Communication Optimization for Image Transmission in Computer-Assisted Surgery

Hanping Lufei[†], Weisong Shi[†], and Lucia Zamorano[¶]

[†]Department of Computer Science
Wayne State University, Detroit, Michigan
[¶]Department of Neurological Surgery
Wayne State University, Detroit, Michigan

Abstract. The fast growth of computer network technologies, like IEEE 802.11x series, Bluetooth, GPRS etc., makes it possible to build computer-assisted surgery (CAS) system in a distributed way, e.g. the wireless connection in operation room, telepresent (surgeons participating in surgery remotely) from anywhere, at anytime. However, the communication overhead of huge image transmission is a big obstacle to the wide deployment of distributed CAS systems, especially in a low-bandwidth wireless network environment. In this paper, we proposed *Bitmap-Diff*, a novel network bandwidth reduction algorithm which is especially tailored for medical images. The comparison between our algorithm and four other popular bandwidth reduction algorithms, namely *zip compression*, *delta-encoding*, *fix-sized blocking* and *vary-sized blocking*, shows that *Bitmap-Diff* generates much less transfer bytes, about 35% less than other algorithms. It is the best communication optimization algorithm for medical image we have ever seen. We believe this will pave the way for the deployment of distributed CAS systems.

1 Introduction

Computer-assisted surgery (CAS) uses registered images to guide a surgical procedure. At its current level of development, its utility is limited by some technical problems, for example, 1) the intraoperative use of cumbersome wired technology [1]; 2) low speed of access to surgically relevant information; and 3) lack of access during surgery to remotely located expertise [2]. A more advanced network technology powered distributed CAS system is needed in which the surgeons in operation room are not restricted by cumbersome network wires and cables any more, they can also get the new data on the fly. On the other hand, remotely participating surgeons (even at home with dialup connection) are able to monitor or participate the whole surgery procedure (telepresence) from anywhere, at anytime. In order to implement all of these goals, the data communication overhead in various network environments is one of the biggest obstacles to overcome. For instance, it takes about 5 minutes to transmit a full set of DICOM images, including 75 slices for axial, sagittal and coronal view, across a 11Mbps 802.11b wireless network.

Several application-specific communication optimization techniques have been proposed in different contexts by computer scientists. *Delta-encoding (*$Delta$*)* was proposed by Mogul *et al.* in the context of HTTP traffic [3] to exploit the similarity of

Web documents. Rsync [4] synchronizes different versions of the same data by using fix-sized chunks for communications. LBFS [5] takes a step further by reusing the data chunks across multiple similar files (including multiple versions of the same file). Although their results are promising, none of them are originally designed for medical images transmission over heterogeneous network environments. In this paper, we propose a new algorithm called *Bitmap-Diff*, which is specifically fitting for the characteristic of medical images. A systematic performance evaluation between *Bitmap-Diff* and other bandwidth reduction algorithms, namely *zip compression*, *delta-encoding (vcdiff)*, *fix-sized blocking* and *vary-sized blocking* is reported in terms of three performance metrics: *computing time*, *total time* and *transfer bytes*. In the evaluation, the most widely used document type, DICOM [6], in CAS environment is examined. The experiments are performed under four representative network connection technologies: *100Mbps switched Ethernet*, *802.11b wireless LAN*, *cable modem*, and *phone dialup*. Our experimental results show that the new proposed *Bitmap-Diff* outperforms other algorithms in all four different network environments. We believe this algorithm will pave the way for the real deployment of distributed CAS systems.

2 Background

2.1 Distributed CAS system

With the rapid development of computer network technology, several revolutions could happen in a traditional centralized CAS system which runs within an operation room only. In the distributed CAS system, as shown in Figure 1, surgeons in the operation room can retrieve any new information from data resources located anywhere via network in case of emergence, use wireless micro camera to get real time images and transmit to the Internet; all the computer-human interface are using wireless network with maximum mobility and portability; remote telepresenting surgeons can acquire any information of the patient or surgery as they want. All of them rely on new network technologies and communication optimization techniques to reduce the overall latency.

