On Congestion Control for Streaming Real-time Applications over Wireless Networks with Bandwidth Variation

Ho-Jin Lee, Jae-Han Jeon, and Jong-Tae Lim
Division of Electrical Engineering
Korea Advanced Institute of Science and Technology
Guseong-dong, Yuseong-gu, Daejeon, 305-701
Email: jtlim@stcon.kaist.ac.kr

***Abstract*—A congestion control algorithm for non-TCP flows such as streaming real-time applications should be TCP-friendly. In particular, in the wireless networks, TCP-friendly congestion control algorithm should be based on differentiation of packet losses due to congestion and wireless link error to improve the network utilization. Several algorithms in the wireless networks have been proposed to improve the networks performance. However, they do not consider the bandwidth variation. In particular, the abrupt increase of the bandwidth can cause network under-utilization. In this paper, we introduce a new TCP-friendly congestion control algorithm using the available bandwidth estimator over the wireless networks. We show that the proposed algorithm adapts the abrupt bandwidth variation rapidly, comparing with the existing algorithm. Moreover, the simulation results show that the proposed algorithm utilizes the link bandwidth efficiently.**

I. Introduction

The majority of the Internet flows uses Transmission Control Protocol (TCP)-based protocols such as HyperText Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), and File Transfer Protocol(FTP). However, non-TCP flows such as streaming real-time applications are constantly growing [1].

If non-TCP flows do not adjust their transmission rates, they treat competing TCP-flows in an unfair manner. Since TCP flows reduce their transmission rates to avoid congestion, non-TCP flows can take the remained bandwidth from TCP flows. This situation leads to starvation of TCP traffic. Thus, it is important that new congestion control algorithm for non-TCP flows has compatible traffic regulations with TCP flows [2].

In the wireless networks, several congestion control algorithms such as ARC [3] and RCS [4] for TCP-friendliness have been proposed. They can avoid the degradation of the throughput due to the wireless link error. However, they do not consider the bandwidth variation over wireless networks. To effectively manage the limited resources, the additional data rate channel is dynamically assigned to users, and to enhance the wireless link capacity, the bandwidth is also controlled based on radio condition [5]-[6]. Moreover, in the future, there are mixtures of heterogeneous wireless access technologies co-existed such as WLAN and 3G cellular networks. This will further increase the dramatic variation of bandwidth and delay [7]. In particular, the abrupt increase of the bandwidth can cause network underutilization. To improve the performance of wireless networks with the bandwidth variation, we propose a new TCP-friendly congestion control algorithm for a non-TCP flow based on the available bandwidth estimator and Explicit Congestion Notification (ECN).

II. Congestion Control Algorithm

In this paper, we consider the single-hop wireless access networks. Moreover, since a base station or an access point is liable to be a bottleneck due to the difference of network bandwidth between wired and wireless links, we assume that the outgoing link of the base station is the bottleneck link, where ECN is implemented. The bandwidth of the bottleneck link, C, is piecewise constant with jumps where the duration between two consecutive jumps is larger than the settling time of the system.

TCP-friendliness means that the throughput of a non-TCP flow should be approximately the same as that of a TCP flow under the same conditions of round trip time (RTT) and packet loss rate [8]. In order to investigate TCP-friendliness of the proposed algorithm for non-TCP flows, we assume that each flow has the same RTT, τ_s. Hence, we describe our system as a discrete-time model where the duration of a time slot is determined by τ_s.

Let $w(n)$ be the window size of a TCP flow and $R(n)$ be the number of transmitted packets of a non-TCP flow at time n. Let $q(n)$ be the queue length and $m(n)$ be the packet marking probability by the ECN router at time n respectively. The marking probability $m(n)$ is described by $m(n) = max_p \frac{q(n)-min_{th}}{max_{th}-min_{th}}$ where max_p is the maximum packet marking probability, and min_{th}, max_{th} are the minimum threshold, the maximum threshold of the queue length respectively [11]. Since $m(n)$ depends on the queue length, we can also express the packet marking probability, $m(n)$, as the congestion level. The queue length can be written as follows :

$$q(n+1) = q(n) + N_t w(n) + N_r R(n) - C(n)\tau_s \qquad (1)$$

where N_t is the number of TCP flows, N_r is the number of non-TCP flows, and $C(n)$ is the link bandwidth at time n.

