IEEE 802.11
Wireless Access Method and Physical Layer Specifications

Title: Analysis of a Data Compression Method as a MAC Option

Author: Waychi Doo
National Semiconductor
Wireless Networking
Santa Clara, CA 95052
Phone: 408-721-6887
Fax: 408-721-4115
Email: wcd@berlioz.nsc.com

Abstract

It is a desirable function to increase the data throughput of wireless links by applying compression technology. In this contribution, the compression performance of the LZ77-based data compression scheme is evaluated. Based on the experimental results, we identify desirable service functions and guidelines in the MAC protocol to achieve better data compression performance. These service functions and guidelines apply to the LZ78-based data compression algorithms as well. It is not intended to make a recommendation to a specific lossless data compression algorithm for wireless LAN in this contribution.

1. Introduction

It is a desirable function to increase the data throughput of wireless links by applying compression technology [1]. In this contribution, the compression performance of the LZ77-based data compression scheme is evaluated. Based on the experimental results, we identify desirable service functions and guidelines in the MAC protocol to achieve better data compression performance. These service functions and guidelines apply to the LZ78-based data compression algorithms as well.

The LZ77-based data compression scheme is performed by replacing redundant strings in a data stream with short tokens. It uses a compression history, or sliding window, to keep track of the last n bytes of data. When a phrase is encountered that has already been seen, it outputs a pair of value corresponding to the position of the phrase in the previously seen history buffer of data and the length of the phrase. The key parameter of LZ77-base scheme is the maximum history buffer size [2] [3].

The LZ78-based data compression scheme builds a dictionary by adding phrases from the input data stream. When a repeat occurrence of that particular phrase is found in the dictionary, it outputs the dictionary index instead of the phrase and plus the next character from the input data stream. The maximum dictionary size is the key parameter of LZ78-based data compression scheme [3] [4].

2. Experiment Samples

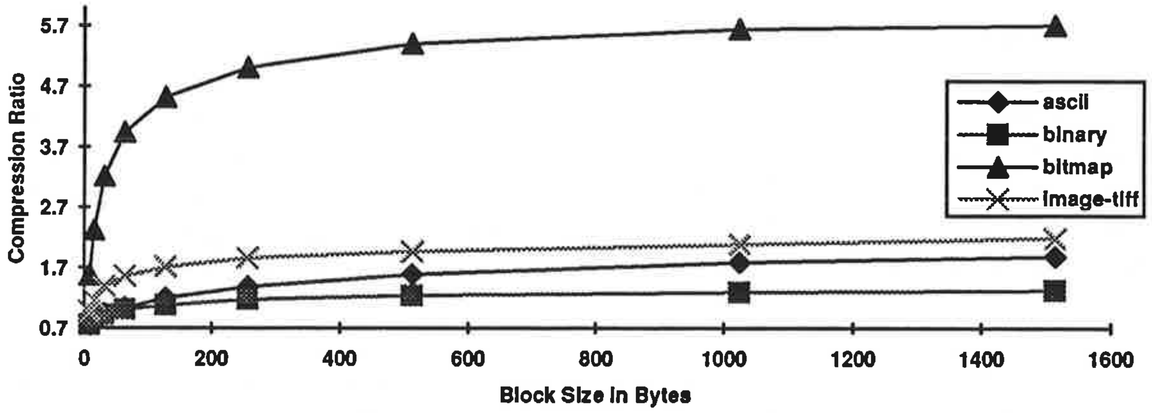
Four types of data stream were evaluated through STAC's LZS demo program, an LZ77-based scheme [5]. The compression software compresses the input data stream with fixed size block. A flush operation of the history buffer is performed at the end of each block.

1. ASCII File
2. Binary File
3. Bit Map File
4. Image File

The compression ratio is defined as

$$\textit{compression ratio} = \frac{\textit{size of source file}}{\textit{size of compressed file}}$$

3. Desired MAC Services



From the above experimental results and the characteristics of the LZ77-based scheme, the compression ratio drops sharply under the following conditions

1. Rather random data pattern
2. Small frames (not much information in history buffer)
3. Non-correlative frames (dilute information density in history buffer)

From the observation item 1 (rather random data pattern), the MAC protocol should apply the compression function before the encryption algorithm to the input data stream because the encryption algorithm creates a rather random output data stream.

To address the small frames issue, the maximum size of the MAC service data unit should be large enough to get a reasonable compression ratio. Because of the noisy wireless environment, the size of MAC protocol data unit may be relatively small. It is desirable for a MAC protocol to support segmentation and reassembly of its large compressed payload.

