TCP Window-size Delegation for TXOP Exchange in Wireless Access Networks

Takayuki Nishio*, Ryoichi Shinkuma*, Tatsuro Takahashi*, and Go Hasegawa†
*Graduate School of Informatics, Kyoto University, Japan
Email: {shinkuma, ttakahashi}@i.kyoto-u.ac.jp
†Cybermedia Center, Osaka University, Japan
Email: hasegawa@cmc.osaka-u.ac.jp

Abstract—We propose a TCP window-size delegation method for TXOP Exchange applicable to the downlink in wireless access networks. In TXOP Exchange, the ‘compliant’ stations (STAs) cooperatively use their available bandwidth in accordance with their required QoSs. TXOP Exchange was previously validated for the uplink. The proposed delegation method enables STAs to delegate their bandwidth for the downlink as well without requiring any modifications to the legacy access point or the STAs. Simulation demonstrated that this method works well.

I. INTRODUCTION

In IEEE 802.11 wireless local area networks (WLANs), which are extensively used as access networks in offices, homes, and public spaces, stations (STAs) connected to an access point (AP) compete for transmission opportunities (TXOPs) using the carrier sense multiple access with collision avoidance (CSMA/CA). When the traffic load is saturated, TXOPs are almost equally assigned to the STAs without considering each STA’s quality-of-service (QoS) requirement. Although many QoS control mechanisms for WLANs have been proposed [4], [5], they are not widely used because most of them require modifications to both the legacy STAs and APs.

TXOP Exchange was developed to solve this problem [1]-[3]. In TXOP Exchange, ‘compliant’ STAs cooperatively use their available bandwidth in accordance with their required QoSs. Consider an example of a wireless access network with one AP, two STAs compliant with our architecture, and other STAs. Each of the STAs is uploading a large file, like a video file or a zipped file of photos, to a storage server. The throughput via the AP is saturated at 800 kbps. Suppose that one of the compliant STAs, STA 1, wants to increase its throughput to 1 Mbps to upload its file faster. If the other compliant STA, STA 2, is not interested in uploading its file faster, then, as shown in Fig. 1, it can delegate some of its TXOPs to STA 1 so that the throughput increases up to 1 Mbps. In this example, STAs 1 and 2 are a TXOP client and a TXOP provider, respectively. The provider and client roles can switch from at the next moment to moment. Thus, cooperative exchange of TXOPs enables the QoS requirements to be better satisfied. Moreover, TXOP Exchange enables compliant STAs in WLAN to flexibly exchange bandwidth without requiring modifications to the legacy APs and without degrading the performances of the other “non-compliant” STAs, as illustrated in Fig. 2. Conventional QoS mechanisms in WLANs including IEEE 802.11e enhanced distributed channel access (EDCA) are unable to let STAs exchange bandwidth only between them; they always affect other STAs [4], [5]. We previously focused on the use of TXOP Exchange for the uplink [1]. We have now extended TXOP Exchange so that it can be used for the downlink as well. In the uplink, the throughput of each STA depends on the number of TXOPs it has. Therefore, for the uplink, we modified only the STA-side media access control (MAC) protocol so that the compliant STAs can directly exchange TXOPs without affecting the performances of the other STAs. In the downlink, the packets are sent to the STAs from the connecting AP; in other words, the STAs share the AP for their downlink communications. Therefore, to enable an STA to delegate its TXOPs to another STA, we have to control the number of packets in the AP sending queue. Considering the implementation constraint and the cost, we created a transport-level control method that uses proxy servers to control the number of packets arriving at the AP sending queue. A proxy server manages TCP connections of compliant STAs. In this method, a TXOP provider delegates a chance to increase its window size to the TXOP client, which is controlled by the proxy server. The proxy server also decreases the window size for the TXOP provider where as that for the TXOP client would be decreased with the legacy TCP. This method does not require any modifications of the APs or the legacy STAs and is applicable to access networks other than WLANs. Moreover, we discuss a coordination mechanism that ensures fair incentives between STAs.

The paper is organized as follows: in Section II, we briefly review our TXOP delegation method for the uplink in WLANs. In Section III, we describe the challenges involved in implementing the TXOP delegation method for the downlink and describe how we meet them. We also show simulation results and discuss how to determine control parameters in our method. In Section IV, we introduce the mechanism we described for promoting cooperation between STAs. We conclude in Section V with a summary of the key point and a look at future work.

