Data Compression in Wireless LAN, General Issues

Robert Lutz
Stac Electronics
5993 Avenida Encinas
Carlsbad, CA 92008

Phone - (619) 431-7474
Fax - (619) 431-5726
Email - Stac/Stac/BobL%Stac_Electronics@mcimail.com

Abstract

Data compression can improve the data throughput of a wireless LAN. The speed gain can range from nothing to 15 times faster, or more depending upon the algorithm used, the efficiency of the MAC and PHY layers, and the data being transmitted. A speed gain of 2 to 3 times faster would be a typical value. The efficiency of data compression could be effected by the level of support offered by the MAC protocol.

Introduction

Wireless LANs appear to offer all the benefits of a wired LAN with the added features of mobility and simple network accessibility. Unfortunately, this comes at the expense of throughput.

There are two phenomena that the end-user may notice when comparing a wireless LAN to a wired LAN:

1) The speed of the network is slower.

2) The network may become saturated with less activity.

The speed of the network will be slower because the raw transmission rate of a wireless node (≅1 Mbit/s) is slower than a wired node (≅10 to 16 Mbit/s). This may not be noticeable for normal inquiry network traffic, but it could be profound during a bulk data transfer, or a file transfer operation as shown in Figure 1.

Another wireless LAN characteristic which may lower the speed of the network is a higher bit error rate. The wireless medium is more prone to errors than its hard-wired counterpart. To counteract the higher error rate, longer and more powerful ECC must be used. Additionally, the data packet size may be smaller than a wired network, increasing packet overhead.

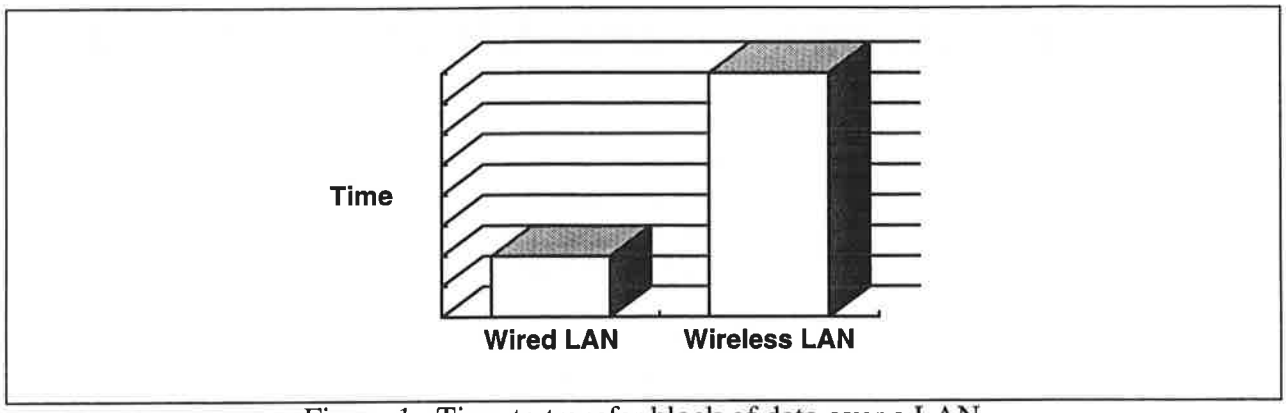

Figure 1. Time to transfer block of data over a LAN

The network will become saturated with less activity due to the limited bandwidth of the transmission medium. Since less data can be transmitted over a period of time, it will require significantly less traffic to cause a collision based protocol to reach its *critical capacity*. The critical capacity is the amount of traffic that will cause a network to begin to lose efficiency. The critical capacity of a typical network is shown in Figure 2

