A Framework and Experimental Study for Discrimination of Collision and Channel Errors in Wireless LANs

Georgios Kyriakou[+], Donatos Stavropoulos[+], Iordanis Koutsopoulos[⊕], Thanasis Korakis[⊕], and Leandros Tassiulas[⊕]

[+]Department of Computer Engineering and Telecommunications, University of Thessaly, Greece
[○]Centre for Research and Technology, Hellas
{geokiria,dostavro,jordan,korakis,leandros}@uth.gr

Abstract. A fundamental unresolved problem in wireless networks is that of distinguishing packet errors that are caused by deteriorated link conditions and noise, from errors that occur due to packet collisions. In this paper, we develop advanced algorithms based on Cyclic Redundancy Check (CRC) [10] that solve this problem. Specifically, our innovation is that we form multiple CRCs, each of which is responsible for a different segment in a packet. The CRCs are appended after each segment. In this way, we can essentially visualize the pattern of errors across the packet. If the number of successive erroneous segments exceeds a threshold, we decide in favor of a collision. We integrate our approach with SampleRate. Our approach is implemented in MadWiFi [7] and is validated through realistic test-bed experiments. Our technique is shown to significantly outperform current error identification techniques, while having low complexity, and it constitutes an approach that can be readily incorporated in existing wireless protocols.

Key words: Packet Loss Differentiation, CRC-based decision making, experimental validation

1 Introduction

In a wireless network, different nodes perceive the channel to be in different states. One node may sense the channel idle and thus get ready for transmission, while another is still transmitting (hidden terminal). This can easily result in "undetectable"collisions.

In the wireless channel, a packet may be lost due to noise or collision. In order to avoid losing a packet due to noise, the sender should choose the optimal transmission rate. If the sender chooses a higher rate than the one that can be supported by the current channel quality, the packet is vulnerable to noise. If the sender chooses a rate that is lower than the one the channel can support, the utilization of the medium is low. It is here where rate adaptation algorithms come into stage. Their goal is to sense the medium in various ways and choose the

appropriate transmission rate by estimating channel quality. Contemporary rate adaptation algorithms achieve this by comparing the number of successful and unsuccessful transmissions. They distinguish between the two cases as follows:

- if a packet is received correctly, the sender shall receive an ACK
- if an ACK is not received, the packet is considered to be lost due to channel errors

The reasoning above is based on the premise that packet losses are caused by noise. Such a crude assumption is naturally the case in the 802.11 protocol [9], since it describes the CSMA/CA (Carrier Sensing Multiple Access/ Collision Avoidance) with back-off mechanism, according to which *collisions are avoided*. The back-off timer is simply a countdown timer which gets its values in a range of values $[0, CW]$. Initially, CW is set to its minimum value. Each time a transmission fails, CW is doubled till an upper limit. After a successful transmission, CW is reset to its initial/minimum value. Every time a node is ready to transmit, it senses the channel. If it senses it as idle, it starts the BackOff timer, while if the channel is sensed busy, the BackOff timer is frozen. When the timer becomes zero, a transmission is attempted.

This mechanism is not designed for current and envisioned large wireless networks densities. Another issue is that CSMA/CA only tries to *prevent* collisions, not to detect them. It is clear that the protocol mistakenly considers packets corruptions due to collisions as corruptions due to noise. As a result, rate adaptation algorithms misinterpret the quality of the channel, especially in congested networks, and therefore they choose a transmission rate that is lower than the appropriate one. Thus, one may suggest that this affects each node independently and therefore the throughput loss linearly increases as the number of the nodes in the system increases. Unfortunately this is not the case.

CSMA/CA and the back-off mechanism are designed to be "fair". That means that after a sufficiently large amount of time, each node should transmit the same number of packets, regardless of the rate each node transmits with. The way 802.11 is implemented today, each data packet has the same size (about 1500 bytes). Therefore, if two nodes share the medium and the one uses 1 Mbps while the other 54 Mbps, after a while each node should have transmitted the same number of packets, thus the same amount of data. This happens although the fast node occupies the channel much less than the slow node. As a result, their throughput will be identical and lower than the throughput the slowest node would have if it were transmitting alone. This is known as the 802.11 anomaly [17] and it clearly shows how a single node, that *mistakenly* chooses a low rate, can significantly reduce the throughput of the entire system.

