DESIGN AND IMPLEMENT A TRANSPARENT INTERNET PROTOCOL PACKETS COMPRESSION SYSTEM

Suha M. Hadi ,*Firas R. Barjas
Departement of Information and Communication Engineering , College of Al-Khawarizmi Engineering, University of Baghdad .baghdad -Iraq
*Asiacell communication company, Sulaymani –Iraq

Abstract

The data compression process is an important aspect that should be given a good attention when dealing with slow computer networks. This paper introduces the design and implementation of a transparent compression system that can be utilized by Microsoft Windows to compress data exchanged among the computers through a Local Area Network (LAN). On each machine, this system transparently intercepts outgoing IP packets and compresses them. As well as it will transparently intercept incoming compressed IP packets and decompresses them.

تصميم وتنفيذ نظام شفّاف لضغطِ رُزَمِ بروتوكولات الانترنيت

سها محمد هادي، * فراس رسمي برجس
قسم هندسة المعلومات والاتصالات ، كلية الهندسة الخوارزمي ، جامعة بغداد . بغداد– العراق
*شركة اسياسيل للاتصالات، السليمانية– العراق

الخلاصة

إنّ الضغطَ من السمات المهمةُ التي يَجِبُ أَنْ تُعطى إنتباه جيدا عندما نتعاملُ مع شبكاتِ الحاسوبِ البطيئةِ. يتناول هذا البحث تصميمَ وتطبيقَ نظام ضغطِ شفّافِ لأُستَخدامه بنوافذِ مايكروسوفتِ لضَغْط البياناتِ المتَبادلةَ بين الكومبيترات من خلال الشبكات المحليَّة (LAN). حيث يقوم النظام وبشكل شفاف باعتراض الرزمَ الخارجةَ من كل جهاز ومن ثم َضغطُها. كماً ويقوم من جهة اخرى بأَعتراضُ الرُزَمَ المضغوطة الواردة من اي جهاز لَغرَض فتح ضغطها.

1.Introduction

It is very clear that computer networks have become an essential part of every day life. However, despite the fact that very fast networks do exist, many people still use slow networks [1]. For such networks that suffer from limited bandwidth, compression is very important. It reduces the redundant information being transferred and leads to much better network utilization. The compression of network traffic has other advantages. Since it reduces the amount of data being transferred over the network, it is useful for networks that use noisy channels. The probability of receiving damaged frames due to noise may be decreased by using the compression process. Moreover, compression can reduce the cost of data transmission in the networks, in which the cost of data transmission depends on the amount of data transferred. A compression algorithm takes an input X and generates a representation X_C that hopefully requires fewer bits. There is a reconstruction algorithm that operates on the compressed representation X_C to

generate the reconstructed presentation Y. Based on the requirements of reconstruction, data compression schemes can be divided into two broad classes [2]:

1- Lossless algorithms, which can reconstruct the original message exactly from the compressed message. As an examples of this class include Huffman coding, LZW, and ZIP.
2- Lossy algorithms, which can only reconstruct an approximation of the original message. As an examples of this class the vector quantization, predictive coding, and fractal compression.

The Lossless algorithms are typically used for text, while the lossy algorithm is used for images and sound where a little bit of loss in resolution is often undetectable, or at least acceptable. It is used in an abstract sense; however, that does not mean random lost pixels, but instead means loss of a quantity such as a frequency component, or perhaps loss of noise [3]. The data that will be compressed by the proposed compression system must be exactly and perfectly reproduced after decompression at its destination; therefore a lossless compression algorithm must be used here. So, the ZIP compression algorithm was selected in this paper because of its efficiency and for the availability of its source code.

2.The system specifications

The following points shed the light on the most important specifications of the proposed system:

1- The system must automatically compress the payloads of outbound Internet Protocol (IP) packets and decompress the payloads of inbound IP packets.
2- The processing of the system (compression and decompression) must be transparent to all networking applications. In other words, there is no need to change software on the computer that uses the compression system.
3- The system has been designed to process the packets in a way that will not conflict with the work of routers, which may exist in the path of the packets during their transit. For this reason, the compression system must not compress the IP headers of processed IP packets. It must as well update the total length field of the IP header of each processed IP packet to reflect the new size of the packet after compression. Updating the total length field of the IP header requires of course recalculation of the IP header Checksum field.
4- The system should provide a friendly Graphical User Interface (GUI), by which the network administrator can control and monitor the operation of the system.
5- The system works under Microsoft Windows 2000 operating system and later versions.
6- The compression system is aimed to work on Fast Ethernet networks.

