LETTER

Enhanced TCP Congestion Control with Higher Utilization in Under-Buffered Links**

Dowon HYUN[†a)], ***Nonmember and*** **Ju Wook JANG**[†*b)], ***Member***

SUMMARY TCP Reno is not fully utilized in under-buffered links. We propose a new TCP congestion control algorithm that can utilize the link almost up to 100% except the first congestion avoidance cycle. Our scheme estimates the minimum congestion window size for full link utilization in every congestion avoidance cycle and sends extra packets without touching TCP Reno congestion control. It has the same RTT fairness and the same saw-tooth wave as TCP Reno does. Our scheme does not affect competing TCP Reno flows since it uses only unused link capacity. We provide a simple mathematical modeling as well as ns-2 simulation results which show that the link utilization is improved by up to 19.88% for k=1/8 against TCP Reno when the buffer is k times the optimal buffer size. We claim that our scheme is useful for transmitting large amount of data in under-buffered links.

key words: *link utilization, TCP Reno, under-buffered link, congestion window*

1. Introduction

For a few long-lived TCP flows, Villamizar and Song [1] suggested a rule-of-thumb to obtain an optimal buffer size B_{opt} (packets) to guarantee full link utilization. For the given link with capacity C (packets/sec) and the minimum round trip time RTT_{min} (seconds), B_{opt} equals the value of $RTT_{min} \times C$ when a drop-tail queue is used. Let B denote the buffer size of a router. When $B \geq B_{opt}$, the link is fully utilized and called an over-buffered link. When $B < B_{opt}$, the link utilization is less than 100% and the link is called an under-buffered link.

According to the rule of thumb, B_{opt} grows linearly with the link capacity. For example, if $C = 1$ Tbps and $RTT_{min} = 250$ ms, $B_{opt} = 250$ Gbits. It is challenging to adjust this huge router buffer size with fast SRAM chips to the ever increasing link capacity. Queueing delays can be long, have high variance, and may destabilize the congestion control algorithm [2]. However, for an under-buffered link, TCP Reno [3] does not utilize the link 100% since congestion window size goes below the minimum congestion window size for full link utilization, W_{full}.

Consider an under-buffered link which requires B_{opt} of 1000 (packets) for full utilization but has $B = B_{opt}/4 = 250$ (packets). Figure 1 shows (a) congestion windows, (b) link utilization, and (c) buffer occupancy for this case. The calculated minimum congestion window size for full link utilization W_{full} is equal to $RTT_{min} \times C$ or B_{opt}. The broken line in Fig. 1(a) shows $W_{full} = 1000$ (packets). If the congestion window goes below this line, the link utilization also goes below 100% as shown in Fig. 1(b). Buffer occupancy during this under-utilization period is empty as in Fig. 1(c). As congestion window grows over W_{full}, link utilization becomes 100% and buffer occupancy grows. When congestion window eventually reaches $W_{max} = W_{full} + B = 1250$ (packets), the link and the buffer are both full, and an eventual packet loss causes 3-dup acks, which halves the congestion window to $W_{max}/2 = 625$ (packets). Since $W_{max}/2$ is less than W_{full}, utilization again goes below 100% and a new saw-tooth wave is started.

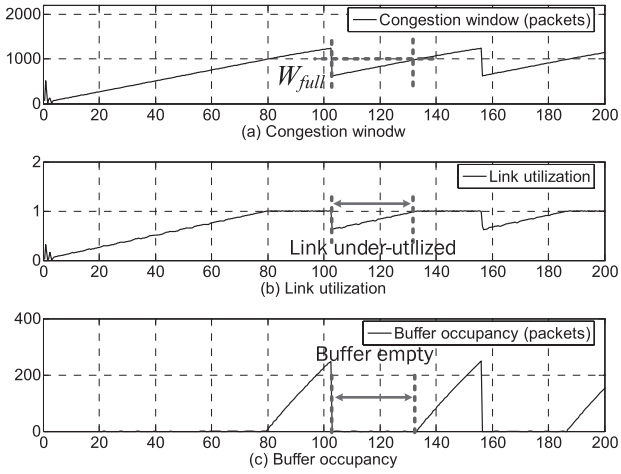

Fig. 1 TCP Reno operation in an under-buffered link.

In this letter, we propose a new TCP congestion avoidance algorithm that can utilize link almost up to 100% and simultaneously has the same RTT fairness and the saw-tooth wave as TCP Reno during under-utilized period.

