A Selective Delayed Channel Access (SDCA) for the high-throughput IEEE 802.11n

Dionysios Skordoulis, Qiang Ni and Charilaos Zarakovitis

Brunel University, UK

Abstract— **In this paper we investigate the potential benefits of a selective delayed channel access algorithm (SDCA) for the future IEEE 802.11n based high-throughput networks. The proposed solution aims to resolve the poor channel utilization and the low efficiency that EDCA's high priority stations adhere due to shorter waiting times and consequently to the network's degrading overall end performance. The algorithm functions at the MAC level where it delays the packets from being transmitted by postponing the channel access request, based on their traffic characteristics. As a result, the flow's average aggregate size increases and consequently so is the channel efficiency. However, in some situations we notice that further deferring has a negative impact with TCP applications, thus we further introduce a traffic awareness feature that allows the algorithm to distinguish which flows are using the TCP protocol and override any additional MAC delay. We validate through various simulations that SDCA improves throughput significantly and maximizes channel utilization.**

Index Terms— **Delayed Channel Access, IEEE 802.11, Medium Access Control, TCP-Aware.**

I. Introduction

THE use of IEEE 802.11 [1] wireless local-area network (WLAN) has expanded rapidly over the last decade and it has been further prevalent in both business and home environments. However, the emergence of bandwidth-intensive real time or not applications, such as HDTV and VoD, along with the wireless internet and the increasingly evolving peer-to-peer technologies, have lead to the need of employing higher throughput (HT) WLANs. Consequently, the main subject of research on wireless access techniques still remains the racing to boost the transmission capacity up to the user's expectations but at the same time provide greater coverage and maintain a Quality of Service (QoS) support.

In late 2005, a set of QoS Media Access Control (MAC) layer enhancements were introduced with the 802.11e amendment [2]. These are considered of great importance because they resolve the constraints that delay-sensitive applications comprise with. Despite of all various amendments, a greater demand for HT connections was foreseen but unfortunately current extensions are bounded with a theoretical throughput limit (TTL) due to the existence of MAC and PHY overhead [3]. So, a new Task Group (TGn) was set off with the intention to attain and define a standard that will support maximum MAC data throughput of at least 100 Mbps. At the time this paper was written, TGn had resolved most of the draft's [4] comments and was ready to issue a final version. Some of the main proposals are MIMO-OFDM usage and an innovative frame aggregation at the PHY and MAC, respectively.

A major dilemma while researching new proposals is how new ideas can be coalesced with previous standards. From the MAC layer's perspective, TGn's draft document is primarily based on 802.11e's two methods of channel access, the Enhanced Distributed Channel Access (EDCA) and the HCF Controlled Channel Access (HCCA). Additionally, MAC and PHY overhead, is treated with a method known as frame aggregation in which all packets contained in the same transmission buffer and destined to the same receiver are concatenated within one frame [5]. However, it has been shown in [6] that poor channel utilization exists because of high priority flows producing small aggregated sizes and as a result higher overhead with increased number of channel accesses. The resolution for this abominable consequence was a delayed channel access (DCA) algorithm that coerces stations (STAs) into further deferring in a way that allows more packets to arrive during that period thus increasing the end aggregate sizes. Although this work is interesting, TCP flows with various TCP window sizes weren't considered during experimentation and as we explain in this paper, these could result in aggravate outcomes. This paper exposes the negative impact that DCA applies over TCP performance and proposes a flow discriminative function that will allow DCA to countermand its operation to TCP segments. Our approach is a MAC exclusively solution that is based on the recording and analysis of the time arrival intervals between consequent packets arriving from higher layers. We name this new algorithm as selective DCA (SDCA) and it can be applied on any future 802.11n device. We achieve to maximize channel efficiency and increase the network's overall performance.

The remainder of this paper is organized as follows. Section II presents the poor channel utilization that high priority applications have over HT networks and an overview of the DCA's algorithm. In Section III we analyze the consequences of further deferring over TCP traffic and how this can be avoided with a MAC layer TCP-aware functionality. Our SDCA algorithm is validated using extended simulations in Section IV and finally Section V concludes the paper.

