PiggyData : Reducing CSMA/CA collisions for multimedia and TCP connections

Jean Tourrilhes

jt@hplb.hpl.hp.com

Hewlett Packard Laboratories, Filton Road, Bristol BS34 6QZ, U.K.

This paper presents a scheme to improve the efficiency of radio MAC protocols in the case of bidirectional connections, like TCP connections and multimedia streams. First, the impact of directionality over the contention process is analysed. Then the PiggyData scheme is presented, which decreases the collision rate and overhead of bidirectional traffic over MAC protocols based on CSMA/CA. Finally, the new protocol is simulated in various scenarios to show how it performs.

1 Introduction

The development of radio LANs is very challenging. The radio medium presents some very specific behaviour (like fading and hidden nodes) and has some very strict constraints (scarce resources) that make it fundamentally different from wired mediums. Through the years, a lot of different techniques and MAC protocol have been developed to accommodate those constraints and to mitigate the effect of these behaviours.

Unfortunately, a MAC protocol is worthwhile only if it delivers its performance to the user. This means that when designing a radio MAC, we should not only worry about the radio medium but also about TCP/IP and the likely traffic usage above the MAC. The adaptation of the MAC protocol to the traffic patterns, requirements and behaviour of TCP and multimedia is essential.

2 Bidirectionality & CSMA/CA

Most MAC protocols are agnostic about the directionality of the data. However, most traffics are bidirectional, which has an impact on CSMA/CA, and the protocol should be optimised for such a case.

2.1 A tale of UDP & TCP

The TCP/IP stack [4] provides two very different transport mechanisms, TCP and UDP. UDP is minimal, with only header encapsulation, offering almost a raw access over the MAC layer. On the other hand, TCP offers all the features expected from a transport protocol, with flow control and end to end reliability.

The TCP protocol has been optimised to deliver excellent throughput in a wide range of configurations. But when doing some benchmark [5] of those two protocols on some Wireless LANs, it's not uncommon to see that UDP offers 25 % more user throughput than unidirectional TCP.

The classical explanation is that the radio losses are seen as congestion by TCP, and therefore TCP reduces drastically the sending rate [3]. However, the Wireless LANs tested include stop and go MAC retransmissions, and the UDP test show that 100 % of the packets are received and that they are delivered strictly in sequence, so this explanation is not valid.

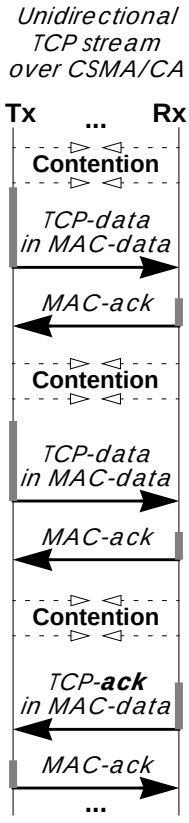

fig. 2.3

The second explanation is the overhead of the TCP ack packets. Those packets don't carry any useful payload, and moreover they are small, so subject to a high overhead [6]. This explains most of the difference, but there is still some additional overhead unaccounted for.

This additional overhead for TCP is in fact due to collisions between the TCP data packets and TCP ack packets over the medium. As opposed to TCP, UDP is unidirectional, so there is no such collisions.

2.2 Why collisions are bad

Packet collisions over the medium is the main challenge facing the designer of MAC protocols. The traffic patterns generated by a network of TCP nodes (asynchronous and bursty) doesn't fit well with a connection oriented approach like TDMA, so most Wireless LANs use CSMA/CA [1], which is based on a per packet contention.

The radio turnaround time is slow, so to get decent performance a low number of contention slots is chosen, leaving a high number of collisions. A two nodes fully loaded 802.11 contention process generates 10 % of collisions [7], and this number increases as the number of nodes goes up.

