Proceedings of the International Conference of
Applied Informatics, AI 2001, February 19-22, 2001, Insbruck, Austria

RESOURCE ADAPTIVE NOMADIC TRANSCODING ON ACTIVE NETWORK

JAVED I. KHAN & SEUNG S. YANG

Media Communications and Networking Research Laboratory
Department of Math & Computer Science, Kent State University
233 MSB, Kent, OH 44242
javed@kent.edu

Abstract

In this paper we present a active network architecture for supporting a new generation of network aware adaptive applications. We present the architecture in the context of a video transcoding system, which is capable of negotiating local network state based rate and let the video propagate over extreme network with highly asymmetric link and node capacities.

Key Words: Adaptive Video Applications, Asymmetric Internet, Active Network.

1. Introduction

In conventional packet network a network junction node (such as a switch or router) only forwards (and occasionally drops) packets [2,9,10]. **Active Network** extends their classical role to a new dimension. Here nodes are also capable of transforming packets in transit through active processing. Active network paradigm can potentially spark a new generation of smart networked applications—which otherwise are not easily realizable on traditional networks.

In this paper, we investigate one such concept application that concentrates on creative adaptation. We present an MPEG-2 rate transcoding mechanism, which addresses the issue of adaptation from two levels. It adapts with respect to the local constraints of two critical network resources—bandwidth and the processing resource at the junction nodes. As opposed to classical server-client model for building network applications under current network software architecture, this application launches itself into an Active Network in three parts-- server, transcoder and the client. The middle component is launched as active capsules in a suitable active junction point which has the capability to intercept the MPEG-2 [5] stream and downscale it if needed.

The transcoder senses local asymmetry in link capacities at various junction points of a network. Based, on that it accordingly adapts the video stream rate. For example, it can be dynamically deployed at nodes splicing a fiber and a wireless network, and thus it enables an incoming high-bandwidth video multicast stream to be re-encoded for the outgoing low-capacity wireless links. In comparison, today's media servers and fixed rate based. They store multiple copies of the same media one for each supported rate class (LAN, DSL/Cable, 56K, 28.8K etc.) [13,1,3,4]. The end-user is required to specify its rate. Although there have been few attempts to automate such specification, but, the difficulty seems to lie fundamentally with the end-to-end network programming paradigm. Either in a live video multicast, or in stored video cache none of the end-to-end solutions seems satisfactory. Classical solutions such as, sending the high-speed stream cuts off the low speed clients. Sending the lowest speed stream penalizes others by forcing the lowest quality to all. Sending multiple streams, burdens the network. In contrast we show that in-stream rate adaptation by a **nomadic active transcoder** can solve this riddle and offer optimality even at the level of individual links.

However, video stream rate adaptation is a challenging task. Because of complex inter packet data dependency, packet level rate adaptation—which is almost blind, offers very limited **down-scalability**. Dropping only 20-25% of the UDP packets can render an entire stream useless [12,2]. Content unaware, just packet slicing backed rate control seems to very wasteful and limited in their efficacy. Smarter content aware transcoding, in contrast can offer much broader range of rate adaptation with much graceful loss of video quality. However, MPEG-2 transcoding is also quite complex and computation intensive task. A number of techniques have been investigated for accelerated transcoding by us and other researchers, such as motion vector reuse, fast DCT domain transcoding, parameter bussing [11,7,8,6] These models provide a three-way tradeoff between the computational complexity, video quality degradation and second stage compression. It seems that in near future with the rapid advancement of the VLSI technology some nodes may be able to garner enough processing power for real-time high fidelity video transcoding. However, on any given large scale network the fact of the matter is that there will be always inequality of processing capability just like the asymmetry in bandwidths. Consequently, we are developing a nomadic transcoder, which also demonstrates **self-organization** behavior and choose the right operating state in this tree-way trade-off based on

