ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijareeie.com

Vol. 6, Issue 11, November 2017

Packet Recovery in Wi-Fi(802.11) Networks

R.S.Anande[1]

Assistant Professor, Dept. of Electronics, PVPIT, Budhgaon, Maharashtra, India[1]

ABSTRACT:Users increasingly demanding WLAN for entertainment and business. They experience dead spots and high loss rates more frequently. Bit errors occur in Wi-Fi networks when interference or noise overcomes the coded and modulated transmission Current wireless network attempts to recover losses by using different techniques. This study introduces a new method for recovering partial packets for 802.11 by using Block-based recovery approach instead of retransmitting corrupted packets.In proposed system the receiver computes checksums over blocks in corrupt packets and bundles these checksums into a negative acknowledgment sent when the sender expects to receive an acknowledgment. The sender then retransmits only those blocks for which the checksum is incorrect, and repeats this partial retransmission until it receives an acknowledgment.

KEYWORDS: Block based packet recovery, Negative acknowledgment (NACK), partial retransmission.

I.INTRODUCTION

Users increasingly depend on Wi-Finetworks for multiple applications. However, they occasionally experience dead spots and high loss rates. Wireless networks can suffer from high packet loss.Packet loss occurs when one or more packets of data travelling across a computer network fail to reach their destination. Packet loss can be caused by a number of factors including signal degradation over the network medium due to multi-path fading, scattering, packet drop because of channel congestion, corrupted packets rejected in-transit, faulty networking hardware, faulty network drivers. When caused by network problems, lost or dropped packets can result in highly noticeable performance issues or jitter with streaming technologies, voice over IP, online gaming and video conferencing, and will affect all other network applications to a degree.

Partial packet recovery approaches attempt to repair corrupt packets instead of retransmitting them. Packet recovery relies on the observation that packets with errors may have only a few, localized errors, or at least some salvageable, correct content. Various approaches have been proposed: some rely on physical layer information to identify likely corrupt symbols to be retransmitted [4], while others embed block checksums into oversized frames to allow the receiver to recognize partially correct transmissions [5]. Some avoid explicit knowledge and adaptively transmit forward error correction information that is likely to be sufficient to repair bit errors [3].

Here we present a block-based partial packet recovery approach to recover the lost packets. We use block-based recovery, meaning that we identify incorrect blocks of consecutive bytes for retransmission, as opposed to aggregating by symbol or estimating bit error rate. We transmit independent repair packets that contain only the blocks being retransmitted, in contrast to other approaches that may bundle repair information with subsequent transmissions to save on medium acquisition time. Repair packets, by being shorter, are more likely to arrive successfully than full size retransmissions and take less time to transmit, improving performance over 802.11. Using immediate repair packets also limits the amount of buffering required at the receiver side [1]. We use the crc-32 checksum to isolate errors to individual blocks; this checksum is sufficient to find all single bit errors, burst errors in a single 16-bit block, and two-bit errors separated by at most 16 bits [10]. Finally, we exploit the deference stations give to acknowledgments of overheard packets: because stations sending acknowledgments have priority over the medium right after a transmission, there is time for a receiver to grab the medium and send prompt feedback about received blocks.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijareeie.com

Vol. 6, Issue 11, November 2017

II.LITERATURE SURVEY

Partial packet recovery protocols attempt to repair corrupted packets instead of retransmitting them in their entirety. In this chapter, we discuss some of the commonly used packet recovery techniques. Also, we discuss the work done in past years for packet recovery schemes. In this section, we classify various wireless error recovery protocols.

