

Cisco IOS Data Compression

Abstract

Many strategies and techniques exist for optimizing traffic over wide-area networks (WANs), including priority queues, access lists, and filters. One of the more effective methods, however, is to reduce the amount of data by compressing it over the network. Several data compression methods are available. Some methods reduce packet header size, while others condense data within the packets by use of sophisticated algorithms. The type of network link must also be considered when evaluating and selecting a data compression method, as well as where the data compression should take place—within the router or in an external device. These features must also have the flexibility to respond to new technologies and requirements with the Cisco Internetwork Operating System (Cisco IOS™) software. This is an update to a white paper written by Kevin Dickson which was previously posted: Dec 19, 1995.

Introduction

Local-area networks (LANs) are connected to each other and to corporate backbone networks via wide-area network (WAN) links. Today, we see several trends placing demands on these links. One trend is the move away from personal computer-based applications to shared applications on a LAN. LAN-based applications tend to generate varying types of traffic across the WAN links. Typical are bulk transfers of files and transaction-oriented traffic from interactive applications. Information transfer services such as electronic mail and file server applications also place increasing demands on link bandwidth. Moreover, these applications differ in the way their data appears as it traverses the network.

Link speed is another consideration with today's WANs. A decade ago, 2400 bits per second (bps) was considered a high transmission speed. Today, transmission rates of 64 kbps are common. Traditional point-to-point leased lines and the X.25 encapsulation format have been most commonly used in WAN transmission service. New services such as Integrated Services Digital Network (ISDN) at Primary Rate Interface (PRI) and Basic Rate Interface (BRI) rates, Frame Relay, Switched Multimegabyte Data Service (SMDS), and Asynchronous Transfer Mode (ATM) are also available and address the need for increased bandwidth. These transmission links can be very expensive by themselves. In fact, in an enterprise-wide network, the cost of transmission links can consume a significantly large portion of the operating cost of the network.

A comparison of the leased-line service costs for U.S., European, and transatlantic links also demonstrates the high cost of bandwidth in different parts of the world. Because the infrastructures of developing countries, with their growing demands for data communication, are not usually large or well developed, networks must be able to use WAN links in the most efficient way possible. Reducing the cost of WAN transmission is the goal of Cisco's WAN technologies.

The Cisco IOS™ software offers a number of features that optimize WAN links to ease the WAN bandwidth bottleneck. Among these features are priority output queuing, custom queuing, access lists, Novell static Service Advertisement Protocol (SAP) tables, and Novell IPX SAP filters. However, one of the more effective methods of WAN optimization is still data compression.

Data compression can serve to significantly decrease the size of a frame, reducing the time the data takes to travel across the network. Data compression works by providing a coding scheme at each end of a transmission link. These coding schemes allow characters to be removed from the frames of data at the sending side of the link and then replaced correctly at the receiving side. Because the condensed frames take up less bandwidth, greater numbers of them can be transmitted per unit of time.

This paper describes data compression solutions. It begins with a technical explanation of how data compression works, and then examines five data compression solutions. It goes on to identify network characteristics to consider when implementing data compression in the internetwork, and concludes with a summary of data compression solutions from Cisco Systems, Inc.[®]

How Data Compression Works

The basic function of data compression is to reduce the size of a frame of data to be transmitted over a network link. There are several ways this can be done.

Data compression schemes used for transmitting voice and image data are referred to as lossy or nonreversible compression. This type of compression allows for some degradation or loss of data in return for the significantly greater compression and lower bandwidth needed to handle the increased size of these types of transmissions. The Cisco IOS software supports Apple QuickTime Conferencing, a modular software architecture that in turn supports video teleconferencing standards such as Joint Photographic Experts Group (JPEG) and Moving Pictures Experts Group (MPEG). These standards are based on encoding schemes that are sensitive to packet loss and serve to smooth out the effects of compression on voice and image data. See the Cisco[®] publication Cisco IOS Support for Apple QuickTime Conferencing, dated March 1995, for more information about video compression. (Check on the validity of the above technology)

Data compression schemes used in internetworking devices are referred to as lossless compression algorithms. These schemes reproduce the original bit streams exactly, with no degradation or loss, a feature required by routers and other devices to transport data across the network. Lossless compression algorithms use two basic types of encoding techniques: statistical and dictionary.

Statistical versus Dictionary Compression

Statistical compression, which uses a fixed, usually non-adaptive encoding method, is best applied to a single application where the data is relatively consistent and predictable. Because the traffic on internetworks is neither consistent nor predictable, statistical algorithms are, in general, not suitable for encoding data for compression on routers.

