Efficient Real-Time Multicast Video Streaming Using Network Processors

Karthik Singaram L, Madhan G, Prasad R M, Ranjani Parthasarathi
Department of Computer Science and Engineering, Anna University, Chennai, India

***Abstract* —** *Streaming Real-time video to multiple heterogeneous clients is largely constrained by the available bandwidth. We propose an efficient method to solve the problem of real-time video streaming over heterogeneous networks to heterogeneous clients by using network processors. We use multicast transmission to save a large portion of the available bandwidth. We perform rate shaping by selective frame reject to adapt the system to the current state of the network by means of a feedback control mechanism. We use hierarchical encoding schemes to facilitate easy adaptation of the video content at the routers to suit the display capabilities of the various clients. The solution also fits seamlessly into the existing Internet architecture. The multicast routers alone keep track of vital information about the network traffic levels and clients' display capabilities.. We used a Network processor (IXP 2400) to implement the solution effectively due to its inherent support for network processing tasks, flexibility, speed, and parallelism, all of which is needed for efficiently solving the problem on hand.*

Keywords *— Multicasting, Real-Time Video Streaming, Network Processors, Heterogeneous Clients, Heterogeneous Networks, Capability Propagation, Rate Shaping, Content Adaptation, Resolution changing, Frame Rate Adjustment, Selective Frame Reject, Hierarchical Encoding, IXP 2400*

1. Introduction

Though rapid advancements in the communication technologies have made it possible to stream video content over the Internet, due to the facts that video content is usually very large and the available bandwidths are very small, real-time streaming is still a challenging task. The problem becomes all the more complex when heterogeneous clients of varying display capabilities like PDAs, mobile phones, desktop systems etc. are to be served. The challenge lies in maximizing the quality of the video transmitted to the clients while maintaining the real-time constraints [3].

Effective solutions to the problem of real-time video streaming to many clients must greatly reduce the latencies in the transmission while maximizing the quality of transmission. Interesting trade-offs begin to appear between transmission delays and the quality of video reaching the client.

The problem is further complicated if the underlying networks are heterogeneous. The data rates supported by the underlying networks will be different and these rates may not be high enough to send high quality video content at a fast rate to meet the real-time constraints. Therefore the size of the transmitted data must be reduced, as much as possible and at the same time the quality must be maintained as high as possible. The solution is to identify and remove the least important information that will cause minimal change in the quality of video [1, 14].

Further, the low-end clients will not be able to play the video at the same quality as the higher-end clients. Streaming higher quality video than the actual client capabilities will only lead to wastage of network bandwidth, as the clients will anyhow not be able to reproduce the same high quality due to their own limitations. We must exploit this and save the network bandwidth.

Thus the problem is to serve the right quality to the right clients while maintaining the transmission delay to a minimum to meet the real-time constraints. It is to this end that we propose to use network processors effectively.

Network processors, are used in many modern day routers due to their flexibility and speed. We propose to exploit this flexibility and the parallelism inherent in their architectures, to provide an efficient solution to the problem of multicast real-time video streaming to heterogeneous clients over heterogeneous networks.

2. Related work

Quality adjustment to heterogeneous clients over the Internet can be accomplished in two ways - static and dynamic [14]. The static approach involves storing multiple copies of the video content at the server. When the client requests the video content from the server, it also sends its capabilities along with its request. The server sends the video content of appropriate quality based on the clients' capabilities and the net available bandwidth across the heterogeneous network along the path to the client. Though this improves the response time, it is not suitable for multicasting and video conferencing like applications, and requires large extra storage at the server.

In the dynamic approach there can again be two standpoints - implementation at the server or at the router. In the server end implementation, on receiving the request from the client, the server changes the quality of the real-time video, based on the client's capabilities and the network traffic, and streams it to the client. While this saves on storage, latency is higher, and it does not support multicasting.

On the other hand, the implementation at the router end is a good solution to multicast real-time streaming video. In this implementation the clients join a multicast group.

The server sends the content at the highest quality as required by the multicast group. The intermediate routers dynamically modify the content by changing the quality of the video based on the available bandwidth and the network traffic in the path to the next hop.

Various methods have been explored to dynamically adapt the quality of the video content. The parameters that have been changed are the frame rate and picture quality.

Frame rate change typically involves dropping selective frames whose loss will affect the video quality to the least possible extent.

Picture quality is varied by removing the high frequency components. High frequency components can be removed because they cause little change in the quality as perceived by the human eye. This can be achieved using low pass filters at the intra frame level [1].