To achieve the required transparency, the system utilizes the NDIS (Network Driver Interface Specifications) Intermediate Miniport (IM) driver to implement compression and decompression of IP packets. This driver is located between the Logical Link Control (LLC) and the Medium Access Control (MAC) sub layers of the data link layer [4]. The NDIS IM driver is well documented, and because of its location in Windows machine network architecture, no outbound or inbound packet bypasses it. Thus, it best suits the system in question. (Figure 1). shows NDIS driver location in Windows network architecture.

3.The proposed system architecture

The system will be consists of four main modules:

1- The Compressor module.
2- The De-compressor module.
3- The Interface module.
٤-The Admin-Panel module

(Figure 2). will illustrate these four modules and the interaction between them (The system modules are shown in the bold color).

The first module that is the Compressor module will compresses the outbound packet payloads using the ZIP compression algorithm. If the data carried in a packet payload is highly pre-compressed (it could belong to a highly compressed file being transferred), this data will have very low redundancy. If such data is compressed by the compressor module, its size may increase. This will negatively affect the overall compression ratio. In addition, if such data is being sent inside a large packet, there will be a possibility that the size of the packet after compression be more than the Maximum Transmission Unit (MTU) size (MTU of Ethernet is 1500 bytes) [5]. Packets larger than the MTU

are not sent by NICs. To solve this problem, there are two solutions:

A-After the payload of each IP packet is compressed, its size is checked. If the resulting IP packet has a size larger than the MTU, will be fragmented to a number of fragments each of which has a size less or equal to the MTU size, thus preventing the NIC from dropping them. This solution will not be used, because it does not resolve the negative impact on the overall compression ratio. Many file extensions whose data are already compressed using lossy or lossless compression algorithms may be exchanged among computers in a network (ex., *.jpg, *.gif, *.zip, *.rar, …etc), and that compression of packets holding such data will frequently increases their sizes[7], and thus negatively affects the compression ratio. Moreover, this solution apparently increases the number of packets being transferred (and eventually processed by intermediate routers if any), along with the overhead introduced by their IP and Ethernet headers

B-The other solution will involves checking the IP packet size resulting after compression. If the resulting size is larger than the original size, the compression process will be useless, and so the packet is sent without compression. This approach requires to include a one-byte flag (from now on will be called compression flag), which indicates to the receiving compression system whether the packet is compressed or not. The overhead implied by the inclusion of this one byte flag in all outbound packets is much less than that implied by the first solution. This was tested practically. The compression flag is inserted directly after the IP header. It is worthy mentioning that this solution requires setting the MTU of each NIC in a host that uses this compression system to 1499 bytes instead of the default MTU value (1500 bytes). This is necessary, because a highly pre-compressed outbound IP packet will have a size 1 byte more than its original size after being processed by the compression system at the worst case. If such a packet has a size equal to 1500 bytes, its size will be 1501 bytes after being processed by the compression system. Hence, such a packet size will cause the NIC to drop the packet, because it will be larger than the MTU size supported by the Ethernet NICs. The task of setting the MTU of each NIC to 1499 bytes is performed by the Admin-Panel Module, as will be shown later.