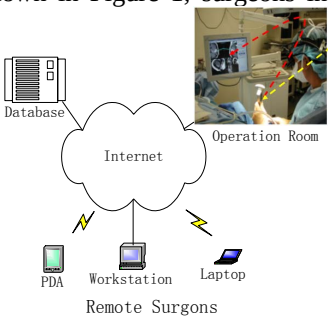


Fig. 1 An example scenario of distributed CAS.

2.2 DICOM images

The Digital Imaging and Communications in Medicine (DICOM) standard is created by the National Electrical Manufacturers Association (NEMA) to aid the distribution and viewing of medical images [6]. DICOM is the most common standard for receiving scans from a hospital.

A single DICOM file contains both a header (which stores information about the patient's name, the type of scan, image dimensions, etc), and all of the image data (which can contain information in three dimensions) [6]. The DICOM header describes

the image dimensions and retains other text information about the scan. The size of this header varies depending on how much header information is stored. The image data follows the header information (the header and the image data are stored in the same file). The header information is organized in groups . Of particular importance is the group which defines the Transfer Syntax Unique Identification which reports the structure of the image data and reveals whether the data have been compressed. DICOM images can be compressed both by the common lossy JPEG compression scheme as well as a lossless JPEG scheme that is rarely seen outside of medical imaging. Besides reporting the compression technique, the Transfer Syntax UID also reports the byte order for raw data. Some of the definitions of Transfer Syntax UID are shown in Table 2.

Transfer Syntax UID	Definition
1.2.840.10008.1.2	Raw data, Implicit VR, Little Endian
1.2.840.10008.1.2.x	Raw data, Explicit VR x = 1: Little Endian x = 2: Big Endian
1.2.840.10008.1.2.4.xx	JPEG compression xx = 50-64: Lossy JPEG xx = 65-70: Lossless JPEG
1.2.840.10008.1.2.5	Lossless Run Length Encoding

Fig. 2 Different DICOM formats

2.3 Previous Bandwidth Reduction Algorithms

Computer scientists have proposed several algorithms to reduce network transfer bytes in different contexts, such as, delta-encoding for Web documents [3] and email files, object composition for dynamic and personalized Web contents [7], and block-based techniques for file synchronization [5, 4]. Next we will briefly describe each algorithm and the corresponding protocol: 1) *Delta Encoding*, it was first proposed in the context of HTTP traffic [3]. Currently, the best delta-encoding algorithm is `vcdiff` proposed by Korn and Vo [8]. `vcdiff` is a general and portable encoding format for delta encoding. 2) *Fix-Sized Blocking*, It is used to synchronize different versions. In this approach, files are updated by dividing both files into fix-sized chunks. The client sends digests of each chunk to the server, and the server responds only with new data chunks. Based on old version and the differencing, the new version can be rebuilt. 3) *Vary-sized Blocking*, it divides files into chunks, demarcated by points where the Rabin fingerprint [9] of the previous 48 bytes matches a specific polynomial value. It tends to identify portion even after insertions and deletions have changed its position in the file. The boundary regions are called breakpoints. The client then computes a digest of each chunk and sends them to the server, which maintains a database of chunks from all local files. The server responds to the client with bits for new chunks. The client then sends each new chunk to the server. 4) *Zip compression*, we use gzip to compress the file at the sender and decompress it at the receiver. Gzip is a popular data compression program which uses LZ77 algorithm. We use `gzip 1.3.3` [10] in our implementation.

3 Bitmap-Diff

3.1 Motivations

RAW format DICOM is one of the most popular medical image formats. RAW format works in a very different way as compressed format does. When an image has been taken, the information from the sensor is stored without being processed. Whereas a TIFF file sacrifices the exceptional image quality for the accompanying large file size and a JPEG sacrifices the small file size for reduced quality, a RAW file combines the best of both. That is one of the reasons why raw data DICOM images are widely used

across the medical industry. However there is no free lunch RAW format is unavoidable associated with much bigger size which seriously compromises its transmission efficiency across network. So we plan to propose a new bandwidth reduction algorithm especially for the RAW image format like DICOM or BMP.

