Differentiation mechanisms for IEEE 802.11

Imad Aad and Claude Castelluccia
PLANETE project, INRIA Rhône-Alpes.
ZIRST - 655, Avenue de l'Europe - Montbonnot
38334 Saint Ismier Cedex - France
[imad.aad, claude.castelluccia]@inrialpes.fr
http://www.inrialpes.fr/planete.html

Abstract— **The IETF is currently working on service differentiation in the Internet. However, in wireless environments where bandwidth is scarce and channel conditions are variable, IP differentiated services are sub-optimal without lower layers' support.**

In this paper we present three service differentiation schemes for IEEE 802.11. The first one is based on scaling the contention window according to the priority of each flow or user. The second one assigns different inter frame spacings to different users. Finally, the last one uses different maximum frame lengths for different users.

We simulate and analyze the performance of each scheme with TCP and UDP flows.

Keywords—**QoS, DiffServ, TCP, UDP, CBR, Wireless communications.**

I. INTRODUCTION

WIRELESS communications are an emerging technology and are becoming an essential feature of everyday's life. Not only computer networks are becoming mobile, eventually each device will have one or several wireless interfaces (e.g. laptops, cameras, phones etc.) [1], [2]. Simultaneously, multimedia is having an equivalent growth. Multimedia applications impose requirements on communication parameters, such as data rate, drop rate, delay and jitter. Guaranteeing those requirements in wireless environments is very challenging because wireless links have variable characteristics (due to noise). To deal with this problem, many wireless communication standards have been defined. Some of them enhance the Quality of Service (QoS) of the whole system [3], others differentiate between the priorities of each mobile host, offering them different quality of service parameters (e.g. different data rates or delays etc.)[4]. In this paper we propose mechanisms for service differentiation for IEEE 802.11. The paper is organized as follows: Section II describes the IEEE 802.11 protocol. Section III presents simulations and analysis of the IEEE 802.11 when used with TCP *(Transport Control Protocol)*[5] and UDP *(User Datagram Protocol)*[6] transport protocols. Section IV introduces some means of service differentiation on the wireless link with some simulations and mathematical models. Finally, section V gives some hints for future work and section VI concludes this paper. An extended version of this paper is [7].

II. IEEE 802.11

The IEEE 802.11 standard covers the MAC (Medium Access Control) sub-layer and the physical layer of the OSI (Open System Interconnection) reference model. In this paper, we only focus on the MAC part. A general description of the IEEE 802.11 standard is available in [8], [2].

A group of *Wireless Terminals* (WTs) under the control of a *Distributed Coordination Function* (DCF) or a *Point Coordination Function* (PCF) forms a *Basic Service Set* (BSS), and the area it covers is called *Basic Service Area* (BSA). A BSS can either be an independent *ad-hoc*[1] network or an *infrastructure network*, in which an *Access Point* (AP) links the WTs to a *Distribution System* (DS), therefore extending their range to other BSSs via other APs. The whole system is then called *Extended Service System* (ESS). The DS can be any kind of fixed or wireless LAN.

This standard supports two services:
- *Distributed Coordination Function* (DCF): which supports delay insensitive data transmissions (e.g. email, ftp).
- *Point Coordination Function* (PCF): this service is optional. It supports delay sensitive transmissions (e.g. realtime audio/video) and is used in combination with DCF.

In a BSS, WTs and the AP can either work in *contention mode* exclusively, using the DCF, or in *contention-free mode* using the PCF. In the first mode, WTs have to contend for use of the channel at each data packet transmission. In the second mode the medium usage is controlled by the AP, polling the WTs to access the medium, thus eliminating the need for contentions. This last mode is not exclusive, and the medium can be alternated between *contention mode* and *contention-free mode* for CP (Contention Period) and CFP (Contention Free Period) respectively.

A. Distributed Coordination Function (DCF)

As mentioned earlier, the DCF is an asynchronous data transmission function, which best suits delay insensitive data. It is the only possible function in ad-hoc networks. When used in an infrastructure network, DCF can be either exclusive or combined with PCF. Each WT gets an equal share of the channel through contention, i.e. a WT contends for channel use before each frame waiting for transmission.

The basic scheme for DCF is *Carrier Sense Multiple Access* (CSMA)[9]. This protocol has two variants: Collision Detection (CSMA/CD) and Collision Avoidance (CSMA/CA). A collision can be caused by two or more stations using the same channel[2] at the same time after waiting for the channel to become idle, or (in wireless networks) by two or more hidden terminals[3] transmitting simultaneously.

CSMA/CD is used in Ethernet (IEEE 802.3) wired networks. Whenever a node detects that the signal it is transmitting is different from the one on the channel, it aborts transmission, saving

[1]An ad-hoc network is a group of wireless nodes connected together without control of any centralized point.

[2]On the physical layer, in *spread spectrum* technology, a channel is the pseudo-random sequence used to "spread" data.

[3]Hidden terminals are terminals which cannot hear each other [10].

useless collision time. This mechanism is not possible in wireless communications because a WT cannot listen to the channel while it is transmitting, due to the big difference between transmitted and received power levels. To deal with this problem, the sender should wait for an acknowledgment (ACK) from the receiver after each frame transmission, as shown in Fig. 1. *Source* axis shows the data transmitted by the source. The destination replies with an ACK, shown on the *Destination* axis. The third axis shows the network status, as seen by *Other* WTs. Note that transmission delays are not shown. The Interframe Spacings DIFS and SIFS will be explained later in this section.

If no ACK is returned, a collision must have occurred and the frame is retransmitted. This technique may waste a lot of time in case of long frames, keeping the transmission going on while collision is taking place (caused by a hidden terminal for example).

To solve the hidden terminal problem, an optional RTS/CTS (Request To Send and Clear To Send respectively) scheme is used in addition to the previous basic scheme, as shown in Fig. 2: a station sends an RTS before each frame transmission to reserve the channel. Note that a collision of RTS frames (20 octets) is less severe and less probable than a collision of data frames (up to 2346 octets). The destination replies with a CTS if it is ready to receive and the channel is reserved for the packet duration. When the source receives the CTS, it starts transmitting its frame, being sure that the channel is reserved for itself during all the frame duration. All other WTs in the BSS update their *Network Allocation Vector* (NAV) whenever they hear an RTS, a CTS or a data frame. NAV is used for *virtual carrier sensing*, as detailed in the next paragraph.

The overhead of sending RTS/CTS frames becomes considerable when data frames sizes are small, and the channel is suboptimally used. References [10] and [11] discuss optimal data frame sizes (*RTS_Threshold*) above which it is recommended to use the RTS/CTS scheme. Very large frames may reduce transmission reliability too. e.g. an uncorrectable error in a large frame wastes more bandwidth and transmission time than an error in a shorter frame. So another optimization parameter is used, which is *fragmentation_threshold*, above which packets are fragmented.

Not all packet types have the same priority. For example, ACK packets should have priority over RTS or data frames. This is done by assigning to each packet type a different *Inter Frame Spacing* (IFS), after the channel turns idle, during which a packet cannot be transmitted. In DCF two IFSs are used: Short IFS (SIFS) and DCF IFS (DIFS), where SIFS is shorter than DIFS (See Fig. 1 and 2). As a result, if an ACK (assigned with SIFS) and a new data packet (assigned with DIFS) are waiting simultaneously for the channel to become idle, the ACK will be transmitted before the new data packet (the first has to wait SIFS whereas the data has to wait DIFS).

