Evaluation and Improvement of Multicast Service in 802.11b

Christian Bravo[1] and Agustín González[2]

[1] Universidad Federico Santa María, Department of Electronics. Valparaíso, Chile
chbravo@elo.utfsm.cl

[2] Universidad Federico Santa María, Department of Electronics. Valparaíso, Chile
agustin.gonzalez@elo.utfsm.cl

Abstract. Wireless Technologies have allowed a fast growing of the Internet service in both public and private environments where wireless networks mostly consist of nodes interconnected with a fixed infrastructure; nevertheless, they do not offer a good performance in all the wide variety of services that are required for applications. Although the IEEE 802.11 MAC protocol is the standard for wireless LANs, this protocol shows a very poor performance and reliability compared with the multicast traffic transmitted in wired networks, this represents a significant challenge for existing 802.11 networks because it requires transmission over multiple unreliable channels to heterogeneous receivers with different connection bit rates and very limited feedback information to the sender. In this paper, we discuss the drawbacks of several protocols proposed in the literature that offer reliable multicast service. In addition, this work evaluates the performance of the wireless networks under multicast traffic and presents a proposal for actual IEEE802.11 networks to improve their efficiency. It uses a reliability control based on a polling service along with controlled retransmissions; this allows servicing nodes applications with a high efficiency without deteriorating fairness in the service. We present details of a prototype implementation and results that suggest that our protocol performs better than other proposed in terms of reliability as well as data throughput in our measurement scenarios.[3]

Key words: IEEE 802.11, Multicast, Wireless LAN, Reliability.

1 Introduction

The wireless technology has grown fast as a way to interconnect computers. Wireless networking applications continue to proliferate at an incredible pace as wireless features, functions, security, and throughput improve. IEEE802.11b [**?**] [**?**] is the standard on which wireless networking are based today, and products that employ the technology support a broad range of uses for enterprises

[3] We thank to the USM-23.04.26 Research Project of the Federico Santa María University for the support to develop this work.

and home users.
An important difference between the wireless and the wired networks is the greater datagram loss rate of the first ones. These networks operate in a radio band that was originally reserved internationally for non-commercial use of RF electromagnetic fields and its power of transmission is limited. This condition, added to the interference of the radio frequency spectrum and the medium access protocol defined in 802.11b, causes a great number of datagram losses. Unlike most of the data-link layer protocols used in wired networks [**?**], 802.11 incorporates acknowledges (ACK's) to decrease the high rate of frames lost due to interferences, decay of signal, and collisions. In this way, it is attempted to reach reliability in unicast transmissions over wireless links. However, the ACK's and the retransmissions are omitted when the data destination is a multicast address, thus an implosion of ACK's or requests of retransmissions at the Access Point is avoided. Based on the same reason, the IEEE 802.11 standard does not include the optional extension RTS/CTS for multicast/broadcast transmissions [?] [?]. The multicast datagram loss rate causes that application programs such as radio broadcasting stations, television, distributed computing, chat and whiteboard applications and all type of Internet conferences show a very poor performance compared with them on wired networks [**?**]. This is explained because they are normally designed to operate with low error rates and because they interpret the datagram loss as congestion, reacting with complex mechanisms like changing of coders to decrease the transmission rate. Obviously, this does not solve the problem that has its origin in a high Block Error Rate (BER) at the physical level.
This paper goal is oriented to evaluate the problem that appears in a network with infrastructure, operating in the *Basic Service Set (BSS)* mode, where the stations are fixed in indoor and outdoor scenarios. Thus, we centered our interest on the multicast performance degradation of the wireless link that connects the AP with the wireless stations. In addition to this evaluation, we propose a protocol to improve the reliability of the multicast transmissions until achieve a performance close to the one observed in wireless unicast transmissions. The previously proposed solutions use Error Correction Codes [**?**] or suggest modifications to IEEE802.11 MAC protocol [**?**]- [**?**] which does not solve the problem of the networks that were already installed. We present related work in more detail in the next section.