We use the dynamic model of Additive Increase Multiplicative Decrease (AIMD) data rate control on the average sense to achieve TCP-friendliness. The probability that at

least one packet among the window size, $w(n)$, is marked can be described by $1-(1-m(n))^{w(n)}$. Since $m(n)$ is typically small, we show $1-(1-m(n))^{w(n)} \simeq m(n)w(n)$ [12]. Then, the window size will increase by 1 packet with probability $1-m(n)w(n)$ and be halved with probability $m(n)w(n)$. Thus, on the average sense, $w(n)$ can be written as follows:

$$w(n+1) = \left(1 - \frac{m(n)w(n)}{2}\right) w(n) + 1 - m(n)w(n) \quad (2)$$

Hence, in the proposed algorithm, $R(n+1)$ of a non-TCP flow should be adjusted with the equation (2) for TCP-friendliness as follows:

$$R(n+1) = \left(1 - \frac{m(n)R(n)}{2}\right) R(n) + 1 - m(n)R(n) \quad (3)$$

To obtain the level of congestion, $m(n)$, we use a number of ECN marked acknowledgement (ACK) packets. Let num_{Ack} be the number of arriving ACK packets during one RTT. A sender of a non-TCP flow counts the number of ECN marked ACK packets, num_e, in num_{Ack}. Then, in the average sense, the overall ratio of num_e and num_{Ack} is equal to the probability that the ECN router randomly marks packets, i.e.,

$$\frac{num_e}{num_{Ack}} = \frac{R(n)m(n)(1-e(n))}{R(n)(1-e(n))} = m(n) \quad (4)$$

where $e(n)$ is the stochastic process of the wireless packet loss probability at time n. The wireless packet loss probability depends on many factors such as distance and multipath interference. Hence, by collecting ECN messages during RTT, we can derive $m(n)$, irrespective of the wireless packet loss. ARC [3] and TFRC [9] need explicit information such as the statistics of RTT and packet loss probability to set the transmission rate. Moreover, RCS [4] uses the low priority dummy packets to differentiate between packet losses caused by network congestion and caused by the wireless link error. Here, the dummy packet transmission may produce overhead. However, the proposed algorithm does not need any additional information and the dummy packets except ECN messages to differentiate the reason of a packet loss.

Now, we should consider the abrupt variation of the bandwidth. If the bandwidth increase large, the available bandwidth of a flow may be much larger than its current transmission rate. Then, it would be shame to increase the rate linearly, thereby incurring a long delay to achieve the full link utilization. On the other hand, in case that the bandwidth decreases abruptly, congestion collapse can happen. Thus, to achieve the full link utilization and to avoid the congestion under the bandwidth variation rapidly, we estimate the available bandwidth of the bottleneck link based on the inter-arriving time of ACK packets corresponding to each data packet as follows :

$$ABW_i = \frac{L_i}{t_i - t_{i-1}} \quad (5)$$

where t_i is the arriving time of the ith ACK packet and L_i is the packet size of the ith data packet. Note that the inter-arriving time increases or decreases according to the bandwidth variation. Thus, sources do not need to know several parameters such as $C(n)$, N_t, and N_r to estimate the available bandwidth of the bottleneck link.

Since ABW_i fluctuates due to the varying load at the bottleneck link, we define the average estimated available bandwidth as follows:

$$\overline{ABW} = \frac{\sum_{i=2}^{num_{Ack}} \frac{L_i}{num_{Ack}-1}}{\sum_{i=2}^{num_{Ack}} \frac{t_i - t_{i-1}}{num_{Ack}-1}} = \frac{\sum_{i=2}^{num_{Ack}} L_i}{\sum_{i=2}^{num_{Ack}} t_i - t_{i-1}} \quad (6)$$

