Multiple Target Tracking Across Reliable Multicast Wireless Networks

James Griffioen W. Brent Seales

Department of Computer Science
University of Kentucky
773 Anderson Hall
Lexington, KY 40506-0046
{griff,seales}@dcs.uky.edu

Rajendra Yavatkar

Communication Architecture Lab
Intel Corporation, JF3-206
2111 NE 25th Ave
Hillsboro, OR 97124
yavatkar@ibeam.intel.com

ABSTRACT

This paper presents a novel approach to high-performance video processing that avoids the use of special purpose hardware and achieves parallelism via a distributed system of high-performance workstations. An efficient network protocol allows mobile computers or mobile robots to remotely access the distributed video processing system over low-bandwidth wireless links. Despite the lack of special purpose hardware and expensive computing equipment, the system provides exceptional performance processing data at speeds that approach real-time.

The novel contribution of our approach is the ability to process video in compressed format and efficiently distribute video data to a distributed system of workstations for processing. We describe the algorithms employed to achieve these features and demonstrate our approach by showing how it can be applied to two problem domains: multiple target tracking and video archive searching. We present experimental results that show that video data can be efficiently disseminated to a network or workstations with only a small (7%) increase in transfer time. We also show that processing can be performed efficiently (20 fps) directly on the compressed video without special purpose hardware.

Keywords: Wireless, Multicast, Multimedia, Recognition

1 Introduction

Video data is rapidly becoming an integral part of many application domains. The shift towards the use of video data has elevated the need for high-performance video processing techniques. Unfortunately the majority of video processing techniques and solutions are computationally intensive. Conventional approaches often employ special purpose image processing hardware to perform the processing. Other systems use expensive (and often

This research was supported by the National Science Foundation under Grants CDA-9320179, CDA-9502645, IRI-9308415, and the Advanced Research Projects Agency under grant DAAH04-96-1-0327.

special purpose) large-scale parallel multiprocessor systems to achieve the desired performance.[12] Parallel systems that equip all processors with special purpose image processing hardware only amplify the cost.

This paper presents a novel approach to high-performance video processing that avoids the use of special purpose hardware and achieves parallelism via a distributed system of high-performance workstations. We also describe an efficient multicast protocol that allows mobile computers or robots to remotely access the distributed video processing system via slow-speed wireless links. Despite the lack of special purpose hardware and expensive computing equipment, the system provides exceptional performance processing data at speeds that approach real-time.

The novel contribution of our approach is the ability to (1) process video in compressed format and (2) efficiently distribute video data to a distributed system of workstations for processing. Working with data in compressed format has several advantages. It reduces the space required to store the data and the network bandwidth required to transmit the data. It also reduces the processing time needed to recognize/locate objects in the video stream, taking advantage of the work already done by inexpensive compression hardware on the video capture board. Efficient dissemination of the video means that video data can be obtained at any machine on the wired or wireless network and quickly transmitted to a group of workstations that process the video in parallel.

There are a wide range of applications that can benefit from our solutions to the problems of efficient video distribution and compressed video processing. In this paper we illustrate the applicability of our design using two different but similar problem domains: *multiple-target-tracking* in mobile systems, and *video searching* of archived video clips.

1.1 The Multiple Target Tracking Problem

Multiple-target-tracking in a mobile system involves the use of a wired distributed system of workstations to simultaneously track multiple targets in the video transmissions emanating from mobile robots (computers) connected via a wireless network. The basic approach we take is to distribute the live video across the wireless network to a set of wired machines, where each machine tracks a distinct object. The tracked results are then sent to a single node which composites the results of all tracked targets.

In this model, the camera that collects the data is mounted on a relatively slow, resource-limited, mobile computer node, which is incapable of performing the video-processing task in real-time itself. In addition, the mobile unit is linked to the network via wireless technology, providing positional flexibility but restricting communication bandwidth.

1.2 Video Archive Searching

Video archive searching refers to the problem of searching an archive of compressed video for a specified model. The video server on which the archive resides is incapable of satisfying video access requests while simultaneously decompressing and searching large videos for the specified models. An efficient solution similar to the tracking problem can be used. To achieve high-speed model identification, the video server distributes the video in its compressed form to a secondary set of nodes where each node is instructed to search for a single model or multiple models. The greater the parallelism the better the performance. Consequently, we envision large-scale distributed processing systems consisting of many secondary processing nodes that must all receive the data. Even though the data may traverse high-bandwidth wired networks, we cannot assume the bandwidth will be able to support multiple communication circuits each carrying a copy of the video stream.

