

Technical Briefs #2

by

Chaelynn M. Wolak
wolakcha@scsi.nova.edu

A paper submitted in fulfillment of the requirements
for DISS 780 – Assignment Eight

School of Computer and Information Sciences
Nova Southeastern University

April 2000

Compression

Data and image compression will be a \$100 billion business this year according to IDC (Millman, 1999, January 18). Essentially compression eliminates unnecessary information such as redundant data or empty fields. Elimination of unnecessary information reduces the requirements for storage space. A storage space saving is not the only driving force for compression.

Multimedia productions tend to have large file sizes due to the number of images and sound files. Thus, the current storage devices are too slow and networks have too narrow of a bandwidth to be able to handle such productions. Therefore, to reduce the number of data bytes being transferred, algorithms have been developed to compress this data so it can be stored or sent through the network (Tannenbaum, 1998).

By compressing data, it can be stored more conveniently and retrieved in less time. Internet businesses rely on compression techniques since information is the main product/service. These techniques are important because they help accomplish transferring of information more efficiently. Incidentally, this results in savings of time, space, and money (Lawson, 1997, September 1).

There are two basic forms of compression. They are lossless and lossy. To capitalize on these two forms of compression, there are four main coding schemes. Each scheme has a theory of how to eliminate the redundancy in data. In the following sections, a discussion of lossless, lossy, and the four main coding schemes are briefly described.

Lossless Compression

Lossless compression applies to data, text, and program files (Millman, 1999, January 18). Lossless means that upon decompression, all file information will be completely restored.

These types of compression programs identify repeating data patterns and replace them with a code (Lawson, 1997, September 1). For example, data “A0A0A0A0” can be replaced with “4 A0”. The replacement code “4 A0” has the same meaning as the original data. Thus, no information is eliminated from the file to compress it.

Lossy Compression

Lossy compression applies to graphics, audio, and video files (Millman, 1999, January 18). This compression technique works by eliminating similar data bits located next to one another in a file. Thus, when the file is restored (i.e. decompressed), only a portion of the original information is present (Lawson, 1997, September 1). The lost data is not important to the functionality of the data file. “For example, a lossy image of a sandy beach will be less detailed, because the grains of sand are very similar” (Millman, 1999, January 18).

Compression Coding Schemes (Theories)

The four main compression coding theories are Huffman coding, Arithmetic coding, Lempel-Ziv coding, and Shannon-Fano coding. Huffman coding assigns short codes for the most frequent characters. For the less frequent ones, a longer code is assigned. By assigning these characters, it reduces the amount of repetitious data. The Huffman encoding has to read the data twice. First, the frequency distribution of the data characters is determined and second to encode the data (Lawson, 1997, September 1).

Arithmetic coding assigns a code to a string of data. This type of coding was developed because the Huffman coding could not recognize characters between 0 and 1. However, arithmetic coding does not go without its disadvantages. It requires more CPU processing to run and suffers from speed difficulties (Lawson, 1997, September 1).

Lempel-Ziv coding uses a dictionary model to reduce the amount of data. It is similar to reducing a stream of words to acronyms. This type of compression coding relies on how long the dictionary replacements are and how much past text the program uses for comparison (Lawson, 1997, September 1).

Shannon-Fano coding uses a tree symbol approach. “For a given list of symbols, a corresponding list of probabilities and frequency counts is tallied so each symbol’s relative frequency of occurrence is known” (Lawson, 1997, September 1). The list is sorted according to frequency where the most frequent appears at the top.

Conclusion

These four compression-coding techniques are included in the latest compression software programs. For example, Ascent Solutions uses the Hoffman coding technique whereas PKZIP uses Shannon-Fano’s. Compression is not a new concept. Recently, it has been revolutionized.

Several developments have continued the need for advanced compression techniques. The Internet has been a major contributor. However, so has the increasing use of digital representation of video and analog signals, improved understanding of the HVS and HAS characteristics, and the development of MPEG (Robin). It is these advancements in technology where compression is a necessity rather than a luxury.

References:

- Lawson, M. (1997, September 1). Data compression: Accomplishing more with less. *Enterprise Systems Journal*.
- Millman, H. (1999, January 18). Image and video compression. *ComputerWorld* [Online]. Available: <http://www.electrictlibrary.com> [2000, March 19].
- Tannenbaum, R. S. (1998). *Theoretical foundations of multimedia*. New York, New York: Computer Science Press.

Multimedia Authoring Software

A multimedia production is developed from authoring software. Authoring software is the intermediary agent where commands from a developer are accepted and converted to the appropriate computer instructions for a final multimedia production (Tannenbaum, 1998). These types of software programs save a developer time. In addition, a developer does not need to learn the complex computer code to make a multimedia production work.