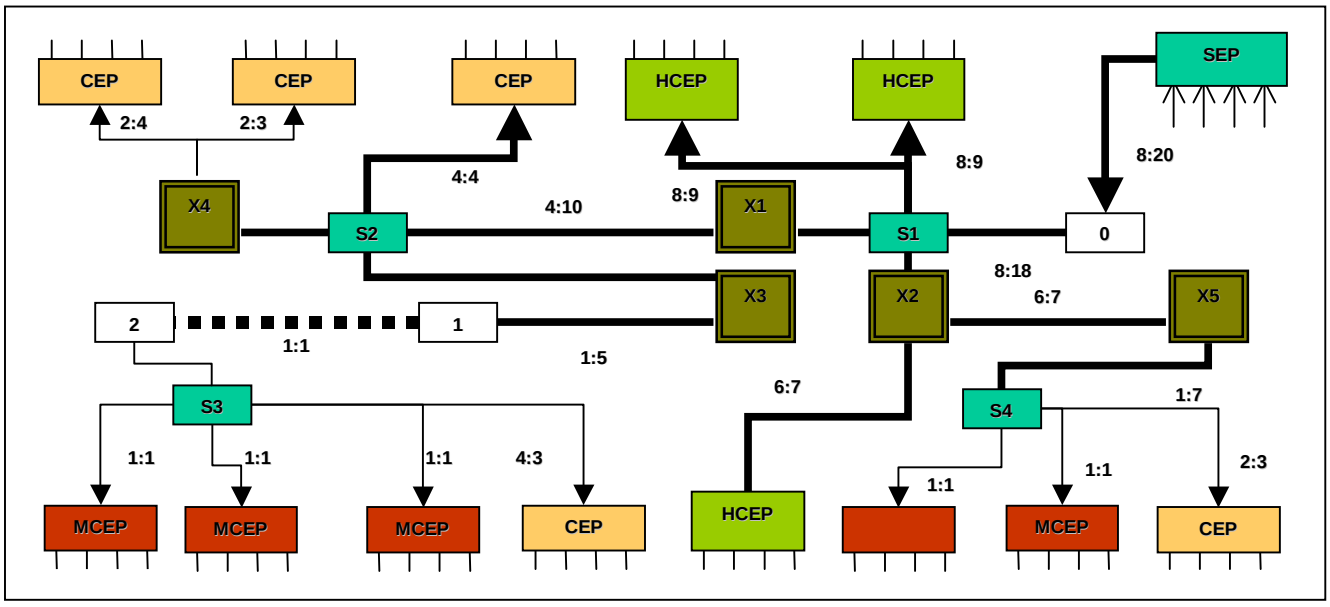


Fig-1 the adaptive video distribution system

the available network resource of processing power at the junction points. While the link bandwidth adaptation has been addressed up to some extent in few of the recent research little attention has been paid to the node capacity adaptation.

We are currently implementing an proto-type transcoder on the **Medianet Active Switch** [6]. The nomadic self-organizing transcoder represents a new generation of network aware reactive system. As a reactive system, it requires different network service model for automatic launching, management, and seamless operation, which is not present to-date. The proposed active service model offloads the end-systems from the undue burden of adaptation. It anyway seems to be a network level service – particularly when it is compelled by network constraints. While, active network presents a powerful but bare-bone architecture to insert and execute programs at junction points, however, considerable work is needed in defining the high-level service construct to provide a generic reusable adaptation infrastructure.

In the paper, we present this active service model. It can support such a nomadic resource adaptive systems by providing a rich new generic formalism for custom adaptation. In section 2 first we explain the operation of the model at application level with a simple network example and the application level issues involved in building a large scalable and adaptive version of it. Then in section 3 we explain our approach and present the proposed network service architecture.

2. Nomadic Transcoding

The model we are investigating has four application service modules (a) server (b) splitter (c) transcoder and (d) the client. A splitter duplicates streams in a junction. A transcoder downscales incoming video rate.

Fig-1 shows a target deployment of the modules on a sample core network. The server is attached to the *server end-point* (SEP). There are various client classes (regular, mobile and high definition), which are attached to the network via *client-end points* (CEP), *mobile client-end-points* (MCEP), and *high definition client-end-points* (HCEP). Each end-point acts as a distribution/ concentration center. The network connecting these end-points has unequal link and node capacities. Each link is labeled using two quantities R:B where R is the requested video rate and B is the link bandwidth.

