utilization, Bottleneck Link, Queuing Delay

1. INTRODUCTION

Incorporation of mobile and wireless devices to the Internet has led to the need for optimizing major applications to perform satisfactorily over wireless link. Typically, wireless links are prone to a higher link error rate than their wired counterparts. Since conventional TCP protocols as TCP NewReno are designed to be used in wired networks with low link error rates, their designs do not take link errors into account. That is, they assume that segment losses occur solely due to congestion in networks [1]. Whenever a segment loss occurs, conventional TCP infers congestion and the sender handles the situation by reducing its sending rate as a remedial measure. Therefore, when conventional TCPs are used over wireless networks, they interpret even the losses occurring due to link error as originating from congestion and thus frequently lower their transmission rates unnecessarily. Therefore, deployment of conventional TCP over wireless links results in a decreased throughput [2].

Various end-to-end congestion control mechanisms have so far been proposed. In addition, the comparatively newer family of delay based congestion control mechanisms like Caia-Hamilton Delay (CHD) [3] offer effective solutions for wireless link loss. However, all these mechanisms aim at improving solely the performance of the connection under concern. Therefore, while aggressively increasing the performance of the desired connection, they affect the performance of other types of connections severely.

DOI : 10.5121/ijcnc.2012.4507

In this paper, we propose a new end-to-end congestion control mechanism that we name as Utilization-based Congestion Control (UCC), focusing on the techniques to detect the network congestion in order to overcome the above issues. UCC provides a solution based on the utilization at the bottleneck link. We estimate the utilization on the bottleneck link using queuing theory and apply this for end-to-end congestion control.

From functional point of view, UCC is free from critical parameters like queuing threshold. Moreover, it is friendly to other flows over wireless links when deployed together. Simulation results show that (i) UCC can yield a performance improvement of 150% or more compared to conventional loss-based schemes over wireless links and (ii) it is friendly to conventional loss-based congestion controls over wireless links.

The rest of this paper is organized as follows. Section 2 summarizes the most recent relevant research works. In section 3, we describe a method that can be used for practically calculating link utilization. We present our proposed scheme in section 4. In section 5, we discuss in detail how we have evaluated UCC. Finally we conclude in section 6.

2. RELATED WORK

A number of congestion control mechanisms using only lost packets to detect congestion exits to date [4, 5, 6, 7]. Since these are already quite well established mechanisms, we omit mentioning them in this section. To the contrary, congestion control mechanisms using delay parameters are getting popular recently. They are relatively new and therefore, we discuss them here.

2.1. Hamilton Delay-based Congestion Control (HD)

Leith et al. [8] describe the case for delay-based Additive Increase Multiplicative Decrease (AIMD) congestion control, which provides end-to-end control with high utilization, low delay and zero congestion related packet loss. This idea was improved by Budzisz et al. [9] for fair coexistence with loss-based TCP algorithms.

On receipt of every ack, cwnd (w) is evaluated as follows:

$$w_{i+1} = \begin{cases} \frac{w_i}{2} & X < g(q_i) \\ w_i + \frac{1}{w_i} & otherwise \end{cases} \quad (1)$$

where $g(q_i)$ is the backoff probability function shown in Figure 1 (as reported in [9]), $X \in [0,1]$ is random number, p_{max} is the maximum probability of backoff, $q_{max}=RTT_{max}-RTT_{min}$ is an estimate of the maximum observed queuing delay, q_{min} is a target minimum queuing delay, and q_{th} is a threshold that divides regions A and B.

When loss-based flows are on the link, the queue is pushed into region B. The delay-based flows have a lower probability of backoff in this region, enabling them to receive a fairer share of the available bandwidth. When loss-based flows are no longer on the bottleneck link region B is unstable, ensuring delay-based flows converge to a low delay state in region A.

2.2. Caia-Hamilton Delay-based Congestion Control (CHD)

Caia-Hamilton Delay-based congestion control (CHD) uses the same probability function shown in Figure 1, but with three key modifications.

