A MULTIPATH CONGESTION CONTROL SCHEME FOR HIGH-QUALITY MULTIMEDIA STREAMING

Sunghee Lee[1] and Kwangsue Chung[2]
Trust Media Research Lab, Electronics and Telecommunications Research Institute
218 Gajeong-ro, Yuseong-gu, Daejeon, Korea
sh2@etri.re.kr
Department of Communication Engineering, Kwangwoon University
20 Kwangwoon-ro, Nowon-gu, Seoul, Korea
kchung@kw.ac.kr
*Corresponding author: Kwangsue Chung

Received June 14, 2016; revised September 30, 2016; accepted November 28, 2016; published January 31, 2017

Abstract

As network adaptive streaming technology becomes increasingly common, transport protocol also becomes important in guaranteeing the quality of multimedia streaming. At the same time, because of the appearance of high-quality video such as Ultra High Definition (UHD), preventing buffering as well as preserving high quality while deploying a streaming service becomes important. The Internet Engineering Task Force recently published Multipath TCP (MPTCP). MPTCP improves the maximum transmission rate by simultaneously transmitting data over different paths with multiple TCP subflows. However, MPTCP cannot preserve high quality, because the MPTCP subflows slowly increase the transmission rate, and upon detecting a packet loss, drastically halve the transmission rate. In this paper, we propose a new multipath congestion control scheme for high-quality multimedia streaming. The proposed scheme preserves high quality of video by adaptively adjusting the increasing parameter of subflows according to the network status. The proposed scheme also increases network efficiency by providing load balancing and stability, and by supporting fairness with single-flow congestion control schemes.

Keywords: multipath congestion control, MPTCP, multimedia streaming, fairness, load balancing

This work was supported by ICT R&D program of MSIP/IITP. [R0101-16-293, Development of Object-based Knowledge Convergence Service Platform using Image Recognition in Broadcasting Contents]

https://doi.org/10.3837/tiis.2017.01.022 **ISSN : 1976-7277**

1. Introduction

While deploying streaming, preventing discontinuity of video playback as well as preserving high quality is important for high-quality multimedia streaming that has a high bitrate, such as Ultra High Definition (UHD). Meanwhile, network adaptive streaming technology to prevent discontinuity of video playback is becoming increasingly common. As network adaptive streaming technology changes the quality of video according to the transmission rate, transport protocol is an important element that decides the quality of streaming service. Increasing the speed of transmission rate determines the length of time until streaming reaches the original video quality. The decrease of the transmission rate degrades the quality of video. Moreover, recursive increase and the decrease of the transmission rate degrades the quality of user experience. Therefore, deploying high-quality multimedia streaming requires a transport protocol that can quickly increase the transmission rate, and preserve the transmission rate at a high value.

The Internet Engineering Task Force (IETF) has approved Multipath TCP (MPTCP) as an experimental standard for multipath transmission. MPTCP has attracted attention as a means of serving high-quality multimedia, as by simultaneously using multiple network interfaces, it improves network capacity at a low price [1-3]. MPTCP allows the creation of multiple simultaneous TCP subflows between two end hosts, where each subflow performs a congestion control function on a path. However, MPTCP is not efficient for a high-quality multimedia streaming service, because upon detecting a packet loss, TCP subflows halve the congestion window, and as TCP increases the congestion window per RTT at the congestion avoidance phase, it takes a long time until the congestion window reaches the maximum [4, 5]. Therefore, despite using multiple paths, MPTCP slowly increases the transmission rate. This means that when a network adaptive streaming technique is used, users watch low quality video until the congestion window reaches the maximum.

To guarantee a high-quality multimedia streaming service over a multipath environment, a new multipath congestion control scheme is required that allows each subflow to quickly increase the transmission rate while preventing packet losses. Also, the multipath congestion control scheme should basically satisfy the design goals defined in IEEE RFC 6356 to improve network efficiency and provide network stability, which are as follows [6]:

Goal 1 (Improve throughput.) A multipath flow should perform at least as well as a single path flow would on the best of the paths available to it.

Goal 2 (Do no harm.) A multipath flow should not take up more capacity from any of the resources shared by its different paths, than if it were a single flow using only one of these paths. This guarantees that it will not unduly harm other flows.

Goal 3 (Balance congestion.) A multipath flow should move as much traffic as possible off its most congested paths, subject to meeting the first two goals.

In this paper, we propose a multipath media transport protocol (MPMTP) for high-quality multimedia streaming services. MPMTP quickly increases the transmission rate and prevents

packet losses by adjusting the increasing parameter of each subflow according to the network status. Also, the proposed scheme satisfies the three design goals for a multipath congestion control scheme, by adjusting the congestion window of subflows according to the RTT, packet loss rate, and congestion window of each subflow.

2. Background and related works

multipath TCP is a set of extensions to regular TCP that allows one TCP connection to be spread across multiple paths. MPTCP distributes load through the creation of separate subflows across potentially disjoint paths. **Fig. 1** shows the protocol stacks of TCP and MPTCP [3].

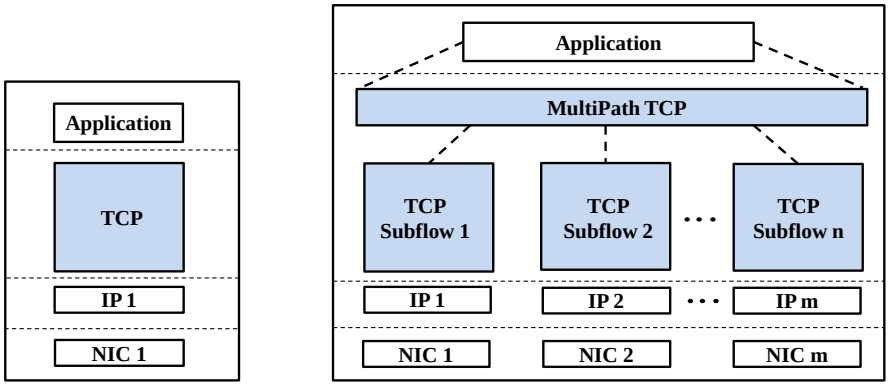


Fig. 1. Comparison between TCP and MPTCP protocol stacks

MPTCP can achieve Goal 1 by simply running standard TCP congestion control on each subflow. However, this solution is unsatisfactory, because when the paths taken by its different subflows share a common bottleneck, it gives the multipath flow an unfair share. Also, multipath congestion control has to take on a role that is normally associated with routing, namely moving traffic onto paths that avoid congestion hotspots, so that the Internet will be better able to accommodate localized surges in traffic, and use all available capacity. However, MPTCP cannot shift its traffic, because the congestion control algorithm of each subflow operates independently. Thus, MPTCP cannot satisfy design goals 2 and 3 for a multipath congestion control scheme.

Several MPTCP enhancement schemes, such as equal weight transmission control protocol (EWTCP), COUPLED, and Linked Increases, have suggested simple extensions of TCP's congestion control to satisfy the three design goals [6-8]. However, these schemes are designed to be backward compatible with regular TCP, which slowly increases and fluctuates the transmission rate as TCP linearly increases the congestion window, and upon detecting a packet loss at the congestion avoidance phase, halves the transmission rate. Thus, it takes quite a long time until a sender can serve high-quality multimedia, such as Ultra High Definition (UHD) contents; and frequent quality changes occur that degrade user-perceived quality. Several MPTCP variants, such as multipath binomial and MPCUBIC, have been proposed to quickly increase transmission rates and by extending TCP variants [9-11] to multipath, reduce the variation in transmission rates. However, previous MPTCP variants decrease the quality of the media streaming service, because of the burst packet losses caused by overshooting of the transmission rate.