3.2 File Similarity

Intuitively, the RAW format implies that there may have localized similarity between two continuous slices of DICOM images. Localized similarity means the difference between two images happen in some small areas instead of global scope. So a proper chunksize should be able to capture the changes by scanning the corresponding byte streams of each DICOM image file. The result of a comparison experiment is shown in Figure 3. We define similarity as the ratio of total size of matched chunks to the total file size. Each file is chopped into a series of chunks with the specific chunksize as shown on x axis. Then we compare chunks one by one correspondingly between two files to get the similarity between the two files as shown on y axis. Although the similarity is still poor at double bytes chunksize, it upsurges to 54.3% at single byte level.

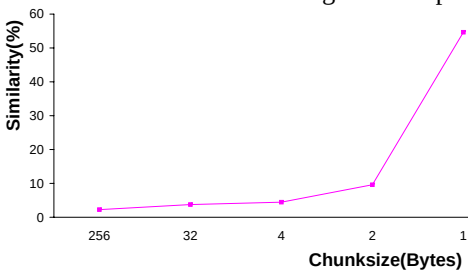


Fig. 3 Similarities for different chunksize

3.3 Algorithm Principle

Based on the similarity observation, we proposed the Bitmap-Diff algorithm which marks and saves the difference between two files in a bitmap style. Let us use an example to show how it works. In Figure 4, there are two files, image #1 and image #2, as the input of the algorithm. Both of them have totally 4 bytes in their byte streams. They have the same first two bytes, Byte 0 and Byte 1, as shown in same color. They have different last two bytes, Byte 2 and Byte 3 as shown in different colors. The Bitmpa-Diff works as following: First, it creates a bit array which uses one bit in the array to represent one byte of the file. We call this bit array as *Bitmap*. Then the algorithm scans byte streams of two images, byte by byte, from left to right in this example. If the corresponding bytes are same it will set the corresponding bit in the Bitmap as 0. Otherwise, the bit is set to 1 and the corresponding byte in image 2 is saved. For instance, the algorithm first compares Byte 0 of image 1 and Byte 0 of image 2, since they are identical, the first bit in Bitmap is set to 0, the second bytes of two files are still same, so the second bit in Bitmap is also 0, but for last two bytes they are different, therefore the according bits are set to 1 in the Bitmap and bytes are saved in Diff. Finally, Bitmap-Diff can use other complicated encoding algorithms to combine Bitmap and Diff together in order to reduce the size further. More details about the algorithm can be found in the technical report version [11].

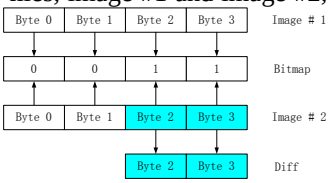


Fig. 4 Bitmap-Diff algorithm

4 Design Implementation and Metrics of Interest

We abstract a simple communication model, as illustrated in Figure 5, which con-

sists of a client and a server at each end. Without losing generality, we assume that both ends have an old version of DICOM image, for example the predecessor of two continuous images, shown as the shadow block in Figure 5. The difference, called delta, i.e., the gray triangle , is calculated and sent to the counterpart during the updating phase. Based on the delta and the old version we can faithfully rebuild the new version, the successor of the two continuous images.

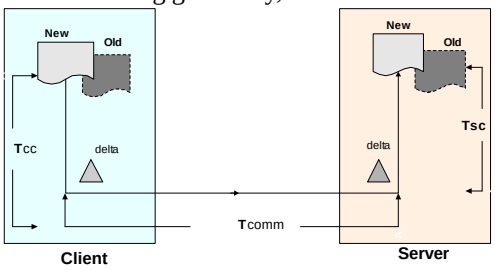


Fig. 5 A basic communication model

In Figure 5, T_{cc} and T_{sc} are the computing time on client and server side respectively. They are related to the server and client hardware configuration and optimization algorithms. T_{comm} is the communication time between client and server. Let us denote the current bandwidth between a client and a server by $B_{current}$, the total time of image transmission by T_{total}, the delta file size by S_{delta}, and the total computation overhead by T_{comp} respectively. Then we have

$$T_{total} = T_{comp} + T_{comm}; T_{comp} = T_{sc} + T_{cc}; T_{comm} = \frac{S_{delta}}{B_{current}}$$

