Adaptive Header Compression for Wireless Networks*

Changli Jiao
Dept. of Elec. and Computer Eng.
Wayne State University
Detroit, MI 48202
Email: aa0796@wayne.edu

Loren Schwiebert
Dept. of Computer Science
Wayne State University
Detroit, MI 48202
Email: loren@cs.wayne.edu

Golden Richard
Dept. of Computer Science
University of New Orleans
New Orleans, LA 70148
Email: golden@cs.uno.edu

Abstract

TCP/IP header compression has long been used to send information efficiently and to improve the response time of communication systems. It is also well known that errors on the link where header compression is used can deteriorate the performance. In addition, the recently noticed high frequency of some computer networking problems can make the performance of header compression even worse. These problems include packet reordering and packet errors that avoid link layer error detection. In this paper, we analyze the influence of these problems on existing header compression algorithms. We also propose an adaptive header compression that gives better performance.

1 Previous Work

Van Jacobson proposed a TCP/IP header compression algorithm [5] for low-speed links. TCP/IP packets with compressed headers normally traverse a single link, with a compressor on one side and a decompressor on the other. The idea is to avoid transferring redundant information whenever possible. The decompressor should use information already known to recover the headers. The TCP/IP header fields can be divided into several groups. Constant fields, which usually do not change during the lifetime of a connection, can be eliminated from most packets. Instead, a unique value, the Compression Identifier (CID), is assigned and added to the packet header to identify these fields. Inferable fields can be inferred from other fields and need not be transferred. Delta fields are expected to change only slightly from the previous packet. These fields are transferred using only the differences, which can be expressed with fewer bits and are referred to as "delta" values. Random fields are transferred without change.

Correctly decompressed packets refresh the information on the decompressor side, based on which following packets can be recovered. A corrupted packet is dropped and makes the decompressor desynchronized. Thus, subsequent packets received by the decompressor may also be dropped even though they are transmitted correctly, which is called *error propagation*. When the packet error rate is high, e.g., over a wireless channel, this might also result in senders not receiving ACKs, which is assumed as network congestion and which deteriorates TCP performance [2]. In order to solve this problem, the "twice" algorithm [2] was designed to repair the desynchronization. When a packet cannot be decompressed correctly, it is assumed that one or more previous packets, which carry the same delta values, are lost. The decompressor will apply these values two or more times. If the result can pass the TCP checksum, the packet is considered as correctly decompressed. The twice algorithm improves the performance of header compression in certain circumstances.

Window_based LSB [3] is an encoding method for delta values. If the compressor can ensure that the decompressor has received a group of packets, it can then transfer the delta values as the differences from that of the group, which are expected to occupy fewer bits. TCP_Aware RObust Header Compression (TAROC) [4] uses the sliding window of the TCP sender to decide which packets have already been received. A new packet cannot be transferred without getting all the acknowledgments for the previous window. Hence, the compressor tracks the size of the sliding window according to the arriving sequence of packets.

2 New Challenges

Packet reordering, due to packets of one connection following different paths, may seem uncommon. Bennett et al. showed that reordering is not necessarily a rare occurrence in the Internet [1]. Even when a consistent path is chosen, the existence of multiple paths between routing neighbors, or within the data path of a single router, can cause packet reordering. Some header compression algorithms are thus challenged since they assume that the packets arrive in order or with minor reordering.

Cyclic Redundancy Check (CRC) is very powerful in error detection and is the usual error detection method used in the link layer. TCP/UDP checksum was thus argued as unnecessary. But both experiment and tracing analysis have shown that there are a wide variety of error sources which cannot be detected by link-level CRCs [8].

*This material is based upon work supported by the National Science Foundation under Grant No. 0086020.

In TCP/UDP, these errors can be detected only by Internet checksum [6] [7]. The error detection performance should be analyzed for header compression algorithms.

3 Header Compression Performance

Both VJ and twice compression algorithm prohibit negative delta values, since these packets could be sent due to the desynchronization. Full headers are sent to refresh the decompressor, but packet reordering also triggers this. This introduces errors on estimating compression efficiency, which increases with more reordering. The error on the estimate also increases with the compression ratio.