2.1 MPTCP enhancements

MPTCP just runs a regular TCP congestion control algorithm on each subflow. The congestion avoidance algorithm of TCP consists of additive increase behavior when no loss is detected, and multiplicative decrease when a loss event is observed, as follows:

$$Inc.: w(n+1) = w(n) + \frac{\alpha}{w(n)},$$
$$Dec.: w(n+1) = w(n) - \beta w(n) \quad (1)$$

where, w(n) is the congestion window, α is the increasing parameter, and β is the decreasing parameter. However, if MPTCP were to run regular TCP congestion control at a common bottleneck, then the multipath flow would obtain twice as much throughput as the single path flow. EWTCP has been proposed to provide fairness with single path flows [3]. It provides this by adjusting the increasing parameter according to the number of subflows, as follows:

$$Inc.: w(n+1) = w(n) + \frac{\alpha}{w(n)},$$
$$Dec.: w_r(n+1) = w_r(n) - \beta w_r(n) \quad (2)$$

where, $w_r(n)$ is the congestion window on subflow r , and N is the number of subflows. By choosing $\alpha = \frac{1}{\sqrt{N}}$, the multipath flow gets the same throughput as a regular TCP at the bottleneck link [11]. Although EWTCP can be fair to regular TCP traffic, it would not make very efficient use of the network, because EWTCP cannot move traffic off the congested path to an uncongested path.

COUPLED has been proposed to provide load balancing while guaranteeing fairness with single-path flows [6]. COUPLED moves traffic by adjusting the congestion window based on the total congestion window size, w_T, as follows:

$$Inc.: w_r(n+1) = w_r(n) + \frac{1}{w_T(n)},$$
$$Dec.: w_r(n+1) = w_r(n) - \beta w_T(n) \quad (3)$$

Consider a case where all paths have the same loss rate p. Each window w_r is made to increase on ACKs, and made to decrease on drops; and in equilibrium, the increases and decreases must balance out, as follows:

$$(1-p)\frac{1}{w_T} = p\frac{w_T}{2} \quad (4)$$

If we approximate p as a small value, we can calculate w_T as follows:

$$w_T = \sqrt{\frac{2(1-p)}{p}} \cong \sqrt{\frac{2}{p}} \tag{5}$$

where, $\sqrt{\frac{2}{p}}$ is the congestion window of regular TCP in equilibrium. Thus, if the subflows of COUPLED share a bottleneck with a single path flow, COUPLED solves the fairness problem by reducing w_T to regular TCP.

For the case that the loss rates are not all equal, let p_r be the loss rate on path r, and let p_{min} be the minimum loss rate seen over all paths. The increase and decrease amounts are the same for all paths, but paths with higher p_r will see more frequent decreases. Thus, the traffic on a congested path will move to an uncongested path, and the loss rate across the whole network will tend to balance out. However, COUPLED has shortcomings when the RTTs are unequal, because COUPLED only considers the packet loss rate as a sign of congestion. The throughput of TCP is $\frac{\sqrt{2/p}}{RTT}$. However, if a path has long delay and low drop rates, whereas another has short delay and high drop rates, COUPLED moves all traffic to the path that has a high drop rate. But in some cases, throughput could degrade as much as the long delay.

Linked Increase has been proposed to prevent degradation of throughput when paths have different RTT and drop rates [7]. Linked Increase increases the congestion window by $\frac{a}{w_T}$ and decreases by $\frac{1}{2}$, as follows:

$$Inc.: w_r(n+1) = w_r(n) + \frac{a_r}{w_T(n)},$$
$$Dec.: w_r(n+1) = w_r(n) - \frac{w_r(n)}{2} \tag{6}$$

a_r is calculated based on RTT of each path to achieve the design goals of the multipath congestion control scheme, as follows:

$$a_r = \frac{\hat{w}_T max\left\{\frac{\hat{w}_r}{RTT^2}\middle| r \in R\right\}}{\left(\sum_r \frac{\hat{w}_r}{RTT}\right)^2} \tag{7}$$

where, $\hat{w}_r$ is the equilibrium congestion window, and $\hat{w}_T$ is the total congestion window in equilibrium on path r. a_r decreases as the delay on a path increases. Thus, traffic on the long delay paths moves to the short delay path.

MPTCP enhancement schemes have been proposed to support load balancing and fairness with single-path flow. However, these TCP-based congestion control schemes slowly increase and linearly vary the transmission rate as they increase the congestion window, and halve the transmission rate upon detecting a packet loss.

TCP-based congestion control schemes have difficulty in serving high-quality media streaming, even if multiple paths are used

2.2 MPTCP variants

To improve the performance of Additive Increase/Multiplicative Decrease (AIMD), many alternatives to the regular TCP congestion control have been proposed such as Binomial congestion control, CUBIC, Compound TCP, TCP-FIT and CUBIC-FIT.

SQuare-RooT (SQRT) and Inverse-Increase/Additive Decrease (IIAD) are two of the binomial algorithm. SQRT and IIAD provide smooth throughput compared to regular TCP by adjusting the increasing and decreasing parameter of AIMD. CUBIC quickly increases congestion window and reduces burst packet losses by adjusting congestion window based on cubic function. Compound TCP uses not only packet loss but also delay information to increase congestion window. Therefore, Compound TCP can mitigate low link utilization problem of delay based congestion control scheme when competing with loss based congestion control schemes. TCP-FIT uses N virtual TCP Reno sessions, which can be adjusted according to the end-to-end queuing delay to achieve high throughput. CUBIC-FIT introduces the idea of delay-based TCP into the TCP CUBIC algorithm framework. CUBIC-FIT can improve throughput performance over wireless networks more than plain CUBIC and maintain graceful friendliness with widely deployed plain CUBIC servers [12-14].

For the multipath environment, MPTCP variants have been proposed for multimedia streaming services. MPTCP variants extend the TCP variants that have been proposed to deploy multimedia streaming services on multipath. Multipath binomial algorithms have been proposed to mitigate the quality change in multimedia streaming services [9]. The multipath binomial is an extended binomial congestion control scheme that adjusts the congestion window in a square root (SQRT) or inverse increase additive decrease (IIAD) manner to prevent frequent quality changes. We can express the increase and decrease rules of the binomial congestion control scheme as follows:

$$\begin{aligned} Inc.: w(n+1) &= w(n) + \frac{\alpha}{w^k(n)},\ \alpha > 0 \\ Dec.: w(n+1) &= w(n) - \beta w^l(n),\ 0 < \beta < 1 \end{aligned} \tag{8}$$

For the SQRT control algorithm, the parameters are set as $k = \frac{1}{2}$, $l = \frac{1}{2}$, $\alpha = 1$, and $\beta = \frac{1}{2}$. For the IIAD control algorithm, the parameters are set as $k = 1$, $l = 0$, $\alpha = 2$, and $\beta = 1$. The basic ideas of SQRT and MPIIAD are increasing the congestion window more aggressively and decreasing it more conservatively than AIMD. Multipath SQRT (MPSQRT) and multipath IIAD (MPIIAD) extend SQRT and IIAD to multipath. Each subflow of MPSQRT and MPIIAD adjusts the congestion window based on the SQRT and MPIIAD, respectively, and uses an additional parameter δ to provide load balancing and fairness with single-path flow, as follows:

$$\begin{aligned} Inc.: w_r(n+1) &= w_r(n) + \frac{\delta_r \alpha_r}{w_r^k(n)}, \\ Dec.: w_r(n+1) &= w_r(n) - \beta_r w_r^l(n) \end{aligned} \tag{9}$$

Fig. 2 shows that MPSQRT and MPIIAD can thus reduce the variation in transmission rate.

