Forward Error Control in Wireless Local Area Networks

Ivy Chow ischow@sfu.ca
Horace Yat-Hong Chan horace_chan@pmc-sierra.com
Sameer Khushal skhushal@sfu.ca

1. Abstract

In this project, we investigate the performance improvement of applying the Forward Error Control (FEC) to the IEEE 802.11b Wireless LAN standard. In the IEEE 802.11b standard, bit error introduced by the channel noise is protected by Cyclic Redundancy Check (CRC) and results in packet loss. Reliable end-to-end connection is provided by higher layer protocols such as TCP, whose performance is severely degraded by the bit error rate in the channel. We evaluate the performance of two different approaches of FEC implementation, Reed Solomon code, and Turbo code over TCP Wireless LAN. From the OPNET simulations, we concluded that both implementations reduce the packet loss and the congestion in the network.

Table of Contents

1. Abstract ii

2. Introduction 1

3. Overview of 802.11b Wireless Local Area Networks 2

4. Introduction to Forward Error Correction 4

4.1. Reed-Solomon Code 4

4.2. Turbo Codes 5

5. Implementation Details 6

5.1. Reed Solomon Code 7

5.2. Turbo Code 7

6. Simulation Scenarios 9

6.1. Simple Topology 9

6.2. Complex Topology 11

7. Results Discussion 12

7.1. Simple Topology 12

7.2. Complex Topology 14

8. Conclusions 17

9. Reference 18

10. Appendix A – Code Listing 19

Table of Figures

Figure 1 802.11 PHY Frame 2

Figure 2 Example Wireless Network 3

Figure 3 Sender Based Repair Techniques 4

Figure 4 Turbo Encoding 5

Figure 5 Radio Link Transceiver Pipeline 6

Figure 6 Wireless LAN Workstation Node Model 8

Figure 7 Simple Topology Implementation 9

Figure 8 Complex Topology Implementation 11

Figure 9 Simple Topology: TCP Delay 12

Figure 10 Simple Topology: TCP Window Size 13

Figure 11 Complex Topology: TCP Delay 14

Figure 12 Complex Topology: TCP Delay 15

Figure 13 Complex Topology: Window Size 16

2. Introduction

The popularity and growth of the Internet makes TCP/IP the de-facto network layer protocol in data transmission between computers. Traditionally, within a local area network area, TCP/IP is carried on wired Ethernet connected to the computers. Since TCP is originally designed for low noise channel, the recent development of wireless LAN created many challenges in TCP performance. This project is to investigate the application of Forward Error Control (FEC) to TCP transmitted over the IEEE 802.11b Wireless LAN standard.

A brief introduction of the IEEE 802.11b Wireless LAN standard is discussed in section 2. In section 3, there is the principle of Forward Error Control and introduction of the two different FEC approaches, Reed-Solomon Code and Turbo Code. Section 4 describes the simulation environment with two network scenarios A simple one with just one transmitter and one receiver, and a more complex one involves simultaneous transmission between several mobile hosts. In the last section, we will discuses the simulation results and conclude our research with suggestions.

3. Overview of 802.11b Wireless Local Area Networks

Wireless Local Area Network (LAN) 802.11b is a standard out of the IEEE 802.11 committee. It is primarily used for low cost wireless LANs in enterprise networks. The cost is kept low by using a low power unlicensed spectrum band.

The data in 802.11b is encoded using DSSS (direct-sequence spread-spectrum) technology. The way that DSSS operates is that it first takes a data stream of zeroes and ones and modulates it with a second pattern, what is called the chipping sequence. In this case the chipping sequence is known as the Barker code. This is an 11 bit sequence (10110111000) that has been derived to have certain mathematical properties that make it easy for modulation radio waves. The method that the Barker code operates on the data is to exclusive OR with the data and this generates data object called chips. The format is each bit is encoded by the 11 bit Barker code so 11 chips is equivalent to one bit of data.

For 1Mbps transmission BPSK (Binary Phase Shift Keying) is used. For 2Mbps transmission QPSK (Quadrature Phase Shift Keying) is used. To maintain quality of transmission at the higher bitrate power must be increased or range must be decreased. So the radio adapts by using a slower encoding mechanism for larger ranges.

The wireless physical layer has tow parts the PLCP (Physical Layer Convergence Protocol) and the PMD (Physical Medium Dependent) sublayer. The PMD is responsible for the wireless encoding. and the PLCP enables an interface for higher-level drivers through CCA (Clear Channel Assessment) this is the signal that MAC (Media Access Control) to determine channel availability. The figure below shows the PHY Frame.

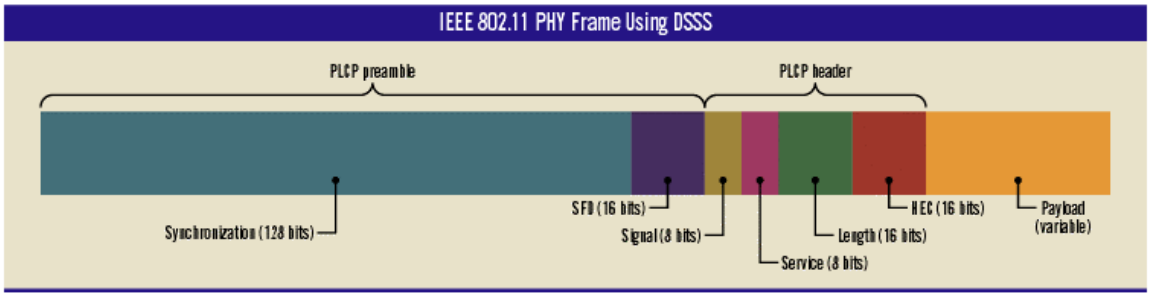


Figure 1 802.11 PHY Frame

192bits of the header is always transmitted at 1Mbps so 802.11b is at best 85 percent efficient at the physical layer.

The MAC layer must sense a quiet time on the network to transmit. Each station listens to the network and determines when to transmit by waiting for the random back-off timer to reach 0. When the node has transmitted the timer is reset. Other features that enable radio communication, which can be unpredictable, is the RTS/CTS (request to send/clear to send) feature. However this adds significant overhead to the network and must also be enabled on the client and access point. The MAC layer also identifies the source and destination addresses, data payload & CRC.

An example implementation of a wireless network is shown in Figure 2. Here the wireless network is connected, through an IP gateway and unsecure network to a remote wired network.

