Experimental Analysis on How Access Point Scanning Impacts on TCP Throughput over IEEE 802.11n Wireless LAN

Kento Kobayashi, Yoshiki Hashimoto, Masataka Nomoto, Ryo Yamamoto, Satoshi Ohzahata, and Toshihiko Kato
University of Electro-Communications
Tokyo, Japan
e-mail: k1552010@edu.cc.uec.ac.jp, hys3224@net.is.uec.ac.jp, noch@net.is.uec.ac.jp, ryo_yamamoto@is.uec.ac.jp, ohzahata@is.uec.ac.jp, kato@is.uec.ac.jp

***Abstract*— In IEEE 802.11 wireless LAN, there is a problem that the access point scanning at stations which uses the power management function gives impacts on the performance of TCP communication. In paper, we show the result of experiments on this problem for uploading and downloading TCP data transfer over 802.11n wireless LAN. For the uploading transfer, we analyze the influence to TCP throughput focusing on the TCP small queues that limit the amount of data in wireless LAN's sending queue. As for the downloading transfer, we discuss the influence by the TCP congestion control algorithm at TCP senders and A-MPDU transmission rate at the access point.**

Keywords- WLAN; IEEE802.11n; Access Point Scanning; Power Management; TCP Small Queues; TCP Congestion Window Validation.

I. Introduction

Nowadays, 802.11n [1] is one of most widely adopted IEEE wireless LAN (WLAN) standards. It establishes high speed data transfer using the higher data rate support (e.g., 300 Mbps), the frame aggregation in Aggregated MAC Protocol Data Unit (A-MPDU) and the Block Acknowledgment mechanism. On the other hand, TCP introduces some new functions to establish high performance over high speed WLANs. CoDel [2] and TCP small queues [3], which aim to resolve the Bufferbloat problem [4] are examples.

We reported some performance evaluation results on TCP behaviors over 802.11n [5][6]. In those evaluations, we have encountered the situation that the TCP throughput decreases periodically. By analyzing the captured WLAN frame logs during the performance degradation, its reason is detected to be periodically transmitted data frames without data (*Null data frames*) with the power management field set to 1 in the WLAN header. These frames are used by WLAN stations to inform access points that they are going to sleep and to ask the associated access point not to send data frames. It is pointed out that WLAN stations use Null data frames to scan another available access point periodically [7].

Through more detailed performance analysis, we found that the impacts of Null data frames on TCP data transfer vary by the functions of TCP used in the communication. Specifically, the TCP sender behaviors and the throughput degradation depend on the direction of TCP data transfer (uploading from station to access point or downloading in the reverse direction), whether the TCP small queues are used or not, and what kind of congestion control algorithm is used. This paper shows the results of experimental analysis about the impacts on TCP throughput given by Null data frames used for the access point scanning. The rest of this paper consists the following sections. Section 2 shows the technologies relevant to this paper. Section 3 explains the experimental settings. Sections 4 and 5 give the results of the experiments for uploading TCP data transfer and downloading TCP data transfer, respectively. In the end, Section 6 gives the conclusions of this paper.

II. Relevant Technologies

A. Power management function and Null data frames

As described above, IEEE 802.11 standards introduce the power management function. In the WLAN frame format depicted in Fig. 1 (a), bit 12 of the Frame Control field is the *Power Management field* (shown in Fig. 1 (b)). By setting this bit to 1, a station informs the associated access point that it is going to the *power save mode*, in which the station goes to sleep and wakes up only when the access point sends beacon frames. By setting the bit to 0, it informs the access point that it goes back to the *active mode*.

This function is used for several purposes. One example is the case that a station is actually going to sleep to save its power consumption for some time period. In this case, a station wakes up only at the timing of receiving beacon frames from the access point. If the access point has data frames to deliver to the sleeping station, it indicates, in the Traffic Indication Map element in a beacon frame, that there are some frames to the station. In response to this information, the station requests the delivery of data frames by use of a PS-Poll frame.

Another is the access point scanning. For example, a Lunix terminal executing the NetworkManager application searches periodically for an access point which provides

Octets:	2	2	6	6	6	2	6	2	4	0–7951	4
	Frame Control	Duration /ID	Address 1	Address 2	Address 3	Sequence Control	Address 4	QoS Control	HT Control	Frame Body	FCS

MAC Header

(a) Structure of WLAN frame.

