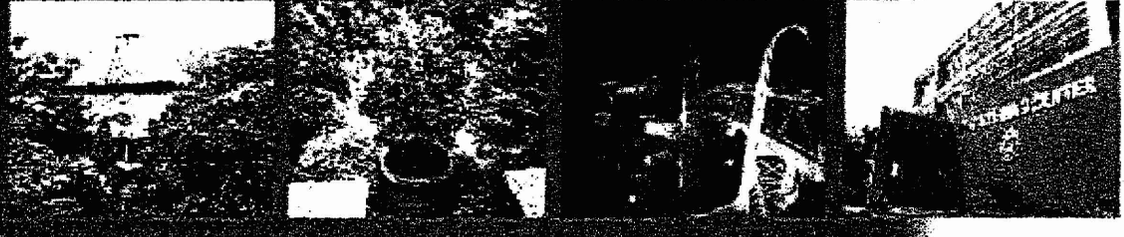
PROCEEDING

The 11th International Conference on QiR (Quality in Research)

QiR



Organized by:

Faculty of Engineering
University of Indonesia

3 - 6 August 2009,
Faculty of Engineering
University of Indonesia
http://qir.eng.ui.ac.id

ISSN 114-1284

ISSN 114-1284

Selecting a Cyclic Redundancy Check (CRC) Generator Polynomial for CEH (CRC Extension Header)

Supriyanto[a], Abidah M. Taib[b], Rahmat Budiarto[c]

[a,c]*School of Computer Sciences*
Universiti Sains Malaysia, Penang Malaysia 11800
[a]*Email: supriyanto@nav6.org*
[c]*Email: rahmat@cs.usm.my*

[b]*Department of Computer Science*
Universiti Teknologi Mara (UiTM) Perlis Malaysia
Email: abidah@perlis.uitm.edu.my

ABSTRACT

Computation and regeneration of CRC code in each router may cause slower IPv6 packet transmission. Utilizing advantages of IPv6 features namely IPv6 extension header and fiber optic medium, we proposed CRC extension header (CEH) to do error control in Network layer rather than in Data Link layer. The purpose is to reduce error checking process in IPv6 packet transmission over high speed networks. The CEH will utilize CRC-32 to do error detection. This paper investigates which CRC-32 generator polynomial would be suitable for CEH. To find out the answer we developed a simulation program in Java that generates IPv6 packet and CRC-32 code. The simulation produced CRC processing time both at the sender and receiver. The result showed that CRC-32 generator polynomial proposed by Castagnoli is the fastest generator polynomial to generate CRC code. We then conclude that Castagnoli generator (CRC-32C) is the best generator to apply in CEH on IPv6 transmission over high speed networks.

Keywords

CRC-32, error, generator polynomial, transmission

1. INTRODUCTION

Internet has been connecting million people in the world. They use Internet not only to communicate each other but also military, business and administration purposes. Sending information from one side to another via Internet is fast and easy but may get some changes on the data due to weakness of the medium or noise affecting the channel. To ensure the data is accurate and free from error, designers have already equipped the protocol stacks such as TCP/IP with error control mechanism.

Error control in TCP/IP is divided into two types: error control in upper layer and lower layer. Upper layer is Transport layer that employ TCP or UDP checksum for segment data. While lower layer is Data Link layer that handle transmission error. This paper focuses on lower layer error control that ensures link by link data transfer of adjacent node is free from error. Transmission errors which change one or more bit of data may be caused by medium used in data transmission. Following the OSI reference model, data from Network layer will be added with header and trailer at the Data Link layer. Trailer is actually frame check sequence (FCS) that contains cyclic redundancy check (CRC) 32 bits to do error checking for the whole fields of link layer frame.

The traditional protocol stacks such as TCP/IP does error checking process by calculating and regenerating the CRC code in each intermediate node. It has to calculate CRC-32 of each IPv6 packet in incoming port of router and regenerate CRC-32 code before forwarding to the next hop. In response to increasing network speed and advance of fiber optic technology, the error checking mechanism in each router eventually become a bottleneck [1]. This study addresses improving IPv6 packet processing by eliminating CRC computation in each router. We proposed a new IPv6 extension header called CRC Extension Header (CEH) to do error checking in Network layer. The CEH uses the same CRC-32 algorithm which is table lookup algorithm. Format of CEH can be seen in figure 1 [2] below.