There are many types of authoring software. They are frame-based, icon-based, or time-based. In the following pages, each type is described as well as the advantages and disadvantages of using them for creating a travelogue.

Frame-based Authoring Software

In frame-based software programs, a developer can control elements of a multimedia production via a slide, a card, or a page in a book. An example of a slide interface is Powerpoint. This is a type of presentation graphics program. HyperCard or SuperCard uses the metaphor as a stack of cards. ToolBook is an example of a page in a book (Tannenbaum, 1998).

In looking at one example of a frame-based authoring software, such as HyperCard version 2.4, design cards (called screens) are relatively easy to program. A developer needs to add buttons to link the cards, fill in dialog boxes to program the buttons, and write a little HyperTalk code. For someone developing a travelogue of his/her city can be quite easy. By creating scenes on cards and arranging them to play in sequence, a person can create a multimedia travelogue with minimal training. However, with this software, there are disadvantages. It is hard to create colored stacks without complicated programming as this software still uses a black-n-white interface. In addition, it cannot be used on a Windows-based machine since it cannot operate in that environment (Beekman & Shechter, 1998, September).

So, if one designs a travelogue for visitors, he/she must make sure the software is capable of handling all types of operating systems (i.e. cross platform functionality).

Icon-Based Authoring Software

For icon-based authoring programs, basic icons represent standard functions supported by the system required for final production. A developer clicks and drags these icons to a desired position in the sequence that the icons are being built (Tannenbaum, 1998).

An example of icon-based authoring software is Macromedia's Authorware. Here a developer can drag function icons onto a flow line where the multimedia production moves along this line in response to the developer's actions. All interaction and control are shown as icons on this flow line (Long, 1995, August 21).

If one is more comfortable with graphics (icons) then icon-based authoring software is the way to go. Dropping and dragging is common practices in most software packages. So learning this type of software should be relatively easy. However, if one wanted complicated interaction or specialized functionality, then this may not be the program to use.

Time-Based Authoring Software

The developer indicates a time sequence for the presentation of the various elements in a production. "The metaphor of a time line is employed by the authoring system and the developer indicates which elements are to be presented at various times along the line" (Tannenbaum, 1998).

Macromedia's Director 7 is an example of this time-based authoring software. One of the greatest strengths of this software is its cross-platform functionality. It supports distribution to PCs, Macs, UNIX, and even 3DO game machines. In previous versions of this package, one would have to learn Lingo (Heck, 1996, August 1). This is Director's programming language.

Now, with the latest software package, a developer does not have to know Lingo since they have a library of canned-Lingo programs.

Saying that, however, does not make the program excel for developing a travelogue.

What if a canned-Lingo program does not exist that a developer may want? This program would still require the developer to be able to program in Lingo. For designing a multimedia travelogue in this program can be easy if one likes to work with a timeline though.

Conclusion

It is not easy to quantify the advantages or disadvantages of frame-based, icon-based, or time-based authoring software. It is dependent on the individual who is developing in it. No software product is capable of addressing every purpose; although authoring software seems to be getting easier to learn and program. The best multimedia authoring software is the one that fits not only your project but also your level of expertise. It is only then that one can determine if he/she has selected the right one.

References:

Beekman, G., & Shechter, T. (1998, September). HyperCard 2.4. *Macworld*.

Heck, M. (1996, August 1). Multimedia authoring: If you build it, they will come. *PC World Monthly* [Online]. Available: <http://www.electrictlibrary.com> [2000, March 26].

Long, B. (1995, August 21). Authorware 3.0 writes new chapter on interactivity. *MacWeek*, 9 (33).

Tannenbaum, R. S. (1998). *Theoretical foundations of multimedia*. New York, New York: Computer Science Press.

Quality of Service (QoS)

The increased performance of computers has resulted in the expanded use of multimedia communication and mobile computing. Multimedia communication such as video conferencing (e.g. desktop conferencing) is subject to system and network performance. For communication-intensive applications such as this, the quality of service (QoS) must be guaranteed (Kosuga, Yamazaki, Ogino, & Matsuda, 1999).

“Multimedia QoS is typically measured using technical parameters such as end-to-end delay and jitter” (Ghinea & Thomas, 1998, September 13 - 16). The minimization of end-to-end delay and jitter is especially important when a multimedia application is distributed to multiple users. The human element plays a very important role in multimedia QoS, which is often overlooked (Ghinea & Thomas, 1998, September 13 - 16). The end-user’s satisfaction with the quality of the multimedia presentation along with his/her capacity to understand, analyze, and synthesize the informational content of the presentation is also part of QoS.

