QoS-aware Adaptive MPDU Aggregation of VoIP Traffic on IEEE 802.11n WLANs

Shinnazar Seytnazarov and Young-Tak Kim
Department of Information and Communication Engineering,
Graduate School, Yeungnam University
Gyeongsan-si, Kyungbuk, 712-749, KOREA
seytnazarovsho@ynu.ac.kr, ytkim@yu.ac.kr

Abstract— MPDU aggregation is not applied to real-time voice traffic (e.g., VoIP) in currently available IEEE 802.11n WLAN implementations considering the strict upper bounds of end-to-end delay and jitter. As a result, when real-time voice and non-real-time heavy load traffics are intermixed the resource utilization and overall throughput become poor due to the overhead produced by individual voice MPDU transmissions. In this paper, we propose QoS-aware Adaptive MPDU aggregation scheduler which applies aggregation to voice traffic and adaptively adjusts the aggregation size based on time-varying end-to-end delay and WLAN contention, and QoS requirements (e.g. less than 150 ms end-to-end delay). Experimental results of the proposed scheme show that the overall throughput increased by 57% when 10 stations generate 64 Kbps PCM voice traffic on 270 Mbps PHY rate for 2 ms transit delay representing campus network communication. For 120 ms transit delay configuration that represents intercity communication, the throughput enhancement was 45 % compared to the existing scheme[1].

Keywords— IEEE 802.11n, frame aggregation, aggregation scheduler, QoS.

I. INTRODUCTION

Currently, most mobile portable Internet devices (such as smartphones, tablet PCs and notebooks) are using Wi-Fi for broadband mobile Internet accesses for various real-time multimedia applications such as Voice over IP (VoIP), Video over IP, and online games [1-3].

The IEEE 802.11n high throughput WLAN employs MPDU (MAC protocol data unit) aggregations to solve the MAC inefficiency problem at high PHY rates of up to 600 Mbps [4-6]. In currently available IEEE 802.11n WLAN implementations (such as ath9k driver [7]), however, the MPDU aggregation is not applied to real-time voice traffic (e.g., VoIP) considering the strict upper bounds of end-to-end delay and jitter. In result, when real-time voice and other heavy load traffics are intermixed, the network resource utilization and overall throughput degrades significantly because of the overhead introduced by individual voice MPDU transmissions.

In this paper, we propose a QoS-aware Adaptive MPDU aggregation scheduler for IEEE 802.11n WLANs. The scheduler applies MPDU aggregation scheme to voice traffic and maximizes the aggregation size (i.e. number of MPDUs) based on time-varying end-to-end delay and WLAN contention measurement statistics, and the QoS requirements (i.e., less than 150 ms end-to-end delay for voice MPDUs).

[1]This research work was supported by the MSIP (Ministry of Science, ICT & Future Planning), Korea, under IT/SW Creative research program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2013-H0502-13-1085), and in part by the MSIP, under IT/SW Creative research program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2013-H0502-13-1085); the MSIP /Innopolis, Korea, under 2013 Innopolis Daegu R&BD program.

Performance evaluations of proposed and existing scheduler schemes were done for different transit delay configurations on real-world test-bed. When the transit delay was set to 140 ms and each of 10 STAs (station) generated uplink 64 Kbps VoIP and saturated best-effort traffics at 270 Mbps PHY rate, the proposed scheme improved the network throughput by 45%. When the transit delay was decreased up to 2 ms representing campus network communication, the throughput improvement was 57% compared to the existing scheme. In all scenarios the proposed scheme provided less than 150 ms end-to-end delay and 5 ms jitter for voice traffic.

The rest of this paper is organized as follows. Section II briefly reviews the frame aggregation mechanisms in IEEE 802.11n standard and related work. Section III explains the proposed QoS-aware Adaptive MPDU aggregation in detail. Section IV is dedicated to the performance analysis. Finally section V concludes this paper.