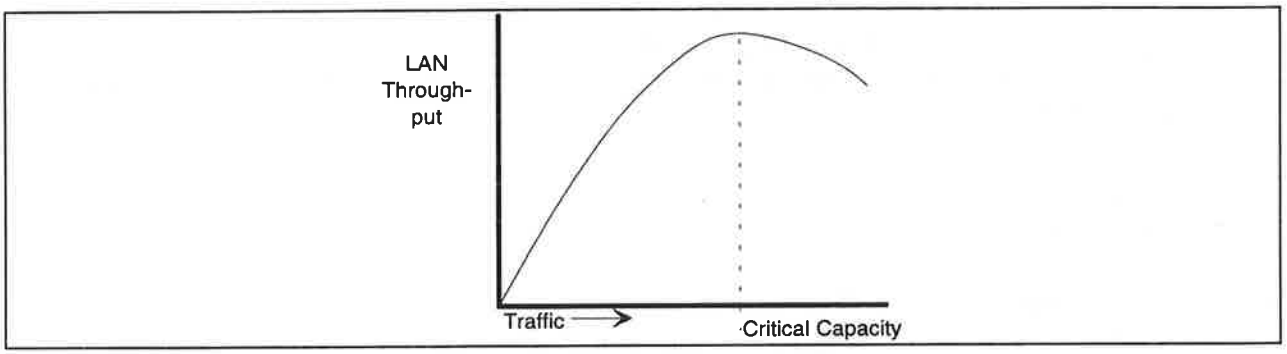

Figure 2. Bandwidth vs. traffic

Data Compression

The purpose of data compression in a network is to increase the network throughput by reducing the amount of data to be transferred. The *compression ratio* is a measurement of the amount of data reduced by a compression algorithm. The compression ratio is represented as x:1, where x is the result of the following formula:

(original byte count) / (resulting byte count).

For example, if a block of data is compressed to half its original size, the compression ratio is stated as 2:1.

The compression ratio that can be obtained on a block of data is strongly related to several things:

1) Lossy vs. lossless compression algorithm.
2) Specific compression algorithm characteristics.
3) Property of the data to be compressed.
4) Block size of data to be compressed.
5) Compression history retention.

Lossy and Lossless Compression Algorithms

Lossy compression algorithms allow some lose of data during the compression operation. On the receiving side, the decompressed data will not exactly match the original data. Lossy compression is often used for image or audio data where the loss of this precision can be tolerated.

Lossless compression algorithms retain all the original information. It is used for all normal types of data, where a loss of information cannot be tolerated, such as spreadsheets, data bases, text documents, etc. This paper only covers the issues of lossless compression algorithms.

Compression Algorithm Characteristics

There are many compression algorithms which may be used for lossless data compression. These algorithms can be evaluated by comparing several parameters including speed, minimum compression ratio, maximum compression ratio, average compression ratio, memory requirements, hardware availability, support for multiple compression histories, industry standards, etc. I will leave the discussion of comparing the features of different lossless compression algorithms for another paper.

Data Characteristics

Some types of data are highly compressible (such as database and spreadsheet files). Other types of data are more difficult to compress (such as binary executables). Typical compression ratios for a variety of file types are shown in Figure 3.

Due to the wide variety of compression ratios achievable by a single compression algorithm, it is unclear how to specify the average compression ratio that will be achieved. Therefore, marketing claims of average compression ratios are practically worthless. Only a side-by-side comparison of compression algorithms using the same set of files may be used accurately determine relative strengths.