Let $\overline{ABW}(n)$ be the value of $\overline{ABW}$ at time n. Then, with $\overline{ABW}(n)$, $R(n+1)$ is adjusted under the bandwidth variation as follows :

$$R(n+1) = \begin{cases} \text{if} \quad 0 \le m(n)R(n) < 1, \\ \left(1 - \frac{m(n)R(n)}{2}\right) \overline{ABW}(n)\tau_s + 1 - m(n)R(n) \\ \\ \text{else} \\ R(n)/2 \end{cases} \quad (7)$$

Consequently, the transmission rate of a non-TCP flow every RTT is set by $Tr = \frac{R(n)}{RTT}$. This algorithm is effective to achieve the full link utilization rapidly with providing TCP-friendliness. In section III, we describe $\overline{ABW}(n)$ as a discrete-time model and analyze the performance of the proposed algorithm.

When a connection begins, our proposal is similar to the slow start phase of TCP congestion control algorithm. A sender starts at the low transmission rate, 1 packet/RTT, and it doubles its transmission rate every its RTT. A sender continues to increase Tr exponentially until it receives an ECN marked ACK packet. After receiving the ECN marked ACK packet, the sender halves the transmission rate and regulates Tr with equation (7).

If no ACK packet arrives during one RTT, we halve the transmission rate and hold our current transmission rate until the sender receives an ACK packet. However, if no ACK packet is received for a certain timeout period, the sender sets its transmission rate to 1 packet/RTT. Then, the sender grows the transmission rate exponentially every its RTT until it receives an ECN marked ACK packet. In our proposal, ACK packets are used only for congestion control and no retransmissions are performed due to tight time constraints of the real-time applications. RTT and the timeout period are estimated by the same method with TCP.

III. Asymptotic Stability

In this section, we find the equilibrium point and its condition for the asymptotic stability. We describe $\overline{ABW}$ as a discrete-time model. For the purpose of simplicity, we assume $L_i = 1$ in (6). Then, according to [10],

$$\overline{ABW}(n)\tau_s = \frac{C(n)\tau_s R(n)}{N_t w(n) + N_r R(n)} \quad (8)$$

Let $\gamma = \frac{C(n)\tau_s}{N_t w(n) + N_r R(n)}$. If all flows utilize the link bandwidth fully and the link bandwidth does not change, $\gamma = 1$ and $\overline{ABW}(n)\tau_s = R(n)$. Then, the proposed algorithm follows TCP AIMD data rate control under the constant bandwidth, i.e., TCP-friendly.

If the link bandwidth increases large and the queue length decreases below min_{th} at time n, $R(n+1) \cong \gamma R(n)+1$ with $\gamma > 1$. That is, our proposal can increase the transmission rate very fast. On the other hand, if the link bandwidth decreases abruptly, the value of γ is less than 1. It is also effective to avoid congestion. As the link utilization approaches to 100%, γ approaches 1. Then, the proposed algorithm converges to the fair share with TCP flows. Therefore, we can guarantee TCP-friendliness and the full link utilization under the bandwidth variation.

Now, we find the equilibrium point of $R(n)$, $w(n)$, and $q(n)$. For simplicity, we assume that $min_{th} = 0$. Then, we can express $m(n)$ as

$$m(n) = \frac{max_p}{max_{th}} q(n) = \beta q(n) \tag{9}$$

where $0 < \beta < 1$. From the assumption that the bandwidth is piecewise constant, $C(n)$ can be expressed as C_s at the equilibrium point. Let q_s, w_s, and R_s be the equilibrium point of $q(n)$, $w(n)$, and $R(n)$, respectively. From (1), (2), and (7), we have

$$q_s = q_s + N_t w_s + N_r R_s - C_s \tau_s \tag{10}$$

$$w_s = \left(1 - \frac{\beta q_s w_s}{2}\right) w_s + 1 - \beta q_s w_s \tag{11}$$

$$R_s = \left(1 - \frac{\beta q_s R_s}{2}\right) \frac{C_s R_s}{N_t w_s + N_r R_s} \tau_s + 1 - \beta q_s R_s \tag{12}$$

The solutions of the equations of (10)-(12), is written by

$$q_s = \frac{1}{\beta} \frac{2N^2}{C_s \tau_s (C_s \tau_s + 2N)}, \quad w_s = -1 + \sqrt{1 + \frac{2}{\beta q_s}}$$
$$R_s = -1 + \sqrt{1 + \frac{2}{\beta q_s}} \tag{13}$$

where $N = N_t + N_r$. The throughput of a TCP flow and a non-TCP flow, which are represented by w_s/τ_s and R_s/τ_s, are same as $\frac{C_s}{N}$. Therefore, our proposal shares the bandwidth fairly with TCP flows, i.e., TCP-friendly.