There are other parameters that can be varied namely the resolution and the color depth. We propose to change these parameters too at the routers based on the clients' capabilities.

3. Proposed Solution

Our solution differs from those in the literature, in that it adapts to both network bandwidth and the clients' display capabilities. It augments the multicasting algorithm with capability propagation.

Capability propagation: This works as follows. A client that wants to join a multicast group creates a request packet containing its capabilities and broadcasts the packet to its neighboring multicast routers. The multicast routers that come across this request packet will check their multicast routing tables for a link to the multicast group. If the multicast router does not have any link to the intended multicast address, it simply appends its address to a path field in the request packet and forwards the packet to its neighboring multicast routers. At every point of time the request packet thus maintains the list of routers in the route back to the client. If the multicast router has a link to the intended group then it creates a reply packet and forwards it back to the client along the path mentioned in the request packet. The multicast routers in the route back to the client shall make entries in their table about the path to the multicast group; the client receives this reply packet and thus becomes a part of the multicast group. The multicast router creating the reply packet also has to update the maximum quality of video, which it requires, based on the newly joined client's capabilities. If the multicast router is already getting higher quality than expected by the newly joining client, then it simply makes an entry for the new client in its multicast table. If the capabilities of the newly joining client are more than that which can be met by the current quality which the multicast router has registered for, then the multicast router makes an entry in its multicast table and it also sends an update message to the other neighboring multicast routers in the same multicast group, informing them of its new higher quality requirements.

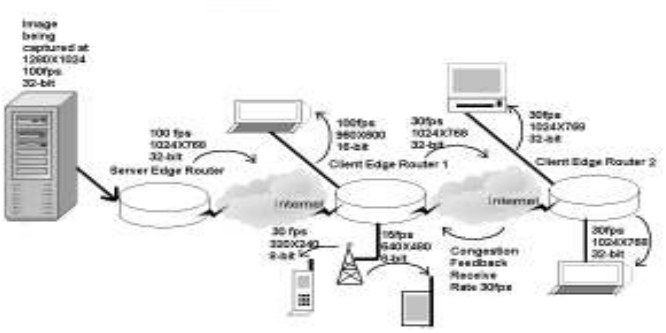

Fig.1 Overview of the proposed system

A similar procedure has to be adopted when a client is leaving the group. The above-described process ensures that all the intermediate routers that are part of the multicast group can maintain entries in their multicast routing tables regarding the quality with which they have to forward packets along each of the interfaces belonging to the particular multicast group. The process of joining and leaving the group is shown in Fig 2.

The other type of information maintained to perform quality adjustment is the network congestion level. Each multicast router starting from the one next to the server will periodically send a feedback report with information about the current frame rate along the link at which they are getting packets. The multicast router will use this to update its multicast table regarding the packet rates for each of interfaces along which it is going to forward packets belonging to the multicast group. Thus the multicast routers maintain information about both the network bandwidth, and the clients' capabilities.

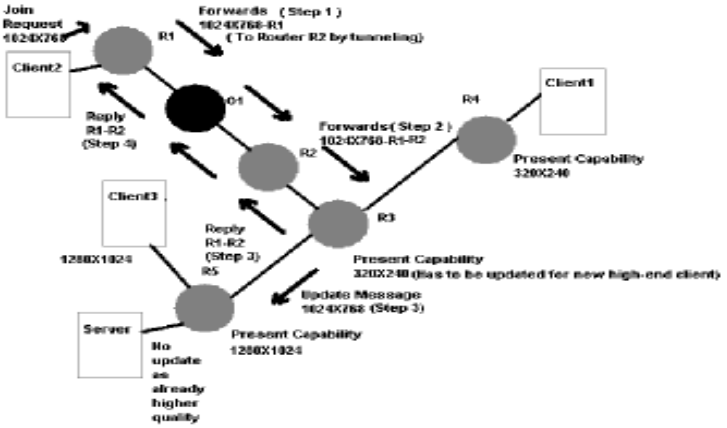

Fig.2 Client Request Processing

This solution does not disturb the normal working of the routers in any way. The multicast routers can transparently undertake the above-mentioned process. Whenever a multicast router or a client or a server wants to send any information to a neighboring multicast router, it can do so by using a simple tunneling mechanism and use the unicast address of the multicast router to forward the packets to it. This will ensure that the broadcast messages to multicast

routers, which have been discussed above, do not require any changes to the existing ordinary routers, which are not aware of multicasting. In essence, it can be integrated with any existing multicasting algorithm.

