Adaptive Delayed Channel Access for IEEE 802.11n WLANs

Dionysios Skordoulis[1], Qiang Ni[1], Geyong Min[2] and Kevin Borg[1]
[1] Electronic & Computer Engineering Division, School of Engineering and Design, Brunel University, West London, U.K.
[2] Department of Computing, School of Informatics, University of Bradford, Bradford, U.K.
Email: {Dionysios.Skordoulis, Qiang.Ni}@brunel.ac.uk, g.min@brad.ac.uk

Abstract— **In this paper we investigate potential benefits that an adaptive delayed channel access algorithm can attain for the next-generation wireless LANs, the IEEE 802.11n. We show that the performance of frame aggregation introduced by the 802.11n adheres due to the priority mechanism of the legacy 802.11e EDCA scheduler, resulting in a poor overall performance. Because high priority flows have low channel utilization, the low priority flows throughputs can be amerced further. By introducing an additional delay at the MAC layer, before the channel access scheduling, it will retain aggregate sizes at higher numbers and consequently a better channel utilization. Also, in order to support both UDP and TCP transport layer protocols, the algorithm's operational conditions are kept adaptive. The simulation results demonstrate that our proposed adaptive delayed channel access outperforms significantly the current 802.11n specification and non-adaptive delayed channel access.**

Adaptive delayed channel access; frame aggregation; IEEE 802.11n; medium access control.

I. Introduction

Nowadays, the most dominant wireless network is the IEEE 802.11 wireless local area network (WLAN) [1] with impressive growing support across the enterprises, the public sector, homes and many data service providers. Over the past years, the research and development communities have narrow down their studies within innovations that can offer mainly higher throughput and performance, better reliability and robustness, but also in protocols that allow the devices to provide sufficient Quality of Service (QoS) for either the applications or clients. Since the current standards and further resolutions are bounded from a theoretical throughput limitation [2], the IEEE 802.11 working group established in September 2003 a Task Group (TG), known as TGn ('n' stands for next-generation), in order to compose a High Throughput (HT) amendment. Their main aim is to achieve higher data rates of at least 100 Mbps as measured at the MAC service access point and at the same time provide co-existence with previous amendments.

The latest TGn draft document includes various pioneering PHY and MAC enhancements, such as MIMO and frame aggregation, respectively [3]. The later is considered as a major contribution for reaching high data rate targets since it consents to mitigate transmission overheads, namely, backoffs prior to accessing the shared channel, physical layer preamble, by concatenating multiple data units into a single frame [4]. Additionally, the asynchronous data service is handled by IEEE 802.11e's mandatory coordination function, the Enhanced Distributed Channel Access (EDCA) [5]. EDCA defines a set of QoS mechanisms, where delay-sensitive applications can be concerned with high importance and preeminent low priority flows. However, since a station (STA) with high priority traffic defers, on average, for less period than a STA with low priority traffic, the number of data packets assigned in each aggregated frame turns out low too, thus more overhead is required. This abominable consequence was firstly described in [6] where the authors review the poor channel utilization which consequently reduces in overall the network's throughput and QoS performance. A delayed channel access (DCA) algorithm was initially proposed that it impels STAs into further deferring in a way that it allows throughout more packets to arrive and it results to the end aggregate size to accumulate. Although this work is very interesting, Transmission Control Protocol (TCP) flows with various TCP window sizes, wasn't considered during the evaluation and as we explain in this paper, these conditions can result in aggravate and negative behavior. This paper exposes the impact that DCA applies over TCP performance and proposes a solution with an enhanced MAC based algorithm with conditional triggers that will be regularly adapted over the progressive traffic status. Our enhanced algorithm, named as Adaptive DCA (ADCA), can be applied on future 802.11n device and can improve channel efficiency that will increase the network's overall performance.