Carrier sensing can be performed on both physical and MAC layers. On the physical layer, *physical carrier sensing* is done by sensing any channel activity caused by other sources. On the MAC sub-layer, *virtual carrier sensing* can be done by updating a local NAV with the value of other terminals' transmission duration. This duration is declared in data, RTS and CTS frames. Using the NAV, a WT's MAC knows when the current transmission ends. NAV is updated upon hearing an RTS from the sender and/or a CTS from the receiver, so the hidden node problem is avoided.

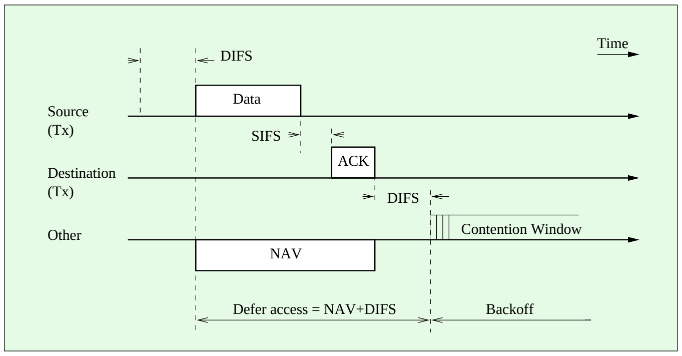


Fig. 1. Basic access scheme.

The *collision avoidance* part of CSMA/CA consists of avoiding packet transmission right after the channel is sensed idle for DIFS time, so it does not collide with other "waiting" packets. Instead, a WT with a packet ready to be transmitted waits the channel to become idle for DIFS time, then it waits for an additional random time, *backoff time*, after which the packet is transmitted, as shown in Fig. 1 and 2. Collision avoidance is applied on data packets in the basic scheme, and on RTS packets in the RTS/CTS scheme. The backoff time of each WT is decreased as long as the channel is idle (during the so called *contention window*). When the channel is busy, backoff time is freezed. When backoff time reaches zero, the WT transmits its frame. If the packet collides with another frame (or RTS), the WT times out waiting for the ACK (or the CTS) and computes a new random backoff time with a higher range to retransmit the packet with lower collision probability. This range increases exponentially as 2^{2+i} where i (initially equal to 1) is the transmission attempt number. Therefore, the backoff time equation is:

$$\textit{Backoff\_time} = \lfloor 2^{2+i} \times rand() \rfloor \times Slot\_time \tag{1}$$

where $Slot\_time$ is function of physical layer parameters, and $rand()$ is a random function with a uniform distribution in [0,1]. There is a higher limit for i, above which the random range remains the same. The packet is dropped after a given number of retransmissions.

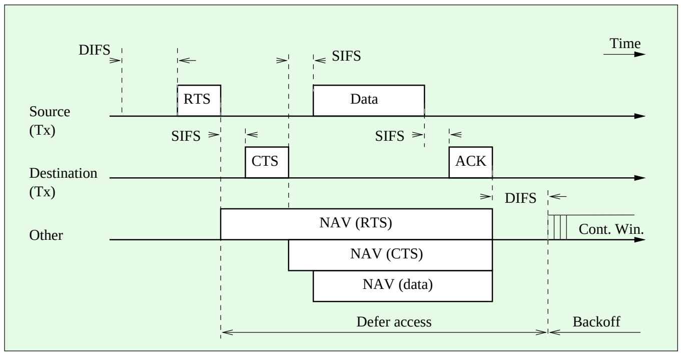


Fig. 2. RTS/CTS access scheme.

All WTs have equal probability to access the channel and thus share it equally. But this method has no guarantees for queuing delays, so it is not optimal for time-bounded applications. Time-bounded applications are better supported with the optional PCF.

III. UDP and TCP over IEEE 802.11

In this section, we present simulation[4] results and we analyze the behavior of UDP and TCP when running on top of an IEEE 802.11 MAC sub-layer: UDP and TCP. The topology of the simulation network is rather simple (see Fig. 3): Three WTs, denoted by WT_i where $i = 1, 2$ and 3 respectively, are uniformly distributed around an AP and they are sending their packets to a fixed host wire-attached to the AP.

A. UDP flows

Let us first consider the use of Constant Bit Rate (CBR) traffic sources over UDP. WT_1, WT_2 and WT_3 start sending their CBR/UDP packets at seconds 50, 100, and 150 respectively, using the RTS/CTS scheme. Simulation ends at second 250. During time interval [50,100[, WT_1 can get the desired data rate as long as it doesn't exceed the effective radio link data rate, i.e. 1.6 Mbps in our simulation[5]. In this example a single traffic overloads the link, sending 1100-byte packets each 0.005 seconds (giving a data rate of 1.76Mbps $>$ 1.6Mbps, so the channel is busy most of the time). As shown in Fig. 4, WT_1 has a stable throughput. It also has short delays and jitters[6]. The drop rate, which is about 10% in this case, depends on the used bit rate. During the second phase (i.e. between seconds 100 and 150), WT_1 and WT_2 share the data rate almost equally as they both have the same probability to access the medium (Fig. 4). The average delays of both traffics are higher than in the first period due to a higher number of RTSs denied: The channel is occupied by one terminal, the other terminal must wait during that time. It can also be the case that RTSs collide. Jitter also gets higher due to the more variable channel usage, caused by a higher number of WTs contending to access the channel. During the third period, between seconds 150 and 250, WT_3 shares the medium with the previous two. Throughput gets lower, as data rate is shared among the three WTs. Delay, jitter and drop rate get higher.

[4]Using *NS (Network Simulator)* from Berkeley [12].
[5]We consider a raw data rate of 2 Mbps which corresponds to an effective data rate of approximately 1.6 Mbps .
[6]We consider that the jitter is the standard deviation of the delay.

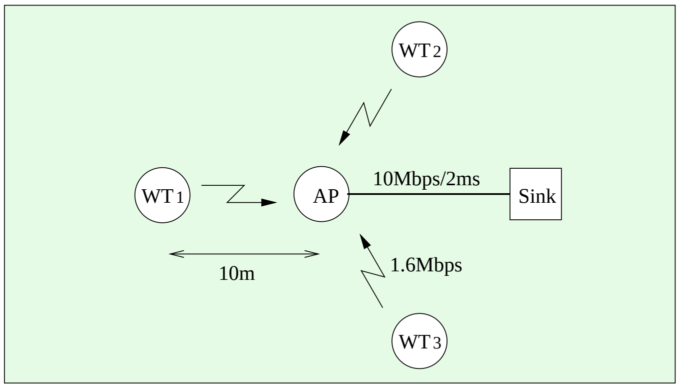


Fig. 3. Simulation network topology.

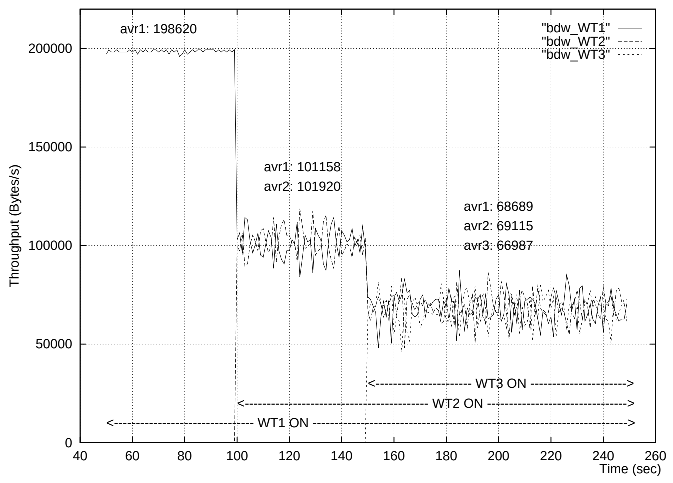


Fig. 4. Throughput using UDP.