Now, we investigate the asymptotic stability at the equilibrium point. Let $X(n)$ be the state vector by $X(n) = [R(n) - R_s \;\; w(n) - w_s \;\; q(n) - q_s]$. Let $k_1 = -\frac{C_s \tau_s (C_s \tau_s + 2N)}{2N^2}$ and $k_2 = \frac{(C_s \tau_s)^2 - 2N^2}{C_s \tau_s (C_s \tau_s + 2N)}$ where $k_1 < 0$ and $k_2 < 1$. Linearization of the system at the equilibrium point yields

$$X(n+1) = AX(n) \tag{14}$$

with

$$A = \begin{pmatrix} k_2 - \frac{N_r}{N} + \frac{N_r}{C_s \tau_s + 2N} & -\frac{N_t}{N} + \frac{N_t}{C_s \tau_s + 2N} & \beta k_1 \\ 0 & k_2 & \beta k_1 \\ N_r & N_t & 1 \end{pmatrix} \tag{15}$$

Note that if $\frac{C_s \tau_s}{N} > 0.618$, then $-1 < k_2 < 1$. Thus, given C_s, τ_s, and N such that $\frac{C_s \tau_s}{N} > 0.618$, the equilibrium point is asymptotically stable if

$$0 < \beta < \frac{k_2 - 1 - \frac{N_r}{N} + \frac{N_r}{C_s \tau_s + 2N}}{k_1 (N_t + 1)} \tag{16}$$

IV. Responsiveness

At the previous section, it is shown that the available bandwidth is very helpful to utilize the bandwidth efficiently. In this section, we investigate how many slots our proposal needs to achieve the full link utilization under the abrupt bandwidth increase, where one slot means the RTT of each flow. Each flow controls its transmission rate every slot. The smaller number of slots, the better responsiveness.

First of all, we consider the case that all flows increase just 1 packet per a slot. Let n_{s1} be the number of slots to achieve the full link utilization at the first case. We assume that all flows use the link bandwidth, C, fully and fairly before the link bandwidth increase. When the link bandwidth increases from C to λC, then n_{s1} can be described as follows :

$$n_{s1} = \left\lceil \frac{(\lambda - 1) C \tau_s}{N} \right\rceil \tag{17}$$

where $\lceil x \rceil$ is the smallest integer larger than x and $\lambda \gg 1$. This means that n_{s1} is linearly proportional to λ. As λ becomes larger, it takes longer time to achieve the full link utilization.

Secondly, we consider the case that TCP flows increase 1 packet per a slot and non-TCP flows use our proposal. Like the first case, we assume that all flows use the link bandwidth fully and fairly before the variation of the link bandwidth. Let n_{s2} be the number of slots to achieve the full link utilization at the second case. We also express $m(n)$ as $m(n) = \frac{max_p}{max_{th}} q(n) = \beta q(n)$. We set $C(n-1) = C$, $C(n) = \lambda C$, $w(n) = R(n) = \frac{C\tau_s}{N}$, and $q(n) = \frac{1}{\beta} \frac{2N^2}{C\tau_s (C\tau_s + 2N)}$. Hence, we obtain this relation:

$$R(n+l) \cong \frac{\lambda^l C \tau_s}{N_t + \lambda^{l-1} N_r},$$
$$n_{s2} = \left\lceil 1 + \frac{\ln(\frac{N_t}{N_r}(\lambda - 1))}{\ln \lambda} \right\rceil + 1 \tag{18}$$

where $l < n_{s2}$.