II. RELATED WORK

A. Existing Frame Aggregaton Scheduler Schemes

IEEE 802.11n standard does not define any specific scheduling mechanism for frame aggregations, and it was remained to be implemented based on the preference of vendors. For example, *ath9k* driver specifies two types of scheduling algorithms: one for voice AC (access category) and second for remaining ACs (i.e., video, background and best-effort) [7]. Scheduler of voice AC does not use aggregation mechanisms i.e., voice MPDUs are transmitted individually as in legacy 802.11a/b/g.

Unlike voice scheduler, the scheduler of non-voice ACs employs MPDU aggregation mechanism. A-MPDUs can include up to 32 MPDUs. It uses SwQueue (software queue) where MPDUs will be waiting before aggregation and HwQueue (hardware queue) where up to two frames can wait for transmissions; frame can be either an A-MPDU or MPDU. The scheduler queu1es MPDU into SwQueue if any one of the following conditions is true:

- If SwQueue has MPDUs, i.e., SwQueue length is greater than 0.
- If there are already two frames in HwQueue, i.e., HwQueue does not have empty space.
- If incoming MPDU is out of BAW (Block ACK Window).

If none of above conditions is true the incoming MPDU will be queued directly into HwQueue without aggregation. This scheduler scheme can perform very well if the SwQueue is always saturated with traffic, so it can always make long A-MPDUs consequently increasing the resource utilization

efficiency and network throughput. But under light load traffics it frequently transmits individual MPDUs without aggregation. More details of *ath9k* schedulers can be found in [3, 7].

There are number of other scheduler proposals for 802.11n frame aggregations. For example, Selvam et al. in [8] proposed a scheduler mechanism that dynamically adapts the aggregated frame size and aggregation mechanism based on different parameters. Kim et al. in [9] proposed adaptive two-level frame aggregation scheduler schemes that dynamically adapt aggregation type based on MPDU delimiter parameter. In [10], Hajlaoui et al. made similar proposal as in [8] but for multimedia applications; the work proposes some parameters for making the aggregations effectively but the values of the parameters are not defined thus the proposed mechanism is not clear.

III. QoS-Aware Adaptive MPDU Aggregation Scheduler For IEEE 802.11n Networks

A. Cross-Layer Optimization of RTP and IEEE 802.11n MAC layers for QoS-aware Adaptive MPDU Aggregation

Real-time applications such as VoIP usually use RTP protocol. We extended RTP in a way that it reports every second end-to-end delay statistics from destination to source. The RTP at source, after each reception of end-to-end delay report, immediately passes it to aggregation scheduler in 802.11n MAC layer. The scheduler further employs it for making QoS-aware adaptive MPDU aggregations leading to improved resource utilization and throughput. IPERF software was modified to support the extended RTP features [11]. To synchronize the source and destination clocks, we used PTPd over Ethernet in order to get microsecond precision [12].

B. QoS-aware Adaptive MPDU Aggregation Scheduler

The QoS-aware adaptive MPDU aggregation scheduler adaptively adjusts the A-MPDU size for each Access category (AC) based on end-to-end delay requirements and time-varying transit delay and WLAN contention levels. For that purpose, it configures end-to-end delay thresholds ($T_{e2e}[AC_i]$) for each AC based on their QoS requirements (TABLE I).

In proposed scheduler shown in Fig. 1, unlike the reference scheduler in ath9k, any newly arrived MPDU of access category *i* (ACi) from upper layer will be queued into SwQueue[ACi] regardless of current status of HwQueue[ACi]. The earlier arrived MPDUs will wait for other incoming MPDUs in SwQueue. Waiting time of the earliest arrived MPDU in SwQueue[AC_i] is defined as software delay (D_{sw}).

TABLE I. End-to-End Delay Threshold and Jitter Values for Different Access Categories

Access Category	End-to-end delay thresholds ($T_{e2e}[AC_i]$), (ms)
Voice	150
Video	150
Best-effort	1000
Background	1000