2.1. Module Mapping

The first application issue is the optimized module placement. The video rate in the links should be calculated bottom up based on the requests from the CEPs downstream and the available link bandwidths. At each junction point, the downstream requests are aggregated and the maximum of them is propagated upstream as the new request. The requested rate either increases or stays same as it propagates upstream. The number of required transcoder is equal to the number of rate steps inside the distribution network. However, instead of catering to each requested rate, the number of transcoders can be minimized by using rate classes. A possible placement of the transcoders seems to be at the junction nodes. This requires each m-way junction node to have m-1 step down transcoders. However, each step down requires significant computation. Therefore, to distribute the computational load the step-down transcoders are not placed at the junctions, but one node down along each of the downstream paths.

2.2. Dynamic State Information Exchange

As can be noted, that the placement optimization depends on local network states. These include link bandwidth,

congestion, switching speed, processor speed, and available computational power of the nodes. The link capacity allocated to the video distribution can also vary with time such as due to congestion. Similarly, the computation cycles available to the adaptation mechanism such as the transcoder can also vary based on the multiprocessing load at the nodes. The transcoders must be able to optimize its rate conversion factor in response to the change in link states up and downstream. Similarly, the transcoding operation can be performed at several quality/computational effort choice levels. The particular level should be chosen based on the available computation power at the node. Thus a prerequisite to any adaptive system is dynamic and efficient monitoring of various local network state information.

2.3. Dynamic Module Relocation

Change in local state not only can spark change in step-down rate, but it can effectively require activation of new and deactivation of existing transcoder modules. Dynamic departure of clients can similarly trigger activation/deactivation of splitters and transcoders. Congestion, reduction in CPU cycle allocation may necessitate migration of transcoding operations to neighboring nodes. Thus, ideally, in a large adaptive system these modules should be occasionally able to relocate, however, without apparent disruption of the basic stream. Relocation should not cause loss or duplication of data. Information order should also be preserved. Additionally it is desirable that it should not impose significant delay.

3. System Architecture

3.1. Overview

While the above explains the target that we would like to achieve, however in a large network our principle goal is to automate the entire placement, deployment and running process of the above system. Consequently, we first identify the critical network layer services needed for adaptive systems. We then present the proposed organization for the entire system.

First we envision the rate adaptive service to be a special and generalized form of abstract communication. While, traditional networking provides only two types of communication channel constructs TCP and UDP, we generalize the concept and consider that the automatic rate adaptive communication service indeed is a higher level of communication service to the applications. The channel construct provides two important benefits. First it provides a separation between the network related and network independent tasks. Secondly, it allows the entire design and modules to be reused by various actual server and player applications. For example, while, in actual server application VCR operations can be supported, the network layer modules are only concerned about the

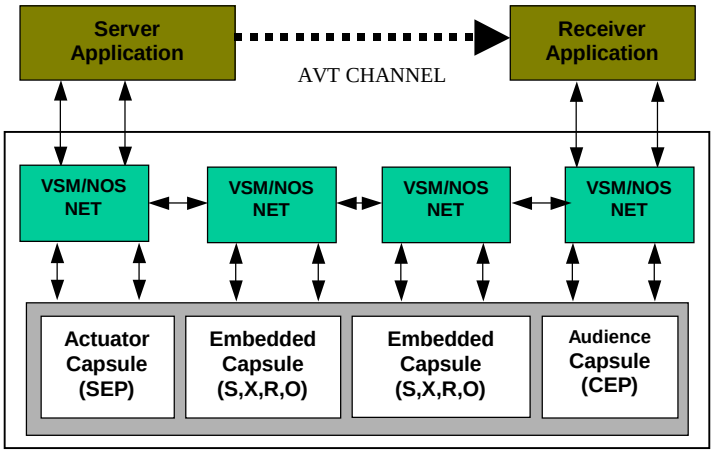


Fig-2 three tier-system architecture for the nomadic transcoder based video transmission

transport aspect of the service. We will call it rate *adaptive transcoding video* (ATV) channel.

