International Journal of Principles and Applications of Information Science and Technology
February 2010, Vol.3, No.1

Design of Network Dependent Congestion Avoidance TCP (NDCA-TCP) for Performance Improvement in Broadband Networks

ManKyu Park[1], BeomGyu Lee[2], DaeIg Chang[1], DeockGil Oh[1]
JaeYong Lee[3], and ByungChul Kim[3],

[1] Satellite Broadcasting & Telecommuncations Convergence Research Team, ETRI
138 Gajeongno, Yuseong-gu, Daejeon, 305-700, Korea
{neomkpark, dchang, dgoh}@etri.re.kr

[2] Korea Institute of Geoscience & Mineral Resources,
92 Gwahangno, Yuseong-gu, Daejeon, 305-350, Korea
fobetter@kigam.re.kr

[3] Department of Information Communications Engineering, Chungnam National University
79 Daehangno, Yuseong-gu, Daejeon, 305-764, Korea
{jyl, byckim}@cnu.ac.kr

Abstract. The Internet environment is increasingly changing to broadband networks and many end users are utilizing fixed bandwidth networks in broadband access networks such as VDSL or passive optical networks. However, the congestion controls of traditional TCP implementations have the problem of under-utilization in high bandwidth delay product networks. In addition, it is very difficult for the traditional TCP to obtain stable bandwidth in fixed bandwidth networks because of its unstable behavior of congestion control. In this paper, we propose a new advanced congestion control algorithm to solve these problems in broadband networks. Simulation results show that the proposed algorithm improves the throughput, utilization, and RTT fairness for fixed broadband networks.

Keywords: High speed TCP, congestion control, high bandwidth delay product networks, fixed bandwidth networks.

1 Introduction

The legacy TCP protocol is the most commonly used transport protocol in the Internet due to its reliability and stable performance in various network environments. However, the congestion control algorithm of the legacy TCP designed for narrow bandwidth networks is not suitable for high bandwidth delay product (BDP) networks, because it takes a long time to utilize the large available bandwidth and its performance decreases abruptly with a single packet loss.

In this paper, we propose a new TCP adopting an advanced congestion control algorithm called “Network Dependent Congestion Avoidance TCP (NDCA-TCP)” that effectively utilizes the large available bandwidth and provides better RTT fairness. The proposed algorithm also can solve the saw-tooth congestion window behavior of the AIMD (Additive Increase Multiplicative Decrease) algorithm by limiting the congestion window growth to be used in fixed bandwidth networks such as VDSL and PON networks.

In a slow-start period, the proposed algorithm measures the available bandwidth whenever it receives TCP ACK packets, in order to dynamically determine the slow-start threshold. The proposed algorithm changes the congestion avoidance phases from exponential growth to linear growth when the congestion window exceeds the slow-start threshold determined by bandwidth measurement. The algorithm performs an improved slow-start mechanism after half-decrease of the congestion window due to three duplicate ACKs to rapidly get the large available bandwidth, until the congestion window reaches the slow-start threshold or the congestion window limit configured to cope with fixed bandwidth networks. The simulation results using ns-2 [10] show that the NDCA-TCP improves the throughput performance and RTT fairness compared to the representative High Speed TCP [3] standardized by IETF. It also shows good performance characteristics for fixed broadband networks.

The rest of the paper is organized as follows. Section 2 describes the previous related work in this area. Section 3 explains the proposed congestion control algorithm in detail. In Section 4, we show the simulation results and discussion. Finally, we conclude the paper in Section 5.

2 Related Work

Traditional TCP, i.e., TCP-Reno, is compliant with the RFC 793 [1] and RFC 2581 [2] IETF standards. During the initial start-up phase (slow start), the TCP exponentially increases the amount of transferred data until detecting either a packet loss by timeout or triple duplicate ACKs. When a loss is detected, the TCP halves the congestion window (*cwnd*), which constrains the number of packets to be sent without acknowledgement, and goes into the congestion avoidance (CA) phase. During the CA phase, it increases the *cwnd* by one packet per window, and halves the window for a packet loss. Thus, we identify TCP's congestion control algorithms as Additive Increase Multiplicative Decrease (AIMD). However, the AIMD control of the current TCP is not dynamic enough to fill a big pipe for large BDP networks. For example, for a standard TCP connection with 1500-byte packets and a 100 ms round-trip time, achieving a steady-state throughput of 10 Gbps would require an average congestion window of 83,333 segments and a packet drop rate of at most one loss event for every 5,000,000,000 packets [3]. The average bit error rate of at most ($2x10^{-14}$) is necessary for full link utilization in this environment, which is an unrealistic requirement for current network technology.