B. TCP flows

When we replace the UDP transport layer with the TCP one, the throughput, delay and jitter behave the same way as in UDP. However packet dropping due to buffer overflow at the sender is avoided with TCP. We observe absolutely no TCP dropped packets due to its adaptability and to the use of the RTS/CTS scheme: When the sender requests to transmit and the channel is idle, no dropping is observed as long as the traffic is adapted to the offered throughput, which is the case of TCP. Some RTSs collide, are dropped, then retransmitted by the MAC sub-layer transparently to the TCP layer.

The TCP *congestion window (cwnd)* sizes of all three WTs are shown in Fig. 5, for the whole simulation time[7].

At each new period, more congestion occurs and the general slope decreases. However the *congestion window* never decreases during the simulation time, even at the instant values scale. After the *Slow Start* period, in which the *cwnd* increases by 1 at each ACK reception, the *cwnd* reaches the *ssthreshold* (20 in this case) then the *congestion avoidance* period starts, during which *cwnd* increases by $1/cwnd$ at each ACK reception. If a packet is dropped (detected by timing out the ACK, or by receiving multiple similar ACKs), the *ssthreshold* is set to $cwnd/2$ and the *cwnd* is reset to 1 [5].

Fig. 6 is a “zoom” of WT_2 *congestion window* during the period [153,154]. This figure also shows TCP-ACK packets reception, the RTS dropping and the *contention window* sizes[8].

At each ACK packet arrival, the *congestion window* increases by $1/cwnd$ as we are in the *congestion avoidance* period, and it never decreases because TCP never times out for an ACK reception: dropped RTSs, for TCP-ACK and data packets, are retransmitted by the MAC sub-layer much faster than the TCP timeout.

[7]Even if *cwnd* is a byte counter in TCP, we express *cwnd* in packets for convenience.
[8]In *NS*, the network simulator that we used, it starts with 31 and the sequence is: 31, 63 , 127... ($2^i - 1$).

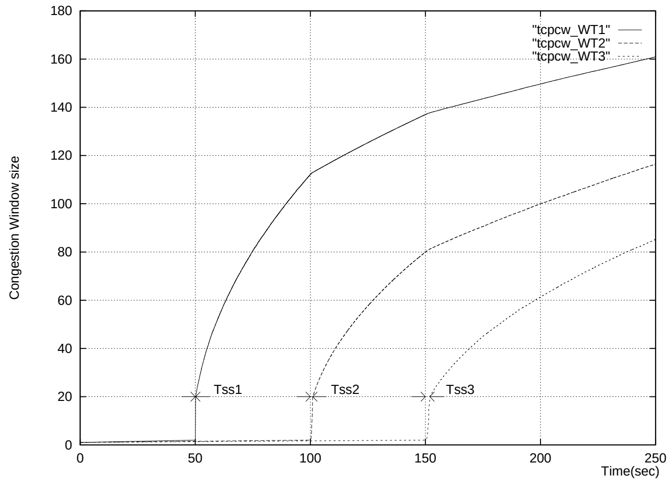


Fig. 5. TCP congestion window sizes.

When two or more WTs are used simultaneously, the delay between two TCP-ACK packets reception is obviously higher than when using a single WT (more collisions, less free channel periods etc.). Therefore the TCP *congestion window* increases at a slower rate (as seen in Fig. 5) and the slope is lower. This can also be noticed when comparing the WTs' respective *slow start* periods Tss_1, Tss_2 and Tss_3 shown in Fig. 5. Surely, these delays not only affect the $cwnd$, but the data rate too. In fact, when using TCP the data rate is $\lim_{t\to\infty} \frac{1}{t\times RTT}\int_0^t cwnd\ dt$, where RTT is the Round Trip Time [13]. Last, we should note that a TCP source won't receive the ACK of a packet if:

- after several RTS attempts, the data packet has been dropped by the MAC sub-layer.
- after several RTS attempts, the ACK has been dropped by the MAC sub-layer.
- either the data packet or the ACK did not reach its destination, because of *noise* on the channel.

A severe or busy channel could lead to such scenario:

Consider the case where WT_1 uses TCP while WT_2 and WT_3 use UDP flows of the same packet size as TCP. Even though each of the CBR/UDP flows is configured to “fill” the available data rate separately, we see that WTs equally share the available data rate. No TCP timeouts were observed and the contention window keeps increasing during the simulation time. Even when we increase the number of UDP flows in WT_2 and WT_3, we observe no effect on the contention window of WT_1: available channel data rate is shared among WTs and not among different flows. Several flows in a single WT share the same MAC sublayer and so they have the effect of a single flow towards other WTs. Decreasing (resp. increasing) the CBR packet sizes in WT_2 and WT_3 would decrease (resp. increase) the TCP *cwnd* slope in WT_1.

To force TCP timeouts, we increased the number of TCP[9] flows in WT_1 from 1 to 2 and 3, while WT_2 and WT_3 use UDP flows.

When two TCP flows use the same MAC sublayer, each of them will have longer delays before accessing the channel than when acting alone. This reduces the slope of the *cwnd* considerably. Adding a third TCP flow in the same WT introduces more delays for channel access, causing TCP timeouts before receiving the waited ACK.

A similar observation is made on TCP *cwnd* (in WT_1) when we increase the number of WTs from 3 to 13, using either UDP or TCP. When the number of WTs is large enough, TCP may also time out after several consecutive collisions. Note that there is no possible congestion at the AP or the fixed host in our simulations.

IV. Differentiation mechanisms

As mentioned in the introduction, in order to give WTs either statistical or absolute QoS guarantees, we can get differentiated services between WTs by giving them different QoS parameters.

When we aim to introduce priorities in the IEEE 802.11 using the DCF (Distributed Coordination Function)[10], several parameters can be considered, among which:

1. Backoff increase function: Each priority level has a different backoff increment function.
2. DIFS: Each priority level is assigned a different DIFS, after which it can transmit its RTS or data packet.
3. Maximum frame length: Each priority level has a maximum frame length allowed to be transmitted at once.

In the following subsections we analyze them separately and show simulation results with corresponding mathematical analysis.

A. Backoff increase function

As we have seen in section II, (1):

$$\textit{Backoff\_time} = \lfloor 2^{2+i} \times rand() \rfloor \times Slot\_time$$

the only configurable term in this equation is 2^{2+i}. Our first attempt to introduce priority was to replace it by ${P_j}^{2+i}$ where P_j is a priority factor of the WT j. So, instead of multiplying the range by two at each retransmission, we multiply it by P_j.

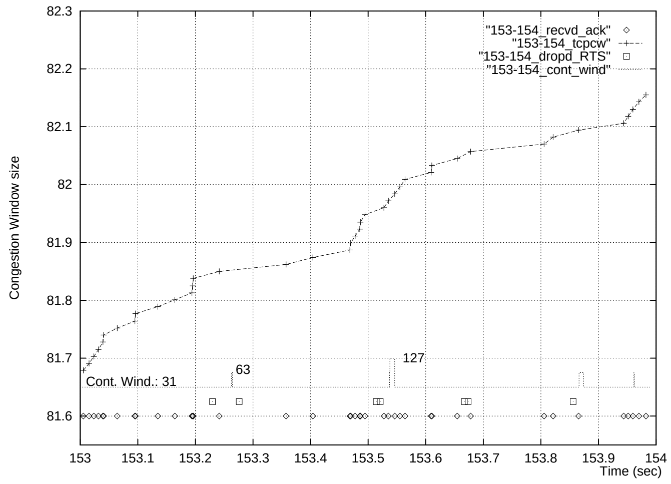


Fig. 6. Congestion window size of WT_2, during the period 153-154.