2 Related Work

The problem of object identification and tracking is fundamental in the computer vision and image processing literature.[1,8,6] Most tracking algorithms can be divided into two categories based on the type of model used for matching: area-based methods, and feature-based methods. Area-based methods rely on 2-D templates in some form to provide a model and a matching criterion.[3] Feature-based methods preprocess the image in order to extract and track a model which is constructed as a combination of higher-level features.[5]

Our approach to tracking is area-based and uses template-matching. One novel aspect of our tracking algorithm is that we solve the problem in the compressed domain.[21,20] Other research on compressed domain processing has concentrated mostly on video effects, filtering and cut detection, and very little has been done with vision algorithms.[22,2] Note that our approach is similar to frequency-domain methods for focus[14] and motion analysis,[17] but we obtain the frequency information very efficiently from the compression process.

A second novel aspect of our work is the distributed solution to video processing problems provided by the reliable multicast protocol. While there has been significant work in (reliable) multicast protocols,[7,11,19,16,24,4] we are unaware of related work which uses multicasting to solve video processing tasks such as multiple-target tracking in a distributed setting.

3 Reliable Multicast Communication

3.1 Objectives

Multiple target tracking and video archive searching both require a communication protocol that can efficiently transmit data from a single source to multiple receivers. Because the wireless link has limited bandwidth and the mobile host has limited processor cycles available, the protocol should use as little bandwidth as possible, should avoid increasing the mobile host's processing load, but yet deliver data as quickly as possible to all receivers. More specifically, the protocol must solve the following communication problems:

Reliable and Delay Sensitive Delivery: To ensure the distributed tracking system/video search engine receives the entire video feed, the communication protocol must reliably deliver the data to all receivers. To increase throughput and reduce end-to-end transmission delay, errors such as packet loss at routers or receivers must be corrected as close to the source of the error as possible to reduce latency.

Bandwidth Conservation: Because wireless networks have limited bandwidth[1], the communication protocol must minimize the amount of bandwidth used to transmit data to all receivers. Even wired networks have bandwidth limits that may not be large enough to support concurrent communication channels to multiple receivers. Therefore, a multicast communication protocol is necessary.

Although retransmissions are necessary to ensure reliable delivery, the system should try to limit the bandwidth consumed by retransmissions. Consequently, retransmissions should also use multicasting but should restrict retransmissions to the area of the network where the error occurred.

Immediate Error Recovery: To reduce the end-to-end latency and ensure that all receivers get that data at approximately the same time, the communication protocol needs to detect errors as soon as they occur and must begin the recovery process as soon as the error is detected.

[1] Conventional 802.11 wireless networks operate at approximately 2 Mbps as opposed to the 10 Mbps and 100 Mbps speeds of conventional and fast ethernet.

Load Distribution: To prevent communication overhead from consuming excessive processing power on the sender (mobile host), the communication protocol should offload as much of the reliable delivery duties to wired hosts as possible. Similarly, given the high-demand on a video server, the communication protocol should consume as few CPU cycles on the video server as possible.

Scalability: To optimize performance, the system should use as many machines as possible to analyze the incoming video stream. As the system scales to tens or hundreds of nodes, the communication protocol will need to scale to prevent the network from becoming the bottleneck to performance.

Reducing Wireless Packet Loss: The majority of packet losses in both wired and wireless network occur as a result of buffer overflows at routers or receivers. Statistically, multicasting data to several recipients has a higher probability for packet loss than unicasting data to a single recipient because there are more receivers that can drop the packet. Thus it is advantageous to use unicast communication across the wireless link, where lost packets and retransmissions are particularly costly, while using multicast communication across the wired network.

3.2 The TMTP protocol

We are currently designing and evaluating the TMTP (Tree-based Multicast Transport Protocol) reliable multicast protocol which addresses the problems described in the previous section. The following briefly outlines the features of TMTP that meet the communication requirements listed above for our distributed tracking/search system. A more detailed description of the TMTP protocol can be found in [25].

TMTP provides scalable reliable multicast communication from any member of a group to all other members of the group (M $\times$ N communication). TMTP runs on top of any unreliable multicast packet delivery mechanism (such as IP multicast forwarding) and enhances the underlying multicast mechanism by adding scalable error detection and retransmission mechanisms to ensure reliable delivery. In particular, TMTP arranges the receivers into a hierarchical tree structure (called the *control tree*) and distributes the task of error recovery across the tree instead of placing the entire error recovery task on the sender. Data is initially transmitted using the underlying unreliable multicast protocol while the control tree handles all retransmissions. Information about active TMTP multicast sessions is stored with a session directory service similar to the *sd* session directory from LBL.[13] TMTP also supports dynamic group membership, allowing senders and receivers to join or leave streams already in progress, but our system does not currently use this feature.