- Delay-based cwnd operations are performed only once per RTT
- CHD infers (and tolerates) packet losses that are likely to be unrelated to congestion
- CHD improves coexistence with loss-based TCP algorithms

As with HD, CHD only modifies the TCP sender's behavior. No change is required at the existing TCP receivers.

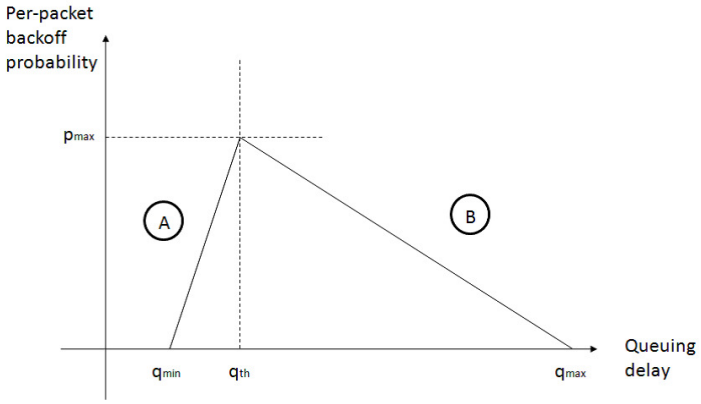

Figure 1. Per-packet backoff probability function [9]

2.2.1. Delay-based window updates

CHD operates on cwnd in a similar way to equation (2), except that CHD updates once every RTT and uses the maximum queuing delay experienced during the last RTT. In other words, if $hr=max_r(q_i)$ is the maximum queuing delay observed in RTT r, CHD updates w per RTT as follows:

$$w_{i+1} = \begin{cases} \frac{w_i}{2} & X < g(h_r) \\ w_i + 1 & otherwise \end{cases} \qquad (2)$$

The case $X<g(h_r)$ represents a delay triggered window reduction.

2.2.2. Loss-based window updates

CHD's ability to compete fairly with loss-based flows involves the use of a *shadow window*.

$$s_{i+1} = \begin{cases} \max(w_i, s_i) & X < g(h_r) \wedge A \\ \max(w_i, s_i) & X < g(h_r) \wedge h_r > q_{th} \\ 0 & otherwise \end{cases} \qquad (3)$$

where $A=h_r<q_{th} \wedge h_r>h_b$ and h_b is the value of hr when the last delay triggered w reduction occurred.

When $s\neq 0$, if the last recorded h is in region B, a packet loss will trigger a NewReno like behavior. To the contrary, if a packet loss occurs in region A, w remains unchanged. The later behavior corresponds to the assumption that a loss in region A is non-congestion loss. CHD's rule for updating w on packet loss is:

$$w_{i+1} = \begin{cases} \frac{\max(w_i, s_i)}{2} & packetloss \wedge h_r > q_{th} \\ w_i & otherwise \end{cases} \qquad (4)$$

2.3. Caia Delay-Gradient Congestion Control (CDG)

RTT is a noisy signal - a cleaner signal is required for inferring congestion from the gradient of RTT over time. CDG uses the maximum RTT (τ_{max}) seen in a measured RTT interval, along with the minimum RTT (τ_{min}) seen within a measured RTT interval. Based on these, two measures of gradient (change in RTT measurement per RTT interval) are kept, where *n* is the n-th RTT interval:

$$g_{\min,n} = \tau_{\min,n} - \tau_{\min,n-1} \qquad (5)$$

$$g_{\max,n} = \tau_{\max,n} - \tau_{\max,n-1} \qquad (6)$$

The maximum and minimum measurements are less noisy than per packet RTT measurements. Nevertheless they apply the moving average smoothing of Equation (7), which may be calculated iteratively using Equation (8)(where *a* is the number of samples in the moving average window).

$$\overline{g}_n = \sum_{i=n-a}^{n} \frac{g_i}{a} \qquad (7)$$

$$\overline{g}_n = \overline{g}_{n-1} + \frac{g_n - g_{n-a}}{a} \qquad (8)$$

where $g_i = g_{min,i}$ for calculating $\bar{g}_{\min,n}$ or $g_i = g_{max,i}$ when caluclating $\bar{g}_{max,n}$.