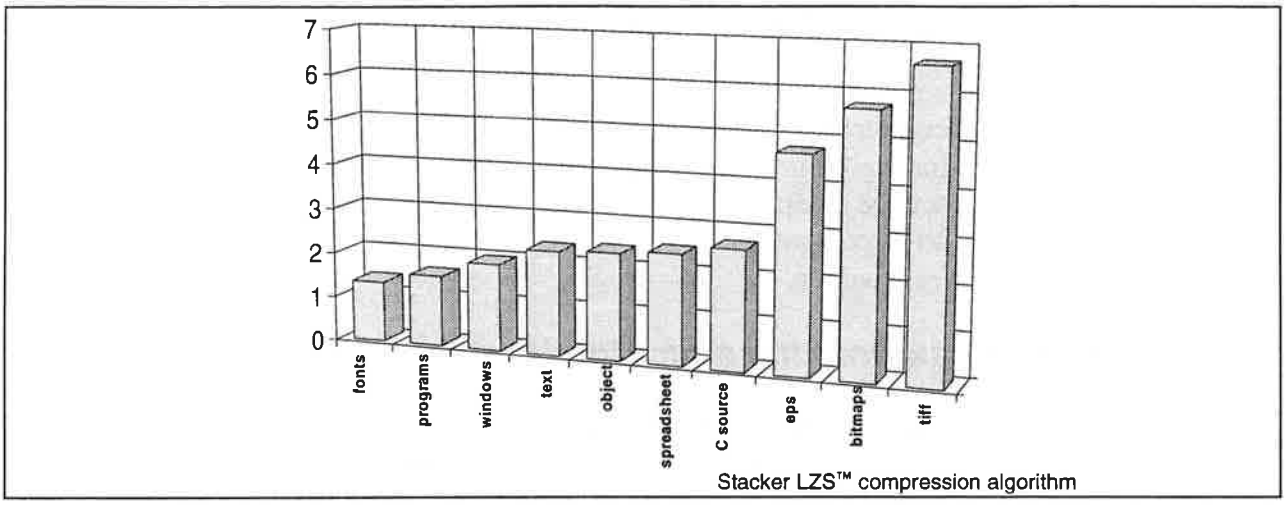

Figure 3. Typical compression ratio by file type for lossless compression

Data Block Size

The efficiency of compression is related to the amount of data that may be compressed without resetting the compression history.

Figure 4 shows the relationship between compression ratio, and the number of bytes processed. As you can see, since the compression history is initially empty, the compression ratio begins at nearly zero. Soon, however, the compression ratio begins to rise as more data is processed. Eventually, the compression history will become completely full This will maximize the likelihood of finding redundant strings.

The Stacker LZS algorithm reaches the point of maximum compression ratio after 2 Kbytes of data have been processed. Since the compression history remains adaptive, the maximum compression ratio will remain at its optimum point forever, or at least until data from a different source or type is processed.

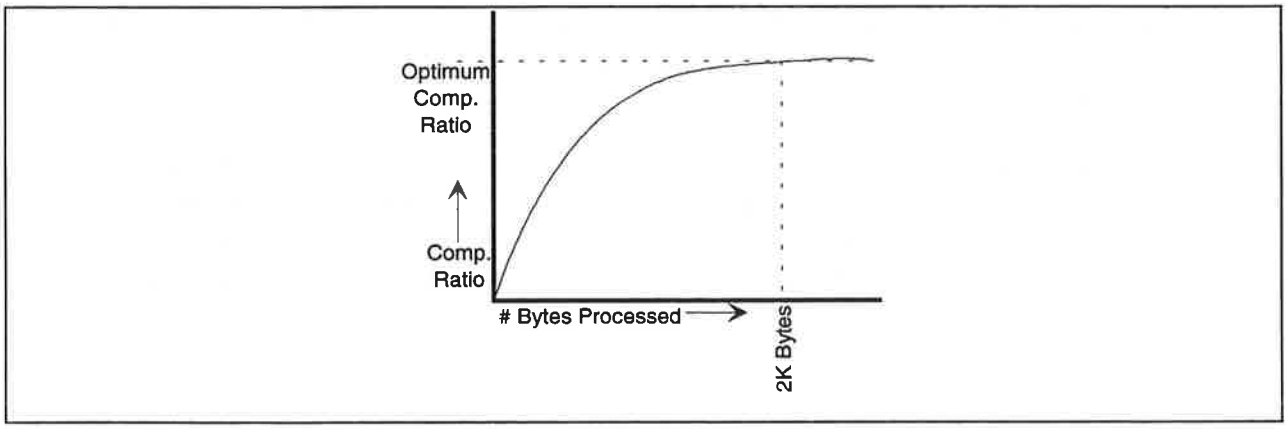

Figure 4. Compression ratio vs. number of bytes processed

The number of bytes in a packet of data on a LAN can be quite small. If the compression history is reset after each packet, the compression ratio would never reach the optimum value. At the

beginning of each packet, the compression ratio would begin at near zero. This is shown in Figure 5.