2 Related Work

Several approaches have been proposed in the multicast communications area to reduce the effective packet loss rate and, at the same time, provide a reliable service. Some of them provide reliable multicast transmissions in the end-to-end sense assuming the existence of underlying routing protocols. Other solutions propose to analyze the subject of the losses recovery using errors correction techniques like FEC (Forward Error Correction) [?]. Several multicast protocols

have been developed as an extension for MAC Layer described in the standard IEEE802.11. They try to improve the performance in the data transmission process between the nodes. Our proposal is in the same direction, looking for a mechanism based on the IEEE802.11 standard that improves the performance of the reliable multicast protocols proposed so far. Following, the proposals of our best knowledge to date are briefly described. The protocol proposed in [**?**] tries to extend the IEEE802.11 standard for multicast/broadcast transmissions using messages RTS/CTS (Request To Send/Clear To Send) in all network configurations. Obviously, this protocol presents the CTS's implosion problem in the source node. The protocol proposed in [?], known as Broadcast Support Multiple Access (BSMA), is an extension to the previous protocol [?]. Briefly, after the source node received the CTS of some neighbor, this one sends the data and waits for a determined time for a Negative Acknowledgment (NACK). A NACK is sent by the node that sent the CTS if the data does not arrive before a timeout occur. If the source node does not receive a NACK then it assumes that the transmission was completed. In the other case, it senses the channel to restart the RTS/CTS process. In this approach, the node that sent the CTS does not consider if the other clients received data successfully or not before sending an ACK or NACK which means that some clients could be losing data. Reference [**?**] describes another protocol known as Broadcast Medium Window (BMW). Where each multicast/broadcast message is treated like multiple unicast ones. In summary, the idea is to handle a list of the receivers and to make all the DCF (*Distributed Coordination Function)* contention, transmission and error recovery process for each one interested in receiving the message. Although it is an interesting option because it improves reliability, we considered that in this solution there is too much redundancy since retransmissions can be done using multicast frames. Thus, the error recovery process in a receiving node aids to the transmission of the same frame to others. Other proposals as [?] and [**?**] are focused on the IBSS *(Independent Basic Service Set)* network configurations also known as Ad-Hoc Networks [?], where it is assumed that all nodes are within the same transmission radio. The presented works have contributed to improve the performance of the IEEE 802.11 standard for multicast/broadcast transmissions; nevertheless, most of the proposals discussed previously are difficult to integrate to the already spread systems based in this standard. In our work a novel scheme is presented which does not require modifications to the MAC layer because it works in the application level and uses an interceptor between IP and MAC layer as it is described in section 4.

3 Measurement Scenarios

The first step in this investigation was the reproduction of the problem observed in the FDI-Corfo "IP Wireless Diffusion" project [**?**], where it was possible to detect serious performance problems on the applications when the physical medium was unable to reach a transmission rate similar to those observed in

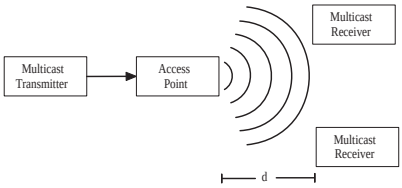

Fig. 1. Different Scenarios. Indoor scenario used d < 50 [m]. Outdoor scenario used d > 1 [Km]

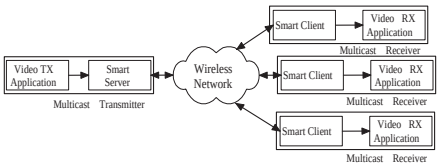

Fig. 2. Proposed solution structure

wired networks using the RealNetworks technology. For that, two test scenarios were structured in which the problem had been observed. The first scenario is an Indoor environment (WLAN Network with a radio less than 50 [m]) and the second one is an Outdoor environment (WLAN Network with a radio over 1 [km]). The second scenario may need special hardware that allows elevating the power of the signal to obtain long distance transmissions (amplifying) or different antennas configurations. Figure 1 shows the previously described scenarios. A multicast server and client application were developed to transmit multicast traffic to multicast client applications. The server application transmits streams varying the transmission rate, the size of the datagrams and the time between each datagram (which emulates transmission of different codification rates videos). In the other side, the client application includes functions to measure the loss percentage of frames sent by the server. These functions were used to look for dependencies between the percentage of losses in receivers, the transmission rate and the size of frames. Besides, physical layer information provided by the driver of the WNIC was collected, for example, power level of the signals received or the Signal to Noise Ratio (SNR) to find the relation between wireless link quality and losses percentage in receivers.