An alternative to the moving average would be exponential smoothing. However, if the measured $\tau_{max,n}$ ceased to grow because a queue along the path was full, an exponential average would only approach $\bar{g}_{max,n}=0$ in the limit. A moving average would achieve $\bar{g}_{max,n}=0$ in *a* samples.

In order to tolerate the kinds of packet loss common in, say, wireless environments, they must infer whether or not packet loss is related to congestion. For simple drop tail queues, congestion related loss is due to overflow of a queue along the packet's path. To infer such events CDG uses both $\bar{g}_{min}$ and $\bar{g}_{max}$.

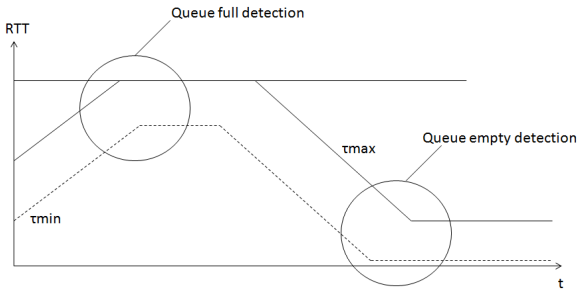

Figure 2. Idealized RTT dynamics for queue full and empty events [10]

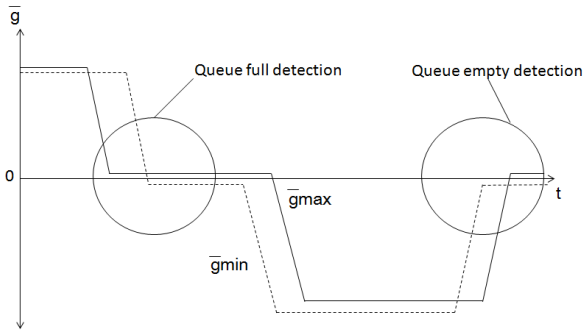

Figure 3. Idealised gradient dynamics for queue full and empty events [10]

Figure 2 illustrates their assumption that when a queue fills to capacity, τmax stops increasing before τmin stops increasing, and that the reverse is true for a queue moving from full to empty. Figure 3 shows the idealised gradients for these two conditions (with the lines for $\bar{g}_{min}$ and $\bar{g}_{max}$ offset slightly for clarity). Based on this CDG estimates the state of the path queue to be $Q \in \{full, empty, rising, falling, unknown\}$. Only when $Q=full$ are packet losses treated as congestion signals.

2.3.1. RTT independent backoff and Shadow window

CDG introduces the RTT independent backoff algorithm to achieve fairness between flows having different base RTT [10]. Also, CDG recovers some of its lost sending capability by utilising the shadow window idea from [3] to mimic the loss based backoffs of TCP NewReno [10].

2.3.2. Loss-based window updates

If a packet loss occurs, the congestion window (w) is updated as follows:

$$w_{i+1} = \begin{cases} \dfrac{\max(w_i, s_i)}{2} & packetloss \wedge Q = full \\ w_i & otherwise \end{cases} \qquad (9)$$

In the case of packet losses, the multiplicative decrease factor is 0.5 (as in NewReno), and w is set to half the bigger of s (the shadow window) and w. Using the shadow window concept from [3] improves CDG's coexistence with loss based flows. They do not reclaim the lost transmission opportunities, but this approach does lessen the impact of the extra delay-gradient based backoffs.