Our motivation to design a scheme, which would successfully discriminate a packet loss from a collision and therefore make appropriate rate adaptation decisions, stemmed from the need to avoid unnecessary degradations in transmission rate, that prove detrimental in the performance of the system.

1.1 Related Work

There is significant amount of work on the problem of the packet loss differentiation. CARA [1] was one of the first rate adaptation algorithms that could determine whether a packet was lost due to a collision or due to a noisy channel. The key idea of CARA is that the transmitter combines adaptively the Request-to-Send/Clear-to-Send (RTS/CTS) exchange with the Clear Channel Assessment (CCA) functionality to differentiate frame collisions from frame transmission failures caused by channel errors.

Robust Rate Adaptation Algorithm, also known as RRAA [2], uses an adaptive RTS filter to suppress collision losses when it estimates the loss ratio. The basic idea is to leverage the per-frame-RTS option in 802.11 standards and selectively turn on RTS/CTS exchange to suppress collision losses. While RTS is well known as an effective means to handle hidden terminals, loss estimation is used to decide when and how long RTS should be turned on or off.

Loss Differentiated Rate Adaptation or LDRA [3] introduced the lowest rate retransmission to differentiate packet loss. When a packet loss occurs for the very first time, the sender immediately retransmits the frame at the lowest rate instead of the same rate, so as to determine if the channel did not further degrade. Also it checks if a beacon frame is received during the latest period to determine whether it has lost connection with the receiver.

Congestion-Aware Rate Adaptation [4] is an attempt to probabilistically identify congestion packet losses and minimize their impact on rate adaptation. A congestion measurement technique was used in the design and implementation of a new rate adaptation scheme called Wireless congestion Optimized Fallback (WOOF). The use of a congestion metric enables the rate adaptation algorithm to differentiate packet losses due to congestion from those due to poor link quality.

Another attempt to solve the problem of packet loss differentiation is COLLIE [12] which manages to do this by using newly designed metrics that examine error patterns within a physical-layer symbol in order to expose statistical differences between collision and weak signal based losses. It can also use an optional COLLIE server which increases accuracy of the collision detection.

LD-ARF [5] is a loss differentiation algorithm which uses a short checksum field after the packet header to differentiate the packet losses and adjust its rate accordingly. If all stations in a WLAN can hear each other (i.e., there is no hidden terminal), a collision occurs only when more than one station send data frames in the same time slot. In this case, both the header and the body of the packet will be corrupted. However, if there is only one station sending a data frame and the frame is lost due to link errors, there is a high probability that the receiver will receive the header correctly. This is because in general the header is much shorter than the whole frame.

1.2 Our Contribution

With the exception of the last algorithm, all methods above have a common characteristic. They try to detect a collision *after it has happened*, e.g by sensing the channel after a transmission or by RTS probing. Only LD-ARF tries to detect the reason the packet was lost in real-time by looking into the erroneous packet and simply checking if the entire packet or only part of it was lost. However LD-ARF has the issue that it is not designed to work well when there are hidden terminals.

This is where our algorithm differentiates from LD-ARF. Based on the approach of the small checksum field, we take one step forward by *assigning different small CRC fields to different segments in the body of the packet*. Thus, we are able to detect collisions, even when there are hidden terminals and we can visualize the error patterns in the packet in order to differentiate between channel error and collision. When the number of successive erroneous CRCs is beyond a specific threshold, we decide that the packet was lost due to collision. Otherwise we decide that the packet was lost due to a channel error. Thus, we adapt the rate more efficiently and we utilize the medium much better than the algorithms that do not have the ability to discriminate the losses, including legacy 802.11.

Since we did not make use of the RTS/CTS like previous algorithms (e.g. CARA, RRAA), the overhead of our approach is low, and a more efficient utilization of the medium is achieved. The amount of additional transmissions of the CRCs in the packets is much lower than the overhead caused by the retransmission of the whole erroneous packet back to the sender in the COLLIE approach. We integrate our new scheme with SampleRate [6], which is a very sophisticated algorithm, and compare our results with SampleRate, contrary to most of the previously mentioned algorithms that use the simpler ARF approach [11].

The rest of the paper is organized as follows. In section 2, we describe the problem in detail, in section 3 we present our novel framework of introducing multiple CRCs in the packet body and in section 4 we depict our derived experimental results.