[9]“Full data rate” CBR/UDP flows added in WT_1 will consume the whole available data rate, without sharing it with TCP.

[10]When using PCF (Polling Coordination Function), introducing priority is centralized and somehow simple as in TDMA. We will not get into it in this paper.

(Here, the higher the priority factor is, the larger is the backoff range, the lower is the chance to first access the channel, the lower is the throughput.)

A.1 UDP flows

We used this scheme in the same network configuration as section III. WTs send UDP packets, using the RTS/CTS scheme. At second 50, WT_1 starts transmission with a priority factor P_1=2 (meanwhile WT_2 and WT_3 are idle). Then, at second 100, WT_2 starts transmission with P_2=6. Finally, at second 150, WT_3 starts transmission with P_3=8. The AP uses a priority factor of 2. Results are shown in Fig. 7, 8 and 9.

When only WT_1 is on, it uses the whole link data rate, exactly as in the case with no priorities (cf. section III). When WT_2 goes on (at second 100), the link is unequally shared between the two WTs, WT_1 having a higher data rate share (1.42:1). At second 150, the third WT goes on and the results show that the three WTs get different data rate shares. Obviously, we can change the ratios P_i/P_j $(i \neq j)$ to obtain other data rate shares with a wider range (so better priorities). But as this range increases (high priority ratios) the system becomes unstable, showing more data rate variability and higher jitters[11]. This instability is more visible with low priority traffics (high priority factors, as with WT_3). From the data rate point of view, the whole system efficiency gets slightly better when using more WTs, due to more sensing , "filling" more channel idle periods and getting the channel more busy (comparing the data rates of WT_1, WT_1 and WT_2 together, and all three WTs in Fig. 4: $(avr_1 + avr_2 + avr_3)_{150-250} > (avr_1 + avr_2)_{100-150} > (avr_1)_{50-100}$). As shown in Fig. 4 and 7 these data rate sums remain almost the same after introducing the priority scheme.

A.2 TCP flows

Note that when we replace UDP by TCP in all WTs, the results are quite different: they show no considerable prioritization effect, and all three WTs almost equally share the data rate, as shown in Fig. 10. In fact, TCP is an adaptive transport protocol based on the feedback control embedded in the reception of ACK packets. In both *Slow Start* and *Congestion Avoidance* periods, TCP sends new data packets only at ACK reception. There are two reasons that explain why using *Backoff_time* is not efficient for differentiating TCP flows:

[11] High delays are caused by the channel overload, even with a single WT.

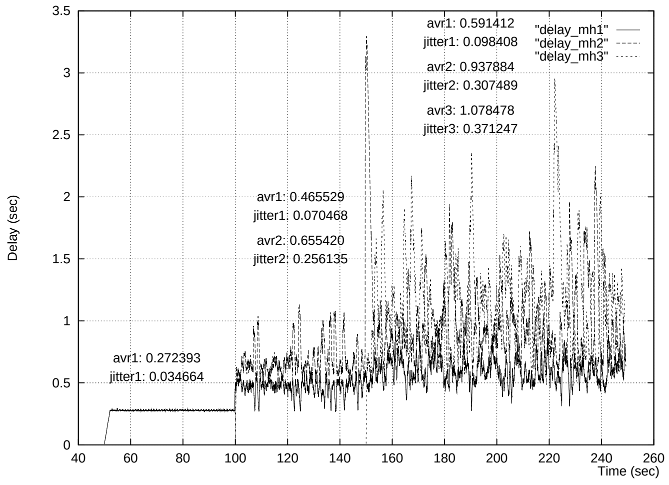


Fig. 8. Delays using UDP with priorities

First, the AP sends all TCP-ACKs for all WTs using the same priority (the highest in our simulations) as our differentiation is on a per station basis, not per-flow basis. In a per-flow differentiation scheme, the AP would have to look into the header of each packet to check the destination address/port. This gives additional load for the AP. It could be also the case that differentiation is made on a packet basis, which supposes that each packet has a priority field that sets the differentiation parameters (similar to DiffServ [14]). The additional field causes overhead for short packets. These approaches are left for future work.

Second, the backoff prioritization mechanism works only if a WT does not receive any CTS upon sending an RTS, it then increases its contention window. The contention window increases proportionally to the different priority factors P_i assigned to each WT. Therefore the probability of scaling the con-

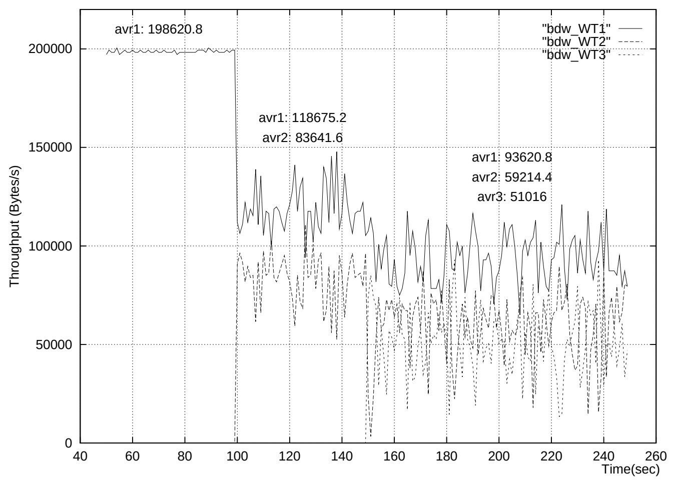


Fig. 7. Throughput using UDP with priorities

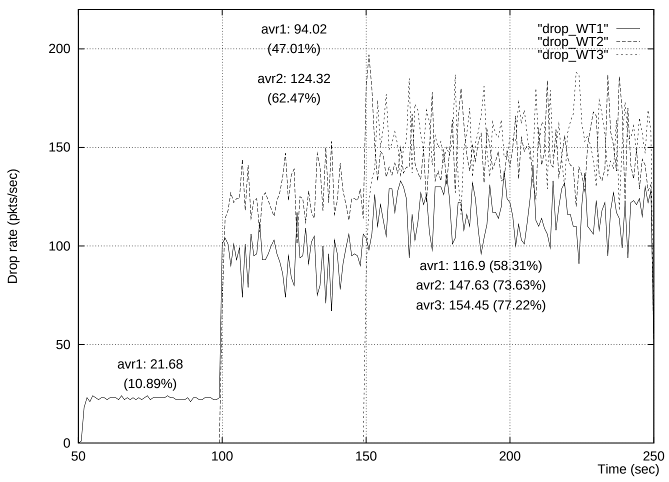


Fig. 9. Drop rates using UDP with priorities.

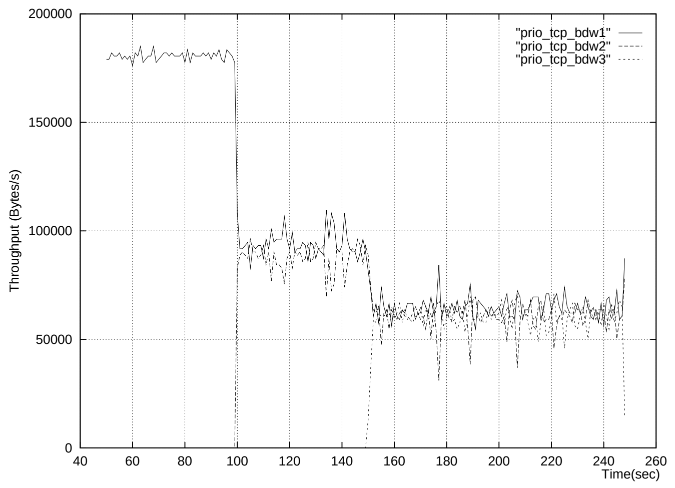


Fig. 10. Throughput using TCP with priorities.

tention window size is proportional to the probability of RTS collision which is proportional to the number of contending RTSs.