For the observation item 3 (non-correlative frames), it happens when the application data mix with the control messages of the high level protocols as MAC service data units. Most of high level protocol control messages are small. The MAC protocol should not compress the service data units, if they are small, to avoid the non-correlative frame effect that dilutes the information density in the history buffer. The second case of non-correlative frame happens when frames go to different destination nodes. A separate history buffer has to be maintained for each pair of communicating nodes to not break the data compression scheme. Maintaining a separate history buffer for each pair of nodes takes more memory and also adds extra burden in the data compression scheme, especially in the case of re-synchronization of their history buffer due to lost or damaged frames. There are two ways to avoid this problem of out of synchronization between sender and receiver history buffers. The first one requires that the MAC protocol provides a reliable transmission service like ISDN LAPD service. The second solution is that no accumulated history information is maintained for any pair of nodes; each MAC service data unit starts with an empty history buffer before going through the data compression algorithm. The second solution performs the compression on each packet and does not keep the history buffer at the end of the compression operation. It sacrifices the overall compression ratio but provides a much simple data compression scheme.

Without maintaining the accumulated history information, the MAC should transmit the raw payload instead of the compressed one if the size of the compressed data is larger than the original payload. The following pseudo code presents the implementation of the data compression in the MAC protocol under no accumulated history information assumption.

```
IF sizeof (MAC payload) less than MIN_COMPRESS_UNIT
THEN
    transmit raw payload
ELSE
    compress raw payload
    IF compress-ratio less than or equal to 1.0
    THEN
        transmit raw payload
    ELSE
        transmit compressed payload
    END
END
```

During the association phase, mobile station and access point/mobile station negotiate which data compression algorithm will be used for later data exchange. After selecting the compression algorithm, the MAC frame header may need to convey the information to indicate the payload that is compressed or not.

4. CONCLUSION

In this contribution, desirable MAC service functions and guidelines to achieve better data compression performance over wireless links were discussed. The MAC protocol should apply the compression function before the encryption algorithm. The segmentation and reassembly functions of the MAC protocol are desirable to get better compression ratio. The decision to maintain accumulated compression history information to achieve higher compression ratios is a performance/complexity trade-off that should be outside the scope of the standard. If the MAC protocol does deliver a reliable transmission service, it makes data compression algorithm a little bit simpler because it does not need to consider the lost or damaged frames that cause out of synchronization between sender and receiver history buffers. Since the LZ78-based data compression scheme is similar to the LZ77-based scheme, these MAC services and guidelines apply to the LZ78-based scheme as well.