Based on this channel abstraction we propose a three-tier system architecture. First is the application itself. The network independent part of the application resides in this tier. Second is the channel layer. This is the layer that houses the reactive components that bridge between the network and the application. The modules in this layer are programmable however, they execute strictly under the

Placement Map

```
DEFAULT URL=shttp://medianet.kent.edu/capsules/*.cap
MAP VALID ON=tree, path.
Rover       LOC=all nodes       INVOKE= 200s.
Organizer LOC= all nodes with rover.XCODE=ON INVOKE=5
Server      LOC=src
            URL=shttp://tv.kent.edu/capsules/server.cap
Splitter    LOC=all junctions between src and sinks
Xcoder      LOC=all nodes with rover.XCODE=ON
Player      LOC=all sinks
NETORDER   (Rover,Organizer,Server,Splitter,Xcoder,Player)
EXEORDER   (Rover,Organizer (Server|Splitter|Xcoder|Player))
```

Fig-3 (a) Capsule Definitions and Constraints

Connection Map

```
Rover.port0→(Request-notify)→ prev.Rover.mport1
Rover.mport2→(Grant-notify)→ anynext.Rover.port3
Server.port1→(MPEG2-stream)→ next.Xcoder.port0 |
                    next.Splitter.port0| next.Player.port0
Xcoder.port1→(MPEG2-stream)→ next.Xcoder.port0|
                    next.Player.port0
Xcoder.port3→(Control)→ prev.Player.port2|
                    prev.Xcoder.port2
Splitter.mport5→(MPEG2-stream)→ anynext.Xcoder.port0|
                    anynext.Player.port0

Request-notify=     Request:UDP
Grant-notify=       Grant:UDP
MPEG2-stream=       stream:TCP
Control =           switch:UDP+RatePar:UDP
```

Fig-3 (b) The Capsule Interconnection Rules

Rover Capsule

```
….
downstreamrequest[0]=self-request;

/*propagate a request upstream*/
if( not receivernode)
Receive(from all-1st-downstream, in &downstreamrequest[I]);
requesting=min(max(all downstreamrequest[I]), uplinkcapacity);
Send to parent Send(requesting);

/*the request granting phase begins from the top*/
        Receive(from 1st-upstream, &granted, &xcoder);
        Else granted=requesting;
        If(xcoder==ON) rover.xcount++; /*initially zero*/

/*and propagates downstream*/
For all children I {
        If(granted>=requested[I])
        rover.granting[I]=requested[I];
        Else rover.granting[I]=granted;
        If(granted>rover.granting[I]) {
                rover.xcoder[I]=ON;
                rover.xcount++;
        }
        /* Sample controvertial optimization */
        If(xcoder[I]==ON & downlink[I]>granting[I])  {
                rover.xcoder[I]=OFF;
                rover.xcount--;
                pushdown[I]=ON;
        }

/*More optimization is possible by pushing up xcoders here*/

        if(not receivernode)
        Send(all 1st- downstream,granting[I],pushdown[I]);
}
if (rover.xcount>0) rover.xcode=ON;
NOS_SetData(rover);          /* NOS variable is updated*/
```

Fig-4 (a) The rover pseudo-code that determines the self-placement behavior of the nomadic transcoders

Self-Organizer Capsule

```
….
/*set the xcoder configuration state value*/
now=NOS_GetData(system.now);
then=NOS_GetData(saved.then);
frame_now=NOS_GetData(xcoder.frameno);
frame_then=NOS_SetData(saved.frameno);
NOS_SaveData(saved.timethen, now);
NOS_SaveData(saved.framethen,frame);
rate=frame_now-frame-then)/(now-then);
SNR=NOS_GetDate(xcoder.snraverage);
organizer.xcodestate=CalculateConfigure(framerate,SNR,
QoS);
NOS_SetData(organizer);/* NOS variable is updated*/
```

Fig-4 (b) The organizer pseudo-code that determines the Self-reorganization state of the transcoders

control of the network and help in the local state dependent adaptation. The third is the enhanced network layer, which acts as a "glue" layer connecting the application layer, channel layer and the actual network and performs automation and facilitation. It is placed on top of the traditional network and OS. Fig-2 explains the three-tier model. We denote this enhanced network layer as the *network operating system* NOS (NET/VSM/BEE).