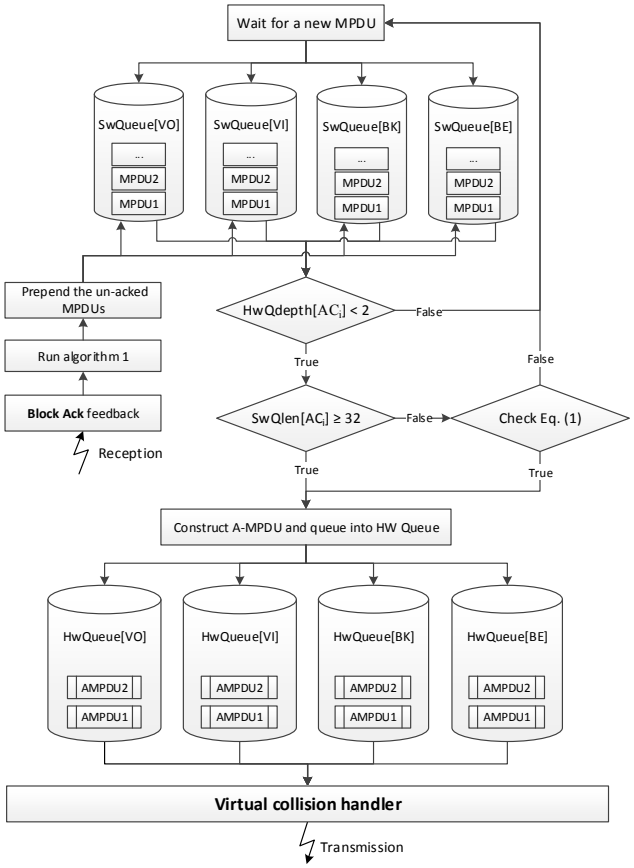

Fig. 1. QoS-aware Adaptive MPDU aggregation scheduler

Whenever HwQueue[AC_i] has empty space (i.e., HwQdepth[AC_i] < 2) and SwQueue[AC_i] is not empty (i.e., SwQlen[AC_i] > 0) the scheduler checks if the SwQueue[AC_i] has already more than 31 MPDUs i.e., SwQlen[AC_i] ≥ 32; if so, the scheduler constructs A-MPDU and queues it into HwQueue[AC_i] since SwQueue[AC_i] already has enough number of MPDUs that is allowed to include in one A-MPDU; otherwise the scheduler further checks if the inequality given in Eq. (1). If it is true then the scheduler constructs A-MPDU out of the existing MPDUs in SwQueue[AC_i] and queues it into HwQueue[AC_i] for further transmission; otherwise, SwQueue[AC_i] will start waiting for more MPDUs.

$$E_{e2e} + T_{arr} > T_{e2e} \quad (1)$$

Inequality given in Eq. (1) is needed to determine whether the scheduler should wait for the next incoming MPDU from upper layer in order to make longer A-MPDU or compose A-MPDU out of existing MPDUs in order to provide guaranteed QoS (e.g., less than 150 ms for voice MPDUs as depicted in TABLE I). Now we discuss about each parameter involved in Eq. (1):

- $T_{arr}[AC_i]$ represents the MPDU inter-arrival period from upper layer to SwQueue[AC_i], it cn be easily obtained as a difference between the arrival times of sequentially arrived MPDUs in SwQueue[AC_i]; for instance, it is

always equal to 10 ms for 64 Kbps VoIP application with packet length of 80 bytes.

- $E_{e2e}[AC_i]$ is an expected end-to-end delay for the first MPDU in SwQueue[AC_i] which is calculated by Eq. (2) where $D_{sw}[AC_i]$ is current software delay of the first MPDU in SwQueue[AC_i], $D_{avg\_hw}[AC_i]$ is an average hardware delay that each A-MPDU faces in HwQueue before the successful transmission starts, $T_{tx}[AC_i]$ is transmission duration of A-MPDU that can be constructed out of current pending MPDUs in SwQueue[AC_i] and $D_{tr}[AC_i]$ is transit delay between AP and destination which is updated every second by Algorithm 1.

$$E_{e2e}[AC_i] = D_{sw}[AC_i] + D_{avg\_hw}[AC_i] + T_{tx}[AC_i] + D_{tr}[AC_i] \quad (2)$$

- $T_{e2e}[AC_i]$ represents the end-to-end delay threshold of MPDUs that belong to AC_i (TABLE I).