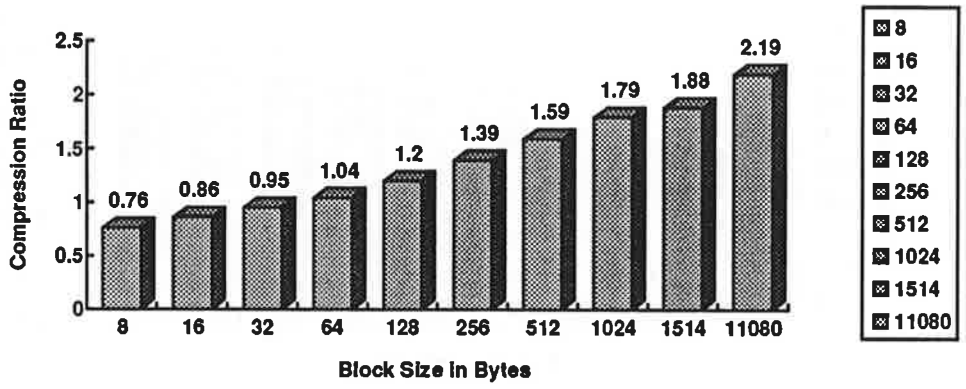
5. BIBLIOGRAPHY

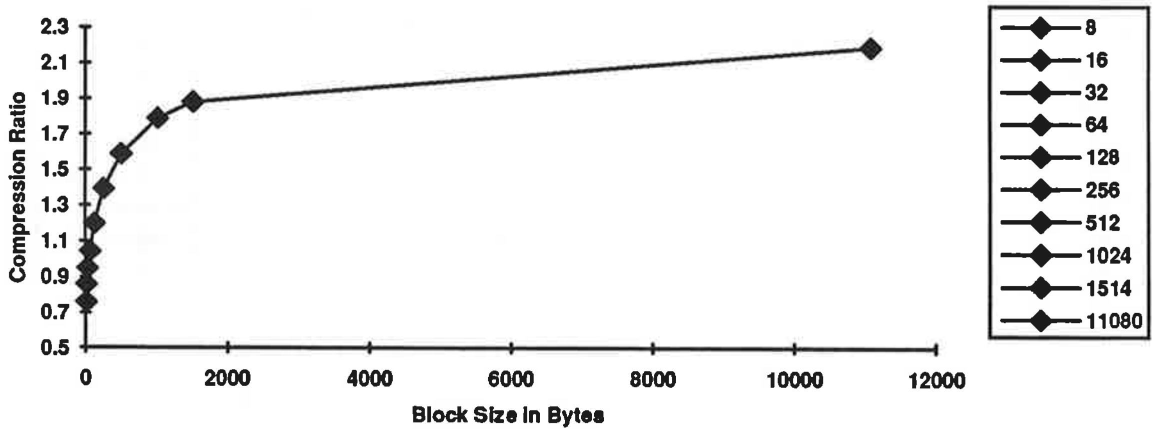
1. Frederic J. Bauchot, Wireless LAN MAC Protocol: Data Compression as a MAC Option to Improve Effective Throughput, IEEE P802.11/93-29
2. Jeffrey Weiss, Putting Data on a Diet, IEEE Spectrum 8/1993, P.36-39
3. T.C. Bell, J.G. Cleary, and I.H. Witten, Text Compression by Prentice Hall
4. Clark Thomborson, The V.42bis Standard for Data-Compressing Modems, IEEE Micro 10/1992, P.41-53
5. STAC Electronics, LZS demo program and data sheet

CASE 1 - ASCII file (regular text file)

File size: 11080 bytes

block size (in byte)	compression ratio		compressed file size from 11080 to (in byte)
8	-31%	0.76:1	14529
16	-16%	0.86:1	12897
32	-5%	0.95:1	11714
64	4%	1.04:1	10620
128	16%	1.20:1	9248
256	27%	1.39:1	7991
512	36%	1.59:1	6981
1024	44%	1.79:1	6173
1514	46%	1.88:1	5887
11080	54%	2.19:1	5049

