DYNAMIC AND EFFICIENT TUNING OF IEEE 802.11 FOR MULTIMEDIA APPLICATIONS

Fethi Filali
Institut Eurécom, Mobile Communications Department, 2229 Route des Crêtes, BP-193
06904 Sophia-Antipolis, France, filali@eurecom.fr

Abstract - In order to exploit the properties of multimedia applications in IEEE 802.11-based wireless networks in more appropriate ways, we interest in the dynamic and efficient tuning of the MAC protocol by taking into account the inherent characteristics of this kind of applications.

We focus here on the try limit parameter used at the 802.11 MAC layer to limit the number of retransmissions of a data frame by a source until the reception of a link-level acknowledgment from the destination. Based on analytical preliminaries, we determine the optimal value of this parameter in order to achieve an expected average end-to-end delay or a maximum packet loss rate. Moreover, an approximated value of the try limit parameter optimizing both QoS criteria (delay and packet loss rate) could also be obtained. Furthermore, these values are approximated dynamically using a simple algorithm that is easily implementable in 802.11 protocol.

Keywords - IEEE 802.11-based networks, adaptive tuning of 802.11 MAC protocol, multimedia applications, cross-layer design.

I. INTRODUCTION

Support for video and audio applications is important in single and multihop mobile wireless networks whether they are used as extensions to the Internet or not. Due to the variability of response of the air medium and that multimedia applications are sensitive to lost packets, delayed packets and jitter, it is accepted that such applications must dynamically adapt to network conditions, taking advantage of the different content representations achieved by advances in coding. On the other hand, it is also well known that in 802.11-based wireless networks, the cross-layer paradigm [3], [4] should be considered when developing applications on top of them. Hence, we believe that it is important to tune the parameters used by lower layers, such those of the Medium Access Control (MAC) layer, according to the expected requirements of user applications. However, the IEEE 802.11 MAC mechanism is not always aligned to the requirements and properties of the application that needs to be supported in the upper layers.

In this paper, we explore how the attributes of multimedia applications should be taken advantage by the MAC in order to efficiently use the wireless medium; this may be achieved by properly tuning the functional parameters of IEEE 802.11 to adjust to the application specific requirements.

The 802.11 MAC's ARQ (Automatic Repeat reQuest) is a stop-and-wait ARQ with positive acknowledgments after each packet [1]. The detection of lost frames is achieved by an ACK timeout. The standard uses two different retry limits for short and long frames specifying how many times a packet should be retransmitted before it is dropped, can be set for each packet. The MAC ARQ is more effective for transmission of video over WLANs than the application layer ARQ because it incurs a much shorter delay but only with a reasonable number of retransmissions. In this work, we explore the issue of determining the optimal value of the retry limit parameter in order to achieve a given value of the end-to-end delay and/or an expected packet loss rate. We derive and validate, through simulations, a set of analytical inequalities which are used to approximate this value.

The remainder of this paper is organized as follows: We explore our tuning mechanism of the try limit parameter in Section II. Some implementation details of our proposal in the 802.11 MAC protocol are provided in Section III. The performance evaluation of our scheme is reported in Section IV. In Section V, we discuss the advantages of our proposal. Section VI concludes this paper and outlines our future works.

II. EFFICIENT TUNING OF THE TRY LIMIT PARAMETER

A. Analytical preliminaries

The probability P_s that an occurring transmission is successful is given by the probability that a station is transmitting and the remaining $n-1$ stations remain silent, conditioned on the fact that at least one station transmits: $P_s = \frac{n\tau(1-\tau)^{n-1}}{1-(1-\tau)^n}$ where n is the number of stations and τ is the probability that a station transmits during a slot time. Since no hidden nodes are considered, collisions take place in the case when two or more contending stations choose the same backoff slot to transmit. The transmission time of a frame starts when a frame becomes head of station's queue and ended when an positive acknowledgment is received. Assuming that the frame drop probability is very low and can be neglected, the average frame delay can be given by:

$$E[D] = E[N_c](E[BD] + T_c + T_o) + (E[BD] + T_s) \quad (1)$$

where $E[N_c]$ is the number of collisions of a frame until its successful reception, $E[BD]$ is the average backoff delay that the station chooses before accessing the channel under busy channel conditions, T_o is the time that a station has to wait when its frame transmission collides, before sensing the channel again. This time out depends on the access method and equals to either $SIFS + ACK_{timeout}$ for the basic CSMA/CA scheme or to $SIFS + CST_{timeout}$ when using the RTS/CTS mechanism. Note that according to IEEE Std. 802.11-1999, the ACK timeout is defined as SIFS time, plus ACK transmission time, plus a slot time. T_s is the average time that the channel is captured with a successful transmission, and T_c is the average time that the channel is captured by stations which collide. The values of T_s and T_c depend on the channel access method and are defined as $T_s^{ack} = H + P + \delta + SIFS + ACK + \delta + DIFS$ and $T_c^{ack} = H + P + \delta + DIFS$ for the basic CSMA/CA and as $T_s^{rcts} = RTS + \delta + SIFS + CTS + \delta + SIFS + H + P + \delta + SIFS + ACK + \delta + DIFS$ and $T_c^{rcts} = RTS + \delta + DIFS$ when using RTS/CTS virtual carrier sensing mechanism. $H = PHY_{hdr} + MAC_{hdr}$ is the frame header transmission delay, P is the packet payload transmission delay, and δ is the propagation delay. We assume that all frames have the same fixed size.

The average number of collisions before transmitting a frame can be calculated by using the probability P_s that a transmission is successful. If the probability P_s is known, the average number of retransmissions is $1/P_s$, thus: $E[N_c] = \frac{1}{P_s} - 1$.

The average backoff delay depends on the value of its counter and the duration the counter freezes when the station detects transmissions from other stations. The time the counter is stopped is denoted by the random variable X and its average is given by,

$$E[X] = \frac{b_{0,0}}{6(1-p_b)} \frac{w^2(1-p-3p(4p)^m)+4p-1}{(1-4p)(1-p)} \quad (2)$$

where p_b is the probability that the channel is busy (there is at least one station transmits during a time slot: $p_b = 1-(1-\tau)^n$) and p is the probability that a frame collides.

We denote by F the time that the counter of a station freezes. When the counter freezes, it remains stopped for the duration of a transmission. This duration depends on the transmission success. So, in order to calculate the average time $E[F]$ that the counter remains stopped, we have to find $E[N_{Fr}]$, the average number of times that station detects transmissions before its counter reaches state 0. Based on $E[F]$, the average backoff delay of each station and on ($E[\Psi] = \frac{1}{p_b} - 1$), the mean number of consecutive idle slot times before a transmission proceeds, then $E[N_{Fr}] = \frac{E[X]}{max(E[\Psi],1)} - 1$ and $E[F] = E[N_{Fr}](P_sT_s + (1-P_s)T_c)$.

Eq. (1) gives $E[D] = (1+E[N_c])E[BD] + E[N_c](T_c + T_0) + T_s$.

The average backoff delay $E[BD] = E[X] + E[F]$ is given by:

$$E[BD] = E[X]\left(1 + \frac{P_sT_s + (1-P_s)T_c}{max(E[\Psi],1)}\right) - (P_sT_s + (1-P_s)T_c) \quad (3)$$

From the two previous equations, we can write:

$$E[D] = (1+E[N_c])\,E[X]\left(1 + \frac{P_sT_s+(1-P_s)T_c}{max(E[\Psi],1)}\right) + (1+E[N_c])(P_sT_s + (1-P_s)T_c + E[N_c](T_c+T_0) + T_s. \quad (4)$$

Replacing $E[X]$ by its value given by Eq. (2)

$$E[D] = (1+E[N_c])\left(1 + \frac{P_sT_s+(1-P_s)T_c}{max(E[\Psi],1)}\right)\frac{b_{0,0}}{6(1-p_b)} \frac{w^2(1-p-3p(4p)^m)+4p-1}{(1-4p)(1-p)} + (1+E[N_c]) (P_sT_s + (1-P_s)T_c + E[N_c](T_c+T_0) + T_s. \quad (5)$$

$$E[D] = E[X]B_1 + A_1 = \frac{b_{0,0}}{6(1-p_b)} \frac{w^2(1-p-3p(4p)^m)+4p-1}{(1-4p)(1-p)} B_1 + A_1 \quad (6)$$

