CERIAS Tech Report 2005-144
Multimedia Data Transmission and Contol Using Active Networks
by B Bhargava, S Wang, M Khan, A Habib
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

Available online at www.sciencedirect.com
SCIENCE DIRECT
Computer Communications 28 (2005) 623–639
computer communications
www.elsevier.com/locate/comcom

Multimedia data transmission and control using active networks ☆

Bharat Bhargava[a], Sheng-Yih Wang[b], Maleq Khan[a,*], Ahsan Habib[c]

[a]Department of Computer Sciences, Purdue University, West Lafayette, IN 47907, USA
[b]Foundry Networks, Inc. 2100 Gold Street, Alviso, CA 95002, USA
[c]School of Information Management Systems, University of California, Berkeley, CA 94720, USA

Received 5 August 2004; accepted 5 August 2004
Available online 21 September 2004

Abstract

Active network is an excellent paradigm to provide customized network services to the applications by allowing them to inject specific program to the intermediate routers. Active networks provide the flexibility for the application programs to modify the services that a router can provide to suit its specific needs. Therefore, it has the potential to provide application-level quality of service (QoS) at the transport and network layers. In this paper, we present an adaptable network architecture, called ADNET, which provides mechanisms to allow the application adapt to the resource constraints to achieve improved QoS. Our design aims to unify different QoS control mechanisms (e.g. integrated services, differentiated services, and active networks) together to provide a wide range of network services to all users to meet their specific needs. We propose a new fragmentation scheme with low overhead ($<5\%$) to transfer large-size multimedia data. Using this fragmentation scheme, a new transport protocol, called ACtive Transport Protocol (ACTP) is integrated with the design. We use a new measure, called usefulness, to better reflect the QoS perceived by the end-users. In our experiments, we compare different schemes of video transmissions: non-active transport protocols such as UDP and TCP with IP fragmentation, ACTP framework with active networks, and ACTP framework without active networks. The ACTP scheme with active networks outperforms the others in achieving application level QoS.

Keywords: Multimedia data; Active networks; Quality of service; Fragmentation

1. Introduction

Distributed Multimedia Systems [1] are promising to cause major changes in our everyday life. For example, desktop video conferencing and collaboration systems enable people to interact and collaborate with others who may be far away. Home shopping systems allow people to browse merchandize at home through virtual reality techniques and to order on-line. Video-on-Demand systems allow people to watch their favorite programs at home any time they want. To make all these scenarios reality, distributed multimedia systems need support from the underlying transporting networks. Transmitting multimedia data, especially for continuous media such as video or audio, over the network is a challenging problem. The network requirements for multimedia data are different from those for traditional network traffic such as TELNET or FTP. For example, multimedia data are very sensitive to packet delivery delay, but can endure some loss of data. In contrast, traditional data network traffic can endure packet delivery delay, but have to be delivered 100% error free.

Multimedia data flows through several subsystems. For example, a video frame is encoded at the video server program, sent through the underlying transport network, and is decoded at the receiving application. To provide better understanding of the requirements of the multimedia applications, these requirements are abstracted as a set of Quality of Service (QoS) parameters. The problem of providing satisfactory QoS to applications can then be formulated as satisfying constraints on QoS parameters. Since each subsystem in the data transmission process

☆ This research was partially supported by CERIAS, and NSF grant ANI-219110 and CCR-001712.

* Corresponding author.
E-mail addresses: bb@cs.purdue.edu (B. Bhargava), swang@foundrynet.com (S.-Y. Wang), mmkhan@cs.purdue.edu (M. Khan), habib@sims.berkeley.edu (A. Habib).

0140-3664/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2004.08.020

contributes to the degradation of the QoS, QoS control mechanisms are studied at all these different components. To facilitate the study, QoS are usually grouped into different layers [2]. The application QoS layer (which includes User-oriented and Application-specific QoS parameters) and the network QoS layer (which includes the whole network protocol stacks, especially the transport and network layers in networking terminology) are the two major layers where the current research on QoS parameters and QoS control mechanisms is focused.

From the discussion above, it is clear that the integration of different layers of QoS control mechanisms are needed to provide satisfactory QoS to the applications. In this regard, several design issues need to be investigated: (1) The network design should be flexible to accommodate emerging new applications. (2) The design should provide mechanisms for the application to convey their QoS needs to the network. (3) The application-specific and network-specific QoS control mechanisms should be integrated to provide a unified system of QoS provision.