Dionysios Skordoulis, Qiang Ni and Charilaos Zarakovitis are with the School of Engineering and Design, Brunel University, London, UK (e-mail: {Dionysios.Skordoulis, Qiang.Ni and Charilaos.Zarakovitis}@brunel.ac.uk).

II. Delayed Channel Access

A. 802.11e and 802.11n in conjunction

The 802.11e EDCA is a QoS extension of the legacy 802.11 DCF and is designed to provide distributed channel access for 8 different user priorities (UPs). Primitively speaking, the higher the UP assigned to a packet, the greater the delay-constraints from the originated application. Thus, these UPs are mapped to separate access categories (ACs) with their own queue buffer and each have an analogous channel access waiting time by assigning a unique set of EDCA parameters: Arbitrary Interframe Space (AIFS) and a pair of minimum/maximum values for the Contention Window (CW_{min} and CW_{max}). Further information regarding the EDCA operation can be found in [2] and [7]. The main point to keep in mind is that higher priority categories are capable of acquiring more bandwidth than the lower priority categories when they are competing against each other and since channel access is "expensive" this can cause starvation to lower ACs.

As we mentioned earlier, TGn's draft standard builds upon 802.11e's probabilistic priority mechanisms along with other MAC enhancements, like frame aggregation. There are two types of aggregation, Aggregated MAC Service Data Unit (A-MSDU) and Aggregated MAC Protocol Data Unit (A-MPDU). The analysis in [5] has shown that the maximum ideal throughput is bounded by a maximum relative MAC throughput that is just over half of the average peak PHY. This occurs because of MAC and PHY overhead which is the additional channel time that is consumed for the successfully transmission of each data payload. The main principle of A-MSDU operation is to allow multiple MSDUs to be sent to the same receiver concatenated in a single MPDU, while A-MPDU aggregation joints multiple MPDU subframes into one PPDU. A main differentiation from A-MSDU aggregation is that it functions after the MAC header encapsulation process. However, both choices are adequate in there own manner to eliminate defective overhead and as a result to extensively improve the channel efficiency and the data throughput. However, in the interest to increase the aggregated size, there is a need of packets to be piled in the stack.

In order to understand the poor effect that EDCA mechanism has over frame aggregation, we can simply consider an HDTV application that has a mean rate of 19.2 Mbps and a constant MSDU size of 1500 bytes. From the above we can determine the packet interarrival time at the MAC layer as 625 μsec. Now, as the first packet bursts in the queue buffer, the STA initiates a channel access process which assuming that this is a voice (VI) priority packet the mean channel access delay should be equal to ~241 μsec (derived after calculating the sum of AIFS[VI] + mean BO, see [8] for further information). By the time the second packet arrives, the first one has already been transmitted and there is no chance for the packets to be concatenated. As a result, the application has high overhead and low channel utilization.

B. DCA Review

So far, we briefly explained that a STA with high priority flows sends frames in small aggregates because of the short channel access delay and since the overall channel utilization is defined as the overall traffic usage from all types of ACs, the performance tilts relating to the low-efficiency high priority flows. In order to rectify this issue we need to increase the aggregation threshold for the high priority flows by preceding some additional delay before the decisive aggregated frame accesses the channel. Then again, this situation might lead to unnecessarily idling even when the packet queue isn't empty. For that reason, an algorithm needs to be applied so that it can match the aggregated packet formation with the traffic burst with an appropriate time scale. The burst formation requires being proportional to the channel load, given that burst time scale can be either high when the channel load is elevated or low when the channel load is minimal. Also, to support the possible QoS constraint placed on each flow, the formation must also consider the allowed maximal delay for the packets. A good measurement for each station's channel load was proposed by Changwen Liu and Adrian Stephens with their DCA algorithm [6].