Over the past few years, researchers have made efforts to solve this under-utilization problem of legacy TCP in high BDP networks [3], [4], [5], [12], which can be classified into two main groups; loss-based and delay-based solutions. Loss-based solutions, such as HighSpeed TCP [3] and Scalable TCP [4], focus on a modification of increase and decrease parameters of the TCP-AIMD algorithm. These loss-based solutions use packet-loss as the only indication of congestion. However, these protocols have some drawbacks for deployment in terms of convergence times, compatibility, fairness and so on [6], [7]. In contrast, delay-based solutions propose the queuing delay as a congestion measure, as in FAST-TCP [5]. The delay-based solutions reduce or increase the transmission rate only based on RTT variations. However, when competing with loss-based solutions, delay-based solutions get penalized due to the aggressive nature of loss-based solutions [8]. The compound TCP in [12] have combined the loss-based congestion control and the delay-based congestion control for better bandwidth scalability and good TCP-fairness.

Although there have been many congestion control algorithms proposed for high BDP networks, there is no specific algorithm that satisfies all aspects of scalability, fairness, stability and TCP-friendliness. In particular, most high speed TCP protocols show good performance only in a low packet loss environment, and reveal some problems in TCP fairness. Therefore, further research is necessary into TCP congestion controls for high BDP networks having various characteristics.

3. Design of the NDCA-TCP algorithm

In this section, we propose a new congestion control algorithm for designing the NDCA-TCP that utilizes the large available bandwidth promptly and operates stably for fixed bandwidth networks.

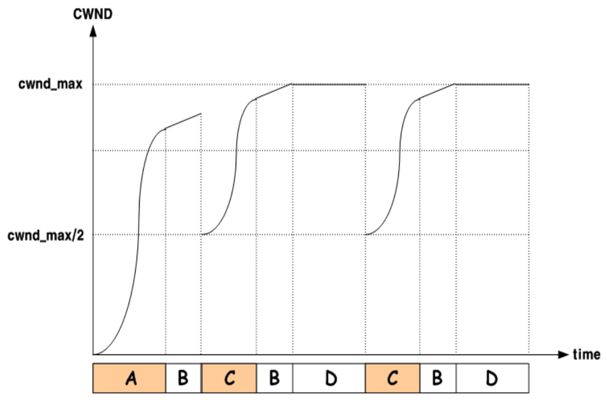

Fig.1. Congestion window evolution in the NDCA-TCP

The proposed NDCA-TCP modifies the TCP's congestion control mechanism to be used in TCP connections with large congestion windows. The legacy TCP congestion control algorithm is composed of Slow Start (SS) and Congestion Avoidance (CA) phases. Since the linearly increasing TCP congestion avoidance algorithm is not dynamic enough, the required packet drop rate to fill a Gigabit pipe using the current TCP protocol is beyond the limit of currently achievable fiber optic error rates. The proposed congestion control mechanism can be divided into four phases: A, B, C, and D according to the congestion window evolution characteristics, as shown in Fig. 1.

3.1 Algorithm for interval ‘A’

In the interval ‘A’, which is a slow-start (SS) period that begins when a TCP connection starts for the first time or the congestion window becomes 1 after timeout, the congestion window increases exponentially to find the maximum available bandwidth, by increasing the congestion window 1 whenever the TCP sender receives one ACK packet, as follows:

$$cwnd \leftarrow cwnd + 1 \tag{1}$$

However, a very rapid increase in the congestion window can cause a slow-start overshoot problem which results in multiple packet losses and reduction of the utilization, if the slow-start threshold (*ssthresh*) is incorrectly configured. Thus, in the proposed NDCA-TCP, we determine the appropriate *ssthresh* and try to prevent overshoot during the SS period by measuring the approximate available bandwidth using the RTT and ACK information, as explained in section 3.4.

3.2 Algorithm for interval ‘B’

The evolution of the congestion window in the interval ‘B’ is the same as that of the congestion avoidance (CA) phase of the legacy TCP Reno, as follows:

$$cwnd \leftarrow cwnd + \frac{1}{cwnd} \tag{2}$$

In this interval, the mechanism continuously measures the available bandwidth by the same algorithm in section 3.4. If the available bandwidth and corresponding *ssthresh* becomes larger than the current congestion window, the interval ‘B’ is changed into interval ‘C’ which is another slow start period in the CA interval. That is, if the measured available bandwidth becomes larger than the current bandwidth utilization, another slow-start tries to quickly adapt the situation to increase its bandwidth utilization.