Active network is a paradigm that provides flexibility and mechanisms for the applications to convey their QoS needs to the network. This powerful communication paradigm can adapt upcoming traffic by methods, which are part of the transported multimedia streams. This promising approach has lots of unexplored potential applications. New high level protocols can be dynamically integrated without changing network software, and new adaptive algorithms become possible which increase the flexibility of multimedia communication systems [3].

Various network operations such as fragmentation, error recovery, and traffic control can be performed efficiently and with high quality of service when application specific information is available. Using active paradigm, an application can tell the network nodes (routers) how the operations can be performed to achieve higher quality of services because only the application program has the knowledge of the internal properties of the data being transported. For example, when multimedia data are fragmented arbitrarily and if a packet is lost, the whole video frame containing the packet might be useless. On the other hand, in active paradigm, the application can provide the fragmentation procedures/information/hints, and using this information data can be fragmented in such a way that the data in other packets are still useful even if some packets are lost. In some cases, it might be possible to reconstruct the lost packets from the other related packets. This active paradigm has the potential to provide the application-level QoS at the transport or network layers [4,5]. We show that active network techniques can be used as a mechanism to extend the Differentiated Services (DiffServ) techniques in providing application-level QoS. In addition, active network techniques can be used to provide dynamic re-negotiation in DiffServ approach.

This paper focuses on the investigation and development of approaches to manage and control the transmission of multimedia data using active network paradigm, which can provide satisfactory QoS to the applications. We propose an adaptable network architecture, called ADNET, which allows different QoS provisioning schemes such as Active Network, Integrated Services (IntServ) and DiffServ to co-exist and provides customized services to achieve better application-level QoS. This architecture also enables the application programs and the networks to adapt and perform the tasks of traffic management and control together. We design a new transport protocol, called ACtive Transport Protocol (ACTP), to transmit multimedia data using the proposed architecture. ACTP provides datagram delivery similar to UDP, but with some extensions such as built-in fragmentation scheme and security mechanisms to better support the active network traffic. ACTP does not rely on the underlying IP-fragmentation in the network layer. Instead, it uses a new fragmentation scheme designed for multimedia data transmission in active environment. The fragmentation scheme utilizes the properties of active network paradigm and the proposed adaptable architecture ADNET. Experimental results show that ACTP protocol with the proposed fragmentation scheme in active network environment provides better QoS than traditional IP networks. Using ADNET architecture, we transmit three sets of video clips: *Jurassic Park, Lion King*, and *Tai Chi* from video sources to destinations via active routers. The results show that the packet loss rate for ACTP fragmentation in active environment is reduced by 10% than that in traditional IP delivery.

A new metric, called *usefulness*, to measure QoS received by the application is developed systematically. Usefulness is shown to provide a more accurate assessment of the user-perceived QoS than other commonly used measures. Experimental results show that when usefulness is higher, the visual quality of the received clips are also better (clear and sharp). In Section 6, we see that usefulness provided by the proposed solution is always higher than that of the traditional IP networks. Usefulness in ADNET is as high as twice of usefulness in IP networks.

The rest of the paper is organized as follows. The works related to the topics addressed in this paper are discussed in Section 2. In Section 3, we propose an adaptable architecture of an active router. Section 4 presents a fragmentation scheme for large multimedia data. A new measure of QoS is developed in Section 5. Experimental results are presented in Section 6. Finally, the conclusions and future works are in Section 7.

2. Related works

Wetherall et al. [6] develop a toolkit called ANTS for building and dynamically deploying network protocols. ANTS is based on mobile code, demand loading, and caching techniques. ANTS allows new protocols to be dynamically deployed at both routers and end systems,

without the need for coordination and without unwanted interaction between co-existing protocols. A group at University of Pennsylvania developed a programming language for active networks called PLAN [7], a self-learning network device called Active Bridge, an active network encapsulation protocol called ANEP, and a security infrastructure based on the idea of *Query Certificate Managers*. Hicks et al. [8] reported the experience on capsule-based active networking for ANTS and PLAN systems. The authors reported that both systems can achieve useful levels of flexibility, performance, and usability. In our approach, we put the responsibilities of managing code movements to the capsules. In contrast, the demand loading and caching scheme in ANTS rely on the active routers to make decisions.