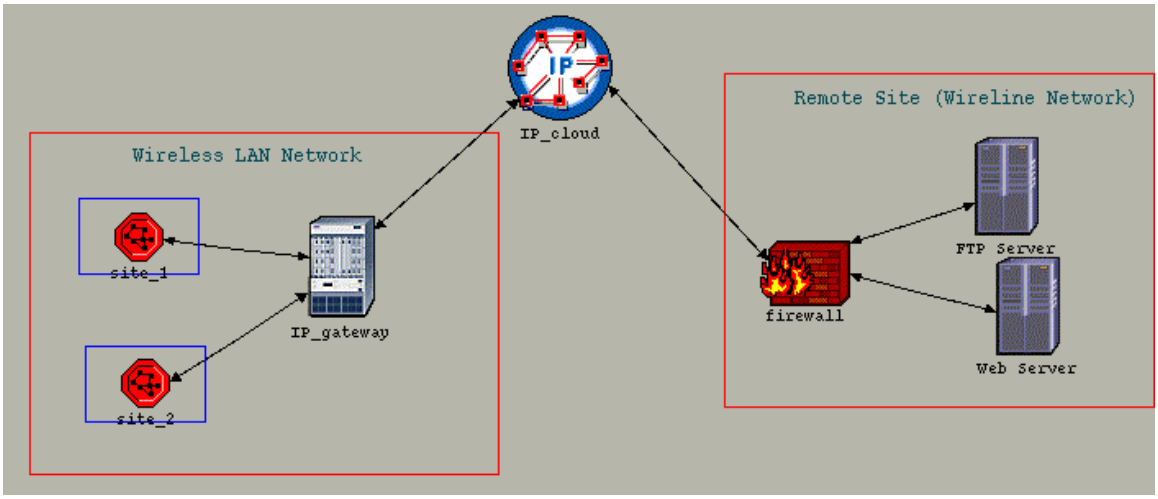


Figure 2 Example Wireless Network

4. Introduction to Forward Error Correction

The sender based repair techniques are shown in Figure 3. Out of these techniques we will be investigating forward error correction under the passive heading.

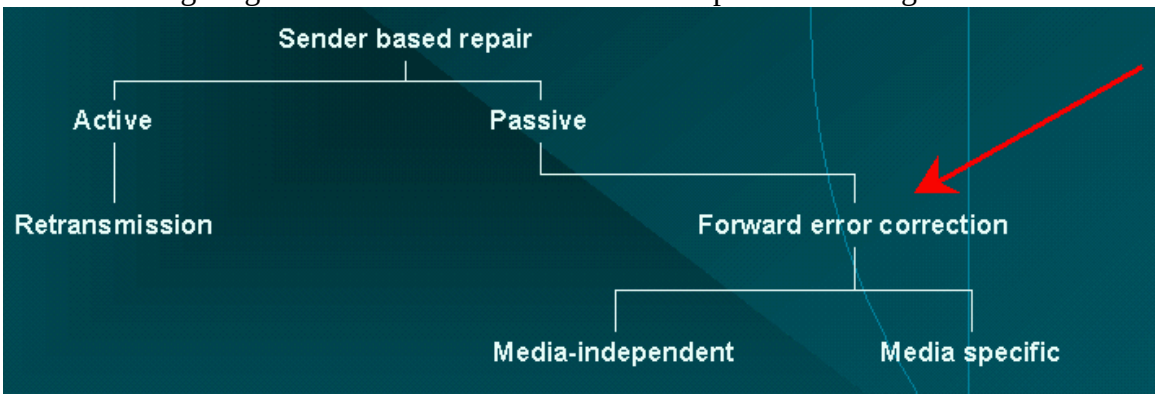


Figure 3 Sender Based Repair Techniques

There are two major kinds of Forward Error Correction (FEC), media independent and media specific FEC. In this project, we use the Reed Solomon Code to represent media independent FEC, and Turbo code to represent media specific FEC.

The major reasons that FEC is important for WLAN applications are 1) Packet loss in TCP causes window size decrease and retransmission of the lost packet. 2) The goal is to maximize the window size, for optimal utilization of bandwidth in channel. 3) In wired network, mostly traffic congestion lead to decrease in window size, and not bit error rate. 4) In a noisy wireless networks, window size is very small due to the high bit error rate of the channel 5) FEC reduces retransmission by transmitting error correction code with packets

4.1. Reed-Solomon Code

This kind of media independent FEC code uses block or algebraic codes that produces additional packets for transmission. This aids in the correction of packet losses. It is important to note that each code takes a codeword of k data packets and generates n-k additional packets, which is used in the transmission of n packets over the network

The origins of the Reed-Solomon (RS) code are in the detection and correction of errors in bit streams. Here it has been adapted to do the same for packet streams. The advantages that the RS codes have over other codes is that thy have excellent error correction properties and they are robust against burst losses. The coding procedure is based on an encoding scheme using polynomials and readily available algorithms make the computing cost small.

Using media independent codes such as the RS code results in recovery and repair with the exact replacement. Some disadvantages of this are a possible delay penalty, increased bandwidth and possible difficulties in implementation.

4.2. Turbo Codes

The media specific codes include a special type of code called the Turbo Code. This method of coding allows close to the Shannon capacity, the theoretical maximum, of the channel to be used.

This code is media specific because it addresses several non-ideal characteristics of the wireless communication network. These include multipath fading and propogation losses. The use of the turbo codes will result in acceptable bit error rates.

The turbo code principal is shown in Figure 4 consists of two Recursive Systematic Convolutional Codes (RSC) are and the interleaver on the left. If the interleaver length or the code length constraints are increased the turbo code performance can be improved.

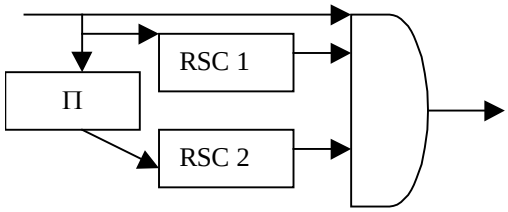


Figure 4 Turbo Encoding

However, one of the disadvantages of the turbo coding method is that the decoder is very complex to implement and an efficient memory management scheme is needed.

5. Implementation Details

To investigate the benefit of the Reed Solomon code and the Turbo code to the TCP performance of Wireless LAN, we modeled the behavior of these two FEC using OPNET and simulated the network with traffic traces. We modified the OPNET built-in WLAN model by adding the behavior of FEC with Reed Solomon Code and Turbo Code to the Radio Link Transceiver Pipeline. (Figure 5)