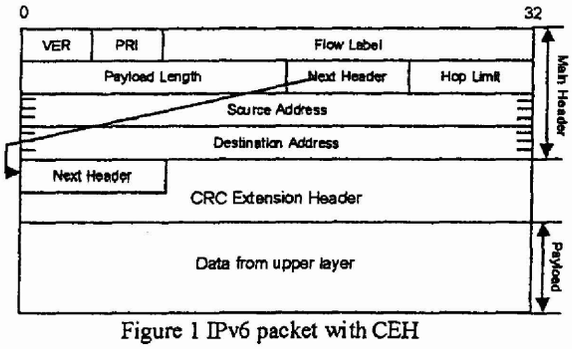


Figure 1 IPv6 packet with CEH

To generate CRC-32 code, it needs certain generator polynomial g(x). There are many generator polynomial

have been proposing by researchers. This paper intends selecting the best generator polynomial to be used in CEH. The rest of this paper is structured as follows. Section 2 describes the work by explaining overview of CRC operation. Section 3 discuses the candidates of generator polynomial that will be used in CEH. In Section 4, method of the selection is presented followed with result and discussion in section 5. The end of this paper is conclusion.

2. OVERVIEW OF CYCLIC REDUNDANCY CHECK

Cyclic redundancy check (CRC) is a code that used to detect errors that occur during transmission or storage of digital data information. Error detection is conducted by adding a code on the data transmitted and check the code at the receiver. There are various lengths of CRC codes such as 16 bits (CRC-16) [3] and 32 bits (CRC-32) [4]. This paper focuses on CRC-32 to be used in CEH to detect transmission error in Network layer. This section gives overview of CRC algorithm.

There are many algorithms utilized to generate CRC-32 code. This section introduces the simplest algorithm in order to understand the concept easily. Furthermore, it discusses the fastest algorithm which is table lookup algorithm that widely used today. The simplest way to know CRC code generation is algebraic approach as discussed in [5] and more explanation in [6]. We present a short overview of the algorithm.

In the algebraic approach, an information or message is interpreted as coefficient of polynomial called data word d(x). The data word is divided by pre determined CRC generator polynomial g(x) using equation 1 giving a code word c(x).

$$c(x) = q(x).g(x) + r(x) \quad (1)$$

The operation called modulo two division, meaning that all of the term on equation 1 is base two. q(x) is quotient of the division and r(x) is remainder of the division. CRC operation concerns on the remainder rather than quotient q(x). The remainder also could be obtained using equation 2, which m is the number of bits of the generator polynomial g(x) used or the highest degree of its polynomial. Data word d(x) is appended by 0s and represents with multiplication x^m.

$$r(x) = d(x).x^m \bmod g(x) \quad (2)$$

This operation to obtain r(x) is performed at the transmitter side. Code word c(x) which is d(x) + r(x) is transmitted along the network to reach a destination. Let say the receiver gets a code word c'(x) from the sender. Receiver conducts similar operation using the same generator.

$$c'(x) = c(x) + e(x) \quad (3)$$

There are two ways to justify whether there is an error in the code word received. Firstly, divide c'(x) using the same generator, if c'(x) is divisible by g(x) meaning there is no error or the error is undetected. Secondly, extract r(x) from the code word. Do operation of equation 2 using the same g(x) to get a new CRC code (remainder) r'(x) and then compare the r'(x) obtained with original r(x) extracted from the code word. If the two remainder is the same, there is no transmission error otherwise there is an error in the packet received.

Modulo 2 division generally could be performed by a sequence of shifts register. The division process makes addition and subtraction equal to bitwise XOR. Bitwise XORs are performed for all data including 0s appending till finish and obtain the remainder. The process is equal to shift bit by bit data to register from left most bits. In the hardware implementation uses LFSR (linier feedback shift register) of length m.