3. ESTIMATION OF UTILIZATION ON BOTTLENECK LINK

The following equation is true in each queuing model M/M/1, M/D/1, M/G/1, G/M/1, G/G/1, where ρ is the link utilization, p_0 is the probability when the queuing delay is 0 at the link [11].

$$\rho = 1 - p_0 \qquad (10)$$

Here, $1\text{-}p_0$ is equal to the probability when the queuing delay is not 0. Suppose out of N packets that were transmitted, N_{min} experienced a round trip time, $RTT=RTT_{min}$ (the minimum RTT). The utilization ρ on the bottleneck link can then be expressed as follows [12].

$$\rho = \frac{N - N_{\min}}{N} \qquad (11)$$

However, in the presence of TCP SACK option, the sizes of the ACKs are not constant, that is, the processing time of the ACKs at the intermediate nodes will be different. This will make equation (11) invalid under the above assumptions.

Therefore, a more general way of calculating ρ is required. Rather than using RTT, we base our proposed mechanism on one way delay while measuring N_{min}. This is achieved through the use of TCP timestamp option. As in [13], the UCC sender puts its current clock value in the *TSval* field of the timestamp option. The receiver, while returning the ACK, follows RFC 1323, and echo replies it in the *TSecr* field along with attaching its own clock value. This is sufficient to yield one way delay. It should be noted that there is no need for synchronizing the sender and receiver clocks because the relative time difference is enough for our calculations. Now, in our scheme, N_{min} is measured as the number of times that the value of the one way delay equals to the minimum one way delay among the N transmitted packets.

4. OUR PROPOSAL: UTILIZATION-BASED CONGESTION CONTROL

In this section we propose a new end-to-end congestion control based on link utilization for wireless networks. We name this new congestion control mechanism as utilization-based congestion control or UCC.

Conventional and many newer congestion control schemes work on a reactive approach triggered by the loss of transmitted data and/or measurement of queuing delay. Many of them have been quite effective in handling instantaneous congestion scenarios in the network. However, these schemes basically aim solely at keeping the performance of the connections under consideration high while ignoring the overall condition of the network. To the contrary, UCC considers congestion from a global perspective. In UCC, congestion control is performed by taking into account (i) the loss of a data segment and (ii) the link utilization as a long term condition of the bottleneck link. As a result, despite keeping own performance high enough, UCC can avoid undesirable performance degradation of co-existing conventional congestion controls mechanisms through sensing an incipient congestion and shrinking its congestion window early enough.

4.1 Loss-based Window Update

We assume that $n\times RTT$ is the time after detecting a packet loss until detecting the next packet loss. UCC estimates the utilization ρ using equation (11), measuring one way delays by the transmitted data packet and the ACK during the $n\times RTT$.

CHD shrinks the congestion window (or, shadow window) to half when it decides that the packet loss is by congestion using queuing delay. On the other hand, UCC controls the congestion window based on the utilization on the bottleneck link of equation (11). UCC shrinks the congestion window to half, deciding that the potential congestion occurs if the utilization ρ on the bottleneck link is more than a tuning parameter α (for example, $\alpha=0.99$) when detecting a packet loss by duplicate ACKs.

That is, when detecting a packet loss by duplicate ACKs, UCC updates the congestion window as follows.

$$w_{i+1} = \begin{cases} \frac{w_i}{2} & packetloss \wedge \rho > \alpha \\ w_i & otherwise \end{cases} \qquad (12)$$

The following conditions are necessary for equation (11) to hold.

1) UCC should know the accurate minimum value of RTT (or, the one way delay).
2) The minimum value of RTT (or, the one way delay) should not change.
3) When the queuing delay is 0 on the bottleneck link, the queuing delays on other links in the same path should be 0.

Regarding the condition 1) above, from queuing theory, the probability that queuing delay is 0 is $p0=1-\rho$. Then, even when $\rho=0.95$, if the sender transmits 100 data packets, the minimum RTT (or, the minimum one way delay) is measured about 5 times. (Here we do not take account of the queuing delay under non-bottleneck link.) That is, if many data packets are transmitted, the minimum RTT (or, the minimum one way delay) is measured. Then, the assumption that the minimum RTT (or, the minimum one way delay) is measured in long-range holds.