2 The Problem of Distinguishing Collisions from Channel Errors

2.1 Packets Corrupted due to Channel Errors

In a wireless channel, errors due to interference or noise occur primarily in bursts [14]. This means that the pattern of erroneous bits in a packet is completely random, and there are some parts of the packet that can be decoded correctly (Fig. 1).

Depending on the statistical assumptions about channel errors, each separate bit is subject to error with a certain probability. Such a pattern of sporadic errors

in a packet can be used as an indication that the packet was lost due to channel errors.

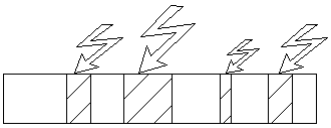

Fig. 1. A sporadic pattern of erroneous bits in a packet can be assumed to be the outcome of channel errors.

2.2 Packets Corrupted due to Collisions

Due to the specifics of the CSMA/CA mechanism, the back-off timers of **at least** two nodes have to reach zero simultaneously in order for a collision to occur. In that case, all packets of involved transmitters will be corrupted completely, including their headers (Fig. 2). The chances for collision to happen are roughly proportional to the number of nodes in the area. The increasing number of WiFi devices makes these chances non-negligible.

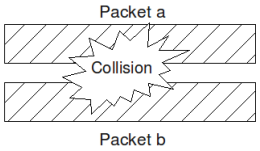

Fig. 2. A packet corrupted due to a collision in case all involved nodes hear each other.

The collision event described above with the simultaneous expirations of back-off times occurs when all nodes hear each other. Nevertheless, the situation is different in case there exist hidden terminals.

Hidden Terminal
When two or more nodes are hidden from each other (e.g. nodes A and C in Fig. 3), each node senses the medium as idle and can initiate transmission even though another (hidden) node could already be transmitting. This can cause severe collisions at nodes that hear all the transmitters (e.g. node B).

When two packets collide, a part of the one packet overlaps with a part of the other packet. These parts are corrupted and packets cannot be decoded correctly (Fig. 4). One can easily deduce that in this case the pattern of corrupted bits consists of *successive* corrupted bits.

When a chunk of data is to be transmitted, it is divided into packets of the same size (usually around 1500 bytes), which are then transmitted over the air.

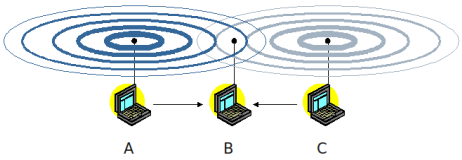

Fig. 3. Hidden terminal problem.

Therefore, each transmitted packet will be of the same size, except for the last packet of each data chunk, whose size will be the remainder of the data chunk size. This process is the packet segmentation [13]. As a result, the average error burst due to collision will be half the length of the packet size, given that the packets are transmitted at the same rate, since the different ways two packets can collide have the same chances to occur.

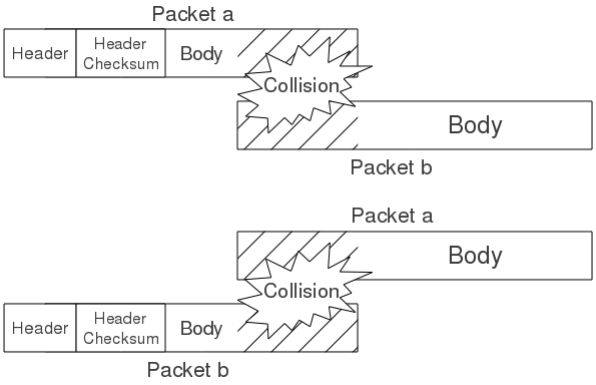

Fig. 4. Packets corrupted by a collision in a hidden terminal scenario.

The Capture Effect

An important issue that must be taken into consideration in case of a collision is the capture effect [15]. When two packets with different signal strength collide at a receiver, the one with the high signal strength will be decoded correctly, whereas the other, weaker-signal-strength packet will be treated as not received. As a result, the transmitter that sends packets with the high signal strength will suffer few losses as opposed to the other transmitter. Nevertheless, there is also an important side effect. Since the transmitter that sends the weak packets suffers all the collisions, it will eventually increase its back-off timer, while the transmitter that sends the strong packets will keep its back-off timer to minimum. This will lead to a small probability of claiming the channel for the former node. This phenomenon has important repercussions in fairness.