II. TXOP EXCHANGE FOR UPLINK

The CSMA/CA was originally designed for assigning TXOPs equally to the STAs when the traffic load is heavy. An
STA is allowed to send a data frame every time it obtains a TXOP. Since there has been no procedure in CSMA/CA that allows an STA to delegate its TXOPs to another STA, we need to newly implement a TXOP delegation method in CSMA/CA. However, such a method should not adversely affect the legacy STAs. Moreover, the legacy APs should not require modifications. In addition, the modifications needed to make compliant STAs should be minimal compared with the conventional IEEE 802.11. To meet these requirements, we described a TXOP delegation method in which the sender address in the request-to-send (RTS) frame is replaced with sender [1], [2].

In CSMA/CA with RTS/CTS, an STA sends the AP an RTS frame before sending a data frame, and the AP sends a CTS frame back to the STA to allocate a TXOP to it and to prohibit the other STAs from sending any frames during the network allocation vector (NAV) period. Since the RTS frame includes the sender address field, the AP can recognize which STA sent it. The TXOP provider replaces the source address field in the RTS frame with the client’s to force the AP to allocate a TXOP to the client with the corresponding CTS frame. We call this TXOP the coop RTS frame. The provider calculates the transmission rate of the client and includes it in the coop RTS frame. The TXOP provider replaces the source address field in the RTS frame before sending a data frame, and the AP sends a CTS frame back to the STA to allocate a TXOP to it and to prohibit other STAs from sending any frames during the network allocation vector (NAV) period. Since the RTS frame includes the sender address field, the AP can recognize which STA sent it. The TXOP provider replaces the source address field in the RTS frame with the client’s to force the AP to allocate a TXOP to the client with the corresponding CTS frame. We call this TXOP the coop RTS frame. The provider calculates the transmission rate of the client and includes it in the coop RTS frame.

III. TXOP EXCHANGE FOR DL

In the downlink of WLANs, the AP sends the packet at the top in its sending queue when it obtains a TXOP on the basis of CSMA/CA. Therefore, the throughput for each STA connected to the AP depends on the number of packets for each STA in the AP sending queue. Consider for example a wireless access network with one AP and three STAs, STA A, STA B, and STA C. The downlink throughputs of the STAs are \( \theta_A = a \cdot \theta_{\text{max}} \), \( \theta_B = b \cdot \theta_{\text{max}} \), and \( \theta_C = c \cdot \theta_{\text{max}} \), where \( a \), \( b \), and \( c \) are the ratio of the number of packets for STA A, B, and C in the queue, \( a + b + c = 1 \), and \( \theta_{\text{max}} \) is the throughput achieved when the AP sends packets to only one STA. Therefore, for instance, \( a \) and \( b \) should be increased and decreased, respectively, while keeping \( c \) constant if we want to assign a portion of the bandwidth for STA B to STA A without affecting STA C, as illustrated in Fig. 2.

There are two ways of controlling the ratios of the number of packets for the STAs in the AP sending queue: (i) implement a packet dropping mechanism in the AP, which requires AP-side modification, and (ii) adjust the number of packets arriving at the AP, which requires a modification in the servers sending them. Here, we consider the second way and discuss the transport-level modification. Since it is unrealistic to modify the transport protocol for every server, we assume that proxy servers are located between the STAs and corresponding servers, as illustrated in Fig. 3. This is described in more detail in the following subsection.

A. TXOP delegation by TCP window-size delegation

1) System model: The system model of our TXOP Exchange for the downlink is shown in Fig. 3. A proxy server is located in each autonomous system (AS) network, and our compliant STAs maintain sessions with the proxy server and download via the proxy server. Here, we consider a pair of provider and client, which connects to an access point. Since the number of packets sent by a server to an STA in the round trip time is approximately the same as the window size of the TCP flows for the STA [7], we described the proxy server to change the provider’s and client’s window sizes while keeping the total size the same so as to control the ratio of the numbers of their packets arriving at the AP with keeping the number of packets for other. The proxy server manages flows for the provider and client and delegates the window size of a flow for the provider to a flow for the client. Since the proxy server terminates TCP connections from original servers for the provider and client, our method only cares about TCP flows between the proxy server and them.

