Image Transfer: An End-to-End Design

Charles J. Turner and Larry L. Peterson[1]

TR 92-1

Abstract

The transfer of digital images between data archives and scientific workstations is likely to consume a significant amount of network bandwidth in the very near future. This paper examines the image transfer problem from an end-to-end perspective, that is, it describes a complete image transfer protocol that takes into account both the nature of digital imagery and the properties of the underlying network. Specifically, it describes a simple algorithm for encoding images into network packets in such a way that the receiver can recover from dropped packets without requiring the sender to retransmit them. Avoiding retransmissions has the advantages of improving response time and eliminating the need to buffer data at the sender.

July 2, 1992

Department of Computer Science
The University of Arizona
Tucson, AZ 85721

[1]This work supported in part by National Science Foundation Grant IRI-15407 and DARPA Contract DABT63-91-C-0030.

1 Introduction

In the near future, members of the scientific community will require the ability to transfer extremely large amounts of digital imagery—still images typically ranging in size from 1MByte to 25MBytes. Earth observing platforms will be collecting data continuously at a rate of several hundred gigabytes per day [7], with researchers around the world performing interdisciplinary, multi-sensor experiments using data from distributed archives. At this collection rate, these users could become one of the largest consumers of wide-area network bandwidth. In order to use this volume of data effectively, it is necessary to give researchers the fastest possible transfer methods.

This paper addresses the problem of effectively transferring digital images across high-speed, wide-area networks. In contrast to the way images are currently transferred using general purpose bulk transfer protocols [11, 3, 4], our approach is based on an end-to-end design that considers all aspects of the problem, including user requirements, characteristics of the data, data compression, and network performance. In other words, rather than use protocols designed to reliably transport arbitrary data, our design is tailored to the transport of digital images. Although the techniques described in this paper are general enough to apply to all types of images, for concreteness we focus on satellite imagery. Consider the following two properties of satellite imagery, and how our image transfer mechanism takes advantage of them.

First, almost every aspect of remote sensing involves high degrees of uncertainty. Atmospheric turbulence, variations in solar illumination, and viewing geometry all influence the image formation process. Furthermore, errors are introduced by sensor optics and detectors, as well as during down-link transmission. At typical image resolutions, there is also a high degree of surface brightness variation due to differences in material reflectance, topography, lighting, climate, and so on. The net result is that the remote sensing community has designed applications that are able to deal with systematic uncertainty in the input data. For example, browsing large databases for candidate tests images, quicklook viewing prior to sensor activation, and large scale thematic mapping are applications that are particularly tolerant of imperfect data. Because the data already contains substantial errors and the applications can tolerate these errors, our image transfer protocol is able to relax the reliability guarantees of conventional bulk transfer protocols.

The second significant property of satellite images is spatial redundancy, or highly correlated brightnesses within local neighborhoods. When observed from space, surface brightnesses tends to change gradually in most scenes. Additionally, many of the same processes that contribute to image uncertainty do so in such a way as to smooth out irregularities at the pixel and local neighborhood levels—individual pixels actually represent an average brightness of the underlying substances. Because the images contain redundant information, our image transfer protocol is able to recover from errors (i.e., dropped packets) introduced by the network. This is very much like forward error correction [2], except we take advantage of the spatial redundancy already present in the images instead of explicitly adding redundant information to the data prior to transmission.

Our approach is to not retransmit lost packets, but instead to use the spatial redundancy in the data to recover the missing information. Although intuitively simple, our design is complicated by three factors. First, it is necessary to minimize the impact of each lost packet on image quality. This is accomplished by encoding images in such a way as to insure that the pixels contained in a single packet are scattered throughout the image. Then, when a packet is lost, redundancy in the remaining image data can be used to reconstruct lost information. Second, in order to support a wide range of network bandwidths, it is sometimes necessary to compress the data. Therefore, our encoding algorithm also preserves those qualities of the image that support data compression. Third, we have designed our protocol in such a way that all packets can be processed independently and in any order [5]. This has benefits with respect to parallel

protocol design and overall system simplification.

The key characteristic of our design is that by properly encoding images in network packets, we are able to avoid the retransmission of lost packets. Avoiding retransmission has two important advantages.

- Retransmission-free protocols can improve end-to-end response time. Although raw network bandwidth is improving, the effective throughput rate of a retransmit-based protocol will not improve proportionally if the network suffers from either: (1) high latency, or (2) significant packet loss. For example, a 1Gbps network is able to transport a 10Mbyte image in 100ms. If even one packet needs to be retransmitted, then the response time doubles on a wide-area network with 100ms round-trip latency. Should a significant number of packets be dropped, then multiple round-trip times may be necessary to successfully transmit the data, further degrading performance.