The remainder of this paper is organized as follows: Section II introduces the poor channel utilization that high priority applications have over HT networks and an overview of the related work on delayed channel access algorithms. In Section III we analyze the consequences of further deferring over TCP traffic and how this can be avoided by adapting the triggering thresholds. Our ADCA algorithm is validated using extended simulations in Section IV where in most cases outperforms the current TGn specification. Finally, Section V we conclude the paper by pointing out the importance of our findings.

II. Overview Of Delayed Channel Access

A. 802.11e and 802.11n in conjuction

As we mentioned earlier, TGn's latest draft standard builds upon 802.11e's probabilistic priority mechanisms along with other MAC enhancements, such as frame aggregation. In the specification there are two types of frame aggregation suggested that reduces significantly the MAC and PHY overhead, Aggregated MAC Service Unit (A-MSDU) and Aggregated MAC Data Unit (A-MPDU). Overhead is defined as the length of time that the wireless medium is engaged for each header, control frame, or interframe space (IFS) period, which are adapted for the transmission or reception of each data payload. The analysis in [2] has shown that as a result of the overhead, the maximum ideal throughput is bounded by a maximum relative MAC throughput which is less than 50% of the average peak PHY. The principle of the A-MSDU is to allow several MSDUs being sent to the same receiver concatenated in a single MPDU, where in A-MPDU aggregation is to joint multiple MPDU subframes in order to use a single PHY header. Both choices are adequate, in their own manner, to extensively improve the channel efficiency and the data throughput with the condition that there are enough data units bided in the buffered queues [4].

Apart from the traffic load, where a high offered load from the application will signify a big pile in the MAC stack, we need to investigate the operation of EDCA on each prioritized flow. Within this QoS mechanism, there are separate access categories (ACs) which each has a separate queue buffer and an analogous channel access waiting time depending on the AC's importance. So, higher priority categories can acquire channel access faster than the lower priority as a set of distinct parameters are assigned: Arbitrary IFS (AIFS) and a pair of min/max values for the Contention Window (CW). Although this situation can induce unfairness to the lower ACs, it is the most adequate mechanism for the higher ACs to attain channel access within the delay-constraints appointed from the higher-layers [7]. But, as the waiting period is decreasing for the higher ACs, so is the number of packets within an aggregated frame (aggregate size), thus less channel efficiency.

To check the effect that EDCA mechanism has over the frame aggregation in HT networks, let us consider a simple scenario, named as Scenario 1 for future reference. We consider an overloaded 802.11n WLAN that includes three STAs and an Access Point (AP). All STAs are relative close with each other and in line of sight (LOS). Their operational PHY rate is 117 Mbps since we've set a 64-QAM modulation, a ¾ coding rate and 800 ns guard interval (see MCS parameter table for two spatial streams at 20 MHz in [3]). Also, we set two types of HDTV flows over User Datagram Protocol (UDP) with 200 ms maximum end to end delay between the AP and two of the STAs and an internet file transfer from the third STA over TCP transmitted to the AP. All MSDUs are 1500 bytes in size and the offered loads are 19.2 Mbps, 24 Mbps and 120 Mbps for the HDTVs and FTP, respectively. The scenario's model is implemented and simulated in OPNET Modeler [8] and since we examine the potentials of frame aggregation the channel is regarded as error-free.

The results that we are interested in are the goodput (the total throughput minus the packets that miss the time constraints), the average aggregate size of each transmitted frame, the average delay and the maximum delay. Both, Table 1(a) and Table 1(b) list the simulation results for the last 4 seconds of a 5 seconds run as we allow the TCP congestion window (CWND) to fully build up. In Table 1(a), the average packet number per aggregate for the video flows (UDP traffic) is 1.80 and 1.31, and for the best effort flow (TCP traffic) is 24.57. Also, we observe that the delay constraints are met in all flows but the overall MAC efficiency is 46.9% as the MAC throughput is 54.987 Mbps out of the total 117 Mbps Peak PHY rate. The later validates our earlier analysis regarding the poor interaction between EDCA and frame aggregation because of small aggregates within the high priority ACs.

