International Journal of Advanced Science and Technology
Vol.84 (2015), pp.55-64
http://dx.doi.org/10.14257/ijast.2015.84.05

Coding Header Compression

Koj Sambyo[1], Anish Kumar Saha[2] and C. T. Bhunia[3]

Department of Computer Science and Engineering, National Institute of Technology, Arunachal Pradesh-791112, India
[1]sambyo.koj@gmail.com, [2]anishkumarsaha@gmail.com, [3]ctbhunia@vsnl.com

Abstract

Header compression increases efficiency. Further increase in efficiency is obtained by differential header compression. But subsequent packets have to be discarded if the context is not synchronized in differential header compression. Many algorithms had been proposed earlier to reduce discarding of subsequent packets. One of the basic techniques is use of reference value through which packets are encoded and decoded in compressor and decompressor respectively. Major disadvantage of this technique is that if the reference value is lost or erroneous then receiver cannot decompress the packets correctly. We have addressed this problem in the proposed scheme. In the proposed scheme even if the reference value or packets are lost or erroneous, receiver can regenerate the reference value and the packets. We have also shown that the proposed scheme has low average delay in high packet error rate and also works well in long round trip time. Some level of burst error is also handled by the proposed scheme.

Keywords: *Differential Header Compression, Header Compression, Coding, PDR, Delay, Latency, Common base, Packet Error, TCP, IP, Discard, RTT, ROHC*

1. Introduction

In many applications like VoIP, messages and internet based game; payload size is smaller than the header size. The payload of multimedia application ranges from 20 bytes to 160 bytes [1, 10]. The current overhead of 40 bytes per RTP/UDP/IPv4 packet is inefficient. The header size of IPv6 introduces higher overhead of 60 bytes. Tunneling protocols when used in IPv6 further increases the overhead by 100 bytes. Header compression techniques are needed to reduce the packet overhead. Packet overhead can further be reduced by differential header compression. Major problem with differentially header compression is error propagation. Decompressor has to discard subsequent packets even single packet is lost in differentially compressed header.

We propose to apply forward error correction on differentially compressed header [2]. Our scheme detects erroneous packet and correct the same in link with no feedback. It can also detect and correct some level of burst error.

To generate differentially header compression we will use [3], as it does not depend in context of previous packet received unlike Van Jacobson algorithm [4]. The size of difference is constant when uncommon portion of the set of packets to find common base is constant unlike Perkins and Mutka's [5] proposed algorithm. (*e.g.*, for finding common base, if packets are group from sequence number 0 to 15, 16 to 31…. the size of differentially compressed header is just 4 bit). Proposed scheme can also work even if the common base or reference is lost or erroneous. Simulation result shows that proposed scheme performs better than other existing schemes.

ISSN: 2005-4238 IJAST

2. Motivation and Background

2.1 Basic of Header Compression

Basic header correction scheme is of two types, one is forward error correction (FEC) and other is backward error correction (BEC). FEC is simplex link in which compressor sends compressed packet and if the packet is loss then subsequent packets are discarded until uncompressed packet arrives in decompressor. Whereas BEC is duplex link and for every erroneous packet decompressor encounter it send NACK to compressor.

Header fields belonging to a particular packet stream can be classified into 1) Inferred 2) Static 3) static known and 4) dynamic. The static fields are seldom sent since they never change during the lifetime of a packet stream. Static known are never sent since they are constant in any packet stream. Inferred field can be inferred by the decompressor from other fields or with the support of lower layer. The dynamic fields are communicated either directly or derived from other fields from other fields for every packet.