To achieve greater scalability, TMTP organizes the set of receivers into distinct groups according to the multicast subnet on which they reside. Each group is called a *domain*. Typically, all receivers on the same subnet belong to the same domain and a single *domain manger* acts as a representative for all receivers in the domain. The domain manager is responsible for recovering from errors and handling local retransmissions if one or more of the group members within its domain do not receive some packets.

In addition to handling error recovery for the local domain, domain managers also provide error recovery for other domain managers in the vicinity. Localized error recover has several advantages. First, it offloads the work of recovery from the sender onto nearby domain managers. Second, it restricts the retransmission to the subnets where the error occurred. Localized recover is particularly important for a wireless distributed tracking system where the network bandwidth over the wireless link is limited and communication overhead (processing load) imposed on the mobile host must be as low as possible. Localized recovery restricts the entire recovery process to the area of the network where the error occurred, only consuming bandwidth and processor cycles in that area. Consequently, retransmissions across the wireless link only occur if localized recovery fails and a retransmission for the mobile unit becomes necessary. The goal is that recovery from lost packet should occur entirely on the wired network, regardless of the number of receivers.

To support localized error recover, the domain managers are organized into a *control tree* as shown in Figure 1.

The domain manager on the network where the sender resides serves as the root of the tree and has at most K domain managers as children. Similarly, each internal domain manager will accept at most K other domain managers as children, resulting in a tree with maximum degree K. The value of K is chosen at the time the multicast channel is established and only counts domain manager children (not the number of receivers on the subnet served by the domain manager). The degree of the tree (K) limits the processing load on the sender and the internal nodes of the control tree.

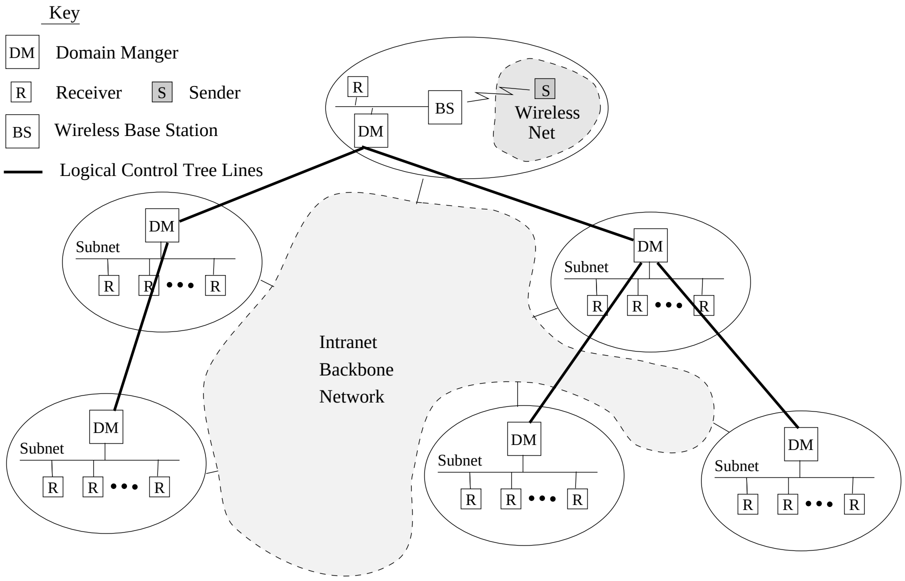

Figure 1: An example control tree for a wireless network connected to an intranet were the video processing/tracking machines are located. Assuming the wireless network is an extension of a wired subnet (i.e., they are logically on the same subnet and share a domain manager), the root domain manager can be placed on a wired or wireless machine. Placing the root domain manager on a wired machine ensures that all multicast retransmission occur over the wired network and all communication over the wireless network uses unicast.

Packet transmission in TMTP proceeds as follows. When a sender wishes to transmit data, TMTP uses the underlying unreliable multicast protocol to transmit packets to all receivers. Data is sent at a fixed rate combined with the use of a sliding-window-based protocol for flow control. Each node of the control tree (including the root) is only responsible for handling the errors that arise in its immediate K children. Likewise, children only send periodic, positive acknowledgments to their immediate parent, not the sender. When a parent receives an acknowledgment from its child, the parent knows that the child received the packet and will take on the responsibility of ensure that the packet is reliably delivered to all the child's children, grand-children, etc.