where $A_1 = E[N_c](T_c+T_0) + T_s + (1+E[N_c])(P_sT_s + (1-P_s)T_c)$, $B_1 = (1+E[N_c])\left(1 + \frac{P_sT_s+(1-P_s)T_c}{max(E[\Psi],1)}\right)$ and

$$b_{0,0} = \frac{p_b+p(1-p_b)}{(1-p_b)} \frac{2(1-p_b)^2(1-2p)(1-p)}{(1-2p)(2(1-p_b)^2(1-p)+(p_b+p(1-p_b))(w+1))+pw(p_b+p(1-p_b))(1-(2p)^m)} \quad (7)$$

B. Optimizing m to minimize the average end-to-end delay

According to [5], τ is given by

$$\tau = \frac{A_2}{B_2 - a(2p)^m} \quad (8)$$

where $A_2 = 2(1-p_b)(1-2p)$, $B_2 = 2(1-p_b)^2(1-2p)(1-p) + (p_b + p(1-p_b))(1-2p)(w+1) + pw(p_b + p(1-p_b))$, and $a = pw(p_b + p(1-p_b))$.

From Eq. (7) and Eq. (8) , we can write $b_{0,0} = (p_b + p(1-p_b))(1-p)\tau = (p_b + p(1-p_b))(1-p)\frac{A_2}{B_2 - a(2p)^m}$.

Eq. (6) $\Leftrightarrow$

$$E[D] = \frac{(p_b+p(1-p_b))(1-p)}{6(1-p_b)} \frac{A_2}{B_2-a(2p)^m} \frac{w^2(1-p-3p(4p)^m)+4p-1}{(1-4p)(1-p)} B_1 + A_1 = d(m) \quad (9)$$

Assuming that d_{max} is the tolerated end-to-end delay, the optimal value of the try limit parameter is then given by:

$$\mathbf{m^*_{delay}} = \min\{\min\{\mathbf{m},\ \mathbf{d(m)} \leq \mathbf{d_{max}}\}, \mathbf{m_{max}}\} \quad (10)$$

where m_{max} is the maximum value of the number of retransmissions.

C. *Optimizing m to minimize the packet loss rate*

A transmitted frame collides when at least two stations transmit during a slot time, so the probability p that a frame collides is given by $p = 1 - (1-\tau)^{n-1}$. Thus, the drop packet probability p_{drop} after m retransmissions is given by $p^m = (1 - (1-\tau)^{n-1})^m$.

If the target packet drop rate is p_{max}, we would like to insure that $p_{drop} \leq p_{max}$. Thus, the loss rate constraint could be written as

$$\begin{aligned} p_{drop} = (1-(1-\tau)^{n-1})^m \leq p_{max} \\ \Leftrightarrow \tau \leq 1 - (1 - p_{max}^{\frac{1}{m}})^{\frac{1}{n-1}} \end{aligned} \tag{11}$$

$\Leftrightarrow$ (replacing τ by its value of Eq. (8))

$$\frac{A_2}{B_2 - a(2p)^m} \leq 1 - (1 - p_{max}^{\frac{1}{m}})^{\frac{1}{n-1}} \tag{12}$$

Finally, we get the following inequality

$$l(m) = \frac{A_2}{B_2 - a(2p)^m} + (1 - p_{max}^{\frac{1}{m}})^{\frac{1}{n-1}} \leq 1 \tag{13}$$

The optimal value of the try limit parameter m^*_{loss} is then given by:

$$\mathbf{m^*_{loss}} = \min\{\min\{\mathbf{m},\ \mathbf{l(m)} \leq \mathbf{1}\},\ \mathbf{m_{max}}\}. \tag{14}$$

<u>**Special case :**</u> <u>$p = 0.5$</u>

If $p = 0.5$, Eq. (12) $\Leftrightarrow$

$$\begin{aligned} \tfrac{B_2-a}{A_2} \geq \frac{1}{1-(1-p_{max}^{\frac{1}{m}})^{\frac{1}{n-1}}} \\ \Leftrightarrow p_{max}^{\frac{1}{m}} \geq 1 - (1 - \tfrac{A_2}{B_2-a})^{n-1} \end{aligned} \tag{15}$$