3.3 Algorithm for interval ‘C’

The algorithm enters into the interval ‘C’ after triple duplicate ACKs, in which the congestion window increases exponentially by using another slow-start instead of linear increase as in the TCP Reno. The triple duplicate ACKs reduce the congestion window in half, and then the *cwnd* increases rapidly by slow-start until it reaches the measured *ssthresh*. If there is large bandwidth available to the TCP flow, the measured bandwidth would be larger than the current *cwnd* and the algorithm remains in slow-start phase. After some increase in *cwnd* in slow-start phase, the *cwnd* becomes larger than the *ssthresh*. At that point, the algorithm enters into an interval ‘B’. If there is any more available bandwidth in the routing path, an interval ‘C’ restarts due to increase in *ssthresh*. By repeating this process, the acquired bandwidth by a TCP flow stably saturates and converges to its maximum available bandwidth.

3.4 Algorithm for interval 'D'

When the legacy TCP operates in a fixed bandwidth network, the AIMD-type congestion control cannot fully utilize the available bandwidth due to its congestion window fluctuation. In order to cope with fixed bandwidth networks, the NDCA-TCP limits the *cwnd* increase when it decides that it reaches the maximum available bandwidth. One of the promising decision methods of maximum bandwidth is to use the BaseRTT which represents the minimum round trip time the TCP flow ever experienced. When the measured RTT exceeds some pre-determined threshold BaseRTT*α ($\alpha>1$),. the NDCA-TCP decides that some packets are accumulating inside the network and the maximum bandwidth is reached. At that point, the NDCA-TCP enters into interval 'D' and holds the *cwnd* increase until a smaller BaseRTT is measured or a packet loss occurs.

3.5 Bandwidth measurement

In order to measure the bandwidth used by a TCP flow for calculating the *ssthresh*, we use the measurement method proposed in [9] as shown in Fig. 2.

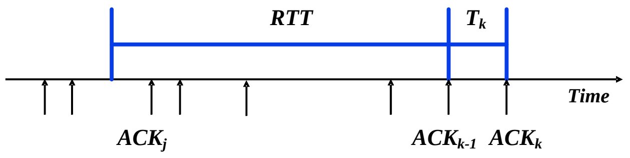

Fig. 2. Bandwidth measurement methods in NDCA-TCP

Let $BW_{E,k-1}$ be the estimated bandwidth for the *(k-1)*th ACK. Then, the newly measured bandwidth $BW_{E,new}$ and its moving average $BW_{E,k}$ become as follows,

$$BW_{E,new} = \frac{BW_{E,k-1} \cdot RTT + PacketSize_k}{RTT + T_k} \tag{3}$$

$$BW_{E,k} = (1-\beta) \cdot BW_{E,k-1} + \beta BW_{E,new} \tag{4}$$

In the above equation, T_k indicates the time interval between the *(k-1)*-th and *k*-th ACKs, and $PacketSize_k$ represents the total packet length newly acknowledged by the *k*-th ACK. β is a parameter used for moving average calculation ($0 < \beta < 1$). Then, the *ssthresh* is calculated as follow,

$$ssthresh = \frac{BW_{E,k} \cdot RTT}{Max.SegmentSize} \tag{5}$$

4. Simulation results

To evaluate the performance of the proposed NDCA-TCP algorithm, we performed various simulations by using ns-2 [10] under the simple network topology shown in Fig. 3. Link bandwidths for the three links are 2.5 Gbps, 2.5 Gbps, and 1.0 Gbps, and propagation delays are 1 msec, 1 msec, and 49 msec, respectively. We implemented a simulator for the NDCA-TCP congestion control algorithm by modifying the ns-2 TCP Reno code.

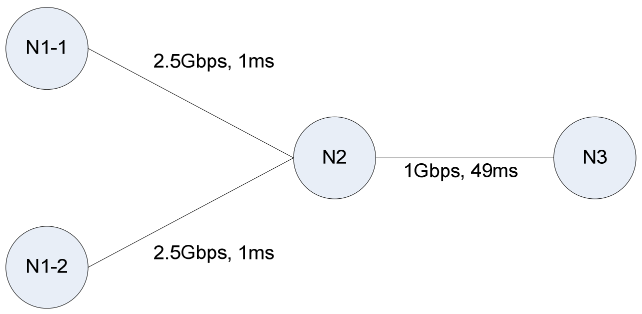

Fig. 3. Simulation topology for the NDCA-TCP

Figure 4 shows the pseudo-code of the simulator for the proposed mechanism. N is the parameter for deciding sampling times for that can be chosen by user. Thus, if *cwnd* is less than N (N = 10 in simulation), the TCP sender calculates its bandwidth estimation for each new arriving ACK packet. However, when the *cwnd* exceeds N, the sender's bandwidth estimation is performed for every N round trip time.

