On The Effect of Feedback Traffic in IEEE 802.11b WLANs

Daniele Miorandi, Eitan Altman

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

On The Effect of Feedback Traffic in IEEE 802.11b WLANs

Daniele Miorandi — Eitan Altman

N° 4908

August 2003

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--4908--FR+ENG

INRIA
SOPHIA ANTIPOLIS

On The Effect of Feedback Traffic in IEEE 802.11b WLANs

Daniele Miorandi , Eitan Altman

Thème 1 — Réseaux et systèmes
Projets Mistral

Rapport de recherche n° 4908 — August 2003 — 21 pages

Abstract: In this work we study the effect of TCP feedback traffic in IEEE 802.11b–based WLANs. In particular, we present an analytical model which allows to evaluate the impact of the delayed acknowledgments option on TCP throughput. Furthermore, we investigate the impact of such mechanism on the session delay for short TCP transfers. We show that, by carefully tuning some system parameters, both bulk TCP throughput and session delay may benefit from the use of delayed ACKs.

Key-words: IEEE 802.11, TCP, wireless networks, delayed ACKs

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Sur l'effet du trafic d'acquittements sur les réseaux locaux sans fil IEEE 802.11b

Résumé : Dans cet article nous étudions l'effet du trafic des acquittements en TCP dans les réseaux locaux sans fils basés sur IEEE 802.11b. En particulier, nous présentons un modèle analytique qui nous permet de calculer l'impact de l'option des acquittements retardés ("delayed Acknowledgements") sur le débit de TCP. En plus, nous étudions l'impact d'un tel mécanisme sur les délais de transferts courts de TCP. Nous montrons qu'en choisissant méticuleusement les paramètres du système, on peut gagner en débit de TCP ainsi qu'en délai de transfert en utilisant les acquittements retardés.

Mots-clés : IEEE 802.11, TCP, réseaux sans fil, acquittements retardés

1 Introduction

IEEE 802.11 [1] is the de facto standard for wireless local area networks (WLANs). WLANs aim at providing wireless connectivity to the Internet, representing a valid alternative to classical Ethernet LANs. Since they require no fixed infrastructure, the deployment of such networks is fast and attractive from the economical point of view. Even if performances cannot be directly compared to those obtainable by means of a wired network, it seems that future developments of the standard (namely, IEEE 802.11a), capable of transfering data at rates up to 54 Mb/s, will make it an even more attractive solution for ensuring nomadic connectivity. Since wireless access to the Internet is the most common application of this kind of networks, it seems reasonable to assume that the traffic will be carried over the well–known TCP/IP protocol suite.

The IEEE 802.11 standard has been deeply studied in the last few years, and a multiplicity of papers appeared in the literature, covering almost all aspects of the standard. However, up to the knowledge of the authors, no satisfactory analytical model of the interaction between the 802.11 medium access control (MAC) layer, and the feedback–based TCP congestion control mechanism has been presented so far. A simple model for TCP throughput over IEEE 802.11 has been presented in [2], where, however, a different scenario (uplink TCP connections to a server connected through a wired Ethernet to the access point) is studied, while we consider downlink traffic only, where files are downloaded from a cache placed directly on the access point [1]. Furthermore, as it will be discussed later in detail, the analysis presented in [2], does not lead to satisfactory results when the number of contending hosts gets large.

In this paper we will address this issue, with particular emphasis on the impact of TCP feedback traffic (namely, the flow of acknowledgment packets) on network performance. Limiting the scope to single–hop networks allows us to neglect some complex phenomena occurring in IEEE 802.11–based ad–hoc networks [3],[4], helping in getting a better understand of the TCP–MAC interactions. Network performance will be evaluated first in terms of throughput (evaluated at the transport layer) and then in terms of session delay for Pareto–distributed file transfers. More in details, we will analyze the impact on such performance metrics of the use of the well–known delayed ACKs option for TCP [5], together with some further improvements proposed in [6].

In this study we will present some simple analytical models, which, albeit representing an idealization of the real standard specifications, are able to capture the fundamental system behavior. All analytical results are verified by means of extensive numerical simulations, carried out using the ns2 [7] package.

The paper is organized as follows: in Sec.2 and Sec.3 a brief description of the protocol behavior and of the delayed ACKs mechanism, respectively, are reported. In Sec. 4 the throughput analysis is presented, whereas Sec. 5 deals with short TCP transfers. Sec. 6 concludes the paper with a summary of the results and some open issues.

[1] This assumption enables us to focus on the impact of the ACKs flow on TCP performance, neglecting the coupling between the traffic flow and some scenario–dependent parameters, such as RTT and bottleneck bandwidth.

2 Protocol Description

In IEEE 802.11 standard, the medium access control is based on a distributed CSMA/CA mechanism [1]. A node listens to the channel for a time equal to the ditributed inter–frame spacing (DIFS). If the medium is sensed idle, then a random backoff is generated (this is the collision avoidance part of the protocol). During the backoff the node keeps on sensing the channel. If, at the end of the backoff, the medium is still found to be idle, the node gets the token and may start trasnmitting. Since no channel load sensing mechanism is provided, an explicit acknowledgment is necessary to inform the node of the success/failure of its transmission. To accomplish that, when a node receives a packet, after a short interframe spacing (SIFS), it sends a short ACK packet to inform the source of the outcome of the previous trasnmission. Since SIFS is much shorter than DIFS, higher priority is given to the feedback flow [8]. The length of the backoff interval, expressed in slots, is randomly chosen in the set $\{0, 1, \ldots, CW - 1\}$, where CW denotes the actual window size. At the beginning, CW is set to a predefined value CW_{min}. If a collision or loss occurs, the network is assumed to be congested; thus, the congestion window of the nodes involved in the collision is doubled and another trasmission attempt is made. The congestion window cannot grow indefinitely, but may reach a maximum value of $2^{m'} CW_{min} = CW_{max}$; moreover, if a packet incurs m collisions, where $m \geq m'$, it is dropped. If a transmission is detected while the backoff counter has not reached zero, its value is frozen and reloaded as soon as the channel is sensed idle for a DIFS.
The standard encompasses also an extended interframe spacing, EIFS, which shall be used whenever the PHY indicates to the MAC that a frame reception was not succesfully completed. EIFS is provided to let the transmitting station retransmit the packet without interference from the receiving station. However, under the assumption of ideal channel conditions, EIFS does not have a great impact on network performance. This is due to the fact that EIFS acts at the receiver side. As an example, consider a pair of nodes exchanging data. In such a situation, collisions may clearly occur. However, since a transmitting node cannot receive data, nodes will become aware of the failure of a transmission attempt only by means of the ACK timeout expiration. Thus, EIFS will never be used. Hence, even if EIFS may actually be used when there are more than two nodes contending for the channel, we will, in the following, neglect its impact on network performance. Simulation results will support the validity of this assumption.
An optional RTS/CTS mechanism is encompassed by the standard to avoid the well–known hidden terminal problem of CSMA–based MAC protocols. In this case, after a DIFS and a random backoff, a RTS packet is sent to the intended destination. After a SIFS, the destination replies with a CTS, signalling to all the stations in range the foreseen duration of the packet exchange. After a SIFS, the sender may thus start its transmission. Note that, in this way, a virtual channel sensing mechanism is provided, since all the stations which received the CTS may update their network allocation vector (NAV) and go in stand–by for the whole duration of the packet exchange (the channel is "virtually" sensed busy). The DCF operations with RTS/CTS are depicted in Fig.1.
What we described above, is the so–called Distributed Coordination Function (DCF) oper-