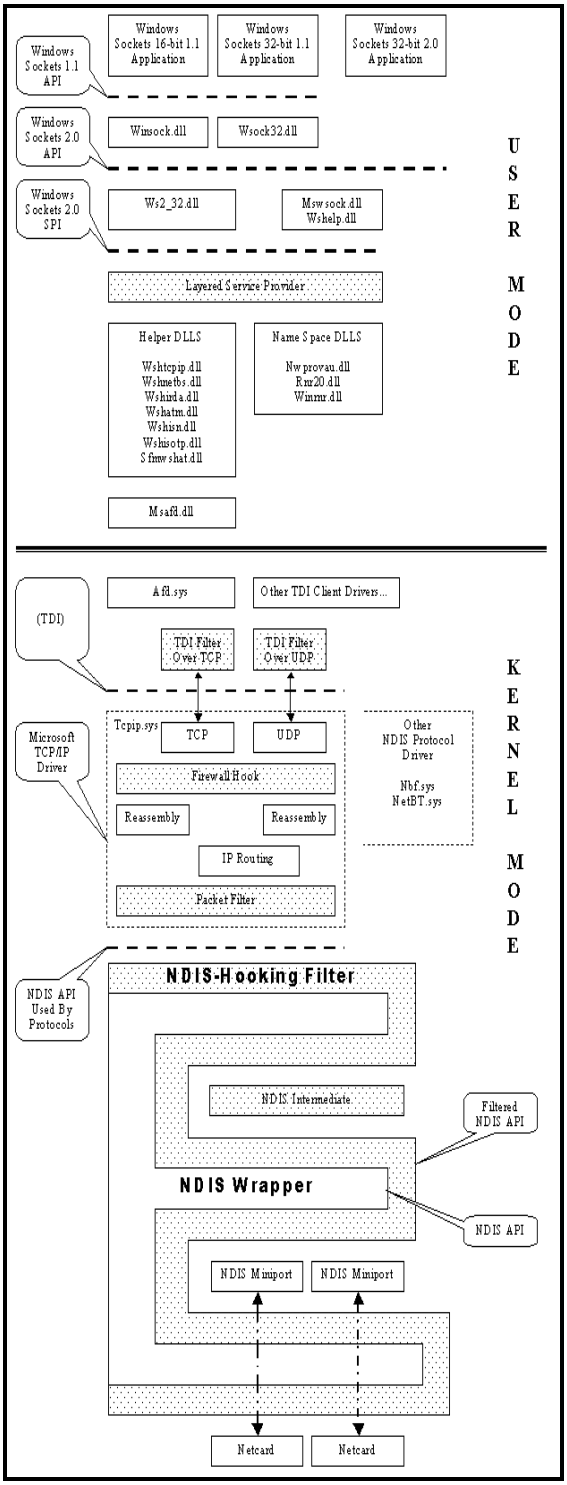

Figure 1:The NDIS Driver Location [6]