2. Proposed TCP Scheme

We propose a new TCP scheme that can be fully utilized in an under-buffered link. First, we describe the estimation of the congestion window size for full link utilization. We then describe how this estimated value is used in our scheme. Note that we do not estimate the maximum and available

Manuscript received November 18, 2011.

[†]The authors are with the Department of Electronic Engineering, Sogang University, Seoul 121-742, Korea.

[*]Corresponding author.

[**]This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2011-0018081) and Sogang University Research Grant of 2010 (201010019).

a) E-mail: snatcher@monet1.sogang.ac.kr

b) E-mail: jjang@sogang.ac.kr

DOI: 10.1587/transcom.E95.B.1427

link capacity.

2.1 Estimation of the Congestion Window Size for Full Link Utilization

RTT increases at the time when the link is full and the buffer starts to fill up. However, RTT in TCP Reno is unsuitable to detect the exact time RTT increases due to small changes and oscillation. Thus, we use a smoothed RTT using log-likelihood (LSRTT) which reflects increasing trend of RTT better than that of RTT [4]. Let $C(t)$ denote the congestion indicator as shown below:

$$C(t) = \frac{LSRTT(t)}{RTT_{min}} > \alpha \tag{1}$$

where t is the number of transmission round and RTT_{min} is the minimum of all measured round trip times as in TCP Vegas [5]. If $C(t) \leq \alpha$, we regard that there is no congestion. When $C(t)$ is larger than α where $\alpha > 1$, we deduce that congestion exists.

We denote by W^i_{full} the minimum congestion window size for full link utilization at the time when $C(t) > \alpha$. Superscript i is the number of the congestion avoidance cycle. W^i_{full} can be obtained as shown below:

$$W^i_{full} = cwnd^i_{proposed}(t^*) \times \beta, t^* = \min\{t | C(t) > \alpha\} \tag{2}$$

where $cwnd^i_{proposed}(t)$ is the congestion window of the proposed TCP scheme. We describe $cwnd^i_{proposed}(t)$ in detail in next sub-section. W^i_{full} may be overestimated since $\alpha > 1$. The overestimated W^i_{full} may cause aggressive transmission against TCP Reno. Thus, we add another parameter β, which reduces overestimated W^i_{full} where $\beta < 1$.

2.2 The Proposed TCP Congestion Avoidance Control

Figure 2 shows the congestion window evolution schematic of the proposed TCP scheme. The shaded region in Fig. 2 is not used in TCP Reno in an under-buffered link. Therefore, our scheme sends extra packets to utilize the shaded region without disturbing TCP Reno. To do so, we add a new congestion window $extra^i(t)$, which transmits extra packets. The proposed TCP congestion avoidance control algorithm can be represented as shown below:

$$cwnd^i_{proposed}(t) = cwnd^i(t) + extra^i(t) \tag{3}$$

where t is the number of transmission round and superscript i is the number of the congestion avoidance cycle. $cwnd^i(t)$ is the congestion window of TCP Reno. Note that we do not touch the congestion control algorithm of TCP Reno. Therefore, $cwnd^i(t)$ is updated the same as TCP Reno.

Our TCP scheme updates $extra^i(t)$ according to the following rules:

- If $C(t) \leq \alpha$ and $cwnd^i_{proposed}(t) < W^{i-1}_{full}$, $extra^i(t)$ increases quickly to acquire spare bandwidth (from t_0 to t_1 in Fig. 2).

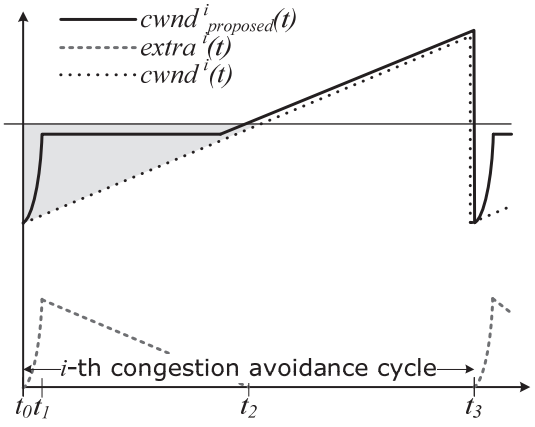

Fig. 2 The congestion window evolution and extra transmitted packets of the proposed TCP scheme during one congestion avoidance cycle.