4 Design of the proposed solution and Protocol Description

This section introduces the main idea of our solution and how our protocol works to improve the actual performance of multicast transmissions. The idea is to use efficient retransmissions methods by polling the clients, with the objective to improve the actual performance of multicast transmissions in wireless networks. As programming language, we chose Java because it allows our applications to be run on any arbitrary Java-capable device with an 802.11 interface without any further changes to the device itself. The JAVA platform that we developed consists of three main parts: The register of clients, the addresses mapping service and the retransmissions protocol.

4.1 The Register of Clients

In order for the Smart Server to be able to perform a retransmission protocol based on a polling system, a register of Smart Clients is needed. This register contains a list with information of all the clients connected to the multicast transmission. The information required includes the *IP address, Port* and *Life-Time* for each client. This allows us the regular maintenance of the state information and keep an updated database of the number of receivers in the group. For this, the Smart Server sends *Multicast Beacons* through a multicast control address. These beacons contain the Unicast IP Address and Port of the Smart Server. Smart Clients are also members of this multicast control group and, after receiving a beacon, they start a random backoff timer to reduce the collision probability of their responses. When this timer ends, each client sends a *Unicast Subscription Message* to the unicast address of the Smart Server. After that, each Smart Client will wait for another five beacons before start this timer again. Thus if this message is lost, the Smart Client will retry its subscription later. This *Unicast Subscription Message* format is *<IP address, Port>*. We added the application port because it will be used to receive unicast polls in our protocol. Once the Smart Server receives the Smart Client information, it adds a field of lifetime to each client pair. If the server already has the pair of the client, it will only update the lifetime field; else it adds the new client to the register. If no Subscription Message is received, the client is deleted from the list when its lifetime reaches zero. Thus, the Smart Server keeps an updated list of the active Smart Clients, and it allows to delete those that have left their multicast group.

4.2 The Addresses Mapping Service

This service consist in redirecting the multicast frames by changing the Multicast Address Destination, the Destination Port and the Time-To-Live (TTL) fields of the incoming frames. We installed a multicast traffic interceptor between the multicast transmitter and the wireless network, and other one intercepting the traffic between the wireless network and the multicast receivers, as the figure 2 shows. The technique we use is similar to that of Secure Shell (SSH) and others where it is possible to send the X-Windows traffic through the same SSH connection using a *tunnel* to obtain its encryption. Basically, the original source sends its traffic to a multicast address *A.A.A.A port XXXX with TTL=0*, which means that the stream will not leave the local host machine. Our Smart Server joins that multicast address and redirects the datagram with new parameters such as *B.B.B.B port YYYY with TTL=N* (where N depends of the network size). Therefore, the datagram leaves the local host with a multicast destination address known by the Smart Clients, they join to that address and perform the retransmission protocol. After that, they apply the mapping service again to deliver the stream to the final destination with parameters like this *A.A.A.A port XXXX with TTL=0*. This has to be the address to which

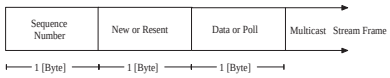

Fig. 3. Retransmission Protocol Header

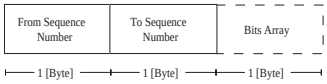

Fig. 4. Poll and Poll Response Message Format. Poll Message does not include the bit array field

the final destinations joined in first place, and by setting the TTL in zero we limit the scope of the traffic to the specific machine which is running the Smart Client application.