3 The Proposed Method

If we can distinguish the cause of a packet transmission failure, we can improve the performance of rate adaptation algorithms. To do this, cooperation between the transmitter and receiver through some signaling is needed. Specifically:

- If the receiver successfully receives a packet, it notifies the sender with an ACK,
- If the receiver gets a packet that is damaged due to channel errors, it sends a NAK,
- If the receiver gets a damaged packet due to collision, it does not send anything.

The NAK is sent at the basic rate of 1 Mbps in order to ensure the integrity of reception. With this methodology, a transmitter can be informed about the reason of a transmission failure and thus be able to adapt its rate appropriately. The detection of the cause of packet error is of course performed at the receiver, based on the pattern of erroneously decoded packet bits.

In order to detect the cause of packet corruption, we need to specify the amount and the distribution of erroneous bits in the packet, i.e. define an error pattern. The straightforward way to do that is to repeat each bit a number of times through different checksums thus increasing the possibility of successful reception. However, this would lead to tremendous amount of overhead. Instead of ensuring each bit by practically having one checksum per bit, *we divide the payload bits into a number of segments and we compute a CRC for each segment before packet transmission* (Fig. 5).

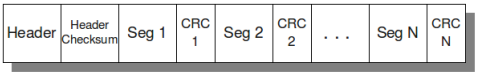

Fig. 5. Structure of our data packet.

We choose to use CRC checksum because it is very efficient in detecting error bursts [14], such as those caused by a poor quality wireless channel. When a packet is received, we verify its integrity by recalculating the CRC for each packet segment and compare it with the CRC in the packet, thus exporting the error pattern.

We also decided to place each CRC after the segment out of which it is computed. If we had placed all CRCs together, say at the beginning, middle or end of the packet, a collision or an error burst could result in the complete loss of those CRCs and therefore it would hinder the receiver in determining the cause of corruption. By distributing CRCs uniformly, *we can determine the cause of erroneous reception by finding the number of successive erroneous CRCs. If this number is below a given threshold, we deduce that the packet was most likely corrupted due to channel errors (Fig. 1). Otherwise we allege that it was corrupted due to collision (Fig. 2 and Fig. 4).*

The rationale behind the scheme is that a burst of channel errors does not affect each bit, unlike collision. Therefore, if we observe that many *successive* erroneous CRCs exist in a row, including a number of successive errors larger that the average size of channel error burst, we can deduce that this was caused by a collision. Moreover, if there exists one or more correct segments between erroneous segments, we can assume that the packet was lost due to channel errors, since a collision would have destroyed all segments in the overlapped area. We note that the extremely rare case that a node ends up with such channel quality and rate that result in complete corruption of packet due to channel errors, our scheme will not detect the packet loss cause. However, the sampling mechanism of SampleRate will be able to restore communication by choosing an appropriate rate for the specific channel conditions.

Besides the payload CRCs described above, we add a small checksum for the MAC addresses of the transmitter and the receiver. The reception integrity of these addresses is very important in order to keep per-node statistics correctly and send the NAK packet to the appropriate node. Since rate adaptation algorithms calculate statistics based on the successful or unsuccessful reception of data packets, we decided to insert our checksums only in data packets. In order to have low overhead, we used 16-bit CRC instead of 32-bit CRC that is used by 802.11 for the whole packet. A 16-bit CRC will detect 99.9985% of all errors and a 32-bit CRC 99.9999% of them. Hence using a 32-bit CRC would double the additional overhead without significantly increasing the detection accuracy. As a result, the size of our overhead, when using 20 segments as we did in our experiments, is very low (6 bytes for MACs' checksum + 20 segments * 2 bytes/segment's CRC = 46 bytes per packet) as compared to the average size of a packet ($46/1500 \simeq 3\%$), and thus the additional overhead load is equally small.

The mechanism above for packet loss differentiation was implemented on the MAC layer of the open-source MadWifi driver for Atheros chip-sets. We decided to incorporate the implementation of our mechanism for packet loss differentiation into SampleRate, since, in our opinion, it is more sophisticated than the other algorithms used by MadWifi and it is the default one in MadWifi. A nice feature of SampleRate is its inertia. SampleRate does not respond immediately to channel changes, in order to prevent degradation of rate due to sparse collisions. This surely makes our goal harder, since our extra information is considerably more useful in a network with several collisions.

An ACK packet is implemented on the hardware (PHY layer). Ideally, we would like the NAK packet to be also implemented on the PHY layer, so it could be sent in SIFS time after reception. Unfortunately, this is not possible and therefore we implemented NAK on the MAC layer. Each NAK carries triplets of statistics (packet size, rate, successive failures) that need to be updated. Multiple triplets, for different packet sizes and rates, can be contained in a single NAK (Fig. 6). These statistics can be used by the receiver rate adaptation algorithm, in order to adjust the rate more accurately.