An example of dictionary compression is the Lempel-Ziv algorithm. This algorithm is based on a dynamically encoded dictionary that replaces a continuous stream of characters with codes. The symbols represented by the codes are stored in memory in a dictionary-style list. Because the relationship between a code and the original symbol varies as the data varies, this approach is more responsive to the variations in the data. This flexibility is particularly important for LAN data, because many different applications can be transmitting over the WAN at any one time. In addition, as the data varies, the dictionary changes to accommodate and adapt to the varying needs of the traffic. While small dictionaries of 2000 to 32,000 bytes are typical, compression ratios can be optimized by the use of larger dictionaries. Compression ratios can be a measure of the efficiency of the compression algorithm and is expressed as a ratio, $x:1$, where "x" is the number of input bytes divided by the number of output bytes. Variations of the Lempel-Ziv algorithm are used in many popular compression programs, such as Stac (LZS), ZIP and the UNIX compress utility.

Cisco's STAC and Predictor Compression Algorithms

Cisco internetworking devices use the STAC (LZS) and Predictor data compression algorithms. STAC (LZS) was developed by STAC Electronics, now known as Hi/fn, Inc., and is based on the Lempel-Ziv compression algorithm. The Cisco IOS software uses an optimized version of LZS that provides good compression ratios but requires many CPU cycles to perform compression. LZS is available in Cisco's Link Access Procedure, Balanced (LAPB), HDLC, X.25, and Frame Relay data compression solutions. FRF.9 and IPComp use the LZS compression algorithm. A brief description of IPComp can be found in the section entitled "Compression of Encrypted Data".

LZS searches the input data stream for redundant strings and replaces them with what is called a token which turns out to be shorter in length than the original redundant data string. LZS creates tables or dictionaries of these string matches and tokens that consist of pointers into the previous

data stream. This dictionary is built and used to begin replacing the redundant strings found in the new data streams.

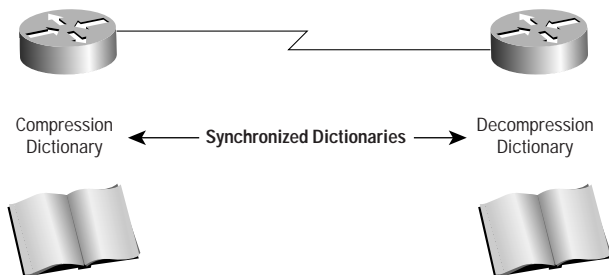
The Predictor compression algorithm tries to predict the next sequence of characters in the data stream by using an index to look up a sequence in the compression dictionary. It then examines the next sequence in the data stream to see if it matches. If so, that sequence replaces the looked-up sequence in the dictionary. If not, the algorithm locates the next character sequence in the index and the process begins again. The index updates itself by hashing a few of the most recent character sequences from the input stream.

The Predictor data compression algorithm was obtained from the public domain and optimized by Cisco engineers. When compared with LZS, it makes more efficient use of CPU cycles but requires more memory. Both LZS and Predictor data compression algorithms can be used with Cisco's Point-to-Point Protocol (PPP) or LAPB protocol.

How Routers Use Compression

Dictionary-based algorithms are used in one of two modes of operation: continuous mode and packet mode. Continuous-mode operation monitors a continuous stream of characters across packet boundaries to create and maintain the dictionary. The continuous stream of data can consist of multiple network protocol packets such as IP and DECnet. Continuous-mode operation requires that the dictionaries on the compression side and the decompression side be kept synchronized through the use of a reliable data link mechanism such as X.25 or reliable-mode PPP, or through a sequencing mode with out-of-band signaling. Figure 1 illustrates synchronized dictionaries over a LAPB link.

Figure 1 Dictionary Synchronization for Continuous-Mode Operation



Synchronization ensures that packet errors and loss do not cause the dictionaries to diverge. In fact, a "golden rule" states that the decompressor cannot be allowed to see a frame that is corrupted or out of order. Otherwise, subsequent packets would be decompressed inaccurately, and both dictionaries would need to be flushed and outstanding packets discarded to recover.

Although packet mode operation also monitors a continuous stream of characters to create and maintain its dictionary, it limits the stream to a single network packet. Packet mode algorithms also require dictionaries but, because the stream used to create the dictionary is limited to a single network packet, the dictionaries need only be synchronized within the packet boundaries.