As soon as a receiver or domain manager detects that it missed a packet, it multicasts a NACK packet to start the recovery process as soon as possible. NACK suppression is used to prevent all the machines in a domain from sending a NACK at the same time. A domain manager that receives a NACK from one of its children immediately multicasts the retransmission to all its children in case other children also lost the packet. To limit the scope of both the multicast NACK and the ensuing multicast retransmission, TMTP uses the *Time-To-Live* (TTL) field of the underlying multicast protocol to restrict the transmission radius of the message. As a result, error recovery is completely localized. Typically, each part of an intranet would be organized into separate subtrees so that error recovery in a region can proceed independently without causing additional traffic in other regions. TMTP's hierarchical structure also reduces the end-to-end delay because the retransmission requests need not propagate all the way back to the original sender. Local retransmission also gets the data to the receivers faster than if the sender had to retransmit.

Although our distributed tracking/search system only requires 1 $\times$ N communication, TMTP's M $\times$ N capabilities turn out to be useful if unicast communication is desired between the wireless node and the wired network. The M $\times$ N facility in TMTP aggregates all transmissions from all senders in a domain through the domain's manager. As a result, senders transmit their data to the domain manager using unicast transmission. The domain manager then combines the data and multicasts it to all receivers using a single multicast stream. Thus all senders share the same control tree. Using the M $\times$ N feature, we can place the root domain manager on the wired network and have the mobile unit communicate across the wireless link to the root domain manager using a unicast protocol. This ensures that all multicast retransmissions will occur solely on the wired network rather than the wireless.

3.3 Concast Communication

Once a receiver has identified the desired object in the video stream, it must transmit the location information and/or control information back to the mobile host on the wireless network. Because multiple targets are being tracked simultaneously, all receivers will be processing the image and sending back location information at the same time.

To support efficient delivery of location information back to the mobile machine, TMTP provides *concast* communication. Concast communication is a communication model in which N senders transmit data to a single receiver. There are a variety of requirements that may or may not be needed by the senders/receiver[2]. In the distributed target tracking environment, concast messages must be reliably delivered to the receiver (mobile host) so that is will not miss a control operation and begin moving in the wrong direction. Similarly, for a distributed video search engine, the results of the search must reliably reach the machine that issued the search (which will not be the video server who sent the video stream). In addition, given the limited bandwidth of the wireless link, it is advantageous to merge messages whenever possible to reduce the header-to-data ratio.

TMTP supports concast communication using a hierarchical tree of domain managers to collect concast messages and forward them on to the receiver. To ensure reliable delivery, the receiver sends ACKS to every sender. The sender retransmits data if the acknowledgment is not received within a certain timeout period. To limit the bandwidth and processing overhead associated with ACK transmission, TMTP reliably transmits ACK packets to all senders using the reliable multicast techniques described in the previous section.

In the distributed tracking environment, domain managers that receive concast messages concatenate the location and control information from all packets into a single packet which it then forwards to its parent in the tree. This results in a single large message with a low header-to-data ratio as opposed to a large number of small messages with a high header-to-data ratio. This reduces the bandwidth consumed over the wireless link and the processing demands placed on the mobile unit. In addition, since results are being produced at regular intervals, each domain manager waits a for a fixed period of time before combining all packets received during the past interval. The delay gives the domain manager time to collect several messages, reduces the number of messages sent to the parent, and puts a bound on the transmission delay.

4 Compressed-Domain Tracker

Object tracking is the task of locking onto a known model of an object in an image frame and then identifying it in subsequent frames. The task of first locating the object is similar to the problem of object recognition. After the object is located in the initial frame, the search to identify it in subsequent frames is simplified because of simple restrictions on where the object could have moved between frame captures. Multiple object tracking (MOT) is the task of tracking multiple, unrelated and independent objects in a video sequence. Because the objects may move

[2] Various type of concast communication are outlined in [9] .

freely and without restrictions relative to each other, tracking more than one object is essentially an independent problem.

Our approach to the MOT problem is to exploit the independence of the problem by distributing the task to many independent nodes. Each node is assigned the task of tracking a single object and there is little or no communication required between nodes. However, each node must have access to the image data. Consequently we use multicast communication to deliver the video reliably to all nodes at approximately the same cost that it takes to deliver the data to a single node. Once nodes obtain the video frames and identify the object location, the position is returned to a single designated host via the concast mechanism (section 3.3).

Although remote processing of the video stream allows for greater parallelism, network bandwidth limitations require that video data be captured and transmitted in compressed format. Compression (together with original image scale) and frame rate are the two parameters of control which directly influence the bandwidth consumed. Conventional wireless networks limit bandwidth to 0 to 2 Mbps. These overall bandwidth constraints impose frame rate limits which we can adjust to some degree by selecting higher or lower levels of compression. By combining video compression with multicast communication, we can deliver the equivalent of 20-30 frames per second to many nodes, even while sending across a 2 Mbps wireless link. Although compression and multicasting solve the problem of data distribution, it creates a problem in that the distributed processing nodes must now track objects in a compressed video stream.