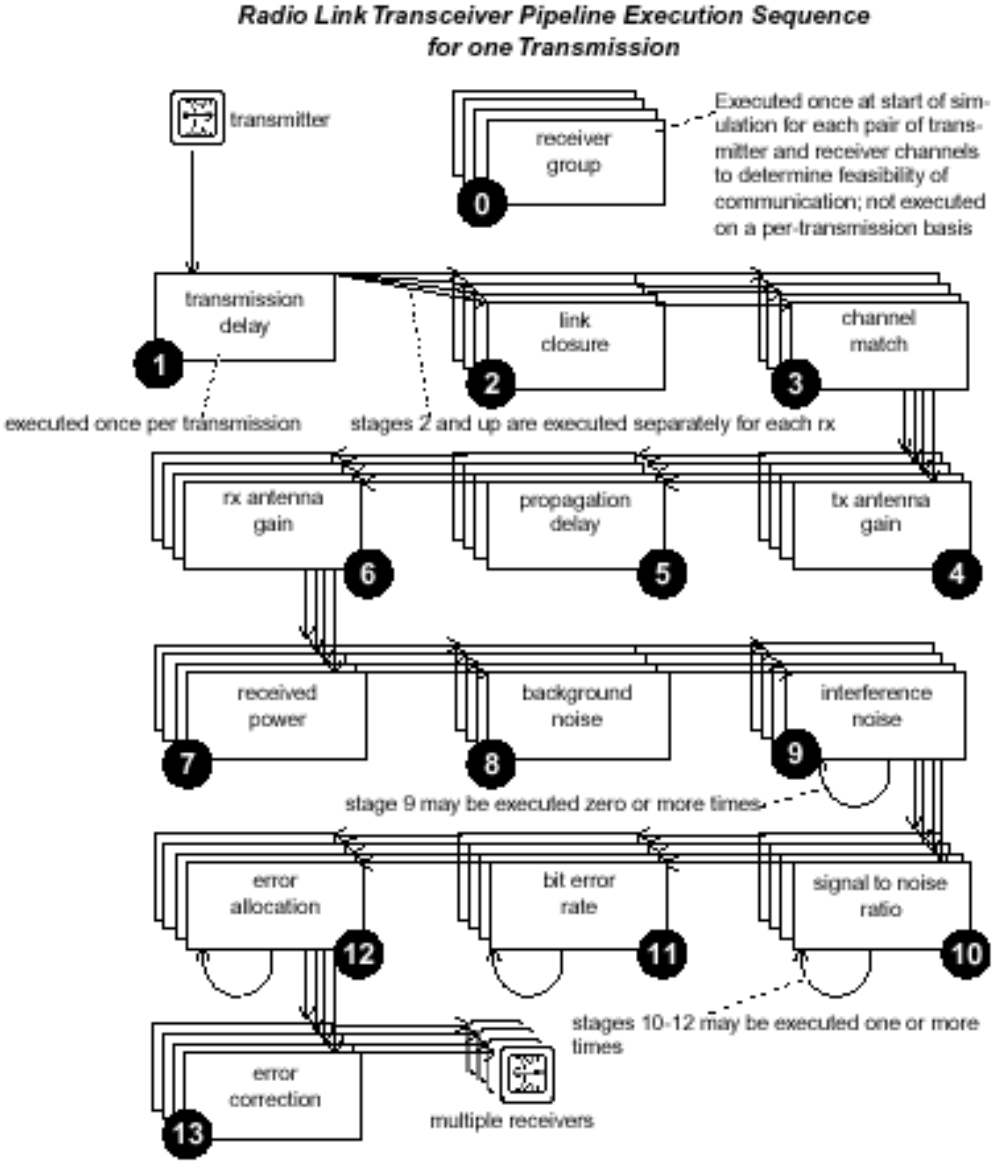


Figure 5 Radio Link Transceiver Pipeline

OPNET is a packet base simulator, the transmitter, receivers and radio channel properties is modeled with 13 Radio Link Transceiver Pipeline stages. Since our simulation is focused on the TCP performance rather than physical properties of the channel, we deactivated all physical layer related stage except stage 11, which introduce the bit error rate (BER). We replaced the original BER model, which calculate the BER from Antenna gain, background noise, Signal-to-Noise ratio etc from previous stages with our custom BER model, which read the BER from the node attribute.

5.1. Reed Solomon Code

To model the behavior of RS Code, we need two new attributes, the length of the code word (n), the length of the data in a code word (k). The number of bit error that can be corrected (t) can be calculated from n and k, where $2t = n - k$. In the RS code for every k data bits, the transmitter will attach n-k bits of parity bits for error correction. We model this behavior by change the first pipeline stage that calculate transmission delay. On each packet, we calculated the extra RS parity bits required and then added extra bits to the packet length, which used to calculate the transmission delay.

On the receiver side, stage 12, the error allocation stage determines the number of error bits in the packet base on the BER from stage 11. In stage 13, error correction, the original model decided whether to accept or reject the packet base on the ecc threshold attribute. Here we replace this stage with our custom model, which reject all packets with number of bit error larger than t.

Please note that the processing time of the RS code is neglected due to the fact that in real world, RS encoder and decoder can be implemented by hardware with only one clock cycle processing delay.

5.2. Turbo Code

To model the behavior of Turbo Code, we need two new attributes, the processing delay to encode and decode the turbo code, and the BER threshold that the code can correct. Turbo code is a math intensive calculation which most of the impact to the network is the delay it introduces. We neglect the overhead turbo code may add to the packet size due to the overhead is usually very small and the size varies and depend on the media content. The encoding delay of the Turbo code is modeled by adding a delay to the link between the wireless_lan_mac process and the transmitter process within the workstation node model. The decoding delay is modeled by adding a delay to the link between the receiver process and the wireless_lan_mac process. (Figure 6)

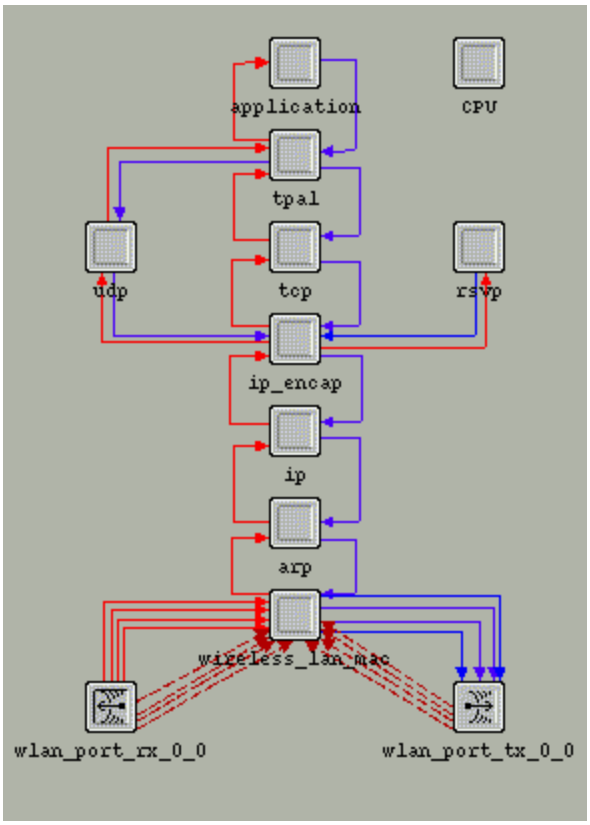


Figure 6 Wireless LAN Workstation Node Model

On the receiver side, the error allocation stage also calculates the actual BER base on the number of error bits in the packet. In the error correction stage (stage 13), when turbo code is used, the model accept all the packets with actual BER lower than the turbo code BER threshold, and reject otherwise.