The process of bit by bit register needs more time and make it inefficient. To address the problem, engineers implement CRC operation in software using table lookup algorithm. The algorithm processes byte by byte instead of bit. The algorithm proposed by Sarwate [7]. *Firstly*, the CRC value is set to an initial value. Then data word is inputted to data stream byte by byte. *Secondly*, every byte of input stream is performed an XOR operation with the least significant byte of initial value. The result is used as index to access a 256 entry table. *Thirdly*, the value from the table is XORed with the rest of initial value by shifted byte by byte to the right. The result is CRC value to the next iteration.

In term of Ethernet, data word is the whole frame except frame check sequence field (FCS). As shown in figure 2, a standard Ethernet frame consist of destination and source Ethernet address, Ethernet type, data from upper layer and FCS.

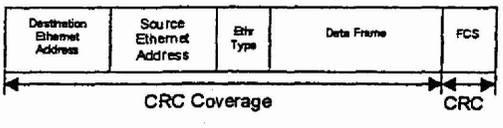


Figure 2 Standard Ethernet frame format

CRC coverage area in figure 2 is data word and FCS is remainder of modulo two division in equation 1. Hence, the whole frame is code word that actually transmitted into the networks.

3. IMPLEMENTATION CRC-32 FOR CEH

Implementation of CRC-32 in CEH uses the same algorithm with the existing CRC in Data Link layer which is table lookup algorithm. The differences of the new mechanism are coverage area and field of the CRC-

32. As mentioned previously, the CRC-32 is used as IPv6 extension header thus it placed between IPv6 main header and upper layer data *(see figure 1)*. The coverage area of CEH is the whole IPv6 packet excluding hop limit and CEH itself as shown in figure 3. From the figure, the size of CEH coverage area consist of 40 bytes IPv6 header minus 2 bytes hop limit field and upper layer data with maximum value 65,535 bytes. However the implementation is adapted with the maximum transmission unit (MTU) of the widely used Data Link layer technology which is Ethernet. Hence, the maximum MTU is 1500 bytes.

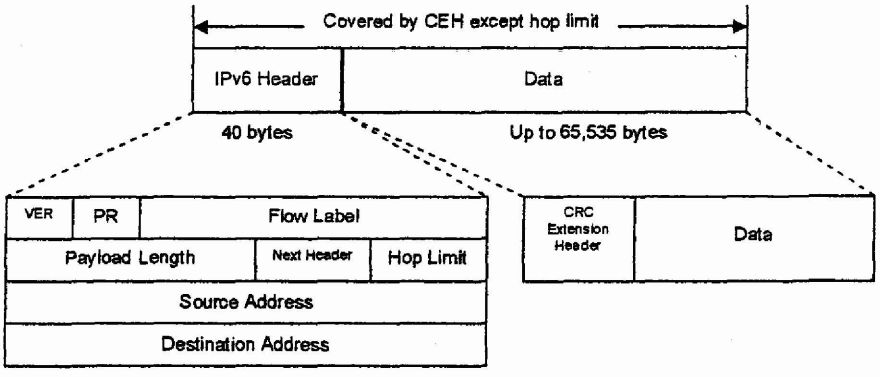


Figure 3 CEH coverage areas for error detection

Using equation 2, the coverage area is divided by g(x) to obtain r(x). r(x) is remainder of the division and represents of CRC code. Then, it is placed in the extension header field as shown in figure 4. Accordingly, pass the IPv6 packet to Data Link layer as usual to transmit through Physical layer without generate link layer trailer.

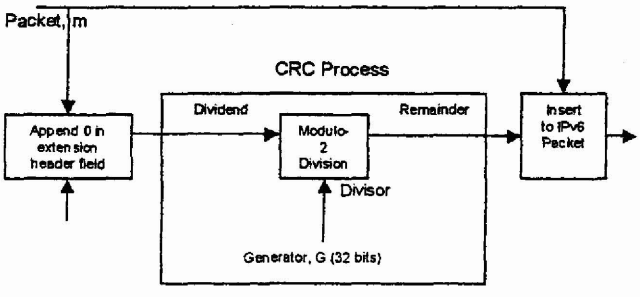


Figure 4 Generation of CEH in transmitter

At the receiver, Data Link layer captures the transmitted packet and pass the packet into Network layer directly. The layer does not compute CRC code any more. In the Network layer of the final destination, the packet will be processed as figure 5. It extracts CEH field from IPv6 packet. The other parts of the packet are divided by a generator to get a new CRC code. The new CRC code is compared with the original CRC inside CEH. If the result is similar, meaning no error. Thus, forward the packet to Transport layer, otherwise reject the packet.