Number of STAs and amount of offered loads at ACs of each STA are time varying parameters of WLAN network and they have great impact on WLAN contention. In order to have knowledge about time-varying contention in WLAN, the scheduler keeps the statistics on hardware delay ($D_{hw}[AC_i]$) and average hardware delay ($D_{avg\_hw}[AC_i]$). Hardware delay represents the delay that certain A-MPDU faces before its successful transmission starts; usually it is referred as *access delay* or *medium access delay* in most works available in literature but we use hardware delay notation since it's the delay happens in HwQueue. Whenever the STA receives Block ACK for the last successfully transmitted A-MPDU, the scheduler runs Algorithm 1 which does following calculations and updates:

- The scheduler calculates $D_{hw}[AC_i]$ of A-MPDU. It can be easily calculated as the difference between the Block ACK reception time (t_{BACK}) and the sum of the time when A-MPDU was queued into HwQueue[AC_i] ($t_{hwq}[AC_i]$), time periods required to transmit A-MPDU ($T_{ampdu}[AC_i]$) and Block ACK (T_{BACK}), and SIFS duration (T_{SIFS}).
- Next it accumulates $D_{hw}[AC_i]$ to $S_{hw}[AC_i]$, $D_{sw}[AC_i]$ of the first MPDU in transmitted A-MPDU to $S_{sw}[AC_i]$ and $T_{tx}[AC_i]$ of transmitted A-MPDU to $S_{tx}[AC_i]$. $S_{hw}[AC_i]$, $S_{sw}[AC_i]$ and $S_{tx}[AC_i]$ represent the sum of hardware delay, software delay and transmission periods for the last second, respectively.
- Then the scheduler increments the number of A-MPDUs successfully transmitted, $S_{ampdu}[AC_i]$.
- Once 1 second passes since last update, the scheduler utilizes $S_{hw}[AC_i]$ and $S_{ampdu}[AC_i]$ to calculate average hardware delay ($D_{avg\_hw}[AC_i]$) using the Exponential moving average (EWMA) level of 25% where the latest statistics have reasonable impact on new $D_{avg\_hw}[AC_i]$ update.
- Finally the scheduler updates the transit delay $D_{tr}[AC_i]$ as the time difference between end-to-end delay $D_{e2e}[AC_i]$ that is obtained from RTP layer each second and average of the sum of $S_{hw}[AC_i]$, $S_{sw}[AC_i]$ and $S_{tx}[AC_i]$. Updated $D_{tr}[AC_i]$ will be used in calculating expected end-to-end delay $E_{e2e}[AC_i]$ given in Eq. (2) during next second.

Algorithm 1. Algorithm to update average hardware delay

```
1:  proc update_avg_hw_delay()
2:      Dhw[ACi] = tBACK – thwq[ACi] – Ttx[ACi] – TSIFS – TBACK
3:      Shw[ACi] += Dhw[ACi]
4:      Ssw[ACi] += Dsw[ACi]
5:      Stx[ACi] += Ttx[ACi]
6:      Sampdu[ACi] += 1
7:      if (current_time( ) – last_update[ACi] ≥ 1 sec) then
8:          Davg_hw[ACi] = Davg_hw[ACi] · (1 - EWMA) +
            Shw[ACi] · EWMA / Sampdu[ACi]
9:          Dtr[ACi] = De2e[ACi] - (Shw[ACi] + Ssw[ACi] + Stx[ACi])
            / Sampdu[ACi]
10:         Shw[ACi] = Ssw[ACi] = Stx[ACi] = Sampdu[ACi] = 0
11:         last_update[ACi] = current_time( )
12:     end if
13: end proc
```

IV. Performance Evaluations

A. Experimental Test-bed Configuration

Experiments were conducted on TL-WDN4800 wireless cards with Atheros AR9380 chipsets. This card can operate in both 2.4 and 5 GHz frequency bands and supports up to 3×3 MIMO configuration, 20/40 MHz channel width, short and long guard intervals (SGI/LGI).