Some data can not be compressed and in fact, will expand as a result of trying to compress it. Examples of data which will not compress are data which has been encrypted or data that has already been compressed

Standards

This paper earlier identified the standards supported by the Apple QuickTime Conferencing architecture. Efforts to develop industry standards for voice and video compression are also being undertaken by the Telecommunications Standardization Sector of the International Telecommunication Union (ITU-T, formerly known as CCITT) and the American National Standards Institute (ANSI). These efforts have not yet been translated into a synchronous compression standard for use in routers. Work is progressing on standardizing asynchronous compression techniques for use with modems and communications servers, and may soon start for synchronous techniques as well. The Internet Engineering Task Force (IETF), working through the PPP Working Group, has defined a compression negotiation mechanism for use by routers and other devices using PPP called Compression Control Protocol (CCP). Although this negotiation mechanism will not be a compression standard, it will allow users to achieve interoperability between devices using the same compression algorithm.

Cisco's Data Compression Solutions

The Cisco IOS currently provides these data compression solutions:

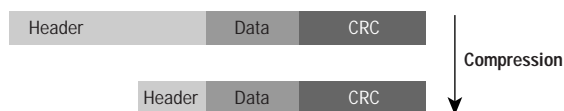
- Van Jacobson Header Compression for TCP/IP, which conforms to RFC 1144.
- Per-interface compression (also called link compression), which has one set of dictionaries per hardware link.
- Per-virtual circuit compression (also called payload compression), which has one set of dictionaries per compressed virtual circuit.
- *Microsoft Point to Point Compression*—MPPC which is RFC 2118 and is a per PPP connection compression algorithm
- *Frame Relay Payload Compression*—FRF.9 is per DLCI

These features are explained in the following sections, and are summarized at the end of this paper.

TCP/IP Header Compression

The Cisco IOS header compression strategy subscribes to the Van Jacobson Algorithm defined in RFC 1144. It is protocol specific and effective on TCP/IP traffic consisting of small packets with few bytes of data such as Telnet. TCP/IP header compression lowers the overhead generated by the disproportionately large TCP/IP headers as they are transmitted across the WAN. Cisco's header compression product supports X.25, Frame Relay, and dial-on-demand WAN link protocols. Transaction-oriented applications such as DEC LAT, Telnet, rlogin, XWindows, and acknowledgment packets typically can take best advantage of this type of compression. (See Figure 2.)

Figure 2 TCP/IP Header Compression



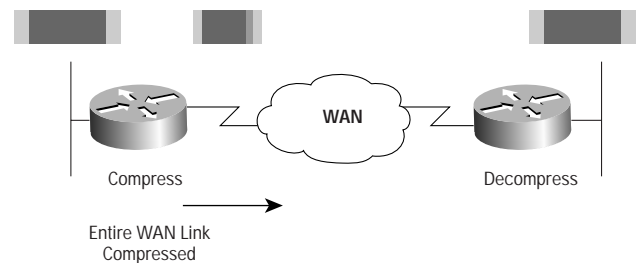
Because of the processing overhead, header compression is generally used at 64 kbps, not at the higher speeds now becoming critical in LAN-to-WAN communication. Header compression can produce varying throughput improvements across low-speed lines depending on line rate. For example, a 50 percent throughput improvement can be achieved with Telnet traffic on a 64-kbps leased line.

Per-Interface Compression

To handle larger packets, support higher data rates, and improve performance across multiple protocols on a LAN, compression can be applied to the entire data stream to be transported across the WAN. This type of compression, called per-interface compression, compresses the entire WAN link as if it were one application. Unlike header compression, per-interface compression is protocol independent. The per-interface compression algorithm uses STAC or Predictor to compress the traffic and then encapsulates the compressed traffic in another link layer such as PPP or LAPB to ensure error correction and packet sequencing. (See Figure 3.)

The PPP software includes the Compression Control Protocol (CCP) which negotiates data compression for PPP links as defined in RFC 1548. CCP compression negotiation supports both STAC, MPPC, Predictor and several other compression algorithms.

Figure 3 Point-to-Point Circuits on a Leased Line



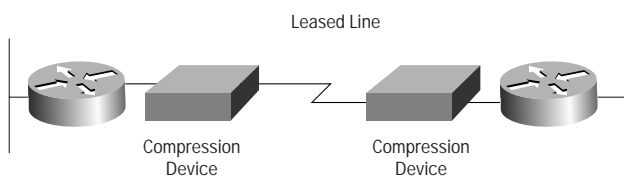
The very definition of per-interface compression makes it a point-to-point-only solution for use in services such as leased line or ISDN. The complete packet—header and data payload—is compressed, and the switching information in the header is not available for WAN switching networks.

The use of per-interface compression imposes another major constraint on network design in multihop topologies. In such a network, the data must be compressed every time it exits a router and decompressed every time it enters a router. The more routers in the path, the greater the number of compression-decompression processes. These processes add delay at each router and create potential for significant delay in the network application. The best applications for per-interface compression are environments where the customer knows that the application is a point-to-point connection with a limited hop path. External devices provide per-interface compression and allow the customer to deploy compression at the appropriate locations without burdening router performance.