MAC level acknowledgements and retransmissions overcome those collisions, and using RTS/CTS reduces the penalty of each collision. But still, the net effect of collisions is to waste the available bandwidth, either in packet retransmissions or in unsuccessful RTS/CTS exchanges.

2.3 Layers interaction

We have been talking so far of TCP ack and MAC level ack. This seems a bit confusing. In fact, in most radio LANs each layer (TCP and the MAC) has it's own acknowledgement and retransmission mechanism : a complex sliding window cumulative acknowledgement for TCP ; and a very simple immediate stop and go at the MAC level.

The result is that when TCP transfers a chunk of data, there is up to 4 transmission over the medium, as can be seen in *fig. 2.3*. This seems like an overkill, but is necessary to get things working and optimal performance.

2.4 Bidirectionality in common traffic usages

As we have seen, a unidirectional TCP connection is bidirectional at the MAC level, because of the flow of TCP acks from the receiver to the sender. As most high layer application and protocol use TCP (http, ftp, SMB, NFS...), we can assume that any bulk data transfer is bidirectional, so for each usage of the network there is two nodes contending for the medium instead of only one.

Applications not using TCP are usually multimedia applications, using UDP and having their own flow control and reliability mechanism (tailored to the exact need of the application). In many cases, those multimedia applications are a real time interactions between two humans (like a voice over IP phone call), so bidirectional as well.

In fact, bidirectionality is the general case for network traffic over the link layer, and only exceptions are unidirectional streams (like multicast traffic).

3 PiggyData

PiggyData is a scheme which takes advantage of bidirectionality to reduce collisions and overhead on CSMA/CA, and therefore achieves an increase of performance for TCP and multimedia connections.

3.1 The main idea

TCP itself includes a scheme to reduce the overhead of the TCP ack transmissions when it is carrying a bidirectional traffic, called piggyback acknowledgement [4]. The data packets transmitted in the reverse direction contain the acknowledgements for the data transmitted in the forward direction : when the receiving node has to acknowledge some received packets, it can wait for the next data packet to be sent and put the acknowledgement information in the header, so doesn't have to send a separate TCP ack packet.

The main idea of PiggyData is to apply the same concept at the MAC level. The MAC acknowledgements are going in the same direction as the reverse flow of data (see *fig. 2.3*), so both could be combined.

However, the timing of the MAC stop and go mechanism are tight, and the MAC level ack is integral part of the transmission frame. The MAC level ack needs to be sent precisely a SIFS period after the data packet, so the MAC can't piggyback the MAC level ack in a later data packet.

The MAC level ack and the reverse stream of data can still be combined by doing the reverse operation : to piggyback a data packet in the MAC level ack transmission.

By sending the data with the ack packet instead of separately, we reduce the level of contention and collision, because this data packet is sent "contention free". The overhead is also reduced, because the transmission of this data packet avoid the time taken to resolve the contention.

3.2 Implementation

PiggyData combines a MAC level ack and a data transmission in a contention free frame. We want to retain the MAC level ack as a separate packet, because its short size guarantees a low error probability, but the two packets can't be separated in time by a SIFS (the shortest timing) because other mechanisms (like fragmentation) depend on it.

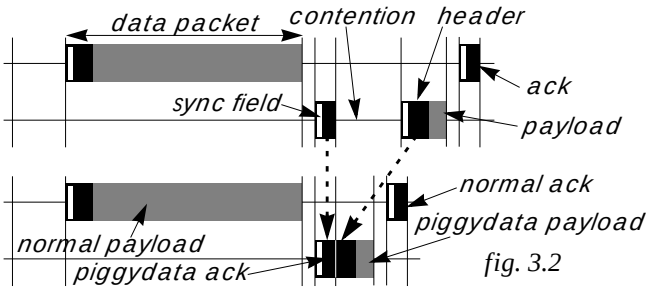

fig. 3.2

PiggyData sends the MAC level ack and the data packet as two separate packets in the same transmission burst (see *fig. 3.2*). The two packets are unmodified and only concatenated, sharing the same synchronisation field. The ack packet needs to signal the data packet following (a flag in the ack header), to allow the receiver to recognise the PiggyData ack.