In the experiments, we used up to 10 STAs each of them generating uplink VoIP and best-effort traffics to destination residing at AP. VoIP packets are generated with the rate of 64 Kbps with 80 bytes payload length. Best-effort packets have size of 1400 bytes and SwQueue was always kept full of MPDUs so HwQueue always has 2 A-MPDUs with maximum size of 32. The series of experiments were conducted for different transit delays and number of STAs. RTP/IPERF at destination was configured to report different end-to-end delay statistics that include transit delays varying from 2 to 120 ms emulating campus and intercity communications. The reason of such configuration of best-effort traffic generation is that we want to demonstrate the improvement of maximum network throughput when the QoS-aware Adaptive MPDU aggregation scheduler is used for intermixed real-time voice and best-effort traffics. The transmission rate is fixed to 270 Mbps and channel is ideal. IPERF was used as a traffic generation tool [11].

B. Performance Comparison of Existing and Proposed Schedulers for 10 STAs at Different Transit Delays

TABLE II depicts the performance comparison between *ath9k* and the proposed QoS-aware Adaptive MPDU aggregation schedulers where 10 STAs generate uplink VoIP and best-effort traffics to the destination with different transit delays. When the transit delay decreases from 120 to 2 ms, the throughput improvement of proposed scheduler increases from 45 to 57.3%. The reason of such improvement is that the proposed scheduler decreases the overhead of each voice MPDU since it applies the MPDU aggregation and aggregates as many MPDUs as possible up to the QoS thresholds of end-to-end delay. Aggregation sizes of voice A-MPDUs are 15, 11, 7 and 3 when transit delays are equal to 2, 40, 80 and 120 ms, respectively.

The average voice end-to-end delay of the proposed and the existing scheduler schemes vary between 142~145 ms and 5~123 ms, respectively. The jitter stays almost unchanged in both cases and equal to around 1.5 and 2.1 ms for the proposed and the existing schedulers, respectively.

TABLE II. PERFORMANCE COMPARISON OF EXISTING AND PROPOSED SCHEDULERS FOR 10 STAS AT DIFFERENT TRANSIT DELAYS

Transit delay (ms)		2	40	80	120
Throughput (Mbps)	Proposed	167.51	165.93	164.35	154.34
	Existing	106.49	106.49	106.49	106.49
End-to-end delay (ms)	Proposed	144.90	142.65	142.81	142.37
	Existing	4.92	42.92	82.92	122.92
Jitter (ms)	Proposed	1.55	1.47	1.59	1.50
	Existing	2.10	2.10	2.10	2.10
A-MPDU size	Proposed	15.00	11.00	7.00	3.00
	Existing	1.00	1.00	1.00	1.00

C. *Performance Comparison of Existing and Proposed Scheduler Schemes for 1-10 STAs*

Since the existing scheduler does not apply aggregation to voice traffic, when the number of STAs in 802.11n network increases, the throughput decreases significantly because of the increased contention and overhead produced by individual voice MPDU transmissions (Fig. 2). The proposed scheme, however, decreases the voice traffic overhead by applying the QoS-aware adaptive MPDU aggregation; the improvement becomes higher if the transit delay decreases. When transit delay is 120 ms, the throughput improvement is 0.69, 11.2 and 45% over existing scheduler when number of STAs is 1, 5 and 10 STAs, respectively. If transit delay decreases to 2 ms, the improvement increases up to 0.82, 16.6 and 57.3 % for 1, 5 and 10 STAs, respectively.

In the case of 120 ms transit delay, the average end-to-end delay of proposed scheduler increases from 141.1 to 142.3 ms as the number of STAs increases from 1 to 10; while the existing scheme increases the average end-to-end delay from 121 to 122.9ms. When the transit delay is 2ms, as the number of STAs increases from 1 to 10 the average end-to-end delay increases from 143.2 to 144.9 ms in the proposed schemes while it is in the range of 3 to 4.9ms in existing scheme, respectively (Fig. 3).