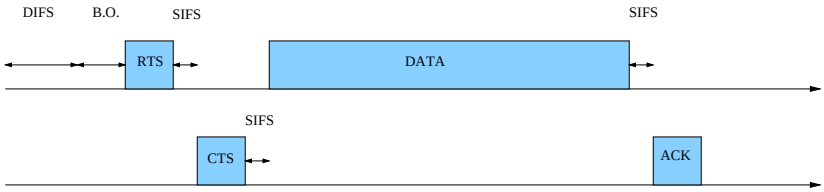

Figure 1: 802.11 DCF operations with RTS/CTS

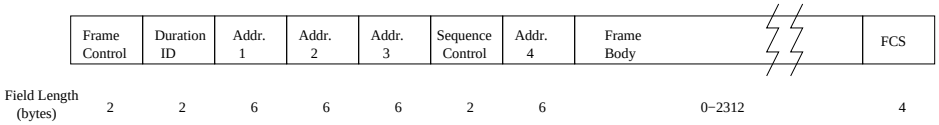

Figure 2: 802.11 MAC frame format

ation mode; the standard encompasses also an optional Point Coordination Function (PCF) mode, which is well suited to centralized operation and handling of delay–sensitive applications. However, most of the WLAN cards actually availlable on the market do not implement PCF for complexity reasons.
At the physical layer, the standard encompasses three different options: infrared, frequency hopping and direct sequence CDMA. We focus on the last option (the one currently implemented), and, furthermore, use the parameters described for operations in the 2.4 GHz ISM band, known as 802.11b [9].

3 Delaying ACKs techniques

When running over bandlimited channels, such as wireless ones, one of the basic techniques to improve TCP throughput is to thin the feedback traffic, by reducing the flow of acknowledgment packets. In such a way, more bandwidth is ensured to the forward TCP link, thus achieving a higher throughput. This idea, of course, presents drawbacks as well: the main one is that, by thinning the ACK flow, at the beginning the congestion window grows slower, and this results, in general, in an increased session delay for short TCP sessions, which constitute most of the traffic carried over the Internet [10]. However, this drawback can be, if not eliminated, seriously mitigated, by using adaptive techniques such as the ones described in [6].
The standard delayed ACK technique for TCP [5] suggests that an ACK packet should be sent every $d = 2$ received packets. If no packet is received for more than a timeout period τ_{out}, an acknowledgment packet has to be generated. It is clear that by further reducing the number of ACKs, we may gain bandwidth, at the expense of a much slower growth of the congestion window at the beginning of the connection or after a packet loss. Note that, in

Parameter	Value
T_{slot}	$20\mu s$
T_{SIFS}	$10\mu s$
T_{DIFS}	$50\mu s$
T_P	$144\mu s$
T_{PHY}	$48\mu s$
CW_{min}	32
CW_{max}	1024
m	7

Table 1: Parameters of 802.11b (long PLCP preamble)

802.11 networks, the feedback traffic has a very strong impact, due to the extremely high overhead encompassed by the standard.

4 Throughput Analysis

In this section, we will provide closed–form approximate formulas for the throughput of a persistent TCP connection. In Tab.1 we reported the parameters we used for evaluation purposes. T_P and T_{PHY} represent the time necessary to transmit the long PCLP preamble and the physical–layer header, respectively. Note that we assume that all nodes use long PCLP header; however, by substituting the appropriate value, also the case of short PCLP preamble may be treated in much the same way. As far as the transmission ranges are concerned, we assume that all nodes operate in good channel conditions, and hence a data rate of $R_{data} = 11$ Mb/s may be used. As a consequence, the control rate is given by $R_{control} = 2$ Mb/s. Besides, we assume TCP packets are $L_{TCP} = 8000$ bits long. The length of TCP/IP header is $L_{IPH} = 320$ bits, whereas the length of RTS and CTS packets are given by $L_{RTS} = 180$ bits and $L_{CTS} = 112$ bits, respectively. A MAC–layer ACK consists of $L_{ACK} = 112$ bits. As can be drawn from Fig.2, the overhead added by the MAC layer (MAC header plus FCS field) amounts to $L_{MAC} = 272$ bits. The raw trasmission time of a TCP forward packet, disregarding the backoff, is given by:

$$T_{TCP\_data} = T_{DIFS} + T_P + T_{PHY} + \frac{L_{RTS}}{R_{control}} + T_{SIFS} + T_P + T_{PHY} + \frac{L_{CTS}}{R_{control}} +$$
$$+ T_{SIFS} + T_P + T_{PHY} + \frac{L_{MAC} + L_{IPH} + L_{TCP}}{R_{data}} + T_{SIFS} + T_P + T_{PHY} + \frac{L_{ACK}}{R_{control}}. \quad (1)$$