Tennenhouse and Wetherall [9] describe their vision of an active network architecture, outline the design, and survey the technologies that can be brought to bear on its implementation. They propose that the research community mount a joint effort to develop and deploy a wide area ActiveNet. Our research can contribute to this vision to design a unified framework of integrated services, differentiated services, and active networks. Our proposed fragmentation scheme will be an excellent choice to transfer a large volume of multimedia traffic and fulfill the user perceived quality of service.

Wolf and Turner [10] discuss the design issues of high speed active routers. Custom processing of packets at the link speeds requires immense computational power. To overcome the limitation, the authors propose to use multiple network processors with cache and memory on a single application specific integrated circuit. The design is used as a vehicle for studying the key issues that need to be resolved to allow active networking [6] to become a mainstream technology. AMnet [11] is an architecture for programmable networks that mostly runs in the user-space of an unmodified Linux installation. The idea is to use the existing operating systems functionalities and give a general specification of an interface between the generic operating system and the particular active node functionalities. On the other hand, VERA [12] defines an interface between router hardware and software. An example of flexible router hardware is the Dynamically Extensible Router (DER) [13], which inserts processing elements between the line cards and the switching fabric.

An active network architecture based on the discrete active network approach is proposed and analyzed in [14]. The proposed architecture is applied to the congestion control problem on multimedia data [15]. Several data reduction techniques such as lossless compression, selective discard, and transcoding are built into the active routers. Prabhavalkar et al. [16] provide an active mechanism for controlling unresponsive connections and minimizing the degradation in network performance caused by bandwidth-greedy applications. This menu-based approach gives strict administrative control over the services that the network can offer in transmitting data in an active environment.

In the Raid laboratory at Purdue University, active network technologies have been applied to multimedia applications such as video conferencing [17]. An architecture, called *active gateway*, is proposed for video conferencing traffic and QoS control. It is shown through experiments that this facility enables more control functions than are seen in conventional video conferencing tools. Uruena et al. [4] provide a pragmatic technical approach to active networks that supports prototyping of multimedia transport protocols. The authors propose an a Simple Active Router Assistant Architecture (SARA) that allows upgrading legacy high-speed routers to work as active nodes by delegating active processing to an external entity called assistant. A transport protocol sub-layer, called Fully Programmable Transport Protocol (FPTP), is located between the application and the traditional transport protocols so that multimedia applications are able to specify the QoS transport required in terms of order, reliability, bandwidth, time and synchronization constraints. In our work, we focus on designing architecture and protocols to achieve high application-level QoS for multimedia data transmission.

Receiver-driven Layered Multicast (RLM) [18] combines a layered source coding algorithm with a layered transmission system. The video sender utilizes a layered source coding algorithm to divide the video data into several layers. Each layer of video data is then sent to a different multicast group. The traffic control mechanism for layered multicast is discussed in [19]. Our goal is to use active networking in efficient transmitting of large volume multimedia data.

SwitchWare [20] project provides higher level of abstraction on network services which are selectable on a per user or even per packet basis. A special feature of SwitchWare approach is that the security mechanism in SwitchWare is provided by using formal methods at the programming language level. Therefore, various safety and security properties of a SwitchWare program can be tested before execution. Security is an important issue in deploying active networks. However, the objective of this paper is to provide a framework to provide application level QoS. Any security prototype such as [21] that provides authorization control with strong end-to-end authentication can be integrated with our design.

3. ADNET: an adaptable network architecture

3.1. Preliminaries

QoS can be provided at different layers of the network stack. The application layer and the network layer are the two major layers where QoS parameters and QoS control mechanisms are studied. For network layer QoS

provisioning, Integrated Services (IntServ) [22] and Differentiated Services (DiffServ) [23] are proposed by Internet Engineering Task Force for IP-based networks such as the current Internet.

The IntServ is a reservation-based QoS control mechanism. The objective of IntServ is to provide customized QoS to each individual traffic flow. In contrast, the existing IP networks do not provide any mechanism to support customized QoS for traffic flows. Scalability to a large number of flows and complexity in implementations are the two most serious drawbacks of IntServ approach. The DiffServ is a new effort to provide QoS support in IP networks. DiffServ tries to eliminate the need of RSVP-style resource reservation by aggregating traffic flows with similar QoS requirements into one single traffic class. The packets belonging to the same aggregated traffic class are marked using the same Differentiated Services Code Point (DSCP), contained in the IP header DSFIELD/ToS. The packets are forwarded according to the per-hop behaviors (PHB) defined for this particular traffic class.