Given this context, we now show how we efficiently solve the compressed video tracking problem at a single node. The data arrives in compressed form, dictated by the particular compression standard being used. In the case of motion JPEG, each frame of the video is a JPEG image compressed according to the JPEG standard.[23,15] Our algorithms operate on the JPEG compressed image at the run-length-encoded stage, which implies that only the very last step of the compression must be undone. The last step, Huffman coding, is one of the least expensive steps in the compression process and can be undone with little cost to produce run-length-encoded data. The resulting data represents the runlength-encoded coefficients of the quantized discrete cosine transform (DCT). Note that the JPEG algorithm divides images into 8×8 blocks for coding. We therefore operate with model templates which are always aligned on a block boundary, corresponding to a resolution of 8×8 pixels.

4.1 Models and Matching

The tracking problem requires a model of the object to be detected in the video. We have two ways of representing this model in order to perform the subsequent matching between compressed images and the pre-stored template. The first, simple approach is to store the run-length-encoded DCT coefficients of a single template to be detected. This template, specified initially as a pixel image, is divided into 8×8 blocks and encoded with the JPEG algorithm to the run-length-encoded step. The template is then compared with candidate template regions in the image using a simple Euclidean distance metric[2,18,20]:

$$\|\mathbf{x}_1 - \mathbf{x}_2\|^2 = (\mathbf{x}_1 - \mathbf{x}_2)\cdot(\mathbf{x}_1 - \mathbf{x}_2) \tag{1}$$

Note that this equation becomes

$$\|\mathbf{x}_1 - \mathbf{x}_2\|^2 = \|\mathbf{x}_1\|^2 + \|\mathbf{x}_2\|^2 - 2\mathbf{x}_1\cdot\mathbf{x}_2 \tag{2}$$

Eq. (2) relates the Euclidean distance between two vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ to their cross-correlation, which is the term $\mathbf{x}_1\cdot\mathbf{x}_2$. When $\|\mathbf{x}_1\| = \|\mathbf{x}_2\| = 1$, Eq. (2) becomes

$$\|\mathbf{x}_1 - \mathbf{x}_2\|^2 = 2(1 - \mathbf{x}_1\cdot\mathbf{x}_2) \tag{3}$$

This shows that the squared distance between two input vectors is a measure of the correlation between those vectors. Vectors that are close to each other in the Euclidean sense are highly correlated.

All, or a selected set, of the DCT coefficients can be included in the Euclidean distance computation. For a set of candidate template positions, the position of minimum distance to the precomputed template is declared the best match. Note that this Euclidean distance is essentially a correlation measure and can be computed with the run-length-encoded DCT coefficients directly, avoiding the need to decompress frames of the video sequence (beyond the Huffman decoding step).

In some cases where the target to be tracked is scale and rotationally invariant, this method is adequate. However, any change in the target over time will cause mismatches to occur because the candidate templates will no longer match the single stored example of the model. For example, a scale change of a 2-D object or a slightly different view of a 3-D object will cause the method to fail.

In the general case we use a second, more robust method, based on the method of Principal Components. We can encode a large set of model templates via their principal components.[20] These principal components are computed directly from run-length-encoded templates. This gives the ability to compare a candidate template, which is a run-length-encoded set of DCT coefficients, directly to the stored templates via the the principal components. This method yields two important benefits. First, it can still be applied directly to compressed data. Second, the model is now a *set* of templates rather than a single template. This gives robustness over changes in a target's appearance since scale, rotation and appearance changes can be precomputed in the set of templates from which the principal components are extracted.

4.2 Algorithm

Figure 2 illustrates how templates are positioned in the tracking algorithm we use to obtain the results reported in Section 5. First, the principal components of a set of templates are computed and stored at the node designated for that model. When the first frames arrive, the node attempts to exhaustively test all positions in the frame for the best candidate match. The metric used is Euclidean distance in the space spanned by the eigenvectors of the principal components of the template set. Once an initial position is identified, a local neighborhood centered at that position is searched in subsequent frames. Within the search neighborhood, the best match, which is specified in terms of the minimum Euclidean distance over all candidate templates, is selected as the target location in the next frame. Given the movement of the target over time, any kind of predictive filter (such as the Kalman filter[10,6]) can be used to improve the search algorithm in subsequent frames.