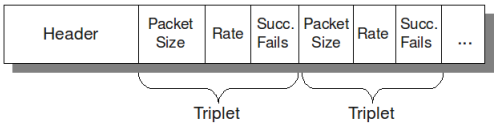

Fig. 6. Structure of a NAK packet.

There exists the issue of how frequently a NAK packet should be sent, considering that it is not possible to send it in SIFS time. It should not be sent too frequently, so as to ensure that no additional traffic is generated, and it should not be sent too rarely, so that the rate adaptation algorithm adapts promptly to changes of the channel conditions. The goal is that the rate adaptation algorithm should be updated in time, while the additional traffic generated by the NAKs remains low. The threshold is chosen in proportion to the rate adaptation algorithm and the frequency of rate change. SampleRate chooses its rate once every few seconds, so we send the NAK packets with twice that frequency. A visualization of our scheme at the receiver and the transmitter side is shown in Figures 7 and 8.

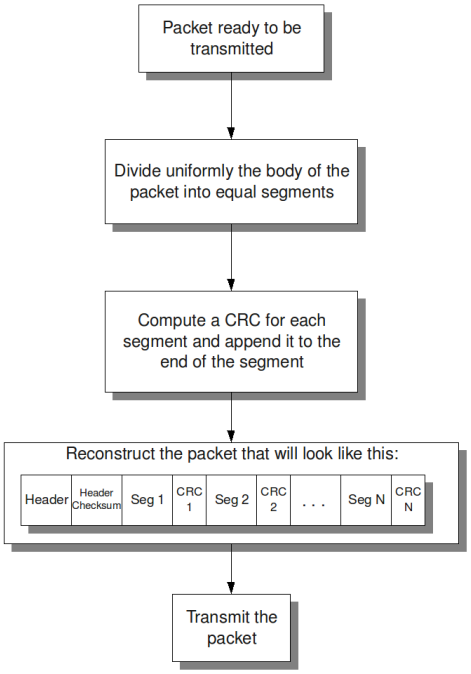

Fig. 7. Transmission flowchart.

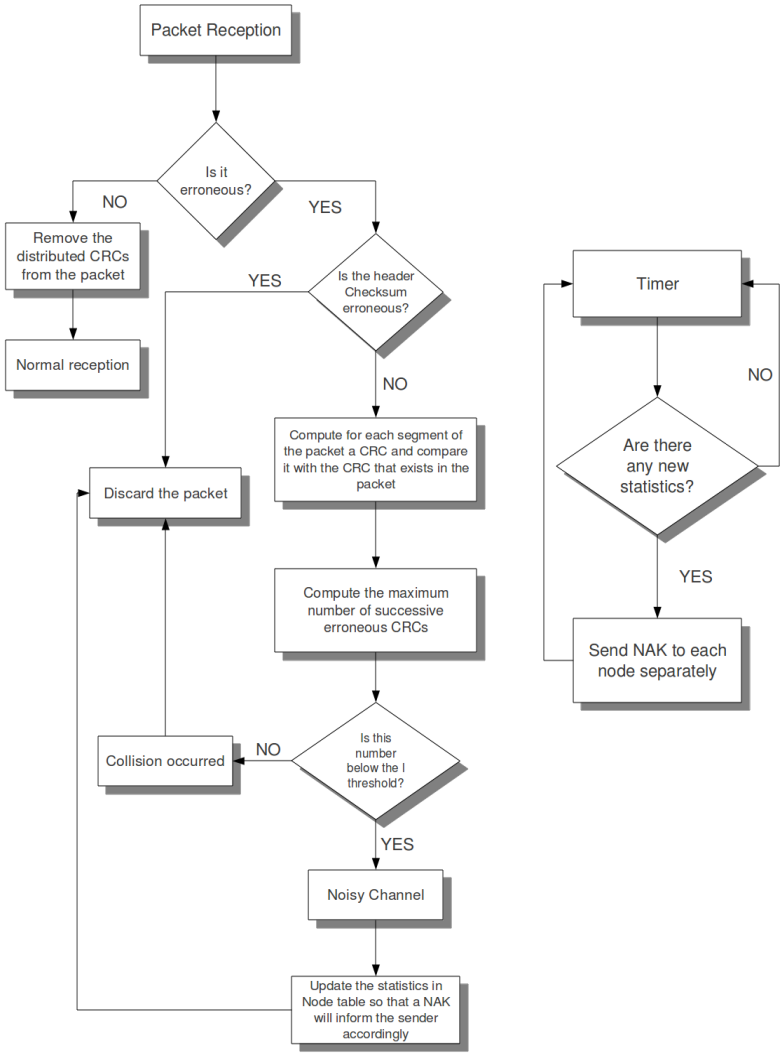

Fig. 8. Reception flowchart.