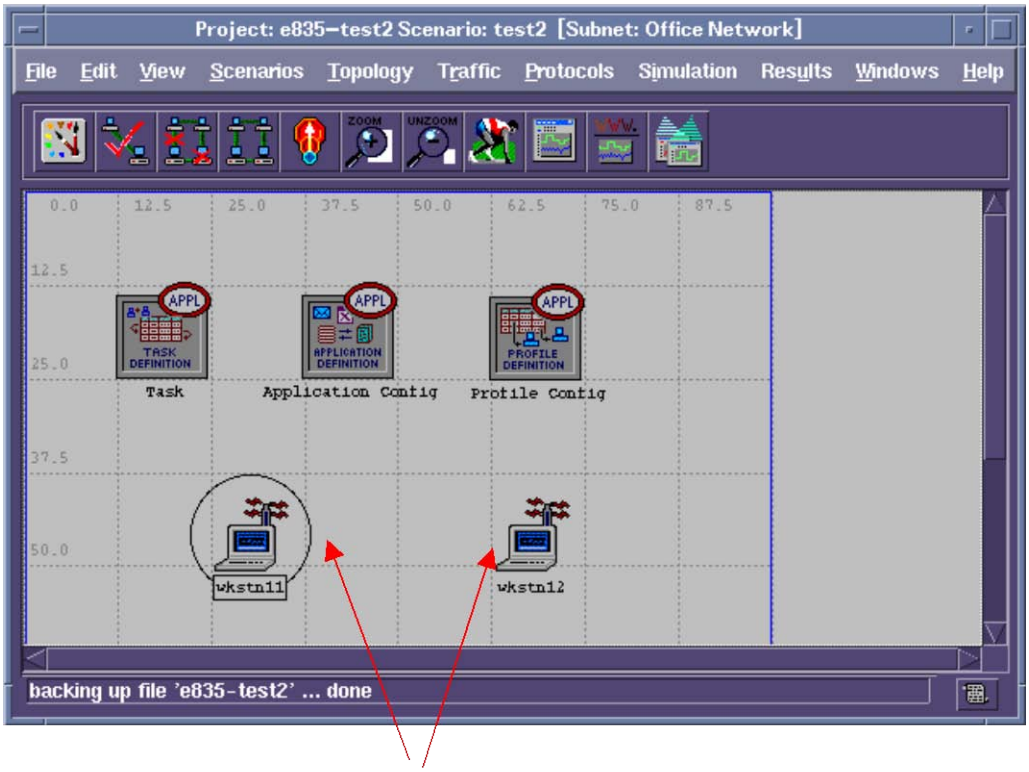
In our simulations, besides the wireless LAN workstation node model, we also used the wireless LAN server node model and the wireless LAN router node model. We applied the above modification to the other two models as well.

The source code of the custom pipeline stage models is located in the appendix A.

6. Simulation Scenarios

We use trace driven simulation in the Opnet to give more realistic simulation result. The trace we are using is the Starwars MPEG trace download from the ENSC835 course web page. The data of trace file is in number bits per 1/24 seconds, while the OPNET requires the number of bytes. We had to write a simple script to convert the bit information into byte information.

6.1. Simple Topology



802.11b Wireless
Workstations

Figure 7 Simple Topology Implementation

First we implemented a simple topology to demonstrate the proof of concept and as the development platform of our custom models. (Figure 7) The simple scenario has two wireless lan workstation, a trace traffic is sent from one workstation to another. Since this simulation is focused on TCP performance, we use the default value in most of the parameters. The following are the list of parameters that we had changed to yield the best simulation result.

TCP Protocol:
We choose TCP Reno with TCP segment size of 2272 bytes. Reno has the fast re-transmit feature that lacks in other TCP implement. The 2272 byte segment size is the default TCP over WLAN value in OPNET. In a noise channel, a smaller segment size reduces packet lost due to bit errors.

WLAN parameters:
We turned off the re-transmission in the WLAN layer. When a WLAN packet is corrupted by bit error, the OPNET WLAN model has there are built function to retransmit the corrupted packet. Without the physical layer retransmission, all retransmission has to be done by the TCP layer, this can further highlight the impact of FEC to TCP preformance.

The WLAN 802.11b standard is operated in DSS mode with 4 different data rate. We simply picked the highest data rate available 11Mbps due to the Starwars trace is a streaming video which require higher bandwidth.

Bit Error Rate:
We set the BER is 10e-6, which is the BER of a typical in door wireless environment such as an office building.

Reed Solomon Code:
We chose the commonly used RS(255,231) code. Where the length of codeword is 255 bits, in which contains 231 bits of data. This code can correct up to 12 bits of error.

Turbo Code:
We chose to model turbo code with 4 iterations, which on average can correct BER up to 10e-4. With assumption the calculation is carry out in generic 100MHz processor, and each iteration requires 1000 instruction, the processing delay of the turbo code is 0.4us. Please keep in mind that in real world BER threshold and the process delay may varies and depends on the processor speed and the media content.

6.2. Complex Topology

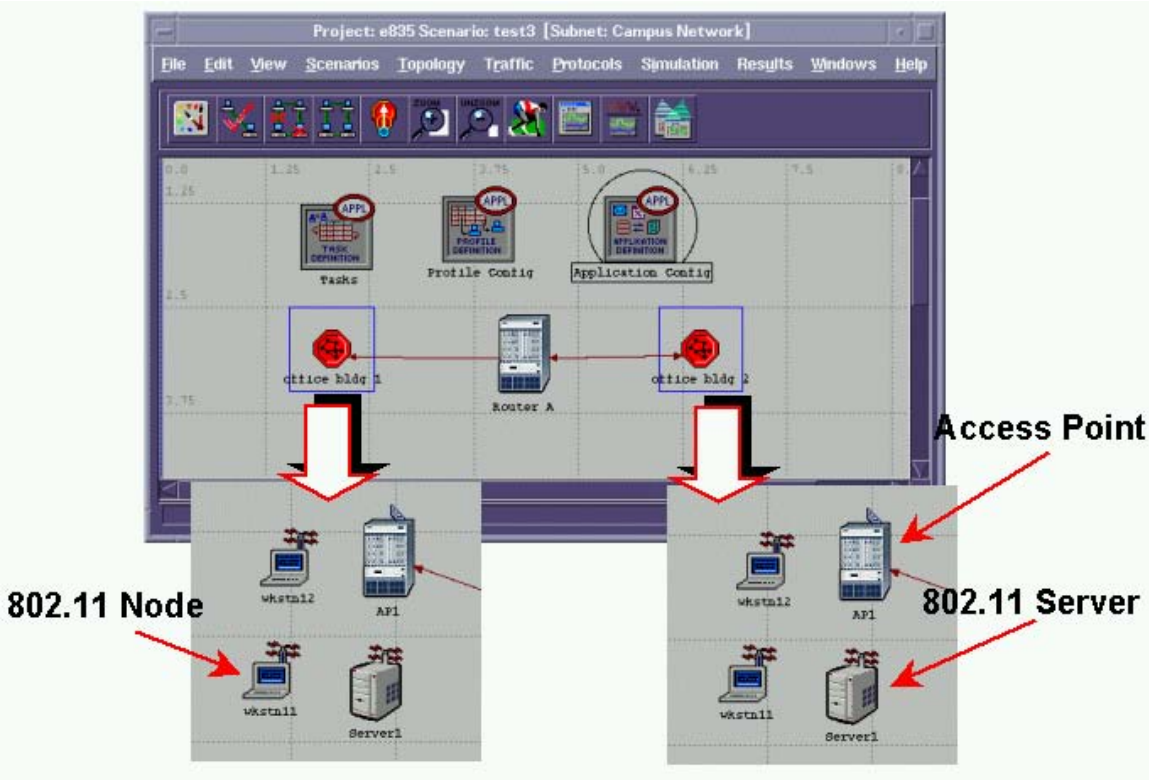


Figure 8 Complex Topology Implementation

With successful result from the simple topology, a complex topology was created with the same settings as the simple topology (Figure 8). Here the scenario is enhanced by the existence of a wired network, which interconnects the two wireless subnets. In each subnet, there are a WLAN access point, a wireless server and two workstations. Each server has two outgoing data stream, one goes to a workstation within the same subnet, while the other one destine a workstation on the other subnet. This complex interaction between the server and workstation create more a realistic simulation environment with the possible of channel conflict and congestion.