The channel access delay for a frame arriving at the MAC is defined as the amount of time between the frame's arrival at the front of the queue buffer and its successful transmission to the intended receiving STA, excluding the time it takes to propagate in the air. The DCA algorithm maintains three attributes (see Table 1) and these are considerably important for the determination of the algorithm's decisions. The algorithm delays the channel access as long as the number of packets in the aggregation buffer hasn't reached the σ (sigma) value, or the period since the first packet was received hasn't exceeded the maximal waiting time τ (tau), or the duration from the last received packet remains below the time that was last needed to access the channel by a factor of λ (lambda).

To evaluate DCA's performance, we simulate a TGn model in OPNET (Optimized Network Engineering Tool) Modeller [9]. We consider a simple scenario, referred as Scenario 1, for an overloaded 802.11n WLAN that holds three STAs and one 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 [4]). Also, we set two types of HDTV flows over 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. Last, the QoS attributes for the DCA are set to $\lambda = 10$, $\tau = ½$ maximal delay and $\sigma = 48$ packets and for the access parameters (CW, AIFS, TXOP, etc.) for each service class the default values are appointed.

Table 1: DCA's QoS Attributes

λ	A positive constant that is the ratio of the inter-arrival time to the access delay.
τ	A constant that is the maximal waiting time for packets in the aggregation buffer.
σ	A constant that determines the maximal number of packets in the aggregation buffer before aggregation is triggered.

TABLE 2: SIMULATION RESULTS WITHOUT DCA FOR SCENARIO 1

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

TABLE 3: SIMULATION RESULTS WITH DCA FOR SCENARIO 1

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

The simulation was run for 5 seconds but the results have been collected from the last 4 seconds as we allow the TCP congestion window (CWND) to fully build up. Table 2 and Table 3 display the goodput, average aggregate size, maximal and average delay with and without DCA. From the data collected when DCA is disabled, we validates that the system's performance suffers due to the poor interaction between EDCA and frame aggregation. As a result of the small aggregates on the higher AC flows, we can determine the network's overall inefficiency at 47%. The total goodput measured at the MAC is 54.987 Mbps out of the 117 Mbps possible PHY rate. On the other hand, when we use DCA the MAC efficiency is boosted at 81% and the overall goodput moves up to 94.98 Mbps. Also, we observe that by introducing additional delay before channel access, the end-to-end delay to the HDTV traffic had an insignificant increase and the maximum delay remain way below the 200 ms delay boundary. DCA doesn't override the AC's higher priority even though it reduces sufficiently the frequent channel accesses in order to provide better channel utilization since the aggregated size is increased. It is obvious that for the specific scenario the DCA algorithm has increased the system's effectiveness.

III. SELECTIVE DCA

A. The TCP Problem with DCA

The Transmission Control Protocol [10] is a reliable, robust and connection-oriented method of data delivery. It is commonly used over the Internet as it is well known for its flexibility since it adapts the transmission behaviour dynamically according to the network's disparate conditions. The form of data that TCP passes over to the Internet Protocol (IP) layer are known as segments and the maximum segment size (MSS) is usual equal to the maximum transmission unit (MTU) of the system's data link layer. Each segment is stamped with a sequence number so the end receiver can reply back with corresponding TCP acknowledgements (ACK) over the segments that it has successfully received. If a TCP ACK is not received within a reasonable round-trip time (RTT), then it will be assumed that the data was lost and a re-transmission will be initiated. Also, in order to maintain a flow control, a congestion control mechanism takes place where the sender is merely allowed to send as much data as the receiver has buffers for, its advertised window. But since an end-to-end link may contain intermediate bottlenecks and routers with smaller window sizes, the sender node sets a CWND were the buffer size is equal to a single segment and every time it receives an ACK it increases it exponentially according to the slow-start algorithm's rules [11].