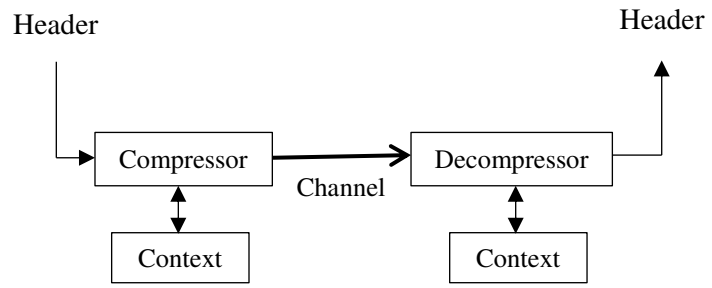

Figure 1. Basic Header Compression Scheme (Uni-directional)

Basic header compressor consists of compressor and decompressor equipped with a context buffer. Initially a regular uncompressed packet header is transmitted and copied into the context buffer on each side. Now compressed header can be transmitted as context has been established. Then packets are differentially encoded with respect to the context at the transmitter side and send as further compressed packet. In unidirectional FEC, if context are loss then it has to wait till timeout and compressor sends uncompressed packet.

In differential encoding loss of single packet leads to discarding of subsequent packets even if next context are received in correct order until compressor and decompressor are synchronized by the uncompressed header as shown in Figure 1.

2.2 Differential Header Compression

To generate fixed size from [3], we group the compressed header as per required size of differential compressed header. For example if we want our differentially compressed header to be of size four bit then sequence number of the compressed header can be group as: sequence number/ 2^4, generalizing this:

$$sizeof\_DHC = \frac{Seq\_No}{2^{sizeof\_DHC}}$$

Packets with same group are taken for finding common base.
Differentially header compression function as follow:

1. From a set of packets find common base
    a. If all bits in a packets are same (mostly MSB) consider the same bit.
    b. Else consider one in common base.
2. Synchronize compressor and decompressor with common base.
3. In compressor; Packet to send $\oplus$ common base.

4. In decompressor Received packet ⊕ common base.

XORing packet to be send with common base eliminate the common bits in MSB. Compressor send only the uncommon portion of the bits (mostly LSB).

Differential header compression proposed in [3] is better than Van Jacobson's algorithm as current packet does not depend on previous sent packet for synchronization. Also it is better than Perkins and Mukta's [5] proposed algorithm as the difference doesn't grow as number of packet increases.

3. Existing Methods

In bidirectional method of header compression, feedback from decompressor intimate compressor that refresh packets are required. If errors are propagated, compressor respond it by sending uncompressed header so that synchronization can be made. In unidirectional method of header compression, periodically uncompressed headers are send to minimize error propagation.

Long Round trip time (RTT) increase end to end delay and it also reduces the benefit of feedback because by the time compressor receives acknowledgement it had already sent large number of packet and decompressor has to discard subsequent packets. So in cases where there is long RTT or feedback is not possible, the lost packets are regenerated by decompressor such that it synchronize compressor and decompressor.

In TWICE algorithm [6], if the packet is lost then difference of packet header or delta value (of last correctly received header) are added twice by the decompressor in order to regenerate next header. The decompressed header are checked by the checksum in transport layer. It assume the fact that delta value are constant. This algorithm works effectively in reducing error propagation in low bit error rate. It fails severely in high bit error rate and cannot handle burst error in the network.

In ROHC U-mode [7], least significant bit (LSB) encoding is used. In this a reference value is stored in the context. Changes that occurs in the LSB are stored in the context. The original values are derive from received LSB and the previously received reference value in the decompressor. From interpretation interval, k bits of LSB are used to uniquely define the value.

$$f(vref, k) = [vref - p, vref + (2^k - 1) - p] \quad \text{(I)}$$

Interpretation interval ranges from $(vref - p)$ to $(vref + (2^k - 1) - p)$. Where p is an integer to shift the interpretation interval. The selection function is referred to as $g(vref,v)$ such that $k = g(vref,v)$. Where v is the uncompressed header. Here there is tradeoff involves in 'k' *i.e.*, the size of 'k' varies. Longer window (interpretation interval) ensures better performance against packet losses but increase the number of k bits. More over unidirectional RFC 3095 recommend all compressed headers to carry a CRC to verify correct decompression.