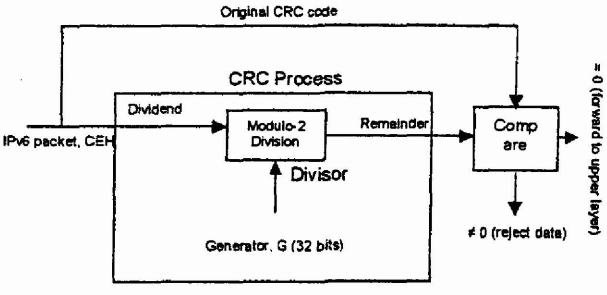


Figure 5 CEH processing in receiver

4. CANDIDATES OF GENERATOR POLYNOMIAL FOR CEH

Generator polynomial is the most important part of CRC code generation. It influences to the result of error detection. Hence selection of the best generator polynomial is also very important. This section discuses three candidates of generator polynomial that will be used in CEH they are standardized generator in IEEE 802.3 (CRC-32E), generator suggested by Guy Castagnoli (CRC-32C) and generator introduced by Philip Koopman (CRC-32K).

4.1 Generator used in Ethernet (CRC-32E)

This generator is widely used in data communication and data storage today. It was standardized by project IEEE 802.3 [8]. It is a polynomial with highest degree 32 and shown as

$$g(x) = x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+ x^{5}+x^{4}+x^{2}+x+1 \quad (4)$$

This polynomial can be formed as binary number as 100000100110000010001110110110111. The left most

1 is coefficient of x^{32} and following bits correspond to coefficient of the polynomial. The number of bits in a generator polynomial is equal to m + 1. The generator can be represented as 32 bit hexadecimal number 0x04C11DB7. However, some CRC implementations use its reverse as 0xEDB88320.

Based on [9], this generator satisfies for 4096≤codeword≤12144 bits. The interval is equal to the size of Ethernet frame (512 – 1518 bytes). For this implementation, it has Hamming distance, HD = 4 meaning the generator are able to detect all 3 bits error and lower.

4.2 Generator Proposed by Castagnoli (CRC-32C)

The name Castagnoli refers to the author of [10]. Castagnoli, Braurer and Herrmann evolved technique of constructing dual code polynomial belong to Fujiwara. They built special purpose hardware to find out new generator to improve performance of IEEE 802.3. Several factorization classes of generator polynomial of size 24 and 32 were evaluated. The evaluation yielded four optimum classes of 32 bits polynomials. *First*, CRC-32/8 code whose factors into $(x + 1)^2$ and three distinct irreducible polynomials of degree 10. The generator equivalent to codes of data length 1023 bits with HD = 8. *Second*, CRC-32/6 is the code of CRC-32 whose factors into $(x + 1)^2$ and two distinct primitive polynomial of degree 15. This generator similar with CRC-32 code for data length 32767 bits and HD = 6. *Third*, generator CRC-32/5 whose consists of one polynomial of degree 32. It gives HD = 5 to 65535 bits data. And the last is generator CRC-32/4. It was resulted from 47000 such codes that factors into (x + 1) times a primitive polynomial of degree 31. This generator keeps HD = 4 but it covers at data words sizes in excess 64 Kb. This generator is represented as

$$g(x) = x^{32}+x^{28}+x^{27}+x^{26}+x^{25}+x^{23}+x^{22}+x^{20}+x^{19}+x^{18}+ x^{14}+x^{13}+x^{11}+x^{10}+x^{9}+x^{8}+x^{6}+1 \quad (5)$$

In the hexadecimal form is 0x1EDC6F41 or 0x82F63B78 in its reverse. RFC 3385 [11] proved this generator to be used in iSCSI (Internet Protocol Small Computer System Interface). Thus we choose the code as a candidate to be used in CEH.