Previously in Section II during DCA's evaluation, the TCP's window size was not indicated. In fact, the Scenario 1 sets a window size of 655350 bytes, following the recommendations in [8] where it is suggests 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 over wireless links since the RTT may vary, so when we set up a scenario in order to investigate the network's behaviour with smaller values, say 8 KB, 16 KB, 32 KB, 64 KB etc, the results diverge. For this scenario we assume one STA and one AP with a single TCP flow of offered load at 120 Mbps transmitted from the former to the latter. Figure 1 and Figure 2 show the outcomes for window sizes equal to 655350 bytes and 65350 bytes. While the results for large window size remain similar before and after DCA as expected, when the same conditions applied for low window sizes we observe a huge impact. Particularly, when DCA was conjointly used with TCP flows, the TCP throughput decreases rapidly from 89.337 Mbps to 13.65 Mbps while the average delay increases dramatically from ~3.846 ms to ~35.583 ms. The DCA algorithm will cause the packets to defer from requesting to access the medium until one of its conditions is triggered. Since this was Best Effort (BE) traffic the τ attribute has no maximum delay defined and the value was set to be very large, thus no expected trigger from this condition. And because the MSS is 1500 bytes and the maximum PPDU allowable size is 65535 bytes [8] which is around 43 packets per PPDU, the condition for the total

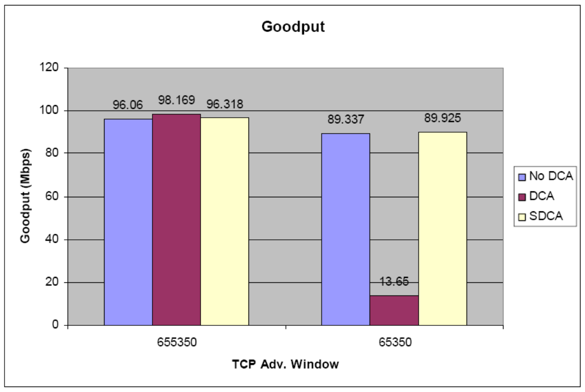

Figure 1: Goodput results for TCP flow

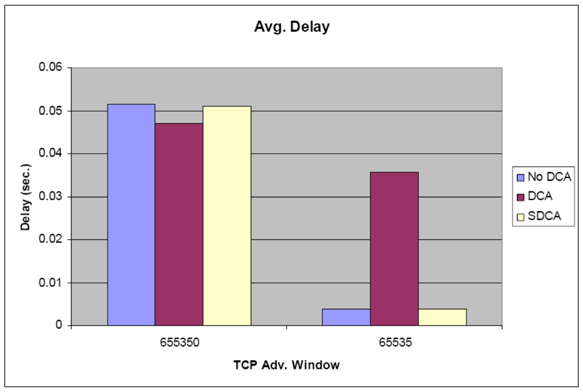

Figure 2: Average Delay results for TCP flow

aggregate size (σ) won't be triggered either as this was set for 48 packets. Thus, DCA is only initiating channel access from the λ condition. This situation results in a case where both the TCP layer and the MAC layer are waiting for data from each other. The TCP is waiting to get acknowledgements for a number of segments that already had sent; before it can continue with the next CWND while at the MAC layer the DCA is waiting for following segments to come from the upper layer before some of the other conditions can trigger the channel access stage. This leads to a point where both layers are dependant on each other to proceed and both are interlocked in a waiting period and cannot do anything, hence the long delay results. Consequently, although we previously validated that DCA increases the channel efficiency for high priority flows when it comes down to the TCP traffic with small window buffers there is an issue which need to be resolved.

B. *Selective DCA algorithm*

This problem was investigated thoroughly and a few solutions arisen and tested. A quick fix to this issue is to decrease the delay within DCA by altering the trigger attributes with smaller values. By doing so, the total performance dropped and most of the times the channel efficiency descents at adjacent levels with no DCA, thus the issue of the 802.11e and 802.11n arises again. Another way out was to implement an adaptive DCA algorithm which will accustom the aggregate size trigger (σ) dynamically with the queue size. This attempt delivered improved outcomes but the aberration wasn't significant. However, we managed to boost the performance even higher by introducing a function that is able to identify the type of traffic that arrives at the MAC based on the packets' interarrival times and size. If the function recognizes a TCP flow, it sends the packet straight to transmission rather advancing into the DCA state. For that reason, we named are TCP-Aware enhanced DCA algorithm as selective DCA (SDCA).