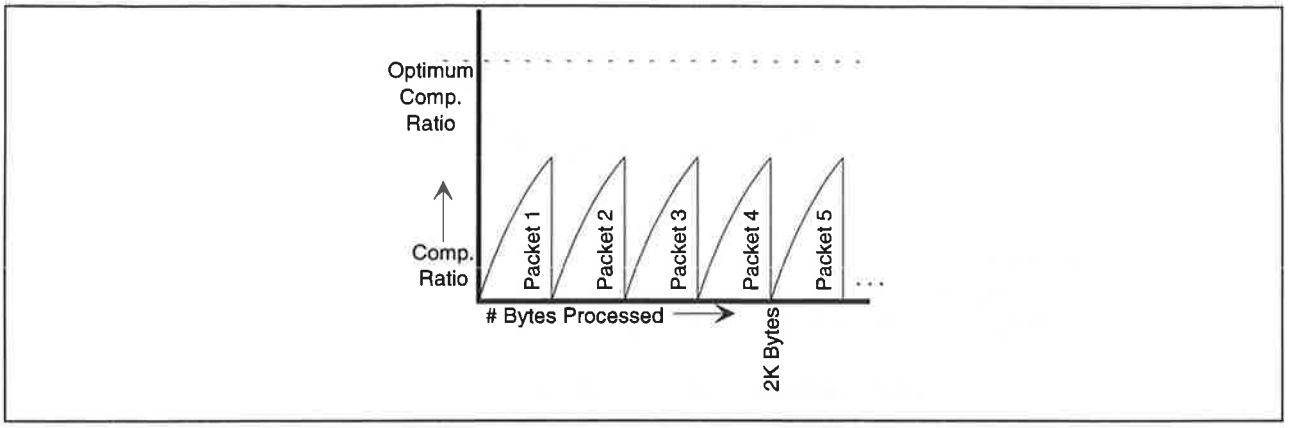

Figure 5. Compression ratio with small packets, resetting the history

If the compression history is maintained (not reset) between packets, the compression ratio would increase as each packet is processed, and will reach the optimum compression ratio, as shown in Figure 6.

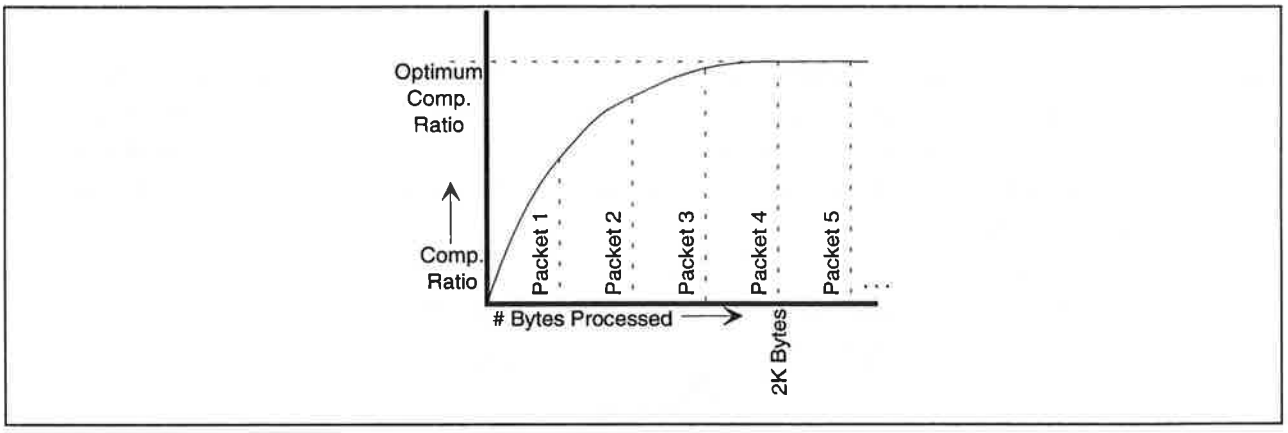

Figure 6. Compression ratio with small packet, without resetting the history

When the compression history is preserved between packets, it is important that ordering be maintained. The data must be decompressed in the same order in which it was compressed.

Error recovery is another issue. A correctable error will not adversely effect data compression. But an unrecoverable error may require a resynchronization (or a reset) of the compression history. Several techniques may be used for this resynchronization. I will leave the discussion of resynchronization for another paper.

Multiple Virtual Connections

The compression history on the receiving end of a LAN is created during decompression from information in the compressed data stream. If one source node is communicating with more than one receiving node, the compression history at each node will be different. This could produce

erroneous data. It is important that the compression history used by a transmitting node be identical to the compression history at the receiving node. This problem is demonstrated in Figure 7.