The PiggyData procedures are quite simple. When a node receives a valid data packet, if the transmit queue is empty, the node sends back a normal ack, otherwise it sends a PiggyData ack including the first data packet from the transmit queue (the one ready for transmission). As each packet of the PiggyData frame retains its own destination address, we can associate any data packet with the ack regardless of its address (and therefore avoid any address comparison).

When any node receives a PiggyData frame, it decodes independently the ack and the data packet, processes each of them if the destination address match its own, and acknowledges the data packet if required.

PiggyData integrates transparently in most CSMA/CA MAC protocols such as 802.11, uses very few resources, is simple to implement, requiring only minor change to the framing, the ack transmission and reception processes.

3.3 Fragmentation and Packet Frame Grouping

The PiggyData scheme is quite simple, but the interactions with fragmentation and Packet Frame Grouping complicate its implementation.

Fragmentation [1] allows to split a large packet into a burst of shorter data-ack exchanges separated by a SIFS, to decrease the impact of channel errors. To reduce the overhead of fragmentation and the resource usage in the receiver, all the fragments of the same packet should be sent into the same contention free frame.

PiggyData allows to group together a data and ack packets with different destination addresses, so may break the fragment train of the initial transmitter. To avoid that, PiggyData must be disabled on the intermediate fragments and enabled only on final fragments and complete packets.

Packet Frame Grouping [6] is similar to fragmentation but applies on the fly to independent packets. As PiggyData is more efficient than Packet Frame Grouping, it should always take precedence, and so no special rule needs to be applied. PiggyData allows more than one node in the contention free frame, so the Frame Size is now defined as the maximum number of bytes that an individual node can transmit between two contentions (and is not enforced globally).

3.4 Fairness

In a normal CSMA/CA MAC protocol, accesses to the medium are made only following a contention, so each node has an equal opportunity to send a packet over the medium.

On the other hand, PiggyData allows all nodes involved in a bidirectional traffic to have additional opportunity to transmit data in the ack frame. So, PiggyData advantages all nodes doing interactive sessions over those doing unidirectional bulk transfer, but otherwise remains fair.

PiggyData also allows the grouping of any data packet with an ack regardless of its destination address, so more than two nodes can transmit in the same contention free frame, and potentially all the nodes of the network in a round robin fashion. In practice, traffic is mainly bidirectional, so this is very unlikely. It is also possible to set a limit on the total contention free size to avoid any excessive length.

4 Simulation model

The models used for these simulations have been carefully chosen to be simple and realistic, to illustrate the PiggyData scheme, its performance and behaviour.

4.1 MAC model

The MAC model includes a fairly complete 802.11 channel access mechanism. This model is based on an 802.11 backoff (slotted exponential contention). All management functionalities have been removed to keep the model simple.

The model implements MAC level acknowledgments and retransmissions (up to 4), RTS/CTS (for packets larger than 250 B) and Packet Frame Grouping [6] (the frame size is 2000 B).

The maximum packet size is 1500 B (non fragmented). All other parameters conform to 802.11 [1] (CWmin = 16 ; SIFS = 28 µs ; Slot = 50 µs ; Headers = 50 B).

Four types of MAC configuration are available :

- *Normal* : the basic standard 802.11 MAC
- *PiggyData* : with the addition of PiggyData only, up to one packet per node after each contention.
- *Packet Frame Grouping* : with the addition of Packet Frame Grouping only (see [6])
- *PiggyData + PFG* : with both scheme combined, each node may transmit in the contention free frame up to the limit set by the Frame Size.