As per condition 2), the minimum RTT (or, the minimum one way delay) should not change in long-range. If the minimum RTT (or, the minimum one way delay) changes, UCC should update the value. For example, UCC updates the minimum RTT (or, the minimum one way delay) periodically.

As regards to condition 3), it is possible that the queuing delay occurs in non-bottleneck links. When the queuing delay occurs in non-bottleneck links, the utilization ρ has upper errors from equation (11). When ρ is measured more than the actual value, the congestion window updated by equation (12) is smaller than the ideal value. It is not optimal in terms that the throughput of the congestion control decreases, but secure in terms that congestion does not occur.

The other algorithms deployed in UCC are as follows.

- When the packet loss is detected by timeout, the sender retransmits the data packet, updates the congestion window to 1 packet, and moves to the Slow Start phase. (To deal with serious congestion.)
- At Slow Start phase, the sender adds 1 packet to the congestion window upon receiving an ACK.
- At Congestion Avoidance phase, the sender adds 1 packet to the congestion window by one RTT.

Our proposed scheme, UCC, has one major functional advantage over its counterparts. That is, it does not have the requirement of any critical parameter. The only parameter that needs to be decided is the utilization threshold, α, which can be decided intuitively. In other cases like CHD, the need for deciding the queuing threshold q_{th} that depends on the buffer size of the bottleneck link poses a very difficult deployment problem.

Congestion window dynamics of UCC is shown in Figure 4.

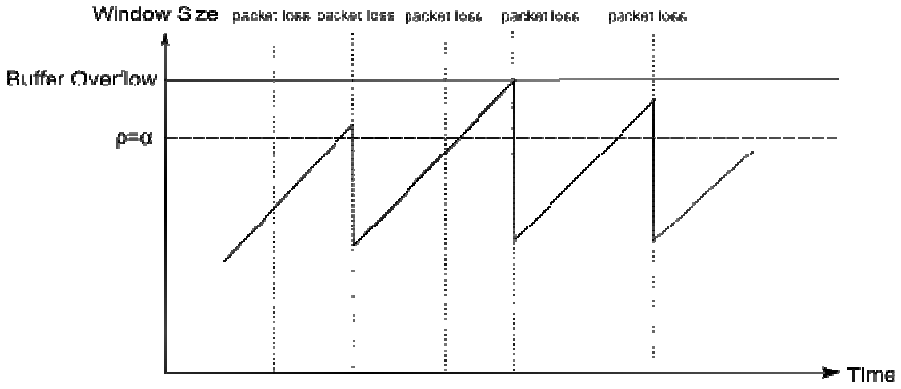

Figure 4. Congestion window dynamics of UCC

5. EVALUATION

5.1. Simulation Configuration

We evaluate UCC and three other existing congestion control mechanisms through simulation. Besides UCC, the congestion control mechanisms under concern are as follows:

- TCP NewReno
- CHD
- CDG

TCP New Reno serves as the baseline congestion control mechanism as it has been widely used and also referenced as the baseline protocol by many TCP studies. The objective is to estimate the following:

1) the performance of each congestion control scheme
2) how they affect TCP NewReno, and
3) how well the overall network behaves as a whole because of their usages.

We use Scenargie [14], a high fidelity commercial simulator for our evaluation. The latest version of Scenargie can execute various TCP codes implemetend in FreeBSD 9.0 directly within its simulation framework. As operational codes for CHD and CDG are currently implemented only in Free BSD 9.0, the use of Scenargie can eliminates our effort to build multiple alternative congestion control mechanisms for this comparative study. Further, it can run various protocol models under realistic simulation scenarios, as it provides a rich set of peripheral model library including a variety of wireless propagation, mobility and user behaviour models. We use a very simple evaluation scenario in this simulation study such that the causes of performance differences among these congestion control mechanisms can be easily inferred, but such a feature leaves us open for more realistic and comprehensive evaluation as our future work without much modelling effort at our end.

We create a simulation scenario whose network topology is depicted in Figure 4. The configuration of network parameters is summarized in Table 1.