As far as the TCP ACK is concerned, we get:

$$T_{TCP\_ack} = T_{DIFS} + T_P + T_{PHY} + \frac{L_{RTS}}{R_{control}} + T_{SIFS} + T_P + T_{PHY} + \frac{L_{CTS}}{R_{control}} + \\ + T_{SIFS} + T_P + T_{PHY} + \frac{L_{MAC} + L_{IPH}}{R_{data}} + T_{SIFS} + T_P + T_{PHY} + \frac{L_{ACK}}{R_{control}}. \quad (2)$$

With the values discussed above, we obtain:

$$T_{TCP\_data} = 0.0018167\ s \quad (3)$$

$$T_{TCP\_ack} = 0.0010895\ s \quad (4)$$

As it becomes apparent, the impact of TCP ACKs is very high, due to the inherently high overhead of 802.11 [11]. Things even worsen when we consider the presence of an ad–hoc routing protocol, which, even if not necessary for WLANs, may in turn provide seamless connectivity for the nomadic user. If the advertised window was set equal to 1 (in this sense we force the TCP source to always wait for the corresponding ACK) we would have that the average backoff time of a given node is:

$$t_b = \frac{T_{slot} \cdot (CW_{min} - 1)}{2}. \quad (5)$$

The above equation holds since, in this case, no collisions occur. However, if the advertised window is much bigger than 1, as it is common in real implementations, we have that both TCP source and sink contend for the channel. Let us start by considering the case where no collisions occur: in that case the freezing mechanism of DCF has to be taken into account. Denoting with C_S and C_D the randomly picked backoff values of source and destination, respectively, we have that the first node accessing the channel will do it after $min\{C_S, C_D\}$, whereas the other will freeze its transmission, wait for the channel to idle again, and then resort with a backoff counter set to $max\{C_S, C_D\} - min\{C_S, C_D\}$. If both hosts have congestion window equal to CW, we easily get:

$$P[max\{C_S, C_D\} = a] = \frac{(a+1)^2}{CW^2} - \frac{a^2}{CW^2}. \quad (6)$$

Then, denoting with $E[\cdot]$ the statistical expectation operator and using Gauss formula, we get, for the average time spent in backoff in case of contention for the channel:

$$\hat{t}_b(CW) = T_{slot} \cdot E[max\{C_S, C_D\}] = T_{slot} \cdot \frac{(CW-1) \cdot (4CW+1)}{6 \cdot CW}. \quad (7)$$

Next step is to consider the probability of collision: since for presistent standard TCP connections nodes may assumed to be always backlogged, we have, by considering that both the involved nodes are in the same congestion stage:

$$P_C(CW) = \frac{1}{CW}. \quad (8)$$

When using the RTS/CTS mechanism, collisions may occur just on RTS packets; by considering that retransmission are triggered after a SIFS (i.e. when no CTS is detected by the sender), we get that the time "lost" in a collision is:

$$T_{coll} = T_{DIFS} + t_b + T_P + T_{PHY} + \frac{L_{RTS}}{R_{control}} + T_{SIFS}. \tag{9}$$

If congestion is a rare event, we may well take $CW = CW_{min}$ in the equations above. Furthermore, we assume that collisions, when occur, are resolved in the next stage (i.e. the congestion window may take two values, CW_{min} and $2 \cdot CW_{min}$. Under such assumptions, the average time spent in backoff for sending a TCP packet and the corresponding ACK on the reverse link is given by:

$$\tau_b = (1 - P_C(CW_{min}))\hat{t}_b(CW) + P_C(CW_{min}) \cdot \left[T_{coll} + \hat{t}_b(2CW_{min})\right] \tag{10}$$

TCP throughput may be calculated as:

$$S_{TCP} = \frac{L_{TCP}}{T_{TCP\_data} + T_{TCP\_ack} + \tau_b}. \tag{11}$$

4.1 Delaying ACKs

By delaying the acknowledgment packets, and using $d \geq 2$, we may get a great performance improvement in terms of TCP throughput. In Fig.3 we reported some simulation outcomes (obtained by using the ns2 simulator [7]), plotted together with the maximum achievable TCP throughput:

$$\hat{S}_{TCP} = \frac{L_{TCP}}{T_{TCP\_data} + t_b}. \tag{12}$$

As it may be seen, decreasing the frequency with which ACKs are sent has a beneficial impact on network performance, although even with $n = 10$ we are not approaching the theoretical limit.
Assume an ACK is sent by the destination node every d received packets. Thus, in steady state, for a fraction of $\frac{d-1}{d}$ of times, no contention for the channel occur, whereas that happens with probability $\frac{1}{d}$. Of course, when no ACKs are sent, no collisions may occur. Hence:

$$S_{TCP} = \frac{L_{TCP}}{T_{TCP\_data} + \frac{d-1}{d} t_b + \frac{1}{d}\left[T_{TCP\_ack} + \tau_b\right]}. \tag{13}$$

As shown in Fig.4, the analytical data match well the simulation ones (the error is below 1.5%).

4.2 Effect of the maximum congestion window

In [4],[12] it is noted that, in multihop networks, there exist an optimum TCP window size which leads to throughput maximization. This optimum window size coincides with the