In Block Checksum method when transmissions fail, receiver start recovery by sending feedback about corrupted blocks based on the per-block checksums transmitted with original data packets. Acknowledgment frames can be extended to include feedback to help error recovery protocols. Seda [12] is a recovery mechanism designed for data streaming in wireless sensor networks. In Seda, a sender divides each packet into blocks and encodes each block with a one byte sequence number and a (one-byte) CRC-8 for error detection. A receiver, after receiving several packets, will test the block-level CRC-8's for packets that fail the CRC-32 and request retransmission of those blocks.FRJ [5] uses jumbo frames to increase wireless link capacity. Each jumbo frame comprises 30 segments and each segment has its own CRC checksum. The receivers can check these segment checksums to perform partial retransmissions when the segments are corrupted. FRJ uses both MAC-layer ACKs and its own ACKs. FRJ sends its own ACKs after 100 ms or 64 received frames.

Forward error correction is veryadvantageous to error recovery because they do not require explicit information about error locations. ZipTx [3] uses a two-round forward error correction mechanism to repair corrupted packets. In the first phase, the transmitter sends a small number of Reed-Solomon bits for a corrupted packet, depending on the feedback received by the receiver. If the receiver still cannot recover the corrupted packets using these parity bits, the transmitter sends more parity bits in the second round. If both rounds fail, the receiver requests a retransmission of the whole packet. To reduce the number of feedback frames, ZipTx receivers accumulate feedback information to be transmitted after receiving eight packets or after a timeout. Although ZipTx [3] increases throughput, it may also increase recovery latency. This is because it disables MAC layer retransmission and generates its own ACKs for a group of packets in the driver. As a result, the delay for the recovered packets may be significantly higher than that of the retransmitted native 802.11 packets.

Wireless transmission systems can employ multiple spatial diverse receive antennas. Since the signals are propagated through slightly different paths, the likelihood is that signal will not suffer the same level of attenuation or multi-path effect. Signals from these diverse radio antennas can be combined to improve signal integrity. There are two ways to realize spatial diversity. The first one is known as MIMO [15], which requires both the sender and receiver have multiple antennas. The second one is known as cooperative communication or virtual MIMO. The requirement for cooperative communication is somewhat released that the sender can choose one or more relays to help it transmit the data. Most of these works is purely theoretical with no system implementation or experimental results. And the main difficulty is that those proposals are based on tight synchronization between terminals. So practically, systems exploiting spatial diversity are MIMO, requiring multiple antennas, which is infeasible in some applications such as wireless mobile networks.

Error recovery protocols can benefit from physical layer information beyond the best guess at the received symbol, although most commercial 802.11 cards do not expose such extra information. PPR [4] requests retransmissions of only those symbols that are likely corrupted. Driven by per-bit confidence from the PHY Layer, SOFT [2] combines several received versions of a corrupted frame to produce a correct frame. To repair packets sent to an AP, several APs share bit confidence over a wired link. To repair packets sent to a client, the client combines per-bit confidence from a corrupted transmission and one or more retransmissions.

III.PROPOSED METHODOLOGY AND DISCUSSION

In this section, we present an overview of the proposed scheme, describe how it achieves the key design goals of a practical partial packet recovery scheme, and justify the choices of block-based recovery and the crc-32 checksum computation.

Figure 1 presents an overview of the proposed protocol, compared to 802.11. When a receiver device receives a frame with errors, it divides the frame into 64-byte blocks (the last block may be smaller) and computes a separate checksum for each block. Then it replies to the transmitter with a NACK that includes these checksums. It saves the corrupted original packet in a buffer, waiting for the sender to transmit correct blocks. This negative acknowledgment is sent when the transmitter expects to receive a positive acknowledgment.