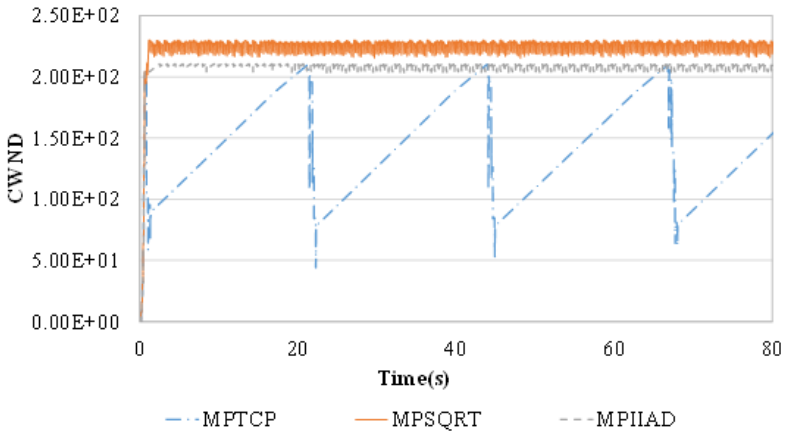


Fig. 2. Comparison of congestion windows of MPTCP, MPSQRT, and MPIIAD

However, MPSQRT and MPIID generate frequent packet losses, because upon a packet loss, they decrease the congestion window less than does MPTCP. **Table 1** compares the number of packet losses between MPTCP, MPSQRT, and MPIIAD.

Table 1. Comparison of packet losses

Protocol	MPTCP	MPSQRT	MPIIAD
Number of packet losses	384	6460	1544

MPCUBIC has been proposed to quickly increase the transmission rate on a multipath [10]. MPCUBIC extends the CUBIC, which adjusts the congestion window in a concave and convex manner to quickly increase the transmission rate while mitigating burst packet losses. The subflows of MPCUBIC use a cubic function to quickly increase the transmission rate, as follows:

$$w_r(t) = \delta_r C(t_r - k_r)^3 + w_r^{max} \tag{10}$$

where, w_r^{max} is the congestion window when the latest packet loss occurs, t_r is the elapsed time from the last packet loss, k_r is the time it takes for congestion window to reach w_r^{max}, C is the cubic parameter, and δ_r is a parameter to provide fairness with single path flows. The subflows of MPCUBIC continue concave growth of the transmission rate until the congestion window reaches w_r^{max}, to quickly increase the transmission rate while preventing overshooting of the transmission rate. After the congestion window reaches w_r^{max}, the function turns into a convex profile, and convex window growth begins. During convex increasing, the transmission rate initially grows slowly, to find the new maximum nearby. After a period of slow growth, if it does not find the new maximum, w_r^{max}, then it switches to a faster increase under the assumption that w_r^{max} is further away. **Fig. 3** shows the evolution

of the congestion window of MPCUBIC when using two paths. The summations of link capacity are 10 Mbps, 50 Mbps, and 100 Mbps, respectively.

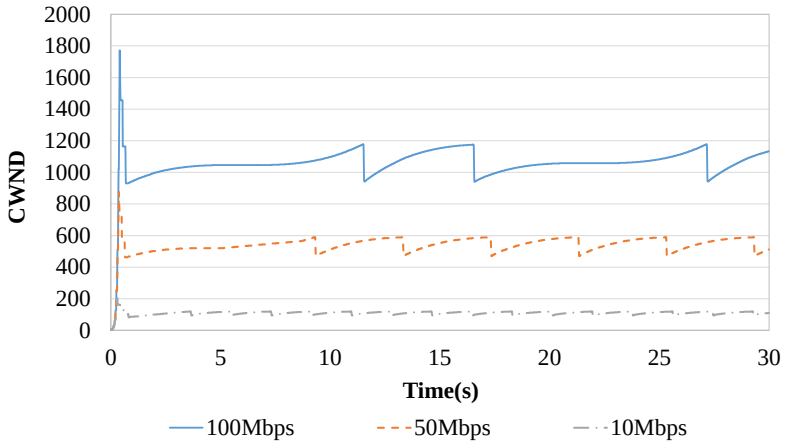


Fig. 3. Comparison of the congestion windows in MPCUBIC according to bandwidth change

MPCUBIC quickly increases the congestion window up to available bandwidth. However, deciding w_r^{max} based on the congestion window when a loss event occurs does not precisely reflect the current network congestion status. Also, the convex behavior overshoots the transmission rate, and results in burst packet losses. **Fig. 4** shows the number of packet losses when the total link capacity is 100 Mbps.

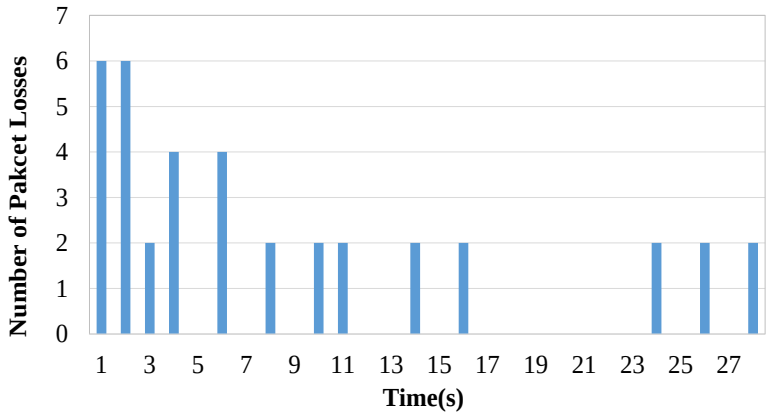


Fig. 4. Comparison of packet losses according to time change

MPTCP variants extended the single-path TCP variants to multi-path. However, MPTCP variants lead to burst packet losses as the transmission rate overshoots and generate frequent packet losses. Thus, to guarantee a high-quality multimedia streaming service, we need a new multipath congestion control scheme that quickly increases the transmission rate and prevents overshooting of the transmission rate, while satisfying the design goals of a multipath congestion control scheme.

3. Multipath media transport protocol (MPMTP)

This section presents a new multipath transport protocol for high-quality multimedia streaming service. To quickly provide high-quality multimedia while preventing burst packet losses, the proposed scheme adjusts the subflow congestion window in a concave, convex, or linear manner according to the RTT and packet loss event. The proposed scheme also calculates parameters to provide load balancing and fairness with single-path flows. We first describe the architecture of the proposed scheme for high-quality multimedia streaming over multipath. Then, we describe algorithms for multipath multimedia transport protocol in detail.