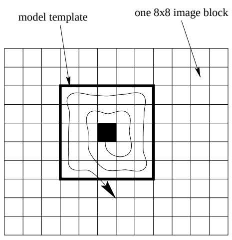

Figure 2: The 5×5 model template (measured in blocks and shown in bold) is positioned in the image over the center of the location to be matched. The template is always positioned on an 8×8 block boundary since image blocks are never fully decompressed. Subsequence candidate positions in the image are matched in a spiral pattern until either a timer runs out or a specified window size has been searched.

We report the particulars of one test sequence in Section 5. The number of template matches allowable per frame is dictated by the application's required frame rate. We have found that an image can be exhaustively/completely searched at a rate of one frame per second, while a 5×5 neighborhood (i.e. 16 template positions) can be searched at a rate of 20 frames per second. These numbers are highly dependent on processor speed, memory, and implementation particulars. Note that the 5×5 neighborhood is specified in terms of blocks, where each block represents 8×8 pixels. Thus the 5×5 neighborhood represents a 40×40 pixel region, which is a substantial displacement at 20 frames per second. Our tests have shown that the compressed-domain method gives a speedup of 3-5 times the number of matches possible with pixel-domain (template-based) matching. The speedup comes from the fact that we avoid decompression (in particular, the expense of the inverse DCT) and we operate on sparse data which has been run-length-encoded.

5 Experimental Results

5.1 TMTP Performance

We have implemented a prototype version of the TMTP protocol that runs on Solaris 2.x SPARCstations. The prototype uses IP multicasting as the underlying unreliable multicast protocol and we have run tests using The Internet MBONE and also a local/private intranet MBONE. To mimic the multiple target tracking environment, we set up a network organization similar to the one shown in Figure 3. We simulated the mobile host and wireless connection using a Sun workstation on the wired networked as the sender, configured to send at slower wireless network speeds. Although the numbers reported here were taken from our simulated wireless link, we have run some preliminary experiments with a still-in-progress Linux port of the TMTP system running on a Pentium laptop and transmitting over a 2 Mbps Wavelan wireless network. Preliminary results from the true wireless tests show that the simulation results are representative of a real wireless network environment. The root domain manager ran on a Sun workstation on the wired network and used unicast communication between the sender and the root domain manager. We have also run with the root domain manager on the same machine as the sender and observed similar results. Ten receiver machines were located throughout the intranet as shown in Figure 3.

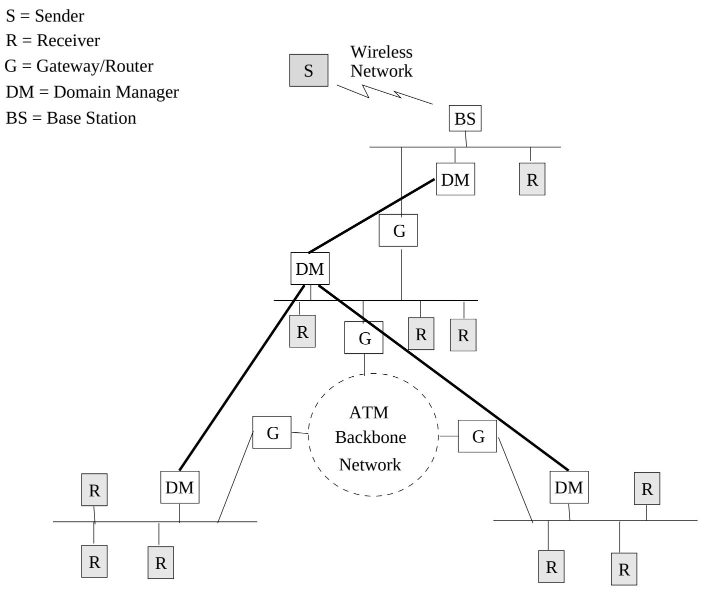

Figure 3: The intranet and domain manager organization used in all tests.

TMTP uses a rate based flow control, so we ran experiments using three different transmission rates, 0.5 Mbps, 1 Mbps, and 2 Mbps, representative of the video rates needed by compressed video connections. Each test involved multicasting a video clip at the specified rate to the 11 receivers (tracking machines). Each test was run 20 times and then averaged. Tests were performed during the day while the intranet was in standard use with general user traffic, experiencing typical packet loss and congestion.