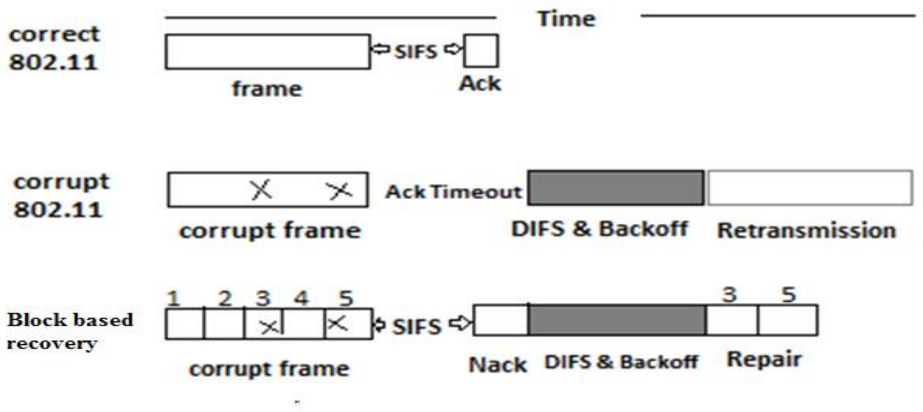


Figure 1: Retransmission policy of 802.11 and proposed scheme

Transmitter will then match the receiver-supplied checksums to those of the original transmission and send a repair packet with only those blocks of the original transmission that was corrupted [1]. Once the repair packet is received correctly, the receiver sends a normal 802.11 ACK. Unmodified senders will treat the negative acknowledgment as garbage and retransmit as normal. Unmodified receivers will fail to transmit a NACK, and cause a sender to retransmit after timeout [1]. At very low transmission rate, the NACK for a large packet may be longer than other stations expect to defer to the acknowledgment if it does; we rely on carrier sense to inhibit collisions with the end of the NACK.

The cases when packets of proposed systems are lost are straightforward. If a NACK is lost, the transmitter will retransmit the packet as in 802.11. If this retransmission has errors, the receiver will send another NACK. If a repair packet is lost or received with errors, the receiver will transmit nothing. One could alter the protocol to send an abridged NACK to recover correct blocks from errored repair packets, but we expect minimal gain from the added complexity.

As shown in Figure1 SIFS (Short Interframe Space), is the small time interval between the data frame and its acknowledgment. SIFS are found in IEEE 802.11 networks. They are used for the highest priority transmissions enabling stations with this type of information to access the radio link first. This value is fixed per PHY and is calculated in such a way that transmitting station will be able to switch back to receive mode and be capable of decoding the incoming packets. Examples of information which will be transmitted after the SIFS has expired include the acknowledgement (ACK) and the Clear to Send (CTS) messages. The SIFS in IEEE 802.11 are defined to be the smallest of all Interframe spaces (IFS). SIFS duration is a constant value and it depends on the amendments. For IEEE 802.11b/g SIFS is 10µs and for IEEE 802.11a SIFS is 16 µs [8].

To generate NACK, the receiver computes block checksums for corrupted frames. Due to hardware limitations, the block size should be a multiple of 32 bytes. We use 64 bytes as the block size. Larger blocks increase computation efficiency and reduce NACKs, while smaller blocks are parsimonious with repair bytes [1].

For some transmission rates, a NACK uses more airtime than a MAC ACK, which may cause problems in the presence of hidden terminals. The size of an ACK frame is only 14 bytes. A NACK frame, based on 64-byte blocks, is at most 96 bytes longer than an ACK frame. A hidden terminal of the receiver may hear from the transmitter the network