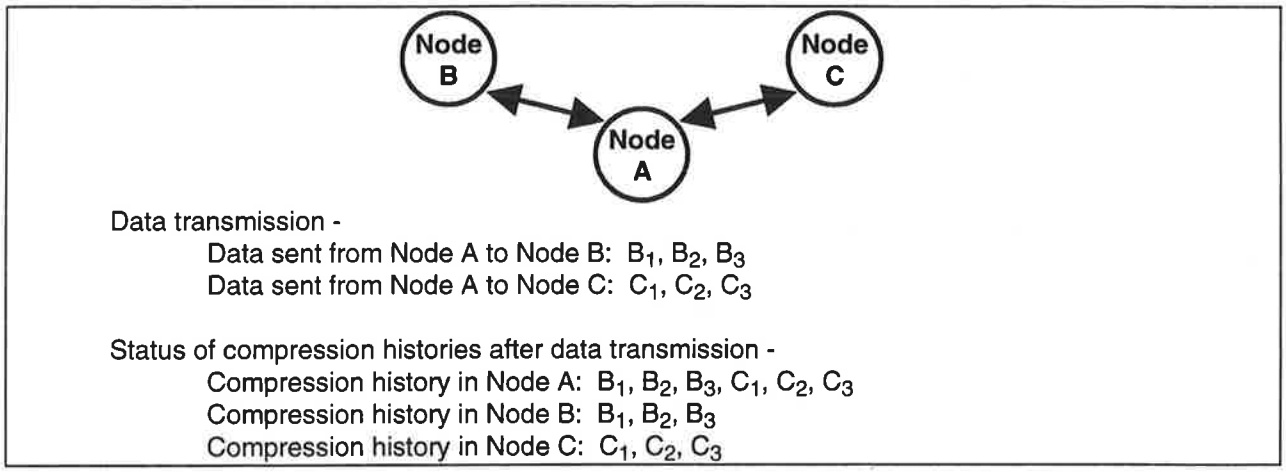

Figure 7. Compression histories not synchronized due to single history and multiple destinations

One technique to solve this problem is to reset the compression history at the beginning of each packet. As described previously, this would result in poor a compression ratio.

Another technique is to support multiple compression histories. A node can maintain a separate compression history for each node that it must communicate with. A node that communicates with only one other node (or an access point) need only maintain a single compression history. A node that communicates with multiple nodes would maintain multiple compression histories. This is shown in Figure 8.

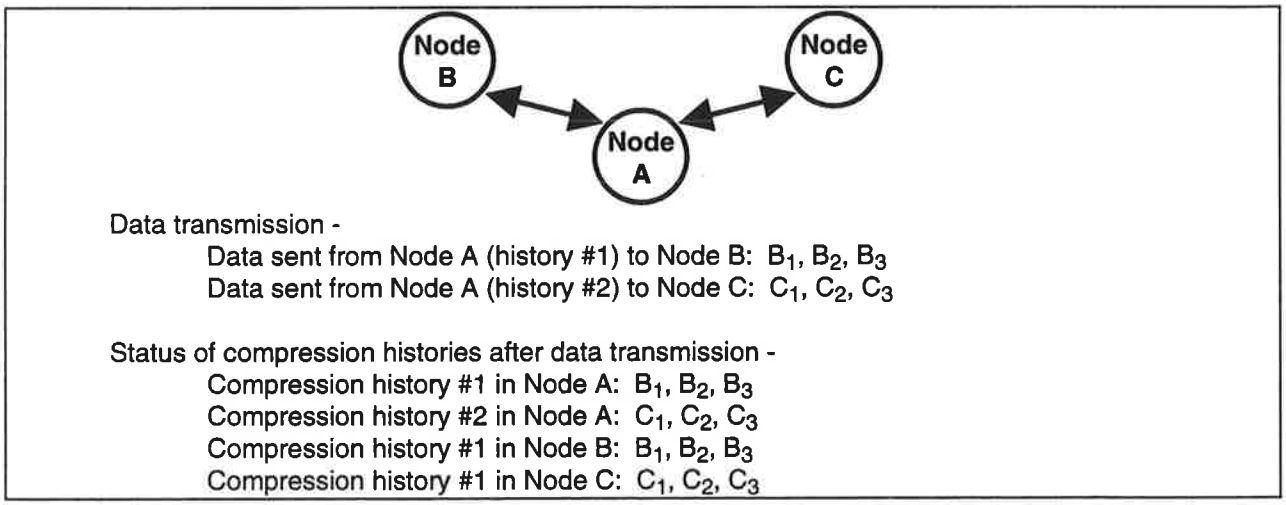

Figure 8. Multiple compression histories for multi-node communication

Multiple compression histories also eliminates compression history *dilution*. Dilution occurs when a single compression history is used for more than one type of data. When switching from one data type to another, the instantaneous compression ratio will drop until the compression history becomes filled with the new type of data. This is shown in Figure 9. Using multiple

compression histories allows a separate compression history to be used for each virtual connection and its data type.