4.3 Generator Proposed by Koopman (CRC-32K)

Koopman did experiment to search a generator polynomial that covers larger data length [4]. He criticized the widely used generator polynomial that is IEEE 802.3. The standard only achieved Hamming Distance (HD) 4 for maximum packet length 12144 bits (Ethernet MTU). Whereas, theoretically it possible to detect HD = 6. The author searched a new generator polynomial that could to be used on HD = 6 for larger data length.

The evaluation was done for several generators including IEEE 802.3 and Castagnoli. The conclusion of the study is a 32 bits generator polynomial whose factors $(x+1)(x^3+x^2+1)(x^{28}+x^{22}+x^{20}+x^{19}+x^{16}+x^{14}+x^{12}+x^{9}+x^{8}+x^{6}+1)$. It constructs a full 32 bits generator represents as

$$g(x) = x^{32}+x^{30}+x^{29}+x^{28}+x^{26}+x^{20}+x^{19}+x^{17}+x^{16}+x^{15}+ x^{11}+x^{10}+x^{7}+x^{6}+x^{4}+x^{2}+x+1 \quad (6)$$

In binary form is 0x741B8CD7 or 0xEB31D82E of its reverse. The authors claimed the generator achieves HD = 6 for 16360 bits data length and HD = 4 to 114663 bits data length.

5. METHOD OF SELECTING GENERATOR POLYNOMIAL

We use two parameters to select the generator suitable for CEH based on previous work and our experiment. The two parameters are error detection capability and processing time to generate CEH in IPv6 packet transmission. The first parameter is analyzed by reviewing previous work on it and the latter analyzed by experiment. We configure small network to do the test bed in order to obtain particular data regarding processing time and delay transmission. The network is shown in figure 6.

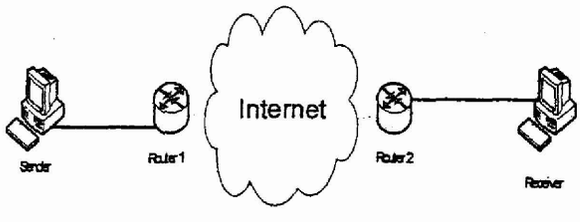


Figure 6 Topology of CRC Generator Polynomial Selection

We develop a program in Java that generates IPv6 packets with CRC extension header. The CRC code is generated from coverage area in figure 3. It is placed between IPv6 main header and TCP header and payload. The complete IPv6 packets are sent through the network. As common extension header, the CEH will not be processed in each router instead of in destination node indicated by destination address field. Another program is run in receiver side to compute the CRC code following figure 5.

6. RESULT AND DISCUSSION

In this section, we present result of our study on data length scope of the three candidates. Capability of generator polynomial to detect transmission error can be measured by its probability of undetected error [9], [10], [12]. This is summarized in table 1.

Based on table 1, HD = 4 and 6 whose available on the three candidates states obviously. This refers to the

existing widely generator polynomial which is Ethernet. It uses HD = 4 with minimum data length 4096 (512 bytes) and maximum 12144 (1518 bytes). The data length interval is used in standard IEEE 802.3 for all type of Ethernet. For CEH implementation, we chose to use HD = 4. It means all burst error with 3 bits and below is able to be detected by the generator polynomial.

Table 1 Summary of Data Length

Generator	Factor	HD	Data length (bit)
CRC-32E	32	3	$2^{32}-1$
		4	**4096 – 12144**
		5	512 – 2048
		6	204 – 300
CRC-32C	1,31	3	-
		4	**5276 – ($2^{31}-1$)**
		5	-
		6	210 – 5275
CRC-32K	1,3,28	3	-
		4	**16361 – 114663**
		5	-
		6	153 – 16360

Consider to RFC 2460 [12], the minimum MTU in IPv6 transmission over Ethernet is 1280 bytes or 10240 bits. This minimum length is able to be covered by CRC-32E and CRC-32C and it is not covered by CRC-32K. The maximum length of IPv6 MTU over Ethernet is 1518 bytes or 12144 bits. This value is also covered by CRC-32E and CRC-32C. However, for the CRC-32E this value is its maximum. Hence, it is difficult to use for the future because of increasing Ethernet MTU such as jumbo frame implementation. CRC-32C is suitable for larger MTU in the future.