4.3 The Retransmissions Protocol

The third and last module contains the retransmission protocol itself, that we named as *GroupPoll protocol (GP)*. The Smart Server map the multicast stream using the mapping service first and then is forwarded to the Smart Clients. Besides, when the multicast traffic arrives to the Smart Server it adds a header to each frame with information that includes a *sequence number*, *new-or-resent* and a *data-or-poll* field. The format of this header is shown in figure 3. The 1-Byte *new-or-resent* field can be used by clients to keep loss statistics of frames and the *data-or-poll* field to indicate when the multicast frame is data or a poll message. The *sequence number* field contains an identification number assigned by the Smart Server to be used in the retransmission algorithm. In our testbed, we assume that the sequence number is assigned to frames in the same order that they have left the original source, but this could be different if the protocol is implemented directly in the AP rather than in the same machine as the original source. Although we set 1-Byte for these fields for simplicity in the later performance analysis, we could have used less bits in both fields. Hence, once a small group of data frames has been forwarded, the Smart Server selects one of the clients included in its *Register of Clients* to be polled. The poll message is sent by the Smart Server to the unicast address formerly sent by the Smart Client and its format is shown in figure 4. The *From Sequence Number* and *To Sequence Number* fields indicate a window of datagrams being polled, and the *Bits Array*[1] associates one bit for each frame included in the window. For our measures, we set the group size in 8 to use just one byte, so each bit represents one of the 8 previously sent datagrams. As data frames arrive to Smart Clients, their sequence number is checked. While there are not missing frames, the retransmission protocol header is removed and the frame is passed to the client application. When a frame is lost, those frames with sequence number greater than the lost frame are held in a buffer until the packet is retransmitted and successfully received, or the buffer filled up at which point the lost packet

[1] This field is added only in the poll response message.

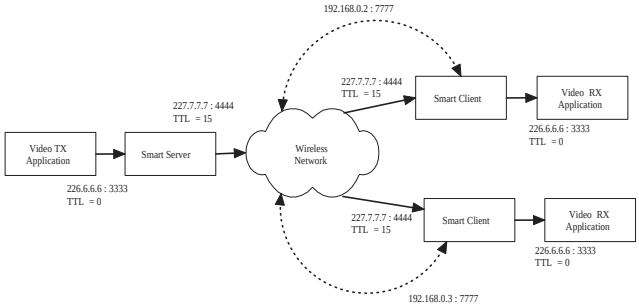

Fig. 5. GroupPoll (GP) Protocol Network Diagram. Solid Lines represent multicast transmissions and Dashed Lines the polling process.

is skipped and all waiting packets (up to the next lost packet in the buffer, if there is one) are sent to the client application. When the poll message arrives asking for the reception status of the last frames group, the Smart Client sets the *Bits Array* of the poll message depending if the frames arrived successfully or not. Once this poll response message is received by the Smart Server, it starts the process of resending the missing frames and advancing the sliding window depending of the message received. Thus, at any moment, the sender maintains a list of sequence numbers it is permitted to send. These are frames sent-but-no-ack and frames not-yet-sent. It is important to notice that due to the use of this sending window, the Smart Server can keep sending frames while is waiting for arrival of the poll response message without blocking immediately after send a group of frames. Once all frames in the sending window are sent-but-no-ack, the server enters in a blocking condition which could generate the loss of frames if they arrive during this condition. To avoid this situation, every time that a poll message is sent a timer is started. When a timeout occurs before receiving a reply from the chosen Smart Client, the whole last group of frames is resent and the lower edge of the window is advanced.

As our goal is not to achieve full reliability in all hosts but a reduction in frame loss for most of the hosts, we accept the loss of a packet if a retransmission limit is reached. Therefore, the retransmission limits have to be carefully set to decide how many times a frame should be resent before giving up. In our measures, we fix this limit to one retransmission, this parameter was chosen to achieve the delay/loss tradeoff required for video applications. A network diagram of the protocol is shown in figure 5. Figure 6 shows how the protocol works over the time. To compare our proposal performance, we also developed two other protocols that represent the worse and the best scenario in terms of reliability. The first one is called *Multicast-to-Multicast protocol (M2M)* and as its name says, it just works like a transparent tunnel for the multicast stream. Thus, this protocol represents the case of a multicast transmission over wireless links using the 802.11 standard. This can be seen as the worse scenario in terms