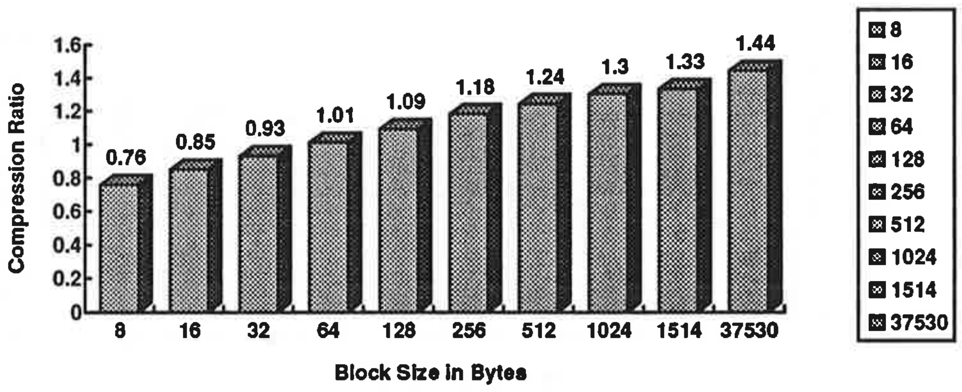


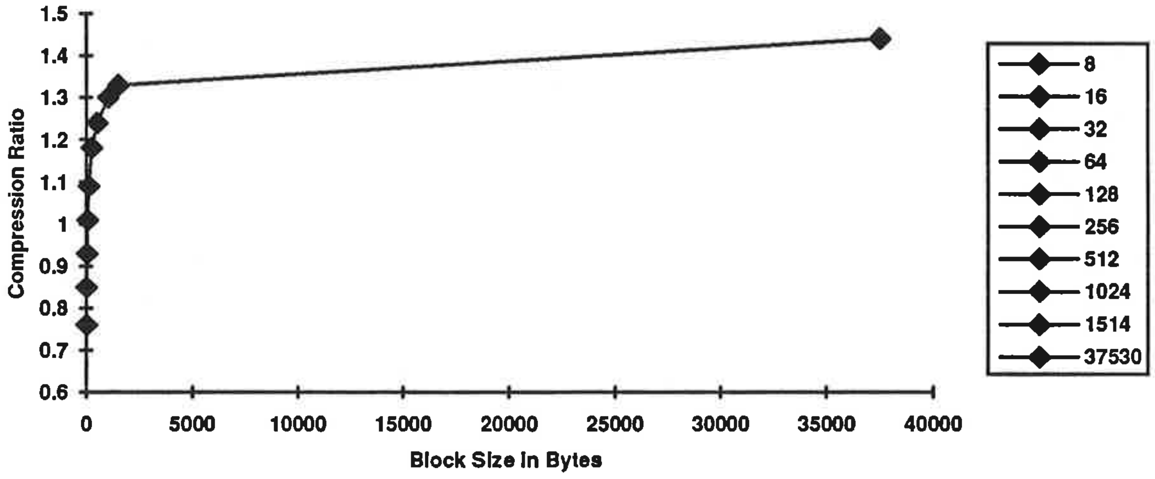


CASE 2 - Binary files

File size: 37350 bytes

block size (in byte)	compression ratio		compressed file size from 37350 to (in byte)
8	-32%	0.76:1	49660
16	-17%	0.85:1	44072
32	-7%	0.93:1	40213
64	1%	1.01:1	37084
128	8%	1.09:1	34344
256	14%	1.18:1	31932
512	19%	1.24:1	30268
1024	23%	1.30:1	28775
1514	24%	1.33:1	28205
37350	30%	1.44:1	26049

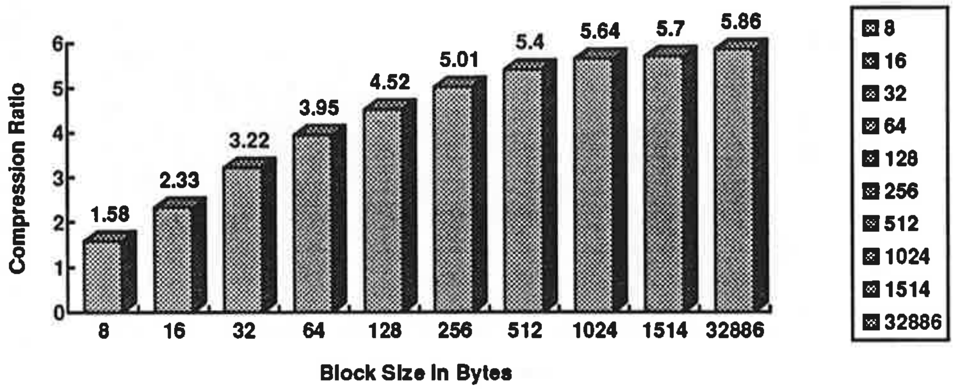


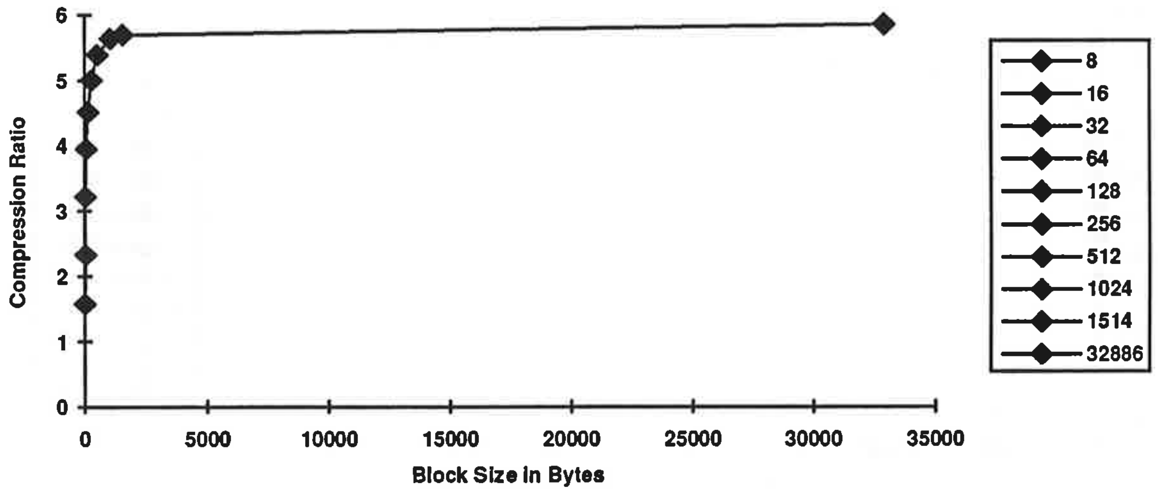


CASE 3 - Bitmap files

File size: 32886 bytes

block size (in byte)	compression ratio		compressed file size from 32886 to (in byte)
8	36%	1.58:1	20817
16	56%	2.33:1	14141
32	68%	3.22:1	10214
64	74%	3.95:1	8324
128	77%	4.52:1	7283
256	80%	5.01:1	6570
512	81%	5.40:1	6087
1024	82%	5.64:1	5829
1514	82%	5.70:1	5768
32886	82%	5.86:1	5614