To measure the added overhead/delay of multicast as opposed to unicasting the video stream, we measured

Sender's Transfer Rate	Multicast 200KB Msg 10 Recvrs	Unicast 200KB Msg 1 Recvr	Theoretical 200KB Msg Time	Multicast to Unicast Ratio	Multicast to Theoretical Ratio	% pkts retrans mitted
0.5 Mbps	4357 ms	4055 ms	3277 ms	1.07	1.33	0.0 %
1.0 Mbps	2141 ms	2009 ms	1638 ms	1.06	1.31	0.1 %
2.0 Mbps	1160 ms	1153 ms	819 ms	1.01	1.42	0.4 %

Table 1: The table shows that the times required to deliver a 200 KB message to 10 receivers using multicasting are within a factor of 1.07 of the times required to deliver the data to a single receiver using unicasting.

the time required to multicast the video to 10 receivers and compared that to the time needed to unicast the video to a single receiver. As a base point, we also compared our times to the theoretical minimum time (assuming wire-speed transfers without any protocol overhead) which we computed as $\frac{Transfer\_Size}{Transfer\_Rate}$ (labeled as *Theoretical* in Table 1). The results are shown in Table 1. The times reported show the average amount to time needed to transmit 200 KB chunks of the video clip. At the lower transfer rates, no packets were lost. Even at the higher transfer rates, the number of retransmissions was relatively low and did not significantly affect performance. All retransmissions occurred on the wired network with no retransmissions occurring over the wireless link. In short, the additional delay/overhead needed to send to ten receivers instead of just one was at most a factor of 1.07 in our experiments.

5.2 Tracker Performance

We have implemented a prototype algorithm for tracking objects in the compressed-domain. The prototype system is implemented in C and C++ and uses the the JPEG libraries[15] to do the partial JPEG decoding. We have implemented the two template matching algorithms, the first based on a single template and the second using the principal component method in the compressed domain. For the test case we present here, the principal component method performed much better since it included a larger number of template variations.

The test video sequence contains three faces which were selected as the targets. Each of the three targets were modeled with the principal component method using 9 views at various orientations and scales. The tracking of the individual targets was distributed to three separate nodes, each with its own distinct target model. The video clip was delivered at 20 frames per second to these nodes via the TMTP multicast protocol.

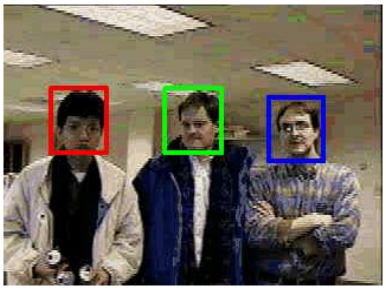

Figure 4: (left) One frame of the 149-frame test sequence. The three faces have been correctly identified in this frame. The tracker for each target executes independently on separate nodes of the network. (right) These nine views were selected as the model for the subject on the left of the video sequence.

Figure 4 (left) shows one view of the sequence. For this frame, the location of each target is identified with a bounding box. The sequence was captured using a hand-held Sony handycam and digitized via a SunVideo

	Left subject	Middle subject	Right subject
Missed frames	4 of 149	14 of 149	15 of 149
Miss percentage	2.7%	9.4%	10.1%

Table 2: Classification Accuracy

compression board. The sequence contains 149 frames and includes substantial camera and object motion. Figure 4 (right) shows the training set used for the person on the left in the sequence.

In this experiment, the model template size was 6×7 blocks. Each of the nine reference templates was shifted to produce an additional 64 templates. These shifts compensate for the JPEG 8×8 blocking factor and produce a total of 576 templates. From these templates, 20 principal components were used to classify candidate image regions. At the 6×7 block template size, we were able to perform 425 template matches per second on a MIPS R8000 processor. This rate corresponds to an exhaustive image search in one second or a 5×5 block search window at 20 frames per second. This represents a speedup of at least 3-4 times the image-domain template match within a comparable pixel neighborhood.[20]

Table 2 shows the classification accuracy for the three targets in the test sequence. Using the principal components from the nine selected templates as a model, we were able to classify each target correctly in 90% of the frames at 20 frames per second. The video sequence contains considerable variation in facial expression and head orientation:

6 Summary

This paper presents a framework for delivering video data to many hosts via multicast for the purpose of performing distributed video processing. We have shown that multicasting to many hosts is very efficient and introduces only a small amount of overhead compared to transmitting the video to just one host. We have used the multicast mechanism to implement a distributed solution to the multiple-target tracking problem. Each node in a pool of receivers with access to the multicast can independently track a target, and our tracking algorithm permits them to exploit the video compression scheme by performing the tracking algorithm directly in the compressed video. Nodes performing the tracking relay their results via the concast mechanism to a designated host, which assembles the results for all tracked targets. This paradigm is useful in a number of video processing contexts, including distributed content-based searches of large video archives.