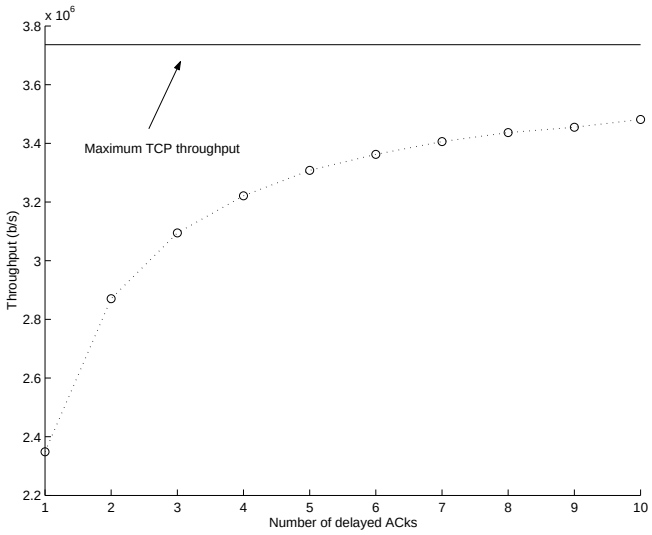

Figure 3: Impact of the number of delayed ACKs

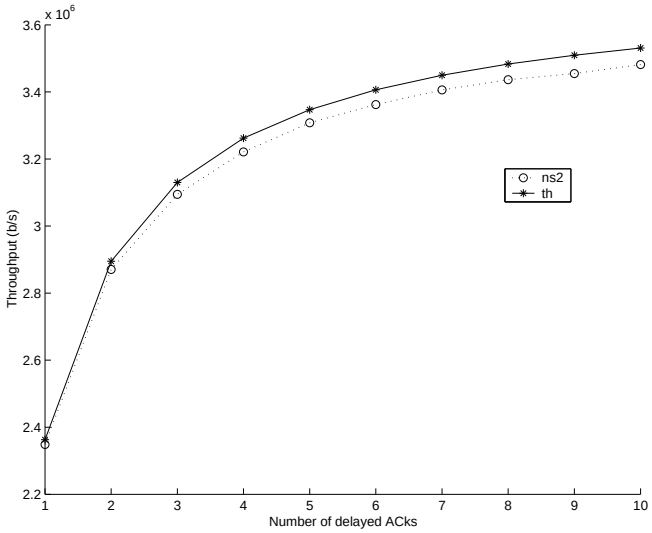

Figure 4: Impact of the number of delayed ACKs: simulation and model

spatial capacity of the network, i.e. with the maximum number of concurrent transmission which may simultaneously take place by exploiting the spatial dimension of the network. For our single–hop network, this would reduce to an optimum window size of 1. However, as we will see, this does not hold. The reason is twofold. When the advertised window W is set to 1, no contention for the channel occurs, and we have a stop–and–wait like transport protocol. With $W \gg 1$, contentions for the channel do occur due to the interference with the concurrent ACKs flow. However, the freezing mechanism provided by the IEEE 802.11 DCF MAC tends to let the nodes behavior become more aggressive when more hosts are trying to access the channel. To take into account the delayed ACK option, we will consider the cases $W = d$ and $W \gg d$, respectively. For the latter, eq. (13) holds, whereas for the first case we have:

$$S_{TCP}(W = d) = \frac{L_{TCP}}{T_{TCP\_data} + t_b + \frac{1}{d}\left\{T_{TCP\_ack} + t_b\right\}} \tag{14}$$

The results are plotted in Fig.5, together with some simulation outcomes.
Furthermore, another phenomenon takes place, which is not taken into account in the previous equations. The queues which try to access the channel in the case $W \gg 1$ are, in fact, coupled. Let us denote by $X_{tcp}(t)$ the number of TCP packets delivered within time t; analogously, $X_{ack}(t)$ denotes the number of ACK packets delivered within the same time. It is clear that, at any t,

$$X_{tcp}(t) \geq X_{ack}(t).$$

If we force $W = 1$, the equality holds at any time instant; on the contrary, for $W \gg 1$, equality holds over an infinite horizon (i.e. for $t \longrightarrow \infty$), whereas in general this in not true for a finite t. Basically, by enlarging the TCP congestion window limit we allow for a "statistical thinning" of the ACK flow, thus providing a performance improvement.
However, in case of many hosts contending for the channel, the probability of collisions would tend to increase with an increase in the maximum allowable congestion window. Hence, it seems reasonable that, for a sufficiently large number of connections, there will be a finite optimal value of the congestion window which leads to throughput maximization. Another observation which may be drawn from Fig.5 is that the delayed ACKs option makes TCP more insensitive, i.e. robust, w.r.t. the advertised window size.

4.3 Multiple connections

Now we aim at extending the model to treat the case when multiple connections are present inside the network. In particular, we will limit ourselves to the case of one–to–many connections, which is the most likely scenario for WLANs, in which most of the traffic will be carried from the access point to the nomadic hosts. To carry on our analysis, we will make some simplifying assumtpions, whose soundness will be supported by means of numerical simulations. Let us denote with n the number of nodes present in the network, and assume there are n concurrent TCP connections, each going from the access point to one of the mobile hosts. First, we need to find the probability of collision given that n stations are

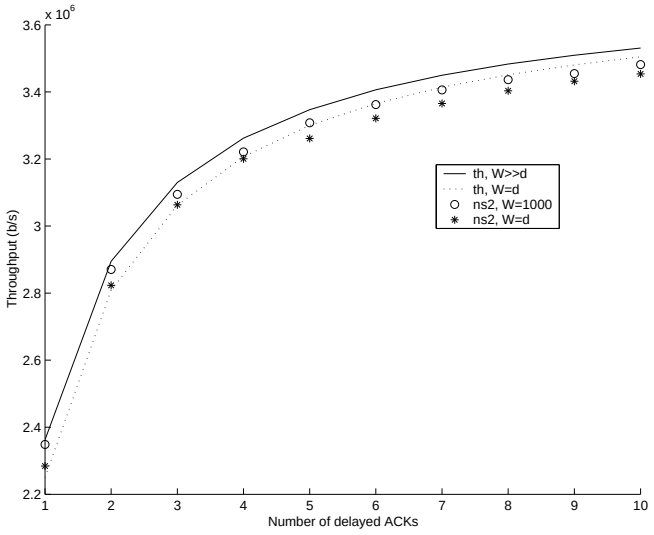

Figure 5: Impact of the maximum congestion window size

backlogged. To do so, we thread the footprints of [13], enhancing the model by considering that, at backoff stage k, the average backoff time is $\frac{2^k CW_{min}-1}{2}$ if $k \leq m'$ and $\frac{2^m CW_{min}-1}{2}$ for $m' \leq k \leq m$. Furthermore, we neglect the possibility that a packet may incur more than $m-1$ collisions (equivalently, we neglect the packet drops due to reached maximum number of retransmissions). Hence, denoting with P_C the probability of collisions, and considering it to be independent of the backoff stage the node is in (this has been proven in [14] to provide good approximation), the average time spent in backoff by a node is given by:

$$t_b = (1-P_C)\frac{CW_{min}-1}{2} + P_C(1-P_C)\frac{2CW_{min}-1}{2} + \ldots + P_C^{m'}(1-P_C)\frac{2^{m'}CW_{min}-1}{2} +$$
$$+ P_C^{m'+1}(1-P_C)\frac{2^{m'}CW_{min}-1}{2} + \ldots + P_C^{m-1}(1-P_C)\frac{2^{m'}CW_{min}-1}{2} =$$
$$= \frac{(1-P_C)}{2}\sum_{k=0}^{m'-1} P_C^k(2^k CW_{min}-1) + \frac{2^{m'}CW_{min}-1}{2}(1-P_C)\sum_{k=m'}^{m-1} P_C^k =$$
$$= \frac{(1-P_C)}{2}\left[CW_{min}\frac{1-(2P_C)^{m'}}{1-2P_C} - \frac{1-P_C^{m'}}{1-P_C}\right] + \frac{2^{m'}CW_{min}-1}{2}(1-P_C)\frac{P_C^{m'}-P_C^m}{1-P_C} =$$
$$= \frac{(1-P_C)}{2}CW_{min}\frac{1-(2P_C)^{m'}}{1-2P_C} - \frac{1-P_C^{m'}}{2} + \frac{2^{m'}CW_{min}-1}{2}(P_C^{m'}-P_C^m) \quad (15)$$