4.2 Channel model

The channel model is a simple radio channel model, including node to node attenuation (80 dB by default), Rayleigh fading (calculated on a per packet basis) and antenna diversity. The bit rate is 2 Mb/s, and there are no hidden nodes and no interferers. The transmitted power is +20 dBm, and the sensitivity is -80 dBm (in a Gaussian channel).

4.3 Traffic models

Various traffic models are used through the simulations. More information on the traffic models and their behaviour may be found in [6].

5 Simulation results

Many simulations have been performed to study how well PiggyData performs. All the simulations have been implemented under the Bones® Designer™ environment.

5.1 Random traffic

The *random* traffic uses a Poisson process, a uniform distribution of packet sizes and random destination address. The network is composed of an increasing number of active nodes. In the throughput simulation the network load is 1.80 Mb/s (saturated), whereas in the latency simulation the network load is 1.40 Mb/s.

When there is only one node active on the network, obviously PiggyData offers no improvement, because the traffic is unidirectional (see *fig. 5.1a*). When the number of active node increases, the length of the contention free frame created by the PiggyData scheme also increases on average (more nodes can participate), which increases the overall network performance. The combination of both Packet Frame Grouping and PiggyData offers the best result.

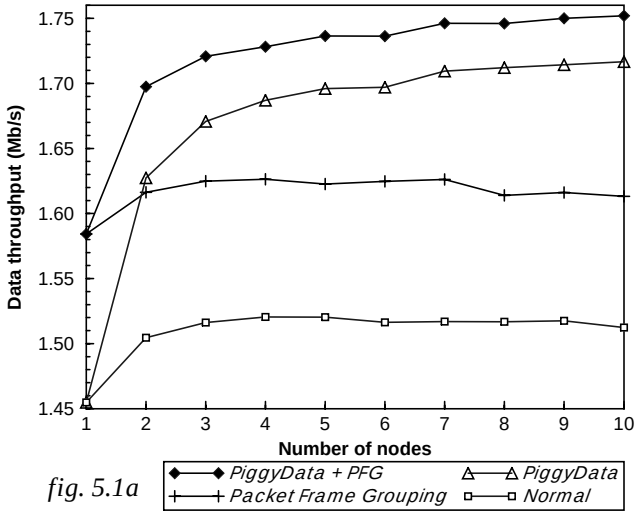

fig. 5.1a

In a similar way, PiggyData decreases the latency experience by packets (see *fig. 5.1b*). The latency doesn't increase with the number of active nodes because the same load is statistically spread on all the nodes (so each node receives a smaller load).

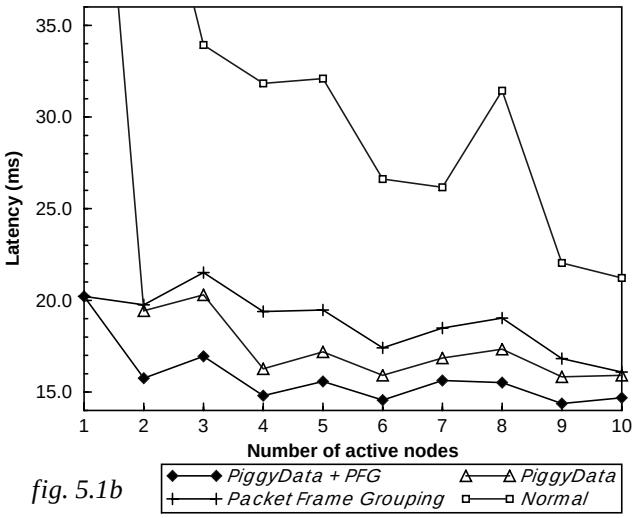

fig. 5.1b

5.2 TCP like traffics

As the *random* traffic doesn't represent accurately real traffic, more simulations have been done with some TCP like traffic patterns.

5.2.1 "TCP2" traffic

The *TCP2* traffic is a simple bimodal traffic, with large (TCP data - 1500 B) and small (TCP ack - 40 B) packets. The traffic is saturated and the destination address random, and there is 5 nodes in the network.