	B0 B1	B2 B3	B4 B7	B8	B9	B10	B11	B12	B13	B14	B15
	Protocol Version	Type	Subtype	To DS	From DS	More Frag-ments	Re-try	Power Manage-ment	More Data	Protect-ed Frame	Or-der
Bits:	2	2	4	1	1	1	1	1	1	1	1

(b) Structure of Frame Control field.

Fig. 1. IEEE 802.11 WLAN frame format [1].

stronger radio signal than the access point with which the terminal is associated [8]. There are two schemes for the access point scanning; the passive scanning in which a station waits for a beacon frame from another access point, and the active scanning in which a station sends a probe request frame and waits for a probe response frame for the request. In either scheme, a station needs to ask the current access point to stop sending data frames to it. For this purpose, the station sends a frame with the Power Management field set to 1.

Null data frames are used by WLAN stations to inform access points of the transfer to the power save mode or to the active mode [9]. This is a frame which contains no data (Frame Body in Fig. 1 (a)). An ordinary data frame has the type of '01' and the subtype of '0000' in the Frame Control field. That is, B3 and B2 in Fig. 1 (b) are 0 and 1, and B7 through B4 are all 0. On the other hand, Null data frame has the type of '01' and the subtype of '0100'. While an ordinary data frame has three Address fields, a Null data frame has only two Address fields; the transmitter is a station MAC address and the receiver is an access point MAC address. By using Null data frames with the Power Management field set to 0 or 1, stations can request the power management function for access points.

B. TCP small queues

In the Linux operating system with version 3.6 and later, a mechanism called TCP small queues [3] is installed in order to cope with the Bufferbloat problem. It keeps watch on the queues in Linux schedulers and device drivers in a sending terminal, and if the amount of data stored in the queues is larger than the predefined queue limit, it suspends the TCP module until the amount of stored data becomes smaller than the limit. During this TCP suspension, the data which applications transmitted is stored in the TCP send socket buffer, which may cause the application to be suspended if the TCP send socket buffer becomes full. After the TCP module is resumed, it processes the application data stored in the send socket buffer.

The default value of the predefined queue limit is 128 Kbyte, and it is adjustable by changing the following parameter;

/proc/sys/net/ipv4/tcp_limit_output_bytes.

This mechanism is different from the other mechanisms against the Bufferbloat problem, such as CoDel, in the point that no TCP segments are discarded intentionally to protect the increase of buffered data in sending queues, which may cause a large communication delay.

C. Congestion window validation

The TCP congestion control uses the congestion window size (*cwnd*) maintained in TCP senders. TCP senders transmits TCP data segments under the limitation of cwnd and the window size advertised by TCP receivers. In general, cwnd is increased when TCP senders receive TCP acknowledgment (ACK) segments and decreased when any data segments are retransmitted.

This mechanism is considered to work well under the assumption that the throughput of data transfer is limited by cwnd. When the throughput is controlled by an application in a TCP sender, however, the data amount floating over network without acknowledged (it is called *flight size*) might be smaller than cwnd. In this case, cwnd also increases when an ACK segment is delivered to the sender and the value of cwnd becomes much larger than the current flight size. This means that the value of cwnd is invalid in this stage. If the TCP sender changes its status from application limited to cwnd limited suddenly, TCP segments corresponding to an invalid, i.e., too large cwnd value will rush into network.

In order to resolve such a problems, the congestion window validation (CWV) mechanism is proposed. RTC 2861 [10] proposes the following two rules. (1) A TCP sender halves the value of cwnd if no dat segments are transmitted during a retransmission timeout period. (2) When a TCP sender does not send data segment more than cwnd during a round-trip time (RTT), then it decreases cwnd to

$$(cwnd + sent\ data\ size)/_2$$

in the next RTT time frame.

RFC 7661 [11] revises the above rules and defines a new rule that, if the data size acknowledged during one RTT is smaller than half of cwnd, a TCP sender does not increase cwnd in the next RTT time frame. It defines the procedure in the case of congestion separately.

III. EXPERIMENTAL SETTINGS

Fig. 2 shows the configuration of the performance evaluation experiment we conducted. Two stations conforming to 802.11n with 5GHz band are associated with one access point, which is connected to a server through 1Gbps Ethernet. One station called STA1 is located at a near position to the access point, and communicates with the server through the access point. The other station called STA2 is used to monitor WLAN frames exchanged between STA1 and the access point.