The *adaptive transcoding video* (ATV) channel is built with four primary types of capsules (a) actuator capsule (server-end-point SEP) (b) splitter (S) (c) transcoder (X), and (d) audience capsule (client-end-point(s) CEP). The actuator and audience capsule interact with the applications, and at the same time co-ordinates with other capsules and hides internal details and provides a single channel abstraction to the application. There are also two maps, which describe the rules for placement of the primary capsules and their interconnection, and two auxiliary capsules, which dynamically evaluate the network dependent placement logics and decides the roving behavior and the self-organizing states of the nomadic components. The channel designer thus can be different from the application programmer.

A multicast video application requests an adaptive transcoding video (ATV) channel in a similar way it requests sockets. NOS installs the capsules and creates the requested channel and application receives the finished product.

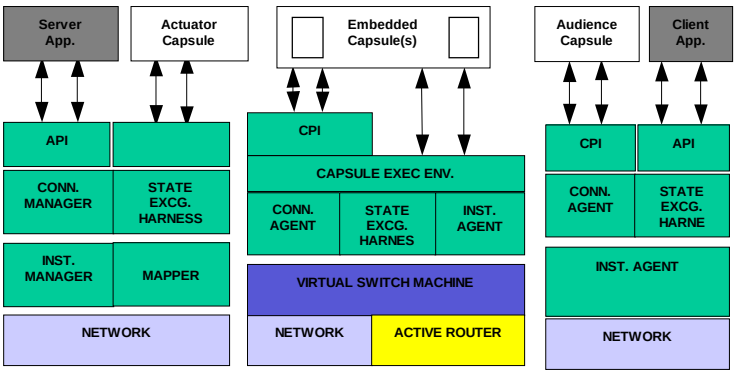
3.2. Channel Components:

Placement Map:

Placement Map (PMAP) and *Connection Map* (CMAP) together tells the installation manager how to cast the capsules in a given network. The specification itself does not require any knowledge about the specific network topology rather it specifies how the components should be connected on a given class of graph.

Fig-3(a) and (b) shows typical example of these maps. Placement MAP shows rules and constraints for placing the channel capsules. Each of the lines in PMAP describes a capsule object as attribute, value pair. Capsule name, placement constraint, URL for automatic loading, etc. are some of the attributes. Placement constraints can use especially defined network state variables such as *XCODE* (it's use will be explained shortly).

The placement is generally interpreted in the context of a specific *context network graph* (such as chain, binary-tree, multi-way tree, general graph etc.) on which the mapping is valid. PMAP statement *VALID ON* indicates this context. The system assigns a partial numeric order to all the components. A unique ordered id is assigned by the system for each channel session based on the specific network on which the channel is cast. The *NETORDER* statement resolved the partial order among the capsules, which operate on the same node. The *EXEORDER* statement specifies the invocation order of the modules (if needed) for deadlock free operation, and is interpreted by the scheduler. Channel designer can also specify an invocation mode for the capsules such as periodic, non-terminating (default), event-driven, etc.



(a) Passive-end stack (b) embedded stack (c) active-end stack

Fig-5 network services for adaptive application maintenance and execution NOS layers

Connection Map:

Connection Map (CMAP) describes the rules for connecting the capsules by specifying the connection rules between each ports pair of the capsules. The capsules have numbered *ports* and *multiports (mport)*. All are simplex and uni-directional. A *port* connects to only one other port. A multiport is an abstract set of ports. The actual number is determined after the casting by the installation process. Each member of multiport set is bound to a specific single port of another module. Each entry in the CMAP corresponds to one sending port of the capsules. It defines the connection type and the receiving ports. The interpreter evaluates the rules from top to bottom and left to right. CMAP also defines the message structure and connection type. CMAP can define complex connections. The aggregator in CMAP then defines the packet structures and the elementary transport mode (TCP or UDP) on which the actual connection is established.

Adaptation Capsules:

The adaptive behavior of the system is determined by two additional capsules (a) *Rover* and (b) the *Organizer*.

Rover dynamically determines the nomadic behavior and optimum placement of the capsules inside the channel. As specified in the PMAP execution order (Fig-3(a)), it is first invoked before other capsules. Fig-4(a) shows a typical Rover body. Rovers first use NOS calls to sense the link bandwidths (*uplinkcapacity*), and then exchange the video rate information. The rate request (*requesting*) propagates upstream from the receiver nodes, and gets aggregated at the junction nodes using a *min/max rule*. Once the aggregate request reaches the server-end, SEP grants a specified rate (*granted*), which then propagates downstream. Once, the propagation completes all the nodes have their Rover variable *XCODE* set appropriately.