4 Experimental Results

4.1 Test-bed Description

In order to evaluate the performance and study the behavior of rate adaptation with loss differentiation that we have implemented, we use our large-scale programmable NITOS wireless test-bed [8].

NITOS Test-bed NITOS (Network Implementation Testbed for using Open Source platforms) is a wireless test-bed, that is designed to achieve reproducibil-

ity of experimentation. Users can perform experiments by reserving resource slices (nodes, frequency spectrum) in the test-bed through the NITOS scheduler, that, together with the OMF management framework, supports ease of use for experimentation and code development. It is *remotely* accessible and currently consists of 15 wireless nodes outdoor located in a non-RF-isolated environment. The nodes are equipped with 2 wireless interfaces using Wistron CM9 - mPCI Atheros 802.11a/b/g 2.4 and 5 GHz cards that run MadWiFi open source driver. NITOS is deployed at the exterior of a University of Thessaly campus building.

Measurement Methodology We use Iperf [16], which is a powerful tool for traffic generation and measurement. In our experiments we tried to demonstrate all the possible scenarios and see how well our scheme performs compared to the original rate adaptation algorithm, SampleRate.

Measurements We conducted several different experiments with different topologies and we empirically chose the number of segments and threshold. We experimented on different channels both in 802.11g and 802.11a bands, using both TCP and UDP protocols, with various number of nodes and different durations. For ease of use and for practical reasons, we chose some representative topologies that clearly demonstrate each of the situations that we analyzed in section 2.

To remove any random effects and short-term fluctuations, we ran each test in each topology several times and for time periods from 5 to 20 minutes, depending on the stability of the channel. We alternated between our modified driver and the official MadWifi driver after each run, so we could ensure the channel did not suffer from major changes (due to weather, surrounding networks, or other reasons). In the charts following each experiment, the percentage gain of our modified driver as compared to the original one is noted over the columns.

4.2 Experimentation Results

Collisions when All Nodes Hear Each Other First, we consider a scenario where many nodes coexist in a relatively small area and try to communicate simultaneously. As a result, there is an increase in number of collisions, which inevitably causes a rate degradation. Our scheme with its ability to differentiate packet losses, manages to avoid incorrect reductions in rate. We observe that our scheme outperforms the original rate adaptation algorithm and maintains system throughput in high levels (Fig. 9).

Hidden Terminals In this scenario we try to demonstrate a topology with the hidden terminal problem (Fig. 3). Two nodes, that do not hear each other try to transmit to a third node. As a result, they end up corrupting each other's packets. The CSMA/CA mechanism tries to ease this effect by decreasing the possibility that these two nodes transmit at the same time, but the incapability of each node to detect the reason of packet losses is more than obvious in system throughput (Fig. 10).

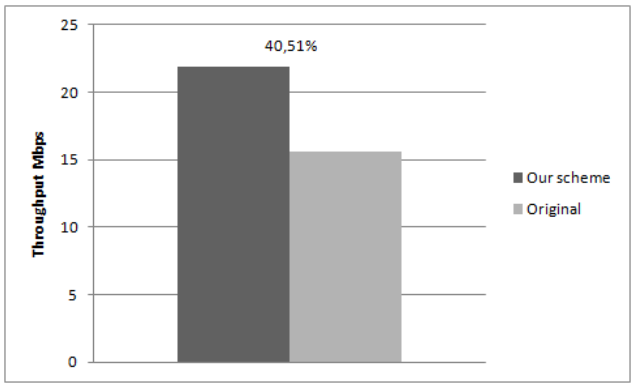

Fig. 9. Congested network throughput.