Thus, taking all n nodes to be backlogged (this happens if, for any TCP connection, the congestion window is bigger than one, which is reasonable), we may, as in [13], find the collision probability by numerically solving (15) together with:

$$P_C = 1 - \left(1 - \frac{1}{t_b}\right)^{n-1}. \tag{16}$$

The probability of dropping a packet may then be evaluated as

$$P_{drop} = P_C^m.$$

For our assumption to hold, it must be $P_{drop} \ll 1$. Numerically, it turns out that with $n = 15$ (which is a reasonably high number of active TCP sessions for a typical WLAN configuration), we have $P_{drop} = 7.9689 \cdot 10^{-4}$, which proves the soundness of our assumption.
As mentioned in the introduction, a model for computing the TCP throughput in presence of concurrent TCP sessions has been presented in [2]. Apart from the different network configuration, we feel that their model does not track well the network behavior when the number of nodes, n, gets large. Indeed, they neglect multiple successive collisions, a simplifying assumption that, although providing good results with a low number of active connections, clearly underestimates the time spent in backoff by any single node as soon as n gets large, leading to an overestimation of the TCP throughput.
Consider that the access to the channel is fair, and hence, in steady state, any backlogged node has the same probability of transmitting. However, we have to take into account that an ACK generation is triggered by the reception of a TCP packet. To do so, we consider an equivalent token–passing systems. In such a system, the token is let circulating among the $n-1$ contending TCP connections (in other words, we are splitting the access point into $n-1$ virtual nodes). In this way, at each time only one sender–destination is contending for the channel. We considered, furthermore, that they have an equivalent congestion window equal to:

$$\tilde{CW} = 2 \cdot t_b + 1. \tag{17}$$

Hence, on average, the equivalent time spent in backoff for a complete TCP packet exchange equals $\hat{t}_b(\tilde{CW})$. On the whole, the throughput of a single connection may be found as:

$$S_{TCP}(n) = \frac{L_{TCP}}{(n-1) \cdot \left[T_{TCP\_data} + T_{TCP\_ack} + \hat{t}_b(\tilde{CW}) + \frac{T_{coll}}{\tilde{CW}}\right]} \tag{18}$$

Simulation and analytical results are plotted versus the number of contending TCP connections in Fig.6. It is apparent that the good match validates in turn our analytical procedure.

In the case of multiple connections, delaying ACKs has a duplex beneficial effect: first, it reduces the contention for the channel and, second, it reduces the time taken by each sender–destination pair to exchange packets. Thus, by considering that any TCP sink sends an ACK every d received packets, we proceed as above, by considering an equivalent number

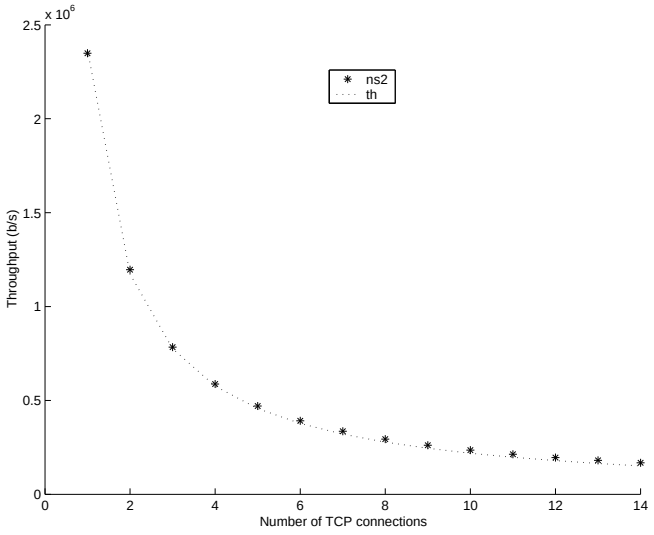

Figure 6: TCP throughput with multiple TCP connections

of TCP connections given by $\frac{n-1}{d}$. Hence eq. (16) becomes:

$$P_C = 1 - \left(1 - \frac{1}{t_b}\right)^{\frac{n-1}{d}} . \tag{19}$$

Then, by solving (19) together with equation (15), we may get the average time spent in backoff and the collision probability. Then, denoting with $C\tilde{W} = 2 \cdot t_b$, we may compute the TCP throughput as:

$$S_{TCP}(n,d) = \frac{L_{TCP}}{(n-1) \cdot \left[T_{TCP\_data} + \frac{1}{d}\left(T_{TCP\_ack} + \hat{t}_b(C\tilde{W}) + \frac{T_{coll}}{CW}\right) + \frac{d-1}{d} t_b\right]}. \tag{20}$$

The level of approximation we may get with the previous formula has been tested through extensive numerical simulations. In Fig.7 we reported the TCP throughput, as a function of d, for $n = 10$. Note that the above formula has been derived neglecting the possibility of timeout to expire: with a high number of nodes, this would lead to an extremely high timeout value, which is not suitable for practical applications due to its negative impact on short TCP sessions, as it will discussed in the subsequent section.