7. Results Discussion

7.1. Simple Topology

The results of simulation are shown in Figure 9. The results show that for RS code and Turbo coding the TCP delay is significantly reduced, which is the expected behavior of the algorithm implementation. It is because the uses of RS code and Turbo code reduce the number of packet retransmission in the WLAN layer and hence reduce the upper layer TCP delay. Also as shown in the figure, RS code has a shorter TCP delay compared to Turbo code because Turbo code requires extra processing delay. In our implementation, the extra processing delay added by the use of Turbo code is

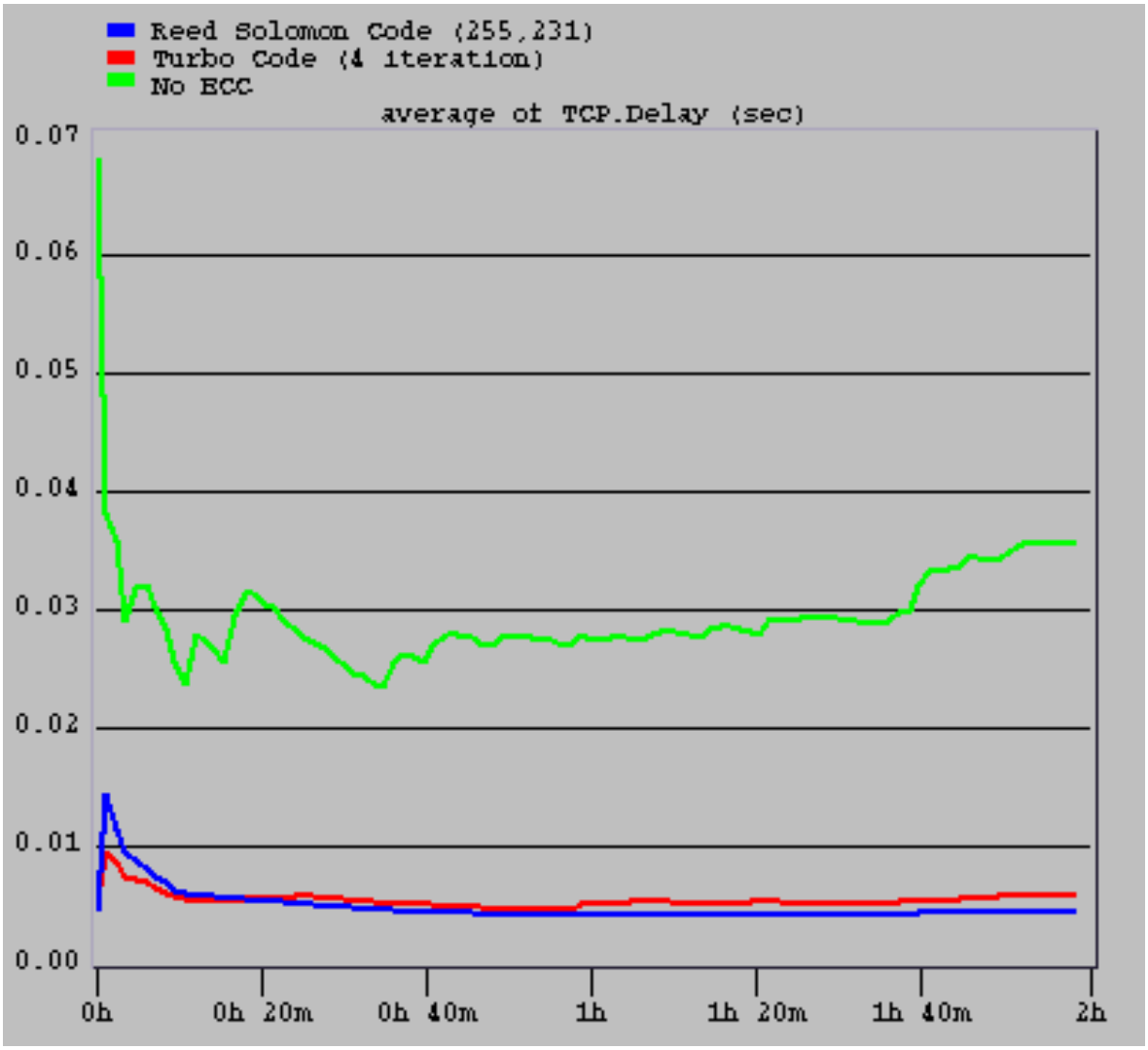


Figure 9 Simple Topology: TCP Delay

The TCP window size results are shown in Figure 10. This is the expected result because TCP Reno continues to increase its congestion window size by one during each round trip time until a packet is lost. Therefore, it results in a periodic oscillation of congestion window size in all of the three cases. The results indicate that the window size for the RS code and Turbo code are much larger than without FEC. Without FEC, errors in the WLAN layer cause frequent packet losses in the TCP layer, and TCP Reno reduces its window size to one half of the current window size when it experiences a packet loss. With FEC, errors are corrected in the WLAN layer, thus less packet losses cause TCP Reno to increase its congestion window size. Also because the rate at which each connection updates its window size depends on the round trip delay of the connection. Hence, the connections with shorter delays can update their window sizes faster than other connections with longer delays. Therefore, RS code, which has a shorter delay as shown in the figure, has larger congestion window size.

