IEICE TRANS. INF. & SYST., VOL.E100–D, NO.12 DECEMBER 2017

PAPER *Special Section on Parallel and Distributed Computing and Networking*

TCP-TFEC: TCP Congestion Control based on Redundancy Setting Method for FEC over Wireless LAN***

Fumiya TESHIMA†*, ***Nonmember***, **Hiroyasu OBATA**†a), **Ryo HAMAMOTO**†**, ***and*** **Kenji ISHIDA**†b), ***Members***

SUMMARY Streaming services that use TCP have increased; however, throughput is unstable due to congestion control caused by packet loss when TCP is used. Thus, TCP control to secure a required transmission rate for streaming communication using Forward Error Correction (FEC) technology (TCP-AFEC) has been proposed. TCP-AFEC can control the appropriate transmission rate according to network conditions using a combination of TCP congestion control and FEC. However, TCP-AFEC was not developed for wireless Local Area Network (LAN) environments; thus, it requires a certain time to set the appropriate redundancy and cannot obtain the required throughput. In this paper, we demonstrate the drawbacks of TCP-AFEC in wireless LAN environments. Then, we propose a redundancy setting method that can secure the required throughput for FEC, i.e., *TCP-TFEC*. Finally, we show that TCP-TFEC can secure more stable throughput than TCP-AFEC.

key words: *wireless LAN, Forward Error Correction, TCP congestion control, Throughput, Streaming*

1. Introduction

Video streaming communication services, such as BIGLOBE [2] and YouTube [3], are often used in wireless Local Area Network (LAN) environments based on the IEEE 802.11 [4] standard. Generally, these services often use UDP [5] as the transport layer protocol for real-time communication. However, UDP may be rejected by some network devices, such as firewalls and broadband routers. To address this issue, streaming communication over TCP [6] has increased. Various studies [7]–[9] have evaluated the effectiveness of control methods based on TCP. However, with TCP, throughput can become unstable due to congestion control caused by packet loss. It degrades QoS and QoE [10], e.g., the video stops temporarily when TCP cannot secure a sufficient transmission rate. This occurs because the TCP congestion control method changes the transmission rate relative to congestion.

Thus, TCP control methods to guarantee an application-specific transmission rate have been proposed [11], [12]. However, these TCP control methods do not work well when the packet loss rate increases. To overcome this problem, TCP control to secure a required transmission rate using a forward error correction (FEC) [13] algorithm is an effective solution. TCP-AFEC [14] can control the transmission rate appropriately according to network conditions using a combination of TCP congestion control and FEC. However, because TCP-AFEC was not developed for wireless LANs, it requires a certain time to set the appropriate redundancy and cannot obtain the required throughput.

In this paper, first, we demonstrate the disadvantages of TCP-AFEC in wireless LAN environments through simulation. Second, we propose a redundancy setting method for a wireless LAN environment. We refer to this method as *TCP-TFEC*. Finally, we show the effectiveness of TCP-TFEC through simulation. The contribution of this paper shows as follows:

- TCP-TFEC can improve throughput by approximately 1.2 Mbps (95th percentile) compared to TCP-AFEC when the packet loss rate is 20% in IEEE802.11g (54Mbps) environment.
- TCP-TFEC can guarantee throughput when the packet loss rate changes dynamically.
- The standard deviation of delay jitter for TCP-TFEC satisfies the criterion of QoS Class0 in ITU-T Y.1541 (50 ms).

The remainder of this paper is organized as follows. In Sect. 2, we provide an overview of carrier sense multiple access with collision avoidance (CSMA/CA), which is IEEE 802.11 media access control for wireless LANs and explain related works. Section 3 describes TCP-AFEC. Then, Sect. 4 evaluates the characteristics of TCP-AFEC over a wireless LAN. Section 5 describes and evaluates the proposed method. Finally, Sect. 6 presents conclusions and suggestions for future work.

2. Preliminary

2.1 CSMA/CA

In IEEE802.11 wireless LANs, each wireless terminal uses CSMA/CA as the media access control. In CSMA/CA, if the channel becomes idle when a data frame arrives in the transmission queue, it defers to DCF interframe space (DIFS) time. Subsequently, if the channel remains idle after DIFS, CSMA/CA waits for the backoff time, which is calculated

Manuscript received January 6, 2017.
Manuscript revised May 26, 2017.
Manuscript publicized July 14, 2017.

†The authors are with the Graduate School of Information Sciences, Hiroshima City University, Hiroshima-shi, 731–3194 Japan.
*Presently, with working with NEC Corporation.
**Presently, with working with Global Engineering Center Kobelco Construction Machinery Corporation Limited.

***An earlier version of this paper was presented to ASON 2015 [1].

a) E-mail: obata@hiroshima-cu.ac.jp
b) E-mail: ishida@hiroshima-cu.ac.jp

DOI: 10.1587/transinf.2017PAP0017

randomly using a contention window (CW). If the channel remains idle after the backoff time, the terminal sends the data frame. The backoff time (Backoff) is calculated independently by each terminal using Eq. (1).

$$\text{Backoff} = \mathit{Random}() \times \mathit{SlotTime} \tag{1}$$

In Eq. (1), $\mathit{Random}()$ and $\mathit{SlotTime}$ are random integers derived from a discrete uniform distribution [0, CW] and the slot time interval specified in IEEE802.11, respectively. At this point, the initial CW is set to CW_{min}. If a collision or bit error by interference causes the data frame transmission to fail, then the terminal resets the backoff time using Eq. (1). In this case, the CW becomes twice the previous value and the upper bound is CW_{max}. If retransmission exceeds the maximum retry limit (usually 7), the terminal discards the data frame.

Here, in wireless LAN environment, it is known that the TCP throughput decreases because of the packet loss by the collision or bit error derived from interference of other wireless LAN systems. It is because that the TCP congestion window size decreases by the packet loss. As mentioned above, in CSMA/CA, the backoff time (i.e. waiting time for data transmission) becomes larger when the terminal fails to send data frame. Then, since the round trip time also increases by the increase of backoff time when collision or bit error occur, the waiting time of TCP-ACK at the sender also increases. As a result, the TCP throughput decreases in this situation because it cannot increase the congestion window size quickly when the backoff time of CSMA/CA increases. Furthermore, the backoff time derived from CW changes drastically when the transmission error by collision/bit error occurs (it becomes twice the previous value) or the data transmission after transmission error succeeds (it returns to the initial value). In this situation, since TCP cannot send data with constant rate, the TCP throughput becomes unstable when the packet loss often occurs.

2.2 Related Works

This section explains the typical existing TCP congestion controls for wireless LAN.