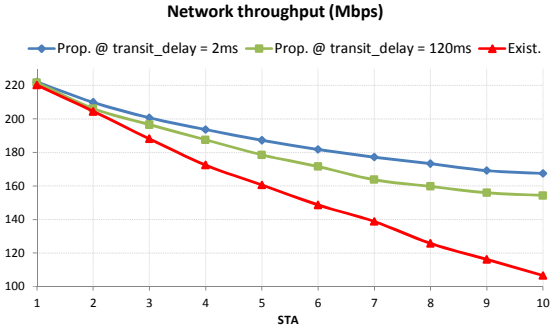

Fig. 2. Comparison of average network throughput at 270 Mbps PHY rate, 2 ms / 120 ms transit delays, and 1 ~ 10 STAs

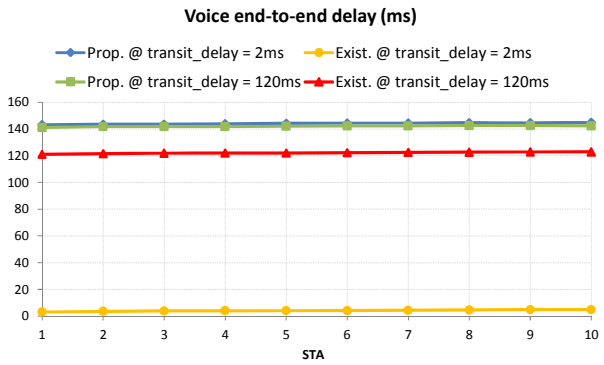

Fig. 3. Comparison of end-to-end delay at 270 Mbps PHY rate, 2 ms / 120 ms transit delays, and 1 ~ 10 STAs

V. CONCLUSION

In this paper, we propose a QoS-aware Adaptive MPDU aggregation scheduler that adaptively adjusts the aggregation size of voice traffic based on time-varying end-to-end delay and WLAN contention levels, and QoS requirements. Experimental results show that the proposed scheme increased the overall throughput by 57% when 10 STAs generate 64 Kbps PCM voice traffic on 270 Mbps PHY rate in intra-campus communications with less than 5 ms transit delay. For inter-city communications at 270 Mbps transmission rate with 80 ~ 120 ms transit delay, the throughput enhancements are 45 ~ 54% compared to the existing scheme.

VI. REFERENCES

[1] H. Hwang and Y. T. Kim, "QoS-aware Fast BSS Transitions for Seamless Mobile Services and Load Balancing," in Proc. of International Conference on Consumer Electronics 2014 (ICCE2014), Las Vegas, USA, January 2014.

[2] Sh. Seytnazarov and Y. T. Kim, "Adaptive Rate Adaptation for High Throughput IEEE 802.11n WLANs," in Proc. of Asia Pacific Network Operations and Management Symposium 2013 (APNOMS2013), Hiroshima Japan, September 2013.

[3] Sh. Seytnazarov and Y. T. Kim, "QoS-aware MPDU Aggregation of IEEE 802.11n WLANs for VoIP Services," in Proc. of EUROPMENT Conference 2014, Prague, Czech Repulic, April 2-4, 2014.

[4] IEEE 802.11n, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Enhancements for Higher Throughput, 2009.

[5] Y. Xiao, "IEEE 802.11n: Enhancements for Higher Throughput in Wireless LANs," IEEE Wireless Comm. Magazine, vol. 12, no. 6, pp. 82-91, Dec. 2005.

[6] Y. Xiao and J. Rosdahl, "Throughput and delay limits of IEEE 802.11," IEEE Commun. Lett., vol. 6, no. 8, pp. 355–357, Aug. 2002.

[7] Ath9k driver - http://linuxwireless.org/en/users/Drivers/ath9k.

[8] T. Selvam and S. Srikanth, "A frame aggregation scheduler for IEEE 802.11n," National Conference on Communications (NCC) 2010, Issue Date: 29-31 Jan. 2010.

[9] Y. Kim, E. Monroy, O. Lee, K. Park and S. Choi, "Adaptive two-level frame aggregation in IEEE 802.11n WLAN," 18th Asia-Pacific Conference on Communications (APCC), 2012.

[10] N. Hajlaoui, I. Jabri, M. Taieb and M. Benjemaa, "A frame aggregation scheduler for QoS-sensitive applications in IEEE 802.11n WLANs," International Conference on Communications and Information Technology (ICCIT) 2012, Issue Date: 26-28 June 2012.

[11] https://code.google.com/p/iperf/.

[12] http://ptpd.sourceforge.net/.