5 Performance Evaluation

5.1 Experiment Platforms and Input Images

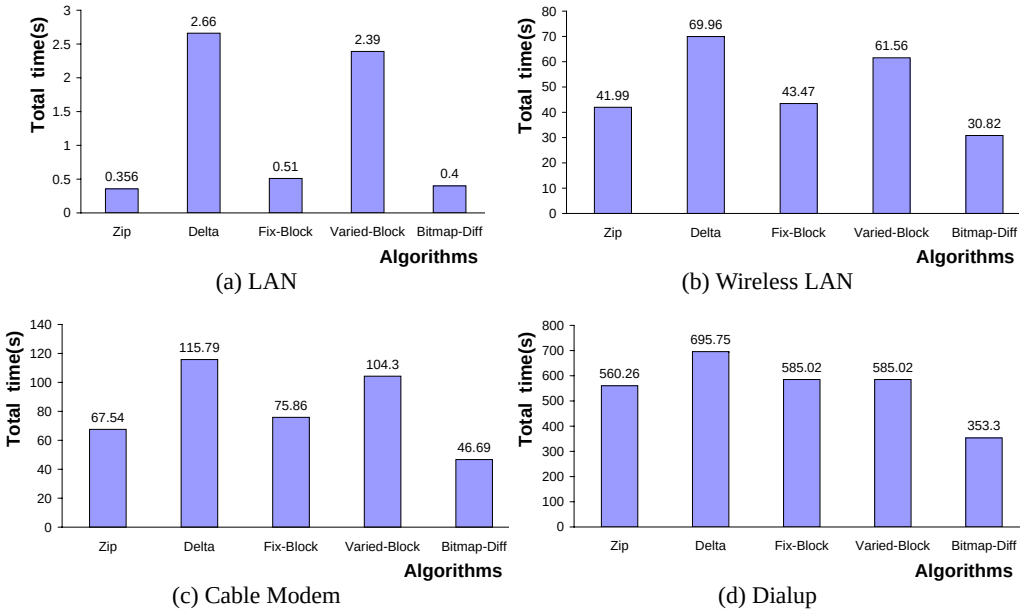
We use four representative network platforms: 1) 10/100Mbps Fast Ethernet LAN, 2) 802.11b Wireless network, 3) Cable Modem, and 4) 56K Dialup modem. Client computer runs Fedora Core Linux on Pentium-4 3.06 GHz, 512MB RAM, Broadcom 440x 10/100 Integrated NIC. Server computer runs RedHat 8.0 on Pentium-4 2.0GHz, 512MB RAM and Realtek RTL8139 Family PCI Fast Ethernet NIC. Cable Modem and 56K Dialup connections are emulated by using NISTNet [12], a network emulation package developed by NIST.

We have a full set of MRI DICOM images with 75 slices for axial view, sagittal view and coronal view respectively. In our experiment we use the sagittal view which contains 75 slices. All of them have the same size of 136KB with Transfer Syntax UID as 1.2.840.10008.1.2.1.

5.2 Comparison of Different Algorithms

In this section, we analyze five algorithms in terms of the total time, the computing time and transfer bytes, in different network connections. We will represent `zip compression` by `Zip`, `delta-encoding` by `Delta`, `fix-sized blocking` by `Fix-Block` and `vary-sized blocking` by `Varied-Block` respectively. Figure 6 shows the total time in four different network environments. The `x` axis represents algorithm types and the `y` axis shows the total time. Figure 7 illustrates the computing time. The horizontal line describes algorithm types and the vertical line stands

for the total computing time consisting of two parts: T_{cc} and T_{sc} in different colors. Figure 8 reports the practical transfer bytes of each algorithm, where the `x` axis still exhibits algorithm types and `y` axis displays transfer bytes.



(a) LAN (b) Wireless LAN

(c) Cable Modem (d) Dialup

Fig. 6. A comparison of the total time of different differencing algorithms in four different network environments: (a) LAN, (b) Wireless LAN, (c) Cable Modem, and (d) Dialup.