It is then interesting to see if, even with multiple connections, enlarging the maximum TCP window size is beneficial. To do so, we note that, applying our model to the case $W = 1$ (for the sake of simplicity we do not take into account delayed ACKs in this case), we get,

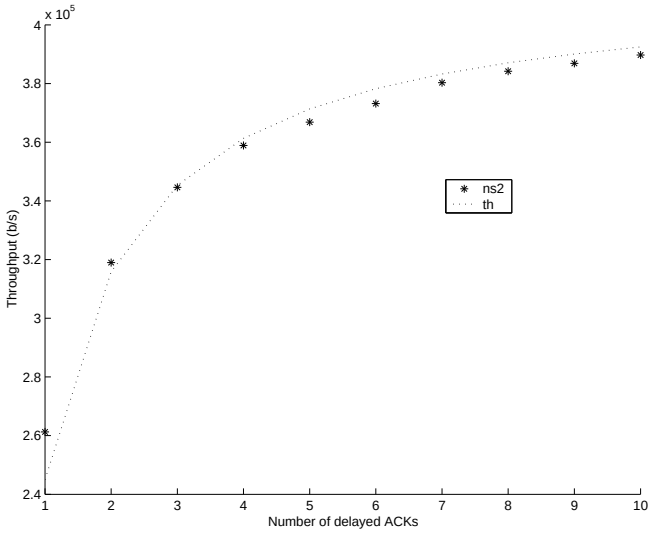

Figure 7: TCP throughput vs. number of delayed ACKs, $n = 10$

neglecting the possibility of collisions to occur:

$$S_{TCP}(n, W = 1) = \frac{L_{TCP}}{(n-1) \cdot \left[T_{TCP\_data} + T_{TCP\_ack} + T_{slot} \cdot (CW_{min} - 1)\right]}. \qquad (21)$$

In Fig.7 we reported a comparison of the curves obtained by using (21) and (18). We note that, as expected, while with $W \gg 1$ better performances may be achieved with a low number of nodes, when the contention for the channel get harsher (i.e. the number of active TCP connections increases), working with $W = 1$ may lead to a little performance enhancement.

At this stage, we would like to stress an interesting feature of the 802.11 MAC–layer specifications. The access to the shared medium is performed via a plesiochronous mechanism, which may, in principle, suffer from unfairness problems due to the different spatial location of the hosts. This, in general, could lead to different RTTs for different TCP connections, providing thus an inhomogeneous throughput to various TCP links. However, by employing a slotted backoff, and by choosing the slot time to be much bigger than the propagation delay, DCF is able to "hide" the differences in RTT due to the propagation delay. This is of course a desirable feature, which allows, from a performance evaluation point of view, to "forget" the spatial dimension of the network (of course under the assumption that we are dealing with a single–hop scenario).

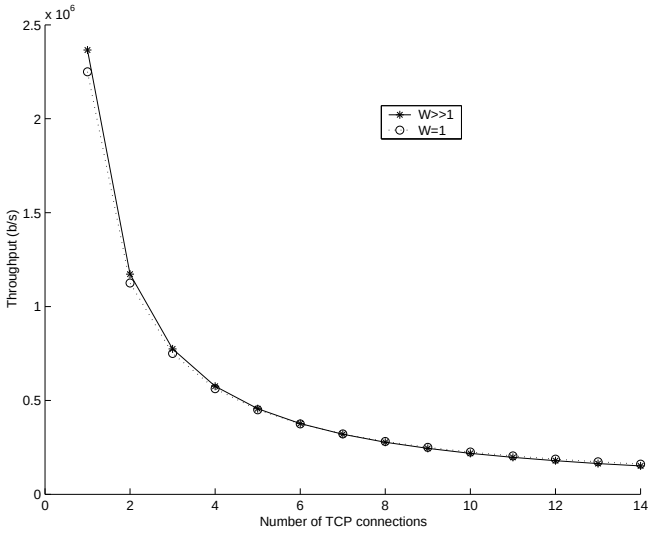

Figure 8: TCP throughput vs. number of nodes: the impact of the advertised window size

5 Short TCP connections

It is widely acknowledged that, although a study of throughput for persistent TCP connections is of interest to characterize network performance, it does not reflect the real nature of Internet traffic. Indeed, most of the TCP sessions are short, and, thus, a characterization in terms of *session delay* is definitely more important to characterize the overall network performance. In this section, we will provide an analytical framework to compute the average session delay for Pareto–distributed files transfer over an 802.11 radio interface. To accomplish so, we will make use of the outcomes of the previous analysis of TCP throughput. In particular, we noted that the maximum congestion window size has a negligible impact on the connection throughput. Hence, our analysis will be based on the assumption $W = 1$. Under such an assumption, 802.11 MAC will provide, in the scenarios we study, collision–free access to the wireless channel. The analytical results will then be compared with the outcomes of extensive simulations done by using ns2 [7].

5.1 Scenario description

We start by considering the simplest scenario, namely a couple of nodes, one of which generates non–overlapping TCP connections at random times (this is compliant with HTTP 1.1). Each session consists of a file transfer, whose size, X, is Pareto–distributed with mean 30 KByte and shape factor $\beta = 1.5$ [15].

5.2 Delay analysis

Let us start by considering standard TCP. If $W = 1$, the average time spent to send a packet and receive the relative acknowledgment, is given by $T_{TCP\_data} + T_{TCP\_ack} + (CW_{min} - 1) \cdot T_{slot}$. Furthermore, we should take into account for the connection setup time, due to the three–way handshaking of TCP protocol. Under the assumption that we are using HTTP1.1 (which uses the same TCP connection for multiple files trasnfer), the impact of the connection setup time should be divided among the files which are delivered in a single session (which represents usually the objects of a web page). For the sake of simplicity, we will assume throughout the analysis that in any session just one single file is downloaded by the client; the framework may however be easily modified to account for different parameter values.
Neglecting the propagation delay, and taking into account that the third packet necessary to setup the connection may be piggybacked within a data frame, the connection setup time may be written as:

$$T_{setup} = 2 \cdot \left[T_{DIFS} + T_P + T_{PHY} + \frac{(CW_{min} - 1)T_{slot}}{2} + \frac{L_{IPH}}{R_{data}} + \right.$$
$$\left. + T_{SIFS} + T_P + T_{PHY} + \frac{L_{ACK}}{R_{control}} \right]. \quad (22)$$