Per-interface compression can be applied in router networks in these ways:

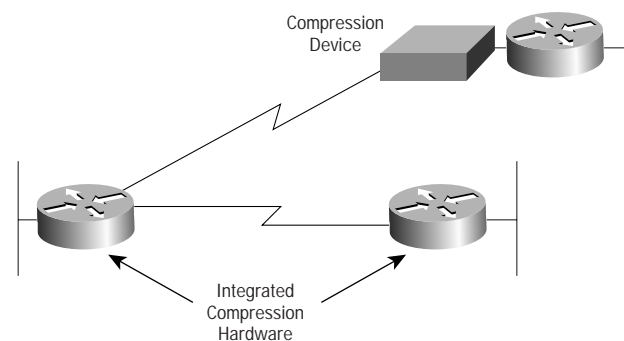
- **An external compression device**—The device acts as a black box, taking in serial data from the router, compressing the data, and sending compressed data out into the WAN (see Figure 4). While readily available and simple to implement, this solution is usually not preferred by network managers. This is because they believe that the integration process is less expensive, and that external devices are ill-suited for remote sites because no one is usually available to manage and configure them.

Figure 4 External Device for Per-Interface Compression



- **Integrated Compression Hardware**—This method involves integrating technology on an external device that will fit on a port adapter, applique, or compression card in the router. After serial line encapsulation, the router performs the compression on the applique or card (see Figure 5). The advantage is better compression performance without the burden of a separate device. The disadvantage is that the integration process is expensive, and the solution requires router users to purchase new hardware.

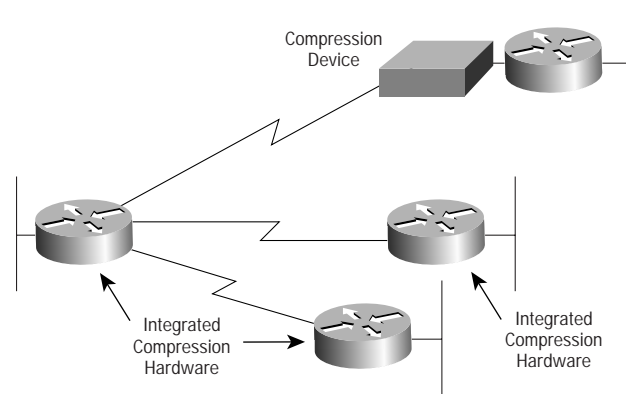
Figure 5 Integrated Compression Hardware



- **Integrated Compression Software**—This method allows compression software to be integrated into the router software and to use the same serial interfaces already installed on the router (see Figure 6). The encapsulation routine is enhanced to perform per-interface compression after the initial encapsulation for the serial line. The router must have available sufficient CPU for compression and RAM for the compression dictionary. This method is ideal

for access routers where low-speed line rates must be supported. Dictionary requirements vary according to the algorithm used for compression.

Figure 6 Integrated Compression Software



Per-Virtual Circuit Compression

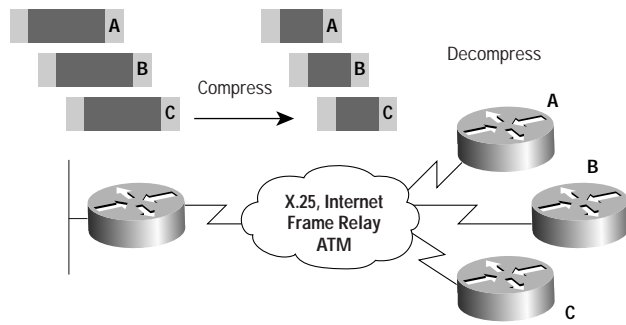
Per-virtual circuit compression is the required method of compression for operation across virtual network WAN services. It is supported over X.25 (Predictor or STAC) and Frame Relay (STAC) WAN.

Because the header information is left unchanged during per-virtual circuit compression, packets still can be switched through a WAN packet network and transmitted through a router network. Thus, compression can be applied while the network takes advantage of the typically lower tariffs of public packet data networks. When designing an internetwork, the customer cannot assume that an application will be going over point-to-point lines only. As a result, the software must be intelligent enough to ensure that the application packet headers are maintained, and that the router can perform compression along with the WAN encapsulation.

The ability to preserve the header makes per-virtual circuit compression a good choice for routers that provide WAN packet services and use a single interface with multiple virtual circuits destined for different locations.