First, there are some studies [15]–[17] for obtaining higher TCP performance on wireless LAN. In wireless environment, it is known that the TCP congestion window size decreases unnecessarily when the packet loss by bit error occurs. As a result, the TCP throughput decreases over wireless LAN environment. Thus, as one of the solutions for keeping the higher TCP congestion window size, the method [15] uses a TCP proxy mechanism at the AP. Since the method [15] sends a pseudo TCP-ACK for the sender and decreases the number of TCP-ACK, TCP sender can keep the TCP congestion window size high even if the packet loss occurs. However, the method [15] cannot be deployed when it is difficult to modify the access point (AP). Furthermore, the method [15] does not have efficient dynamic parameter setting method for TCP proxy parameters when the network condition changes. On the other hand, since the proposed method can keep the TCP congestion window size high without modifying AP and dynamically changes the control parameters of FEC, it has higher applicability than the method [15].

Next, in [16], the authors have proposed a TCP window control based on the channel occupancy time. In order to maintain the quality of video streaming, this method controls the TCP congestion window size by monitoring the congestion in MAC layer (i.e. this method is a cross layer mechanism). However, the method [16] does not have efficient loss recovery mechanism. Thus, the TCP throughput decreases when the packet loss occurs by bit error or collision. Furthermore, because it uses RTS/CTS for estimating the channel occupancy time, the total throughput decreases by the overhead of RTS/CTS. On the other hand, our proposed method can obtain higher throughput by using FEC mechanism without the control overhead for estimating network condition. In addition, the proposed method can improve the throughput without assist of MAC layer. Next, as the TCP layer solution, Compound TCP+ [17] has been proposed by modifying delay based control of Compound TCP for wireless LAN. However, since Compound TCP+ has been developed for improving the fairness among Compound TCP connections, it cannot obtain higher throughput than our proposed method when the loss rate increases.

Second, in order to improve the performance of TCP streaming services, the several TCP congestion controls [11], [12], [14], [18]–[20] have been proposed. To begin with, TCP-Hollywood [18] has been developed for applications with tight latency bounds. In order to achieve the objective, TCP-Hollywood removes head-of-line blocking at the receiver and delivers received data to the application immediately irrespective of ordering. However, because the congestion window control of TCP-Hollywood was not developed for wireless environment, it cannot keep the higher TCP congestion window size to obtain the stable throughput when the packet loss occurs frequently. Next, TCP-AV [11] and TCP-gMB [12] try to keep higher TCP congestion window size when the packet loss occurs in order to obtain the stable throughput. These methods set the TCP congestion window size and slow start threshold based on the target throughput. However, when the round trip time changes drastically (this situation sometimes occur in the wireless LAN), these methods cannot guarantee the target throughput. Thus, in order to overcome the problem, TCP-AV for wireless LAN [19], [20] has been proposed. TCP-AV for wireless LAN uses the smoothed round trip time and tuned control parameters to secure the target throughput on the wireless LAN. Note that we compare the performance of the proposed method with TCP-AV for wireless LAN because it is expected that TCP-AV for wireless LAN can obtain higher performance than other TCP variants. Finally, these TCP variants for TCP steaming services cannot obtain the stable throughput as the packet loss rate increases. It is because that these methods require the certain time to retransmit the lost data and stops sending new data even if they try to keep the TCP congestion window size high. To overcome this problem, we assume that the forward error correction (FEC) [13] algorithm is the effective solution. Thus, TCP-AFEC [14] is one of TCP congestion control methods which uses FEC algorithm. However, since TCP-AFEC was not developed for the wireless LAN, we propose new TCP method based on TCP-AFEC. In the following section, this paper shows the detail of TCP-AFEC and its problem in the wireless LAN environment. Then, we explain the proposed method in detail.

3. TCP-AFEC

We provide an overview of TCP-AFEC, which is a QoS-TCP to guarantee throughput.

3.1 FEC in TCP-AFEC

To secure a transmission rate, TCP-AFEC uses the FEC mechanism at the bottom of the transport layer (i.e., TCP). In the FEC layer, the sender adds redundancy data behind a data packet to recover the packet at the receiver if packet loss occurs. FEC uses the *Reed-Solomon code* because it is strong against burst error. The Reed-Solomon code is one of block code [21]. In the Reed-Solomon code, information is divided into consecutive bits of M information (referred to as a symbol), and the information is encoded and decoded using these symbols. Here, one block comprises N ($N = 2^M - 1$) symbols. In one block, K pieces and $(N - K)$ are data symbols and redundancy symbols, respectively. The ratio between the number of data symbols and total symbols (K/N) is called the code rate or information rate. Thus, although the size of the redundancy data decreases as the code rate reduces, error correction capability decreases. Here, FEC adds redundant data to data segments using $(n + k, k)$ code (K-th redundancy level). For example, if FEC can recover 1 in 20 packets, the redundancy is $(20 + 1)/20 = 1.05$.

3.2 Overview of TCP-AFEC

TCP-AFEC has a redundancy state transition structure (Fig. 1). In Fig. 1, each level is the redundancy level. TCP-AFEC has a Drop Timer (DT) to change the redundancy level dynamically. Note that redundancy decreases if DT timeouts occur. In addition, redundancy increases when packet loss occurs before DT timeout occurs. The DT timeout value increases by α times ($1 < \alpha$) if packet loss or DT timeout occurs. Furthermore, redundancy is reduced by β times ($0 < \beta < 1$) at regular time intervals. Therefore, redundancy increases when α increases and decreases as β decreases. In the state transition structure, when packet loss occurs in the K-th level, the current state changes to the $(K + 1)$-th level. Simultaneously, DT_{K+1} (the DT of the $(K + 1)$-th level) is increased α times. Next, when a timeout occurs in the $(K+1)$-th level, redundancy might become too large. Therefore, the current state moves to the K-th level and the value of DT_K is increased by α times. If packet loss does not occur during RTT and DT timeout does not occur (i.e., timer T_p expires), the DT value of all states, except the current state, decreases β times.

Next, TCP-AFEC calculates the window size to satisfy the transmission rate R bps required by an application. The window size W_{app} is calculated by Eq. (2) using R and the average RTT $\overline{T_{rtt}}$.

$$W_{app} = R \times \overline{T_{rtt}} \tag{2}$$

TCP-AFEC has a threshold $R_{th}(t)$, which is used to determine whether redundancy increases. $R_{th}(t)$ is updated when packet loss occurs or when the retransmission timer times out. If packet loss occurs or the retransmission timer times out at time t, $R_{th}(t)$ is calculated by Eq. (3).

$$R_{th}(t) = W_{app} + \max(W_{app} - W(t), 0) \tag{3}$$

In Eq. (3), $\max(a, b)$ gives the maximum value between a and b. Moreover, the range of $R_{th}(t)$ is $W_{app} \leq R_{th}(t) \leq 2W_{app} - 1$. Therefore, if the current window size is smaller than $R_{th}(t)$, TCP-AFEC does not decrease redundancy even if a DT timeout occurs.