In the simulation we explore different settings of the ratio between the number of TCP data packets and TCP ack packets. This ratio is controlled by the size of the TCP window and the settings of the TCP stack : an optimised TCP stack will produce less TCP acks than an non optimised one.

In this simulation, the number of nodes contending is always the same, five. The Frame Size (2000 B) is set in such a way that a node can group many ack with a data packet but not two data packets together. As we can anticipate, when the frequency of the TCP ack decreases, there is less packet to group, so the improvement offered by Packet Frame Grouping decrease significantly (see *fig. 5.2.1*).

However, PiggyData can always group packets regardless of their size, so its effect is less sensitive to that parameter, except that the improvement is inversely proportional to the size of the packets grouped, so decreases as well.

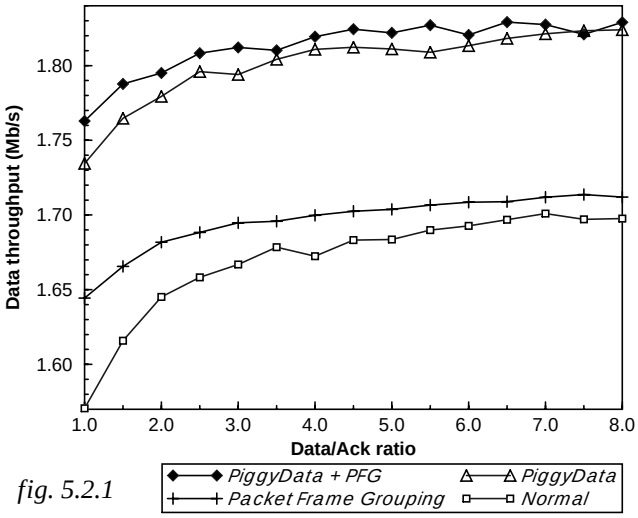

fig. 5.2.1

5.2.2 "TCP1" traffic

The *TCP1* traffic simulates a bulk transfer between two nodes, one sends large packets (TCP data - 1500 B) and the other reply with a small ones (TCP ack - 40 B). The traffic is saturated.

This time, changing the data/ack ratio impact the way the second node contend for the medium : when we decrease the proportion of TCP ack in the traffic, the second node has less and less to send, when the ratio reach infinity we have in fact a unidirectional traffic (like UDP).

Consequently, when we decrease the ack proportion, there is less contention, so less collisions, so the throughput improve dramatically (see *fig. 5.2.2*). This is exactly the effect we were observing in our benchmarks (see section 2.1).

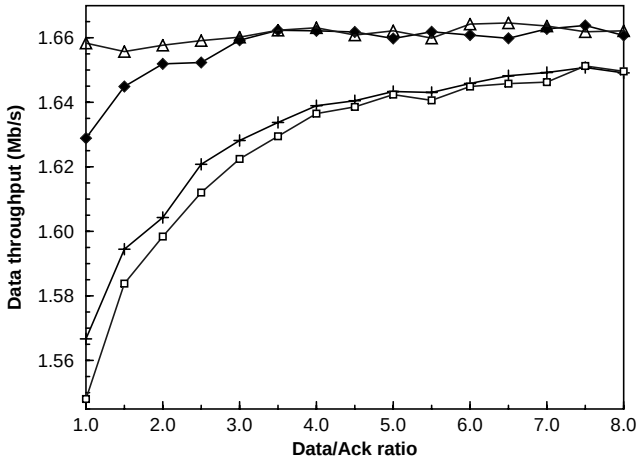

fig. 5.2.2

As observed before (see section 5.2.1), Packet Frame Grouping has little improvement, because it can only group a few TCP acks together, and PiggyData is quite effective when the traffic is highly bidirectional (data/ack ratio small) and less when it becomes unidirectional.