Selective Frame Reject:

We have seen how the information regarding the client's capabilities and the information about the network traffic are being maintained at each multicast router. In addition, the multicast routers measure the rate at which they are receiving frames for a multicast group. They use this information for frame-rate adaptation.

The video frames are sent with RTP headers. The RTP protocol provides fields like timestamp and sequence number, which we use for frame-rate adjustment.

In the case of MJPEG images with no inter frame dependencies, selective frame reject is achieved by choosing to drop frames in an interleaved, equally distributed manner. This can be accomplished by examining the RTP header for the sequence number field and deciding whether or not to drop the frame based on this field. In the case of MPEG formats we can divide the streams into layers - base and enrichment. The enrichment content can be dropped to minimize the loss in quality.

The justification for this selection of frames to reject is that it has to be done to match the rate required by the interface otherwise the routers along that path will experience congestion and therefore drop packets themselves. These routers shall do so at random or in the tail-drop fashion, either ways good quality cannot be ensured as a sequence of frames may get dropped, adversely affecting the video quality. When it comes to transmission using the MPEG video format, the selective frame reject has to make sure that the important frames are retained in the video. The motivation here is also the fact that otherwise the intermediate routers practicing random drop or tail drop may drop important information frames, as they are unaware of the implications and drop packets without examining their contents.

Resolution Changing: The other parameters for quality adaptation are resolution and color information based on the capabilities required along the interface. The important issue is that examining the whole packet for resolution changing must be avoided as much as possible. To facilitate easy resolution changing we propose to use Hierarchical-encoding strategies [11], which were originally proposed as a latency hiding technique. The advantage with hierarchical strategies is that the lower resolution content is sent first, then higher resolution information are sent successively. Therefore resolution changing can be easily accomplished by simply chopping off the tail of the frame contents as required at the intermediate multicast routers.

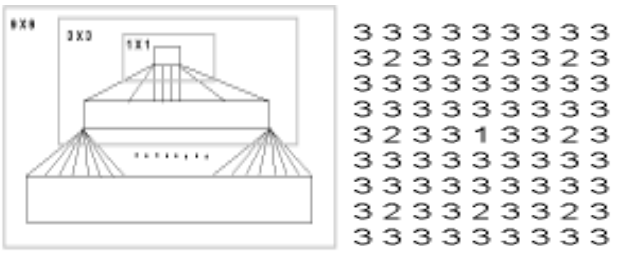

Fig. 3 Hierarchical Encoding

In the AVI format the Hierarchical encoding can be done as illustrated in the Fig. 3

We can see that first the center pixel is sent, then eight pixels surrounding it at even distances from the point and the corners of the image are chosen. Eight pixels are then chosen at equal distances for each of the pixels chosen in the previous step and so on. This form of hierarchical encoding ensures the easy chopping of frame contents to do resolution changing.

The Hierarchical JPEG format [11, 12] can be used in the case of using MJPEG encoding schemes. The hierarchical encoding schemes avoid a great overhead of scanning the entire packet once to do resolution changing which will be required if the ordinary encoding schemes are used.

The color depth reduction can be easily achieved on the fly by considering each frame as a bitmap image and simply scaling down the number of bits per pixel. An illustration of conversion from a 32-bit image to a 16-bit image would involve a simple scale down by a factor of 2 when transmitting the Red, Blue and Green values for the individual pixels.

4. Implementation

To evaluate the proposed design, we have implemented the multicast router on the IXP 2400 network processor [5,6]. The simulations were performed on the IXA developer's workbench and results were obtained from the statistics provided by the workbench. The IXP 2400 Architecture has 8 microengines and an Intel XScale Core. The microengines have hardware support for eight threads and are clocked at 400Mhz. The server and client applications were implemented using the C language. The major focus was on the design and efficient implementation at the multicast routers [6].