We use commercially available notebooks for the stations and the server. The access point used is also off-the-shell product. The detailed specification of the notebook and the access point is shown in Table 1. The access point is able to use the 40 MHz channel bandwidth and provides the MAC level data rate from 6.5 Mbps to 300 Mbps.

In the experiment, the data is generated by iperf [12] in the upload direction from STA1 to the server and the download direction from the server to STA1. The conditions for the experiment are the followings.

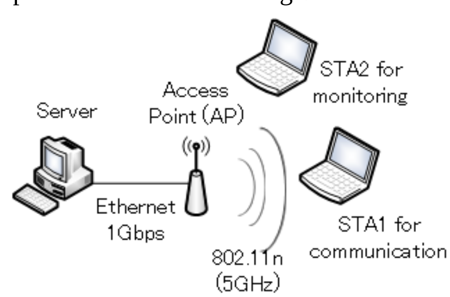

Fig. 2. Network configuration in experiment.

TABLE I. SPECIFICATION OF NOTEBOOK AND ACCESS POINT.

NOTE	Manufacturer/Model	DELL Insilon 14
	Operating system	Ubuntu 14.04LTS (kernel 3.13) or Ubuntu 12.04LTS (kernel 3.2)
	WLAN driver	ath9k
AP	Manufacturer/Model	BUFFALO AirStation WZR-HP-AG300H
	WLAN chip	Atheros AR7161
	WLAN driver	ath9k

- Use or non-use of the TCP small queues, and
- use of CUBIC TCP [13] or TCP NewReno [14] as a congestion control mechanism.

When we use the TCP small queues, we installed Ubuntu 14.04LTS in the notebook, and in the case not to use it, we installed Ubuntu 12.04 LTS.

During the data transmissions, the following detailed performance metrics are collected for the detailed analysis of the communication;

- the packet trace at the server, STA1 and STA2 , by use of *tcpdump*,
- the TCP throughput for every second, calculated from packet trace at TCP sender,
- the WLAN related metrics, such as the MAC level data rate, the number of A-MPDUs sent for a second and the number of MPDUs aggregated in an A-MPDU, from the device driver ath9k [15] at the access point and STA1, and
- the TCP congestion window size at the server, by use of *tcpprobe* [16].

IV. ANALYSIS OF UPLOADING TCP DATA TRANSFER

In the experiments for uploading TCP data transfer, the results were different depending on whether the TCP small queues are used or not. On the other hand, the TCP congestion control algorithms did not affect the results so much. This section shows the results for uploading TCP data transfer focusing on the use or non-use of TCP small queues using CUBIC TCP.

A. *Results when TCP small queues are used*

Fig. 3 shows the time variation of TCP throughput and cwnd in the case that the TCP small queues are used in STA1. From this result, we can say that the throughput degradations occur periodically. Specifically, each *throughput degraded period* is around 10 sec. and such a period happens approximately once in 120 sec. In a *normal period*, the average TCP throughput is 136 Mbps, but it decreases to as much as 57 % in a throughput degraded period.

On the other hand, cwnd does not decrease even in a throughput degraded period, which means that there are no packet losses. Besides that, the increase of cwnd is depressed throughout the TCP communication. In this result, the value of cwnd is limited to around 200 packets.

Fig. 4 shows the packet trace, captured by STA2, of WLAN frames without data in the throughput degraded period starting at time 25 sec. This is the result of analysis

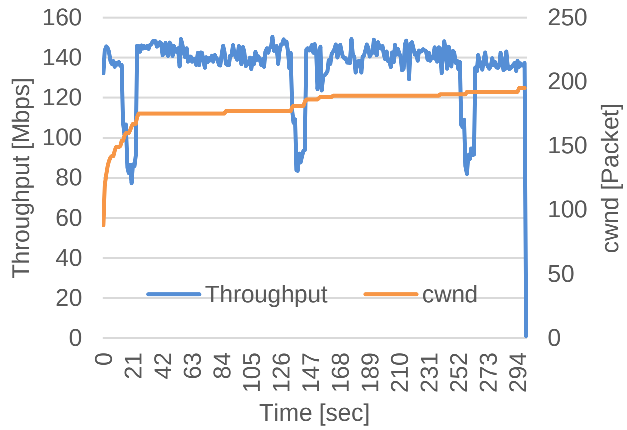

Fig. 3. TCP throughput and cwnd vs. time in uploading data transfer with TCP small queues.

by Wireshark and contains the number of frame, the time of packet capture measeured from the TCP SYN segment, the source and destination MAC address, the protocol type (802.11), the frame length, and the information including name and parameters.