4. Characteristics of TCP-AFEC in a Wireless LAN

In this section, we evaluate the throughput characteristics of TCP-AFEC in a wireless LAN environment using NS2 [22]. Figure 2 shows the network model used in this evaluation. We used IEEE802.11g [23] for the wireless LAN environment. In this environment, one terminal uses TCP-AFEC and the other terminals (background flows) employ Reno. Note that all terminals generate FTP traffic. In this evaluation, the TCP-AFEC parameters are set as follows. The

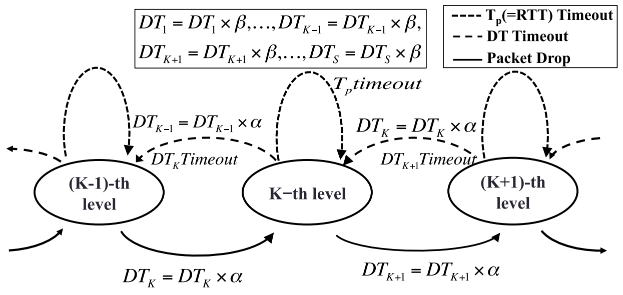
$DT_1 = DT_1 \times \beta, \ldots, DT_{K-1} = DT_{K-1} \times \beta,$
$DT_{K+1} = DT_{K+1} \times \beta, \ldots, DT_S = DT_S \times \beta$
---- T_p(=RTT) Timeout
- - - DT Timeout
—— Packet Drop
$T_p$timeout
$DT_{K-1} = DT_{K-1} \times \alpha$
$DT_K = DT_K \times \alpha$
DT_KTimeout
DT_{K+1}Timeout
(K-1)-th level
K−th level
(K+1)-th level
$DT_K = DT_K \times \alpha$
$DT_{K+1} = DT_{K+1} \times \alpha$

Fig. 1 State transition structure of TCP-AFEC [14]

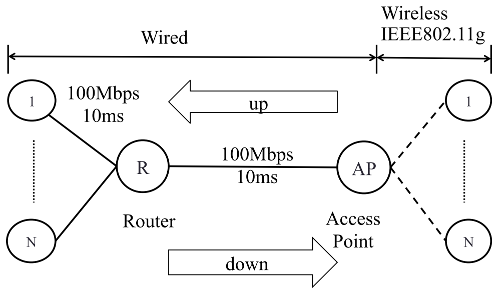

Fig. 2 Network model

Table 1 Simulation parameters

AP Buffer size	500 [packet]
Segment size	1 [Kbyte]
Packet loss rate	0.1 - 20 [%]
TCP version of background flow	Reno
Number of AFEC flows	1
Number of background flows	5 - 20
Simulation time	180 [s]
Number of trials	30

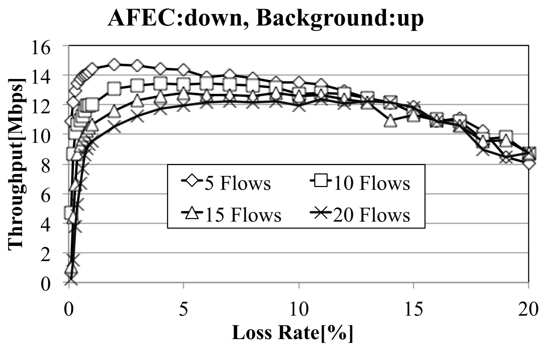

Fig. 3 Relationship between the number of background flows and the average throughput of TCP-AFEC (TCP-AFEC flow is downlink and background flows are uplink)

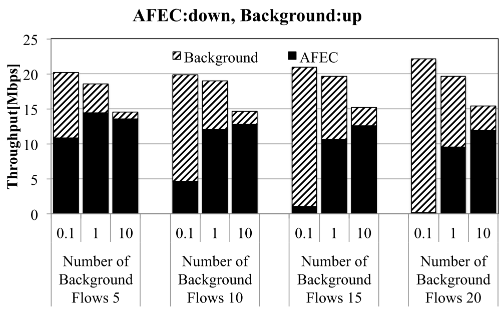

Fig. 4 Relationship between the number of background flows and the total throughput (TCP-AFEC flow is downlink and background flows are uplink)

number of states (redundancy levels) S, α, and β are 20, 2.0, and 0.8, respectively. Because the previous study [14] used these values for evaluation, this study uses same values for fair evaluation. The initial redundancy value (minimum redundancy) is 1.05 [14]. The simulation parameters are listed in Table 1. We evaluate the average throughput, total throughput of all senders, and the time change of redundancy. In addition, we assume a case wherein uplink (from access point (AP) to router) and downlink (from router to AP) flows coexist and the number of background flow changes.

Figure 3 shows the relationship between the average throughput of TCP-AFEC and the number of background flows. Moreover, Fig. 4 indicates the relationship between loss rate and the total throughput of all flows. As seen in Fig. 3, TCP-AFEC can obtain stable throughput even if the loss rate becomes greater to some extent (from 0.1% to 5%).

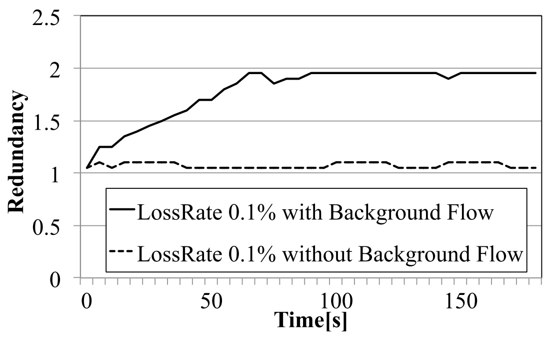

Fig. 5 Time change of redundancy in TCP-AFEC

When the loss rate increases, we confirmed the throughput of the background flows decreases by approximately 50% to 75%. As a result, the throughput of TCP-AFEC increases. However, the loss rate becomes too large (approximately greater than 10%) and TCP-AFEC cannot handle the high packet losses sufficiently. Thus, the average throughput of TCP-AFEC decreases as the loss rate increases. Moreover, with TCP-AFEC in Fig. 4, the throughput of 10% is greater than that of 0.1%. In addition, the total throughput does not change even if the number of flows changes; thus, the bandwidth is used efficiently. However, when the loss rate is small (0.1%) and the number of background flows is large (from 10 to 20), the throughput of TCP-AFEC decreases because the AP's buffer is occupied by the ACK segments of the background flows. Thus, TCP-AFEC data packet losses occur by buffer overflow in the AP.