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijareeie.com

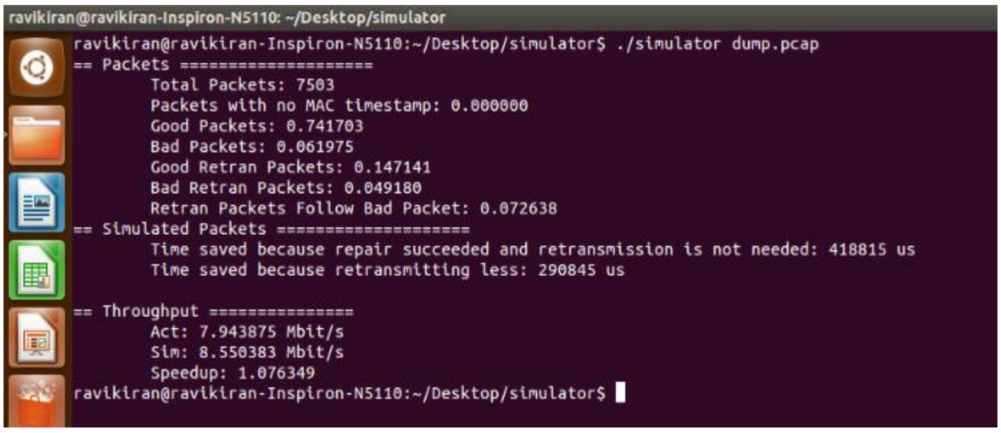
Vol. 6, Issue 11, November 2017

allocation vector (NAV) and the earliest time it can start its own transmission is DIFS, 50 µs for 802.11b/g, after the end of NAV. There will be no collision when NACK's bit rate is higher than 12 Mbps. Otherwise, a transmission from the hidden terminal may collide with our NACK frames, which causes the retransmission of the whole packet. Preliminary experiments on a hidden terminal topology indicate that even in this scenario, enabling proposed scheme can increase overall throughput.

After a transmitter gets a NACK, it compares the received block checksums with the computed checksums and decides which block to retransmit inside a repair packet. We always retransmit the first block of a packet, which contains the important headers of various layers.

IV. RESULT AND DISCUSSION

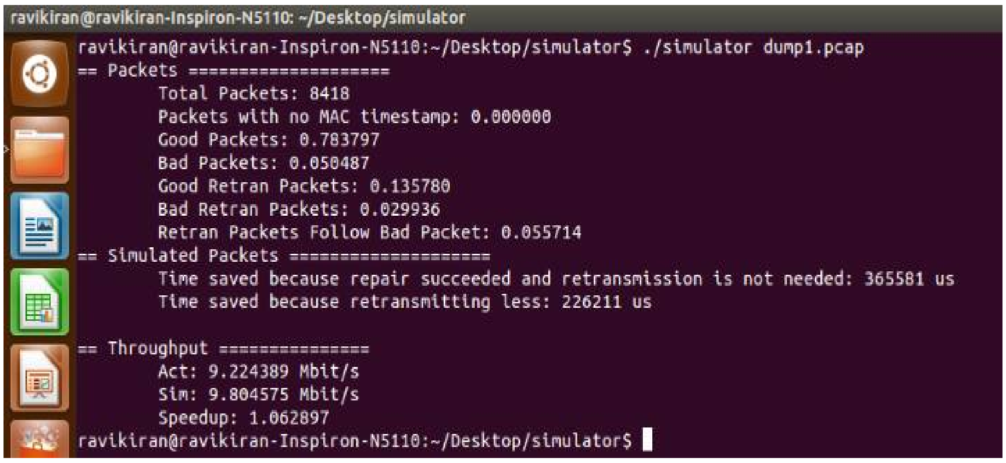
We have used two platforms namely GCC Compiler [7] and WireShark [6].In our scheme we have captured the packets in three different environments using Wireshark tool. The files which we have captured from the Wireshark platform are simulated and their results are as follows.



```
ravikiran@ravikiran-Inspiron-N5110: ~/Desktop/simulator
ravikiran@ravikiran-Inspiron-N5110:~/Desktop/simulator$ ./simulator dump.pcap
== Packets ====================
        Total Packets: 7503
        Packets with no MAC timestamp: 0.000000
        Good Packets: 0.741703
        Bad Packets: 0.061975
        Good Retran Packets: 0.147141
        Bad Retran Packets: 0.049180
        Retran Packets Follow Bad Packet: 0.072638
== Simulated Packets ====================
        Time saved because repair succeeded and retransmission is not needed: 418815 us
        Time saved because retransmitting less: 290845 us

== Throughput ==============
        Act: 7.943875 Mbit/s
        Sim: 8.550383 Mbit/s
        Speedup: 1.076349
ravikiran@ravikiran-Inspiron-N5110:~/Desktop/simulator$
```