- Retransmission-free protocols drastically reduce the buffer space required on the sending machine. Protocols that retransmit lost packets must buffer all transmitted data until it has been acknowledged. For example, on a 1Gbps network with 100ms round trip latency, a reliable protocol like TCP would either need to maintain 12.5 Mbytes of buffer space for each image it is transmitting, or reduce its window size, and consequently, its effective throughput rate.

Note that although this paper focuses on the transmission of digital satellite imagery, we also have some experience with other types of digital images, including digitized text and bit-maps. In particular, we plan to use the protocol to transfer journal page images as part of a nationwide "collaboratory" for a community of Molecular Biologists [13]. We also believe the techniques presented are applicable to the transfer of real-time video, speech, music, and instrument data. Finally, even though we concentrate on the transfer of images between a data archive and an end user's workstation, the same techniques could be applied to the transfer of images form remote sensing devices (e.g., satellites) to the archives.[1]

2 Algorithms

This section outlines a suite of algorithms for efficiently encoding digital images into network packets so as to maximize information recovery when packets are dropped. Intuitively, our approach is to eliminate the temporal to spatial relationship among packets. Furthermore, because our image transfer protocol is designed to work on networks with a variety of bandwidths, our algorithms are constrained by the need to preserve those properties of images required to support data compression.

Information that is lost during transmission must be recovered through some other means. This is accomplished by encoding the source images in such a way as to insure that information that is local in "image space" is not local in "packet space". Specifically, we have devised an encoding scheme that guarantees that no two adjacent image pixels will ever be in the same network packet, or more strongly, in two packets within the same group of N transmitted packets. When a packet is lost, the corresponding lost bytes will be scattered over the full image. This provides the optimal conditions under which to run various reconstruction algorithms around the missing pixels.

[1] Network latency can be a significant issue in such cases. For example, the Venus-to-Earth latency is over 4 minutes. The protocol currently used to transfer images from Magellan to Earth does not retransmit lost data, but at the expense of worse image quality than could be achieved by the encoding algorithm described in this paper.

Because response time is our primary concern, and all recovery operations must be performed on-line, it is important to limit their execution time. Additionally, it is desirable to be able to process incoming data incrementally and in parallel. This is accomplished by decomposing the source image into independent packets, each of which contains enough information to determine its correct location in the destination image. Even though packets are transmitted sequentially, there is no temporal relationship between adjacent packets. The loss of one or more consecutive packets does not prevent the protocol from processing subsequent packets. Loss bursts due to network congestion become less important because it is *average* packet loss rate that ultimately determines image quality. This temporal robustness allows us to adjust the transmission rate with less sensitivity to short term congestion conditions.

Before presenting the encoding algorithm itself, we first consider the issues involved in data compression and image recovery. Note that the purpose of this section is to identify the key parameters of the system. The next section puts the pieces together by describing the complete image transfer protocol.

2.1 Compression Algorithms

The question of whether image compression is appropriate for a particular application is one that can only be answered by considering network bandwidth, quantity of data to be transmitted, and available computer processing power of the communicating hosts. Consequently, it is important to look at the interaction of a variety of image compression techniques with our encoding algorithm. The concepts employed in our image encoding algorithms rely on many of the same properties of images that are employed in compression techniques. Also, a system design that handles all packets independently may affect the performance of certain compression techniques. If compression is required for a particular network configuration, then it is important to insure that the encoding scheme being used does not destroy the potential to achieve significant compression results.

The key to obtaining significant compression in digital imagery is to exploit local redundancy. In the image processing community, redundancy is expressed using the autocorrelation function, which measures the similarity between samples as a function of separation (lag). Autocorrelation can be computed in both the horizontal and vertical directions for stationary imagery, and in the inter-frame dimension for video data. Typically, the function has the same shape in all three directions: it drops off exponentially from a lag of zero and peaks again when the lag is the length of the full line (RowLength), which corresponds to vertical correlation. Figure 1 plots the horizontal autocorrelation function for a typical satellite image. In other words, compression algorithms work best when run on consecutive bytes within the same row or column, with decreasing performance as the distance between pixels increases. Due to the radially symmetric nature of image autocorrelation, they still behave reasonably well along diagonal tracks across the image.

There are two basic classes of compression algorithms: loss-less and lossy. Loss-less compression algorithms are capable of recovering all original information during the decompression phase. The cost of this complete data recovery capability is a limitation in the maximum achievable compression ratio—the best that can be achieved by loss-less algorithms on image data is typically 2:1. Lossy algorithms are employed when higher compression ratios are required. Using lossy algorithms, it is not difficult to obtain compression ratios of 5:1, with much higher rates under more ideal circumstances. The cost of achieving this higher compression ratio is increased computational requirements. The most commonly used lossy compression