4. Proposed Scheme

Proposed scheme is for FEC where acknowledge is not possible or RTT is long. Differential header compression proposed in [2] is used in our scheme. Main disadvantages studies in [2, 4 and 7] is that if the first packet containing common base/ reference is lost, then receiver will not be able to regenerate the packets. We try to minimize the loss of first packet (reference) in the proposed scheme.

Start:

Compressor:

1. From a set of compressed header find common base

a. If all bits in a packets are same (mostly MSB) consider the same bit.
b. Else consider one in common base (mostly LSB).

2. Common base ⊕ packet to be sent (it gives LSB that are uncommon only)
3. Common base ⊕ all one (it gives common MSB only).
4. Arrange the packets in 4 X (2^n/4) matrix, with first packet containing MSB of common base.
5. Find FEC of columns and FEC' of rows by XORing columns and rows respectively.
6. Send packets, along with FEC and FEC'

Decompressor:

7. In receiver side find FEC1 and FEC1' compare with FEC and FEC' respectively
8. **Error Detection:** If FEC1 =! FEC && FEC1' =! FEC', then the intersection of unmatched FEC and FEC' is the erroneous packet.
9. **Error Correction:** XOR non-erroneous packet with FEC or FEC'. The generated bits are the corrected packet.
10. **Decompress of Common base:**
    a. All one ⊕ MSB of common base fill LSB as zeros
    b. Generated bits are required common base
11. Received packet ⊕ common base (Generated bits are packets)
12. Send to higher layer

End

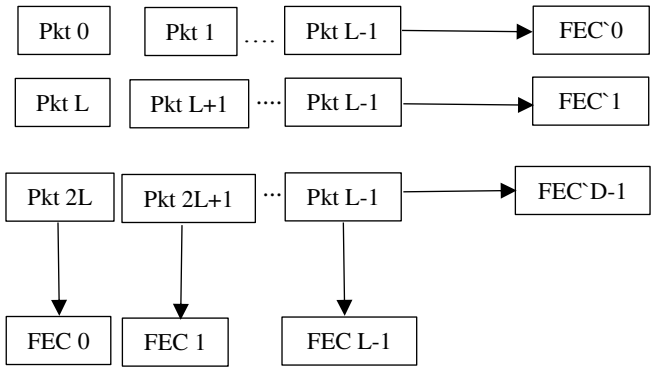

Figure 2. Duel FEC Mode Structure

This scheme consist of L columns and D rows with dual FEC mode structure as shown in Figure 2. It uses XORing for generation of FEC. The first FEC stream can cope with burst losses up to 'L' in length; the second FEC stream can cope with any single packet loss. It should be noted that the FEC can be used when L>= 4. The main advantages of this scheme is that, the size of common base are reduced and packets and common base can be regenerated even if the packets are lost or erroneous moreover it can also handle burst error to some extent.

Since in our proposed scheme, number of packets in groups are 2^n. Where n is the size of differential header compression. So duel FEC mode structure can be arrange as 4 X 2^n/4, as per size of buffer. Example is shown in appendix-I

5. Simulation and Result

5.1 Packet Discard Rate

We consider a situation in which a packets is lost independently of the other packet that are transmitted as in [8]. Let p be the probability of independent packet lost and α be

number of compressed header transmitted. After every α compressed header, one uncompressed header is transmitted for synchronization. Therefor window size is α+1. Let k denote number of packets discarder due to error then:

$$P(k) = (1-p)^{(\alpha+1)} \qquad k = 0 \tag{I}$$

$$= p(1-p)^{(\alpha+1-k)} \qquad k = 1, 2\ldots., \alpha+1 \tag{II}$$

Now average number of packet discarder in window α + 1 is given by:

$$\overline{m} = \sum_{k=0}^{\alpha+1} kp\,(1-p)^{\alpha+1-k} \tag{III}$$

If in a session, t number of packets are send containing uncompressed and compressed header. Then number of uncompressed header ≈ $\frac{t+\alpha}{\alpha+1}$. Therefore average packet discarder over a given session is given by:

$$\overline{M} = \left\lceil \frac{t+\alpha}{\alpha+1} \right\rceil \sum_{k=0}^{\alpha+1} kp\,(1-p)^{\alpha+1-k} \tag{IV}$$

Run of equation (IV) in compressed header, differentially compressed header (proposed scheme) and uncompressed header are depicted in Figure 3.