Table1. Simulation configuration

Capacity for Links N1 – R and N2 – R	100 Mbps
Capacity for Link R – N3	10 Mbps
Delay for Links N1 – R and N2 – R	0 ms
Delay for Link R –N3	20 ms
Drop rate (random) for Link R –N3	0.00 – 0.05 /packets (varied)
Buffer size for Link R –N3	50 packets
Segment size	1000 bytes
IP Version	4
Simulation duration	100 s

5.2. Performance Metrics

We evaluate UCC and other mechanisms in terms of throughput. Throughput is a measure of the amount of data transferred. We choose a conservative definition of throughput that is equivalent to the definition of effective throughput in [1]. That is, Throughput is defined as:

$$Throughput = \frac{AmountOfCumulativelyAcknowledgedData}{ConnectionTime}$$

, where *AmountOfCumulativelyAcknowledgedData* is the number of bytes acknowledged at senders (i.e., excluding retransmitted data and SACKed data).

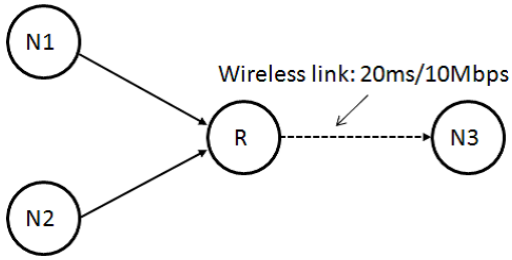

Figure 5. Simulation scenario

5.3. Simulation Results

At first, we evaluate the performance of each congestion control scheme while competing with the same type of flow for the channel. In the simulation scenario of Figure 5, we achieve this by setting the senders of the same congestion control on both nodes *N1* and *N2*. The receiver that supports SACK is set on node *N3*. We evaluate the total throughput of 2 flows on *N1* and *N2*. We assume that the link capacity from router *R* to *N3* is 10Mbps (both up and down), the propagation delay of the link to 20msec (both up and down; i.e., round trip 40msec), the buffer size of the router *R* is 50 segments, and the size of each data segment is 1,000 bytes. The value of *qth* of CHD has been set to 20msec, and α of UCC has been set to 0.99. Every congestion control supports SACK option. The simulation time is 100sec.

Figure 6 shows the results of total throughput of *N1* and *N2*. The curves confirm that UCC can achieve very high performance over wireless link which can be 150% or higher than TCP NewReno.

In next experiment, we set the sender of a target congestion control on *N1*, the sender of TCP NewReno on *N2*, and the receiver of TCP on N3 in the simulation scenario of Figure 5. We evaluate the effects of the target congestion control at *N1* and TCP NewReno at *N2* each other.