Proof: The majority of flows in the Internet are based on TCP. Moreover, since each sender transmits at least more than one packet per one slot, we assume that $\frac{C\tau_s}{N} \gg 1$ and $N_t \gg N_r$. First of all, we use the mathematical induction to prove

$$R(n+l) \cong \frac{\lambda^l C \tau_s}{N_t + \lambda^{l-1} N_r} \tag{19}$$

In case of $l = 1$,

$$\begin{aligned} R(n+1) &= \left(1 - \frac{N}{C\tau_s + 2N}\right) \frac{\lambda C \tau_s}{N} + \left(1 - \frac{2N}{C\tau_s + 2N}\right) \\ &= \left(1 - \frac{1}{\frac{C\tau_s}{N} + 2}\right) \frac{\lambda C \tau_s}{N} + \left(1 - \frac{2}{\frac{C\tau_s}{N} + 2}\right) \\ &\cong \frac{\lambda C \tau_s}{N} + 1 \\ &\cong \frac{\lambda C \tau_s}{N} = \frac{\lambda C \tau_s}{N_t + N_r} \end{aligned} \tag{20}$$

Thus, $R(n+l)$ is true for $l = 1$.

Now we show that if $R(n+l)$ holds when $l=\delta$, then it also holds when $l=\delta+1$. In case of $l=\delta$, $R(n+\delta)$ can be written as follows:

$$R(n+\delta)=\frac{\lambda^{\delta}C\tau_s}{N_t+\lambda^{\delta-1}N_r} \tag{21}$$

Since $\lambda \gg 1$, the marking probability would be 0 for $l>1$. Thus,

$$R(n+\delta+1)=\frac{\lambda^{\delta+1}C\tau_s}{N_t+\lambda^{\delta}N_r}+1 \cong \frac{\lambda^{\delta+1}C\tau_s}{N_t+\lambda^{\delta}N_r} \tag{22}$$

Then, if $R(n+\delta)$ is true, $R(n+\delta+1)$ is also true. Thus, we obtain the result:

$$R(n+l) \cong \frac{\lambda^{l}C\tau_s}{N_t+\lambda^{l-1}N_r} \tag{23}$$

Now, we investigate n_{s2} to be satisfied the following condition :

$$N_t w(n+n_{s2})+N_r R(n+n_{s2}) \geq \lambda C\tau_s \tag{24}$$

Since $w(n)=\frac{C\tau_s}{N}$ at time n, the window size of a TCP flow at slot $(n+1)$ is $\frac{C\tau_s}{N}+1$. From $\frac{C\tau_s}{N} \gg 1$, we approximate that $w(n+1) \approx \frac{C\tau_s}{N}$. Similarly, $w(n+n_{s2})$ is equal to $\frac{C\tau_s}{N}$. Then,

$$\begin{aligned} &N_t w(n+n_{s2})+N_r R(n+n_{s2}) \\ &= N_t\frac{C\tau_s}{N}+N_r\frac{\lambda^{n_{s2}}C\tau_s}{N_t+\lambda^{n_{s2}-1}N_r} \end{aligned} \tag{25}$$

Thus, from (24)

$$N_t\frac{C\tau_s}{N}+N_r\frac{\lambda^{n_{s2}}C\tau_s}{N_t+\lambda^{n_{s2}-1}N_r} \geq \lambda C\tau_s \tag{26}$$

We rewrite (26) as follows:

$$\begin{aligned} &\frac{N_t}{N}+N_r\frac{\lambda^{n_{s2}}}{N_t+\lambda^{n_{s2}-1}N_r} \geq \lambda \\ &\Leftrightarrow N_t^2+\lambda^{n_{s2}-1}N_rN_t-NN_t\lambda \geq 0 \\ &\Leftrightarrow N_t+\lambda^{n_{s2}-1}N_r-N\lambda \geq 0 \\ &\Leftrightarrow N_r\lambda^{n_{s2}-1} \geq \lambda N-N_t \end{aligned} \tag{27}$$

Since $\lambda N-N_t \geq \lambda N_t-N_t$, we obtain

$$N_r\lambda^{n_{s2}-1} \geq (\lambda-1)N_t \tag{28}$$

From $\lambda \gg 1$ and $N_t \gg N_r$,

$$n_{s2} \geq \frac{\ln(\frac{N_t}{N_r}(\lambda-1))}{\ln\lambda}+1 \tag{29}$$

Note that n_{s2} is a positive integer. Moreover, we should compensate the approximation. By adding 1 to n_{s2}, we obtain the result, (18). ■

The value of n_{s2} is not sensitive λ. It is also independent on $C\tau_s$. Thus, we conclude that our proposal has the better responsiveness compared with n_{s1}.