- If $C(t) \leq \alpha$, $extra^i(t)$ decreases linearly to maintain $cwnd^i_{proposed}(t) = W^{i-1}_{full}$ (from t_1 to t_2 in Fig. 2).
- If $C(t) > \alpha$ or $cwnd^i_{proposed}(t) \geq W^{i-1}_{full}$, $extra^i(t)$ is set to 0 (from t_2 to t_3 in Fig. 2).

In the first step, $extra^i(t)$ increases $cwnd^i_{proposed}(t)$ up to the value of W^{i-1}_{full} according to the increasing parameter a. It acquires available bandwidth quickly. If we set $extra^i(t)$ equal to W^{i-1}_{full} at the instant of the starting fast recovery, it can cause packet loss since the transmission speed of burst packet without ACK clocking at sender can be faster than the packet receiving speed of router. It can cause serious performance degradation. Thus, we increase $extra^i(t)$ like a slow start. In the second step, $extra^i(t)$ decreases linearly to maintain $cwnd^i_{proposed}(t)$ equal to W^{i-1}_{full} until $C(t) \leq \alpha$. In this step, the proposed TCP scheme uses only an unused link capacity. Therefore, our TCP scheme neither increases RTT nor affects existing TCP Reno flows. In the third step, $extra^i(t)$ is set to 0 when $C(t) > \alpha$ or $cwnd^i_{proposed}(t) \geq W^{i-1}_{full}$. Our scheme performs additive increase by $cwnd^i(t)$ as TCP Reno does.

We summarize the proposed TCP scheme for $extra^i(t)$ as in (4).

$$extra^i(t+1) = \begin{cases} a \times cwnd^i(t), \text{if } C(t) \leq \alpha,\ cwnd^i_{proposed}(t) < W^{i-1}_{full} \\ \max\left\{0, W^{i-1}_{full} - cwnd^i(t)\right\}, \text{if } C(t) \leq \alpha \\ 0, \text{ if } C(t) > \alpha \text{ or } cwnd^i(t) > cwnd^i_{proposed}(t) \end{cases} \tag{4}$$

The proposed TCP scheme halves its congestion window $cwnd^i_{proposed}(t)$ just as TCP Reno does when the packet loss occurs.

Our scheme has the same RTT compared with TCP Reno since we use only an unused link capacity except during the first congestion avoidance cycle. And it can adapt to the network situation changes since W^i_{full} is estimated in every congestion avoidance cycle. The proposed TCP scheme works just as TCP Reno does in an over-buffered link.

3. Mathematical Modeling

We provide a mathematical modeling of the link utilization for TCP Reno during congestion avoidance in order to evaluate our scheme against TCP Reno. Our scheme keeps congestion window equal to or more than each congestion avoidance cycle except for the first several rounds at the starting fast recovery. Thus, the link utilization is close to 100% for long-lived TCP flows. The link utilization for TCP Reno depends on the buffer size, B. Let us define $k = B/B_{opt}$, where $0 \le k < 1$. If $k \ge 1$, the link utilization is always 100%. If $k < 1$, the ratio of the amount of transmitted data between our TCP scheme and TCP Reno can be obtained as in (5).

$$Ratio = \frac{\frac{3}{8}W_{max}^2 + \frac{1}{2}\left(W_{full} - W_{max}/2\right)^2}{\frac{3}{8}W_{max}^2} = \frac{4}{3}\frac{\left(k^2+k+1\right)}{(k+1)^2} \quad (5)$$

The numerator is the amount of transmitted data in our TCP scheme and the denominator is the amount of transmitted data of TCP Reno. In (5), $W_{full} = B_{opt}$ and $W_{max} = (1 + k)B_{opt}$ since $W_{max} = W_{full} + B$. $\frac{3}{8}W_{max}^2$ is the amount of transmitted packets in one saw-tooth wave of TCP Reno [6]. $\frac{1}{2}(W_{full} - W_{max}/2)^2$ is the amount of extra transmitted packets (the shaded region in Fig. 2) by our TCP scheme. We ignore the amount of extra data which would be sent in the first several RTT during each congestion avoidance cycle. The ratio of transmitted data is $1 < Ratio \le 4/3$, where $0 \le k < 1$. If the router has no available buffer space for the flow, i.e., $k = 0$, *Ratio* is 4/3, which is a gain of 33.3% in terms of the link utilization. For the case of $k = 1/4$, *Ratio* is 1.12, a gain of 12%. In Sect. 4, we show that simulation results are close to our modeling.