Our experiment used topology in figure 6 to note processing time of each generator polynomial. Processing time of the first packet is different from the following packet. In case of the first packet, it runs full algorithm to generate a table 156-entry. Thus, time processing for the first packet is the biggest one. While other following packets consume fewer time to generate the CRC code. This is because it utilizes the preceding table lookup generated by the first packet.

We sent various sizes of IPv6 packet from sender to receiver: 64, 128, 25, 512, 1024, 1280 and 1492 bytes. The data documented are processing time in sender, receiver and total processing time. Processing time is the time required to generate CRC 32 code and insert it in IPv6 packet as extension header. With an assumption time of IPv6 packet generation is constant value, the processing times represent time to generate CRC code. The result is shown in figure 7, 8 and 9 respectively.

Figure 7 shows graph processing time (ms) vs packet size (bytes) at the sender side. The processing time increases with increasing packet size. This is inline with nature of CRC code that is linier code. Three generator polynomials show tight competition. All of packet size demonstrates small differences. However, the average processing time of CRC-32C is the lowest value that is 0.900 ms compare to CRC-32K 0.918 ms and CRC-32E 0.912 ms.

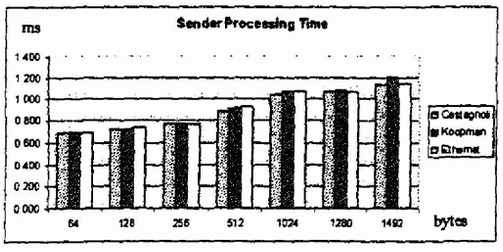


Figure 7 Sender Processing Time

Figure 8 shows processing time (ms) vs packet size (bytes) at the receiver side. The graph also demonstrates similar inclination with sender side. The processing time for the first IPv6 packet in receiver side increases with packet size increasing. However, receiver processing time is smaller than sender side. This is because in the receiver, there is no IPv6 packet generation.

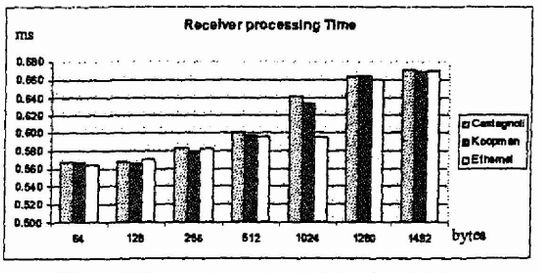
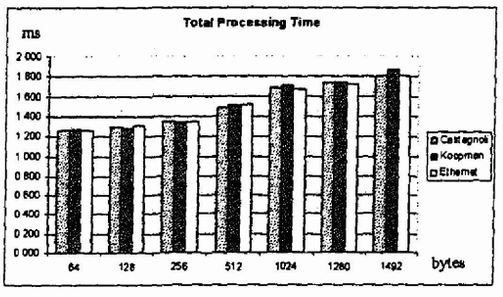


Figure 8 Processing Time in Receiver Side

The smallest average receiver processing time is belonging to CRC-32E that is 0.605 ms and 0.613 ms for CRC-32C and CRC-32K is 0.610 ms.

Total processing time of CRC code generation is shown in figure 9. Similar to previous processing time, the figure also illustrates that the three candidates are homogeny. It means the three generator polynomials are applicable in CEH from processing time point of view. Hence, capability of error detection that is analyzed in early of this section is a good way to determine which polynomial suitable for CEH.