2) The Proxy server: Figure 4 shows ideal change in the window size of flows for the provider, the client, and non-compliant STA. The proxy server increases the provider’s or client’s window size as described bellow when the flow is in the congestion-avoidance phase, in which the proxy server receives an ACK packet from the provider, but not triple-duplicate ACKs:

\[ \text{We here assume Reno-based TCP; our method can be applied to Tahoe and NewReno because they are very similar to Reno in congestion avoidance. It will be in future work to apply our method to RTT-based algorithms like Vegas.} \]
As in the conventional TCP, the increased window size for the provider, \( \Delta p \), is equal to \( 1/W^p \), where \( W^p \) is the current window size for the provider.

2) The virtual window size for the provider, \( W^p_v \), is given by \( W^p_v = W^p + \Delta p \). If \( W^p_v = 0 \), \( W^p = W^p + \Delta p \).

3a) With probability \( 1 - \alpha \), as in the conventional TCP, the window size for the provider is increased to \( W^p = W^p + \Delta p \).

3b) With probability \( \alpha \), the window size for the client \( W^c \) is increased to \( W^c = W^c + \Delta p \).

Note that virtual window size for STA A, \( W^A_v \), is the window size that would be achieved if STA A did not delegate its congestion window and that the virtual window size is reset to zero when congestion occurs. When the flow for the provider is in slow-start phase, the delegation procedures described above does not occur. When the flow for the client is in slow-start phase and the flow for the provider is in congestion-avoidance phase, the increase in the window size is kept held until the flow for the client is in congestion-avoidance phase and then will be added to the client window size.

Although the above procedure enables a client to increase its window size, its throughput does not necessarily increase because packet loss is more likely than before increasing the window size. Therefore, when the proxy server receives triple-duplicate ACKs from a client, it performs the following action:

1a) With probability \( \beta \), a provider is selected; the window size and slow-start threshold of the flow for the provider are decreased, while the ones for the client are not changed.

1b) With probability \( 1 - \beta \), the client is selected; only the window size and the slow-start threshold for the client are decreased.

2) The slow-start threshold of the flow for the STA selected in 1) are given by: \( Th^x_s = W^x_s/2 \).

3) \( W^x_s \) is reset to 0.

4) Client’s corresponding packets are retransmitted.

We will show how the performance of our method changes according to \( \alpha \) and \( \beta \) and discuss how to determine \( \beta \) in the next subsection.

B. Simulation evaluation

We evaluate our delegation method using QualNet [8] simulation, considering only the downlink in a WLAN. We used the network model illustrated in Fig. 5, where the wireless link is modelled as simply a bottleneck link not a CSMA/CA link because every packet in the WLAN is sent only from the AP and, as we mentioned in Sect. I, our method can be used for access networks other than WLANs.

The provider and client use TCP to download data from the proxy server, while the other STA use TCP to download data from the other server. The provider and client each receive one TCP flow, while the other STA receive one or more TCP flows.

We first demonstrate how conventional methods work. The simplest way of increasing throughput for a STA is to assign two or more flows for the STA. In Fig. 6, the horizontal axis indicates the number of flows assigned to STA A. We have to mention that, in our simulations, the packet loss rates were approximately 1%; the network was not overloaded. We observe in this figure, although it certainly increased the throughput for STA A, throughputs for both STAs B and C decreased. As seen in this example, conventional methods may differentiate throughputs but do not enable only a compliant STA to delegate its throughput to another compliant STA.

Figures 7 (a) and (b) show the throughput of each STA as a function of \( \beta \) for \( \alpha = 0.3 \) and 1.0 respectively when each of them is receiving only one TCP flow. “w/o delegation” means the average throughputs for STAs when our method is not used. In both cases, throughput delegation from provider to client increased with \( \beta \) while provider’s throughput decreased. Thus, client throughput can be controlled by simply changing \( \beta \). However, the throughputs for the other STA vary with \( \alpha \), which is undesirable because our goal is to enable the delegation without effect on other STAs.

Figures 8 (a), (b), and (c) show the throughput of each STA as a function of \( \alpha \) for \( \beta = 0.3, 0.5, \) and 1.0 when each of them is receiving only one TCP flow from the corresponding...
or proxy server. Client throughput was mostly constant and independent of $\alpha$. However, the throughput for the other STA increased when $\alpha$ exceeded a certain point. When $\beta$ was smaller than around 0.5, the throughput for the other STA was the same as without our method when $\alpha$ was appropriately set.

Figures 9 and 10 show the throughputs of each STA for $\alpha = 1.0$ and $\beta = 0.3$ when the other STA was receiving two TCP flows from the corresponding server. Our method still worked well as long as the parameters are set appropriately.