A packet error means that packets cannot be decompressed, which includes physically corrupted ones as well as uncorrupted packets influenced by error propagation. Assume every bit has the same probability to be corrupted and any error will be detected by the error detection mechanism. Also assume all original packet lengths are 612 bytes, compressed are 517 bytes. In order to get Figure 1, we also make a random selection on when a full header packet will be sent by the compressor. For the VJ algorithm, packet error probability increases almost linearly until a full header packet is received. If packets are of the

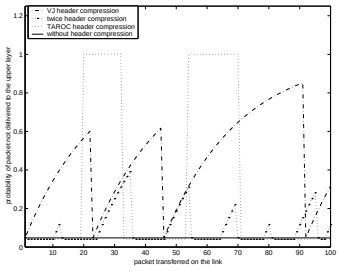

Figure 1: Packet error prob. comparison, $BER = 10^{-5}$

same size and there is no reordering/loss before the compressor, the twice algorithm always recovers the loss on the link between the compressor and decompressor. But, even for applications with same length packets, reordering makes delta values irregular, which causes the twice algorithm to not perform well. When applying delta values multiple times can not decompress the packet, the error probability increases almost linearly just as in VJ header compression. If the algorithm can decompress a subsequent packet, the error probability will become low again. Otherwise, the probability can drop only when a packet with a full header is received by the decompressor. Theoretically, TAROC will work perfectly if the compressor knows when the sender changes the window size, which is not an easy task. TAROC uses the packet arrival sequence as the window size indication. If packet reordering is assumed as an indication of packet loss, the compressor believes that the sender will adopt a smaller window size. Then the decompressor will be assumed to have received correctly the packets in a smaller previous window. This will potentially make the decompressor work incorrectly.

We define *error masking probability* as the probability that errors passing link layer CRC are not detected by the TCP checksum and thus passed to a higher layer. Error masking probability of VJ header compression needs further consideration. But it is clear that the error masking probability of the twice algorithm is higher than that of VJ compression. If the delta values are applied once and the TCP checksum is not correct, the actual reason may be some errors in the data payload or other bytes in the header of the packet. The more times the delta value is applied or the more educated guesses for the acknowledgment stream [2], the more likely that an error can be passed to the layer above TCP.

4 Adaptive Header Compression

We propose an adaptive header compression algorithm. Define a semi-compressed packet as one with all header fields compressed except delta fields. Define two variables, *windowSize* and *distance*. The delta fields are coded using Window_based LSB based on the previous *windowSize* packets. After every *distance* compressed packets, a semi-compressed packets is sent. The value of *windowSize* and *distance* are determined by the average BER of the link and the average packet size. The higher the BER, the bigger the *windowSize* and the smaller the *distance*. Conversely, the lower the BER, the smaller the *windowSize* and the bigger the *distance*. By changing these two variables, we can make the packet error probability less than that without header compression. In this way, the channel usage and TCP performance can be improved. A wireless channel is usually modeled as in *good* and *bad* states following a Markov chain. If we can get some indications as the wireless channel is in *bad* state, a semi-compressed packet should be sent immediately after this to refresh the decompressor, which also lowers the packet error probability.

References

[1] J. Bennett, C. Partridge, and N. Schectman. Packet reordering is not pathological network behavior, 1999.

[2] M. Degermark, M. Engan, B. Nordgren, and S. Pink. Low-loss TCP/IP header compression for wireless networks. In *MobiCom*, 1996.

[3] C. Bormann (ed.) *et al.* RObust Header Compression (ROHC). Internet Draft (work in progress) draft-ietf-rohc-rtp-09.txt, Internet Engineering Task Force, February 2001.

[4] H. Liao *et al.* TCP-Aware RObust Header Compression (TAROC). Internet Draft (work in progress) draft-ietf-rohc-tcp-taroc-01.txt, Internet Engineering Task Force, March 2001.

[5] V. Jacobson. IP headers for low-speed serial links. RFC 1144, Internet Engineering Task Force, February 1990.

[6] J. Postel. Transmission Control Protocol. RFC 793, Internet Engineering Task Force, September 1981.

[7] A. Rijsinghani. Computation of the Internet Checksum via Incremental Update. RFC 1624, Internet Engineering Task Force, May 1994.

[8] J. Stone and C. Partridge. When The CRC and TCP Checksum Disagree. *ACM SIGCOMM*, September 2000.