With TCP, during the congestion avoidance phase, a source waits for a new ACK before generating a new packet, i.e. generating an RTS, because of the congestion control algorithm. In our case, these ACKs are generated by a central entity, the AP. This AP tends to become the “coordinator”. If the AP is slow, most of the WTs will be waiting for an ACK and therefore the number of contending WTs will be lower. Respectively, if the AP is fast enough, each WT will receive an ACK and will be ready to contend to access the channel.

The number of contending WTs (i.e. ready to send an RTS), is shown in the birth-death chain of Fig. 11. The AP succeeds to send a TCP-ACK with a probability β_i, increasing the number of contending terminals. It fails sending its TCP-ACK with a probability α_i, thus increasing the number of waiting TCP-ACKs (the number of waiting packets is then reduced).

If the AP sends TCP-ACKs slowly (i.e. with a low priority), α_i are greater than β_{i-1}, and the chain drifts to the state 0: most WTs will be waiting for a TCP-ACK. This leads to a lower number of contending WTs (each with an RTS) and therefore to a low RTS collision probability. In this case our scheme does not work very well. No priority effect can be seen.

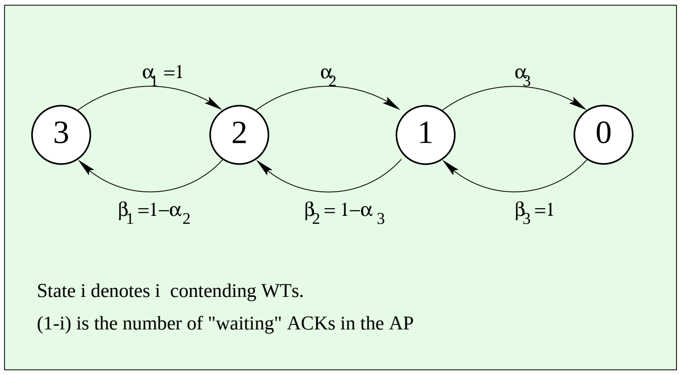


Fig. 11. State transition diagram for TCP generated packets.

TABLE I
CONTENTION WINDOW DISTRIBUTION FOR UDP

Cont. Win. Size	AP	WT_1	WT_2	WT_3
(CWmin)31	555	22718	9182	4740
62	34	922	-	-
124	0	45	-	-
186	-	-	830	-
248	0	5	-	527
496	0	0	-	-
992	0	0	-	-
(CWmax)1023	0	0	75	54

If the AP sends TCP-ACKs fast enough (i.e. it has a much higher priority than all the WTs), β_{i-1} are greater than α_i, and the chain drifts to state 3: TCP sources will receive their ACKs very quickly and most of them will be contenting for the medium. This leads to a higher number of RTSs contending to access the channel, which increases the probability of RTS collisions. In this case, the backoff priority scheme works well and the priority effect is much more visible.

To check out the second assumption, we counted each congestion window size occurrence when using UDP and TCP separately. Results are shown in Tables I and II: Using UDP, 37195 RTSs were sent by all terminals, out of which 2313 $(= 34 + 922 + 830 + 527)$ collided and the contention windows increased proportionally to each terminal’s priority factor, to become 62, 62, 186 and 248 respectively. Using TCP[12], contention windows did not increase as often as with UDP[13]. Note that, with TCP, the contention widow value 62 has been reached more often than with UDP. This is because of an additional node, the AP, contending to access the channel to send the TCP-ACKs.

This shows why introducing priorities in the backoff time incrementation has lower effect on TCP than on UDP. In other words, for the same P_i/P_j used with TCP and UDP, the resulting relative priority range width is much higher with UDP.

[12]With TCP, the AP also has to send TCP-ACK packets.

[13]For TCP, more packets are sent on the network because of the TCP-ACKs. One should compare the ratio: (number of backoffs)/(total number of RTSs) instead of the actual numbers in the tables.

TABLE II
CONTENTION WINDOW DISTRIBUTION FOR TCP

Cont. Win. Size	AP	WT_1	WT_2	WT_3
(CWmin)31	28969	17099	8794	5076
62	1885	1466	-	-
124	33	53	-	-
186	-	-	940	-
248	0	0	-	667
496	0	0	-	-
992	0	0	-	-
(CWmax)1023	0	0	29	22

A.3 Combined TCP-UDP flows

When the AP's priority is not high enough, simulations show that when we apply the backoff priority mechanism on different flow types (in different WTs) simultaneously:

- A UDP flow with high priority won't have considerable advantage over a single TCP flow with lower priority, and the common channel data rate is equally shared. In fact, the UDP RTSs are exposed to collision with AP RTSs, while TCP RTSs collide less often.
- On the other hand, when we apply the priority scheme to a WT with high priority using TCP flows, and another with low priority using UDP flows, high priority TCP flows get more throughput than low priority UDP ones. Backoff priorities enhance the TCP throughput without necessarily enhancing the *cwnd* size, as the RTT is considerably reduced relatively to the no-priority scheme.

A.4 Mathematical analysis

In this section we present a mathematical analysis of the simulation results shown in Fig. 7. The analysis aims to explain the data rate shares and collision probability in the second period (seconds 100 to 150) when using UDP. Similar but more complex reasoning can be applied to the third period.

During the second period (seconds 100 to 150), where only WT_1 and WT_2 are transmitting at full data rates, each of the WTs' data rate share is proportional to its probability to access the channel, i.e. its random backoff value is lower than the other's ($DIFS + Backoff_1 < DIFS + Backoff_2$). This is similar to comparing two random variables (r.v.) X and Y which bounds are [a,b] and [a,d] respectively. The probability of having $X < Y$ (thus WT_1 accessing the channel before WT_2) is given by:

$$P(X < Y) = \begin{cases} 1 - \frac{1}{2} \times \frac{b-1-a}{d-1-a} & \text{if } b \leq d \\ \frac{1}{2} \times \frac{d-1-a}{b-1-a} & \text{if } b > d \end{cases} \qquad (2)$$

Subtracting DIFS from a, b and d simplifies the equations without changing $P(X < Y)$. As time is slotted, where a time slot is equal to the contention window unit, a collision occurs[14] when $X = Y$, and both transmitted packets are dropped. The collision probability is given by:

$$P(X = Y) = \frac{1}{max(b, d)} \qquad (3)$$

Initially, both ranges [a,b] and [a,d] are equal to [0, CWmin], b and d denote the contention window sizes cw_1 and cw_2 of WT_1 and WT_2 respectively. As contention windows increase at each collision and decrease at each successful transmission, the combination of subsequent cw_1 and cw_2 values give the 21-state transition diagram of Fig. 12 and 13. Multiplying the probability of WT_1 success (i.e. $P(X < Y)$ given in (2)) in each state by the probability of that state, then summing over all 21 states, gives the WT_1 data rate share[15] (0.59 in this case). Similar computations give the WT_2 data rate share and collision probability.

[14]This is why b and d where decreased by 1 in (2)

[15]Note that routing packets are not taken into consideration in this analysis, but surely are considered in the simulation, which results in a slight difference between simulation and mathematical results.