TCP encompasses a 4–way handshake to close down the connection; however, after the delivery of the last TCP segment, the file is taken to be sent to the destination, and hence we may neglect it in the calculations For transmitting a message of size X (in bits), after some easy algebra we obtain an average session delay of:

$$T_{session} = T_{setup} + \left\lceil \frac{X}{L_{TCP}} \right\rceil \left(T_{TCP\_data} + T_{TCP\_ack} + (CW_{min} - 1)T_{slot} \right) -$$
$$- \left(\left\lceil \frac{X}{L_{TCP}} \right\rceil - \left\lfloor \frac{X}{L_{TCP}} \right\rfloor \right) \cdot \frac{L_{TCP} - X \bmod L_{TCP}}{R_{data}}. \quad (23)$$

Basically, by assuming $W = 1$ we end up with a model which is linear in the file size. This means that, in our model, the average session delay is sensitive only to the mean packet size, and not to its distribution. Although this seems odd, simulation results show that our first–order approximation, although slightly overestimating the session delay, is nonetheless able to track the system behavior with a good level of approximation. In order to validate the model, we simulated 10000 file transfers over 802.11b using the ns2 simulator [2]; the results are reported in Tab.2. As expected, the more aggressive behavior obtainable by using a bigger congestion window leads to better performance in terms of session delay.

[2] In reality ns2 implementation of TCP provides bit padding and, hence, all the $\left\lceil \frac{X}{L_{TCP}} \right\rceil$ packets generated present a payload of size equal to L_{TCP}. Furthermore, no explicit connection closing is provided.

	ns2, $W = 1$	ns2, $W \gg 1$	analytical
Mean Session Delay (s)	0.1099	0.1064	0.1095
95% Conf.Int.(s)	$[0.0956, 0.1243]$	$[0.0928, 0.1199]$	–

Table 2: Session delay for short TCP connections, mean and 95% confidence interval

5.3 On the use of delayed ACKs for short TCP connections

The use of delayed ACKs for short TCP connections presents mainly three drawbacks. The first one is that, by thinning the ACKs flow, the congestion window grows slower, which may have a negative impact on very short connections [10]. However, as it will become apparent from the discussion below, this factor does not have a great impact in our scenario, mainly due to the bottleneck represented by the access to the radio channel in our one–hop network. The other problems arise at the beginning and at the end of the file transfer. At the beginning of the file transfer, the TCP receiver will wait for d packets before transmitting an ACK. However, if the initial window size is smaller than d segments, a timeout will occur, which triggers the expedition of an ACK by the TCP receiver. A similar situation occurs at the end of the file transfer, in case $\left\lceil \frac{X}{L_{TCP}} \right\rceil \bmod d \neq 0$: the TCP source will wait for the acknowledgment of the last packets transmitted, while the receiver will not send the ACK packet till the timeout expires. The first problem may be solved by enlarging the initial window size to d packets, which can be made according to [16]. It is worth noting that increasing the initial window size does not lead, in our setting, to a more aggressive nodes behavior, as it is the case in wired network, due to the one–packet–at–a–time capacity of our configuration. Note, however, that such an assumption would require the TCP sender to know the receiver delayed ACKs mechanism settings, which, in turn, would require extra–signalling at higher layers.
In order to partially solve the second problem, a careful tuning of the timeout interval at the receiver is necessary. It is apparent that, in any case, the impact of the timeout value on the session delay depends on the number of packets for session. The longer the session, the smaller the impact of the timeout value. As an example, in Fig.9 we reported the simulation outcomes, in terms of session delay vs. timeout interval, for the standard delayed ACKs option ($d = 2$), averaged over 5000 file transfers. It is apparent that the timeout interval may have a great impact on network performance and, in particular, the choice of a high timeout value may have a negative effect on session delay. Accordingly, we set for our simulations $\tau_{out} = 40$ ms. The results are presented in Tab.3, for d varying between 1 (i.e. standard TCP) and 5. We think there is no real reason to go beyond $d = 5$, since, as it is apparent from simulation results, increasing too much the number of delayed ACKs has not a beneficial impact on network performance. This may be understood by considering the effect of the timeout at the end of the file transfer. Indeed, by increasing d, we increase the probability that the receiver will be forced to send an ACK due to timeout expiration. To investigate the impact on session delay of the slower congestion window growth, we simulated also the dynamic delayed ACK scheme proposed in [6]. Basically, it tunes the number of ACKs to

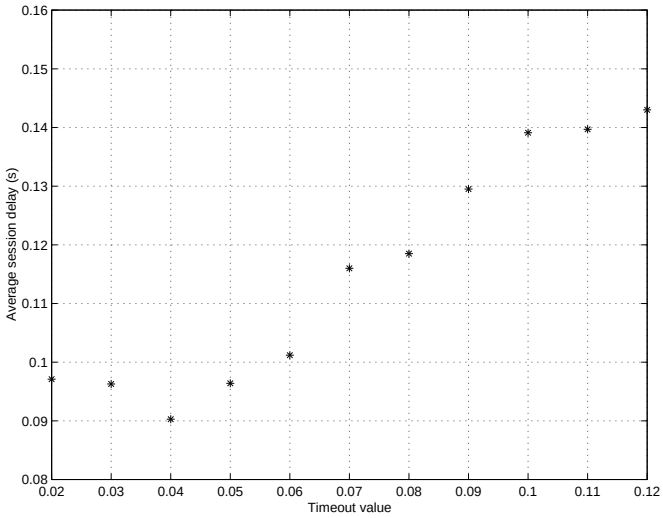

Figure 9: Effect of the timeout value, $d = 2$

$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 5$
0.1064	0.1044	0.1053	0.1043	0.1090
$[0.0928, 0.1199]$	$[0.0934, 0.1154]$	$[0.0951, 0.1156]$	$[0.0945, 0.1141]$	$[0.0993, 0.1188]$

Table 3: Session delay for short TCP connections with delayed ACKs, mean and 95% confidence interval (s), $\tau_{out} = 40$ ms

delay on the sequence number of TCP packets, starting from $d = 1$ (for the first packet, to avoid timeout expiration), $d = 2$ for packets $\{2, 3, 4\}$, $d = 3$ for packets $\{5, \ldots, 8\}$ and $d = 4$ for the rest of TCP session. Surprisingly, we obtain results worse than the "standard" $d = 2$ option, namely an average session delay of 0.1080 s, with a 95% confidence interval of $[0.0980, 0.1180]$ s.
On the whole, we may thus conclude that, by choosing an initial window size equal to d, and by appropriately tuning the timeout interval, the delayed ACK option, with $d = 2, 3, 4$ may overperform standard TCP and also the more complex multi–threshold scheme proposed in [6].