Now we are in a position to compare different differencing algorithms. Intuitively `Zip` should be the fastest one in computing time. Figure 7 proves this intuition although it is only 13% faster than `Bitmap-Diff`. From this figure we can find computing time on server side is about three times more than the computing time on client side because of the asymmetric compression algorithm used in gzip. This is a good character especially for low computing resource clients like hand hold devices or cell phones. With regard to transfer bytes `Zip` is at the same magnitude as `Fix-Block` and `Varied-Block` algorithms but much worse than `Bitmap-Diff`, as shown in Figure 8. In low speed networks, e.g. wireless or dialup, where transfer bytes signifies the total time, `Zip` will be worse than `Bitmap`, as we can see in Figure 6 (b) (c) and (d). In high speed networks, like LAN, where the computing time has substantial effect on the total time, `Zip` may catch up with or even slightly outperform `Bitmap`, as shown in Figure 6 (a). Generally speaking, although `Zip` is better than `Delta`, `Fix-Block`

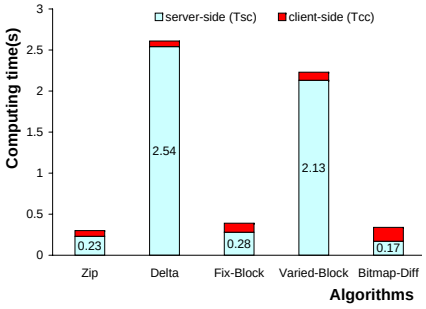


Fig. 7 Computing time for different algorithms

and Varied-Block, it is not as good as Bitmap-Diff.

We can easily find that Delta gets the most transfer bytes, as shown in Figure 8, as well as the longest computing time as shown in Figure 7. These two crucial factors determine that Delta will be the last candidate in any network environments in terms of total time, as shown in Figure 6.

Fix-Block is easy to implement. Varied-Block is an algorithm based on Rabin Fingerprint [9]. Basic idea behind this algorithm is to find the similarity of two unrelated files. In Section 3.2, we showed that it is hard to achieve enough similarity using the Varied-Block algorithm between two DICOM images using large block sizes. While small block sizes may improve the similarity ratio, the computing overhead will overwhelm the benefit. Fix-Bl and Varied-Block have similar transfer sizes and diverse computing time. As we said before in the fast network like LAN, the speed is so fast that it amortizes, to some extent, the difference between transfer bytes of each algorithm, consequently the computing time roughly controls total time. That is why in Figure 6 (a) the total time of Fix-Block and Varied-Block follow the similar shape as in the computing time of Figure 7 . On the contrary, in slow network like Dialup, total time manifests the transfer bytes as we can see the relationship between Fixed-Block and Varied-Block in both Figure 6 (d) and Figure 8. Whereas Fix-Block and Varied-Block are worse than Zip and Bitmap-Diff, they are useful in some other places. For some infrequently changed files like system files dividing them into chunks and saving them into database in advance will greatly improve the overall performance of the Varied-Block algorithm.