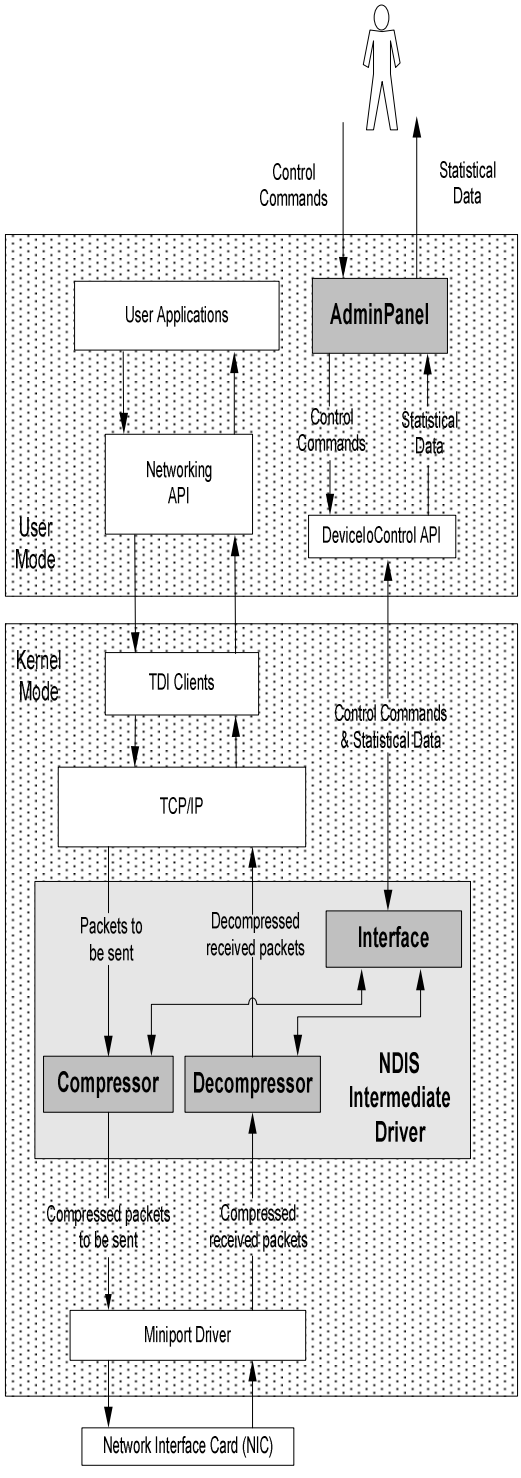

Figure 2: The probosed Compression System Architecture

The compressor module intercepts each outbound packet transmitted by the upper transport protocols, and processes it according to the flow chart shown in(Figure 3).While the De-compression module intercepts incoming IP packets and decompresses their payloads if they are compressed. When the **De-compressor module** receives a packet from an underlying miniport, it processes the packet as specified by the flow chart show in (Figure 4). The third module which is the **Interface module** will exports the interface functions that can be called by the Admin-Panel module to control the start or stop of compression and to get some compression related statistics. These statistics consist of the accumulated number of compressed and decompressed bytes that have been sent or received. The Interface module works within the used NDIS intermediate driver. To enable user mode applications to access the Interface module, the NDIS intermediate driver registers a device object for itself when it is loaded into the memory. The fourth Module which is The **Admin-Panel** module is a Win32 application, through which the administrator can start or stop compression and can view the statistical information provided by the Interface module. The Admin-Panel calculates and displays the average compression ratio as well. The compression ratio is calculated using the following formula:

Compression Ratio = (U - C) / U * 100 %

Where,

U: Is the accumulated number of uncompressed payloads bytes

C: Is the accumulated size of compressed payloads bytes

The Admin-Panel communicates with the Interface module periodically to get updated statistics using the Device-Io-Control API. The Device-Io-Control API sends a control code directly to a specified device driver, causing the corresponding device to perform the corresponding operation [7].

4-The System Implementation

The NDIS intermediate driver, which encompasses the Compressor, De-compressor and Interface modules, was written in C Language using NDIS support routines. The driver was built

using Microsoft Driver Development Kit 2000 (DDK2000).

The Admin Panel was built using Microsoft Visual C++ 6. Simply, it consists of one window that can be used as a panel to control and monitor the work of the compression system. The snapshot of the Admin Panel single window is shown in (Figure 5).

5.Performance results

The compression system was installed on 3 machines, which were connected as 100 BaseTX Fast Ethernet by using Linksys 10/100 dual speed 16-port stackable switch. The specifications of each machine were as follows:

- Processor: Intel Pentium III, 866 MHz
- Physical Memory: 256 Mbytes
- Hard Disk: Western Digital Caviar, 20 Gbytes
- LAN Card: Realtek RTL8139(A)
- Operating System: Windows XP Service Pack 2

Four tests were performed on two machines (from now on, they will be called A and B): Compression ratio test, processor usage test, memory usage test and operability with routers test.