The OPNET trace files collected from the simulations show that the TCP segments assigned in a single CWND were arriving at the MAC layer in a homogeneous Poison process with a constant rate. The period from the point the first segment arrives at the MAC layer until a second full segment appears from the same TCP process was observed to be precisely 10 μsec. The main operation of the proposed function is expressed in C++ and looks like:

```
TCP_DECIDER ( current_time, last_received_time)  {
   double EPSILON = 0.000001;      //Tolerance constant 1 μsec
   double TCP_IAT = 0.00001;        // 10 μsec
   double current_iat = current_time - last_received_time;
   boolean decider = ((TCP_IAT – EPSILON) ≤ actual_iat) &&
   (actual_iat ≤ (TCP_IAT + EPSILON));
   return decider;
}
```

So, each AC buffer collects discrete information for each recipient and reviews the flows separately. Within the function there is a constant double variable named a EPSILON and it defines the level of tolerance, meaning that there could be a margin of deviation on the TCP packet rate as hardware equipment are not such precise as software simulators. Lastly, in order to decrease the level of misconception the function also checks the packet sizes. It is known that when a TCP connection established there is a three-way handshake, a negotiation between the two nodes where they shared information with specific segments with no data but just the headers. The size of these segments can easily be determined (usually 40 bytes long) and audited at the beginning and end of the transaction. As a result whenever, a 40 byte packet shows up at the MAC then the function increases its level of awareness.

TABLE 4: SIM. RESULTS WITHOUT SDCA FOR SCENARIO 1

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

TABLE 5: SIM. RESULTS WITH SDCA FOR SCENARIO 1

Name	Goodput (Mbps)	Avg. Aggregate Size	Max. Delay (sec)	Avg. Delay (sec)
HDTV	23.994	12.66	0.034178	0.012935
HDTV	19.062	11.90	0.038240	0.014899
Internet File	47.517	16.74	0.041683	0.006381

The accuracy of the function's decisions cannot be mathematically determined since the combinations of different type of traffic associations are infinite and difficult to define. However, it has been tested with a set of traffic patterns and found that if there are any misjudgements on the flow type, there isn't going to be any negative effect on the SDCA performance. Assume an example that a UDP traffic with the same AC and same receiver as the TCP flow takes place at the node's MAC layer. Then, the function may not be able to distinguish any distinctness between them as the next TCP packet will be compared with the last arrived UDP traffic. However, the packets in queue increases exponentially and SDCA triggers before the interlock situation discussed earlier occurs. Also, it is unusual to get UDP traffic with a 10 μsec interarrival rate therefore in a series of TCP packets will definitely have successes as at least two consequent packets will be TCP. So, once the first packet initiates a transmission sequence every other following packet UDP or TCP that arrives during that time will be transmitted too. Finally, because TCP traffic are more likely to be Best Effort flows it will ordinarily have a long AIFS and Back Off timer so it helps the aggregates to increase further without the need of SDCA. In conclusion, the possibility of false detection of the type of traffic will not impact to the system's performance but on the contrary the negligence of SDCA usage could result in aggravate outcomes.

IV. PERFORMANCE EVALUATION OF SDCA

In this section, we evaluate closely the performance of SDCA through various simulations using OPNET. The design and choice of network architecture for each scenario corresponds to a home, a large enterprise and a hot spot environment. Therefore, for the home scenario, we use previously defined Scenario 1 while for the large enterprise and hot spot layout, we follow the 802.11n usage Scenarios 4 and 6 in TGn's usage models document [12], respectively. The usage models intend to support the definitions of network simulations with a mixture of applications that will allow 802.11 TGn to evaluate performance of various proposals and outputs of these simulations will be subsequent sufficient for evaluation. Briefly, Scenario 4 contains one AP and 30 associated STAs that carry on a mixture of applications, such as internet and local file transfers, video conferencing, VoIP and some media player usage. On the other hand, Scenario 6 has 41 STAs and an AP, all within the same range. The configuration is arbitrary like most hot-spot networks are and the traffic applicable is VoIP, high and mid quality video/audio streaming, SDTV broadcasting and file transfers. All scenarios use TCP New Reno and the receiver's window buffer is set at 65535 bytes.