The frame whose number is 105,932, shown inverted to blue in the figure, is a Null data frame. For this frame, the Info column says that “Flags=. . .P. . .TC.” This means that the Power Management field is set to 1 in this frame. The transmitter of this frame is STA1, whose MAC address is “LiteonTe_0b:ce:0c,” and its receiver is the access point whose MAC address is “BuffaloI_27:2a:39.” The sequence control (“SN” in the figure) is 1750 in this frame. In this packet trace, this Null data frame is not acknowledged by the access point. Instead, the same Null data frames are retransmitted eight times by the frames with numbers 105,933 through 105,953. They have the same sequence control (1750), and the Retry field in the Frame Control field is set to 1, as indicated “Flags=. . .PR. .TC” in the figure. From the fourth retransmission, a RTS frame is used before sending a Null data frame and the access point responds it by returning a CTS frame, which allows STA1 to send a Null data frame. But, from the fourth to the seventh retransmissions, the access point does not send any ACK frames. In the end, the access point sends an ACK frame at the eighth retransmission (see the frame with number 105,954). During these frame exchanges, it takes 1.8 msec.

Then, the frame with number 105,955 is a beacon frame broadcasted by the access point. The duration between the ACK frame (No. 105,954) and this beacon frame is 90 msec in the figure.

24 msec after the beacon frame is transmitted, STA1 sends a Null data frame with the Power Management field set to 0, whose number and sequence control is 105,956 and 1751, respectively. Again, this Null data frame is retransmitted four times. In this case, although STA2 captures the corresponding ACK frames from the access point, STA1 retransmits it, repeatedly. After this frame is acknowledged by the ACK frame with number 105,966, STA1 transmit the next Null data frame whose sequence control 1752, and it is immediately acknowledged. These frame exchanges take 1.3 msec.

No.	Time	Source	Destination	Protocol	Length	Info
105931	25.814733	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	58	802.11 Block Ack, Flags=........C
105932	25.815296	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	57	Null function (No data), SN=1750, FN=0, Flags=...P...TC
105933	25.815318	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	57	Null function (No data), SN=1750, FN=0, Flags=...PR..TC
105934	25.815323	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	57	Null function (No data), SN=1750, FN=0, Flags=...PR..TC
105935	25.815326	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	57	Null function (No data), SN=1750, FN=0, Flags=...PR..TC
105938	25.815343	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	58	802.11 Block Ack, Flags=........C
105939	25.817033	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	46	Request-to-send, Flags=...P....C
105940	25.817054		LiteonTe_0b:ce:0c	802.11	40	Clear-to-send, Flags=........C
105941	25.817066	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	57	Null function (No data), SN=1750, FN=0, Flags=...PR..TC
105942	25.817070	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	46	Request-to-send, Flags=...P....C
105943	25.817073		LiteonTe_0b:ce:0c	802.11	40	Clear-to-send, Flags=........C
105944	25.817077	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	57	Null function (No data), SN=1750, FN=0, Flags=...PR..TC
105945	25.817080	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	46	Request-to-send, Flags=...P....C
105946	25.817083		LiteonTe_0b:ce:0c	802.11	40	Clear-to-send, Flags=........C
105947	25.817088	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	57	Null function (No data), SN=1750, FN=0, Flags=...PR..TC
105948	25.817091	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	46	Request-to-send, Flags=...P....C
105949	25.817095		LiteonTe_0b:ce:0c	802.11	40	Clear-to-send, Flags=........C
105950	25.817099	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	57	Null function (No data), SN=1750, FN=0, Flags=...PR..TC
105951	25.817102	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	46	Request-to-send, Flags=...P....C
105952	25.817105		LiteonTe_0b:ce:0c	802.11	40	Clear-to-send, Flags=........C
105953	25.817111	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	57	Null function (No data), SN=1750, FN=0, Flags=...PR..TC
105954	25.817116		LiteonTe_0b:ce:0c	802.11	40	Acknowledgement, Flags=........C
105955	25.907580	BuffaloI_27:2a:39	Broadcast	802.11	379	Beacon frame, SN=3924, FN=0, Flags=........C, BI=100, SSID=klab-n/a
105956	25.931649	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	57	Null function (No data), SN=1751, FN=0, Flags=.......TC
105958	25.931677		LiteonTe_0b:ce:0c	802.11	40	Acknowledgement, Flags=........C
105959	25.931682	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	57	Null function (No data), SN=1751, FN=0, Flags=....R..TC
105960	25.931686		LiteonTe_0b:ce:0c	802.11	40	Acknowledgement, Flags=........C
105961	25.931691	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	57	Null function (No data), SN=1751, FN=0, Flags=....R..TC
105962	25.932891	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	57	Null function (No data), SN=1751, FN=0, Flags=....R..TC
105963	25.932914	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	46	Request-to-send, Flags=........C
105964	25.932918		LiteonTe_0b:ce:0c	802.11	40	Clear-to-send, Flags=........C
105965	25.932922	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	57	Null function (No data), SN=1751, FN=0, Flags=....R..TC
105966	25.932927		LiteonTe_0b:ce:0c	802.11	40	Acknowledgement, Flags=........C
105967	25.932937	LiteonTe_0b:ce:0c	BuffaloI_27:2a:39	802.11	54	Null function (No data), SN=1752, FN=0, Flags=.......TC
105968	25.932940		LiteonTe_0b:ce:0c	802.11	40	Acknowledgement, Flags=........C