Then, we can get the following simple inequality of m

$$m \leq \frac{\ln(p_{max})}{\ln(1 - (1 - \frac{A_2}{B_2-a})^{n-1})} \tag{16}$$

In this case, the optimal value of the try limit parameter is given by:

$$\mathbf{m^*_{loss}} = \min\{\min\{\mathbf{m},\ \mathbf{m} \leq \frac{\ln(\mathbf{p_{max}})}{\ln(\mathbf{1} - (\mathbf{1} - \frac{\mathbf{A_2}}{\mathbf{B_2-a}})^{\mathbf{n-1}})}\},\ \mathbf{m_{max}}\} \tag{17}$$

D. *Optimizing m for delay and loss constraints*

When the multimedia application have both delay and loss constraints, the optimal value $m^*_{DelayLoss}$ of the try limit parameter which takes into account both constraints is the maximum of m^*_{delay} and m^*_{loss} computed by Eq. (10) and Eq. (14), respectively. Hence $m^*_{DelayLoss}$ is given by:

$$\mathbf{m^*_{DelayLoss}} = \max\{\mathbf{m^*_{delay}},\ \mathbf{m^*_{loss}}\}. \tag{18}$$

As we can observe the maximum value of $m^*_{DelayLoss}$ is m_{max}.

III. Implementation details

It is well know that the design of future wireless systems and in particular 802.11-based networks should follow the cross-layer paradigm [3], [4]. This means that a communication layer will have the capabilities to communicate with lower layers by sending to them its requirements. At the same time this layer should try to achieve the requirements of layers on the top of it.

In our case, multimedia applications should have the capability to send their QoS requirements such as end-to-end delay and packet drop rate to the MAC layer either directly or through transport or network layers. This can be done by specific system calls letting the MAC layer knowing the values of d_{max} and p_{max} as described in Section II.

We propose the following algorithm to incorporate our proposal in 802.11 MAC protocol:

Algorithm 1 Integration of our proposal in the 802.11 MAC protocol.

Variables:

- `n_(t)`: number of competing nodes at time t
- `tau_(t)`: the probability that a station transmits during a slot time at time t
- `p_(t)`: the probability that a frame sent at time t collides
- `pb_(t)`: the probability that the channel is busy at time t
- `slots_(t)`: the total number of slots at time t
- `sslots_(t)`: the total number of slots at time t where the node have sent a data packet
- `pmax_`: the maximum tolerated packet drop rate
- `dmax_`: the maximum tolerated end-to-end delay

Algorithm:

```
  /* get a data packet from the output buffer to be sent */
  pkt ← get_packet_from_output_buffer()
  /* get the QoS parameters of the corresponding application */
  pmax_ ← get_packet_drop_rate(pkt)
  dmax_ ← get_packet_delay(pkt)
  /* compute an approximation of tau_(t)*/
  tau_(t)=sslots_(t)/slots_(t)
  /*compute p_(t)*/
  p_(t)=1-(1-tau_(t))^(n_(t)-1)
  /*compute pb_(t)*/
  pb_(t)=1-(1-tau_(t))^n_(t)
  compute $m^*_{delay}$ using the method explained in Section II-B
  compute $m^*_{loss}$ using the method explained in Section II-C
  compute $m^*_{DelayLoss}$ using the method explained in Section II-D
```

Note that at a 802.11 node, the number `n_(t)` of its neighbors, used in the algorithm above, could be estimated at the MAC layer based on the broadcasted beacon packets which they send periodically.

IV. Numerical results

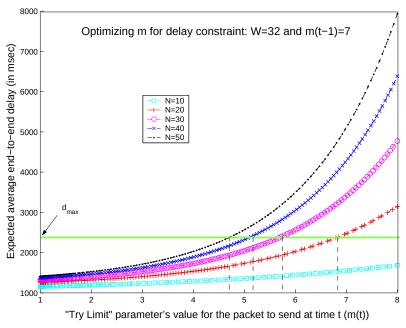
In this section, we investigate, using the Matlab tool, the performance of our proposal in terms of the average end-to-end delay and the loss rate. We consider a mobile ad hoc network with a variable number of nodes. Other simulation parameters are summarized in Table 1.