Because, the size of this paper is bounded to a certain length the following set of standard performance metrics are collected: the goodput for the WLAN and each individual flow, the average aggregated sizes, the maximum and average latency values for every AC, and the packet loss rate (PLR) for QoS flows only. The PLR outcome for a QoS AC is defined as the percentage of packets that have not been delivered within the allowed maximal delay as a result causing unwanted jitters or packet discarding.

Table 4 shows the computed results for Scenario 1 without SDCA while in Table 5 the SDCA algorithm is set to enable. Notice that the average aggregate sizes for both HDTV traffic increase notably from ~1.72 packets and ~1.26 packets per aggregate frame to ~12.66 packets and ~11.90 packets, respectively. Hence, as the channel utilization for the higher ACs increases we would assume that the overall goodput must be improved significantly too. Actually the performance data unquestionably proves this conjecture where the system's overall goodput boosts from 52.91 Mbps to 90.57 Mbps which is a 71.18% increase. Furthermore, all video packets, while having slightly longer delay than those from Table 4, are still delivered well below the allowed maximal delay (200ms). For example, the longest delays for the HDTV packets are ~34 ms and ~38 ms that is around 20% of the 200 ms bound. Also, the BE flow, Internet File transfer, has now build up the number of channel accesses resulting to less end-to-end delay and a huge increase to its goodput from 9.714 Mbps to 47.517 Mbps. Note that the average aggregated size is slightly reduced, so somebody may argue that this behaviour opposes to the frame aggregation philosophy. Nevertheless, as the channel access waiting times are reduced, the transmission data rate is increased. Along this scenario SDCA proves effective of improving the system's goodput with no QoS suffering.

TABLE 6: SIM. RESULTS FOR SCENARIO 4

Scenario 4		Off. Load	Good put	Avg.Aggr.	Max Delay	Avg. Delay	Max PLR
SDCA 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%
SDCA On	BE	460.18	80.46	36.51	0.1385	0.0590	N/A
	VI			8.56	0.0617	0.0187	0%
	VO			1.964	0.0307	0.0094	1%

TABLE 7: SIM. RESULTS FOR SCENARIO 6

Scenario 6		Off. Load	Good put	Avg.Aggr.	Max Delay	Avg. Delay	Max PLR
SDCA 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%
SDCA On	BE	64.88	63.42	56.48	0.3610	0.1510	N/A
	VI			18.191	0.0679	0.0213	0%
	VO			2.459	0.0538	0.0140	5%

Figure 3 and Figure 4 display a bar presentation of each ACs goodput for Scenario 1. The TCP window size is 655350 bytes in the first case and 65535 bytes in the second. For these simulations we also provide the results from the simple DCA algorithm as we want to show its differences with SDCA. For both HDTV flows we see no changes on the goodputs and in all cases the expected offered load is achieved, but for the Internet File transfer we distinguished the same TCP problem as in Section III.A. While the BE flow is 9.714 Mbps with no delayed channel access, when we apply normal DCA it drops down to 4.494 Mbps except when we use the TCP-aware enhancement with SDCA it boosts to 47.517 Mbps. The latter confirms the adeptness of SDCA to handle TCP traffic efficiently in contradiction with the simple DCA algorithm that fails to do so.