4. Simulation Result

In this section, we present an experimental comparison of the proposed TCP scheme and TCP Reno. We implement and simulate our TCP scheme using ns-2 [7]. We consider a simple topology which consists of a single TCP sender, TCP receiver, and router. The sender's access link is higher and faster than the receiver's bottleneck link, which is C=100 Mbps and RTT_{min}=80 ms. The router maintains a single drop-tail buffer. In this topology, the calculated window size for full link utilization is W_{full}=1000 (packets). We set a to 1.1 in (4).

We perform simulation for various k, i.e., k=1/8, k=1/4, k=1/2, and k=3/4. Table 1 shows the link utilization for the proposed TCP scheme compared with TCP Reno for a single flow. In these cases, the proposed TCP scheme utilizes the link almost up to 100% with α=1.02 and β=0.97. We claim that our scheme is useful for transmitting large amount of data in an under-buffered link.

Figure 3 shows the congestion window, the link utilization, RTT, and the buffer occupancy evolution of the proposed TCP scheme against TCP Reno for each single flow when k=1/4. Our scheme sends extra packets (dotted-line in Fig. 3(a)) and can utilize the link almost up to 100%, starting with the second congestion avoidance cycle. We can verify that our scheme has the same saw-tooth wave, RTT, and the buffer occupancy as TCP Reno.

Figure 4 shows the congestion window, the link utilization, and RTT evolution of the proposed TCP scheme and TCP Reno when k=1/4. We have TCP Reno flow start at 0 second, and then have the proposed TCP flow start at 100 seconds. We stop TCP Reno flow at 400 seconds. While we assure that our scheme can share bandwidth fast and fairly with TCP Reno just as other two TCP Reno flows do, the link utilization is 0.970 which is higher than that of two TCP

Table 1 Comparison of the link utilization between the proposed TCP scheme and TCP Reno.

	Link utilization			
	TCP Reno	Proposed TCP scheme	Gain(%) Simulation	Gain(%) From (5)
$B = \frac{1}{8}B_{opt}$	0.830	0.995	19.88	20.16
$B = \frac{1}{4}B_{opt}$	0.891	0.997	11.90	12
$B = \frac{1}{2}B_{opt}$	0.963	0.999	3.73	3.7
$B = \frac{3}{4}B_{opt}$	0.992	0.999	0.71	0.68

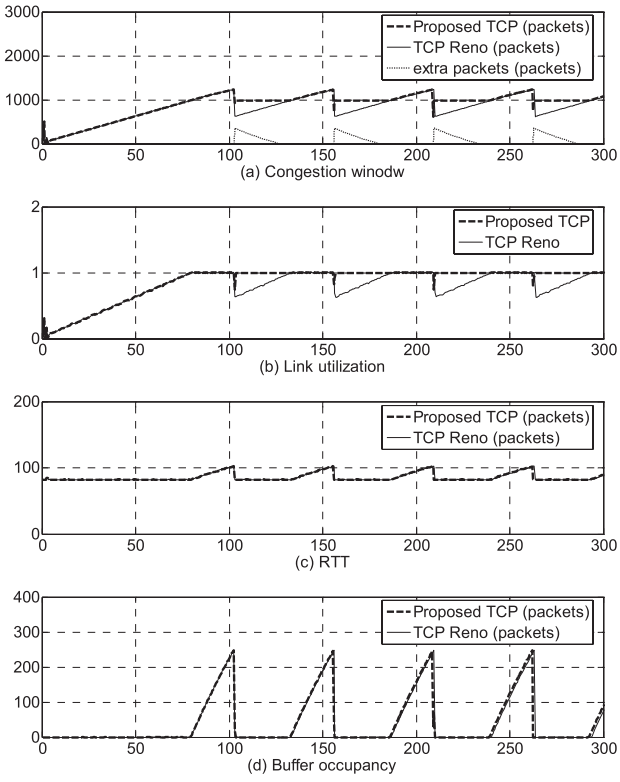

Fig. 3 Comparison between the proposed TCP scheme and TCP Reno.