The use of packet services by routers dictates that per-virtual circuit compression cannot realistically rely on continuous-mode compression algorithms, because each virtual circuit requires its own dictionary. When each router contains many virtual circuits, a large amount of memory is needed for their configuration. Alternatively, a packet mode compression algorithm uses fewer dictionaries, and therefore less memory, to compress the information.

Figure 7 Packets Before and After Per-Virtual Circuit Compression



Compression Before versus After Encapsulation

Implementing per-virtual circuit compression on the router involves hardware and software considerations and can be done in two different ways, with different benefits to the user. The first method applies compression to the data payload after WAN encapsulation on the serial interface. This process compresses the size of the payload and the overall size of the packet but does not change the number of WAN packets. It can be implemented within the existing router software architecture by adding software modules to provide the appropriate compression algorithms. This compression method can be very useful for WAN services such as Frame Relay or SMDS, which are not sensitive to packet count. For packet networks that charge by packet, such as X.25, it may not provide significant cost benefits. External compression devices supporting the appropriate WAN interface can also provide this capability for higher-performance, higher-speed links, and also higher cost.

The second implementation method compresses the network protocol traffic before WAN serial encapsulation takes place. As a result, packet size remains at the standard maximum transmission unit (MTU) for that service or serial interface, while the number of packets sent over the network decreases. This method is particularly useful for connection to X.25-based networks and other WAN packet services where charges are based on the number of packets transmitted over the WAN.

Compression of Encryption Data

Today, compression is a layer 2 function and encryption occurs at layer 3. When a data stream is encrypted by the client application, it is then passed onto the router for routing and/or compression services. When the compression engine receives this encrypted data stream which by definition has no repetitive patterns, the data expands and will not

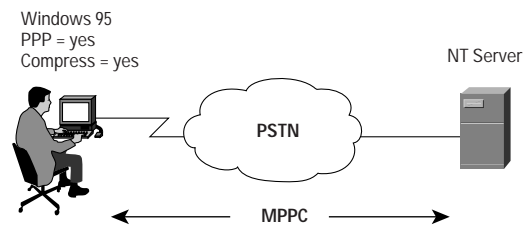
compress. LZS will then compare the before and after images to determine which is the smallest and sends the uncompressed data as it was originally received if expansion occurred.

The solution to this problem is to compress at layer 3 and then encrypt at the same layer. Cisco is very active in providing a solution to this problem with its' work on the IP Security (IPSec) Protocol and the IP Compression Protocol (IPComp). IPSec is a standards-based method of providing privacy, integrity, and authenticity to information transferred across IP networks. IPSec provides IP network-layer encryption. IPComp is a mechanism to reduce the size of IP datagrams and is especially useful when encryption is applied to IP datagrams where layer 2 (e.g., PPP) compression is not effective.

Microsoft Point to Point Compression—MPPC

In March of 1997, Microsoft Corporation introduced the Microsoft Point to Point Compression (MPPC) scheme as a means of representing arbitrary Point to Point Protocol (PPP) packets in a compressed form. MPPC was ratified by the IETF and is known as RFC 2118. Figure 8 illustrates a Windows 95 client connecting to an NT server. The Windows95 client software supports both MPPC and LZS, whereas the Microsoft's NT server only supports MPPC which is negotiated during the Compression Control Protocol (CCP) process.

Figure 8 Windows Client to NT Connection



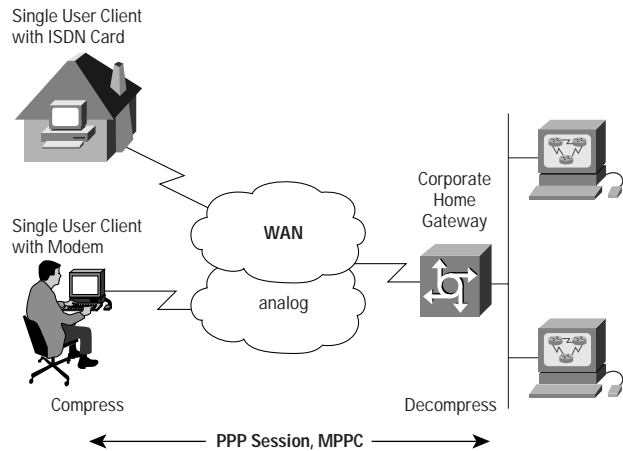
PPP provides a standard method for transporting multi-protocol datagrams over point-to-point links. CCP provides a method to negotiate and utilize compression protocols over PPP encapsulated links. The MPPC algorithm is designed to optimize processor utilization and bandwidth utilization in order to support large number of simultaneous connections. The MPPC algorithm is also optimized to work efficiently in typical PPP scenarios (1500 byte MTU, etc.). Note that MPPC is a layer (3?) compression mechanism and

should not be used with modem compression enabled on the client PC. Again, compressed data does not compress, it expands.