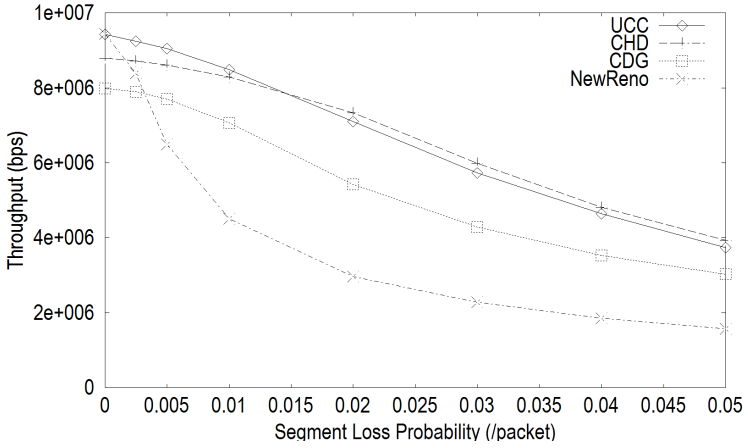

Figure 6. Throughput Comparison

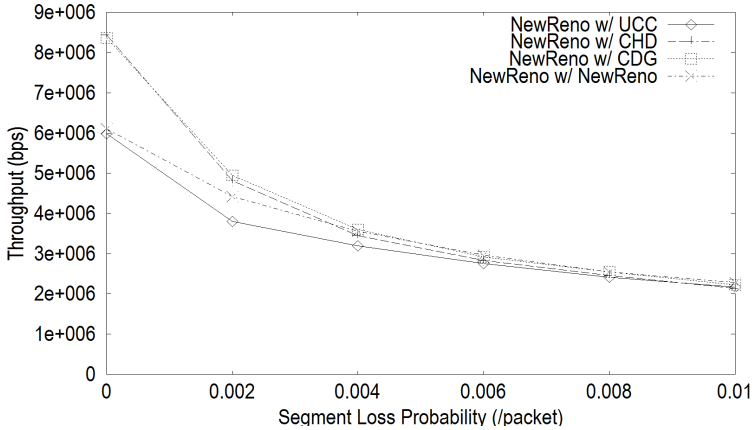

Figure 7. Variation of TCP NewReno Throughput with different flows

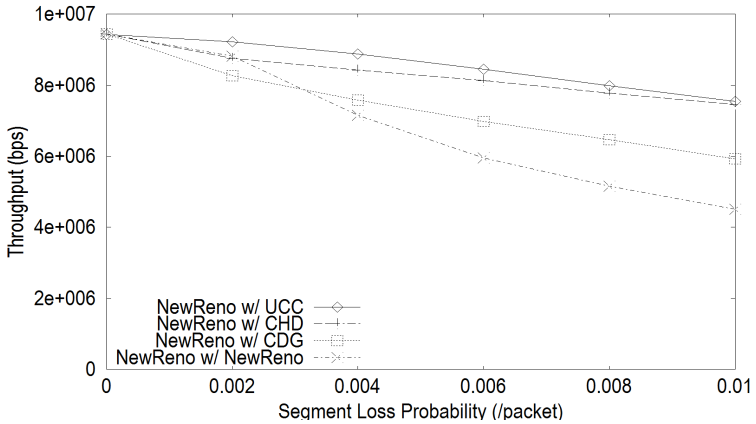

Figure 8. Throughput while deployed together with TCP NewReno

Figure 7 shows corresponding throughput yield for TCP NewReno at *N2*. It should be noted that the curve that corresponds to connecting a UCC flow at *N1* is quite comparable with the curve that corresponds to connecting a TCP NewReno at *N1* for practical operational value of link error. To the contrary, the curves corresponding to connecting a CHD flow at *N1* and a CDG flow at *N1* show much higher TCP NewReno throughput when link error is 0, that is, in wired networks. This implies that CHD flow and CDG flow are pressed by TCP NewReno flow at *N2*. UCC is friendly to TCP NewReno while CHD and CDG are not.

In Figure 8, we observe that the overall throughput is reasonably high while UCC is deployed at *N1* together with New Reno at *N2*. Hence it can be inferred that UCC yields high throughput without affecting the co-existing TCP NewReno connection.

6. CONCLUSIONS

Performance of conventional congestion control mechanisms like TCP NewReno suffers due to incorrect interpretation of losses resulting from link errors. Various modifications, both loss based and delay based approaches, have been proposed to overcome the corresponding performance degradation. Since all these mechanisms aim at improving solely the performance of connections under consideration, they do not take the performance of other connections into account. Therefore, when co-existing with the conventional loss-based congestion control, the performance suffers drastically. In this paper, we proposed a new end-to-end congestion control mechanism that we named as Utilization-based Congestion Control (UCC) in order to overcome the above issues. UCC provides a solution based on the utilization at the bottleneck link. It is free from critical parameters like queuing threshold and is friendly to other flows over wireless links when deployed together. Simulation results show that (i) UCC can yield a performance improvement of 150% or more compared to conventional loss-based schemes over wireless links and (ii) it is friendly to conventional loss-based congestion controls over wireless links.