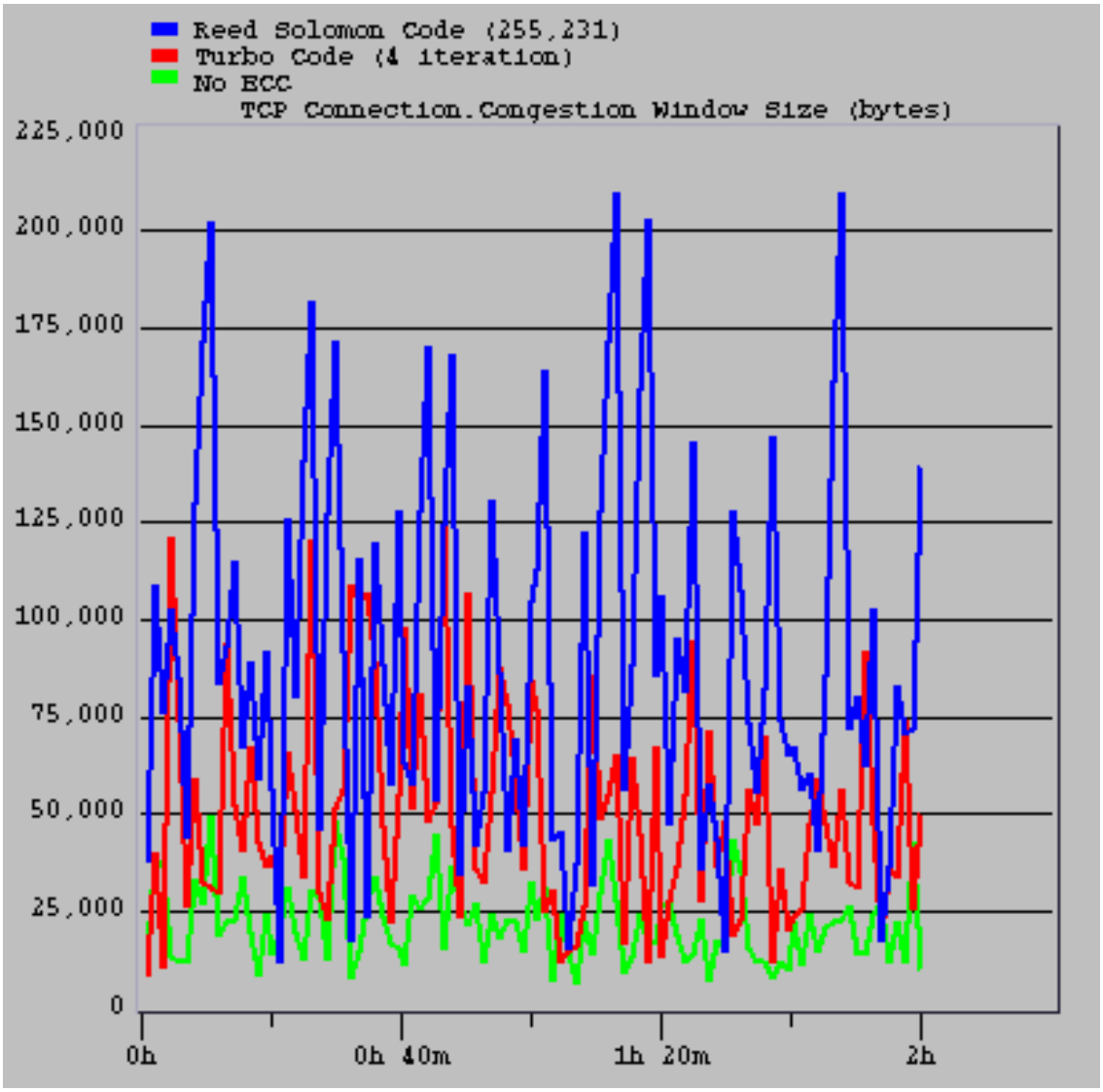


Figure 10 Simple Topology: TCP Window Size

7.2. Complex Topology

The results from the complex topology implementation show that the trends for the simple topology still hold. This is seen in Figure 11, Figure 12, and Figure 13. The conclusions reached for the simple topology case apply equally well for the complex topology case. In Figure 11, the results show that without FEC the data drop rate in the WLAN layer is the highest, and RS code has a higher data drop rate than Turbo code. It is because in this complex topology implementation, simultaneous transmission and reception cause higher chance of collision in the WLAN layer. As RS code requires extra overhead bits and has a larger packet size, the chance of collision increases, and consequently, more data is dropped.

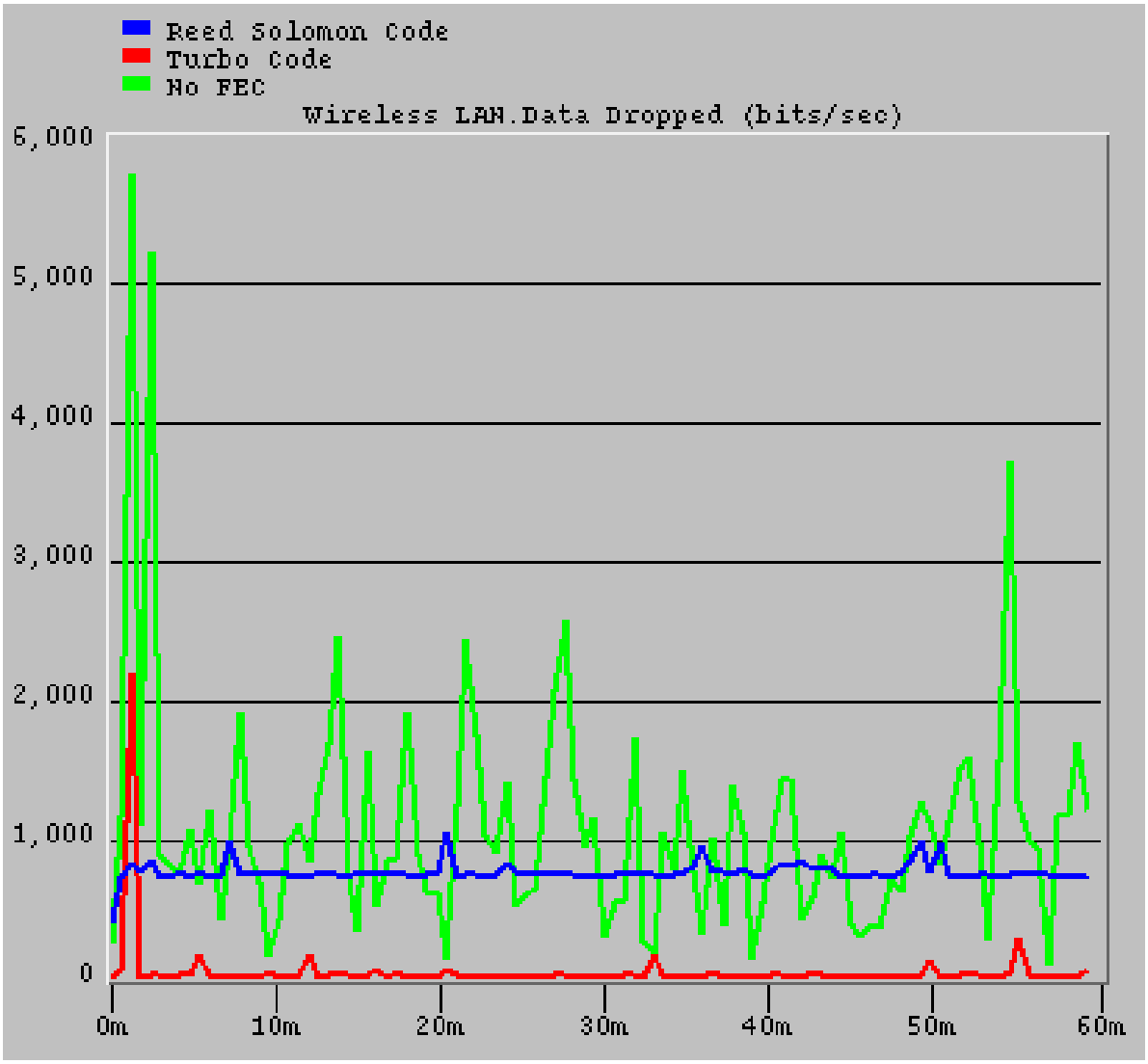


Figure 11 Complex Topology: TCP Delay

Since collisions in the WLAN layer cause packet retransmissions in the TCP layer, RS code, which has a higher data drop rate, has a longer TCP delay than Turbo code. The results are shown in Figure 12.