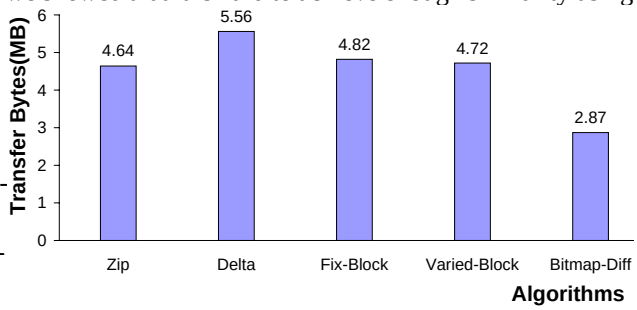


Fig. 8 Transfer bytes of different algorithms

So far Bitmap-Diff is the best choice in terms of all three metrics: the total time, transfer bytes, and the computing time. First, regarding the computing time, it is only about 13% slower than zip. The fast computing speed of Bitmap-Diff inherits from the efficient differencing algorithm inside the Bitmap-Diff. For two files with 50% similarity, almost half of the size is reduced for transmission. Furthermore the computing times at both server and client side are balanced, server side computing time of Bitmap-Diff is much less than that of Zip. This is very useful for heavy load server because short server processing time means more client capacity and throughput; Second, in terms of transfer bytes, Bitmap-Diff saves more than 1/3 of bandwidths compared with Zip, Fix-Block and Varied-Block or even about 1/2 of Delta. In broadband networks this may be not so significant, but in low bandwidth network where transfer bytes signifies the total time Bitmap-Diff can dramatically improve the overall system performance. For example in remote surgery diagnosis procedure, small transfer bytes is extremely crucial for short delay and no jitter in real time image transmission; Finally, as far as the total time is concerned, Bitmap-Diff is much bet-

ter than other algorithms except a little worse than `Zip` in LAN, as shown in Figure 6. With the decreasing of network bandwidth `Bitmap-Diff` enlarges its advance to others thanks to its low transfer bytes and computing time; In summary, `Bitmap-Diff` outperforms other algorithms in almost all the other situations. Its simplicity demands small computing resources on both the client and the server side. `Bitmap-Diff` is capable of communication optimization for CAS.

6 Conclusions and Future Work

In this paper, a new efficient bandwidth reduction algorithm for medical images, *Bitmap-Diff* is proposed. A comprehensive comparison between *Bitmap-Diff* and four other popular bandwidth reduction techniques is presented. We found *Bitmap-Diff* is the best one in all four evaluated network environments for medical image transmission. Our future work is integrating *Bitmap-Diff* into the *Fractal* framework, which is an ongoing project at Wayne State University, aiming to transparently achieve communication optimization and content adaptation for heterogenous devices in distributed CAS systems.

References

1. Zamorano, L., Kadi, A.M., Jiang, Z., Diaz, F.G.: Intraoperative localization of a hand-held laser instrument using an infrared-based system. Proceedings of the Meeting of the American Society for Stereotactic and Functional Neurosurgery, Part II, Vol. 66 (1996) 152
2. Zamorano, L., Jiang, Z., Grosky, W., Kadi, A.M., Diaz, F.G.: Tele-presence and virtual reality in computer-assisted neurological surgery: basic theory and a prototype. Virtual Reality and Medicine: The Cutting Edge, Vol. 1, No. 1 (1994) 63–66
3. Mogul, J.C., Douglis, F., a. Feldmann, Krishnamurthy, B.: Potential Benefits of Delta-Encoding and Data Compression for HTTP. In: Proc. of the 13th ACM SIGCOMM'97. (1997) 181–194
4. Tridgell, P., Mackerras, P.: The rsync algorithm. Technical Report TR-CS-96-05, Department of Computer Science, Australian National University (1996)
5. Muthitacharoen, A., Chen, B., Mazières, D.: A low-bandwidth network file system. In: Proc. of the 18th ACM Symp. on Operating Systems Principles (SOSP-18). (2001)
6. Dicom standard, http://medical.nema.org
7. Shi, W., Karamcheti, V.: CONCA: An architecture for consistent nomadic content access. In: Workshop on Cache, Coherence, and Consistency(WC3'01). (2001)
8. Korn, D.G., Vo, K.P.: Engineering a differencing and compression data format. In: Proceedings of the 2002 USENIX Annual Technical Conference. (2002)
9. Rabin, M.O.: Fingerprinting by random polynomials. Technical Report TR-15-81, Harvard Aiken Computation laboratory (1981)
10. Gzip tool, http://www.gzip.org
11. Lufei, H., Shi, W., Zamorano, L.: Communication optimization for image transmission in computer-assisted surgery. Technical Report Technical Report MIST-TR-2004-006, Feb, 2004, Wayne State University (2004)
12. Nist network emulation tool (nistnet)
13. Spring, N.T., Wetherall, D.: A protocol independent technique for eliminating redundant network traffic. In: Proc. of ACM SIGCOMM'00. (2000) 87–95
14. Manber, U.: Finding similar files in a large file system. In: Proceedings of the USENIX Winter 1994 Technical Conference. (1994) 1–10
15. Berliner, B.: CVS II: Parellizing software development. In: Proceedings of the Annual USENIX Technical Conference. (1990)