Here, Fig. 5 shows the time change of the TCP-AFEC redundancy when the loss rate is 0.1%. In Fig. 5, the horizontal and vertical axes are time and redundancy, respectively. Figure 5 shows the results for two cases, i.e., background flows exist (the number of background flows is 20) where the TCP-AFEC flow is downlink and background flows are uplink (solid line), and only TCP-AFEC exists (dotted line). As seen in Fig. 5, if there are no background flows, redundancy is nearly stable. Conversely, redundancy increases when background flows exist. Subsequently, the redundancy becomes stable because the number of redundancy states S is not an appropriate value in this environment. In other words, TCP-AFEC must have a high S to handle the high packet loss rate. In addition, it takes approximately 60 s to increase the upper limit. The loss rate changes drastically in the wireless LAN environment. Therefore, it is necessary to set the appropriate redundancy quickly to recover the packet losses. Thus, we found that there is room to improve the redundancy setting method in TCP-AFEC.

5. Appropriate Redundancy Setting Method for a Wireless LAN

In this section, we propose and evaluate a new redundancy setting method to handle wireless LAN characteristics.

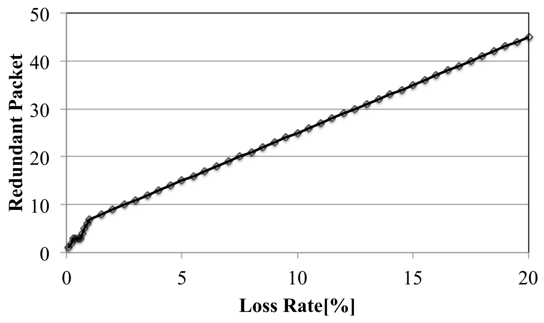

Fig. 6 Redundancy for each packet loss rate when the sender can obtain maximum throughput

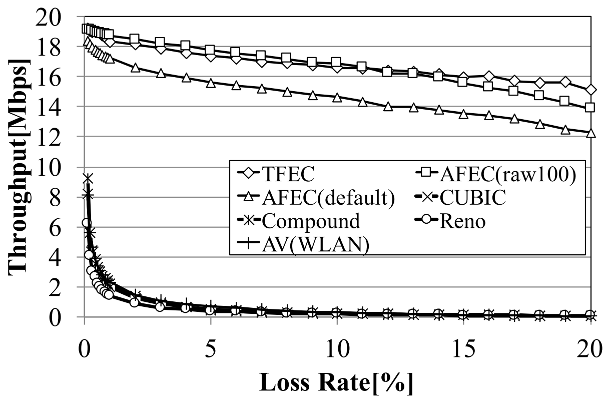

Fig. 7 Average throughput for each random loss rate

5.1 TCP-TFEC: Proposed Method

As mentioned in the previous section, because TCP-AFEC sets redundancy based on the state transition, it is necessary to wait for the state that the redundancy converges the appropriate value. Thus, the proposed method sets the appropriate redundancy directly using the observed packet loss rate. In addition, the proposed method sets the redundancy to obtain maximum throughput. Therefore, if it must secure a specified throughput, the proposed method must set the receiver window size to equal the target throughput.

To determine the appropriate redundancy for the wireless LAN, we evaluated the redundancy that can obtain maximum throughput for each loss rate over the network shown in Fig. 2. Figure 6 shows the redundancy for each packet loss rate when the sender can obtain maximum throughput. In Fig. 6, K of FEC is 100. Here redundancy does not change after data transmission begins. TCP-TFEC sets the appropriate redundancy based on the relationship between redundancy and packet loss rate shown in Fig. 6. Here, in the wireless LAN environment, the packet loss rate changes drastically. Therefore, if the packet loss rate increases drastically, throughput could decrease due to lower redundancy. Thus, to consider the operation margin, we set the appropriate redundancy (*red*) to *redundancy*+1 (Fig. 6) for each loss rate. We refer to TCP with the proposed redundancy setting method as *TCP-TFEC*. Here, we assume the sender can obtain packet loss rate information using a previously reported method [24]. Next, we discuss how to obtain the packet loss rate information.

In TCP-TFEC, the sender calculates the packet loss rate ($L(t)$) in each Δt and uses the number of receipt segments by the receiver which are notified by using an expansion ACK segment (Eq. (4)):

$$L(t) = 1 - \frac{Rp(t) - Rp(t - \Delta t)}{Sp(t) - Sp(t - \Delta t)}. \tag{4}$$

In Eq. (4), $Rp(t)$ denotes the number of segments received by the receiver and $Sp(t)$ describes the number of segments sent by the sender, respectively. In addition, Δt is set to 5 s in this paper. Note that $L(t)$ is smoothed by Eq. (5). Therefore, the notified loss rate is smoothed as follows:

$$SL(t) = \gamma SL(t - \Delta t) + (1 - \gamma)L(t), \tag{5}$$

where γ ($0 \leq \gamma \leq 1$) is a smoothing coefficient. In this paper, γ equals 0.8. It is expected the optimum γ depends on the network environment. In future, we will propose a derivation method for γ considering the network environment. The sender calculates redundancy based on the loss rate in the expansion ACK segment when the ACK segment is received. As a result, the sender can set the optimum redundancy quickly for each loss rate. Here, the accuracy of the loss rate depends on Δt. Therefore, an optimal method to set Δt will be discussed in the future.

TCP-TFEC sets the redundancy to obtain the maximum throughput. When TCP-TFEC guarantees a specific throughput, the receiver sets the received window size (*window*) using Eq. (6):

$$window = target\_bw \times rtt \times red, \tag{6}$$

where *target_bw* and *rtt* denote the target throughput (target rate) and end-to-end round trip time, respectively. Moreover, *red* denotes redundancy obtained from Fig. 6. If the redundancy segments increase, the sender's throughput decreases. Thus, we coordinate the received window size using *red*.