The MPPC algorithm uses an LZ based algorithm with a sliding window history buffer. The MPPC algorithm keeps a continuous history so that after 8192 bytes of data has been transmitted compressed there is always 8192 bytes of history to use for compressing, except when the history is flushed. Before any MPPC packets may be communicated, PPP must reach the Network-Layer Protocol phase, and the CCP Control Protocol must reach the Opened state.

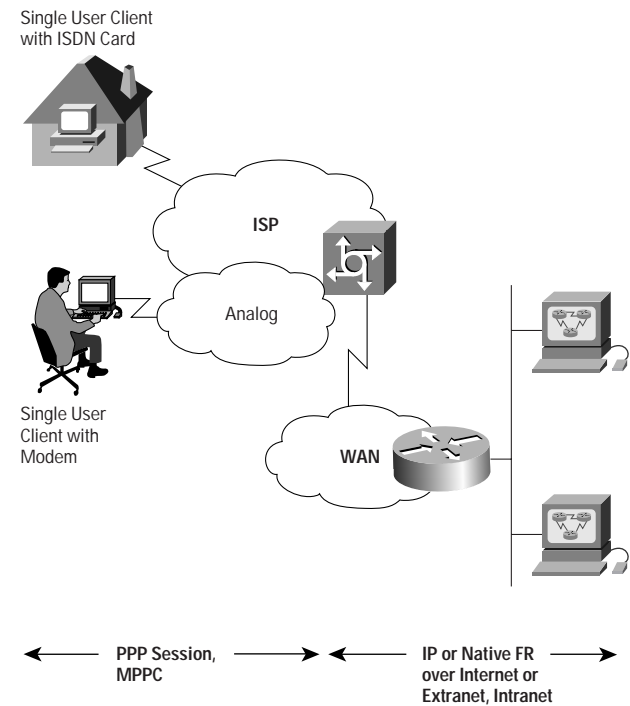
A common application for MPPC is illustrated in Figure 9. Modem compression is not enabled on the client PC and the PPP connection remains compressed end to end through the network until PPP is terminated on the Home Gateway.

Figure 9 MPPC Home Gateway Solution



Another application would be in the whole sale dial market. Many Service Providers (SP) offer wholesale dial services to other SP resellers such as traffic transport, point of presence (POP) collocation, network operation support and other value added services. Figure 10 illustrates one service utilizing MPPC from the client PC to the access server at the edge of the network where the ISP (Internet Service Provider) terminates the PPP session then routes the connection out to the appropriate WAN service.

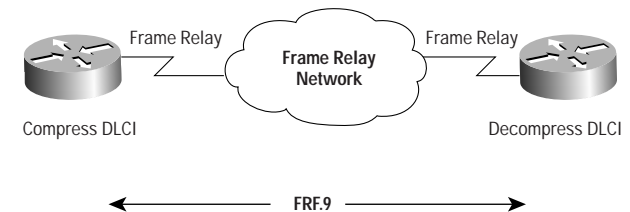
Figure 10 MPPC Wholesale Dial Solution



Frame Relay Payload Compression

In January of 1996, the Frame Relay Forum Technical Committee approved an implementation agreement for data compression over Frame Relay, FRF.9. This agreement specifies a standard for compression to ensure interoperability. FRF.9 is a per virtual circuit compression mechanism for both switched virtual circuits (SVC) and permanent virtual circuits (PVC) and is negotiated at the time the DLCI is initiated. The compressed payload is transported through the Frame Relay network decompressed at its' termination point which implies that FRF.9 is point to point or end to end as illustrated in Figure 11. Cisco currently supports FRF.9 mode 1 and are evaluating mode 2 which allows more parameter configuration flexibility during the LCP compression negotiation.

Figure 11 FRF.9 Solution



Data Compression Design Issues

Data compression is effective in improving WAN transmission only when an application can accept the various characteristics of compression as applied to real-time data traffic, which may or may not allow its use. When considering whether to use compression, keep these network characteristics in mind:

1. Number of remote sites
2. Increase in latency as a by-product of compression
3. Throughput or congestion management
4. Memory requirements
5. Speed as a function of CPU cycles and compression algorithm instructions
6. Compression ratio

Each characteristic is considered in more detail in the following sections.

Number of Remote Sites

When a dictionary-based compression algorithm is used, each point-to-point connection must have dedicated memory for the connection. The greater the number of remote sites, the greater the requirement for memory. Each additional line increases CPU utilization, and the additional burden can increase latency. A typical situation: heavy, compressed traffic from many sites pours into a central hub.