Fig. 5 illustrates the protocol stacks for the multipath multimedia transport protocol. *MPMTP subflow* adjusts the congestion window in a concave-convex-linear manner based on RTT and packet loss rate of each path to quickly increase the transmission rate, while maintaining a low packet loss rate. *MPMTP Controller* has three major functions: *Connection Management*, *Load Balancing*, and *Fairness Support*. *Connection Management* initiates an MPMTP connection, associates a new subflow with an existing MPMTP, and closes the MPMTP connection as MPTCP does. *Load Balancing* calculates a parameter B to move the traffic from a congested path to an uncongested path according to the packet loss rate of each path. *Fairness Support* calculates a parameter δ for fair share of the network bandwidth using path flows based on the equilibrium of MPMTP subflows, RTT, and packet loss rate. *MPMTP Controller* monitors RTT, packet loss rate, and congestion window of each subflow on every ACK for load balancing, and the fairness support.

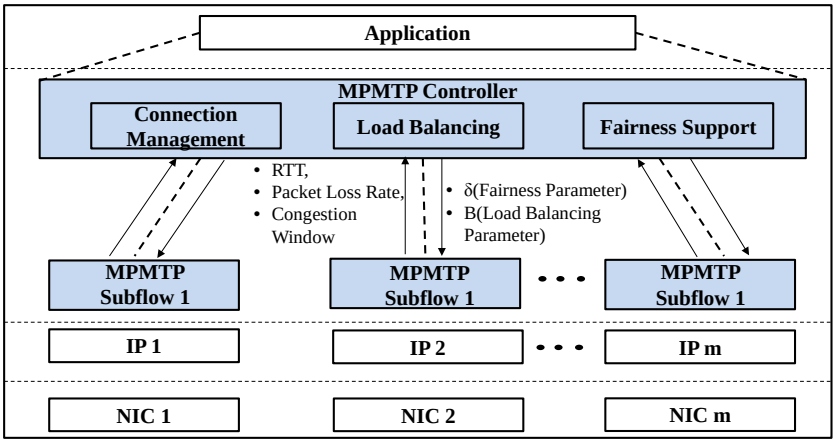


Fig. 5. Protocol stacks for MPMTP

3.1 MPMTP subflow algorithm

The objective of MPMTP is guaranteeing the quality of a high-quality media streaming service while providing network efficiency and stability. The proposed scheme includes a congestion control algorithm for subflows and a mechanism to provide load balancing and fairness support with single path flows. Our congestion control algorithm for subflows aims to quickly transmit high bit rate multimedia and prevent burst packet losses by adjusting the congestion window in a concave-convex-linear manner, as **Fig. 6** shows.

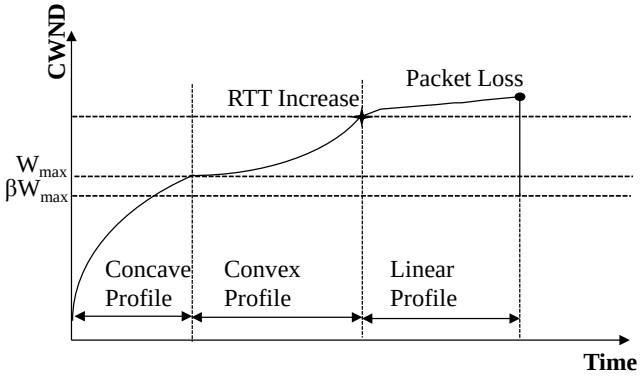


Fig. 6. Concept of congestion control algorithm for MPMTP subflow

In the concave profile, the proposed scheme continues concave growth of the transmission rate until the congestion window reaches w_{max}, to quickly increase the transmission rate. w_{max} is the congestion window where the loss event occurred. After the congestion window reaches w_{max}, the proposed scheme switches to a convex profile. During convex increasing, the transmission rate initially grows slowly, to find the new maximum nearby; and after some time of slow growth, if it does not find a new maximum, w_{max}, then it guesses that the new maximum is further away, so it switches to a faster increase. However, deciding w_{max} based on the congestion window when the loss event occurs does not precisely reflect the current network congestion status. Thus, if the available bandwidth decreases below w_{max} as the network status becomes worse, burst packet losses occur with the overshooting of the transmission rate in the concave profile. Meanwhile, if the bandwidth increases above w_{max}, burst packet losses occur with aggressive increase of transmission rate in the convex profile.

To prevent burst packet losses according to the changes in the network status, if an RTT increment is detected, the proposed scheme switches to a linear profile. In the linear profile, the congestion window linearly increases, until a packet loss occurs to prevent overshooting of the transmission rate. If a packet loss occurs, the proposed scheme decreases the congestion window to βw, and sets the w_{max} to w

Fig. 7 shows the operation of the proposed scheme when the network status is unchanged. RTT_{cur} is the current RTT, and RTT_{min} is the RTT when the RTT increment is detected. After a packet loss occurs, MPMTP subflow continues growth in concave manner to quickly find the available bandwidth. In this case, because the network status is unchanged, RTT increases before the congestion window reaches w_{max}. If an increment in RTT is detected, the proposed scheme linearly increases the congestion window by gradually decreasing the increasing parameter.

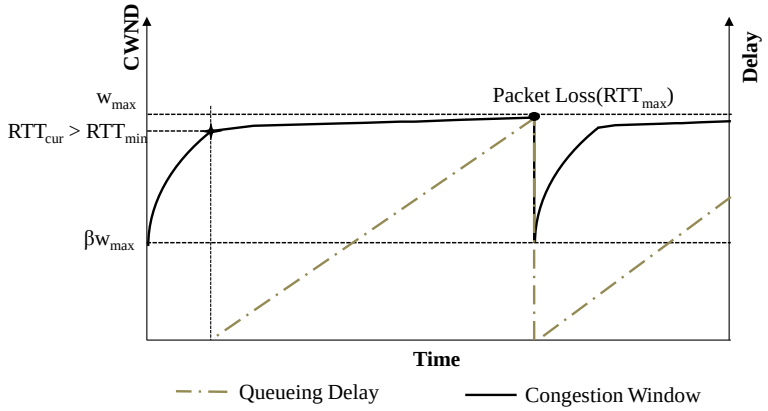


Fig. 7. Operation of MPMTP subflow when the bandwidth is unchanged

Fig. 8 shows the operation of the proposed scheme when the bandwidth increases. w'_{max} is the new w_{max}. The proposed scheme increases the congestion window in a concave manner up to w_{max}. If packet loss does not occur at w_{max}, the proposed scheme gradually increases the increasing parameter to quickly find a new w_{max}. However, at saturation, this convex growth of the congestion window overshoots the transmission rate. To prevent overshooting, the proposed scheme gradually decreases the increasing parameter according to the current RTT, after the increment in RTT is detected in the convex profile.