As stated earlier, the processing time is based on the first packet. The other following packet need smaller time processing to generate CRC code. This is because there is a CRC code stored on table from the first packet. Figure 10 demonstrates processing time for common Internet packet size that is 1518 bytes. The graph shows time processing for the following packet in exponentially decrease. The three generators shows their trend line enumerated below

Figure 9 Total Processing Time

$$CRC\text{-}32C = 0.7601x^{-0.9562}$$
$$CRC\text{-}32K = 0.7626x^{-0.9422}$$
$$CRC\text{-}32E = 0.7612x^{-0.9554}$$

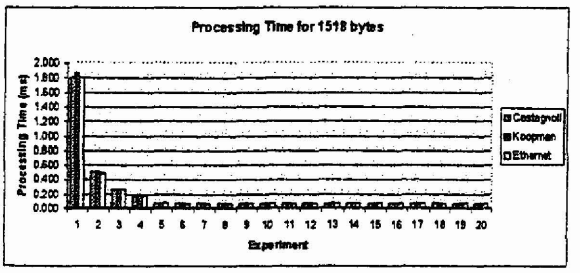


Figure 10 Time Processing for 1518 bytes packet size

The three exponential trend lines demonstrate that CRC-32C has smallest coefficient and power. This means the CRC generator polynomial has the best trend line. The processing time decreases toward smallest value.

7. CONCLUSION

We proposed CRC Extension Header (CEH) as a new mechanism to reduce duplicate error detection in intermediate node. CEH is applied to perform error control in the Network layer. With Network layer error control, we can eliminate error control in each intermediate node which possible to reduce the transmission time. CEH generation needs a generator polynomial. This paper described CEH and how selection of most suitable CRC-32 code generator polynomial for CEH was done. Among the generator used in existing Ethernet (CRC-32E) and two other generator suggested by Castagnoli (CRC-32C) and Koopman (CRC-32K), CRC-32C showed the best result. Its data length range covers larger data and at the same time the minimum data length also smaller than minimum MTU of IPv6 packet. The trend line of CRC-32C has smallest coefficient. We then conclude that CRC-32C suggested by Castagnoli is the most suitable generator polynomial for CRC extension header. Our future works will be implementing the CEH and analyzing its efficiency in handling the error detection in the high speed network.

REFERENCES

[1] F. Braun and M. Waldvogel, *Fast Incremental CRC Updates for IP over ATM Networks*, High IEEE Workshop on Performance Switching and Routing, 2001, pp. 48 – 52.

[2] Supriyanto, Raja Kumar Murugesan, Rahmat Budiarto, Sureswaran Ramadass, *Handling Transmission Error for IPv6 Packets over High Speed Networks*, Proceedings of the 4th International Conference on Distributed Framework for Multimedia Applications (DFMA 2008), Penang, 21-22 Oct., 2008, pp. 159-163.

[3] Castagnoli, G, Ganz, J. & Graber, P. *Optimum cycle redundancy-check codes with 16-bit redundancy*. IEEE Transactions on Communications, Vol. 38, 1990, pp. 111-114.

[4] Koopman, P. *32-bit cyclic redundancy codes for Internet applications*. Proceeding of International Conference on Dependable Systems and Networks, 2002.

[5] Martin Stigge, Henryk Plötz, Wolf Müller, Jens-Peter Redlich, *Reversing CRC Theory and Practice*. HU Berlin Public Report, 2006.

[6] Ross N. Williams. *A Painless Guide to CRC Error Detection Algorithms*. ftp://ftp.rocksoft.com/papers/crc_v3.txt, 1996.

[7] Sarwate, D. V. *Computation of Cyclic Redundancy Checks via Table Look-up. Commun. ACM*, Vol. 31, 1988. pp. 1008-1013.

[8] ANSI/IEEE Standard for Local Area Networks, *Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*. 1984.

[9] Fujiwara, T., Kasami, T. & Lin, S. *Error detecting capabilities of the shortened Hamming codes adopted for error detection in IEEE Standard 802.3*., IEEE Transactions on Communications, Vol. 37, 1989. pp. 986-989.

[10] Castagnoli, G., S. Brauer, and M. Herrmann, *Optimization of cyclic redundancy-check codes with 24 and 32 parity bits*. Communications, IEEE Transactions on, 1993. 41(6): p. 883-892.

[11] Shienwald, D. at. al. *Internet Protocol Small Computer System Interface (iSCSI) Cyclic Redundancy Checks (CRC/Checksum Considerations*. RFC 3385. The Internet Society.

[12] Deering, S. and R. Hinden (December 1998). *Internet Protocol Version 6 (IPv6) Specification*. RFC 2460. The Internet Society