One drawback of the DiffServ approaches is that the QoS parameters cannot be dynamically changed. The QoS in DiffServ is specified by Service Level Agreement (SLA). It is not easy to re-negotiate and adapt when communication requirements change dynamically. Another drawback of IntServ (and DiffServ) is that they provide only network-level QoS. They cannot provide application-level QoS. The QoS parameters in both approaches are specified by network performance parameters such as bandwidth and delay. However, sometimes the application QoS parameters are different from the network QoS parameters. For example, a video application may want to have a smooth playback (constant frame rates) no matter what the conditions of the network are.

Active networks [24] allow application programs to execute specific programs in the routers and provide the flexibility to modify the default services that a router can provide to suit its specific needs. Therefore it has the potential to provide the application-level QoS at the transport or network layers. We show that active network techniques can be used as a mechanism to extend the DiffServ techniques in providing application-level QoS. In addition, active network techniques can be used to provide dynamic re-negotiation in DiffServ approach.

We design an adaptable network architecture, called ADNET, which allows all of the active traffic, IntServ traffic, and DiffServ traffic to co-exist. Our long-term goal is to unify all three paradigms together to provide a wide range of network services to all the users and meet their specific needs.

3.2. Design goals and requirements

Adaptable. The adaptability features in ADNET include: adaptable network services and adaptable applications. ADNET allows the application to adapt to the resource constraints. For applications that do not make reservation in advance, the traffic is aggregated into a specific class based on its attributes such as its DSCP label. The aggregated traffic classes are subject to traffic management similar to DiffServ.

Safe. ADNET explicitly includes the resource management module. In contrast, SANE [25] left the resource management problem untouched, and therefore can not provide true safe execution.

Secure. ADNET employs the standardized Secure IP Protocol (IPSEC) [26] to ensure secure communications. Security requirement is an adaptable feature in our design. It can be specified as a QoS parameter when setting up reservations.

Efficient. The efficiency of ADNET comes in two flavors: short-cut path for non-active traffic and primitive operations for multimedia payloads.

Scalable. Because our design address both the customized aspect and the aggregated aspect of network services, it is scalable to very large networks.

Interoperable. For applications which are based on the current best-effort paradigm, our design provides seamless interoperability so that they can run without any modification. Similarly, in absence of active routers on the path of a multimedia session, the packets of the session will be forwarded without any special treatment by the router.

3.3. Architecture design

ADNET is inspired and partially derived from the work in Integrated Services [22], Differentiated Services [23], and Active Networks [24]. Fig. 1 shows the architecture of ADNET. The design focuses on the explicit management of resources for classified traffic flows. The notion of classified traffic flow is a generalization of the usual sense of the traffic flow which encompasses a variety of aggregated flows. ADNET is comprised of four major modules: *Input Interface*, *Active Execution Environment (AEE)*, *Output Scheduler*, and *Policy Database*. The Input Interface receives packets from the network and performs classification and shaping of the traffic. The AEE is composed of a virtual machine and a CPU scheduler which provides an execution environment for active network traffic. The output scheduler performs output bandwidth allocation to share fairly the output bandwidth among different types of classified traffic flows. The policy database provides policy information to other major components and interacts with AEE to update the policies.

We keep a per-classified-flow state to enforce various policies on the traffic. We argue that per-classified-flow state is a necessity for future networks because it will enhance the security and accountability of future networks. For each classified traffic flow, a resource limitation tuple is set up to limit the usage of various resources in the router. In the current design, a resource limitation tuple consists of maximum input bandwidth, maximum output bandwidth,