5.2 Evaluation of TCP-TFEC

This section evaluates the basic performance of TCP-TFEC. In this evaluation, there is no background flow, and the other simulation parameters are the same as discussed in Sect. 4. Figure 7 shows the average throughput (95th percentile) for each loss rate when the loss rate changes from 0.1% to 20%. In addition, we compare the proposed method to TCP-AFEC, CUBIC-TCP, Compound-TCP, Standard TCP (TCP-Reno), and TCP-AV(WLAN) which is tuned for wireless LAN. In Fig. 7, AFEC(default) means $K = 20$ with 20 redundancy states. Furthermore, AFEC(raw100) means $K = 100$ with 100 redundancy states. In addition, we set the target rate of TCP-TFEC, TCP-AFEC, and TCP-AV(WLAN) to the available bandwidth of the environment. As seen in Fig. 7, when the loss rate is less than 11%, AFEC(raw100) obtains the highest throughput. However, when the loss rate is greater than 11%, TCP-TFEC obtains the highest throughput. For example, the throughput of TCP-TFEC is approximately 1.2 Mbps greater than that of TCP-AFEC(raw100) when the loss rate is 20%. In addition, the throughputs of CUBIC-TCP, Compound-TCP, TCP-Reno, and TCP-AV(WLAN) are less than 1 Mbps when the loss rate is greater than 3%. This happens because these TCP methods decrease the TCP congestion window size drastically when packet loss occurs. Although TCP-AV(WLAN) tries to keep higher TCP congestion window size, the throughput of TCP-AV(WLAN) decreases. That is, even if the terminal uses TCP-AV(WLAN), because it takes long time to retransmit the lost data, the continuous data transmission stops. As a result, the throughput of TCP-AV(WLAN) decreases drastically. As seen in Fig. 7, we found that AFEC(raw100) obtains higher throughput than AFEC(default) in this environment. Therefore, we use AFEC(raw100) in the following comparative evaluation.

Figure 8, Fig. 9, and Fig. 10 show the time change of the throughput when the target rate is 1 Mbps, 5 Mbps, and 10 Mbps, respectively. In these evaluations, we set the receiver window size of TCP-TFEC to be equal to the value calculated by the target rate. Note that the loss rate is 10% in these figures. From Figs. 8, 9, and 10, if the target rate increases, the TCP-AFEC throughput becomes unstable. Conversely, TCP-TFEC obtains stable throughput and can maintain the target rate. Moreover, the convergence time of TCP-TFEC is nearly the same as TCP-AFEC. Next, we show the same results when the loss rate increases. Figure 11, Fig. 12, and Fig. 13 show the time change of throughput when the loss rate is 20%. Figures 11, 12, and 13 show the results when the target rate is 1 Mbps, 5 Mbps, and 10 Mbps, respectively. From Fig. 11, both TCP-AFEC and TCP-TFEC can obtain the same target rate as shown in Fig. 8 even if the loss rate increases. However, from Figs. 12 and 13, TCP-AFEC cannot achieve the target rate. Conversely, TCP-TFEC can obtain stable throughput if the loss rate increases. Moreover, from Fig. 13, the convergence time of TCP-TFEC (the time when the throughput reaches 10Mbps) is approximately twice that of TCP-AFEC. TCP-AFEC increases the redundancy level sequentially when packet loss occurs. Therefore, TCP-AFEC requires a certain time to set the appropriate redundancy. Furthermore, if there is no packet loss before DT timeout, TCP-AFEC reduces redundancy. As a result, TCP-AFEC cannot recover the lost packet; thus, it cannot maintain the target rate. Here, Fig. 14 and Fig. 15 show the time change of redundancy and congestion window size when the loss rate is 20%. As seen in Fig. 14, TCP-TFEC sets higher redundancy than TCP-AFEC and does not change the redundancy after 6 s. Conversely, TCP-AFEC changes the redundancy drastically. Moreover, from Fig. 15, TCP-TFEC can keep the congestion window size high because it can recover the lost packet using the appropriate redundancy. However, TCP-AFEC cannot recover the lost packet due to frequent redundancy changes; thus, the congestion window size changes drastically. As a result, TCP-AFEC cannot obtain stable throughput.

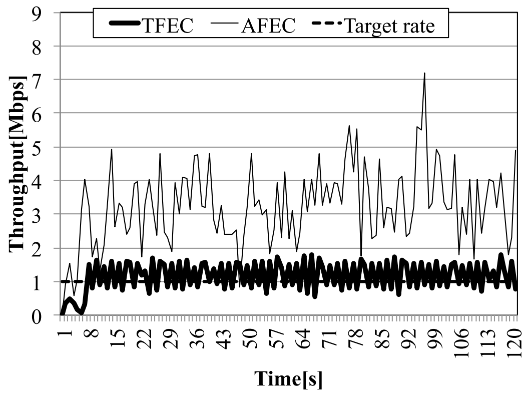

Fig. 8 Time change of throughput (loss rate: 10%, target rate: 1 Mbps)

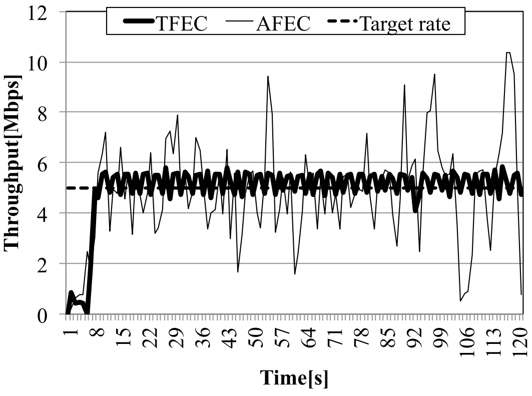

Fig. 9 Time change of throughput (loss rate: 10%, target rate: 5 Mbps)

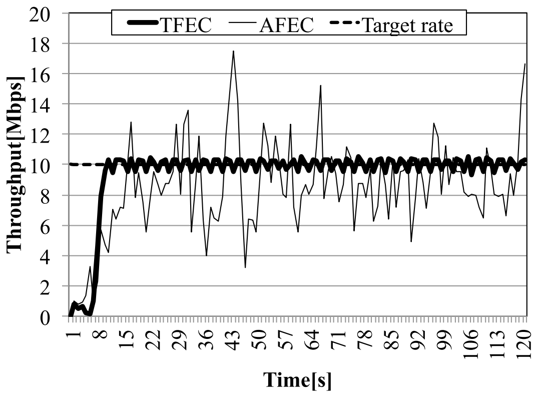

Fig. 10 Time change of throughput (loss rate: 10%, target rate: 10 Mbps)

Next, we evaluate throughput when the packet loss rate changes. In this evaluation, we change the loss rate between 0.1% and 10% every 10 s and 20 s. The initial loss rate is 0.1% when the simulation begins. The number of TCP-TFEC and TCP-AFEC flows is 1, and there are no background flows. In addition, the target rate is 10 Mbps. Figure 16 and Fig. 17 show the time change of throughput when the loss rate changes every 10 s and 20 s, respectively. As