Table 1
Simulation parameters.

Data payload	1023	**ACK Rate**	1 Mbps
Physical header	16 bytes	**Data Rate**	1 Mbps
RTS packet size	20 bytes + Physical header	**RTS/CTS Rate**	1 Mbps
ACK header	14 bytes + Physical header	**Slot time**	20 μs
Propagation delay	1μs	**SIFS**	10μs
DIFS	50 μs	**CTS packet size**	14 bytes + Physical header
MAC header	16 bytes	**CW**	32

In a first experiment, we set the number N of nodes in the network successively to 10, 20, 30, 40, and 50. For each scenario, we show, in Fig. 1, the expected value of the end-to-end delay when varying the try limit parameter's value from 1 to 8.

Fig. 1
The expected end-to-end delay in function of the try limit parameter.



It is quite evident that, independent of the try limit parameter's value, the end-to-end delay increases with the number of competing nodes. On the other hand, for a given value of N, the expected end-to-end delay increases with the try limit parameter's value. Knowing the tolerated end-to-end delay d_{max}, the number of nodes at time t and the try limit parameter's value used at time $t-1$ (m(t-1)), we are able to determine the value of m which leads to an end-to-end delay equal or less than d_{max}. For example for $N = 40$ and $d_{max} = 2400$ $msec$, $m^{*}_{delay} = 5$ which gives an end-to-end delay equal to 2300 $msec$.

In Fig. 2, we show the optimal try limit parameter's value m^{*}_{delay} found for several values of the end-to-end tolerated delay. We can notice that for a given tolerated end-to-end delay and a number of nodes in the network, we are able to compute the minimum value of the try limit parameter that should be used by the MAC layer for the packet to be sent.

Fig. 2
Optimal try limit parameter for the delay constraint.

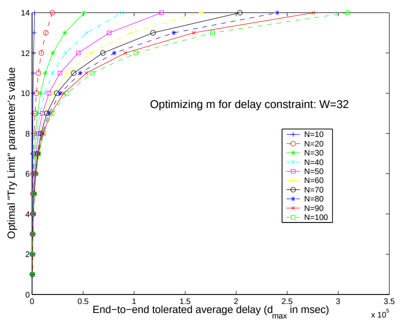
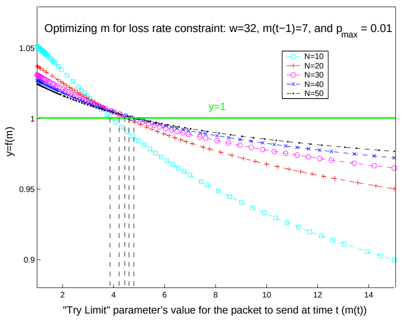


Fig. 3
Optimal try limit parameter for the loss rate. Function l(m) for a maximum loss rate equal to 0.01.



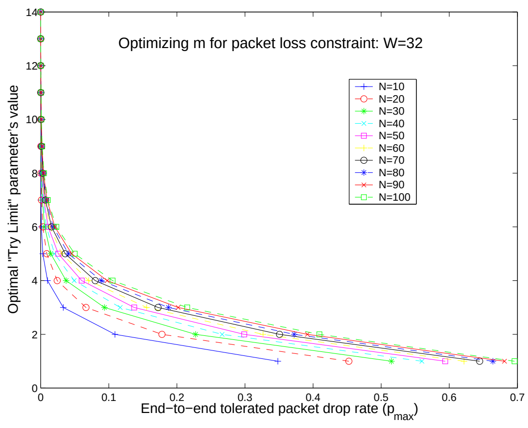
In a second experiment, we interest in the loss rate constraint. Again, we set the number N of nodes in the network successively to 10, 20, 30, 40, and 50. We plot in Fig. 3 the function $l(m)$ for different network sizes in terms of the number of active nodes and the function $y = 1$. We assume that the tolerated packet drop rate p_{max} is set to 0.01.

It is clear from the plots that the packet loss rate decreases when m increases and that, for a given value of m, when the loss rate increases with the number of nodes. The optimal value of m for each case corresponds to the intersection between the $l()$ function's plot and the function $y = 1$. For example for $N = 10$, we got $m^{*}_{loss} = 4$.