B. *The idea of DCA algorithm*

The concept of implementing a delayed channel access algorithm was first introduced in [6] and it was designed to intentionally commence a further delay at the MAC layer in order to increase the number of packets that can be buffered in each AC's queue and correspondingly the end frame's aggregation size. The following equation shows the dependency of the number of packets (N) in the queue over time (t): $N(t) = \frac{t \times \rho}{L}$, where ρ and L are the mean data rate and payload size, respectively. Thus, as time t increases so is the number of packets in the frame formed by aggregation.

The channel access delay for a frame arriving at the MAC is defined as the period from the time that the frame arrives at the front of the queue buffer till its successful transmission to the intended receiving STA, excluding the wireless propagation delay. However, the additional delay may lead to unnecessarily idling or might effect the QoS experienced by the application therefore a set of conditions need to be applied so that it can match the aggregated packet formation with the traffic burst within an appropriate time scale. The initial DCA algorithm has three basic triggers [6]:

- The number of packets in the aggregation buffer has reached or exceeded an aggregated threshold size (σ).
- The maximum delay threshold (τ) of the first packet in the stack is reached or exceeded, usually less than maximal delay allowed from the originated application.
- No packets arrive at the MAC from the higher layers within a dynamically calculated threshold period (α): $\alpha = \lambda \times (T_{tr} - T_{ca})$, where lambda is a predefined factor and T_{tr} and T_{ca} the transmission and channel-access starting time for an aggregate, respectively.

TABLE 1: SIMULATION RESULTS FOR SCENARIO 1 (TCP WINDOW SIZE = 655350 B)

Name	Goodput (Mbps)	Avg. Aggregate Size	Max. Delay (sec)	Avg. Delay (sec)
HDTV	23.994	1.80	0.012666	0.001146
HDTV	19.197	1.31	0.011200	0.000997
Internet File	11.796	24.57	0.654076	0.396930

(a) 802.11n typical process

Name	Goodput (Mbps)	Avg. Aggregate Size	Max. Delay (sec)	Avg. Delay (sec)
HDTV	23.865	13.11	0.044573	0.013416
HDTV	19.116	12.21	0.044710	0.015202
Internet File	51.999	25.27	0.134162	0.088660

(b) 802.11n with DCA algorithm enabled

So, these three conditions are dictating when the additional differing shall be terminated and by followed with the channel access process as normal. The values for the σ, τ and λ attributes may be constants or dynamically adapted based on the traffic behavior.

The DCA's performance is evaluated through the previously defined scenario, Scenario 1. Table 1(b) shows the results for λ = 10, t = ½ maximal delay and σ = 48 packets. Now, both HDTV average aggregated sizes have been increased dramatically, on the point of ensuring a better channel utilization. The efficiency has been increased from 46.9% (no DCA) to 81.17% (with DCA). Also, we observe that by introducing additional delay before channel access, the end-to-end delays of both HDTV traffic flows had an insignificant increase and the maximum delays remain way below the 200 ms delay boundary. DCA doesn't override the AC's priority but it limits the frequent channel accesses from high UPs to less and more efficient. It is obvious that the DCA algorithm has increased the system's effectiveness.

III. ADAPTIVE DCA

A. *The TCP problem with DCA*

The two core communication protocols on which most networks operate are the UDP and TCP. The latter is a reliable, robust and connection-oriented method of data delivery that is commonly used over the Internet because of its flexibility to be adapted according to the network's disparate conditions. On the other hand, it is known to be very troublesome when used over wireless networks, for these reason there many variations of its implementation [9]. In every case, TCP maintains a flow control, known as congestion control, where both sender and receiver control the size (in segments) of the next transmitted information according to the link's conditions but this congestion window (CWND) can not exceed the receiver's TCP advertised window size. But, an end-to-end link may contain intermediate bottlenecks with smaller window sizes, so the sender node sets a starting CWND usually equal to a single segment, and every time it receives a positive acknowledgement (ACK) from the receiver, it increases the CWND according to the running TCP implementation.