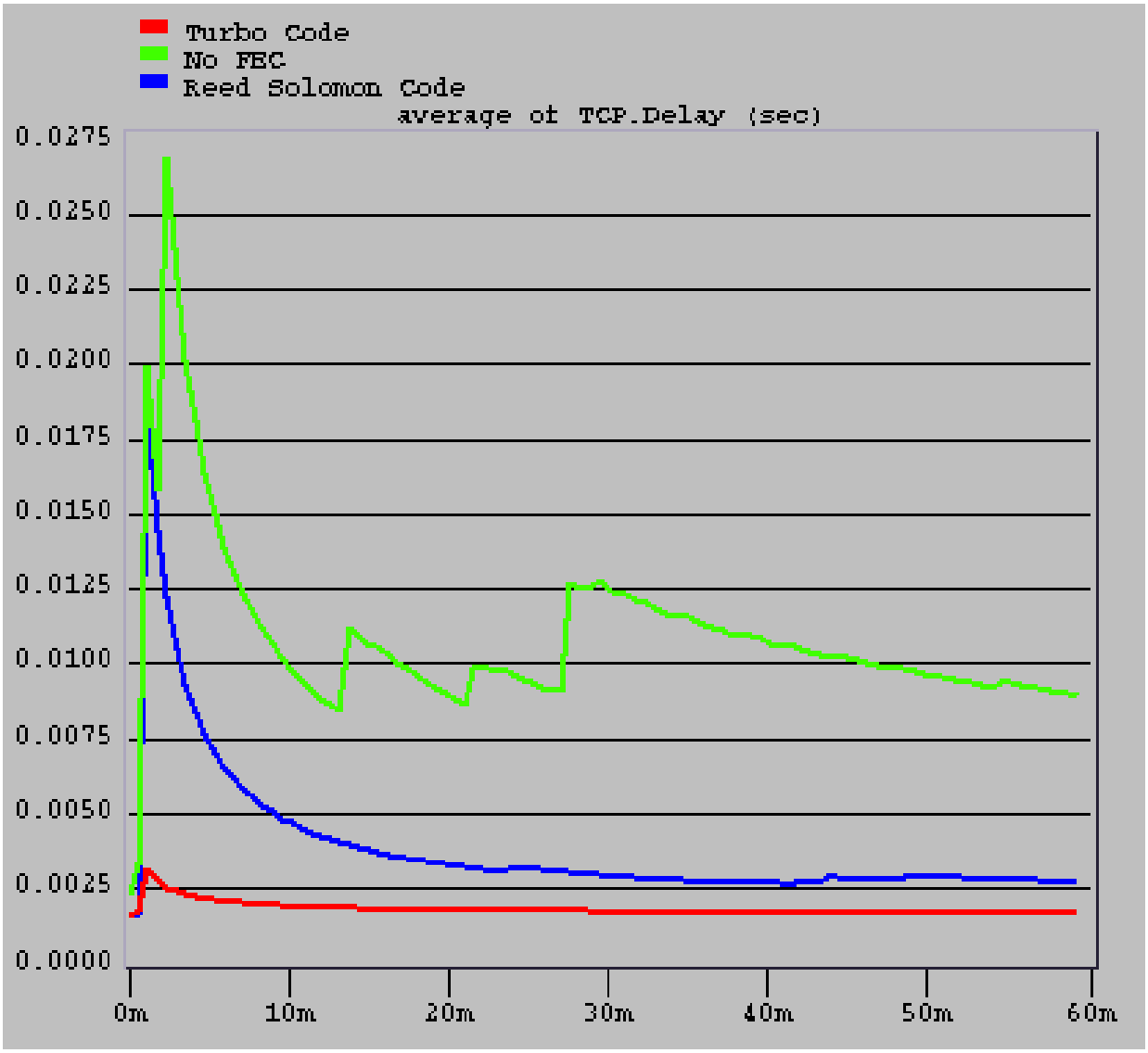


Figure 12 Complex Topology: TCP Delay

Similarly when compared to Turbo code, the more frequent collisions in the WLAN delay resulted by the additional overhead bits cause RS code to have more packet retransmissions, and consequently, smaller congestion window size in the TCP layer. The results are shown in Figure 13.

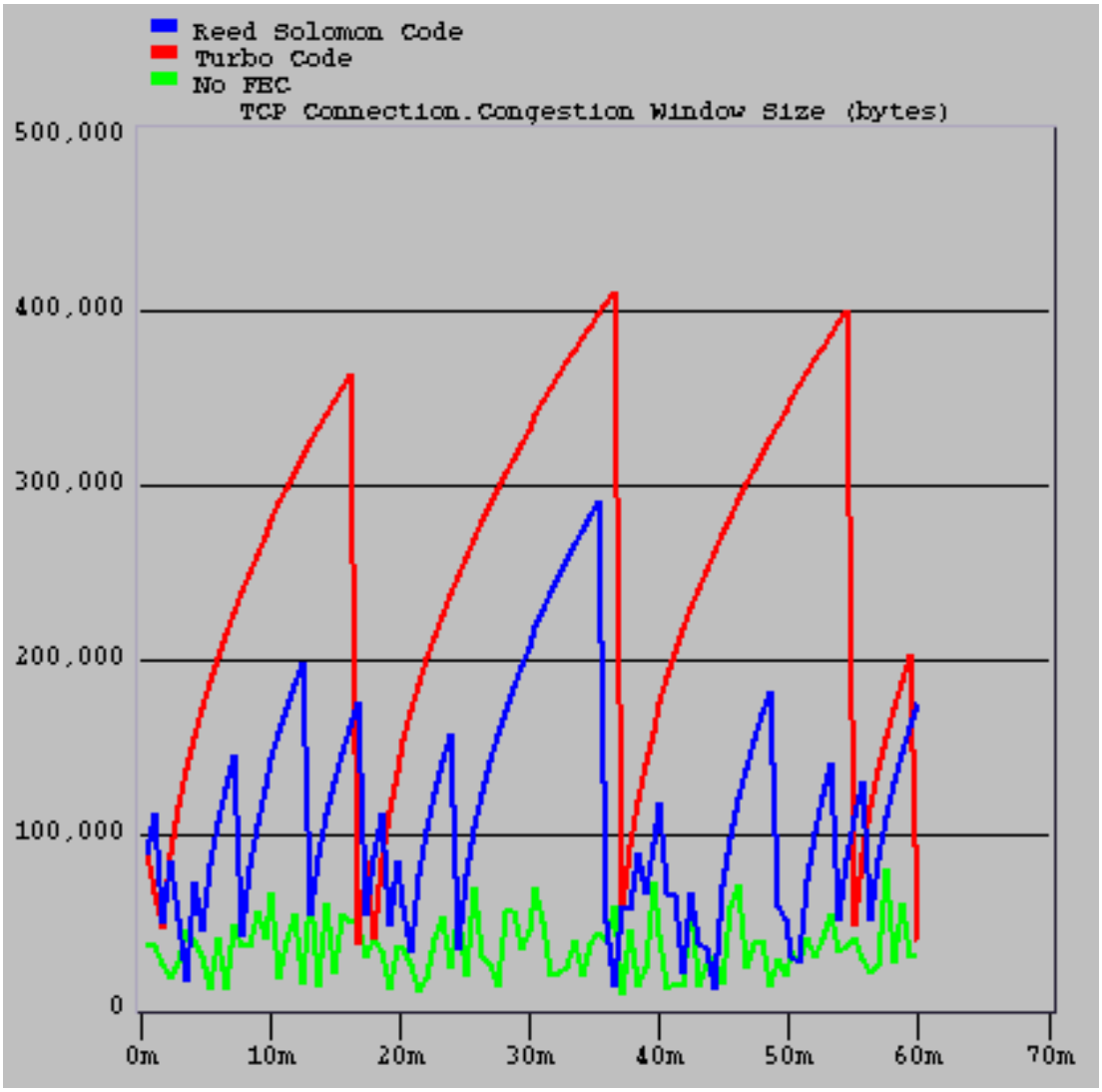


Figure 13 Complex Topology: Window Size

8. Conclusions

- In this project, we performed investigation on how FEC can be applied to IEEE 802.11b Wireless LAN standard to improve TCP performance.