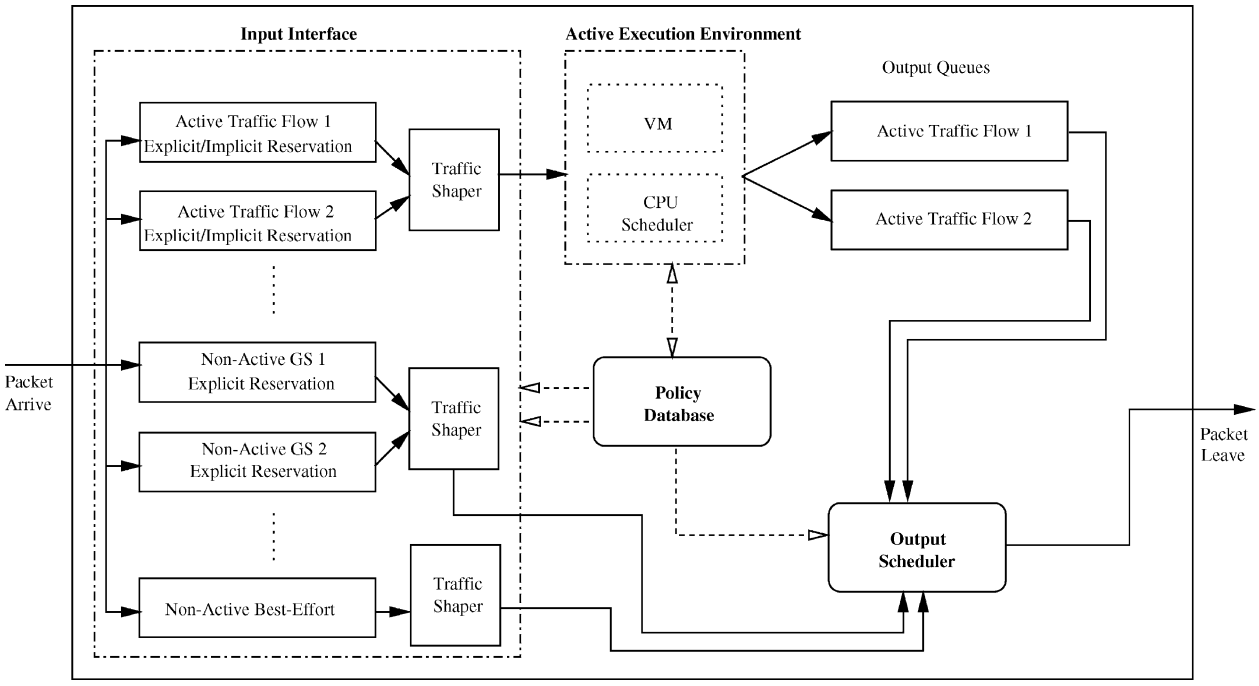

Fig. 1. Architecture of an active router. It shows the components in the input interface, active execution engine, and output scheduler.

and maximum CPU time allowed. Our design is IP-based. On the top of IP, we introduce a transport layer protocol called ACTP for active traffic.

3.3.1. Input interface

The input interface consists of two types of submodules: traffic classifiers and traffic shapers.

Traffic classifier. We classify the network traffics into four categories: active traffic with explicit reservations, active traffic without explicit reservation, non-active traffic with explicit reservations, and non-active traffic without explicit reservation. The router provides a default resource limitation tuple for every active traffic flow. For active traffic with explicit reservations (which include a set-up phase before the actual data are sent), the value of the resource limitation tuple is negotiated using primitives provided by the active router. The resource request is made at the set-up phase, or on-the-fly when necessary. This is how an active traffic which needs guaranteed service can reserve the required resources and become an active (or non-active) traffic with explicit reservations, while adaptable active traffic can simply request additional resource when needed. Once the resource requests are granted, usage of the resources is up to the application. If the system is out of resources, we have two different options. One is to drop the packet, and other is to forward the packet without any special service. Both are easy to deploy. However, a hybrid version can be implemented where the application specifies in the packet what to do in this situation.

The DiffServ traffic flows are aggregated into different super-flows based on the DSCP values in their packets. These super-flows correspond to the category of the non-active traffic with explicit reservations. The router keeps a flow state for each aggregated flow for the purpose of management (policing, billing, shaping, and dropping). We aggregate all the best-effort traffic (traditional network traffic) into a single flow under the category 'non-active traffic without explicit reservations'. The non-active guaranteed service traffic flows correspond to the notion of non-active traffic with explicit reservations.

Traffic shapers. Shaping reduces the traffic variation, makes it smooth, and provides an upper bound for the rate at which the traffic is admitted into the network. A shaper has a finite-size buffer. Packets are discarded if there is not sufficient space to hold the delayed packets. There are three traffic shapers in our design: one for the active traffic flows, one for the non-active flows with explicit reservations, and one for non-active best-effort flows. The logical separation of the traffic shapers into three categories is for functional description only. In practice all the three traffic shapers may be implemented as one module. More details on traffic shaping can be found in [27,28].