5.4 Multiple transmission

We studied also (only by means of numerical simulations) the impact on network performance of the delayed ACK option in case of multiple nodes trying to access the channel. By taking advantage of the results of previous section, we limited ourselves to the cases $d = 2, 3$ and to the dynamic mechanism proposed in [6]. The initial congestion window size

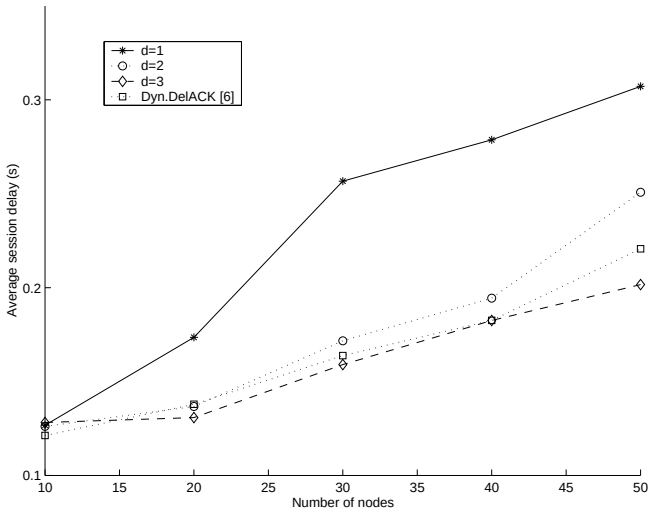

Figure 10: Average session delay vs. number of nodes

was varied accordingly. In our scenario, there are n nodes, any of which requires to the access point the delivery of files whose size is Pareto–distributed according to the parameters described above. After the completion of a file transfer, a node generates a random backoff time, which is exponentially distributed with mean $\lambda = 0.1\ s^{-1}$ [17].
In our simulation scenario, any of the $n-1$ mobile nodes generates 100 TCP connection to the access point. The results are averaged over all sessions. With such a choice of simulation parameters, it is clear that, unless a pretty high number of nodes is present in the network, no superpositions of active sessions will occur, and thus we will get exactly the same results obtained in the previous section. Thus, we simulated our network with a number of nodes varying from $n = 10$ to $n = 50$, which may well represent a hot–spot scenario. The results are reported in Fig.10. As it is clear, delaying ACKs helps in reducing congestion arising in scenario with a high number of concurrent TCP connections, which leads to a substantial performance improvement for 802.11 hot–spots.

6 Conclusions

In this work we investigated the effect of the feedback traffic generated by TCP in a 802.11 WLAN. The results show that the flow of acknowledgment packets strongly reduces the throughput which may be achieved by the forward link. By using delayed ACK techniques, a substantial throughput increase may be achieved, both in single and multi–user scenarios. We then analyzed the impact of such mechanisms on session delay for short TCP sessions.

Simulation results show that, by enlarging the initial window size to a value equal to d, and by carefully tuning the timeout interval at the receiver, even when a single–connection is active, a slight performance improvement may be achieved. In case of many concurrent flows, as it is the case in hot–spot scenarios, a substantial reduction of the average session delay may be observed. However, all these benefits may be achieved only with an appropriate choice of the system parameters; it is thus of interest the study of adaptive algorithms to dynamically tune these settings, in order to adapt to varying network and traffic conditions.

References

[1] *IEEE standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std., Aug 1999.

[2] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance anomaly of 802.11b," in *Proc. INFOCOM*, S. Francisco, US, 2003.

[3] S. Xu and T. Saadawi, "Performance evaluation of TCP algorithms in multi–hop wireless packet networks," *Wireless Communications and Mobile Computing*, vol. 2, pp. 85–100, 2002.

[4] Z. Fu, P. Zerfos, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on TCP throughput and loss," in *Proc. INFOCOM*, San Francisco, CA, 2003.

[5] R. Braden, "Requirement for Internet hosts – communication layers," RFC 2414, Oct 1989.

[6] E. Altman and T. Jimenez, "Novel delayed ACK techniques for improving TCP performance in multihop wireless networks," in *Proc. PWC*, Venice, Italy, 2003.

[7] The network simulator ns2. [Online]. Available: http://www.isi.edu/nsnam/ns

[8] F. A. Tobagi and L. Kleinrock, "The effect of acknowledgment traffic on the capacity of packet–switched radio channels," *IEEE Trans. Comms.*, vol. 26, pp. 815–826, june 1978.

[9] *Supplement to 802.11-1999,Wireless LAN MAC and PHY specifications: Higher Speed Physical Layer (PHY) extension in the 2.4 GHz band*, IEEE Std., Sep 1999.

[10] M. Allman, "On the generation and use of TCP acknowledgment," *ACM Computer Commuication Review*, vol. 28, Oct 1998.

[11] Y. Xiao and J. Rosdahl, "Throughput and delay limits of IEEE 802.11," *IEEE Comm. Letters*, vol. 6, pp. 355–357, Aug 2002.

[12] K. Chen, Y. Xue, and K. Nahrstedt, "On setting TCP's congestion window limit in mobile ad hoc networks," in *Proc. ICC*, Anchorage, USA, 2003.

[13] Y. C. Tay and K. C. Chua, "A capacity analysis for the IEEE 802.11 MAC protocol," *Wireless Networks*, vol. 7, pp. 159–171, 2001.

[14] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE Journal on Sel. Areas in Comm.*, vol. 18, pp. 535–547, 2000.

[15] Y. Joo, V. Ribeiro, A. Feldmann, A. C. Gilbert, and W. Willinger, "TCP/IP traffic dynamics and network performance: a lesson in workload modeling, flow control and trace–driven simulations," *Sigcomm Computer Communications Review*, vol. 31, Apr 2001.

[16] M. Allman, S. Floyd, and C. Partridge, "Increasing TCP's initial window," RFC 2414, Sep 1998.

[17] E. Anderlind and J. Zander, "A traffic model for non–real time data users in a wireless radio network," *IEEE Comm. Lett.*, vol. 1, pp. 37–39, Mar 1997.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399