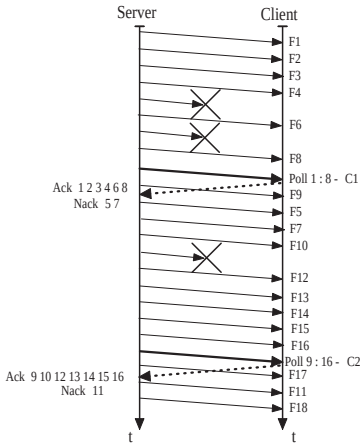

Fig. 6. GroupPoll (GP) Protocol over time Diagram. *Fx* is the Frame*x*, *Poll x:y - Cz* the Poll sent to Client *z* asking for frames in the range [*x*-*y*], and *Ack x y z Nack u v w* the Poll response message saying that frames *x, y and z* were successfully received and *u, v and w* were lost.

of reliability because of the high frame losses caused by the lack of any retransmission technique. The second protocol was called *Multicast-to-Unicast protocol (M2U)* and is a representation of the protocol proposed in [?]. In this protocol, the Smart Server treats each multicast/broadcast message like multiple unicast messages directed to each Smart Client. To do this, it uses the Register of Clients to get the client unicast addresses and then it uses the standard 802.11 unicast transmission for each one interested in receiving the message. As we said before, although it is an interesting option because it improves reliability and is the best scenario in terms of reliability, we considered that in this solution there is too much redundancy which harms throughput and delay. As a consequence of this redundancy, this protocol performance drops quickly as the number of clients or transmission rate grow even though the not-scalable behavior of this approach results incompatible with the multicast concept, we develop it to get upper bounds for reliability and delay. In figures 7 and 8, network diagrams are showed for these protocols. In addition, figures 9 and 10 show how both protocols behave over time.

5 Measurement Details and Results

This section presents the results that were obtained using the scenarios and protocols previously described in sections 3 and 4. In all our measures we used Java-capables machines in the same IP network and that were running Debian Linux with the kernel 2.4.22. Both clients used 802.11b Orinoco Silver Wireless Cards and the network interface of the server was connected through a

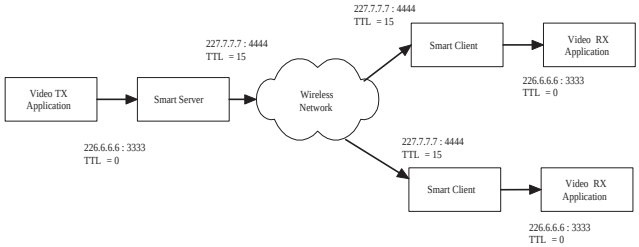

Fig. 7. Multicast-to-Multicast (M2M) Protocol Network Diagram. Solid Lines represent multicast transmissions.

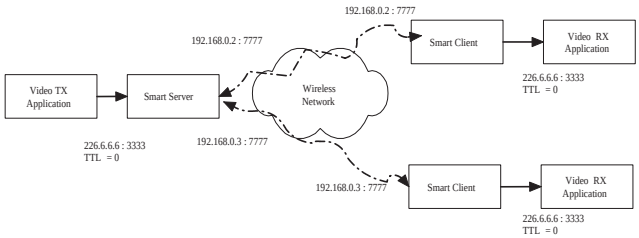

Fig. 8. Multicast-to-Unicast (M2U) Protocol Network Diagram. Solid Lines represent multicast transmissions and Dot-Dashed Lines represent unicast transmissions.

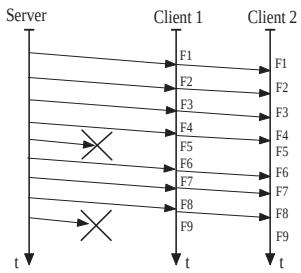

Fig. 9. Multicast-to-Multicast (M2M) Protocol over time Diagram

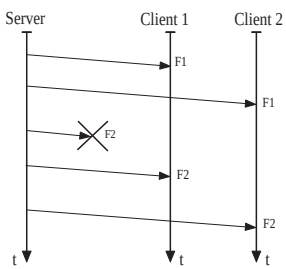

Fig. 10. Multicast-to-Unicast (M2U) Protocol over time Diagram

crossover cable with an AP-1000 Orinoco Access Point. To emulate the outdoor scenario we used an attenuator that covers the AP transmitting antenna. Thus, we were able to obtain the same values of SNR that those observed in the real outdoor scenario. It is important to notice that in the outdoor scenario is possible for nodes on opposite ends of the WLAN coverage area to be unable to hear each other. Under these conditions, the well known hidden and exposed terminal problems are quite likely to occur. To solve this, we forced to all wireless devices to use the RTS/CTS protocol when they transmit frames