- We implemented the behavior of two FEC codes: Reed Solomon Code & Turbo Code. For the RS Code, we calculated the extra parity bits required based on the attributes chosen by the user and then added the extra bits to the packet length on the transmitting side. On the receiving side, we only rejected packets with number of bit error larger than the number of bit error that can be corrected. For the Turbo Code, we modeled the encoding/decoding delay by adding additional user-defined processing delay to the link between the WLAN MAC layer and the transmitter/receiver. Then we rejected packets with actual BER higher than the user-defined Turbo code BER threshold

- We evaluated the performance of Reed Solomon Code and Turbo Code on improving reliable end-to-end connection provided by the TCP layer. Our simulation results show that both the FEC codes reduce packet retransmissions in the TCP delay, and hence reduce the TCP end-to-end delay and increase the TCP congestion window size.

- We compared the overall advantages and disadvantages of Reed Solomon Code and Turbo Code. RS Code is easier to model and has determined processing time; Turbo Code is more difficult to model accurately, but it is possible to increase the channel capacity to the Shannon limit. RS Code has smaller TCP delay and larger TCP congestion window size compared to Turbo code when the network topology is simple and the chance of data collision in the WLAN layer is low. However, Turbo Code has a better performance when the network topology is complex and the chance of data collision in the WLAN layer is high.

- For further enhancement, we recommend to model the Turbo Code encoding/decoding delay more accurately, and to investigate other FEC codes, such as Low Density Parity Check (LDPC) code.

9. Reference

1. Chris Heegard et al, High-Performance Wireless Ethernet, IEEE Communications Magazine, pp. 64-73, November 2001.

2. Gilbert Held, The ABCs of IEEE 802.11, IT Professionals, pp. 49-52, November 2001

3. A. Burr, Turbo-codes: the ultimate error control codes?, Electronics & Communication Engineering Journal, pp.155-165, August 2001

4. Ian F. Akyidiz et al, An Adaptive FEC Scheme for Data Traffic in Wireless ATM Networks, IEEE/ACM Transcations on Networking, pp.419-425, August 2001

5. Brett Schein and Steven Bernstein, A Forward Error Control Scheme for GBS and BADD, MILCOM 97 Proceedings, pp.710-715, Volume 2, 1997

10. Appendix A – Code Listing

“e835_txdel_rs.ps.c”

```
#include "opnet.h"
#include "math.h"

#if defined (__cplusplus)
extern "C"
#endif
void
e835_txdel_rs (Packet * pkptr)
      {
      int               pklen, ecc_n, ecc_k;
      double            tx_drate, tx_delay;
      Objid       tx_objid, node_objid;

      /** Compute the transmission delay associated with the      **/
      /** transmission of a packet over a radio link.             **/
      FIN (dra_txdel (pkptr));

      /* Obtain the transmission rate of that channel. */
      tx_drate = op_td_get_dbl (pkptr, OPC_TDA_RA_TX_DRATE);

      /* Obtain the error correction parameter of the receiver. */
      tx_objid = op_td_get_int (pkptr, OPC_TDA_RA_TX_OBJID);
      node_objid = op_topo_parent(tx_objid);
      op_ima_obj_attr_get(node_objid, "ecc_n", &ecc_n);
      op_ima_obj_attr_get(node_objid, "ecc_k", &ecc_k);

      /* Obtain length of packet. */
      pklen = op_pk_total_size_get (pkptr);

      if (ecc_k > 0)
            {
            pklen += ceil((double)pklen / (double)ecc_k) * (ecc_n -
ecc_k);
            op_pk_total_size_set(pkptr, pklen);
            }

      /* Compute time required to complete transmission of packet. */
      tx_delay = pklen / tx_drate;

      /* Place transmission delay result in packet's */
      /* reserved transmission data attribute. */
      op_td_set_dbl (pkptr, OPC_TDA_RA_TX_DELAY, tx_delay);

      FOUT;
      }
```

“e835_ber.ps.c”

```
#include "opnet.h"

#if defined (__cplusplus)
extern "C"
#endif
void
e835_ber (Packet * pkptr)
      {
      double            ber;
      Objid       rx_objid, node_objid;

      /** Calculate the average bit error rate affecting given packet.
**/
      FIN (dra_ber (pkptr));

      /* Get BER from the Receiver model */
      rx_objid = op_td_get_int (pkptr, OPC_TDA_RA_RX_OBJID);
    node_objid = op_topo_parent(rx_objid);
      op_ima_obj_attr_get(node_objid, "ber", &ber);

      /* Place the BER in the packet's transmission data. */
      op_td_set_dbl (pkptr, OPC_TDA_RA_BER, ber);

      FOUT;
      }
```

“e835_ecc.ps.c”

```
#include <opnet.h>

#if defined (__cplusplus)
extern "C"
#endif
void
e835_ecc (Packet * pkptr)
    {
      int               pklen, num_errs, accept;
      Objid       rx_ch_obid, rx_objid, node_objid;
      int               ecc_n, ecc_k, ecc_t, threshold;
      double      pe, turbo_ber;

      /** Determine acceptability of given packet at receiver. **/
      FIN (wlan_ecc (pkptr));

      /* Do not accept packets that were received     */
      /* when the node was disabled.                        */
      if (op_td_is_set (pkptr, OPC_TDA_RA_ND_FAIL))
            accept = OPC_FALSE;
      else
            {

            /* Obtain length of packet. */
            pklen = op_pk_total_size_get (pkptr);

            /* Obtain number of errors in packet. */
            num_errs = op_td_get_int (pkptr, OPC_TDA_RA_NUM_ERRORS);

            /* Obtain the error correction threshold of the receiver.
*/
            rx_objid = op_td_get_int (pkptr, OPC_TDA_RA_RX_OBJID);
            node_objid = op_topo_parent(rx_objid);

            /* RS code */
            op_ima_obj_attr_get(node_objid, "ecc_n", &ecc_n);
            op_ima_obj_attr_get(node_objid, "ecc_k", &ecc_k);
            ecc_t = (ecc_n - ecc_k) / 2;

            if (ecc_n > 0)
                  threshold = ecc_t * ((double)pklen / (double)ecc_n);
            else
                  threshold = 0;

            /* Turbo code */
            pe = op_td_get_dbl (pkptr, OPC_TDA_RA_BER);
            op_ima_obj_attr_get(node_objid, "turbo ber", &turbo_ber);


            if (pe < turbo_ber)
                  threshold = num_errs;

            /* Test if bit errors exceed threshold. */
            if (pklen == 0)
                  accept = OPC_TRUE;
```

```
            else
                  accept = (num_errs <= threshold) ? OPC_TRUE :
OPC_FALSE;
            }

      /* Place flag indicating accept/reject in the data packet control
field. */
      op_pk_nfd_set (pkptr, "Accept", accept);

      /* Force the simulation kernel to always accpet the packet. The
            */
      /* actual discarding of the packet will take place at the MAC
layer of the  */
      /* receiving node receiving this packet.
      */
      op_td_set_int (pkptr, OPC_TDA_RA_PK_ACCEPT, OPC_TRUE);

      /* In either case the receiver channel is no longer locked. */
      rx_ch_obid = op_td_get_int (pkptr, OPC_TDA_RA_RX_CH_OBJID);
      op_ima_obj_attr_set (rx_ch_obid, "signal lock",
OPC_BOOLINT_DISABLED);


      FOUT;
      }
```