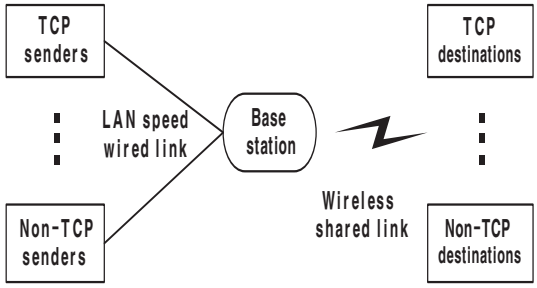

Fig. 1. Network configuration

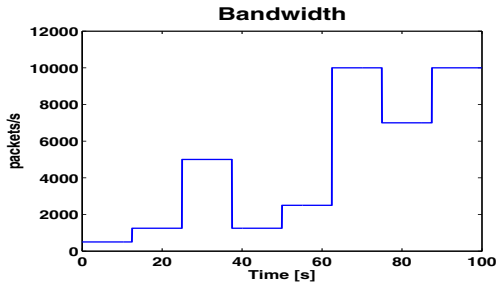

Fig. 2. Bandwidth of bottleneck link

V. Simulation

In this section, we show the performance of our proposal in terms of TCP-friendliness and the link utilization. The network topology with a single bottleneck is a dumbbell as shown in Fig. 1. The wired link is a LAN speed network. As the bandwidth of the LAN speed links is typically much higher than that of the wireless shared link, the wireless link is assumed to be the bottleneck link. The bottleneck link has the bandwidth variation described in Fig. 2, which is shared by 10 flows. They have the same round trip time, 0.05 sec. TCP flows adopt TCP NewReno. Let P_e be the average wireless packet loss probability during the connection. We set $min_{th}=10$ packets, $max_{th}=700$ packets, and $max_p=0.1$. The throughput is in the unit of packets/s.

TABLE I
Average throughput and Link utilization w.r.t., P_e

P_e(%)	0.1	0.3	0.5	0.8
Proposal	599.03	710.59	775.47	836.11
TCP(Proposal)	375.03	291.49	247.73	200.04
Link utilization(%)	97.9	97.8	97.6	97.3
RCS	524.31	608.12	569.86	481.53
TCP(RCS)	386.66	293.12	256.23	215.05
Link utilization(%)	94.2	89.9	81.8	70.3
TCP(only)	417.50	318.08	255.68	218.92
Link utilization(%)	89.1	68.0	54.8	47.8

In Table I, we have 3 simulation results. First, there are 6 TCP-flows and 4 non-TCP flows using the proposed algorithm, represented by Proposal and TCP(Proposal). Second, there are 6 TCP flows and 4 non-TCP flows using RCS, represented by RCS and TCP(RCS). Finally, there are only 10 TCP flows, represented by TCP(only). Table I shows the average throughput and link utilization of our proposal and RCS. It is illustrated that our

proposal provides the higher average throughput and link utilization than RCS does, as P_e increases. RCS halves the transmission rate at each packet loss and recovers from unnecessary rate reduction by using the dummy packets. However, as the packet loss probability increases, RCS decreases the transmission rate more frequently and throughput degrades. On the other hand, the throughput of our algorithm does not degrade significantly. This is because our algorithm does not reduce the transmission rate at each packet loss. We properly manage the transmission rate by the level of congestion. Hence, our algorithm takes the increased available bandwidth while the TCP throughput decreases.