Table 6 and Table 7 show the simulation results for SDCA for Scenario 4 and Scenario 6, respectively. Again, in both scenarios the outcomes include the simulation results for when SDCA is enabled and when SDCA is disabled. Same as in Scenario 1, the SDCA's τ (tau) attribute has been assigned a value half of the maximum allowed delay that the Application layer has preset for the QoS ACs. From the results, we observe that the SDCA algorithm improves the system goodput significantly from 58.69 Mbps to 80.46 Mbps and 49.84 Mbps to 63.42 Mbps, respectively. This is a significant increase by 37% 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 1% and 5% but again is less or equal than the allowed maximal PLR 5% as specified in [12].

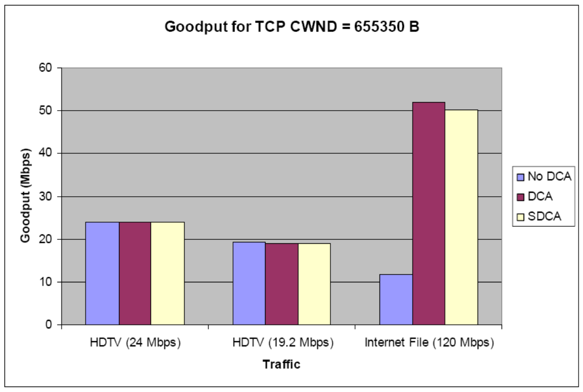

Figure 3: Goodput results for Scenario 1 – TCP CWND = 655350 B

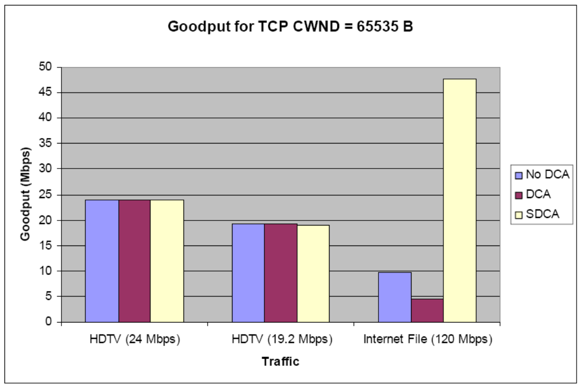

Figure 4: Goodput results for Scenario 1 – TCP CWND = 65535 B

SDCA's key role in increasing performance can be noted extremely when comparing the PLRs for VI in Scenario 6. Without SDCA, the network fails to deliver in time 66.6% of the total video flows while when SDCA is enabled all packets received successfully with 0% PLR. All multimedia flows meet their QoS requirements when SDCA is enabled even though we choose to defer further their transmission. This is because SDCA manages to increase the aggregate sizes for high priority flows and hence uses the wireless medium more efficiently. More specific, in Scenario 4 the VI flows have gone up by ~5.87 packets and the VO flows by ~0.83 packets. On the other hand, we see a decrease of the aggregated size of the VI flows in Scenario 6 but this is normal since SDCA has stabilized the 802.11e's probabilistic priority mechanism and since VO traffic has better channel utilization, the VI flows have increased the chances of channel accesses and consequently the significant drop on the PLRs. Based on these performance data and the above analysis, we can claim that the SDCA fixes the significantly negative performance impact by the poor interaction between EDCA and 802.11n plus it can effectively confine the TCP problem too.

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. We highlighted a significant negative impact on system performance by the interaction through both theoretical analysis and simulation. Using original DCA algorithm we show that these issues are addressed successfully, however when we consider various TCP windows sizes we introduce a further problem. By using a simple function that analyzes and records the flow of packets arriving at the MAC layer we can specify the packet's type of transportation protocol that uses. Finally, we use more that one scenario to evaluate our proposal and prove that the SDCA algorithm improves the system performance significantly and hence it could be considered in the design of next-generation 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] 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.

[3] 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.

[4] 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.

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

[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-Aug. 2005.

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

[9] 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].

[10] J. Postel, *"Transmission Control Protocol"*, STD 7, RFC 793, September 1981.

[11] W. Stevens, *"TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms"*, RFC 2001, January1997.

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