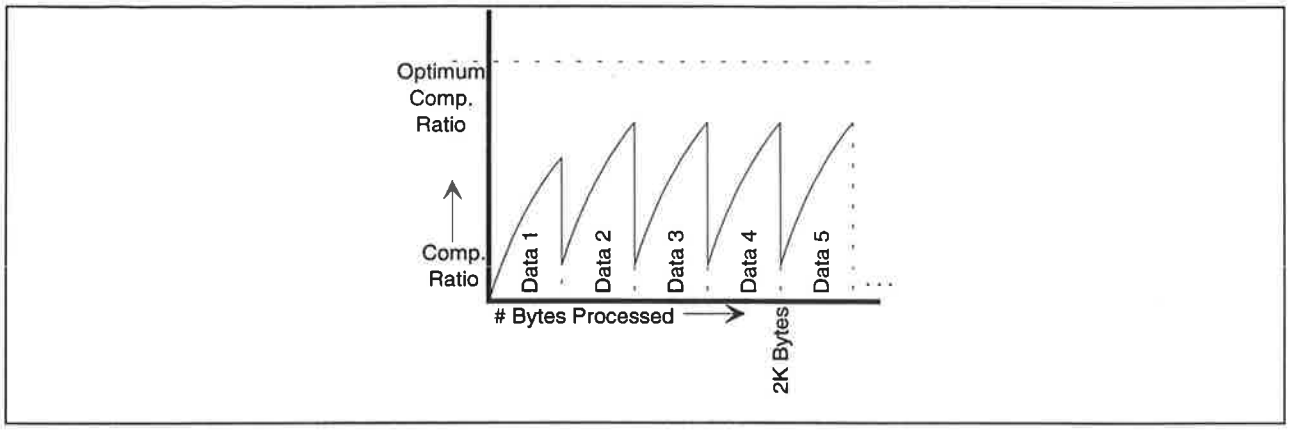

Figure 9. Compression ratio due to dilution

Critical Capacity

Data compression can improve the total throughput of a wireless LAN. Not only is the data transfer rate increased, but the critical capacity point is pushed out, as shown in Figure 10.

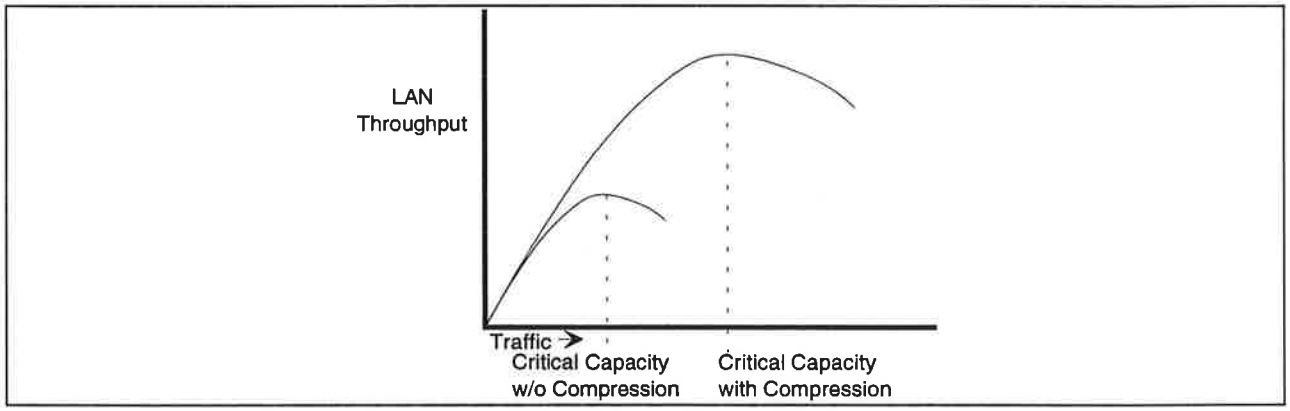

Figure 10. Increased LAN throughput with data compression

Conclusion

The use of data compression in a wireless LAN will improve the performance of a wireless LAN. The MAC layer should be designed with data compression in mind. The issues of compression block size and the support of multiple compression histories should be resolved.