One of the strange thing is that for low ratio, when we increase the proportion of acks, the throughput with PiggyData only increases and is much higher than with the addition of Packet Frame Grouping. When we don't use Packet Frame Grouping, we force all the TCP ack packets to use PiggyData (which presents slightly less overhead), and we also decrease the average number of slots per contention (because there is two nodes contending instead of one).

5.3 Multimedia traffic

This is exactly the same setting as in one of my previous papers [6]. The previous simulations are not very good for exploring latency problems, mixed and multimedia traffics, so in this simulation we combine a TCP like traffic and a voice traffic on the same medium.

5.3.1 TCP1 + Voice traffic

The simulation includes 4 voice nodes and 2 data nodes. The *voice* traffic has a fixed individual load (32 kb/s + UDP overhead), and different average packet arrival times are used in the simulation, which impact the packet size used by the

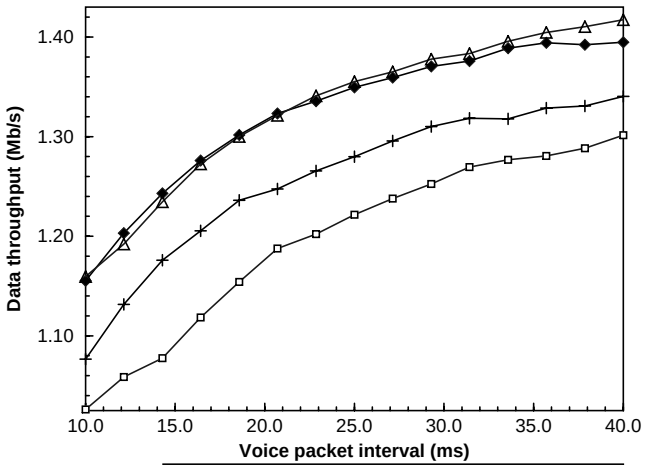

fig. 5.3.1a

voice nodes. The four nodes are grouped in two pairs talking to each other, so PiggyData effect is limited to the two nodes within the pair.

The two other nodes of the simulation are using the *TCP1* traffic (one sender and one receiver, data/ack ratio set to 1). Only the throughput of the two data nodes and the latency of the four voice nodes are measured.

PiggyData offers a very significant improvement of the throughput for the two data nodes (see *fig. 5.3.1a*). As before (see section 5.2.2), adding Packet Frame Grouping to PiggyData doesn't improve the throughput of the data nodes.

As in my previous paper [6], Packet Frame Grouping offer a very significative improvement of the latency experienced on the network by the voice packets (see *fig. 5.3.1b*). PiggyData is not as effective, and the combination of both schemes gives a lower and stable latency across the whole range of packet sending rate.

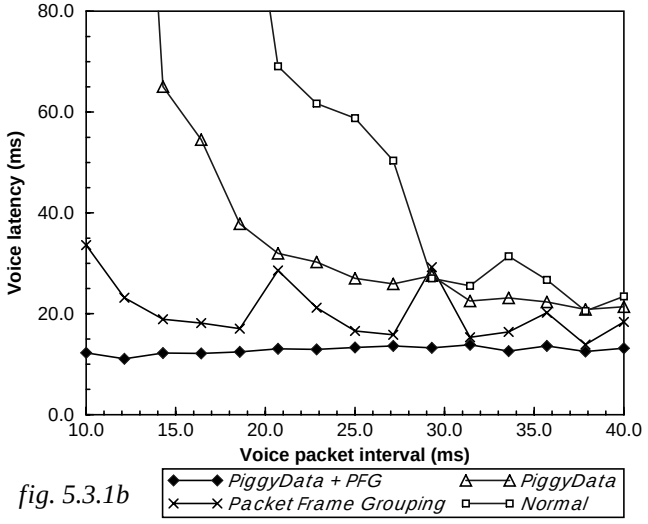

fig. 5.3.1b

5.3.2 Latency target

In the previous simulations, the latency is in fact the average latency of all the voice packets. This is a very good indicator of performance (especially with the huge latency improvements observed), but doesn't show the exact shape of the latency distribution.