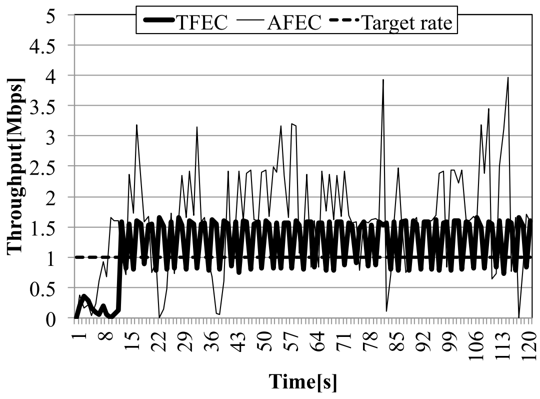

Fig. 11 Time change of throughput (loss rate: 20%, target rate: 1 Mbps)

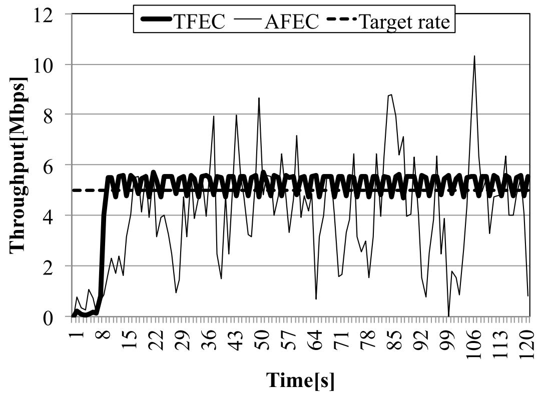

Fig. 12 Time change of throughput (loss rate: 20%, target rate: 5 Mbps)

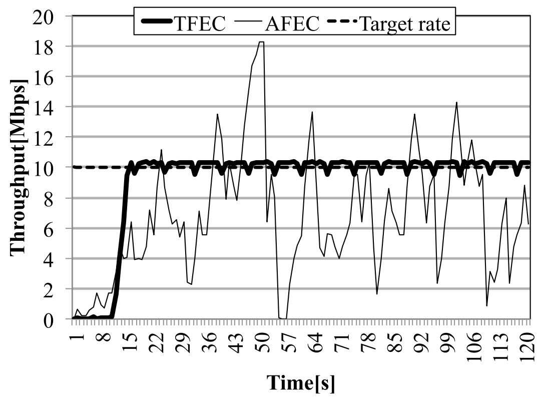

Fig. 13 Time change of throughput (loss rate: 20%, target rate: 10 Mbps)

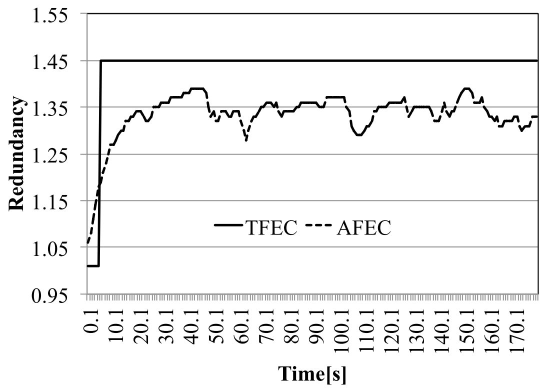

Fig. 14 Time change of redundancy (loss rate: 20%, target rate: 10 Mbps)

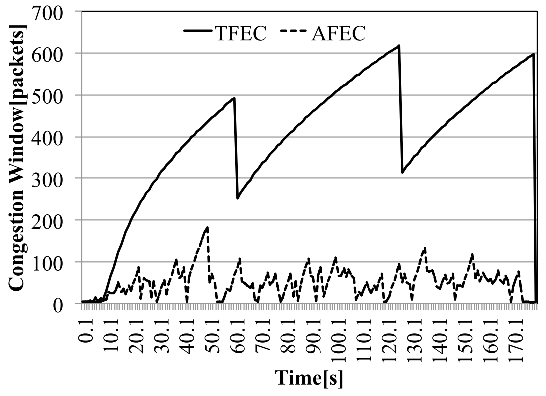

Fig. 15 Time change of congestion window size (loss rate: 20%, target rate: 10 Mbps)

can be seen in Figs. 16 and 17, TCP-AFEC can maintain throughput at the beginning of the simulation. Conversely, when the loss rate changes from 0.1% to 10%, TCP-AFEC cannot maintain the target rate. Here, the redundancy is reduced according to the low loss rate during 0.1%. However, when the loss rate becomes 10%, the congestion window size decreases drastically because TCP-AFEC cannot immediately set the appropriate redundancy. As a result, TCP-AFEC cannot obtain sufficient throughput when the loss rate changes dramatically. Furthermore, if the loss rate duration time increases, it takes a long time to recover throughput when the loss rate changes. Conversely, TCP-TFEC also decreases throughput temporarily when the loss rate changes. However, TCP-TFEC can change the redundancy quickly depending on the loss rate; thus, TCP-TFEC can obtain the target rate because the congestion window size does not decrease due to packet loss.

Here, we evaluate the delay jitter of TCP-TFEC to clarify its effectiveness for real-time communication. In this evaluation, there is no background flow, and TCP-TFEC and TCP-AFEC are downlink flows (from the AP to the wireless terminal). Moreover, the application is FTP, and the number of flows is one. The target rate is 10 Mbps. The other simulation parameters are same as the previous evaluation.

Figure 18 and Fig. 19 show the time change of delay jitter for each TCP when the loss rate is 10% and 20%, respectively. From Figs. 18 and 19, the fluctuation of delay jitter for TCP-AFEC is larger than that of TCP-TFEC when the loss rate is large. On the other hand, the fluctuation of delay jitter for TCP-TFEC is small over time. However, the fluctuation of delay jitter for TCP-TFEC is large when communication begins.

Here, we discuss delay jitter according to transient and steady states. Table 2, Table 3, and Table 4 show the aver-

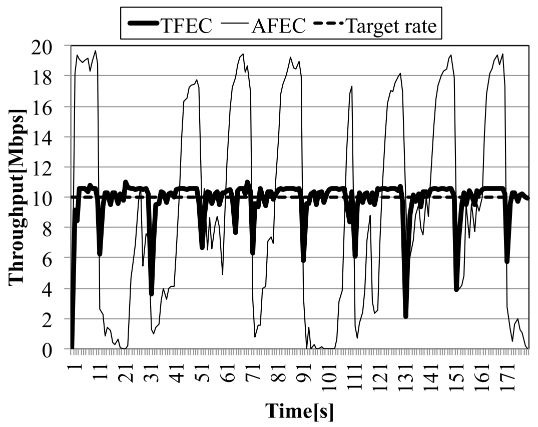

Fig. 16 Time change of throughput when the loss rate changes every 10 s (loss rate: 0.1% and 10%, target rate: 10 Mbps)