The above results show that $\beta$ must be set so that, for a given $\alpha$, the delegation does not affect the throughputs for the other STAs. We here discuss how we should determine $\beta$. We denote the number of times triple-duplicate ACKs occur in a certain period for the client and for the provider as $N^c$ and $N^p$, respectively. We thus got $N^c = (1 - \beta)N_0^c + \beta \cdot N_0^p$ and $N^p = N_0^c - \beta \cdot N_0^p$, where $N_0^c$ and $N_0^p$ are the number of times triple-duplicate ACKs occur when $\beta = 0$. The effect on the other STAs is minimized when $N^c = N^p$ because the total of the decreased window size caused by triple-duplicate ACKs for the provider and the client is almost twice as that for the other non-compliant STA. Therefore $\beta$ should be set to $(N_0^c - N_0^p)/2N_0^c$. Figure 11 shows throughputs of each STA with $\beta = (N_0^c - N_0^p)/2N_0^c$. We can see that the throughput of client is successfully increased without deceasing that of other STA.

IV. Coordination Mechanism for TXOP Exchange

We cannot expect STAs to cooperate without incentive, and in TXOP Exchange there is no incentive for STAs to provide TXOPs to other STAs. To create an incentive, we introduced a coordination mechanism [2], [3]. It is implemented on a coordination server, as illustrated in Fig. 12. It can be applicable to downlink TXOP Exchange.

The coordination server was designed talking into account security restriction and STAs' mobility. It can play the role of the proxy server described in Sect. III-A. It manages the STA information needed for TXOP Exchange, including the MAC/IP addresses and the required STA throughputs. When a compliant STA requests an increase in throughput (becoming a TXOP client), the coordination server (i) finds a (prospective) TXOP provider in the same area; (ii) determines whether the prospective provider can support the request by comparing the number of TXOPs available from the provider with the number required by the client; (iii) sends the provider the client information if the request is supportable; and (iv) instructs the
provider to accept the request while ensuring fair incentive between STAs in accordance with their exchange histories.

There are many strategies for designing incentive mechanisms and algorithms that stimulate cooperation between STAs in TXOP Exchange. We could design such an algorithm based on a microeconomic approach such as auction or pricing [10]. We set up a game in which the players use a mixed strategy profile, i.e. strategies are chosen with certain probabilities. We model the time instances of these TXOP exchanges as instances of a repeated game. Generally, we wish to avoid the apparent inefficient Nash equilibrium in which both nodes choose not to cooperate in the stage game. If the stage game is played repeatedly, the Folk Theorem [11], [12] can be invoked to characterize subgame perfect equilibria as long as the payoffs for STAs strictly dominate that of when STAs do not cooperate. However, there still remains the issue of which specific equilibrium is most desirable. We previously [2], [3] considered a Nash bargaining solution (NBS) based algorithm to obtain an optimal equilibrium, which was inspired by [6]. The NBS maximizes the product of players’ payoffs and brings a Pareto optimal allocation that is also proportionally fair [9].

In our NBS-based coordination algorithm, an STA accepts a TXOP request and provides TXOPs in accordance with the STA’s providing probability, which is calculated by means of the NBS. We simulated the coordination algorithm and observed that the STAs were motivated to cooperate and share the TXOPs so that Pareto optimality with proportional fairness was achieved [2], [3]. We are now considering the application of our NBS-based coordination algorithm to the downlink TXOP Exchange described here.

V. CONCLUSION

We have developed a TCP window-size delegation method for downlink TXOP Exchange, in which a proxy server lets one station (the TXOP provider) delegate TCP window size to another station (the TXOP client). This method enables a STA to flexibly delegate TXOPs to another STA without adversely affecting the legacy STAs using the appropriate parameters. This was demonstrated by computer simulations. We also confirmed that our method requires no modification to legacy access points and STAs and needs only minimal modifications of the TCP functions of the proxy server. Future work includes developing an algorithm that automatically determines the optimal $\alpha$ and $\beta$ so that the other STAs are not adversely affected and developing a coordination algorithm that ensures fair incentives in an environment with both uplinks and downlinks.

ACKNOWLEDGEMENTS

This work is supported in part by the National Institute of Information and Communications Technology (NICT), Japan, under Early-concept Grants for Exploratory Research on New-generation Network.

REFERENCES