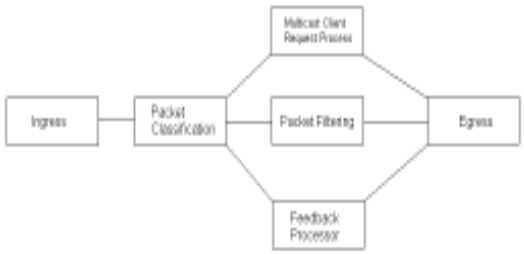

Fig.4 Design Pipeline

The design pipeline at the multicast routers is shown in Fig.4. The allocation of micro-engines to the various functions is as follows:

ME 0:0 Ingress; **ME 0:1** Packet Classification and Multicast Client Request Processing; **ME 0:2** Filtering and Feedback Packet Processing; **ME 0:3 and ME 1:1** Egress. The communication between the microengines is accomplished by means of scratch rings. Egress was replicated to balance the pipeline.

Ingress: -The ingress module takes care of getting the incoming packets and storing them in the SDRAM memory available in the IXP2400 Network processor [5][6].

Classification: -The classification module classifies the packets into our application's packets and the rest. The rest of the packets will go through an alternate flow to reach the egress; this is not shown as a part of the main design pipeline. The packets belonging to our application can be further divided into the Multicast Client Requests, Feedback Packets and Video Packets to be transcoded.

Multicast Client Request Processing: -This module takes care of handling the requests of the various clients joining the multicast group and the clients leaving the multicast group. It also handles update capability messages from the downstream clients and forwards them appropriately. On receiving a request packet it has to check the multicast table to see if it has a connection to the multicast group. If not, then it just appends its address and forwards the request to the neighboring multicast routers. If the router has a link to the multicast group then it sends a reply along the route obtained from the request packet. It also checks if the new client's capabilities require more quality than that presently needed by the router, if so it sends an update message to the multicast routers up the stream till the server. When a client leaves the group the router has to delete the entry corresponding to the client and send update capability message to the servers up the stream if it no longer requires the quality currently reaching it.

Filtering: -This is the heart of the whole system. It takes care of handling the Selective Frame Reject and Capability based video content adjustment. It looks into the multicasting table and gets the link traffic state information, incoming rates and the capability needed for the link. Based on these factors it performs frame rejection and content adjustment like resolution changing to suit the capabilities required at the interface. It iterates through each of the interfaces over which the multicast packet has to be sent. It performs the filtering and then places the data to be sent to the egress module through a scratch ring.

Feedback Packet processing: -This module takes care of updating the network traffic state. It is a lightweight module that simply gets the feedback information from the feedback packets and places the information in the appropriate entries in the multicast routing table. This module goes with the filtering module into the same microengine to ensure balancing of the pipeline.

Egress: -This module takes care of delivering the packets out through an interface [6].

5. Performance Analysis

A sample output of the hierarchical coding implementation at seven different levels (starting from 1 x1 to 640x480) is given in Fig. 5.

Fig.5 Hierarchical coding at different levels

The efficiency of the implementation was analyzed by collecting the data about the output rates for various input rates as shown in Fig. 6.

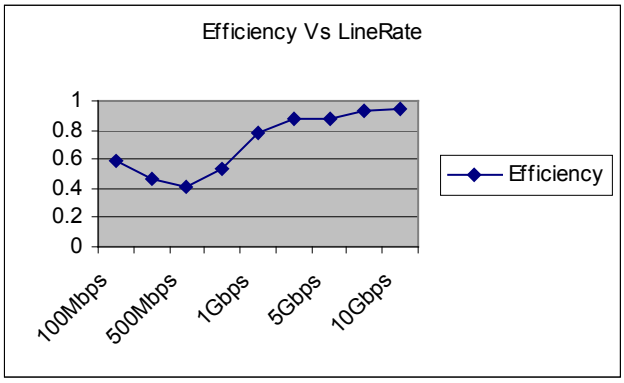

Fig. 6 Variation of Efficiency Vs Line Rate
(Average content replication factor at each mrouter = 2)

An important metric for an efficient design pipeline is the balance of the pipeline. The modules must have equal amount of work to do, otherwise the module with the maximum work will become a bottleneck and the advantages of using the pipelined architecture will not be realized. We carefully implemented the system for high performance on the network processor to make sure that the pipeline is balanced. The results can be seen in Fig. 7.

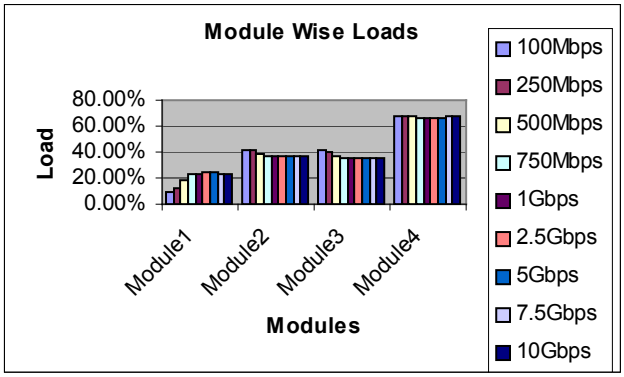

Fig. 7 Module Wise Loads Mod 1: Ingress Mod 2: Class.