However, during our investigation on the DCA, we've found that TCP's speed of transmission is very dependent on the delay caused by DCA, where in some cases it can bear diverge and unwanted outcomes. Figure 1 and Figure 2 show the TCP's flow goodput and maximum delay in Scenario for various TCP window sizes. Previously, while evaluating DCA the window size was not indicated, which was actually set for 655350 bytes following the recommendations in [10] where it is suggested that the maximum TCP window size should be at least as large as the bandwidth-delay product of the wireless link. However, in reality this is not always the case, so when we run the same scenario with smaller values, say 4KB, 8 KB, 16 KB, 32 KB, 64 KB etc, the TCP throughput decreases rapidly while the delay increases dramatically. This is because TCP is waiting for a number of segments that already had sent before it can carry on with the next set of segments while at the MAC layer the DCA algorithm defers as usual till one of the conditions met. But since this is best effort (BE) traffic, the τ attribute has no delay boundaries, hence no triggering from this condition. Also, as the maximum segment size (MSS) is 1500 bytes, any window sizes lower than 65355 bytes can contain less than 43 packets, so the aggregate size threshold (σ) will never be reached as is set for 48 packets. So, the only condition left is the λ condition but this increases waiting times, hence the delays and consequently the low throughputs. In conclusion, although we previously demonstrate that DCA increases the network's performance, when it comes down to TCP traffic with small window buffers there is an issue which needs to be resolved.

B. *The ADCA algorithm*

By observing Figure 1 and Figure 2, we notice that when the TCP buffer size exceeds a specific point, then DCA operation carries out gains in the overall throughput. This point has been found to be equal to 70080 bytes, which is the product of 48 by 1460 bytes (the MSS omitting the TCP/IP encapsulation headers). Considering that 48 packets was the set value for the aggregate size threshold, we justify that a quick solution for this problem would be to always appoint σ similar to the corresponding sender's maximum segments in its CWND. But the TCP and MAC are two layers transparent with each other with no shared information except the data payload (segments). On the other hand, if we try to alter the delay within DCA by changing the other triggering attributes with smaller values, in doing so the total performance drops into adjacent levels than these without the DCA operation. A good

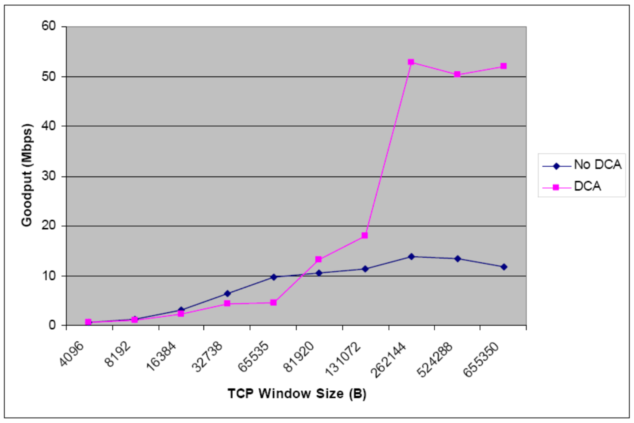

Figure 1: Goodput results for TCP flow with varying window sizes

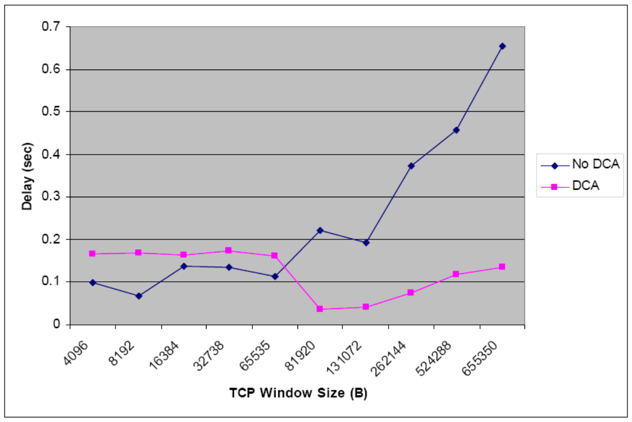

Figure 2: Max. delay results for TCP flow with varying window sizes

solution is to implement an enhanced DCA algorithm which can dynamically adapt its parameters accordingly.