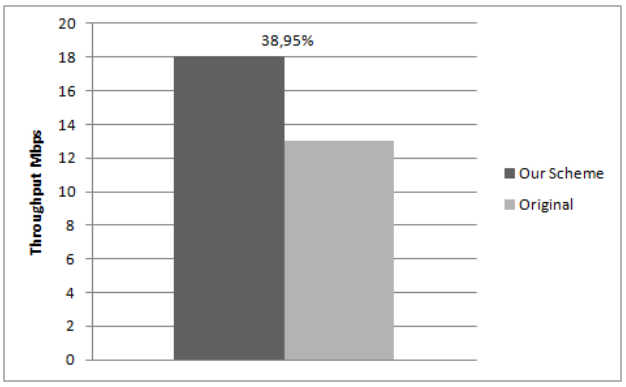

Fig. 10. Performance in a scenario with the hidden terminal problem.

Due to severe collisions occurring when the two nodes transmit at the same time, SampleRate erroneously reduces the transmission rate, thus giving our scheme a clear advantage.

Capture Effect After experimenting with different topologies in order to clearly locate the cause and results of capture effect, we realized that it is a far more common phenomenon than we expected, especially in the 5 GHz band. The following experiment was conducted with 3 nodes in order to be able to isolate the cause of packet losses (Fig. 11). Since all nodes see each other, in order to increase collisions and therefore the capture effect, we decreased the congestion window and therefore the value range of the back-off timer (Fig. 12).

As expected, the throughput of the strong node remained practically the same, since it did not suffer from collisions in the first place. On the contrary, we saw significant improvement in the throughput of the weak node, since it was the only one that was incorrectly reducing its rate due to collisions.

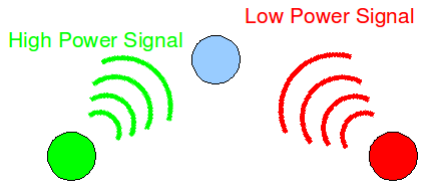

Fig. 11. Topology with the capture effect phenomenon.

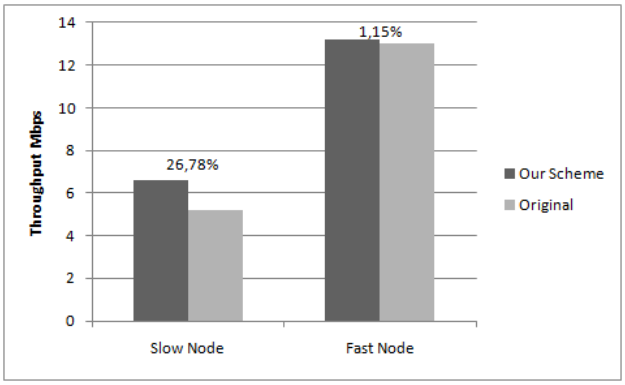

Fig. 12. Performance in a topology with the capture effect phenomenon.

Capture Effect and Half-hidden Terminal While searching for topologies with hidden terminals, we encountered asymmetric situations where many nodes could hear a neighboring node who could not hear them. Thus, by simply making them communicate with a node in between, we ended up with a very interesting scenario (Fig. 13):

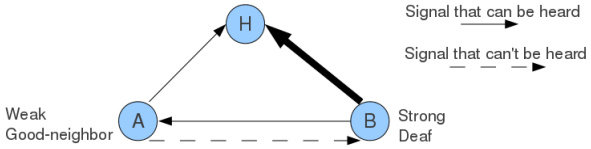

Fig. 13. Asymmetric scenario with hidden terminals.

Node A respects the CSMA/CA protocol and only transmits when node B is idle. On the other hand, node B cannot hear node A. As a result, node B always senses the channel as idle and therefore initiates transmission every time its back-off timer expires. It is obvious that this will cause half of the collisions that a normal hidden terminal topology would cause. Interestingly, if the receiver (node H) is placed near to the deaf node B, the good neighbor (node A) is the only one experiencing collisions due to the capture effect. Unfortunately we end

up with a "deaf-strong"node (node B) that benefits constantly, since it barely gives an opportunity for transmission to the good neighbor / weak node, due to its inability to hear it. In addition, severe collisions force node A to increase its congestion window and get even less transmission opportunities and to reduce its rate, making its packet transmissions longer and more vulnerable to collision.

Our scheme successfully detects that the majority of the packets are lost due to collisions and therefore prevents degradation of transmission rate. Since the high rate keeps transmission time low, packets are less vulnerable to collisions (Fig. 14).