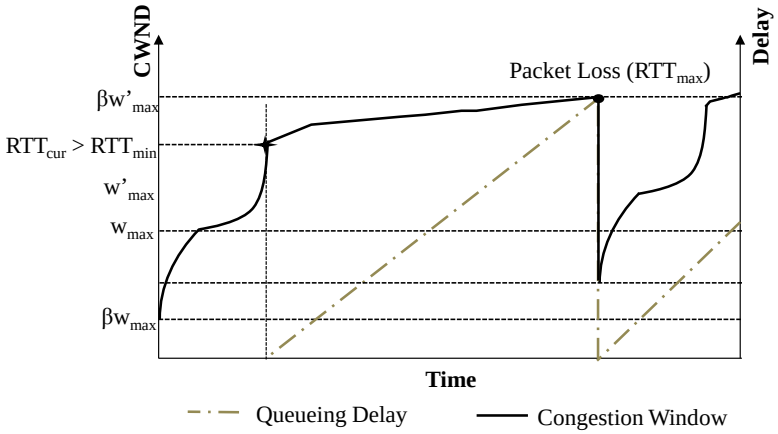


Fig. 8. Operation of MPMTP subflow when the bandwidth is increased

Fig. 9 shows the operation of the proposed scheme when the bandwidth decreases. In the concave profile, the transmission rate aggressively increases at the start of the concave profile and increases more slowly as the congestion window gets closer to w_{max}. Thus, if the bandwidth decreases, the concave increment leads to burst packet losses. The proposed scheme can prevent overshooting of the transmission rate at the concave profile if an increment in RTT is detected by turning into a linear profile. We express the increase and decrease rules of the MPMTP subflow as follows:

$$Inc.: w_r(n+1) = w_r(n) + \frac{\alpha_r}{w_r(n)}, \text{ where } 1< \alpha_r < w_r(n),$$
$$Dec.: w_r(n+1) = w_r(n) - \beta w_r(n), \text{ where } 0 < \beta < 1 \quad (11)$$

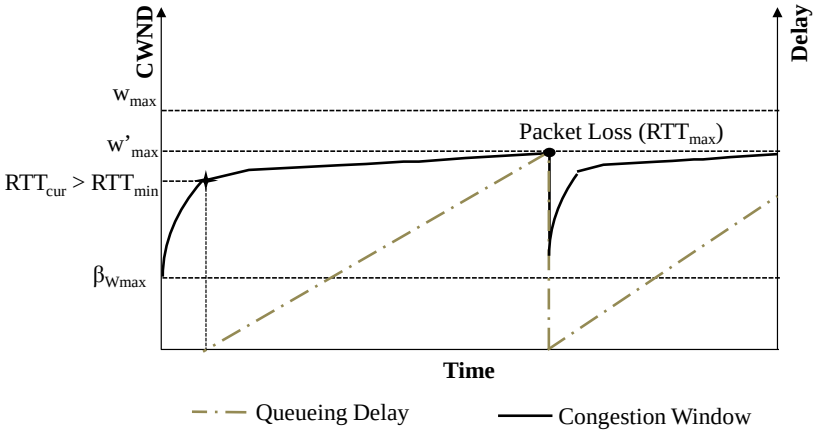
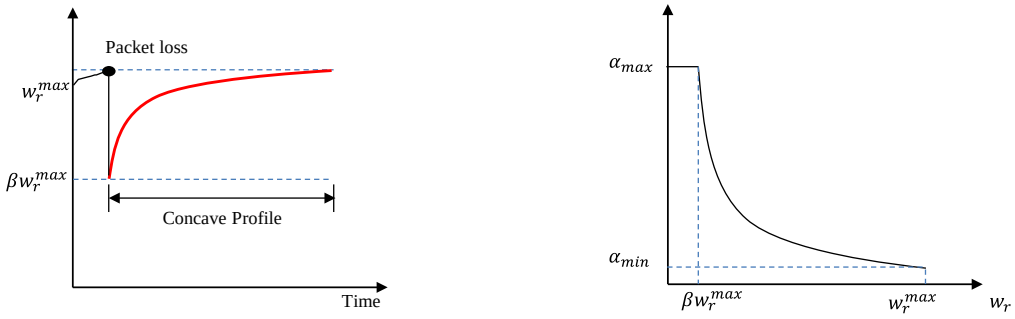


Fig. 9. Operation of MPMTP subflow when the bandwidth is increased

We adjust the increasing parameter, α_r, to increase the congestion window in a concave-convex-linear manner. **Fig. 10** shows that for the concave increase, the proposed scheme gradually decreases α_r from α_{max} to α_{min}, as the congestion window increases from βw_r^{max} to w_r^{max}. We use w_r for α_{max} for the exponential increment near βw_r^{max}, and 1 for α_{min} for the linear increment near w_r^{max}.



(a) Congestion window (b) Increasing parameter

Fig. 10. Congestion window and increasing parameter in the concave profile.

Fig. 11 shows that for concave growth, we update α_r on every ACK.

1: ACK Arrival:
2: IF $w_r < w_r^{max}$
3: THEN $\alpha_r(n+1) = L\{c(w_r(n) - w_r^{max})^2 + \alpha_{min}\}$

Fig. 11. Congestion window increment in the concave profile

L is a linear parameter. In the concave and convex profiles, we use 1 for L . c is the concave parameter, which we calculate as follows:

$$c = \frac{(w_r^{max})^2 - (\beta w_r^{max})^2}{\alpha_{max} - \alpha_{min}} \tag{12}$$

Fig. 12 shows that for the convex profile, we set the initial value of α_r to 1, and increase the congestion window on every RTT to achieve the convex window curve.

1: On Every RTT:
2: IF $w_r \geq w_r^{max}$
3: THEN $\alpha_r(n+1) = Lmax\{w_r(n) + \alpha_r(n+1)\}$

Fig. 12. Congestion window increment in the convex profile

For the linear profile, the proposed scheme adjusts the linear parameter L for a linear increase in the congestion window. If $RTT_{cur} > RTT_{min}$, the proposed scheme switches to a linear profile. **Fig. 13** shows that in the linear profile, the proposed scheme gradually decreases L as RTT_{cur} approaches RTT_{max}.

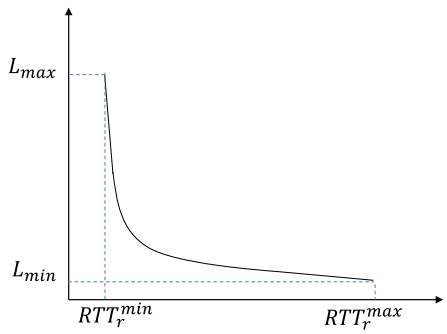


Fig. 13. Change in linear parameter according to the RTT change

Here, L_{max} is the maximum L, L_{min} is the minimum L, RTT_r^{min} is the RTT when the RTT increment is detected in subflow r, and RTT_r^{max} is the RTT when a packet loss occurs in subflow . We use 1 for L_{max} to maintain the concave and convex curves in the concave and convex profiles. Also, we use $\frac{1}{w_r}$ for L_{max} . We can calculate L as follows:

$$L = \frac{l_1}{l_2 + RTT_{cur}} \tag{13}$$

We can calculate l_1 and l_2 as follows:

$$l_1 = \frac{(RTT_r - RTT_r^{min})L_{min}\ L_{max}}{L_{max} - L_{min}} \tag{14}$$

$$l_2 = \frac{(RTT_r^{max} - RTT_r^{min})L_{min}}{L_{max} - L_{min}} \tag{15}$$

3.2 Load balancing

Fig. 14 shows that the proposed scheme moves traffic away from a congested path to an uncongested path, according to the packet loss rate of each path, in order to support load balancing.