Our proposal introduces an adaptive aggregate size threshold that gradually increases or decreases until the flight size (number of segments in the sender's buffer size) of the TCP flow is reached, so it will always be guaranteed that $\sigma \leq$ flight size. Since a cross-layer solution is infeasible or too complex and no information can be shared, the adaptive DCA algorithm will be totally based on its own characteristics, the two main conditions: the burst factor (λ) and the current aggregate size (σ_{cur}). These conditions will be used with a perspective to find a value where the queue is most likely to be at steady state, considering that the TCP connection doesn't have many imbalances. At the beginning, every AC will maintain a σ_{max} as an originated point of start and each time the λ condition is triggered, meaning that there was an excessive waiting time at the MAC, the current aggregate size will be reduced its threshold down to a predefined minimum value (σ_{min}). On the other hand, if the σ_{cur} is met or exceeded then the next value of this threshold is updated by a predefined increment but without exceeding the maximum size (σ_{max}). Eventually, the fluctuation shall be stabilized with σ_{cur} being equal to the number of segments of the TCP connection.

Nevertheless, it is possible for the algorithm to develop an oscillatory behavior where the triggering will bounce from the aggregate size threshold condition to the burst factor condition. In order to balance such an attitude, we also introduce some oscillation controllers which establish the number of times the individual triggers shall occur before an action takes place. If the pattern is serial and the number of count is converged, then we can assure that the oscillations have been reduced and ADCA reduces or increases the σ_{cur} value accordingly. We use two oscillation counters, beta (β) and phi (φ) for the λ and σ triggering, respectively. In Figure 3, we examine ADCA operation by giving an example, where the σ_{cur} varies with time according to the number and type of the triggering conditions; φ = 3 times and β = 2 times. In conclusion, such a variable and adaptable σ, allows the MAC layer to proceed with the ADCA operation without the need to have a priori knowledge of the flight size or what type of flow will be received from the upper layers. The ADCA can work for both UDP and TCP protocols and each AC queue will maintain individual values for the aggregate size threshold.

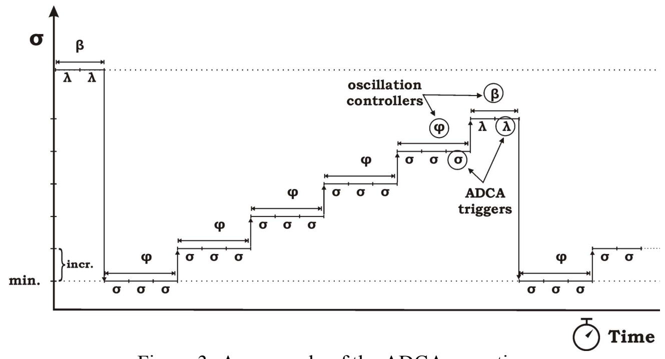

Figure 3: An example of the ADCA operation

TABLE 2: SIM. RESULTS FOR SCENARIO 1 (TCP WINDOW = 65535 B)

Name	Goodput (Mbps)	Avg. Aggregate Size	Max. Delay (sec)	Avg. Delay (sec)
HDTV	23.997	1.72	0.008973	0.000998
HDTV	19.200	1.26	0.009646	0.000869
Internet File	9.714	22.49	0.113258	0.031937

(a) 802.11n typical process

Name	Goodput (Mbps)	Avg. Aggregate Size	Max. Delay (sec)	Avg. Delay (sec)
HDTV	23.970	6.85	0.029742	0.005950
HDTV	19.122	5.98	0.021217	0.004952
Internet File	38.454	19.72	0.101951	0.007747