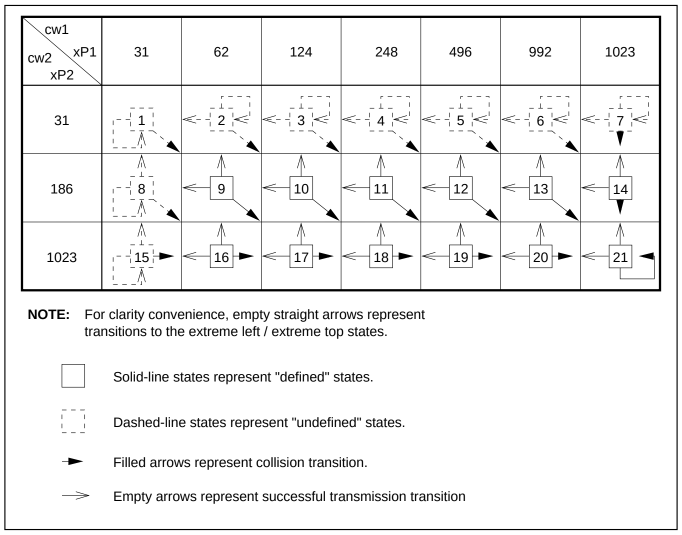
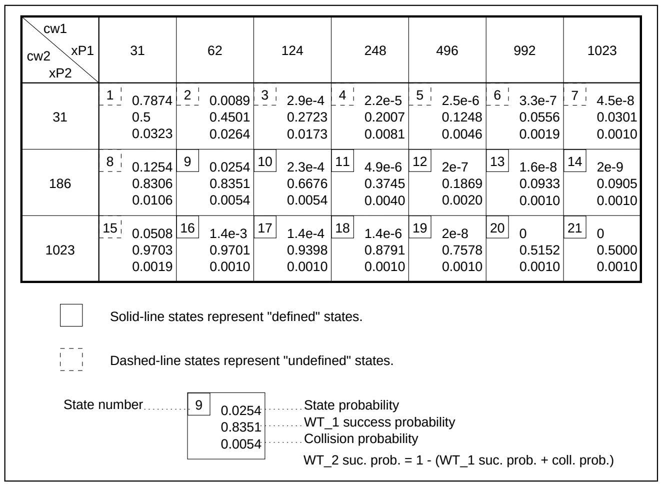

cw1
xP1
cw2
xP2
31
62
124
248
496
992
1023
31
186
1023
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
NOTE: For clarity convenience, empty straight arrows represent transitions to the extreme left / extreme top states.
Solid-line states represent "defined" states.
Dashed-line states represent "undefined" states.
Filled arrows represent collision transition.
Empty arrows represent successful transmission transition

Fig. 12. Contention windows state transition diagram

The transition diagram of Fig. 12 helps finding each state probability and the inter-state transition probabilities: A filled arrow represents a transition due to a collision, after which both contention windows are multiplied by their respective priority factors P_i. In this case, we called the target state a "defined" state, where both *backoff_times* are re-computed. *Defined* states, in which both cw_1 and cw_2 correspond to the indicated values, are shown with a solid line. In *defined* states, applying (2) and (3) to these values give us the transition probabilities.

On the other hand, and empty arrow indicates a transition due to a successful transmission. In this case, the winning WT resets its contention window (to 31)[16], while the other WT keeps reducing its backoff. This is represented with "undefined" states surrounded by dashed lines. An *undefined* state has one reset (31) contention widow size which bounds the new *backoff_time*

[16]For clarity reasons, empty straight arrows represent transitions to the extreme left / extreme top states and not to adjacent states. e.g. in state 19, if WT_1 succeeds accessing the channel, the transition is made to state 15, not state 18.



cw1 xP1 / cw2 xP2	31	62	124	248	496	992	1023
31	1 0.7874 0.5 0.0323	2 0.0089 0.4501 0.0264	3 2.9e-4 0.2723 0.0173	4 2.2e-5 0.2007 0.0081	5 2.5e-6 0.1248 0.0046	6 3.3e-7 0.0556 0.0019	7 4.5e-8 0.0301 0.0010
186	8 0.1254 0.8306 0.0106	9 0.0254 0.8351 0.0054	10 2.3e-4 0.6676 0.0054	11 4.9e-6 0.3745 0.0040	12 2e-7 0.1869 0.0020	13 1.6e-8 0.0933 0.0010	14 2e-9 0.0905 0.0010
1023	15 0.0508 0.9703 0.0019	16 1.4e-3 0.9701 0.0010	17 1.4e-4 0.9398 0.0010	18 1.4e-6 0.8791 0.0010	19 2e-8 0.7578 0.0010	20 0 0.5152 0.0010	21 0 0.5000 0.0010

Solid-line states represent "defined" states.
Dashed-line states represent "undefined" states.
State number 9 0.0254 0.8351 0.0054
State probability
WT_1 success probability
Collision probability
WT_2 suc. prob. = 1 - (WT_1 suc. prob. + coll. prob.)

Fig. 13. Contention windows state transition diagram: numerical values

value, and the second *backoff_time* depends on the previous states. This makes the outgoing transition probabilities function of several previous states, hence the chain is not a Markov chain.

The unknown *backoff_time* bound in *undefined* states could be replaced by the expected contention window size, taking into consideration previous states probabilities and the corresponding transition probabilities. Applying (2) and (3) to each state gives a set of equations which, once solved, gives the probability of each state.

One major observation on this chain is that it strongly drifts to state 1 (with probability 0.79), in which both contention windows are reset to $CWmin$, both equal 31. This fact makes the data rate shares slightly dependent of the P_1/P_2 values. To deal with this, we considered $CWmin$ differentiation, in which P_1/P_2 values strongly influence the data rate shares. The resulting data rate difference can be clearly seen in the simulations, when using UDP or even TCP flows.

B. DIFS

We have seen in the previous paragraph that using *backoff_time* to differentiate wireless users does not always apply to TCP flows. An alternative solution would be to use DIFS for differentiation.

As shown in section II, IEEE 802.11 ACK packets get higher priority than RTS packets, simply by waiting SIFS which is shorter than DIFS (for RTS). We'll use the same idea to introduce priorities for data frames (in the basic scheme) and for RTS frames (in the RTS/CTS scheme). In this approach we give each priority level a different DIFS, say $DIFS_j$ where $DIFS_{j+1} < DIFS_j$. So the WTs having priority j will wait $DIFS_j$ idle period before transmitting the packet. To avoid same priority frames collision, the backoff mechanism is maintained in a way that the maximum contention window size added to $DIFS_j$ is $DIFS_{j-1} - DIFS_j$ as illustrated in Fig. 14. This ensures that no WT of priority $j+1$ has queued frames when WT of priority j starts transmission. Low priority traffic will suffer as long as there are high priority frames queued. It could also be the case that the maximum random range (RR_j) after $DIFS_j$ can be made greater than $DIFS_{j-1} - DIFS_j$, so the previous rule becomes less severe. In this case, a packet which failed to access the channel at the first attempt will "probably" have its priority reduced after consecutive attempts, depending on the DIFSs and RRs values. This technique may be useful for realtime application, where we have more constraints on delays than on packet drops. Simulation results show the following:

- This mechanism offers a very wide range of relative priority: It can be a 1:1 when DIFSs are equal and RRs are equal. The relative priority can be infinite when $DIFS_j \geq (DIFS_{j+1} + RR_{j+1})$.
- Applying DIFS differentiation shows no efficiency loss, as seen in Fig. 15 (here, the packet size is 2312 bytes).
- For the same $DIFS_j$ sets, UDP shows more priority effect (e.g. throughput ratios) than TCP. Here there is no backoff problem with TCP (as when applying *backoff_time* differentiation), but TCP-ACK packets of several WTs are still sent with the same priority, which reduces the priority effect. When we accelerate TCP-ACK transmission by reducing the AP DIFS, prioritization becomes more visible.
- For TCP flows, as we increase the AP DIFS, the relative priority decreases.
- We can apply this mechanism to give UDP priority over TCP (which was not always applicable with *backoff_time* differentiation) and vice versa, same $DIFS_j$s result in the same throughput ratios.