Many companies have developed technologies to improve QoS on networks. Emerging technologies such as resource reservation protocol (RSVP), IP type of service, and IEEE 802.1 tags are tools that try to set priorities for the computer and network resources (Lawson, 1999, June 14). For example, Microsoft incorporates RSVP to improve QoS in Windows 2000. However, these are not believed as the only solution to QoS as they do have their disadvantages.

There are three new emerging QoS architectures being explored to improve it in distributed multimedia applications. They are adaptive QoS management mechanism, Quartz, and OMODIS. Each of these is described next.

QoS Management Mechanism

In this architecture scheme, QoS is broken into six subtasks. These subtasks are then managed by the corresponding agents such as Personal Agent (PA), Application Agent (AA), Stream Agent (SA), Terminal Resource Agent (TRA), Network Resource Agent (NRA), and Network Agent (NA) (Kosuga et al., 1999). This layered QoS model uses these multi-agents to manage resources for distributed multimedia applications.

The PA is defined at the user level, which manages the user interaction. AA QoS manages the media stream by each application. This layer is specified by the application level parameters such as frame rate, frame size, etc. Therefore, when a user starts a multimedia application, an AA is created that attaches to that application. The QoS requirements for the application from the user are translated by the PA and then passed to the AA.

The SA focuses on the maintenance of QoS. The AA determines the stream QoS using QoS negotiation. Once this stream is determined, the AA passes the stream to the SA, which in turns requests the TRA or the NRA to reserve terminal or network resources. The TRA manages the CPU utilization and memory size. The NRA manages the bandwidth and node buffer size. Lastly, the NA manages network control (Kosuga et al., 1999).

Quartz

Quartz is a generic architecture that addresses QoS in open systems. Typically, the focus has been on middleware that provides applications with mechanisms for QoS specification and management. However, this typical focus has a strong dependency on a

particular computing platform and specific suite of applications (Siqueira & Cahill, 1998). Quartz addresses this limitation.

Quartz works by “adopting a highly flexible, extensible, component-based platform-independent design, which allows user transparency from the underlying system and at the same time is suitable for open distributed systems” (Siqueira & Cahill, 1998). Quartz manages the QoS issues related to the application while allowing the programmer to concentrate on the functional aspects of the application.

OMODIS

It is well known that multimedia applications require QoS. However, an area that has been overlooked is the multimedia database systems (MMDBS). The management of QoS between the MMDBS and the middleware has not been handled (Goebel & Plagemann, 1998). One project that focuses on this area is OMODIS.

The OMODIS project is an adaptive distributed multimedia system for Lectures-on-Demand (LoD) which support asynchronous interactive distance learning (IDL). The lecture and other relevant multimedia materials are stored in the MMDBS. Users then connect to the MMDBS for that information. Thus, this project focuses on the QoS support in the LoD system.

The LoD architecture consists of three main areas – browser, MULTE-ORB, and the MMDBS. The browser is used to specify the user’s QoS requirements such as video quality, audio quality, etc. The user interaction via the browser and the browser capabilities determine the target range of QoS.

The MULTE-ORB transports this target range from the browser to the MMDBS. The MULTE-ORB is a highly flexible multimedia ORB that transports QoS

specifications. The MMDBS, which includes the stored lectures and other multimedia materials, receive the query from the user. Then the MMDBS requests the MULTE-ORB to establish a binding with the appropriate QoS support (Goebel & Plagemann, 1998).

Conclusion

As detailed above, these are only three alternatives in developing a methodology for QoS in distributed multimedia applications. Much work has been done in the area of QoS. However, the mechanisms for prioritizing traffic and resources across all platforms are still in their infancy.

References:

- Ghinea, G., & Thomas, J. (1998, September 13 - 16). *QoS impact on user perception and understanding of multimedia video clips*. Paper presented at the Proceedings of the 6th ACM international conference on Multimedia, Bristol, United Kingdom.
- Goebel, V., & Plagemann, T. (1998). *Mapping user-level QoS to system-level QoS and resources in a distributed lecture-on-demand system*. Paper presented at the Proceedings of the The Seventh IEEE Workshop on Future Trends of Distributed Computing Systems.
- Kosuga, M., Yamazaki, T., Ogino, N., & Matsuda, J. (1999). *Adaptive QoS management using layered multi-agent system for distributed multimedia applications*. Paper presented at the Proceedings of the 1999 International Conference on Parallel Processing.
- Lawson, S. (1999, June 14). Better QoS on the horizon. *Info World* [Online]. Available: <http://www.britannica.com> [2000, March 26].
- Siqueira, F., & Cahill, V. (1998). *Delivering QoS in open distributed systems*. Paper presented at the Proceedings of the The Seventh IEEE Workshop on Future Trends of Distributed Computing Systems.