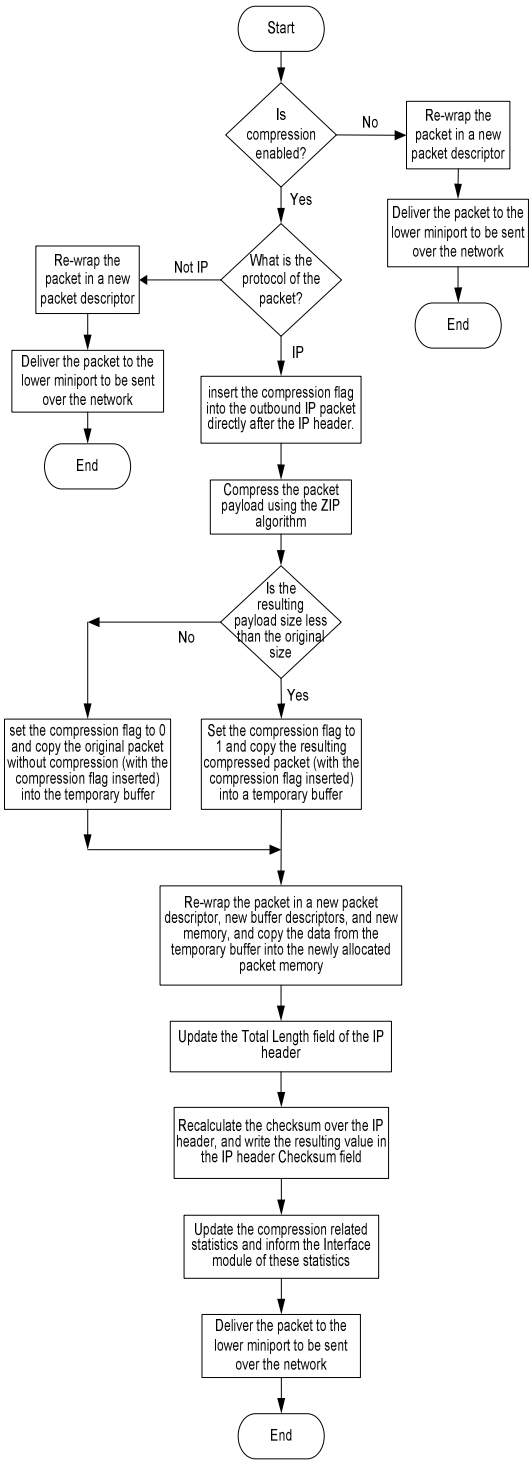

Figure 3: Outbound Packets Processing by the Compressor Module

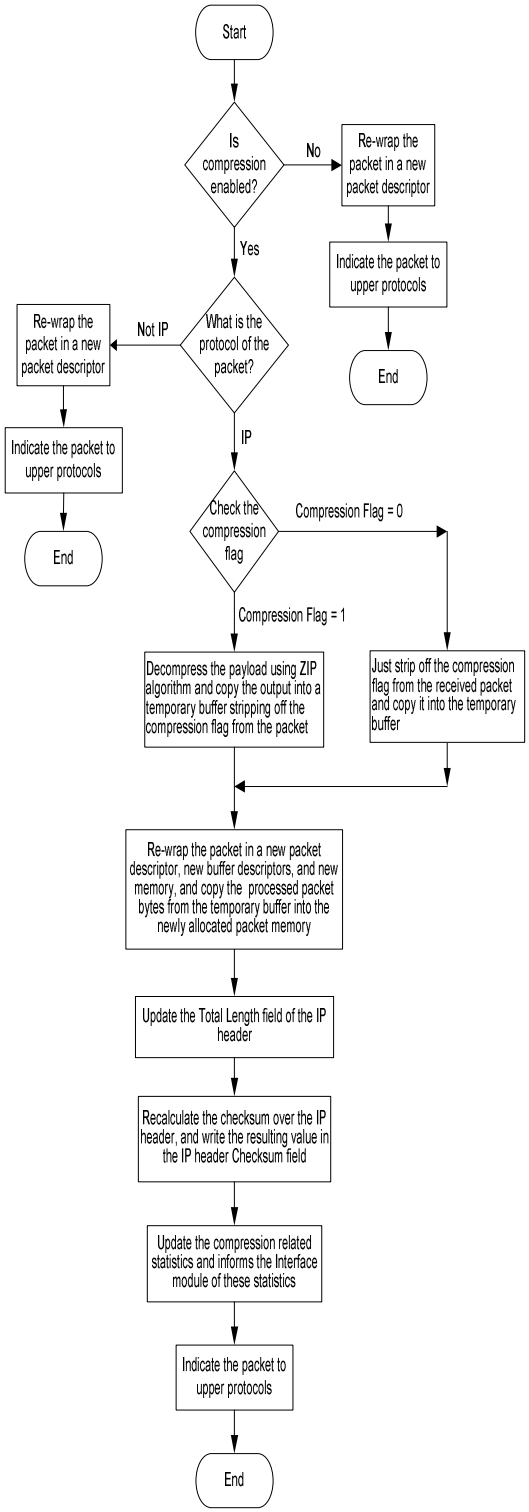

Figure 4: Inbound Packets Processing by the De-compressor Module

6. The compression ratio tests

In the beginning, a collection of files was located in A, while B was used to get all the files from A. The collection of files was (577) Mbytes, and it including many file formats with different degrees of redundancy (bitmap images, Microsoft Office files, executable files, compressed RAR archives, compressed ZIP archives, and real media - .rm - movies). During the transfer of this collection of files, the average compression ratio was measured using the Admin Panel module, and it was (14.374%).
The same test was repeated again but with a 1(88) MByte collection of bitmap images. The compression ratio was (64.545%). As can be noticed, the compression ratio this time is larger than the compression ratio of the first test, because bitmap images have a high degree of redundancy. On the end, the compression ratio was measured again during issuing the following command from machine B to continuously ping machine A with ICMP (Internet Control Message Protocol) messages, which have 1450 bytes in their ICMP payloads : **Ping A_IP_ADDRESS – l 1450 - t** Where (**A_IP_ADDRESS**) is the IP Address of machine A. In this test, the compression ratio was measured to be (96.365%). Since the transferred ICMP messages hold sequenced alphabetical characters that are repeated many times, they have very high redundancy. Thus, compression of such messages will result in a high compression ratio as can apparently be noticed.