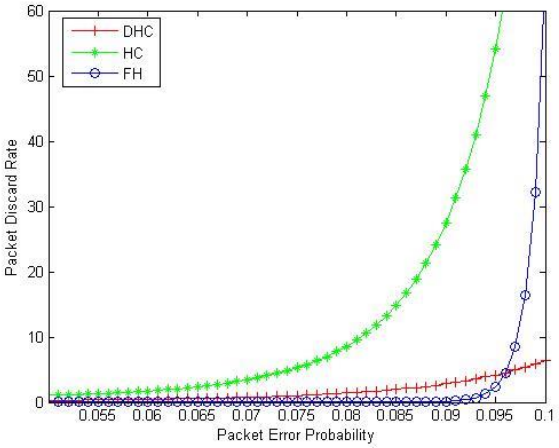

Figure 3. PDR of Full Header (FH), Compressed Header (CH) and Differentially Compressed Header (DCH) (Proposed Scheme)

Number of packets in a session is taken as 500 and window size is 16 packets. Packet error probability is taken from 10^{-3} to 10^{-1}. Figure 3 shows that compressed header has highest average packet discard rate as in the receiver side if a packet is lost then all the subsequent packets are to be discarded. Full header's discard rate is low when error probability is low but it gives high PDR when probability of packet loss is high. Proposed scheme has lowest PDR as it has ability to correct the erroneous packet.

5.2 Average Delay

In this we are considering delay due to encoding and decoding of the differentially compressed header in compressor and decompressor respectively and delay occurrence due to retransmission of the packets in acknowledgement mode. We are not considering nodal delay.

If X is Round trip time (RTT) of the packet and it is sent every 'y' unit of time, where X > y. Then at least X/y packets are sent before compressor could receive an acknowledgement [9]. Also let z be the number of packets the decompressor has to receive before it can send an acknowledgement. Thus number of packets sent before receipt of an acknowledgement is given by:

$$n = \left\lceil \frac{X}{y} \right\rceil + z \tag{V}$$

Therefore average delay when packets are discarder is given by:

$$d = \left\lceil \left\lceil \frac{X}{y} \right\rceil + z \right\rceil \left\lceil \frac{t+\alpha}{\alpha+1} \right\rceil \sum_{k=0}^{\alpha+1} kp\,(1-p)^{\alpha+1-k} \tag{VI}$$

Equation (VI) gives average delay for differentially compressed header when feedback (NACK/ACK) is used. Forward error correction does not use feedback. It try to detect and correct error. If β be the delay encore for encoding and γ be the delay encore for error detection and correction in compressor and decompressor respectively then average delay for proposed scheme is given by:

$$\overline{M\,1} = \beta\gamma \left(\frac{RTT}{2} \right) \sum_{k=0}^{\alpha+1} kp\,(1-p)^{\alpha+1-k} \tag{VII}$$