Multimedia applications often expect packets to arrive within a bounded delay at the receiver. Packets arriving too late are just discarded. The number of packet discarded and the delay tolerance impact the quality offered by the application. For example, a Voice Over IP must feed voice samples at fixed rate to the decoder, and good quality requires less than 100 ms transfer delay between the two nodes and less than 3 % packet losses [2].

The following simulation (see *fig. 5.3.2*) shows some latency distribution. The same setting as the previous simulation is used (TCP1 + Voice latency) with the average arrival time of voice packets set to 25 ms. We set a latency target and measure the probability for a packet to arrive within this bounded delay. This represent in fact the reverse cumulative distribution of latencies.

The improvement offered by PiggyData and Packet Frame Grouping is quite different, Packet Frame Grouping works

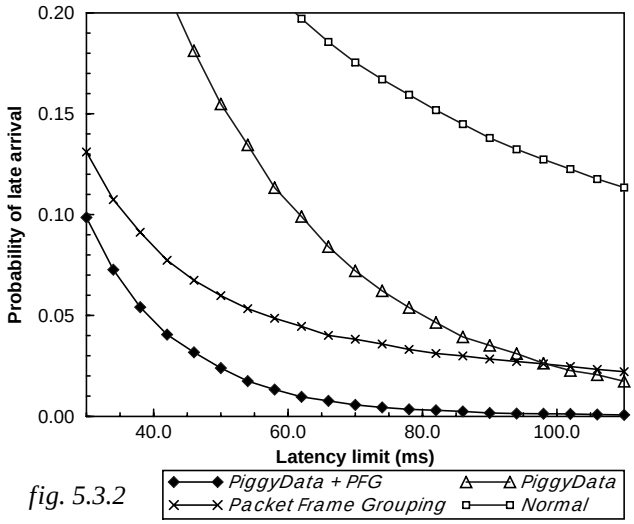

fig. 5.3.2

best for most packets but leaves 3 % with a higher latency than with PiggyData due to the potentially longer contention free frame. Within the same bounded delay, nodes using both PiggyData and Packet Frame Grouping would discard much less packets than a normal 802.11 nodes, giving a much better multimedia quality.

6 Conclusions

Most traffic generated by common applications and networking stacks is bidirectional, which creates a higher collision rate over CSMA/CA.

PiggyData is a very simple modification of CSMA/CA where each node piggyback some data packet in the MAC level acknowledgement following the reception of a packet.

PiggyData decreases the overhead and the collision rate, increasing the network throughput and decreasing the latency in most traffic configurations in very effective way. PiggyData can be combined with Packet Frame Grouping to create a MAC highly optimised for TCP and multimedia traffics.

7 References

[1] *IEEE 802.11 : Wireless LAN medium access control (MAC) and physical layer (PHY) specifications.* IEEE.

[2] V. Varma, M. Thomas, S. Konish, L. Seltzer and D. Goodman. *Performance of 32 kb/s ADPCM in Frame Erasures.* Proc. of IEEE VTC '94.

[3] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan and Randy H. Katz. *A comparison of mechanisms for improving TCP performance over wireless links*. Proc. of ACM SIGCOM '96.

[4] Douglas E. Comer. *Internetworking with TCP/IP*. Prentice Hall.

[5] Rick Jones. *NetPerf : a network performance benchmark*. http://www.netperf.org/.

[6] Jean Tourrilhes. *Packet Frame Grouping : Improving IP multimedia performance over CSMA/CA.* Proc. of ICUPC '98.

[7] Jean Tourrilhes. *Robust Broadcast : improving the reliability of broadcast transmissions on CSMA/CA.* Proc. of PIMRC '98.