The three tests show the importance of the developed compression system. Suppose that machines A and B are connected by a slow network link (such as a dial-up line). Let's say it can support up to (56) Kbytes/sec. Suppose also that the two machines do not use the compression system. If the (188) MByte collection of bitmap images is transferred from A to B, the time required for this transfer will approximately be calculated according to this equation (ignoring the overhead imposed by the headers required to transfer the files, i.e. IP, TCP …etc):

Time = 188 * 1024 / 56 = 3437.714 seconds (i.e. approximately 57 minutes)

If machines A and B are using the compression system, and the same collection of bitmap images is transferred using the same slow link, the compression ratio will approximately equal the

compression ratio measured above (i.e. 64.545%) as the compression ratio is affected only by the used compression algorithm and the nature of the data being compressed. The amount of compressed transferred bytes will be approximately:

C = 188 - 0.64545 * 188 = 66.6554 Mbytes

The time required to transfer this amount of data would roughly be (ignoring the overhead imposed by the headers required to transfer the files, i.e. IP, TCP …etc):

Time = 66.6554 * 1024 / 56 = 1218.8416 seconds (i.e. approximately 20 minutes).

As shown by the previous example, the advantages of the developed compression system by providing better network utilization and by reducing the time required to transfer data using slow networks are clearly explained

Figure 5: the GUI of the Admin Panel Module

7.Conclusions

Throughout the design and implementation of the compression system, a number of points that can be useful for the development of similar systems were concluded. They may be summarized by the following points:

1- Compression is an important aspect that should be given a good attention when using computer networks that use slow links. It can significantly reduce the time required for transferring data among the computers of a network that uses slow links. Moreover, compression increases network utilization by reducing the redundancy inherent in the transferred data.

2- The best way to provide compression for transferred data is by compressing outbound data and decompressing inbound data transparently (i.e. in a way transparent to end users). This saves much labor and let users continue to use their usual network applications without any change. Windows network drivers can be used to provide transparency for applications like compression and encryption. Specially, NDIS IM driver which is the best and the most efficient Windows network drivers that are documented.

3- If the data carried in a packet payload is highly pre-compressed, this data will have very low redundancy. If such data is compressed by a compression system, its size may increase. This will negatively affect the overall compression ratio. In addition, if such data is being sent inside a large packet, there will be a possibility that the size of the packet after compression will be more than the MTU size, and thus it will not be sent by the underlying NIC. Hence, the best way to solve this problem is by sending such packet without compression. This requires the inclusion of a flag in the packet to distinguish those packets, which are compressed by the compression system from those, which are not.

8.Refrences

1. Munteanu,D and Williamson,G. **2004**. An FPGA-based Network Processor for IP Packet Compression, Department of Computer Science, University of Calgary, Canada.
2. Blelloch, G.E. **2001.** Introduction to Data Compression, Computer Science Department, Carnegie Mellon University,.
3. Microsoft Corporation, April **2001.** Microsoft Development Network, Windows NT Workstation 4.0 Resource Kit, "Kernel Mode and User Mode".
4. Hua,W ; Ohlund, J. and Butterklee,B. May **1999**. Unraveling the Mysteries of Writing a Winsock 2 Layered Service

Provider, Microsoft Systems Journal, Microsoft Corporation,

5. Divine,T.F. December **2002**. Packet Filtering Techniques, available at: http://www.pcausa.com/resources/winpktfilter.htm,Printing Communications Assoc., Inc.,
6. Smirnov,V.V. **2003**. Firewall for Windows 9x/ME/NT/2000/XP, available at:Htt://www.ntndis.com/articles/firewalleng.htm, NT Kernel Resources
7. Microsoft Corporation, **2000**. Microsoft Windows 2000 Driver Development Kit, Network Drivers.
8. Hornig,C. April **1984**. A Standard for the Transmission of IP Datagrams over Ethernet Networks, Request For Comments (RFC) 894.