Compression-wise network managers obtain fast routers with lots of memory when employing data compression in their networks, or break up traffic across several routers to avoid overloading the network.

Increased Latency

When compression is applied on the original data stream, the information must be processed and analyzed. As a result, latency is added to the input data before transmission. This increase in latency might affect transmission when protocols sensitive to added network latency, such as local-area transport (LAT), are used over the WAN. Latency is a function of the compression algorithm in use and the CPU speed. It is influenced by factors such as the incoming data rate, the dictionary size, the CPU cycles, and the line rate. As the incoming data rate goes up, the router must process the information faster. If the dictionary size is small and CPU is limited, the router compression slows and significant latency is added to the network.

Throughput or Congestion Management

The compression process varies with data type.

Compressibility can take place externally from a device or internally within router firmware. External compression devices provide a variable clock to the router. This clock manages the difference between the effective data throughput to and from the router and the compressed WAN link. Less compressible data results in a lower clock rate to the router than more compressible data. Clock modulation to the router ensures that input data is not lost during the compression and transmission process. The clock rate to and from the WAN connection does not vary. When compression is performed within router software, normal internal queuing mechanisms buffer the variations in throughput.

Memory Requirements

The amount of memory that the router must have and that the network manager must plan on varies according to the protocol being compressed, the compression algorithm, and the number of concurrent circuits on the router. Memory requirements will be higher for Predictor than for STAC, and per-virtual circuit compression techniques will use more memory than per-interface techniques.

In the final analysis, all algorithms require memory to perform compression. The network manager must plan on additional memory for routers in a network utilizing data compression, regardless of the type of operation used.

Speed

Algorithms perform their functions at different rates. For example, algorithms that can process more compression instructions per CPU cycle than others are faster and able to achieve higher line rates. A fast algorithm allows a remote access router to implement software compression while still performing the normal router functions. Another way to view speed is the absence of latency---the lower the processing overhead, the lower the latency that results from the process.

Compression Ratio's

Compression ratios measure compression performance as a ratio that compares the original size of the string to the size of the compressed output string, and is represented by a number like 5:1. This ratio is also referred to as the static compression ratio.

Another compression ratio is time oriented; it compares the relative time required to transmit a known amount of data with and without compression. Called the dynamic compression ratio, it depends specifically on the transmission rate in use and is directly affected by system speed (CPU cycles) and time.

The compression ratio varies with the type of input and particularly with the amount of redundancy in the input. Because LAN environments have many types of applications and protocols running concurrently, the compression ratio for such environments is lower than for those where only one application is transmitting—for example, plain text files or binary information only. There are other considerations that need to be made with respect to compression ratios, and these are described in the next section.

Dispelling the Myth of the Data Compression Ratio

Producers of products that include data compression capability usually cite their data compression ratios. The ratios come from independent testing labs or from the manufacturers themselves, and are expressed in numbers like 2:1, 5:1, or 8:1. What do these numbers mean? Are these absolute values?

First, consider that many variables affect data compression ratios. The data compression process itself varies with the complexity of the compression algorithm, the location--- internal or external---of the compression process, and the availability of system resources such as processor type and memory. Also consider the lab tests themselves. Did the test overrun the line with thousands of same-type packets, or did the testers re-enact typical link usage by loading up the lines with a mix of packet and data types? Because LAN environments have many types of applications and protocols running concurrently, with varying types of traffic running over the WAN link, compression ratios can vary with the type and amount of redundancy in the traffic. The traffic might include a mix of packet types, such as voice or image, or text that is running in a mix of large and small packets.

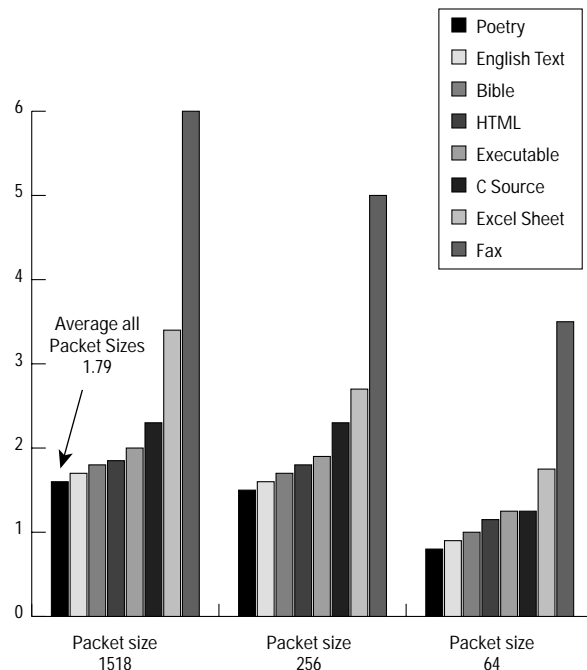
Next consider that only a few compression algorithms are in use today, and that they are based on a similar adaptive or sliding-dictionary algorithm such as the Lempel-Ziv compression scheme. Other examples are the STAC and the Predictor compression schemes. As for evaluating the compression schemes, the de facto standards for evaluating data compression ratios are the Calgary Text Compression Corpus and the Canterbury Corpus files.