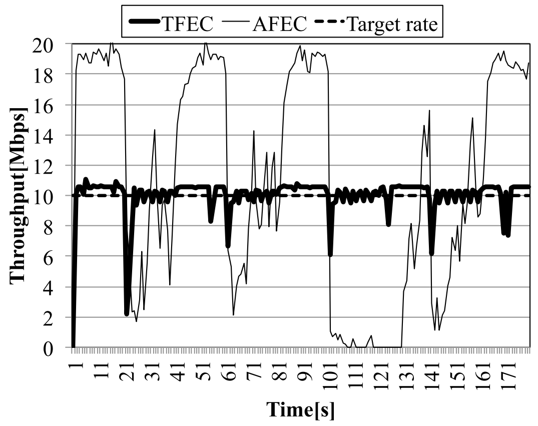

Fig. 17 Time change of throughput when the loss rate changes every 20 s (loss rate: 0.1% and 10%, target rate: 10 Mbps)

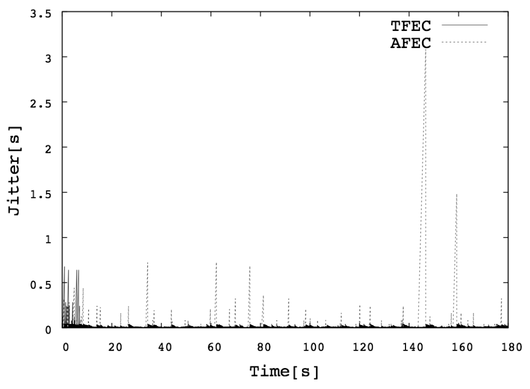

Fig. 18 Time change of delay jitter (loss rate: 10%, target rate: 10 Mbps)

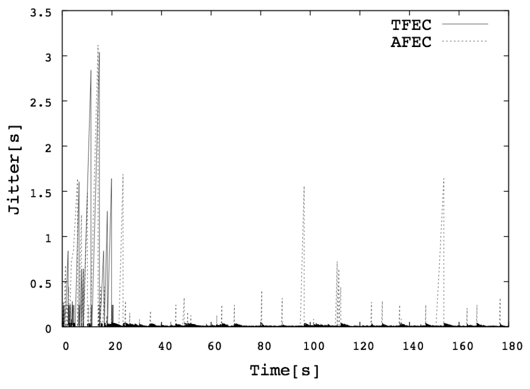

Fig. 19 Time change of delay jitter (loss rate: 20%, target rate: 10 Mbps)

age (Ave.) and standard deviation (Std.) of delay jitter for each TCP and loss rate from 0 s to 180 s (total duration), 0 s to 30 s (transient state), and 31 s to 180 s (steady state), respectively. From Table 2, the average value of the delay jitter through the communication is very small for each TCP. TCP-TFEC can decrease delay jitter by approximately 1 ms compared to TCP-AFEC when the loss rate is 20%. However, when the loss rate increases from 10% to 20%, the standard deviation of delay jitter increases for each TCP. Here, we compare the results of the transient state (Table 3) and the steady state (Table 4). From Table 3, when communication is in the transient state and the loss rate is 20%, the average and standard deviation of delay jitter are large because the appropriate redundancy cannot be set by TCP-TFEC and TCP-AFEC when communication begins. On the other hand, from Table 4, the delay jitter of both TCP-TFEC and TCP-AFEC is small because both TCPs can set the appropriate redundancy. Thus, in case of total duration, the average and standard deviation of delay jitter becomes larger when the loss rate increases. Here, in the steady state (Table 4), when the loss rate is 20%, the standard deviation of delay jitter for TCP-AFEC is 76.1 ms. However, that of

Table 2 Delay jitter of TCP-TFEC and TCP-AFEC (from 0 s to 180 s, target rate: 10 Mbps)

Loss rate	TFEC		AFEC	
	Ave.	Std.	Ave.	Std.
10%	0.91 ms	7.89 ms	0.84 ms	20.21 ms
20%	0.68 ms	400.14 ms	1.60 ms	493.30 ms

Table 3 Delay jitter of TCP-TFEC and TCP-AFEC (from 0 s to 30 s, target rate: 10 Mbps)

Loss rate	TFEC		AFEC	
	Ave.	Std.	Ave.	Std.
10%	1.10 ms	7.89 ms	1.35 ms	20.20 ms
20%	83.14 ms	390.38 ms	100.77 ms	481.30 ms

Table 4 Delay jitter of TCP-TFEC and TCP-AFEC (from 31 s to 180 s, target rate: 10 Mbps)

Loss rate	TFEC		AFEC	
	Ave.	Std.	Ave.	Std.
10%	0.63 ms	1.34 ms	0.79 ms	8.58 ms
20%	0.61 ms	12.52 ms	1.38 ms	76.10 ms

TCP-TFEC is 12.52 ms. Therefore, the difference in delay jitter between TCP-AFEC and TCP-TFEC is approximately 63 ms. Here, ITU-T Y.1541 [25] is known as an indicator of QoS for real-time communication. The standard deviation of delay jitter for TCP-TFEC satisfies the criterion of QoS Class0 in ITU-T Y.1541 (50 ms). Therefore, TCP-TFEC is effective for a bandwidth-guaranteed TCP [11], [12], [14] even if the WLAN network has high loss rate.

6. Conclusion

In this paper, we have discussed the problem and evaluated the performance of TCP-AFEC in a wireless LAN environment. To address the problem, we proposed and evaluated a new TCP based on a redundancy setting method for FEC, i.e., TCP-TFEC. We obtained following results using network simulator NS2:

- TCP-TFEC can improve throughput by approximately 1.2 Mbps (95th percentile) compared to TCP-AFEC when the packet loss rate is 20% in IEEE802.11g (54Mbps) environment.
- TCP-TFEC can guarantee throughput when the packet loss rate changes dynamically.
- The standard deviation of delay jitter for TCP-TFEC satisfies the criterion of QoS Class0 in ITU-T Y.1541 (50 ms).

Future work includes detailed evaluations of the proposed method, such as adaptability to other environments, control overhead, and comparative evaluations with other existing methods. In this study, we used IEEE802.11g in order to evaluate basic performance for the initial study in the evaluations. However, in these days, IEEE802.11n and 11ac which are developed based on IEEE802.11g and have higher transmission rate are often used. Therefore, future works include the evaluation of the proposed method on IEEE802.11n/ac environment.

Acknowledgments

This work was partly supported by JSPS KAKENHI Grant Numbers JP26420367, JP26280032, JP15K00431, and JP15H02688, the Hiroshima City University Grant for Special Academic Research, and Project Research Grants of the Graduate School of Information Sciences, Hiroshima City University. The authors would like to thank Prof. Tutomu Murase for supporting our study.