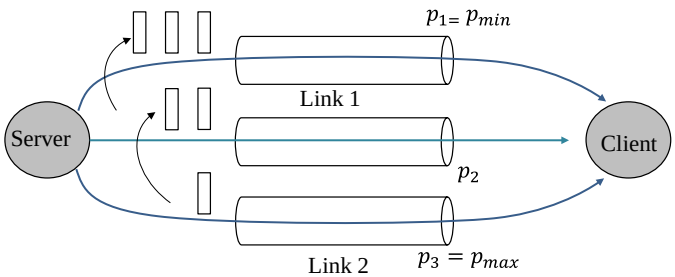


Fig. 14. Concept of the proposed load balancing mechanism

where, p_r is the packet loss rate on the path , p_{max} is the largest packet loss rate, and p_{min} is the lowest packet loss rate among all paths. We adjust the increment of the congestion window by multiplying the load balancing parameter to the increasing rule, as follows:

$$w_r(n+1) = w_r(n) + \frac{B_r(n) * \alpha_r(n)}{w_r(n)} \tag{16}$$

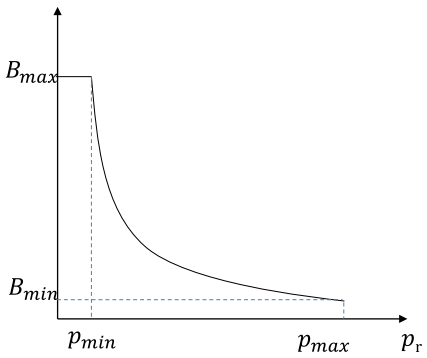


Fig. 15. Change in load balancing parameter according to the packet loss rate of each path

Fig. 15 shows that we decide B_r on the basis of the packet loss rate of the path. B_{max} is the maximum , B_{min} is the minimum B, and we calculate B_r on the basis of the packet loss rate of each path, as follows:

$$\begin{cases} B_{max}, & if\ p_r = p_{max} \\ B_r = \frac{b_1}{b_2 + p_r}, & if\ p_{min} < p_r < p_{max} \\ B_{min}, & if\ p_r = p_{min} \end{cases} \quad (17)$$

To balance the amount of increase and decrease in the congestion window, we estimate B_{max} as follows:

$$B_{max} = 1 - \frac{p_{max} - p_{min}}{p_{avg}} \quad (18)$$

and we estimate B_{min} as follows:

$$B_{min} = 1 - \frac{p_{min} - p_{avg}}{p_{avg}} \quad (19)$$

We can also use simultaneous equations to calculate b_1 and b_2, as follows:

$$b_1 = \frac{(p_{max} - p_{min}) B_{max} B_{min}}{B_{max} - B_{min}} \quad (20)$$

$$b_2 = \frac{(p_{max} - p_{min}) B_{min}}{B_{max} - B_{min}} \quad (21)$$

3.3 Fairness with single path flow

We propose a scheme to support fairness with single path CUBIC, which has been the default congestion control scheme in Linux since kernel 2.6.1. The proposed scheme provides fairness with single-path CUBIC by multiplying the fairness parameter, δ_r, by the increasing rules, as follows:

$$w_r(n+1) = w_r(n) + \frac{\delta_r(n) * B_r(n) * \alpha_r(n)}{w_r(n)} \quad (22)$$

We calculate δ_r on the basis of the equilibrium of the proposed scheme and average throughput of CUBIC.

To improve the throughput while providing fairness, design goal 1 requires that a multipath flow should give a connection at least as much throughput as it would get with the single-path

flow on the best of its paths. Design goal 2 requires that a multipath flow should take no more capacity on any path or collection of paths than if it was a single path flow using the best of those paths. This guarantees that it will not unduly harm other flows at a bottleneck link.

In mathematical notation, we can describe design goal 1 of the multipath congestion control scheme as follows:

$$\sum_r \frac{\hat{w}_r}{RTT_r} \geq max\left\{\frac{\hat{w}_r^{CUBIC}}{RTT_r}\middle| r \in R\right\} \tag{23}$$

where, $\hat{w}_r^{CUBIC}$ is the equilibrium window of CUBIC on path r. We can describe the condition to satisfy design goal 2 of the multipath congestion control scheme as follows:

$$\sum_r \frac{\hat{w}_r}{RTT_r} \leq max\left\{\frac{\hat{w}_r^{CUBIC}}{RTT_r}\middle| r \in S\right\} \tag{24}$$

Thus, we can describe the condition to satisfy both design goals 1 and 2 as follows:

$$\sum_r \frac{\hat{w}_r}{RTT_r} = max\left\{\frac{\hat{w}_r^{CUBIC}}{RTT_r}\middle| r \in R\right\} \tag{25}$$

We can calculate δ_r on the basis of Equation (26), and then balance the equations. At equilibrium, the window increases and decreases balance out on each path, as follows:

$$(1-p_r)\frac{\delta_r B_r a_r}{\hat{w}_T} = p_r \frac{\hat{w}_r}{\beta} \tag{26}$$

If p_r is small enough for $1-p_r \cong 1$. we can calculate p_r, the packet loss rate of path r, as follows:

$$p_r = \frac{\delta_r B_r a_r \beta}{\hat{w}_r^2} \tag{27}$$

Substituting Eq. (28) into the equation for the CUBIC's average congestion window [13], we obtain:

$$\overline{w}_r^{CUBIC} = \left(\frac{C(4-\beta)RTT_r^3}{4\beta\frac{\delta_r B_r a_r \beta}{\hat{w}_r^2}}\right)^{\frac{1}{4}} \tag{28}$$

Using Eqs. (26) and (29), we obtain:

$$\sum_r \frac{\hat{w}_r}{RTT_r} = \max\left\{\frac{\left(\frac{C(4-\beta)RTT_r}{4\beta\left(\frac{\delta_r B_r \alpha_r \beta}{\hat{w}_r^2}\right)^3}\right)^{\frac{1}{4}}}{RTT_r} \middle| r \in R\right\} \tag{29}$$

Solving for δ_r, we obtain:

$$\delta_r = \frac{\hat{w}_r^2 max\left\{\left(\frac{C(4-\beta)RTT_r^2}{4\beta}\right)^{\frac{1}{3}} \middle| r \in R\right\}}{\alpha_r B_r \beta\left(\sum_r \frac{\hat{w}_r}{RTT_r}\right)^{\frac{4}{3}}} \tag{30}$$

Multiplying δ_r to the increasing rule, we can provide fairness with the single path CUBIC. This formula for δ_r obviously requires measuring the round trip time.

4. Performance evaluation

In this section, we evaluate the evolution of the congestion window, load balancing, and fairness of MPMTP. To evaluate the performance of the proposed scheme, we implemented the proposed scheme in a network simulator, ns-2. In all the simulations, we set the transport protocols by the selective acknowledgment (SACK) option and a 1500-byte data packet.

Fig. 16 shows that to evaluate the evolution of the congestion window, we run two-path MPTCP, MPCUBIC, and the proposed MPMTP flow on separate paths with the same network parameters. Here, MS is a sending node for multipath congestion control schemes, MR is a receiving node for multipath congestion control schemes, SS is a sending node of a single path congestion control scheme, SR is a receiving node of a single path congestion control scheme, and R is a network router.