Fig. 2Results obtained after simulating first packet capture file



```
ravikiran@ravikiran-Inspiron-N5110: ~/Desktop/simulator
ravikiran@ravikiran-Inspiron-N5110:~/Desktop/simulator$ ./simulator dump1.pcap
== Packets ====================
        Total Packets: 8418
        Packets with no MAC timestamp: 0.000000
        Good Packets: 0.783797
        Bad Packets: 0.050487
        Good Retran Packets: 0.135780
        Bad Retran Packets: 0.029936
        Retran Packets Follow Bad Packet: 0.055714
== Simulated Packets ====================
        Time saved because repair succeeded and retransmission is not needed: 365581 us
        Time saved because retransmitting less: 226211 us

== Throughput ==============
        Act: 9.224389 Mbit/s
        Sim: 9.804575 Mbit/s
        Speedup: 1.062897
ravikiran@ravikiran-Inspiron-N5110:~/Desktop/simulator$
```

Fig. 3Results obtained after simulating second packet capture file

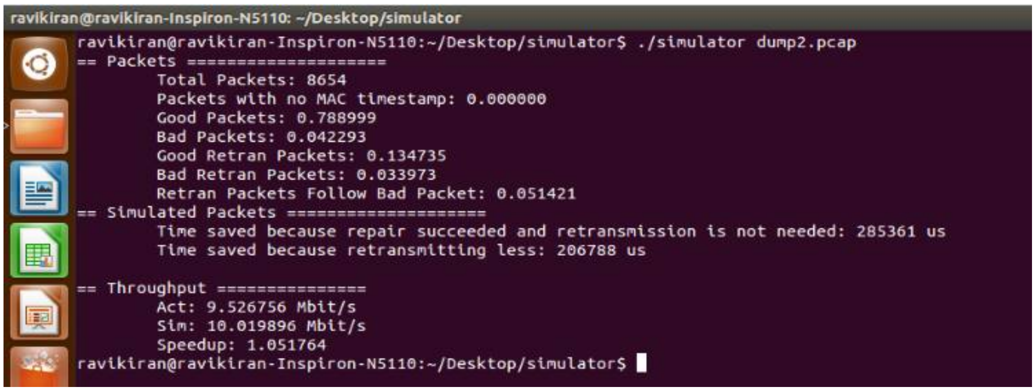
ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijareeie.com

Vol. 6, Issue 11, November 2017



```
ravikiran@ravikiran-Inspiron-N5110: ~/Desktop/simulator
ravikiran@ravikiran-Inspiron-N5110:~/Desktop/simulator$ ./simulator dump2.pcap
== Packets ====================
        Total Packets: 8654
        Packets with no MAC timestamp: 0.000000
        Good Packets: 0.788999
        Bad Packets: 0.042293
        Good Retran Packets: 0.134735
        Bad Retran Packets: 0.033973
        Retran Packets Follow Bad Packet: 0.051421
== Simulated Packets ====================
        Time saved because repair succeeded and retransmission is not needed: 285361 us
        Time saved because retransmitting less: 206788 us

== Throughput ==============
        Act: 9.526756 Mbit/s
        Sim: 10.019896 Mbit/s
        Speedup: 1.051764
ravikiran@ravikiran-Inspiron-N5110:~/Desktop/simulator$
```

Fig .4Results obtained after simulating third packet capture file

The results obtained after simulating the captured files are compared in terms of throughput as shown below:

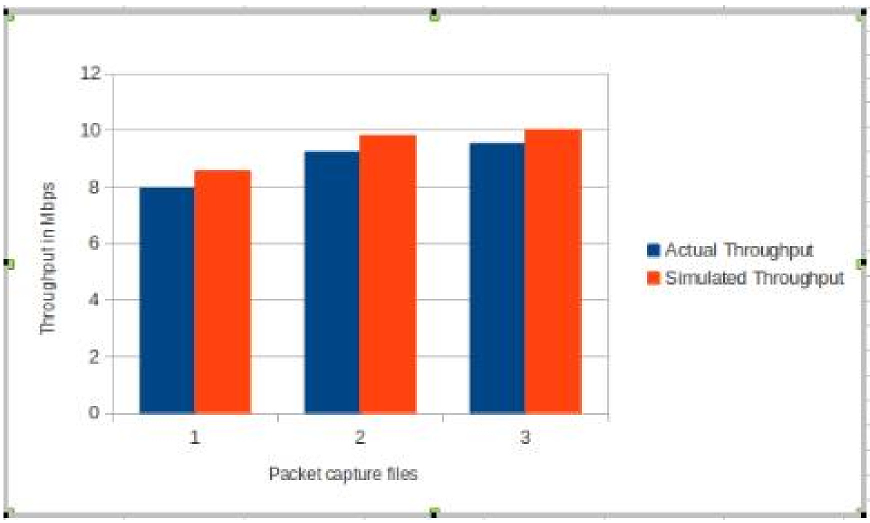


Fig .5 : Comparison between Wi-Fi and proposed scheme in terms of throughput

V .CONCLUSION

In this thesis, we have studied how to recover the lost packets using the block based packet recovery scheme. This scheme divides the errored packet into number of blocks and sends only that block which has errors. We design, simulate, and evaluate a block based practical partial packet recovery protocol for 802.11 wireless networks. This scheme is advantageous as:

- It reduces recovery latency,
- It increase throughput

As compared to other available techniques.

The simulation results show that this scheme increase throughput of system compared to existing techniques which reflect superiority of our scheme over others.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijareeie.com

Vol. 6, Issue 11, November 2017

REFERENCES

[1] Bo Han, Aaron Schulman et al. Maranello: practical partial packet recovery for 802.11 In NSDI 2010.
[2] G. Woo, P. Kheradpour, D. Shen, and D. Katabi. Beyond the bits: Cooperative packet recovery using physical layer information. In MOBICOM, 2007.
[3] K. C.-J. Lin, N. Kushman, and D. Katabi. ZipTx: Harnessing partial packets in 802.11 networks. In MOBICOM, 2008.
[4] K. Jamieson and H. Balakrishnan. PPR: Partial packet recovery for wireless networks. In SIGCOMM, 2007.
[5] A.P. Iyer, et al. Fast resilient jumbo frames in wireless LANs. In IWQoS, 2009.
[6] Wireshark User’s Guide: for Wireshark 1.11 by Ulf Lamping, Richard Sharpe and Ed Warnicke.
[7] Using the GNU compiler collection by Richard M. Stallman and GCC developer community
[8] IEEE Std. 802.11 – 2007 by IEEE computer Society.
[9] Peterson, W. W. and Brown, D.T. “Cyclic Codes for Error Detection.” In Proceedings of the IRE, January 1961, 228–235.
[10] J. Stone, M. Greenwald, C. Partridge, and J. Hughes. Performance of checksums and CRC’s over real data. IEEE/ACM Transactions on Networking, 6(5):529–543, 1998.
[11] A. Miu, H. Balakrishnan, and C. E. Koksal. Improving loss resilience with multi-radio diversity in wireless networks. In MOBICOM, 2005.
[12] R. K. Ganti, P. Jayachandran, H. Luo, and T. F. Abdelzaher. Datalink streaming in wireless sensor networks. In SenSys, 2006.
[13] Linux kernel mac80211 framework for wireless device drivers. http://linuxwireless.org/en/developers/ Documentation/mac80211/.
[14] Iperf. http://sourceforge.net/projects/iperf/
[15] D. Tse and P. Vishwanath, “Fundamentals of Wireless Communications,” Cambridge University Press, 2005.