The congestion window is changed by the estimated *ssthreshold*. When the current *cwnd* is less than the estimated *ssthreshold*, the *cwnd* increases by 1 for each received ACK packet. But if *cwnd* exceeds the *cwnd_essthreshold*, the *cwnd* increases 1/cwnd per each received ACK packet. If the bandwidth reaches the maximum, the *cwnd* will be set to the current *cwnd*. When the TCP sender has three duplicated ACKs, the congestion window is reduced by half, similar to the legacy TCP Reno.

Fig. 5 shows the variation of congestion windows of the HS-TCP [3] and the NDCA-TCP for 50 seconds with packet loss probability 10^{-5}. The proposed NDCA-TCP shows no overflow loss in initial window variation, because the proposed algorithm prevents slow-start overshoot by configuring the *ssthresh* appropriately using bandwidth measurement. In addition, the figure shows that the congestion window variation in NDCA-TCP is very stable around the maximum bandwidth.

In Fig. 6, we compare the throughputs of the HS-TCP and the NDCA-TCP for packet loss probabilities from 10^{-4} to 10^{-7}. We performed the simulation for 30 minutes in simulation time unit and averaged the corresponding throughputs for each case by performing 10 simulations with different seeds for random number generation. From this figure, the throughput of the NDCA-TCP is much higher than that of the HS-TCP for relatively high packet loss rates. From this result, one can see that the NDCA-TCP is more effective in broadband networks with rather large packet loss and delay environments.

```
#when an ACK for a new packet arrives
if( cwnd < N)
  sampleInterval = 1;
else
  sampleInterval = (int) cwnd /N;
if( ackCnt >= sampleInterval ){
  deltaTime = currTime - preTime;
  timeRatio = deltaTime/(baseRTT + deltaTime);
  currBW = (packetSize×ackCount)/deltaTime;
  BW= (1- timeRatio)×preBW+ timeRatio×currBW;
  cwnd_essthresh = (int)(BW×baseRTT/maxSegSize);
  preBW = BW;
  preTime = currTime;
  ackCnt = 1;
}
else
  ackCnt += 1;
#when TCP sender calculates the cwnd_essthresh
if(cwnd <= cwnd_essthresh)
  cwnd +=1;
else
  if( BW == maxBW)
    cwnd = cwnd;
  else
    cwnd += 1/cwnd;
#when triple-duplicate ACKs arrive
 cwnd = cwnd/2;
 preBW = 0;
 cwnd_essthresh = 0;
```

Fig. 4. Pseudo-code for the proposed congestion control mechanism

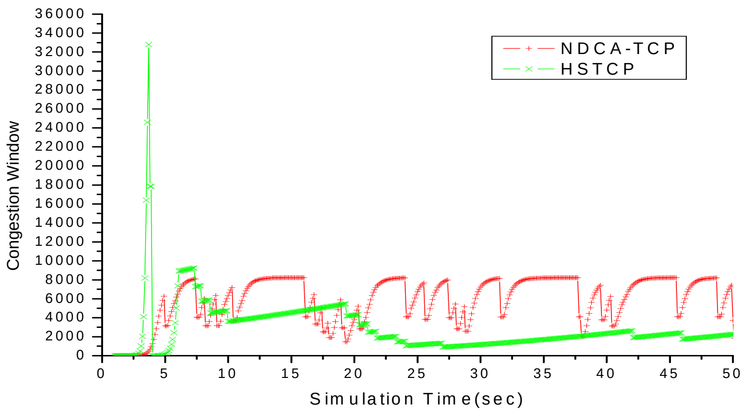

Fig. 5. Congestion window behavior during start-up.

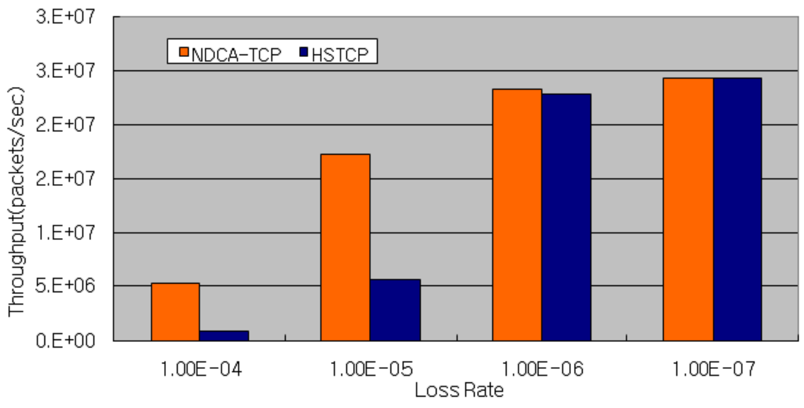

Fig. 6. Time averaged throughput versus loss rate

Fig. 7 shows the throughput fairness between two flows for loss probability 10^{-5} as measured by Jain's fairness index [11], which is expressed as follows,

$$\text{Fairness index} = \frac{\left(\sum x_i\right)^2}{n \times \sum (x_i)^2} \tag{6}$$