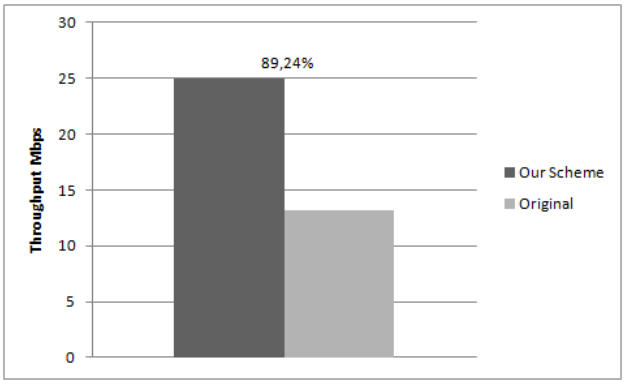

Fig. 14. Performance in the asymmetric scenario with hidden terminals.

5 Conclusions and Future Work

In this paper we proposed a novel scheme based on multiple cyclic redundancy checks per packet, so as to distinguish between collisions and channel errors as the reason behind packet losses. The key idea of our approach is the creation of multiple CRCs at each packet, which allows better visualization of the packet corruption patterns, and hence more reliable decisions as to whether the corruption stemmed from collision or channel error. Subsequently, the correct choice of transmission rate can be performed. Our scheme is shown to outperform legacy IEEE 802.11 schemes, and it is also lightweight, since it can be implemented with minimal overhead. We have implemented our approach and tested it in the realistic environment of the NITOS test-bed.

In the future we plan to systematize our study by employing detection theory to derive optimal number of CRCs and optimal threshold that designate the decision regions for our packet loss identification algorithm.

Acknowledgements

The authors acknowledge the support of the European Commission through STREP project OPNEX (FP7-ICT-224218).

References

1. J.Kim, S.Kim, S. Choi, and D.Qiao, "CARA: Collision-aware Rate Adaptation for IEEE 802.11 WLANs", *in Proc. of INFOCOM,* 2006.
2. S. H. Y. Wong, S. Lu, H. Yang, and V. Bharghavan, "Robust Rate Adaptation for 802.11 Wireless Networks", *in Proc. MobiCom,* 2006.
3. S. Biaz and Shaoen Wu, "Loss Differentiated Rate Adaptation in Wireless Networks", *in Proc. IEEE WCNC,* 2008.
4. P. Acharya, A. Sharma, E. M. Belding, K. C. Almeroth, and K. Papagiannaki, "Congestion-Aware Rate Adaptation in Wireless Networks: A Measurement-Driven Approach", *in Proc. SECON* 2008.
5. V. L. Qixiang Pang and S. C. Liew, "A Rate Adaptation Algorithm for IEEE 802.11 WLANs Based on MAC-Layer Loss Differentiation", *in Proc. IEEE BROADNETS,* 2005.
6. J.C. Bicket,"Bit-rate Selection in Wireless Networks",MIT Master's Thesis, 2005.
7. http://madwifi-project.org/
8. http://nitlab.inf.uth.gr/NITlab/
9. IEEE802.11 http://standards.ieee.org/getieee802/download/802.11-2007.pdf
10. Cyclic redundancy check http://en.wikipedia.org/wiki/Cyclic_redundancy_check
11. A. Kamerman and L. Monteban, "WaveLAN-II: a high-performance Wireless LAN for the Unlicensed Band", *Bell Labs Technical Journal,* vol.2, no.3, pp.118-133, Aug. 1997.
12. S. Rayanchu, A. Mishra, D. Agrawal, S. Saha, Suman Banerjee, "Diagnosing Wireless Packet Losses in 802.11: Separating Collision from Weak Signal", *in Proc. INFOCOM 2008.*
13. http://en.wikipedia.org/wiki/Packet_segmentation
14. ERROR CONTROL CODING From Theory to Practice, Chapter 1.12 CODING FOR BURST-ERROR CHANNELS, P. Sweeney, University of Surrey, Guildford, UK.
15. K. Whitehouse, A. Woo, F. Jiang, J. Polastre, and D Culler, "Exploiting the capture effect for collision detection and recovery", *in Proc. EmNetS-11,* 2005.
16. http://iperf.sourceforge.net/
17. M. Heusse, F. Rousseau, G. Berger-Sabbatel, A. Duda, "Performance Anomaly of 802.11b", *in Proc. IEEE INFOCOM,* 2003.