3.3.2. Execution environment for active flows

Virtual Machine. Active nodes run a Node Operating System (NodeOS) and one or more Execution Environments (EEs) on top of NodeOS [29,30]. Each EE implements a virtual machine (VM) which runs the active programs coming with the data packets. The VM can be very general (such as Java Virtual Machine, which is the most popular one) or very specific (such as Spanner [31], which provide specialized functions for network management tasks). Since emerging applications will contain a lot of multimedia data such as compressed video and audio, some specialized

processing functions are necessary in order to provide better services to these data.

We propose a new VM for our design. Although Java Virtual Machine (JVM) is popular, we argue that a new customized VM for running programs inside the router is necessary because of the following reasons: (1) *Compactness*. The object code (or bytecode) format for currently available general-purpose VMs such as JVM is not very compact. The original design of these VMs is for execution in the end system or device, which usually do not need to handle programs of huge size. The router may need to process thousands or even millions of active programs in a very short period of time. Even the existing VM specifically designed for active networks such as Spanner [31] is not able to encode any interesting algorithms in few bytes. (2) *Resource management*. Currently available general-purpose virtual machines such as JVM do not include fine-grain resource management mechanisms. In particular, the VM is not tightly coupled with the OS, therefore resource management cannot be done effectively. In our design, the VM includes operations dedicated to resource management tasks.

VM plays several roles in the overall architecture. Specifically, VM can (1) work as a Bandwidth Broker (BB), (2) work as a signaling mechanism, (3) change DSCP definitions and update policy database, (4) constrain resource consumptions of active flows. VM includes some instructions that are unique: (1) Resource management (Request and Setup). (2) Specialized multimedia data processing (MPEG and H261 encoding/decoding). (3) User installable operation codes. The program can ask the virtual machine to install some particular part of the code into a single opcode. There are two modes of installable opcodes: the *secure mode* and the *light-weight mode*. In secure mode, the security hash algorithm MD5 is used to generate the footprint of the user code. In light-weight mode, the opcode is determined by the user as a 16-bit number.

We conduct experiments to compare VMs implemented with Java and C. In our experiments, the Java decoder is 6–10 times slower than the C decoder under Linux and Solaris environment. Although these results are not a surprise to us, it does provide an idea of the magnitude of slowdown when using Java in real-world. We also found that the JIT compiler does provide substantial improvement in execution time over the bytecode interpreter (over four times of speed up). However, there is a room for significant improvement for the performance of Java bytecodes to be close to native object codes. The results suggest that although there are potentials in the mobile code technologies in the future, a native (C/C++) implementation is favored under current technologies if multimedia data processing capabilities are needed in the VM.

CPU scheduler. For each active traffic flow, a thread (light-weight process) of the designated execution environment (virtual machine) is created to handle the flow. The execution of the thread is under the control of the CPU scheduler to compete for CPU time with other active traffic flows. Since all threads are scheduled together, the CPU scheduler ensures that each flow receives a fair share of CPU time. After processing, the generated active capsules is put in an output queue. For each active traffic flow, there is a corresponding output queue. These output queues, together with all the output queues of the traffic shapers for non-active guaranteed-service traffic and best-effort traffic flows, are linked to the output scheduler. For more details on scheduling processing resources, readers are referred to [32].

3.3.3. Output scheduler

The output scheduler is responsible for fairly allocating the output bandwidth among different classified traffic flows. The output scheduler schedules the packet delivery using fair queuing scheduling disciplines. Note that since every active traffic flow has its own output bandwidth limitation enforced, it is not possible for a misbehaving active traffic to shut down the output link.

3.3.4. Active transport protocols (ACTP)

Our design introduces a new transport protocol called Active Transport Protocol (ACTP) to isolate the interactions (possibly very complex) among active traffic flows and other traffic flows based on other transport protocols such as TCP or UDP ACTP reduces the complexity of the traffic and resource management tasks. It also simplifies the programming interface for active network programming by providing APIs similar to BSD-sockets.

ACTP is a transport protocol on top of the IP protocol. The ACTP provides datagram delivery similar to UDP, but with some extensions such as built-in fragmentation scheme and security mechanisms to better support the active network traffic. The ACTP supports the concept of source/destination port similar to those in TCP or UDP to differentiate different sessions. Since the behavior of a transport protocol can greatly influence the network performance, care must be taken when new functionality are added. We have developed a fragmentation scheme on ACTP for multimedia traffic (Section 4) and performed a series of experiments (Section 6).