References

[1] F. Teshima, H. Obata, R. Hamamoto, and K. Ishida, "Redundancy setting method for TCP congestion control based on FEC over wireless LAN," Proc. ASON 2015, pp.259–264, 2015.
[2] BIGLOBE Home Page. available from http://broadband.biglobe.ne.jp/, 2016.
[3] YouTube Home Page. available from http://www.youtube.com/, 2016.
[4] IEEE Standard, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," ANSI/IEEE Std 802.11, 2012.
[5] J. Postel, "User Datagram Protocol," RFC 768, 1980.
[6] M. Allman, V. Paxson, and E. Blanton, "TCP Congestion Control," RFC 5681, 2009.
[7] C.-F. Wong, W.-L. Fung, C.-F.J. Tang, and S.-H.G. Chan, "TCP Streaming for Low-Delay Wireless Video," Proc. 2nd international conference on Quality of Service in Heterogeneous Wired/Wireless Networks, 2005.
[8] J. Yan, W. Muhlbauer, and B. Plattner, "Analytical Framework for Improving the Quality of Streaming Over TCP," IEEE Trans. Multimedia, vol.14, no.6, pp.1579–1590, 2012.
[9] M. Oulmahdi, C. Chassot, and E. Exposito, "An Energy-Aware TCP for Multimedia Streaming," Proc. SaCoNeT 2013, pp.1–5, 2013.
[10] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," Proc. ACM SIGCOMM 2000, vol.30, no.4, pp.43–56, 2000.
[11] H. Shimonishi, T. Hama, and T. Murase, "TCP congestion control enhancements for streaming media," Proc. IEEE CCNC 2007, pp.303–307, 2007.
[12] K. Akase, H. Obata, and K. Ishida, "A Gentle TCP Congestion Control Method for Obtaining Stable Throughput," IEICE Trans. Commun., vol.J92–B, no.4, pp.624–632, 2009. (in Japanese)
[13] R.E. Blahut, Theory and Practice of Error Control Codes. Reading, Massachusetts: Addison-Wesley, 1983.
[14] T. Tsugawa, N. Fujita, T. Hama, H. Shimonishi, and T. Murase, "TCP-AFEC: An Adaptive FEC Code Control for End-to-End Bandwidth Guarantee," Proc. 16th International PV 2007, pp.294–301, 2007.
[15] K. Ijiri, S. Ohzahata, and K. Kawashima, "TCP window control for QoS improvement based on channel occupancy information over wireless LAN," Proc. INDIN2010, pp.1016–1021, 2010.
[16] W. Hui, Y, Fukushima, and T. Yokohira, "Throughput improvement of TCP proxies in network environment with wireless LANs," Proc. 2014 IEEE Region 10 Symposium, pp.82–87, 2014.
[17] H. Oda and H. Hisamatu, "Compound TCP+ for fairness improvement among Compound TCP connections in a wireless LAN," Proc. CQR2010, pp.1–6, 2010.
[18] S. McQuistin, C. Perkins, and M. Fayed, "TCP Hollywood: An unordered, time-lined, TCP for networked multimedia applications," Proc. 2016 IFIP Networking Conference (IFIP Networking) and Workshops, pp.422–430, 2016.
[19] K. Akase, H. Obata, T. Murase, Y. Hirano, and K. Ishida, "Setting methods of a slow start threshold in TCPs for guaranteeing a bandwidth over wireless LAN," IEICE Trans. Commun., vol.J93–B, no.2, pp.153–165, 2010. (in Japanese)
[20] H. Obata, A. Momota, J. Funasaka, and K. Ishida, "A Control Method of Guaranteeing Throughput for TCP Communication Considering Handover over WLAN," Proc. ADSN2014, pp.113–118, 2014.
[21] R. Roth, Introduction to Coding Theory, Cambridge University Press, 2006.
[22] Network Simulator - ns (version 2), available from http://www.isi.edu/nsnam/ns, 2017.
[23] IEEE802.11g, "IEEE Standard for Local and Metropolitan Area Networks Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," 2016.
[24] K. Taira, H. Obata, and K. Ishida, "TCP Proxy Mechanism TCP-gSTAR for High-Speed Satellite Internet," IEICE Trans. Commun., vol.J91-B, no.12, pp.1587–1599, 2008. (in Japanese)
[25] Recommendation ITU-T Y.1541, "Network performance objectives for IP-based services," 2011.

Fumiya Teshima received B.E. and M.E. degrees in Computer Engineering from Hiroshima City University, Japan, in 2014 and 2016, respectively. Currently, he is working with NEC Corporation, Japan. His research interests are wireless network systems and high speed transport protocols.

Hiroyasu Obata received B.E., M.E., and Ph.D. degrees in Computer Engineering from Hiroshima City University, Japan, in 2000, 2002, and 2007, respectively. From 2002 to 2003, he was with KDDI Co., Ltd. From 2003 to 2009, he was a Research Associate at Hiroshima City University. From 2009 to 2016, he was a Lecturer at Hiroshima City University. He is currently an Associate Professor in the Graduate School of Information Sciences, Hiroshima City University. His research interests include computer communications on wireless networks, such as satellite links and wireless LANs, access control, and high-speed transport protocols. He received the Information Network Research Award of IEICE in 2015. Dr. Obata is a member of IEEE (U.S.A.) and IPSJ (Japan).

Ryo Hamamoto received B.E., M.E., and Ph.D. degrees in Computer Engineering from Hiroshima City University, Japan, in 2012, 2014, and 2016, respectively. Currently, he is working with Security From 2016 to 2017, he was working with Security Research Laboratories NEC Corporation, Japan. Currently, he is working with Global Engineering Center Kobelco Construction Machinery Corporation Limited., Japan. His research interests lie in the area of both wireless communication systems and security systems for IoT. He received the IEICE Information Network Research Award in 2012 and 2015. He is a member of IEEE (U.S.A.) and ACM (U.S.A.).

Kenji Ishida received B.E., M.Sc., and Ph.D. degrees from Hiroshima University, Japan, in 1984, 1986, and 1989, respectively. He was at Hiroshima Prefectural University from 1989 to 1997. From 1997 to 2003, he was an Associate Professor at Hiroshima City University. Since 2003, he has been a Professor in the Department of Computer Engineering, Faculty of Information Sciences, Hiroshima City University. His research interests include distributed computing systems and designing control procedures for computer networks. He received the Information Network Research Award of IEICE in 1998, 2000, 2012, and 2015. Dr. Ishida is a member of IEEE (U.S.A.), ACM (U.S.A.), and IPSJ (Japan).