(b) 802.11n with ADCA algorithm enabled

IV. PERFORMANCE EVALUATION OF ADCA

In this section we evaluate the performance of ADCA using various scenarios that easily represent a home, a large enterprise and hot spot environment. All scenarios use TCP New Reno and the receiver's window size equal to 65535 B, since this is a common used window size and a point where a huge variation was shown at Section III when simple DCA was operated. For the home scenario, the previously defined Scenario 1 is applied and for the other two, we utilize scenarios 4 and 6 from TGn's usage models document [11]. The usage models intend to support the definitions of network simulations with a mixture of applications that will allow TGn to evaluate performance of various proposals in terms of network throughput, delay, packet loss and other metrics. The outputs of these simulations that have used the specified scenarios, will be subsequent sufficient for evaluation.

Due to the page size limit, the following set of standard performance metrics is shown: the goodput for the WLAN and each individual flow when applicable, the average aggregated sizes, the maximum and average latency values for every AC, and the packet loss rate (PLR) for QoS flows. PLR for a QoS AC is defined as the percentage of packets that are delivered successfully before the allowed maximal delay of the AC. All scenarios use as ADCA parameters: σ_{max} = 48 packets, σ_{min} = 10 packets, $\sigma_{incp.}$ = 2 packets, φ = 5 times and β = 2 times.

Table 2(a) and Table 2(b) show the results of our model for Scenario 1 with ADCA disabled and enabled, note that when ADCA is disabled we just use current TGn's specification. From Figures 1 and 2, we established that any further delay with TCP window sizes less than 70080 B delivers abhorrent results. But, with ADCA on and TCP window size of 64 KB, notice that the average aggregate sizes for both HDTV traffic increases notably from 1.26 and 1.72 packets per aggregate to 6.85 and 5.98 respectively. Hence, we can assume that the overall channel utilization and the goodput must be improved significantly too. Actually the performance data unquestionably proves this conjecture: the system's overall goodput boosts from 52.91 Mbps to 81.55 Mbps which is a 54.12% increase. Furthermore, all video packets, while having slightly longer delay than those without DCA are still delivered well below the allowed maximal delay (200ms). For example, the longest delay for the video packet is 29ms and 21ms that on average are about 1/8 of 200ms. Thus, as the use of ADCA improves performance, it proves to be effective.

Table 3 shows the simulation results for ADCA for Scenario 4 and Scenario 6. In both scenarios the simulations include the outcomes for when ADCA is on and when is off. Same as in Scenario 1, we've adapted the ADCA attributes according to the maximum allowed delay that the included applications have preset. Again, we observe that the ADCA algorithm improves the system goodput significantly from 58.69 Mbps to 80.73 Mbps and 49.84 Mbps to 62.53 Mbps, respectively. There is a significant increase of 37.1% for Scenario 4 and 27.25% for Scenario 6. Furthermore the maximal PLR for video flows is 0% in both scenarios and only for voice flows is 0.16% and 2.53% but again is less or equal than the allowed maximal PLR 5% as specified in [11]. ADCA's key role in increasing performance can be noted extremely when comparing the PLRs for video (VI) in Scenario 6. Without ADCA, the network fails to deliver 66.6% of the total video flows in time while when ADCA is enabled all packets received successfully with 0% PLR. All multimedia flows meet their QoS requirements when ADCA is enabled even though we defer further the transmission at the MAC layer. This is because ADCA manages to increases aggregate sizes for high priority flows and hence uses the channel more efficiently. More specifically, in Scenario 4 the VI flows have gone up 5.8 packets and the voice (VO) flows by 0.85 packets. On the other hand we see a decrease on the VI flows in Scenario 6 but this is normal since ADCA has stabilized the EDCA prioritization and now that VO flows have better channel utilization, the VI have more channel access and consequently the significant drop on the PLRs. Based on the above analysis, we can claim that the ADCA fixes the significantly negative performance impact by the poor interaction between EDCA and 802.11n plus it can effectively confine the TCP problem too.