Organizer capsule determines the self-organization state of the transcoders (Fig-4(b)). Each capsule instance can save some data for their subsequent instantiations. This enables then to keep track of the video frame-rate (*rate*) between two instantiations. This particular self-organizer capsule acts based on the outgoing frame-rate from the local transcoder and thus can be implemented inside the transcoder as well.

3.3. NOS Components

The overall implementation of the system now requires the following services from the NOS.

Programming Interface:

Unlike classical channels, the NOS provides two interfaces. First is the *Application Programming Interface* (API) to the applications (server and player) to request and run a rate transcoding channel. The other is the *Capsule Programming Interface* (CPI) for capsules to execute and co-ordinate their functions. We have proposed a detail of both the interfaces in [14].

Services:

Below the interface, the following new network layer services are now required for the ATV channels: (a) channel installation service (b) intra-capsule communication service, and (c) network state exchange service. All services have three NOS components. A manager, which works at the actuator end of the connection, an audience-agent, which works at the audience end-point, and a network agent, which works in the intermediate nodes. Fig-5 shows the service stacks in these three positions. The service stack involved at the junction nodes is little different form the end-point stacks. Junction nodes do not need any API, as there is no application component there. However, since, the junction points can be a typical active router, instead of relying on general OS, a *Virtual Switch Machine* (VSM) is created on top of the router. VSM identifies active communication from regular routing operation and when an active data arrives it diverts the active packets towards the capsules. VSM also allocates the CPU and memory resources between the competing capsules and acts as a capsule scheduler. Immediately above VSM an execution environment called channel *Building and Execution Environment* (BEE) provides the utilities upon which the capsules actually execute. A critical service is the network state exchange service, which provides the capsules access to the network states includes SNMP link MIB-II variables, and additional active node states such as CPU, memory allocation, and execution time to the capsules.

Dynamic Relocation:

The channel installation service not only provides the initial placement of the capsules. It also coordinates dynamic relocation of the modules. The Rover is periodically invoked by the VSM/BEE. Thus a new calculation is made on the placement variable. If they change, installer accordingly dynamically starts and stops capsules. For seamless invocation, the new capsule is first loaded and placed in the background without changing the current operation. When, the module is ready to operate, only then the VSM switch is invoked to route active packets to the newly activated module. Seamless

revocation is performed by disabling the packet forwarding first. For graceful revocation the VSM/EE invokes a particular capsule module called *CAP_Destroy()*, given by the channel designer.

Channel Relative Addressing:

In addition to the above services, the VSM/BEE also provides an abstract addressing scheme, where capsules and maps can use a channel relative addressing mechanism to refer to each other based on their logical position in a channel, without knowing the actual deployment. The addressing scheme has three references-- *src*, *sink*, and *this node*. Complex references are built on them using set operators and modifiers such as upstream, downstream, all etc. Once, the actuator and the audience end-points are determined, the CRA interpreter returns determined the actual network addresses for all CRA addresses. Fig-3(b), and 4(a) and (b) shows examples of such network relative references.

4. Conclusions

It seems that the asymmetry of the Internet is increasing. While the first generation of internet growth have to cope with heterogeneity of software systems and standards, the next generation of internet will have to deal with much fundamental (and perhaps harder to overcome) issues—the asymmetry of network resource such as bandwidth, switching capacity, end-to-end delays.

For seamless operation and information transfer over such asymmetry, this will require a new generation of applications technology with built in adaptability. This will however, also require innovation in the supporting network infrastructure. Without such network layer support, the applications will be pushed into improvising piece-meal solutions. Current end-to-end paradigm of network applications does not seem to hold any easy solution at hand.

In this paper, we have outlined an active network based network service model to support one such concept application-- a nomadic MPEG-2 rate transcoding mechanism, which addresses adaptation from two levels. It adapts with respect to the local constraints of two critical network resources—bandwidth and the processing resource at the junction nodes. Very little previous work has been reported for node capacity adaptation.