Fig. 4. An example of packet trace during throughput degraded period focusing on WLAN frames without data

During this sequence, STA1 and the access point do not send any data frames. This paper refers to the time period as a *sleeping period*. After the last Null data frame with the Power Management field set to 0 is acknowledged, the data transfer from STA1 is restarted. But, several hundred milliseconds later, the similar communication sequence including the Null data frames and beacon frame occurs, and it goes to another sleeping period. This paper refers to the period when data frames are transmitted between sleeping periods as an *awoken period*. During a performance degraded period, there are around 26 pairs of sleeping period and awoken period.

The relationship among those periods are shown in Fig. 5. In the evaluation result, the throughput degraded periods and the normal periods are repeated as shown at the top of this figure. The average duration for them is around 10 sec. and 110 sec., respectively. A throughput degraded period consists of sleeping periods and awoken periods. The average duration for them is 110 msec and 320 msec,

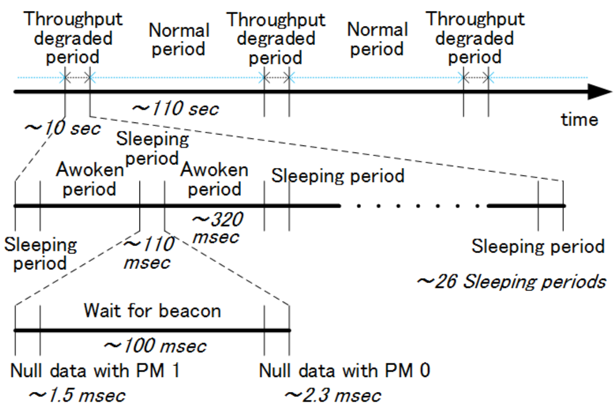

Fig. 5. Detailed analysis of time periods.

respectively. As described in Fig. 4 before, a sleeping period consists of a period sending Null data frames with the Power Management field set to 1, a period waiting for a beacon frame, and a period sending Null data frames with the Power Management field set to 0. The average durations for the individual periods are shown in Fig. 5.

On the other hand, Fig. 3 shows that the increase of cwnd is suppressed even if there are no packet losses. The reason is considered to be the collaboration of the TCP small queues and CWV. As described before, CWV intends to be used when a TCP communication is application limited. In the case the TCP small queues are used, however, it is possible that the data transfer stops when the buffered data in the sending queue for WLAN device exceeds the predefined queue limit, even if the flight size is smaller than cwnd. In this case, the MAC level data rate dominates the throughput instead of cwnd in the TCP level, and therefore, the control by CWV becomes effective. Actually, the source program of the TCP small queues implements a procedure such that, when the unacknowledged data amount is smaller than the current value of cwnd in the slow start phase, cwnd is not incremented even if a new ACK segment arrives. Note that this procedure itself is not conforming to RFC 2861 strictly but similar to RFC 7661, which was not standardized when the TCP small queues were introduced.