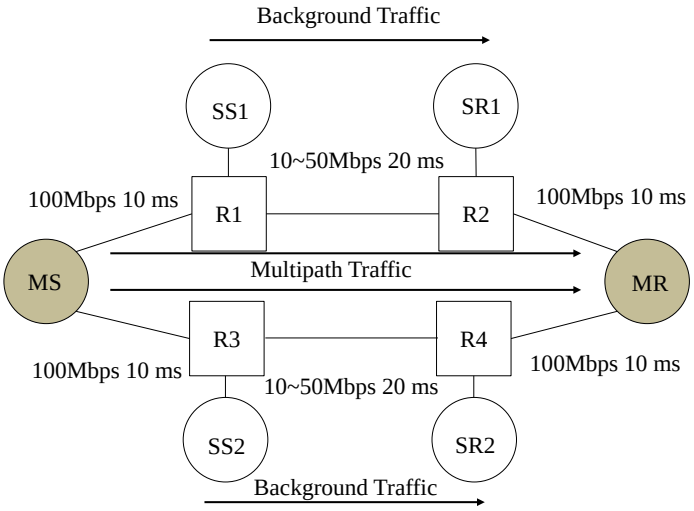
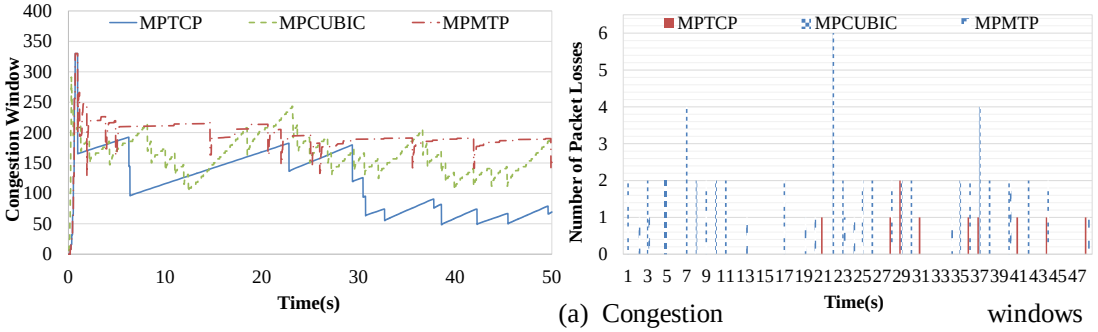


Fig. 16. Network topology to evaluate the evolution of the congestion window

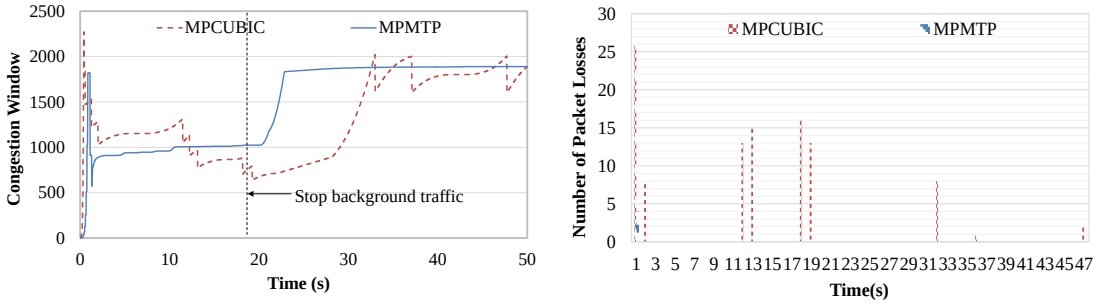
The first experiment shows the performance of the proposed scheme in a congested link. We use 10 TCP flows for background traffic on each path. The simulation results in **Fig. 17** (a) shows that the MPMTP quickly increases and smoothly changes the congestion window. **Fig. 17** (b) shows that because it prevents overshooting of the transmission rate by linearly increasing the congestion window at saturation, the MPMTP generated less packet losses than the other schemes.



(a) Congestion windows (b) Packet losses

Fig. 17. Comparison of congestion windows and packet losses in a congested link

We run the second experiment with a link capacity of 50 Mbps, and remove the background traffic at the 20s to show the increase in speed of the congestion window when the bandwidth increases. **Fig. 18** (a) shows that MPMTP more quickly increased the congestion window faster than MPCUBIC; while **Fig. 18** (b) shows that because of overshooting the transmission rate, MPCUBIC generated burst packet losses. The proposed scheme does not generate a packet loss after the 20s, because it increases the congestion window linearly at saturation.

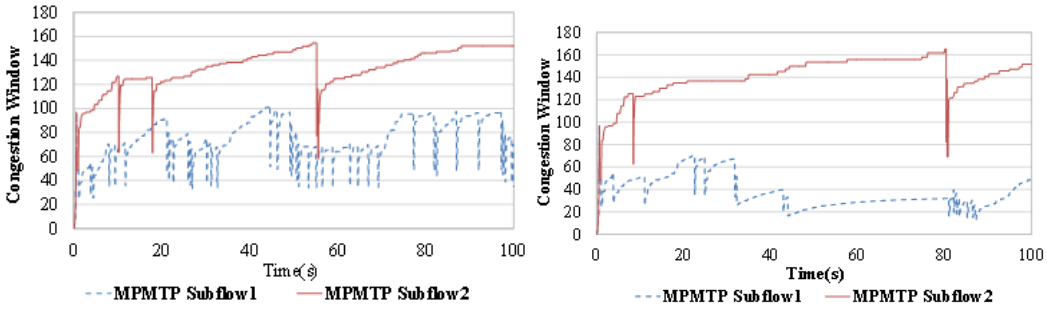


(a) Congestion windows (b) Number of packet losses

Fig. 18. Comparison of congestion windows and packet losses according to bandwidth increment

To evaluate the performance in load balancing, we run a simulation with a link capacity of 10 M. The first experiment shows the congestion window and a number of packet losses when 20 background flows compete with subflow 1 of MPMTP on one path, and five background flows compete with subflow 2 of MPMTP on the other path. **Fig. 19** (a) shows the congestion window of MPMTP without load balancing, while **Fig. 19** (b) shows the congestion window of MPMTP with load balancing. In (a), the congestion window of subflow 1 frequently

decreases at a high packet loss rate. Meanwhile, in (b), the congestion window of subflow 1 infrequently decreases, because the traffic of subflow 1 moves to subflow 2.



(a) Without load balancing (b)With load balancing

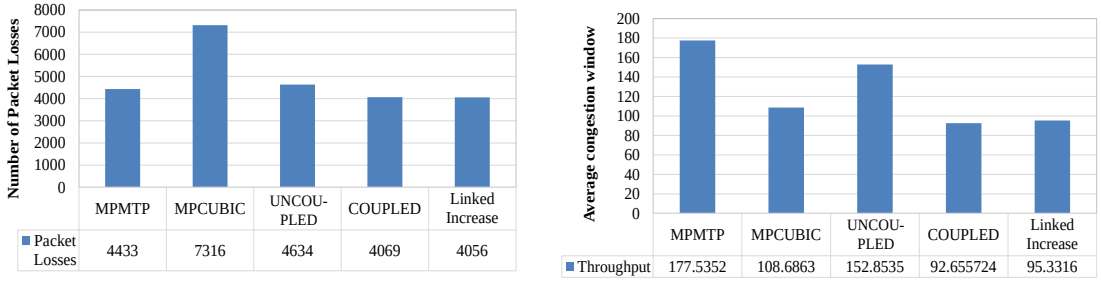
Fig. 19. Comparison of MPMTP congestion windows according to the application of load balancing.

Table 2 shows that when using load balancing, the number of packet losses in subflow 1 reduces by 30%.

Table 2. Comparison of packet loss rate.

Subflow number / Application of load balancing	MPMTP subflow1	MPMTP subflow2
Without load balancing	3.19×10^{-5}	10^{-7}
With load balancing	8.3×10^{-6}	10^{-7}

In the second experiment, we compare the number of packet losses and average throughput between the MPMTP, MPCUBIC, UNCOUPLED, COUPLED, and Linked Increase schemes. **Fig. 20** shows that the proposed scheme displayed a similar number of packet losses to the AIMD-based schemes, and the highest average throughput, because of the fast increasing behavior and load balancing.



(a) The number of packet losses (b)The average congestion windows

Fig. 20. Comparison of the number of packet losses and average congestion windows.