& Multicast Client Request Processing Mod 3: Filtering & Feedback Packet Processing Mod 4: Egress

We can see that the loads on the Egress are high. This justifies our design decision to replicate the egress over 2 microengines to balance the pipeline.

6. Conclusion and Future Work

We have proposed and evaluated an efficient scheme for real-time multicasting of streaming video across heterogeneous networks to clients with diverse capabilities. Specifically, our contributions are in capability propagation and the use of hierarchical coding for high-speed resolution changes at the routers.

We are working on dealing with prioritized clients in this scenario of multicast streaming of real-time video over heterogeneous networks for heterogeneous clients. The problem with prioritized clients is that the filtering decisions at the multicast routers are heavily influenced by the priorities of the client they are dealing with. A low priority client demanding a high-resolution video will be sacrificed to provide higher frame rates to a high priority client with a lower resolution.

7. References

[1] T. Yamada, N. Wakamiya, M. Murata and H. Miyahara "Implementation and evaluation of video quality adjustment for heterogeneous video multicast," in Proceedings of APCC 2002, (Bandung), pp. 454-457, Sept. 2002

[2] N. Shaha, A. Desai, M. Parashar, "Multimedia Content Adaptation for QoS Management over Heterogeneous Networks", Proceedings of the International Conference on Internet Computing (IC 2001), Nevada, USA, June 2001

[3] W.Y. Ma, I. Bedner, G. Chang, A. Kuchinsky, and H.J. Zhang, A framework for adaptive content delivery in heterogeneous network environments, *SPIE Multimedia Computing and Networking 2000*, pp. 86–100, San Jose, January 2000

[4] Thomas Phan, George Zorpas, and Rajive Bagrodia. " An Extensible and Scalable Content Adaptation pipeline architecture to support heterogeneous clients", Proceedings of the 22nd ICDCS, 2002

[5] Intel IXP 2400/IXP 2800 Network Processor Programmer's Reference Manual, Intel Press, November 2003

[6] Intel IXP 2400 Network Processor Hardware Reference Manual, Intel Press, November 2003

[7]. Hvamstad, U., Griwodz, C., Halvorsen, P., "Offloading Multimedia Proxies using Network Processors", Proceedings of the International Conference (INC' 05), Samos Island, Greece, July 2005, PP. 133-120.

[8]. F. Zhai, Y. Eisenberg, C.E. Luna, T.N. Pappas, R. Berry and A.K. Katsaggelos, "Packetization schemes for forwarding Error Correction in Internet Video Streaming", Proceedings of the 41st Allerton Conference on Communication, Control, and Computing, Oct 2003.

[9] Yanlin Liu and Mark Claypool, "Using Redundancy to Repair Video Damaged by Network Data Loss", Proceedings of the ACM Multimedia Computing and Networking (MMCN) Conference, San Jose, California, USA, Jan 25-27, 2000.

[10] G. Apostolopoulos, Wai-tian Tan, Susie J. Wee, "Video Streaming: Concepts, Algorithms and System", Mobile and Media Systems Laboratory HP Laboratory, Palo Alto HP Labs, 2002-10-04.

[11] Johanson M. (1999). "*Scalable video conferencing using sub band transform coding and layered multicast transmission"*, Proceedings of ICSPAT'99.

[12] M. Rabbani and R. Joshi, "An overview of the JPEG 2000 still image compression standard", *Signal Process. Image Commun.* 17~1!, 3–48 ~Jan. 2002!.

[13] G. K. Wallace, "The JPEG Still Picture Compression Standard," *Commun. ACM,* vol. 34, pp. 30-44, Apr. 1991.

[14] K. W. Lee, R. Puri, T. Kim, K. Ramchandran, and V. Bharghavan. An Integrated Source Coding and Congestion Control Framework for Video Streaming in the Internet. In *Proc. IEEE INFOCOM*, March 2000.

[15] Erik J. Johnson, Aaron R. Kunze, "IXP 2400 Programming", The Complete Micro Engine Coding Guide, Published by the Intel Corporation Press.