B. Results when TCP small queues are not used

Fig. 6 shows the time variation of TCP throughput and cwnd in the case that the TCP small queues are not used in STA1. In this case, the throughput degradations also occur periodically. In the normal periods, the average throughput is 174 Mbps, which is 38 Mbps higher than the case using the TCP small queues. This result seems to come from the fact that the cwnd value goes up to 970 packets. On the other hand, in the throughput degraded periods, the

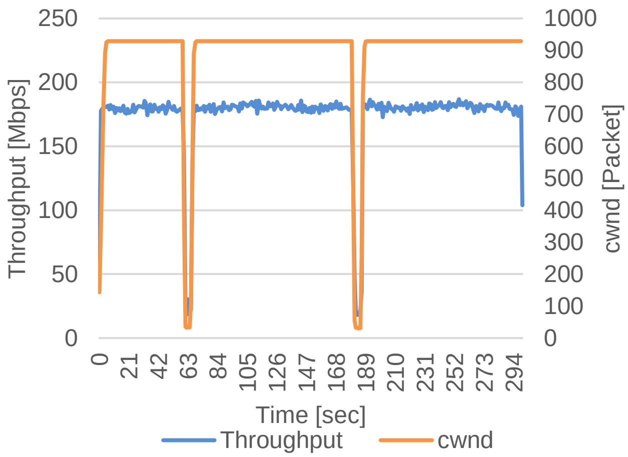

Fig. 6. TCP throughput and cwnd vs. time in uploading data transfer without TCP small queues.

throughput decreases as low as 10 % of that in the normal period. The value of cwnd decreases largely in the throughput degraded periods.

The reason for the periodic throughput degradation is also the periodic access point scanning using Null data frames with the power management function. In this case, however, there are some differences compared with the case of the TCP small queues. At first, the value of cwnd in a normal period is large. The reason is that there are no packet losses in this period and ACK segments increase cwnd because the TCP communication is cwnd limited. It should be noted that the large cwnd results in a lot of packets being stored in the sending queue at STA1. It is considered that these queued packets invoke packet losses and, as a result, cwnd decreases largely in a throughput degraded period. This is the second difference. In turn, the decrease of cwnd reduces the throughput in this period largely.

V. Analysis of Downloading TCP Data Transfer

In the experiments for downloading TCP data transfer, the results were not different depending on the use or non-use of TCP small queues. This is because neither the TCP small queues nor CWV are implemented at the access point. Instead, the results slightly depended on the TCP congestion control algorithms in the server. This section shows these results.

A. *Results when CUBIC TCP is used*

Fig. 7 shows the time variation of TCP throughput and cwnd in the case that CUBIC TCP is used at the server. From this figure, we can say that the periodic throughput degradation also occurs at the downloading TCP data transfer. By analyzing the monitoring results of WLAN frames captured by STA2, we confirmed that there are periodic exchanges of Null data frames and beacon frames between STA1 and the access point, which is similar with the sequence in the uploading TCP data transfer. So, in the case of downloading TCP data transfer, the access point scanning by WLAN stations reduces the throughput.

As shown in the figure, cwnd at the server takes the value between 350 and 500 packets. The drops of cwnd indicate that packet losses occur frequently. The increase of cwnd takes a cubic curve of time, which is characteristic for CUBIC TCP.

In contrary to the upload results, the throughput in a throughput degraded period drops sharply, although the cwnd value does not decrease largely during this period. In addition, the throughput just after a throughput degraded period is rather low. In order to investigate those results, we checked the number of A-MPDUs sent in one second by the access point, and the number of MPDUs contained in one A-MPDU. The results are given in Fig. 8. This figure shows that both A-MPDUs and MPDUs per A-MPDU decrease largely in throughput degraded periods. This result accounts for the throughput reduction. Besides that, the number of MPDUs aggregated in an A-MPDU is low just after a throughput degraded period. This is considered the reason for low throughput in this time frame.