unicast without considering the size of the frames. To perform our measures, we generated multicast frames with a 972 [Bytes] size in the server application layer simulating a video transmission. The first 4 [Bytes] of each frame sent by the original source were used to add a sequence number and control information to automate the measures. Figures 11 (right) and 11 (left) show the percentage of packet loss calculated over a total of 10000 frames that were sent over the measurement time using a specific rate. We run this experiment several times varying the transmission rates and data were compiled until obtain averages values with an intervals of confidence of 5% for a confidence level of 95%. Also, it is important to notice that, like in the case of the measurements made with the RealNetworks technology, the same measurement was made using a Fast-Ethernet wired network instead of a wireless network, the losses remain close to 0% using the same two clients. This is an important point because it means that our protocol passed a sanity check that allows us to discard losses that could have exist because our implementation of the protocol. Besides, our network scenarios consider that the clients distance to the transmitter is one hop (without intermediate nodes) so losses due to buffers saturation during the intermedia routing of frames can be discarded. Let us start by examining the loss percentage of the indoor scenario. In figure 11 (right) curves of the two clients are presented. The M2U protocol losses remains below 0.05%, while the M2M protocol shows that all results are less than 0.2%. Our proposal measures appear in the upper curves, nevertheless it is possible to appreciate that the loss percentage do not exceed the 0.6%. All curves show a small growth as the transmission rate increases because more frames are sent in a less time, and in presence of fading or channel interference it causes greater losses. Although in our proposal losses are over the other protocols results in this scenario they are still negligible, for instance, in terms of video quality. Furthermore, one reason of this loss is the overhead added by the unicast polling process which could be useless when the channel is almost ideal. It could be avoided if the server had wireless channel information to decide when the conditions are good enough to stop this process and just directly deliver frames in standard multicast fashion. This issue will be developed as part of our future work. In the other hand, we have the outdoor scenario. Curves in figure 11 (left) shows that M2U has a poor performance because it has to perform independent transmissions for each client. In addition, when the wireless channel quality is poor, the M2U protocol have to perform several error recovery which makes it highly inefficient. The M2M protocol average losses is around 12% while our proposal average losses remains nearly the 6% which traduces in a much better video quality. Thus, our proposal reaches a better throughput as the channel conditions become worse.

6 Performance Analysis

In the following, we analyze the theoretical performance of our proposal. In figure 12, a transmission scheme for all protocols is shown. As our protocol

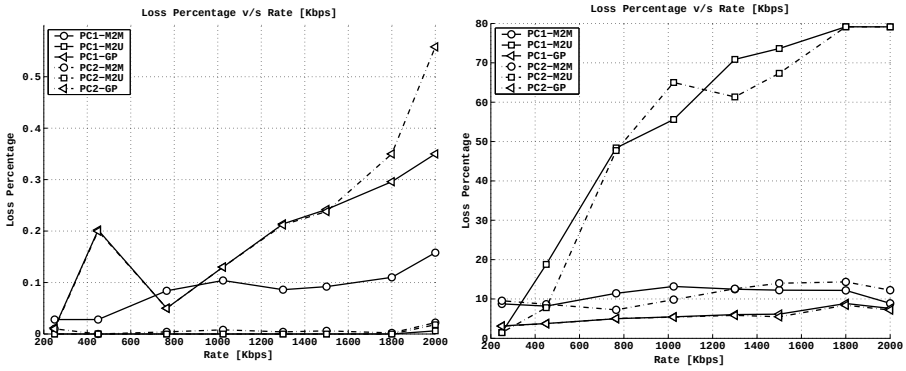

Fig. 11. Results obtained in the indoor (left) and outdoor (right) scenarios for the transmission protocols implemented using transmission rates between 250 and 2000 [kbps] with frames of 1000 [Bytes]