ACTP supports built-in security mechanisms. For a secure ACTP session, the key management protocol that is used with IP layer security is performed at the ACTP layer. The authentication, encryption, and other security functions are performed at the IP layer using IPSEC [26] ('IP Authentication Header' [33] and the 'IP Encapsulating Security Payload' [34]). The format of the ACTP packet is shown in Fig. 2.

Each packet consists of an ACTP header and a body. An ACTP header consists of the following fields: source port number, destination port number, total packet length, checksum, active program length, and fragment offset. The body consists of two parts: the active program and additional data. The active protocol implemented in our tool provides an easier way to construct large active programs when

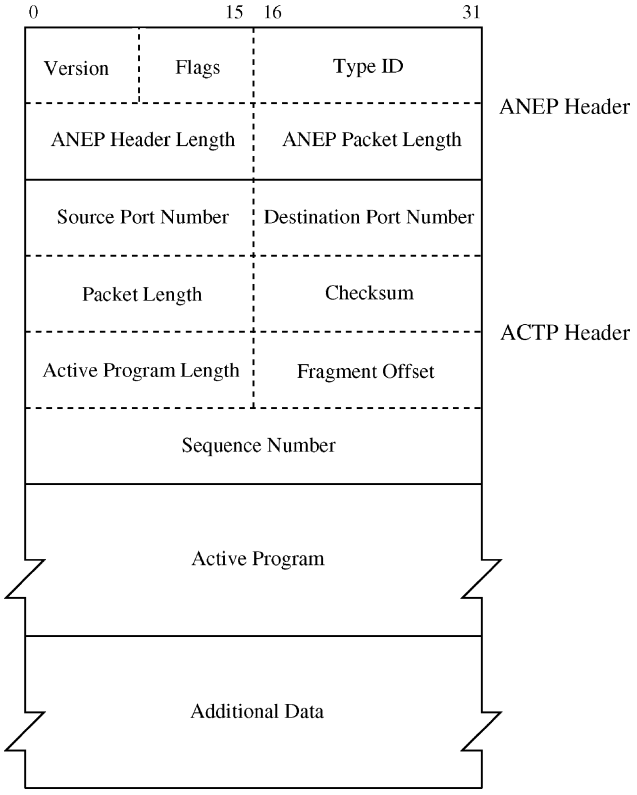

Fig. 2. Data format of active packets.

compared to using IP options. In order to provide interoperability, we wrap our ACTP packet with an ANEP header [35].

3.4. Features of ADNET

A unique feature of our design is the explicit consideration of resource management. We introduce the concept of *default resource reservation* to provide better services to the active traffic.

ADNET explicitly includes the CPU scheduler as one of the resource management component. It addresses simultaneously the issues of the overheads associated with per-flow accounting and customized services for individual traffic flow. ADNET seeks to provide a broad spectrum of services ranging from traditional stateless best-effort to IntServ-style per-flow accounting. In one extreme, per-flow customized service such as IntServ or active traffic can be provided. On the other extreme, traditional best-effort service can be provided. In the middle, aggregated QoS schemes such as DiffServ can be provided.

ADNET routers play different roles, depending on where they are located: (1) As *leaf (first hop) routers*: The sender simply sends any traditional traffic as usual and active traffic with or without reservation. If a premium service (as in DiffServ) is required, the sender sends an active program to the router to request resources. After validation, the request is granted by the router and the traffic from the sender is tagged as DiffServ traffic with specific DiffServ class (Fig. 3). (2) As *boundary (border) routers*: The egress router issues a setup request to the ingress router of ISP. (3) As *core routers*: The AEE only processes the active flows that are authorized to go across the domain and enter the current domain. The router does not accept Non-Active Explicit Reservation to ensure safety.

In Section 4, we describe a fragmentation scheme which is designed to support ACTP protocol to transfer a large volume of multimedia data using the architecture.

4. Fragmentation scheme for ACTP

ACTP does not rely on the underlying IP-fragmentation in network layer. Instead, it uses a new fragmentation scheme designed for multimedia data transmission in active environment, which addresses the unique need of active