We thank Babu Yama, Rajitha Sumanasekera, and Cheng Jiun Yuan for their hard work in performing many experiments and helping to implement the ideas in this paper. Finally, we gratefully acknowledge the support of this work by the National Science Foundation (grants CDA-9320179, CDA-9502645 and IRI-9308415) and the Advanced Research Projects Agency (grant DAAH 04-96-1-0327).

7 REFERENCES

[1] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

[2] J. Bethge. Object recognition for compressed data using template matching, August 1997. M.S. Thesis, University of Louisville, Department of Electrical Engineering.

[3] David Coombs and Christopher Brown. Real-time smooth pursuit tracking for a moving binocular robot. In *Proc. Computer Vision and Pattern Recognition*, pages 23–28, 1992.

[4] Stephen E. Deering and David R. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–109, May 1990.

[5] Rachid Deriche and Olivier D. Faugeras. Tracking Line Segments. In *Proceedings of the 1st ECCV*, pages 259–268. Springer Verlag, April 1990.

[6] O. D. Faugeras. *Three-Dimensional Computer Vision: A Geometric Approach*. MIT Press, 1993.

[7] S. Floyd, V. Jacobsen, S. McCanne, C-G Liu, and L. Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. In *Proceeding of the ACM SIGCOMM'95 Conference*, November 1995.

[8] R.C. Gonzales and R.E. Woods. *Digital Image Processing*. Addison-Wesley, 1993.

[9] James Griffioen and R. Yavatkar. Clique: A Toolkit for Group Communication Using IP Multicast. In *The Proceedings of the Workshop on Services in Distributed and Networked Environments*, June 1994.

[10] Per-Olof Gutman and Mordekhai Velger. Tracking Targets Using Adaptive Kalman Filtering. *IEEE Transactions on Aerospace and Electronic Systems*, 26(5):691–698, September 1991.

[11] H.W. Holbrook, S.K. Singhal, and D.R. Cheriton. Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation. In *Proceeding of the ACM SIGCOMM'95 Conference*, November 1995.

[12] K. Inoue, A. Nakamura, M. Nivat, A. Saoudi, and P. Wang. *Special Issue-Parallel Image Analysis and Processing (Ube, Japan, December 21-23, 1992), IJPRAI*, 8(2), 1992.

[13] Van Jacobson. SD: Session Directory. Technical report, Lawrence Berkeley Laboratory, March 1993.

[14] E. Krotkov. Focusing. *International Journal on Computer Vision*, 1:223–237, 1987.

[15] T. Lane, P. Gladsonte, L. Ortiz, and J. Boucher. The Independent JPEG group's JPEG software, Febuary 1996. Distribution available at archie site: *ftp.uu.net*.

[16] B. N. Levine, D. B. Lavo, and J. J. Garcia-Luna-Aceves. The Case for Concurrent Reliable Multicasting using Shared ACK Trees. In *Proceedings of the ACM Multimedia '96 Conference*, November 1996.

[17] A. Mitiche and P. Bouthemy. Computation and analysis of image motion: A synopsis of current problems and methods. *Int. Journal of Computer Vision*, 19(1):29–56, 1996.

[18] H. Murase and S. Nayar. Visual learning and recognition of 3-D objects from appearance. *International Journal of Computer Vision*, 14:5–24, 1995.

[19] S. Paul, K. Sabnani, J. Lin, and S. Bhattacharyya. Reliable Multicast Transport Protocol (RMTP). *To Appear in the IEEE Journal on Selected Areas of Communication*, 1996. (see also the Proceedings of IEEE INFOCOM'96).

[20] W.B. Seales, M.D. Cutts, C.J. Yuan, and W. Hu. Content analysis of compressed video. Technical Report 265-96, Computer Science Dept., University of Kentucky, Lexington, Kentucky, 1996.

[21] W.B. Seales, J. Lumpp, and C.J. Yuan. Model-based tracking in compressed video. In *IEEE Aerospace Applications Conference*, 1997.

[22] B. Smith. Survey of compressed domain processing techniques. In *Reconnecting Science and Humanities in Digital Libraries, a Digital Library Symposium at the University of Kentucky*, Oct 1995.

[23] G. Wallace. The JPEG still-picture compression standard. *Communications of the ACM*, 34(4):31–44, April 1991.

[24] B. Whetten, S. Kaplan, and T. Montgomery. A High Performance Totally Ordered Multicast Protocol, August 1994. ftp://research.ivy.nasa.gov/pub/docs/RMP/RMP_dagstuhl.ps.

[25] R. Yavatkar, J. Griffioen, and M. Sudan. A Reliable Dissemination Protocol for Interactive Collaborative Applications. In *The Proceedings of the ACM Multimedia '95 Conference*, pages 333–344, November 1995.