B. *Results when TCP NewReno is used*

Fig. 9 shows the time variation of TCP throughput and cwnd in the case that TCP NewReno is used at the server. Fig. 10 shows the time variation of the number of A-MPDUs sent in one second by the access point, and the number of MPDUs contained in one A-MPDU. From Fig. 9, it is

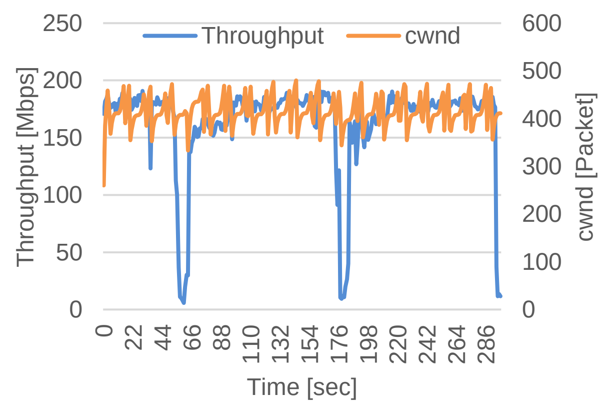

Fig. 7. TCP throughput and cwnd vs. time in downloading data transfer using CUBIC TCP at server.

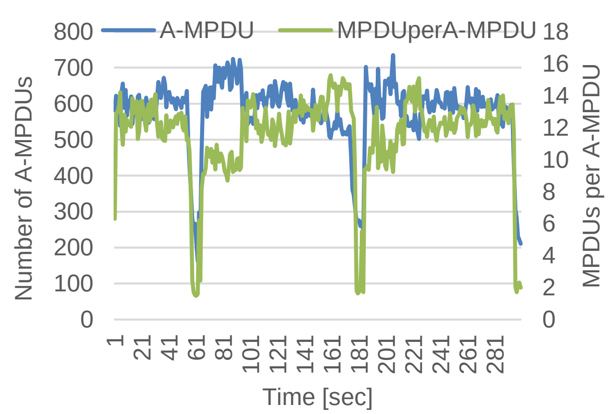

Fig. 8. Number of A-MPTU and number of MPTUs in A-MPDU vs. time in downloading data transfer using CUBIC TCP at server.

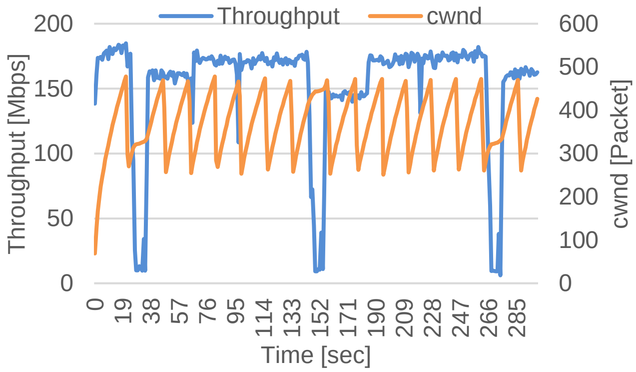

Fig. 9. TCP throughput and cwnd vs. time in downloading data transfer using TCP NewReno at server.

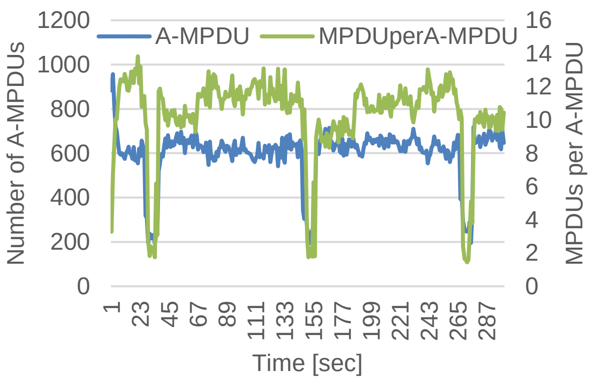

Fig. 10. Number of A-MPTU and number of MPTUs in A-MPDU vs. time in downloading data transfer using TCP NewReno at server.

confirmed that the throughput is degraded sharply in every 120 sec. From the monitoring results of WLAN frames captured by STA2, we also confirmed the periodic exchanges of Null data frames and beacon frames between STA1 and the access point. This is an access point scanning by STA1 and the reason for the throughput reduction.

This figure also shows that cwnd at the server takes the value between 300 and 500 packets, and that cwnd drops frequently, similarly with the case of CUBIC TCP. The increase of cwnd takes a linear curve along with time, which is characteristic for TCP NewReno.