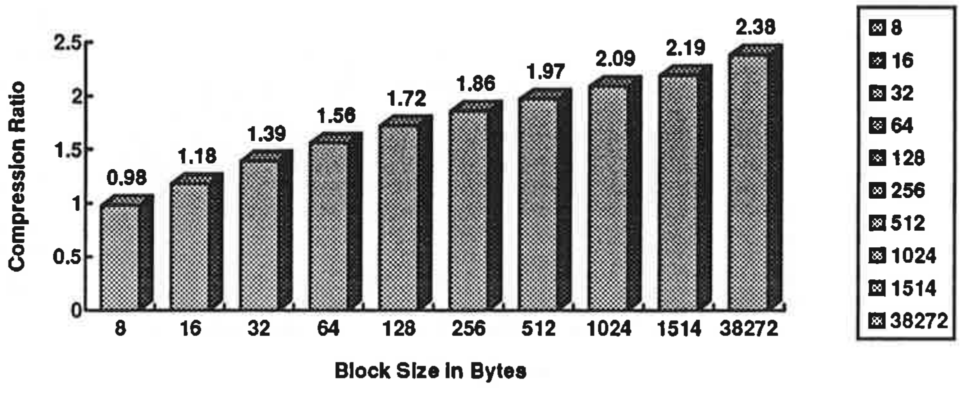


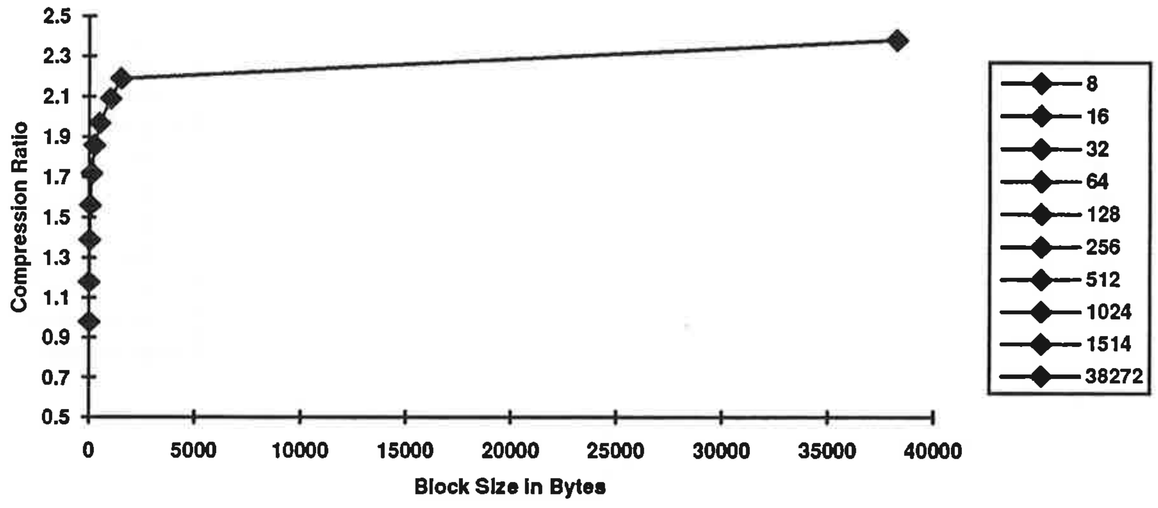


CASE 4 - image file (TIFF format)

File size: 38272 bytes

block size (in byte)	compression ratio		compressed file size from 38272 to (in byte)
8	-1%	0.98:1	38931
16	15%	1.18:1	32353
32	27%	1.39:1	27557
64	35%	1.56:1	24595
128	41%	1.72:1	22234
256	46%	1.86:1	20623
512	49%	1.97:1	19381
1024	52%	2.09:1	18272
1514	54%	2.19:1	17469
38272	57%	2.38:1	16038


Compression Ratio
Block Size in Bytes


Compression Ratio
Block Size in Bytes