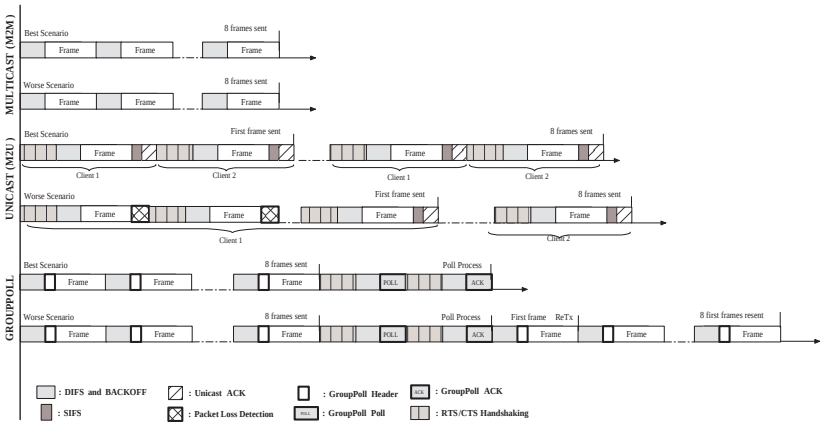

Fig. 12. Best and Worse Performance Scenarios for the M2M, M2U and GroupPoll protocols. Dot-Dashed Lines represents variable time where several transmissions took place.

works based on a group of eight multicast frames, we set this comparison over that ground. We omit some frame transmissions in the figure for space reasons, but they were considered in this analysis. Besides, we assume that there are only two clients present in the network. With the purpose of study these effects in detail, we point out differences in protocols considering both, the best and worse scenarios depending of the success of the transmission of frames. Hence, we assume that in the best scenario there is not frame loss while in the worse scenario, a successful transmission always takes place in the last attempt before the frame is dropped. Based in the figure 12 we define the time equations for all protocols. These equations are shown below.

$$\text{M2M Best Scenario Time}=(\text{DIFS}+\text{BACKOFF}+\text{FRAME})^{*}\ 8\ \text{Frames} \tag{1}$$

$$\text{M2M Worse Scenario Time}=(\text{DIFS}+\text{BACKOFF}+\text{FRAME})^{*}\ 8\ \text{Frames} \tag{2}$$

$$\begin{aligned}\text{M2U Best Scenario Time}=(((&\text{DIFS}+\text{BACKOFF}+\text{RTS}+\text{SIFS}+\text{CTS}\\ &+\text{SIFS}+\text{FRAME}+\text{SIFS}+\text{ACK})^{*}\ 2\ \text{Clients})^{*}\ 8\ \text{Frames})\end{aligned} \tag{3}$$

$$\begin{aligned}\text{M2U Worse Scenario Time}=((((&\text{DIFS}+\text{BACKOFF}+\text{RTS}+\text{SIFS}+\text{CTS}\\ &+\text{SIFS}+\text{FRAME}+\text{SIFS}+\text{ACK})^{*}\ 4\ \text{Retries})^{*}\ 2\ \text{Clients})^{*}\ 8\ \text{Frames})\end{aligned} \tag{4}$$

In [4], Retries represents the LongRetryLimit constant of the IEEE802.11b standard. We use this value because we set our RTSThreshold limit in 0 using the RTS/CTS for all unicast transmission. Thus, every frame size is greater than this threshold which sets this limit in 4.

$$\begin{aligned}\text{GroupPoll Best Scenario Time}=((&\text{DIFS}+\text{BACKOFF}+\text{GPFRAME})\\ &^{*}8\ \text{Frames})+(\text{DIFS}+\text{BACKOFF}+\text{RTS}+\text{SIFS}+\text{CTS}+\text{SIFS}\\ &+\text{POLLFRAME}+\text{SIFS}+\text{ACK})+(\text{DIFS}+\text{BACKOFF}+\text{RTS}\\ &+\text{SIFS}+\text{CTS}+\text{SIFS}+\text{POLLACK}+\text{SIFS}+\text{ACK})\end{aligned} \tag{5}$$