ACKNOWLEDGEMENTS

We evaluated our scheme using network simulator Scenargie presented by Space-Time Engineering. We thank Space-Time Engineering for their kind support.

REFERENCES

[1] V. Jacobson, (1988) “Congestion avoidance and control”, Proc.ACM Special Interest Group on Data Communications (SIGCOMM).

[2] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz (1997) “A comparison of mechanisms for improving TCP performance over wireless links”, IEEE/ACM Trans. Networking, Vol.5, No.6,pp.756-769.

[3] D. A. Hayes, and G. Armitage, (2010) “Improved co-existence and loss tolerance for delay based TCP congestion control”, 35th Annual IEEE Conference on Local Computer Networks (LCN2010), Denver, Colorado.

[4] R. Ludwig, and R. H. Katz, (2000) “The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions”, ACM Computer Communication Review, Vol. 30, No.1.

[5] S. Utsumi, S. M. S. Zabir, and N. Shiratori, (2008) “TCP-Cherry: A new approach for TCP congestion control over satellite IP networks”, Computer Communications, Elsevier, vol.31, pp. 2541-2561.

[6] S. Utsumi, S. M. S. Zabir, and N. Shiratori, (2009) "TCP-Cherry for satellite IP networks: Analytical model and performance evaluation", Computer Communications, Elsevier, vol. 32, issue 12, pp. 1377-1383.

[7] M. Sakar, K. K. Shukla, and K. S. Dasgupta, (2010) "Network State Classification based on the Statistical Properties of RTT for an Adaptive Multi-State Proactive Transport Protocol for Satellite based Networks", International Journal of Computer Networks & Communications (IJCNC), vol. 2, No. 6, Nov. 2010.

[8] D. Leith, R. Shorten, G. McCullagh, J. Heffner, L. Dunn, and F. Baker, (2007) "Delay-based AIMD congestion control", in Proc. Protocols for Fast Long Distance Networks, California.

[9] L. Budzisz, R. Stanojevic, R. Shorten, and F. Baker, (2009) "A Strategy for Fair Coexistence of Loss and Delay-Based Congestion Control Algorithms", IEEE Communications Letters, vol. 13, no. 7, pp.555-557.

[10] D. A. Hayes, and G. Armitage, (2011) "Revisiting TCP Congestion Control using Delay Gradients", IFIP/TC6 Networking, Valencia, Spain.

[11] Jain, R. (1991) "The Art of Computer Systems Performance Analysis", John Wiley and Sons Inc., USA.

[12] K. Igai, and E. Oki, (2011) "A Simple Estimation Scheme for Upper Bound of Link Utilization Based on RTT Measurement", Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT), pp. 10-16.

[13] A. Kuzmanovic, and E. W. Knightly, (2006) "TCP-LP: low-priority service via end-point congestion control", IEEE/ACM Transactions on Networking, Volume 14 Issue 4.

[14] M. Takai, Y. Owada, and K. Seki, (2009) "A Comparative Study on Network Simulators for ITS Simulation: IEEE802.11 Media Access Control (MAC) models", 19th ITS World Congress.

Authors

Satoshi Utsumi received his B.E., M.S. and Ph.D. degrees in information science from Tohoku University, Japan, in 2000, 2003 and 2009 respectively. He worked in Advantest Corporation from April 2003 to October 2006. He worked as a postdoctoral fellow of Japan Society for the Promotion of Science from April 2009 to March 2011. He joined Tsuruoka National College of Technology, Japan at 2011. He is researching about Internet congestion controls and so on. He received the JC-SAT Award at 2011.

Salahuddin Muhammad Salim Zabir received his B.E. degree in computer science and engineering from Bangladesh University of Engineering and Technology, Bangladesh, in 1994. He received his M.S. and Ph.D. degree in information science from Tohoku University, Japan, in 2000 and 2004 respectively. He joined France Telecom Japan, Orange Labs Tokyo at 2008. He is researching about Internet congestion controls and so on. He is a member of IEEE.