In our design we emphasized reusability. We placed core services in the network layer needed for supporting nomadic and self-organization behavior of adaptive systems. Additionally, we have introduced the concept of programmable channel construct to encourage reusability, instead of building the video transmission specific adaptive logic directly into the application core. However, such programmability is an uncharted area in current networking. Eventually a number of detailed issues, particularly related to efficiency of nomadic behavior and self-organization have to be addressed with much focus. Within the scope of this paper, we have not elaborated on the transcoding operation itself. More information about the transcoder itself can be found in [6,7].

The work is currently being funded by the DARPA Research Grant F30602-99-1-0515 under it's Active Network initiative.

5. References:

[1] Amir, Elan, Steven McCanne, and Randy Katz, Receiver-driven Bandwidth Adaptation for Light-weight Sessions, Proceedings of ACM Multimedia '97, Seattle, WA, Nov 1997,

[2] Bhattacharjee, S., Kenneth L. Calvert and Ellen W. Zegura. An Architecture for Active Networking. High Performance Networking' 97 ,White Plains, NY, April 1997. [also available at http://www.cc.gatech.edu/projects/canes/papers/anarch.ps.gz, October 98]

[3] Fluckiger, F, Understanding Networked Multimedia Applications and Technology, Prentice Hall, UK, 1995.

[4] Haskell B. G., Atul Puri and Arun Netravali, Digital Video: An Introduction to MPEG-2, Chapman and Hall, NY, 1997.

[5] Information Technology- Generic Coding of Moving Pictures and Associated Audio Information: Video, ISO/IEC International Standard 13818-2, June 1996.

[6] Javed I. Khan, S. S. Yang, Medianet Active Switch Architecture, Technical Report: 2000-01-02, Kent State University, [available at URL http://medianet.kent. edu/ technicalreports.html, also mirrored at http:// bristi.facnet.mcs.kent.edu/medianet]

[6] Javed I. Khan, Q. Gu, Darsan Patel and Oleg Komogortsev, , Medianet Active Video Transcoder Performance, Technical Report: 2000-01-02, Kent State University, [available at URL http://medianet.kent. edu/ technicalreports.html, also mirrored at http:// bristi.facnet.mcs.kent.edu/medianet]

[7] Keesman, Gertjan; Hellinghuizen, Robert; Hoeksema, Fokke; Heideman, Geert, Transcoding of MPEG bitstreams Signal Processing: Image Communication, Volume: 8, Issue: 6, pp. 481-500, September 1996,

[8] Javed I. Khan, Motion Vector Prediction in Interactive 3D Video Stream, Proceedings of the World Congress on Advanced IT Tools, IFIP '96 IT, Canberra, Sept 96 , pp533-539.

[9] Tennenhouse, D. L., J. Smith, D. Sincoskie, D. Wetherall & G. Minden., "A Survey of Active Network Research", IEEE Communications Magazine, Vol. 35, No. 1, Jan 97, pp 80-86

[10] Wetherall, Guttag, Tennenhouse, "ANTS: A Tool kit for Building and Dynamically Deploying Network Protocols", IEEE OPENARCH'98, San Francisco, April 1998. Available at: http://www.tns.lcs.mit.edu/publications/openarch98.html

[11] Youn, J, M.T. Sun, and J. Xin, "Video Transcoder Architectures for Bit Rate Scaling of H.263 Bit Streams," will beappeared to 'ACM Multimedia 1999', Nov., 1999. pp243-250.

[12] Jill M. Boyce and Robert D. Gaglianello, Packet loss effects on MPEG video sent over the public Internet; Proceedings of the 6th ACM international conference on Multimedia , 1998, Pages 181 - 190

[13] White Paper, Delivering RealAudio® or RealVideo® from a Web Server, RealNetworks Technical Blueprint Series, 1998 [URL:http://service.real.com/help/ library/blueprints/ hosthtml/webhost.htm Last Download: Sept 20, 2000]

[14] Javed I. Khan, & S. S. Yang, Made-To-Order Custom Channels for Netcentric Applications over Active Network, Proc. Of the Internationaal Conf. On Internet and Multimedia Systems and Applications, IMSA 2000, Nov 2000, Las Vegas, pp22-26.