Fig. 21 shows that to evaluate the performance of fairness with the single-path flow, we compare the congestion window when two MPMTP subflows share a bottleneck link with a CUBIC flow.

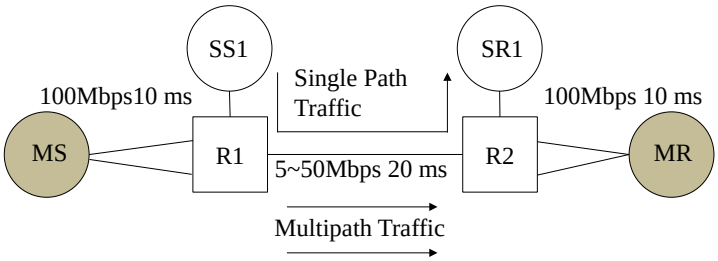
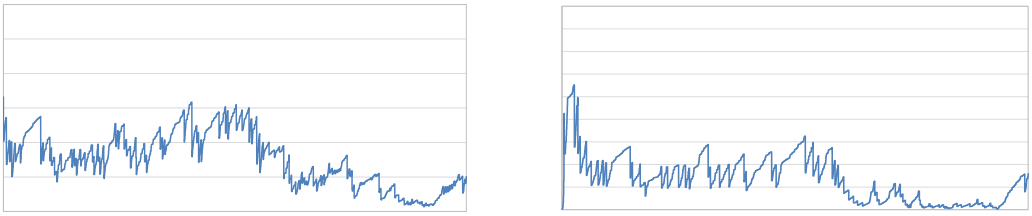


Fig. 21. Network topology to evaluate the fairness with single path flow.

We run a simulation with a link capacity of 50 M. MS transmits data with two MPTCP subflows, while SS1 transmits data with a CUBIC flow. **Fig. 22** (a) compares the congestion window between MPMTP and single-path CUBIC. We can see that even if the MPMTP uses two subflows, the total congestion windows of the proposed MPMTP and single-path CUBIC are similar. **Fig. 22** (b) compares congestion windows between MPMTP subflows. The simulation results show that the subflows of MPMTP have almost the same congestion window.



(a) MPMTP and single-path CUBIC (b) Comparison between MPMTP subflows.

Fig. 22. Evolution of the congestion window at a shared bottleneck.

5. Conclusions

In this paper, we propose a multipath multimedia transport protocol to guarantee a high quality multimedia streaming service. The proposed scheme quickly increases the congestion window, and by adjusting the increasing parameter for the congestion window based on the current congestion window and RTT, prevents burst packet losses. Also, by using load balancing and a fairness parameter based on the network status of all paths, the proposed scheme improves network efficiency and fairness with single path flows.

Our performance evaluation shows that the proposed scheme can improve the quality of multimedia streaming by increasing the transmission rate 33% faster than the MPCUBIC while reducing packet losses by 39%.

References

[1] P. Vo, T. Le, S. Lee, C. Hong, B. Kim, and H. Song, “mReno: A Practical Multipath Congestion Control for Communication Networks,” *Computing*, vol. 96, no. 3, pp. 189-205, March 2014. Article (CrossRef Link).

[2] T. Fujii, D. Shirai, Y. Tonomura, M. Kitamura, T. Nakachi, T. Sawabe, M. Ogawara, T. Yamaguchi, M. Nomura, and K. Shirakwak, "Digital Cinema and Super-High-Definition Content Distribution on Optical High-Speed Networks," in *Proc. of the IEEE*, vol. 101, no. 1, pp. 140-153, January 2013. Article (CrossRef Link).
[3] A. Ford, C. Raiciu, and M. Handley, "TCP Extensions for Multipath Operation with Multiple Addresses," *IETF Internet Draft*, October 2014. Article (CrossRef Link).
[4] Q. Zhang, W. Zhu, and Y. Zhang, "End-to-end Qos for Video Delivery over Wireless Internet," in *Proc. of the IEEE*, vol. 93, no. 1 pp. 123-134, January 2005. Article (CrossRef Link).
[5] M. Scharf, "Fast Startup Internet Congestion Control Mechanisms for Broadband Interactive Application," *Ph.D. dissertation*, *Dept. Computer Science, Electrical Engineering and Information Technology, University of Stuttgart*, Germany, April 2011. Article (CrossRef Link).
[6] C. Raiciu, M. Handly, and D. Wischik, "Coupled Congestion Control for Multi path Transport Protocols," *RFC 6356*, October 2011. Article (CrossRef Link).
[7] M. Honda, Y. Nishida, L. Eggert, P. Sarolahti, and H. Tokuda, "Multipath Congestion Control for Shared Bottleneck," in *Proc. of the International Workshop on Protocols for Future, Large-Scale and Diverse Network Transports*, May 2009. Article (CrossRef Link).
[8] C. Raiciu, M. Handley, and D. Wischik, "Practical Congestion Control for Multipath Transport Protocols," *University College London, London/United Kingdom, Tech. Rep*, 2009. Article (CrossRef Link).
[9] T. Le, C. Hong, and S. Lee, "Multipath Binomial Congestion Control Algorithms," *IEICE Transactions on Communications*, vol. E95-B, no. 6, pp. 1934-1943, June 2012. Article (CrossRef Link).
[10] T. Le, C. Hong, and S. Lee, "MPCubic: An Extended Cubic TCP for Multiple Paths over High Bandwidth-Delay Networks," in *Proc. of the International Conference on ICT Convergence*, pp. 34-39, September 2011. Article (CrossRef Link).
[11] S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 5, pp. 64-74, July 2008. Article (CrossRef Link).
[12] T. Kun, S. Jingmin, Z. Qian, and S. Murad, "A Compound TCP Approach for High-speed and Long Distance Networks," in *Proc. of the IEEE INFOCOM*, pp. 1-12, April 2006. Article (CrossRef Link).
[13] J. Wang, J. Wen, and Y. Han, "CUBIC-FIT: A High Performance and TCP CUBIC Friendly Congestion Control Algorithm," *IEEE Communications Letters*, vol. 17, no. 8, pp. 1664-1667, June 2013. Article (CrossRef Link).
[14] J. Wang, J. Wen, J. Zhang, Z. Xiong, and Y. Han, "TCP-FIT: An Improved TCP Algorithm for Heterogeneous Networks," *Journal of Network and Computer Applications*, vol. 71, pp. 167-180, August 2016. Article (CrossRef Link).

Sunghee Lee received his B.S. and Ph. D. degree in Communications Engineering from Kwangwoon University, Seoul, Korea, in 2009 and 2015. He joined the ETRI(Electronics and Telecommunications Research Institute), where he is currently a senior researcher of Broadcasting & Telecommunications Media Research Laboratory. His research interests include QoS, QoE support and rate control for video communications over wired and wireless networks.

Kwangsue Chung (M'93, SM'01) received the B.S. degree from Hanyang University, Seoul, Korea, M.S. degree from KAIST(Korea Advanced Institute of Science and Technology), Seoul, Korea, Ph.D. degree from University of Florida, Gainesville, Florida, USA, all from Electrical Engineering Department. Before joining the Kwangwoon University in 1993, he spent 10 years with ETRI(Electronics and Telecommunications Research Institute) as a research staff. He was also an adjunct professor of KAIST from 1991 to 1992 and a visiting scholar at the University of California, Irvine from 2003 to 2004. His research interests include communication protocols and networks, QoS mechanism, and video streaming.