On run of equation (VI) and (VII) are given in Figure 4

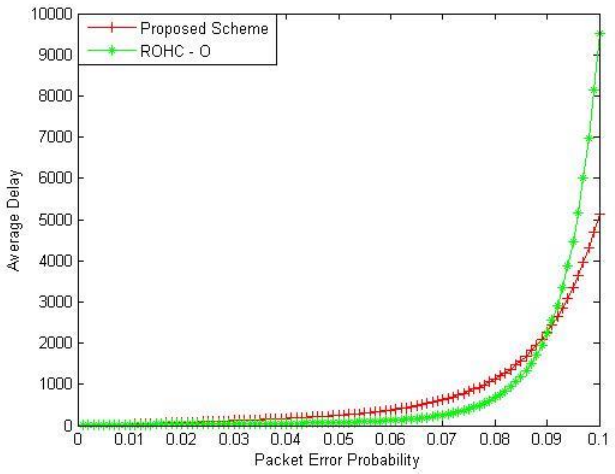

Figure 4. Average Delay of ROHC-O and Proposed Scheme with Respect to PER

For run of equation (VI) and (VII), the value of β and γ are taken as 8 unit each and RTT is taken as 100 and value of z ranges from 0 to 50 and y is 1.

Figure 4 show that in low packet error rate differential header compression works (ROHC-O) well but lost synchronization frequently as packet error rate is high. Thus in high packet error rate proposed scheme works better.

Figure 5 show that the proposed scheme also has less delay as round trip time increase. We can conclude that for low RTT and low PER differential header compression gives good result but for high PER and long RTT the proposed scheme or FEC gives better result.

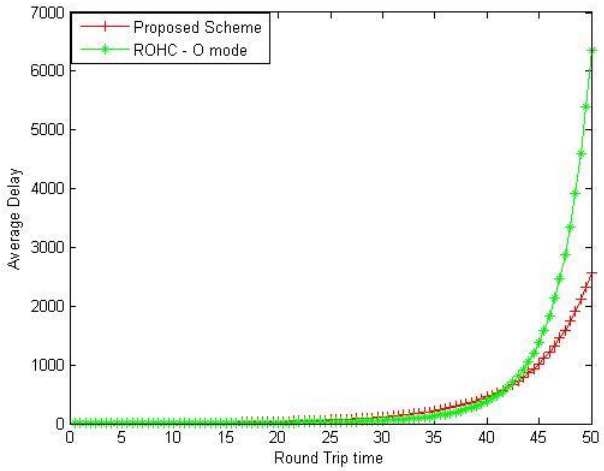

Figure 5. Average Delay of ROHC and Proposed Scheme with Respect to RTT

6. Conclusion

Forward error correction technique is proposed for differential header compression. XOR is used for differential header compression and also for FEC in compressor and decompressor. The proposed scheme can detect random error as well as can correct them. It can also handle burst error of some length. Proposed scheme has lower delay and also packet discard rate then BEC of differential header compressor. It gives better result than BEC in high packet error probability and also in long RTT.

Appendix

Consider an example where packets containing sequence number from 16 to 31 are to be sent. Let size of DHC is 4 bits. Group = 16/16 =17/16 =18/16 ….. = 31/16 *i.e.*, all falls under same group.

Transmitter:

As per proposed scheme the common base:

	0001 0000
	0001 0001
	…….
	0001 1111
Common base (MSB)	0001 1111

(0001 common (MSB) and 1111 uncommon (LSB)

XORing 16 to 31 with common base gives [Encoding Packets (LSB)]:

1) *0001 1111 ⊕ 0001 0000 = 1111*
2) *0001 1111 ⊕ 0001 0001 = 1110*
3) *0001 1111 ⊕ 0001 0010 = 1101*
4) *0001 1111 ⊕ 0001 0011 = 1100*
5) *0001 1111 ⊕ 0001 0100 = 1011*
6) *0001 1111 ⊕ 0001 0101 = 1010*
7) *0001 1111 ⊕ 0001 0110 = 1001*

8) 0001 1111 ⊕ 0001 0111 = 1000
9) 0001 1111 ⊕ 0001 1000 = 0111
10) 0001 1111 ⊕ 0001 1001 = 0110
11) 0001 1111 ⊕ 0001 1010 = 0101
12) 0001 1111 ⊕ 0001 1011 = 0100
13) 0001 1111 ⊕ 0001 1100 = 0011
14) 0001 1111 ⊕ 0001 1101 = 0010
15) 0001 1111 ⊕ 0001 1110 = 0001
16) 0001 1111 ⊕ 0001 1111 = 0000

Encoding common base (MSB):

1111 1111 ⊕ 0001 1111 = 1110

Place 1110 in (0, 0) and place encoded packets subsequently in (0, 1) …. (n-1, n-1) of the matrix.