While the throughput in a throughput degraded period drops sharply, the cwnd value does not decrease largely. The throughput just after a throughput degraded period is rather low. These are similar with the case of CUBIC TCP. As Fig. 10 shows, the numbers of transmitted A-MPDUs and MPDUs per A-MPDU decrease largely in throughput degraded periods. Besides that, the number of MPDUs aggregated in an A-MPDU is low just after a throughput degraded period. This will be the reason for low throughput in this time frame. These results are similar with the case of CUBIC TCP.

VI. CONCLUSIONS

This paper discussed the results on performance evaluation on the periodic TCP throughput degradation in IEEE 802.11n WLAN. The degradation is invoked by the periodic access point scanning using Null data frames with the Power Management field set to on and off. We showed that the throughput degradation is different depending on whether the TCP small queues are used or not in an uploading TCP data transfer, and what type of TCP congestion control algorithms are used in a downloading TCP data transfer. In the uploading data transfer, the TCP small queues and the congestion window validation suppress both of the increase of congestion window size in a normal period and its decrease in a throughput degradation period. So, the throughput degradation invoked by the access point scanning is smaller than the case when the TCP small queues are not used. In the downloading data transfer, the congestion control algorithms give some impacts on the time variation of congestion window size. However, the actual reason for the throughput degradation is the decrease in the transmission rate of A-MPDUs and the number of MPDUs in one A-MPDU, which are observed at an access point.

REFERENCES

[1] IEEE Standard for Information technology: Local and metropolitan area networks Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 2012.

[2] K. Nichols and V. Jacobson, “Controlling Queue Delay,” ACM Queue, Networks, vol.10, no.5, pp. 1-15, May 2012.

[3] Eric Dumazet, “[PATCHv2 net-next] tcp: TCP Small Queues”,http://article.gmane.org/gmane.network.routing.codel/68, 2012, retrieved Sept. 2016. .

[4] J. Gettys and K. Nichols, “Bufferbloat: Dark Buffers in the Internet,” ACM Queue, Virtualization, vol. 9, no.11, pp. 1-15, Nov. 2011.

[5] M. Nomoto, T. Kato, C. Wu, and S. Ohzahata, “Resolving Bufferbloat Problem in 802.11n WLAN by Weakening MAC Loss Recovery for TCP Stream,” Proc. PDCN 2014, pp.293-300, Feb. 2014.

[6] Y. Hashimoto, M. Nomoto, C. Wu, S. Ohzahata, and T. Kato, “Experimental Analysis on Performance Anomaly for Download Data Transfer at IEEE 802.11n Wireless LAN,” Proc. ICN 2016, pp.22-27, Feb. 2016.

[7] My80211.com, “802.11: Null Data Frames,” http://www.my80211.com/home/2009/12/5/80211-null-data-frames.html, Dec. 2009, retrieved Jun. 2016. .

[8] ath9k-devel@lists.ath9k.org, “disable dynamic power save in AR9280,” http://comments.gmane.org/gmane.linux.drivers.ath9k.devel/5199, Jan. 2011, retrieved Jun. 2016. .

[9] W. Gu, Z. Yang, D. Xuan, and W. Jia, “Null Data Frame: A Double-Edged Sword in IEEE 802.11 WLANs,” IEEE Trans. Parallel & Distributed Systems, vol. 21, no. 7, pp.897-910, Jul. 2010.

[10] N, Handley, J. Padhye, and S. Floyd, “TCP Congestion Window Validation,” IETF RFC 2861, Jun. 2000.

[11] G. Fairhurst, A. Sathiaseelan, and R. Secchi, “Updating TCP to Support Rate-Limited Traffic,” IETF RFC 7661, Oct. 2015.

[12] iperf, http://iperf.sourceforge.net/, retrieved Jun. 2016.

[13] S. Ha, I. Rhee, and L. Xu, “CUBIC: A New TCP-Friendly High-Speed TCP Variant,” ACM SIGOPS Operating Systems Review, vol. 42, no. 5, pp. 64-74, July 2008.

[14] S. Floyd, T. Henderson, and A. Gurtov, “The NewReno Modification to TCP’s Fast Recovery Algorithm,” IETF RFC 3728, April 2004.

[15] ath9k Linux Wireless, http://sireless.kernel.org/enusers/Drivers/ath9k, retrieved Jun. 2016.

[16] Linux foundation: tcpprobe, http://www.linuxfoundation.org /collaborate/workgroups/networking/ tcpprobe, retrieved Jun. 2016.