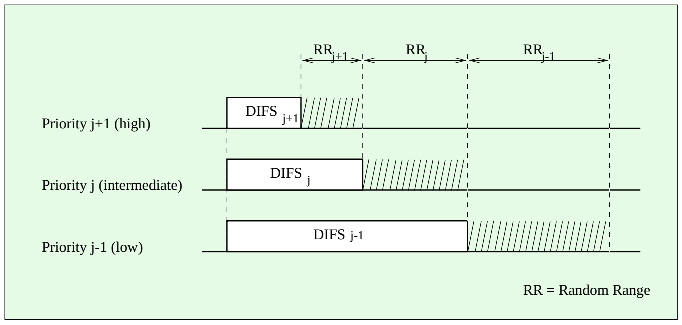


Fig. 14. Including priority in DIFS

B.1 Mathematical analysis

In order to find the interpretation for the data rate shares of the various WTs when using UDP, let us start by analyzing the second period (with two active WTs), then we'll move to period three and generalize the analysis.

With two active WTs, and as packet types are equal, we can say that the data rate share of a given WT (say WT_1) is equal to the probability that WT_1 accesses the channel first. That is the corresponding (*DIFS + backoff*) value is less than the others. This leads us to the problem of two random variables (r.v.) X_1 and X_2 with different bounds [a,b] and [c,d] respectively, uniformly distributed over these ranges (see Fig. 16(a)). We can easily show that, the probability of having $X_1 \leq X_2$ is:

$$P(X_1 \leq X_2) = \begin{cases} 1 - \left(\frac{1}{2} \times \frac{b-c}{d-c} \times \frac{b-c}{b-a}\right) & \text{if } b \geq c \\ 0 & \text{if } b \leq c \end{cases} \quad (4)$$

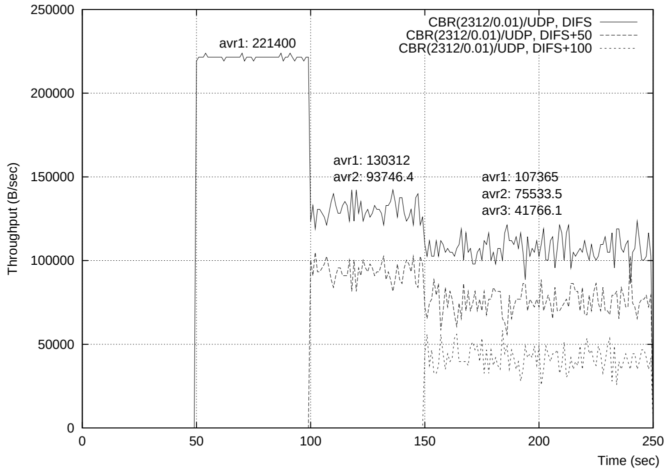


Fig. 15. Using CBR/UDP, $DIFS_{BS} = 50\mu s$, $DIFS_{WT_1} = 50\mu s$, $DIFS_{WT_2} = 100\mu s$, $DIFS_{WT_3} = 150\mu s$

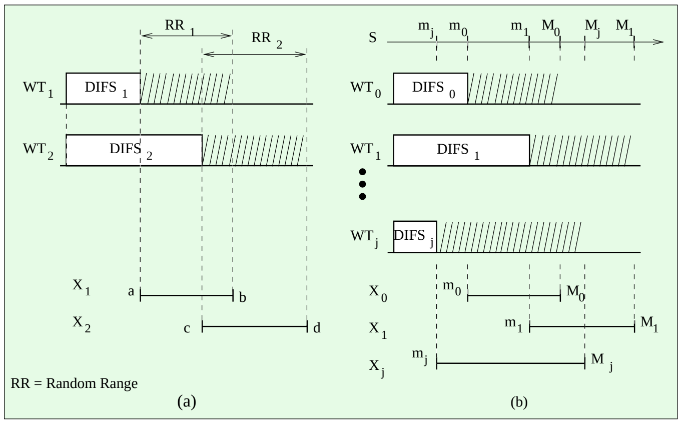


Fig. 16. Corresponding r.v. for DIFS

This equation complies, with a difference of just 0.7%, to the data rate shares of WT_1 and WT_2 of our simulation during the second simulation period (seconds 100 to 150). In fact, The initial contention window size is 31, a $Slot\_time$ is $20\mu s$ which gives a random range of $620\mu s$, for both of the WTs. $DIFS_{WT_1} = 50\mu s$ and $DIFS_{WT_2} = 100\mu s$, together with the random ranges, give a=50, b=670, c=100 and d=720 for (4). The slight difference of 0.7% between (4) and the simulation results can be due to:

- the non perfect random number generator of the simulator.
- not taking subsequent backoffs into consideration in the mathematical analysis.

In order to apply the analysis to period 3 (seconds 150 to 250), we have to consider more than two r.v. and the analysis becomes less intuitive and more complex. We also generalize the case to any disposition of the various r.v. bounds, as in Fig. 16(b).

Let $N+1$ be the number of WTs (as well as the number of r.v.). Let m_i and M_i be the lower bound and the upper bound respectively of r.v. X_i. Let S be the ordered set of the bounds (lower and upper) of all the r.v. For all $i = 0, .., N$, let S_i be the ordered set of the bounds (lower and upper) of all the r.v. such that $s_i \in S_i, m_i \leq s_i < M_i$.

Given a r.v. X_0, we show that the probability that X_0 is less than all other r.v. $X_i, \forall i \neq 0$ is given by:

$$P(X_0 \leq X_{k\neq 0}) = \sum_{s_j \in S_j, j=0,..,N} \left(\prod_{i=0}^{N} \frac{s_i^+ - s_i}{M_i - m_i} \times \delta_s \right) \quad (5)$$

where,
s_i^+ is the element succeeding s_i in S and

$$\delta_s = \begin{cases} 1 & \text{if } s_0^+ \leq s_i \ \forall i \neq 0 \\ 0 & \text{if } s_0 \geq s_i^+ \ \exists i \neq 0 \\ 1/(n+1) & \text{otherwise, where } n \text{ is the number} \\ & \text{of "i"s where } s_i = s_0 \end{cases}$$

Equation (5) is useful to explain the data rate shares of several WTs when using DIFS differentiation. Inverting this equation is useful to determine $DIFS_i$ function of the desired data rate shares among the WTs, e.g. when using *DiffServ* or to optimize *End-to-End* parameters. The number of operations (divisions and multiplications) needed when applying (5) directly grows as N^N, which shows that further computing optimization is needed to be applied in realtime admission control for large numbers of WTs. Equation (5) is not applicable to WTs with TCP flows, where we should take the base station flow (TCP-ACKs) into consideration, as well as packets of different sizes, which adds new factors to the equation.

C. *Maximum frame length*

The third mechanism that can be used to introduce service differentiation into IEEE 802.11 is to limit the maximum frame length used by each WTs. Here, we should distinguish between two possibilities:

- Either to drop packets that exceed the maximum frame length assigned to a given WT (or simply configure it to limit its packet lengths), or
- To fragment packets that exceed the maximum frame length. As mentioned in section II, this mechanism is actually used to increase transmission reliability, we'll also use it for differentiation.