FEC and FEC' are generated by XORing packets in columns and rows respectively:

					FEC'
	1110	1111	1110	1101	**0010**
	1100	1011	1010	1001	**0100**
	1000	0111	0110	0101	**1100**
	0100	0011	0010	0001	**0100**
FEC	**1110**	**0000**	**0000**	**0000**	1110

Packets along with FEC and FEC' are sent to the receiver.

Receiver:

If erroneous packets are received as (Packets with bold are error):

					FEC'1
	1010	1111	1110	1101	**0110**
	1100	1011	1010	1001	0100
	1000	0111	0110	0101	1100
	0100	0011	0010	0001	0100
FEC1	**1010**	0000	0000	0000	1110

Error Detection:

Generate FEC1 and FEC'1 by XORing packets in columns and rows respectively.
If FEC1= FEC and FEC'1=FEC1 then there is no error in the packets.
If FEC1=! FEC, there is error in column, if FEC'1 =! FEC1 there is error in row. Intersection point of row and column contain erroneous packet.

Error Correction:

XOR non erroneous packet with FEC, Generated bits is required packet.
i.e 1100 ⊕ 1000 ⊕ 0100 ⊕ 1110 = 1110
In similar way if any of row is erroneous (burst error of four packet (16 bits)) can be detected and also can be corrected.

Decoding Common base (MSB):
1111 1111⊕1110 0000 = 0001 1111

Decoding packets:

1) 0001 1111 ⊕ 0000 1111 = 0001 0000
2) 0001 1111 ⊕ 0000 1110 = 0001 0001
.
.
.
15) 0001 1111 ⊕ 0000 0001 = 0001 1110
16) 0001 1111 ⊕ 0000 0000 = 0001 1111

Decoded packets are sent to higher layer.

References

[1] Caida, Packet Length Distribution, 4 Aug'2004, Available http://www.caida.org/analysis/AIX/plen_hist/.
[2] Intellectual Property Terms of the Pro-MPEG Forum WAN Group published on the internet at www.pro-mpeg.org, 2002, 2003, 2004.
[3] K. Sambyo, A. Kumar Saha and C. T. Bhunia. "Annalysis of Differential Header Compression towards reducing latency".
[4] V. Jacobson, "TCP/IP Compression for Low-Speed Serial Links", RFC 1144 IETF Network Working Group, http://www.ietf.org/rfc/rfc1144.txt, **(1990)** February.
[5] S. J. Perkins and M. W. Mutka, "Dependency Removal for TransportProtocol Header compression over noisy channels", Proc. of IEEE International Conference on Communications (ICC), vol. 2, **(1997)** June, pp. 1025-1029.
[6] M. Degermak, M. Engan, B. Nordgren and S. Pink, "Low loss TCP/IPheader compression for Wireless Networks", mobicam96, New York, NY, **(1997)** October.
[7] C. Bormann, C. Burmeister, M. Degermark, H. Fukushima and H. Hannu, "RFC 3095-Robust Header Compression (ROHC): framework and four profiles: RTP,UDP,ESP and uncompressed", RFC 3095 IETF Network Working Group, July 2001, http://www.ietf.org/rfc/rfc3095.txt.
[8] V. A Suryavanshi, A. Nosratinia and R. Vedantham, "Resilient Packet Header Compression through Coding", Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE, vol. 3, pp. 1635-1639, **(2004)**, November 29-December 3.
[9] C. Westphal and R. Koodli, "IP Header Compression: A Study of Context Establishment", IEEE communication, **(2003)**, pp. 1025-1031.
[10] C. T. Bhunia, "Performance analysis of IP header compression", IETE Journal of Research, vol. 53, no. 2, **(2007)** March-April, pp. 113-118.