$$\begin{aligned}\text{GroupPoll Worse Scenario Time}=((&\text{DIFS}+\text{BACKOFF}+\text{GPFRAME})\\ &^{*}8\ \text{Frames})+(\text{DIFS}+\text{BACKOFF}+\text{RTS}+\text{SIFS}+\text{CTS}+\text{SIFS}\\ &+\text{POLLFRAME}+\text{SIFS}+\text{ACK})+(\text{DIFS}+\text{BACKOFF}+\text{RTS}\\ &+\text{SIFS}+\text{CTS}+\text{SIFS}+\text{POLLACK}+\text{SIFS}+\text{ACK})+((\text{DIFS}\\ &+\text{BACKOFF}+\text{GPFRAME})^{*}\ 8\ \text{Frames})\end{aligned} \tag{6}$$

Here, in [5] and [6], GPFRAME represents a frame with a GroupPoll Header, POLLFRAME and POLLACK the Poll and Poll Response messages respectively. As transmission time for each frame depends on the transmission rate used, in figure 13 we present curves obtained from this equations for all data rates used in the IEEE802.11b standard. We also assume the values showed in table 1 for this analysis. These results allow us to examine the impact in latency added by our protocol and the tradeoff between reliability and delay when we compare our proposal with other protocols. In figure 13 can be appreciated that M2U protocols present a larger delay in both scenarios and this increases quickly as the number of clients grows. It is also clear that our protocol remains very close to the M2M delay in the best scenario (no retransmissions are needed), but when the channel quality gets worse it reaches almost the double (retransmissions are always needed). In consequence, our protocol leads us to achieve a better throughput over a wireless channel with poor quality when multicast traffic is sent. We also showed that in the worse scenario the latency of our protocol would be constrained to nearly the double of a normal multicast transmission but reducing almost by a half its losses. In this way, our tradeoff is better quality adding controlled latency to improve the overall efficiency in multicast transmission.

Table 1. Size and Time values used in the IEEE802.11b

Parameter Name	Size [Bytes]
RTS	20
CTS	14
ACK	14
MAC Header	34
IP Header	20
UDP Header	8
Application Layer Packet	972
GroupPoll Header	3
Poll Frame	2
Poll Response Frame	3
Parameter Name	**Time [μs]**
DIFS	50
SIFS	10
Average Backoff Time	310 *
PHY Preamble and Header Duration	192 **

Note*: see [?]; Note**: Long preamble, 24 [Bytes] sent with a 1 Mbps rate.

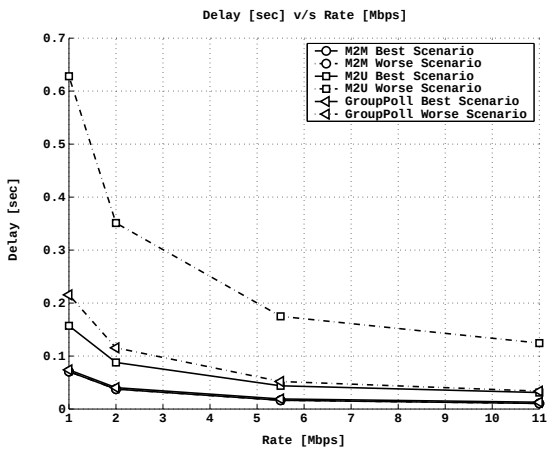

Fig. 13. Best and Worse Time Performance Scenarios for the M2M, M2U and GroupPoll protocols.

7 Conclusions and Future Work

In this paper, we presented a proposal for actual IEEE802.11 networks to improve their reliability for multicast data. A novel feature of this proposal is that its reliability control based on a polling service along with controlled retrans-

missions; besides, it does not require modifications to the 802.11 MAC layer because it works in the application level and use interceptors to forward the multicast stream. We recognize that the Smart Server could be seen as a single point of failure in our implementation, but our proposal considers the use of one Smart Server for each AP in the network. Thus, a commercial implementation could include the Smart Server protocol in the AP, inheriting in that case, the robustness of a network with infrastructure.

Our results show that our protocol performs better than other proposed in terms of reliability as well as data throughput in our measurement scenarios. Further work in this area will concentrate in improving our protocol adding features that will allow us to obtain delay and jitter measures.