The ten year old Calgary Corpus is a suite of nine ASCII encoded documents consisting of normal English fiction and non fiction as well as some unusual styles of English such as a bibliography and a batch of unedited news articles. Additionally, there are several computer and a transcript of a terminal session. The rest of the Calgary Corpus consists of non-ASCII files such as executable code, some geophysical data and a bit-map black and white picture.

The Canterbury Corpus consists of 14 different files and are English text, a CCITT fax test set, C source program, Excel spreadsheet, SPARC executable file, technical writing, poetry, HTML, LISP source, gnu manual page, Shakespeare play, genome of the E.Coli bacterium, the King James version of the bible and the CIA world fact book.

Cisco uses both the Calgary and the Canterbury Corpus files since they represent a wide mix of data types that might be found traversing today's networks, and therefore provides a benchmark for measuring data compression schemes. Figure 12 represents testing that Cisco has conducted using some of the Canterbury Corpus files, a dedicated workstation and the Hi/fn (STAC) LZS compression algorithm. This test illustrates the compressibility of these files or better yet, the true compression ratio's without regards to CPU types, platform latency, memory restrictions and input or output bandwidth bottlenecks.

Figure 12 Sample of Canterbury Corpus File Compression Ratio's



In the final analysis, the true data compression ratio is impossible to determine. One theory states that data that is 75 percent redundant can be compressed only 4:1—a good rule of thumb to remember. A realistic goal is a dynamic overall compression ratio of 2:1 to 1.7:1. When evaluating data compression ratios, use the factors described in this paper that can affect data compression and throughput in your network—the process itself, packet types and sizes, data type and redundancy—to sift out myth from reality.

Conclusion

Increasing demands are being placed on today's WANs, due in part to increases in LAN-based applications and information transfer services such as electronic mail that transmit large amounts of traffic. The costs of starting and maintaining a WAN continue to increase, as well. Data compression, which can also be viewed as a method of eliminating redundancy, makes more efficient use of a WAN link by decreasing the size of a frame and, therefore, the time required to transmit the data across the link. Cisco provides data compression solutions that satisfy a broad range of traffic types—from ASCII text to video images. Over the past twelve years, Cisco's approach has been to provide customers with flexible and scalable solutions. Cisco currently offers three integrated software data compression solutions—TCP/IP header compression, per-interface compression, and per-virtual circuit compression.

Cisco Products That Support Data Compression

Data compression is available on most Cisco routers, where appropriate, including the Cisco access router product family. Combined with the modular features of Cisco IOS™ software, the access product family offers a wide range of solutions to a broad spectrum of users. For a complete list of products that support Cisco IOS software, see the Cisco Systems Products Catalog.



Corporate Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
<http://www.cisco.com>
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

European Headquarters
Cisco Systems Europe s.a.r.l.
Parc Evolic, Batiment L1/L2
16 Avenue du Quebec
Villebon, BP 706
91961 Courtaboeuf Cedex
France
<http://www-europe.cisco.com>
Tel: 33 1 6918 61 00
Fax: 33 1 6928 83 26

Americas
Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
<http://www.cisco.com>
Tel: 408 526-7660
Fax: 408 527-0883

Asia Headquarters
Nihon Cisco Systems K.K.
Fuji Building, 9th Floor
3-2-3 Marunouchi
Chiyoda-ku, Tokyo 100
Japan
<http://www.cisco.com>
Tel: 81 3 5219 6250
Fax: 81 3 5219 6001

Cisco Systems has more than 200 offices in the following countries. Addresses, phone numbers, and fax numbers are listed on the

Cisco Connection Online Web site at <http://www.cisco.com>.

Argentina • Australia • Austria • Belgium • Brazil • Canada • Chile • China (PRC) • Colombia • Costa Rica • Czech Republic • Denmark
England • France • Germany • Greece • Hungary • India • Indonesia • Ireland • Israel • Italy • Japan • Korea • Luxembourg • Malaysia
Mexico • The Netherlands • New Zealand • Norway • Peru • Philippines • Poland • Portugal • Russia • Saudi Arabia • Scotland •
Singapore