Fig. 17 shows how a WT would send a fragmented packet. We can see there are no RTSs between packet fragments, so a given WT keeps sending its packet fragments as long as it is receiving the corresponding ACKs. Meanwhile, all other WTs are "quiet." This leads us to almost the same data rate shares as if there were no fragmentation, unless there is fragment loss (thus a new RTS), due to a noisy channel for example. In the case of no fragment loss, both above cases can then be described by the former one, i.e. limiting packet lengths to a given value.

Simulations showed, as one would intuitively expect, that data rate shares are directly proportional to the maximum frame lengths allowed for each WT. That is, for a given WT_0:

$$\frac{B_0}{\sum_{i=1}^{N} B_i} = \frac{L_0}{\sum_{i=1}^{N} L_i} \quad (6)$$

where B_i and L_i are the throughput and the maximum frame length respectively of the WT_i. This gives an infinite prioritization range, with no cost over system stability for high priority ratios P_i/P_j. Equation (6) shows no computing or inversion problems in order to apply it in realtime admission control with *DiffServ* or *End-to-End* optimization. Note that (6) applies to WTs with UDP flows as well as to TCP flows. This scheme also

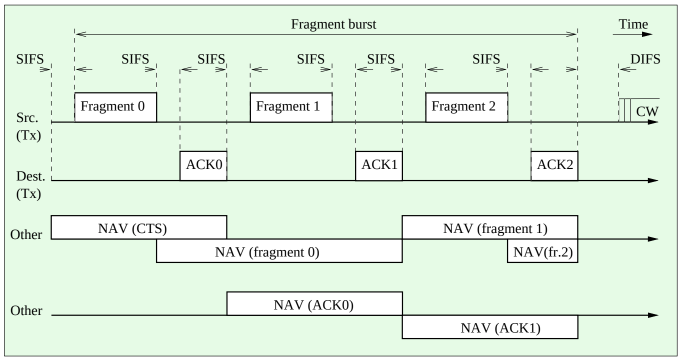


Fig. 17. Packet fragmentation

applies properly to give TCP flows priority over UDP one and vice versa.

D. Noisy channels

In the previous sections we only considered perfect channels, without noise (interference, fading or multipath). This section provides a brief (due to space limitations) description of our simulation results with noisy channels. Simulations show that noise changes the performance of the three described schemes. Consider the *packet error rate* (PER) to be

$$PER = 1 - (1 - BER)^L$$

where BER is the *bit error rate*, and L is the packet length (in bits). We first considered a channel with a BER of 10^{-6}. This leads to a low PER (for a 1100-bytes packet, the PER is 0.9%), and no considerable effects can be seen on any of the three mechanisms.

However, when we apply a $10^{-4} BER$ to all packets, simulations show that:

- With no priorities, the data rates of all WTs drop almost proportionally to the PER, and the data rate ratios remain the same, equal to unity, for both UDP and TCP flows respectively.
- With *backoff_time* priorities, two effects can be seen: The first is data rate drop due to packet errors. The second is that the data rate ratios increased dramatically, even with TCP flows (such differentiation couldn't be seen without channel errors). When an RTS is corrupted, the sending WT times out waiting for the corresponding ACK. The WT then increases its contention window for collision avoidance as if there was a collision. As different WTs increase their contention windows differently, because they have different priorities, they get different data rate shares. As a result, the priority that each WT gets depends directly on the channel conditions. This property is of course not desirable.
- With *DIFS* priorities, the data rates drop proportionally to the PER, and the relative priority of each WT remains the same.
- With *maximum frame length* priorities, long packets are more likely to be corrupted than short ones. This decreases the priority effect of the *maximum frame length* scheme.

V. Future work

Beyond the results presented in this paper, future work should address the following issues:

- *Per flow / per packet differentiation*. In this paper we only worked on differentiations based on the transmitting WT. It can also be the case where each WT transmits flows/packets with different priorities. This solves the problem, shown earlier in this paper, of TCP-ACKs sent by the base station independently of the destination WT.
- *Mapping DiffServ to MAC differentiation* [15]. i.e. How must DiffServ parameters be mapped to MAC differentiation in order to get the optimal performances, including end-to-end ones.

VI. Conclusion

This paper presents some results of our work on introducing service differentiation mechanisms into IEEE 802.11 MAC sublayer. We propose a scheme based on the contention window variation, another based on DIFS variation and a third one based on the maximum frame length allowed to each wireless terminal. The first scheme consists of scaling the contention window according to the priority of each flow or user. We show via simulations that this scheme performs well with UDP but does not always work with TCP flows. The second mechanism, which consists of assigning different DIFSs for different priority WTs, showed better results as it can be applied to TCP and UDP flows. The third mechanism, which assigns different maximum frame sizes to different priorities, showed less complex results and works well with both kinds of flows too. The three different mechanisms do not introduce any efficiency loss: the data rate sums remain almost the same after introducing the priority schemes. On the other hand, the whole system is much less stable with *backoff* priority, but keeps the same stability level with *DIFS* and *maximum frame length* priorities. We also show that in noisy environments, the *backoff* and *maximum frame length* schemes do not perform well anymore, while DIFS based schemes' performances remain unchanged. The data rate ratios increase for *backoff* mechanisms due erroneous backoffs. These ratios decrease for *maximum frame length* mechanism, but they keep the same values with *DIFS* mechanism which shows to have the best general properties among the three. As a final conclusion we would recommend to use the DIFS based schemes for service differentiation.

References

[1] *Specification of the Bluetooth system*, http://www.bluetooth.com/.

[2] Bob O'Hara and Al Petrick, *IEEE 802.11 handbook. A designer's companion.*, IEEE Press, 1999.

[3] A. Wolisz F. H.P. Fitzek, "QoS support in wireless networks using Simultaneous MAC packet transmission (SMPT)," in *ATS*, April 1999, http://www-tkh.ee.tu-berlin.de/bibl/ours/WWW_ATS.ps.

[4] Larry Taylor, *HIPERLAN white paper*, June 1999, http://www.hiperlan.com/.

[5] Van Jacobson, "Congestion avoidance and control," in *Proceedings of SIGCOMM '88*, August 1988, pp. 314–329.

[6] J. Postel, *User datagram protocol*, Request For Comments 768.

[7] Imad Aad and Claude Castelluccia, "Priorities in WLANs," Tech. Rep., INRIA, 2001.

[8] Brian P. Crow, Indra Widjaja, Jeong Geun Kim, and Prescott T. Sakai, "IEEE 802.11 wireless local area networks," *IEEE Communication magazine*, September 1997.

[9] Andrew S. Tanenbaum, *Computer networks*, Prentice-Hall, third edition, 1996.

[10] Harshal S. Chhaya and Sanjay Gupta, "Performance modeling of asynchronous data transfer methods of IEEE 802.11 MAC protocol," *Wireless networks*, vol. 3, 1997.

[11] J. Weinmiller, H. Woesner, JP Ebert, and A. Wolisz, "Analyzing the RTS/CTS mechanism in the DFWMAC media access protocol for wireless LANs," in *IFIP TC6*, 1995.

[12] "http://www.isi.edu/nsnam/ns/," .

[13] Kostya Avrachenkov Eitan Altman and Chadi Barakat, "A stochastic model of tcp/ip with stationary random losses," in *Proceedings of ACM SIGCOMM*, Stockholm, Sweden, August 2000.

[14] K. Nichols, S. Blake, F. Baker, and D. Black, *Definition of the differentiated services field in the IPv4 and IPv6 headers*, Request For Comments 2474.

[15] Torsten Braun, Claude Castelluccia, Günter Stattenberger, and Imad Aad, "An analysis of the DiffServ approach in mobile environments," http://www.inrialpes.fr/planete/people/MobiQoS/paper2.ps , unpublished, April 1999.