As shown Table I, our proposal also achieves the higher average link utilization, irrespective of the bandwidth variation. Fig. 3 shows that our proposal utilizes the link bandwidth fully, but RCS does not, i.e., our proposal has better responsiveness. In particular, at 60s, the proposed algorithm still utilizes the link bandwidth fully due to the good responsiveness, but the link utilization of RCS degrades significantly. This is because RCS does not increase its transmission rate quickly enough when the link bandwidth increases suddenly. Moreover, the average throughputs of TCP(Proposal) and TCP(RCS) are similar to TCP(only). This means that both our proposal and RCS do not penalize TCP flows. That is the proposed algorithm is TCP-friendly. Therefore, the proposed algorithm utilizes the bandwidth more efficiently compared to RCS without penalizing TCP flows.

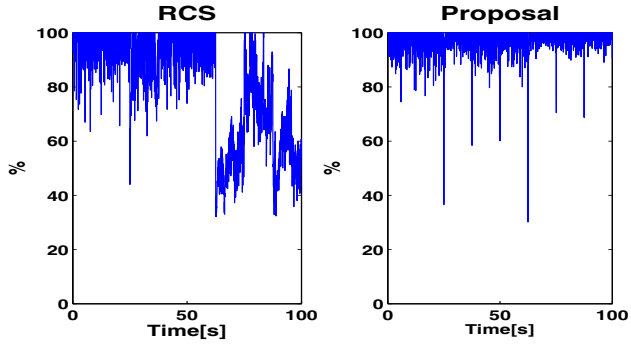

Fig. 3. Link utilization of RCS and Proposal, $P_e = 0.5\%$

VI. Conclusion

In this paper, we propose a new congestion control algorithm to adjust the transmission rate according to the bandwidth variation. It derives the level of congestion from a number of marked ECN ACK packets and uses the available bandwidth estimator. The simulation results show that our proposal utilizes the link bandwidth efficiently without penalizing TCP flows.

Acknowledgment

This work was supported by the SRC/ERC program of MOST/KOSEF (grant ♯R11-1999-008).

References

[1] J. Widmer, R. Denda, and M. Mauve, "A survey on TCP-friendly congestion control," *IEEE Network*, vol. 15, pp. 28-37, May-June 2001.

[2] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE/ACM Trans. Networking*, vol. 7, pp. 458-472, Aug. 1999.

[3] Ö. B. Akan and I. F. Akyildiz, "ARC : The analytical rate control scheme for real-time traffic in wireless networks," *IEEE/ACM Trans. Networking*, vol. 12, pp. 634-644, Aug. 2004.

[4] I. F. Akyildiz, Ö. B. Akan, and G. Morabito, "A rate control scheme for adaptvie real-time applications in IP netwroks with lossy links and long round trip times," *IEEE/ACM Trans. Networking*, vol. 13, pp. 554-567, June 2005.

[5] J. -C. Moon and B. -G. Lee, "Rate-adaptive snoop : a TCP enhancement scheme over rate-controlled lossy links," *IEEE/ACM Trans. Networking*, vol. 1, pp. 603-615, June 2006.

[6] R. Ludwig, A. Gurtov, and F. Khafizov, "TCP over second (2.5G) and third (3G) generation wireless networks," IETF RFC 3481, 2003.

[7] Q. Zhang, W. Zhu, and Y.-Q. Zhang, "End-to-end QoS video delivery over wireless Internet," *Proc. of The IEEE*, vol. 93, pp. 123-134, Jan. 2005.

[8] L. Cai, X. Shen, J. W. Mark, and J. Pan, "QoS support in wireless/wired networks using the TCP-friendly AIMD protocol," *IEEE Trans. Wireless. Commun.*, vol. 5, pp. 469-480, Feb. 2006.

[9] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proc. ACM SIGCOMM 2000*, pp. 43-56, Aug. 2000.

[10] H.-J. Lee, H.-J. Byun, and J.-T. Lim, " TCP window control for variable bandwidth in wireless cellular networks," *IEEE Commun. Lett.*, vol. 11, pp. 152-154, Feb. 2007.

[11] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, pp. 397-413, Aug. 1993.

[12] X. Wang and D. Y. Eun, "Global stability of TCP/RED with many-flows:a numerical approach," in *Proc. IEEE ICC 2005*, vol. 1, pp. 369-373, May 2005.