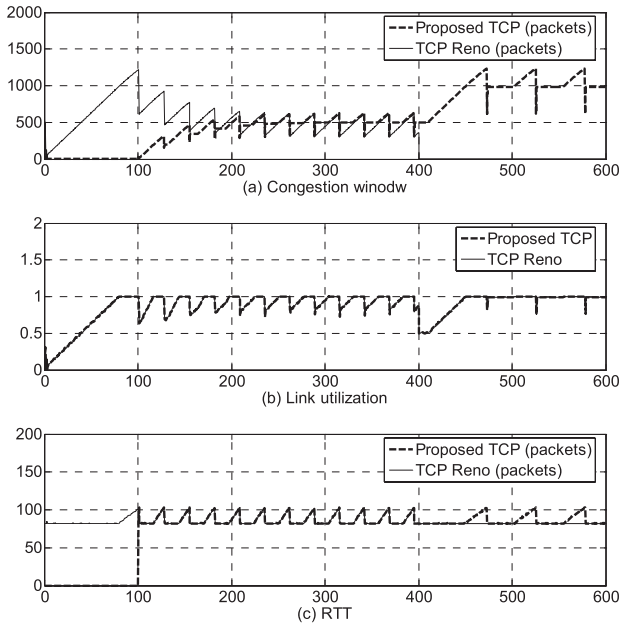

Fig. 4 Comparison of the proposed TCP scheme and TCP Reno.

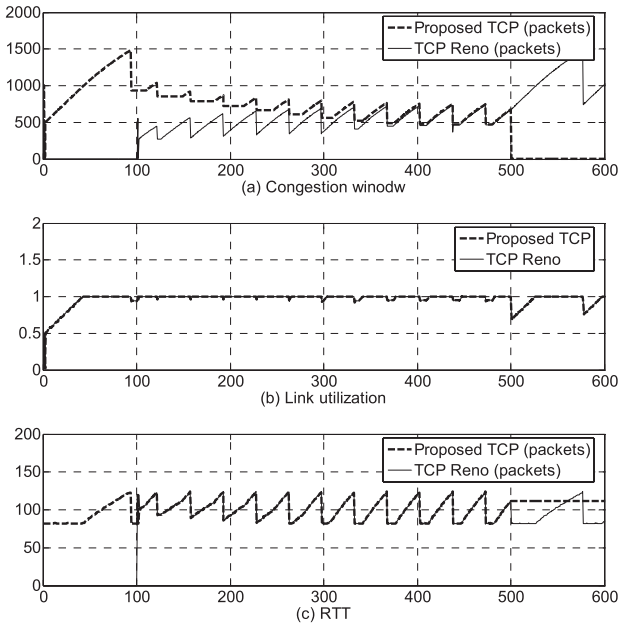

Fig. 5 Comparison of the proposed TCP scheme and TCP Reno.

Reno flows, 0.945. We can also verify that our TCP scheme can adapt to the network situation changes.

Figure 5 shows the congestion window, the link utilization, and RTT evolution of two proposed TCP flows when k=1/2. We verify that two proposed TCP flows can share the bandwidth fast and fairly just as other TCP Reno flows do, simultaneously with higher link utilization, 0.983.

5. Conclusion

Our new TCP congestion control scheme shows the ns-2 simulation results in this letter. We used only unused link capacity to verify that the link utilization can be achieved almost up to 100% from the second congestion avoidance cycle. Our scheme has the same RTT fairness and the sawtooth wave as TCP Reno. However, it does not affect existing TCP Reno flows. The link utilization is improved up to 19.88% when k=1/8 against TCP Reno. We claim that our TCP scheme is useful for transmitting large amount of data in an under-buffered link.

References

[1] C. Villamizar and C. Song, "High performance TCP in ANSNET," ACM Comput. Commun. Rev., vol.24, no.5, Oct. 1994.
[2] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," Proc. ACM SIGCOMM 2004, Portland, OR, Aug. 2004.
[3] V. Jacobson, "Congestion avoidance and control," ACM SIGCOMM, pp.314–329, 1988.
[4] D.W. Ngwenya and G.P. Hancke, "The effects of using change detection algorithms in estimation of the average RTT," Proc. SATNAC, Western Cape, South Africa, Sept. 2004.
[5] L.S. Brakmo, S.W. O'Malley, and L.L. Peterson, "TCP Vegas: Net techniques for congestion detection and avoidance," Proc. ACM SIGCOMM, London, UK, Aug. 1994.
[6] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," ACM SIGCOMM Comput. Commun. Rev., vol.27, no.3, July 1997.
[7] The Network Simulation — NS2. http://www.isi.edu/nsnam/ns/