From the figure, one can see that the fairness of the NDCA-TCP converges more rapidly to 1.0 than the HS-TCP, which represents better fairness performance.

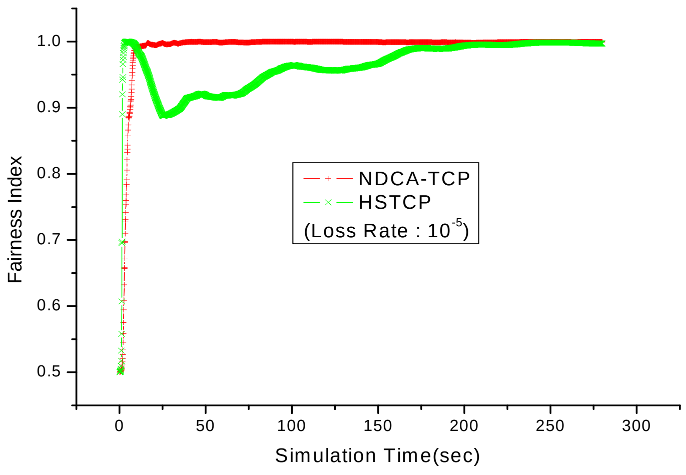

Fig. 7. Fairness index of the NDCA-TCP and the HSTCP

5. Conclusion

In this paper, we proposed a new congestion control algorithm for high BDP networks including fixed bandwidth networks. The proposed NDCA-TCP utilizes the slow-start algorithm during the congestion avoidance period, and effectively prevents the slow-start overshoot by simple bandwidth measurement. It also performs well in fixed bandwidth networks by virtually holding the *cwnd* variation after detecting bandwidth saturation using the BaseRTT method. The simulation results show that the NDCA-TCP has better throughput and fairness performance than HS-TCP. Thus, the NDCA-TCP can be used as one of the promising transport protocols for high BDP networks.

Acknowledgments. This work was supported by the IT R&D program of KCC [2009-S-039-02, Development of Satellite Return Link Access Core Technology for High Efficient Transmission].

References

1. Postel, J. Transmission Control Protocol. IETF RFC 793 (1981)
2. Allman, M., Paxson, V., Stevens, W. TCP Congestion Control. IETF RFC 2581 (1999)
3. Floyd, S. HighSpeed TCP for large Congestion windows. IETF RFC 3649 (2003)
4. Kelly, T. Scalable TCP: Improving performance in high speed wide area networks. http://www-lce.eng.cam.ac.uk/~ctk21/ scalable/ (2002)
5. Jin, C., Wei, D., Low, S. FAST TCP: Motivation, architecture, algorithms, performance. IEEE INFOCOM ’04 (2004) 2490-2501
6. Wang, R., Pau, G., Yamada, K., Sanadidi, M., Gerla, M. TCP startup performance in large bandwidth delay networks. IEEE INFOCOM ‘04 (2004) 796-805
7. Mascolo, S., Casetti, C., Gerla, M., Sanadidi, M., Wang, R. TCP Westwood: Bandwidth estimation for enhanced transport over wireless links. ACM/IEEE MobiCom ’01 (2001)
8. Mo, J., La, R., Anantharam, V., Walrand, J. Analysis and comparison of TCP Reno and Vegas. IEEE INFOCOM ’99 (1999)
9. Xu, K., Tian, Y., Ansari, N. TCP-Jersey for wireless IP communications, IEEE J. Select. Areas Communications, vol. 22 (2004) 747-756
10. The network simulator ns-2. http://www.isi.edu/nsnam/ns/
11. Jain, R. The art of computer systems performance analysis: techniques for experimental design, measurement, simulation and modeling. John Wiley & Sons (1991)
12. Tan, K., Song, J., Zhang, Q., Sridharan, M. Compound TCP: A Scalable and TCP-Friendly Congestion Control for High-speed Networks, PFLDdnet ‘06 (2006)