TABLE 3: SIM. RESULTS FOR SCENARIO 4 AND 6

Scenario 4		Off. Load	Good put	Avg.Aggr.	Max Delay	Avg. Delay	Max PLR
ADCA Off	BE	460.18	58.69	38.7046	0.2247	0.1117	N/A
	VI			2.69	0.0578	0.0077	0%
	VO			1.1333	0.0339	0.0053	0.25%
ADCA On	BE	460.18	80.73	38.16	0.1519	0.0726	N/A
	VI			8.49	0.0572	0.0182	0%
	VO			1.98	0.0297	0.0096	0.16%

Scenario 6		Off. Load	Good put	Avg.Aggr.	Max Delay	Avg. Delay	Max PLR
ADCA Off	BE	64.88	49.84	30.715	0.6131	0.2448	N/A
	VI			40.753	0.5554	0.2937	66.6%
	VO			1.4687	0.0495	0.0091	2.4%
ADCA On	BE	64.88	62.53	46.121	0.3692	0.1857	N/A
	VI			14.337	0.0727	0.0175	0%
	VO			2.258	0.0422	0.0118	2.53%

V. CONCLUSION

In this paper, we first identified issues arising from the poor interaction of the EDCA prioritized channel access mechanism defined in the 802.11e standard and the frame aggregation mechanisms proposed by TGn in the latest draft standard. Using original DCA algorithm with static parameter, we show that these issues are addresses successfully however when various TCP windows sizes are considered a further problem was found. By incrementing the aggregate size threshold gradually until the flight size of the TCP flow is reached, we eliminate any TCP problems and MAC deadlocks. The static DCA is too rigid and there is no flexibility in dynamically adjusting the parameters. Our proposed adaptive DCA administers the contingency to incorporate adaptability. The simulation results evinced that the ADCA operation with various TCP window sizes improves the system performance significantly as compared with systems abstaining delayed channel access and hence it could be considered as a guide for the future High-Throughput standards.

REFERENCES

[1] IEEE Std. 802.11 WG, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", IEEE-SA Standards Board, August 1999 (Reaffirmed June 2003).

[2] Y. Xiao and J. Rosdahl, "Throughput and Delay Limits of IEEE 802.11", IEEE Communications Letters, Vol. 6, No. 8, pp. 355-357, August 2002.

[3] IEEE P802.11n, Draft 2.5, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Enhancements for Higher Throughput", IEEE-802.11 WG, July 2007.

[4] D. Skordoulis, Q. Ni, U. Ali & M. Hadjinicolaou, "Analysis of Concatenation and Packing Mechanisms in IEEE 802.11n", PGNET 2007, Liverpool, UK, June 2007.

[5] IEEE Std. 802.11e WG, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements", IEEE-SA Standards Board, November 2005.

[6] L. Changwen; A.P. Stephens, "Delayed Channel Access for IEEE 802.11e Based WLAN", IEEE International Conference on Communication '06, vol.10, pp.4811-4817, June 2006.

[7] Q. Ni, "Performance analysis and enhancements for IEEE 802.11e wireless networks", IEEE Network, vol.19, no.4, pp. 21-27, July 2005.

[8] OPNET Technologies, Inc., OPNET Modeler: Accelerating Network R&D [www] Available from: http://www.opnet.com/solutions/network_rd/modeler.html [Accessed 20th September 2007].

[9] J. Postel, "Transmission Control Protocol", RFC 793, Sept. 1981.

[10] IEEE P802.11.2, Draft 1.0, "Recommended Practice for the Evaluation of 802.11 Wireless Performance", IEEE-802.11 WG, April 2007.

[11] IEEE P802 Wireless LANs, "Usage Models", 11-03-0802-23-000n-usage-models.doc, May 2004.