In Fig. 4, we show the optimal try limit parameter's value m^{*}_{loss} found for several values of the packet drop rate. We can notice that for a given packet drop rate and a number of nodes in the network, we are able to compute the minimum value of the try limit parameter that should be used by the MAC layer for the packet to be sent.

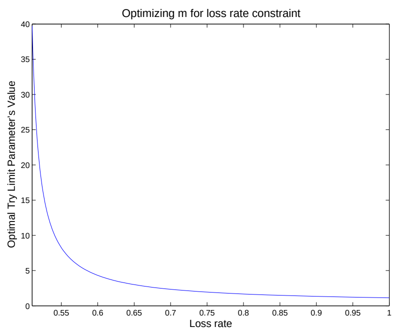
As we have explained in Section II-C, when $p = 0.5$ we are able to have a more comprehensive plots given that the optimal value of m should verify Eq. (16). In this case, we plot in Fig. 5 for $N = 10$, the variation of the optimal value

Fig. 4
Optimal try limit parameter for the packet loss rate constraint.



of m (m^*_{loss}) as a function of the tolerated packet loss rate.

Fig. 5
Variation of the try limit's optimal value as a function of the tolerated packet loss rate when $p = 0.5$.



V. Proposal discussion

It is trivial that the number of retransmissions needed to successfully send a packet depends on current channel conditions and application requirements. Although an increase in the allowable number of retransmissions can increase the probability of a successful transmission in bad channel conditions, such an increase leads to a greater energy consumption. Additionally, the retry limit can be used to control the average number of packets dropped. Under bad channel conditions, this parameter can be useful when applications are able to tolerate a certain percentage of dropped packets.

Limiting the number of retransmissions without affecting the application quality has several advantages. First of all, this allows the sending node to start processing the next packet in the buffer. Second, the receiving multimedia application (for example a video decoder) will receive the most up-to-date information in reasonable delay given that the MAC protocol of the source node (for example a video source) limits the number of retransmissions of multimedia packets (containing for example animated images). Third, this allows a kind of fairness between competing applications in the sense that the applications that need full reliability (such as applications using the TCP protocol) will have less concurrent packets in the network and this will increase considerably their goodput.

VI. Conclusion and future works

In contrast to textual applications, multimedia applications tolerate a loss rate which may be even close to 20% for some of them. Hence, retransmitting lost packets at the MAC level is sometimes useless.

In this work, we studied how we can tune the 802.11 MAC protocol according to multimedia applications' requirements in terms of end-to-end delay and packet loss rate. We interested in the "Try Limit" parameter used in the protocol to control the number of retransmissions of a packet until the sender receives an acknowledgment from the receiver. Based on some analytical preliminaries, we derived two inequalities for computing the "optimal" try limit value. Therefore, we determined the try limit value that should be used by the MAC layer in order to guarantee an end-to-end delay and a loss rate equal to or less than those required by a multimedia application.

In our future work, we target to focus on how the 802.11 MAC layer should interact with upper layers to have an idea about the QoS parameters that the applications would like to have. Real experimentations will also be conducted in order to evaluate the performance of our proposal in a real testbed network.

References

[1] G. Bianchi, "Performance evaluation of the IEEE 802.11 Distributed Coordination Function", IEEE Journal of Selected Areas of Communications, Vol. 18, No. 3, pp. 535-547, March 2000.

[2] G. Bianchi and I. Tinnirello, "Kalman Filter Estimation of the number of Competing Terminals in an IEEE 802.11 network", In Proc. of IEEE INFOCOM 2003, April 2003.

[3] S. Shakkottai, T. S. Rappaport and P. C. Karlsson, "Cross-layer Design for Wireless Networks", IEEE Communications Magazine,Vol. 41, No.10, pp. 74-80, October 2003.

[4] Q. Zhang, W. Zhu and Y. Zhang, "A Cross-layer QoS-Supporting Framework for Multimedia Delivery over Wireless Internet", 12th International Packet Video Workshop, April 2002.

[5] E. Ziouva and T. Antonakopoulos, "CSMA/CA Performance under High Traffic Conditions: Throughput and Delay Analysis", Computer Communications, Vol. 25, No. 3, pp. 313-321, February 2002