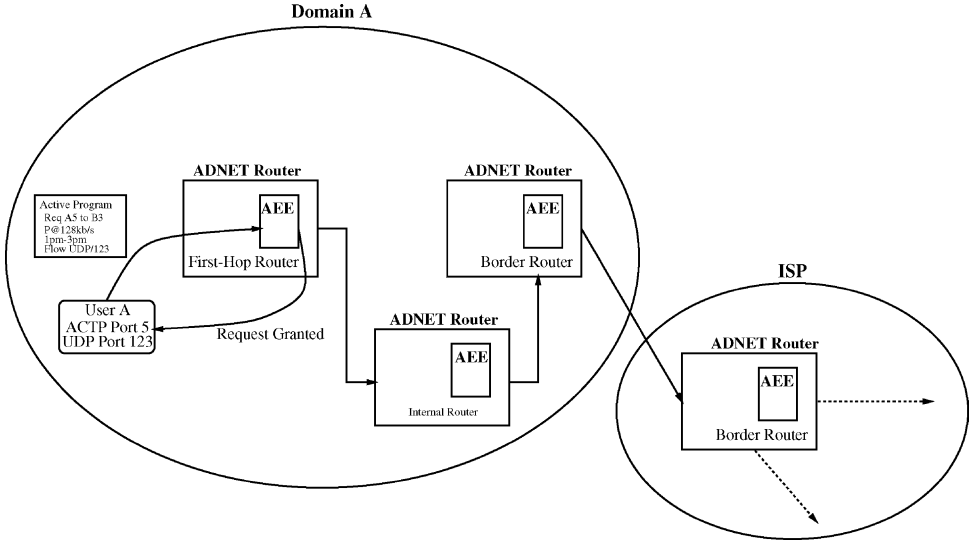

Fig. 3. Active execution environment as a bandwidth broker.

networks and utilizes the special properties of active networks to achieve the goals.

4.1. The need for fragmentation

Multimedia objects such as audio, video and image are usually very large in size. For example, one MPEG-1 movie in NTSC video quality (which we digitized from a VHS video tape) has an average size of 8617 bytes/frame. The average I-frame size is even larger (17,997 bytes/frame). When transporting multimedia data in the networks, the sizes of the payload are usually larger than the Maximum Transmission Unit (MTU) of the underling physical networks. For example, the MTU of Ethernet is 1500 bytes. It is inevitable that the multimedia data have to be fragmented into smaller-size units when transmitting through the networks. In the past, this issue was not particularly significant because the application programs can simply assume that the lower level protocols (such as IP and ATM) can reliably deliver reasonably large packet (64 KB in IP and unlimited size in ATM AAL5) and let the lower level protocols handle the fragmentation process.

In the recent years, the researches on active networks [36] show some potentials in providing better network services to the emerging applications which are rich in content and require large bandwidth. For example, video gateways, which used to be implemented in application-level, may be possible to be implemented using active networks [17]. However, since any network traffic, no matter it is from active network or not, has to be transported using conventional link-layer technologies such as Ethernet or ATM, the problem of fragmenting large packets still exists in active networks. Furthermore, some new issues emerge when active capsules are fragmented into smaller units. Therefore new fragmentation schemes are needed for the active communication environment.

4.2. Alternatives to the fragmentation of large data units

There are several alternatives to handle the fragmentation problem. They are:

- *Fragment at network layer*. This is the current practice when passive protocols such as TCP and UDP send large packets. The internals of the payload are not revealed to the network and the fragmentation points are decided arbitrarily and this task is up to the network layer protocol. IP fragmentation and ATM Adaptation Layer (AAL) are examples of this approach. A performance analysis of IPv6 fragmentation for multimedia traffic is discussed in [37]. The advantage of this approach is that the fragmentation process is transparent to the higher layer protocols (both transport and application layers).
- *Fragment at transport layer*. Transport layer can also fragment data into units of appropriate sizes. However, current practice of IP networks allow very large packet (64 KB) to be sent and let the IP layer handle the fragmentation.
- *Fragment at application layer*. The application layer is the best place to decide how to fragment the data because it has complete information about the data to be sent. The idea of *application-level framing* [38], *integrated layer processing* [39], and *trace-adaptive fragmentation* [40] explore some aspects of this alternative. However, applications may not be able to know the limitation of the underlining networks. Therefore it may not be able to make the best decision in case there are several alternatives available. Practically speaking, there is no mechanism in current network architectures to support the cooperation between the application programs and network protocol in dealing with data fragmentation.

Although active networks are promising network architectures for the next-generation networks, some issues specific to the active networks exist regarding to the fragmentation of data. They are:

- Fragments of the same capsule may travel different routes. The intermediate active routers may never have the chance to execute the active programs inside a capsule simply because the successful execution of the active programs needs all the fragments of the same capsule. This is not a problem in conventional IP networks because the reassembly only occurs at the receiver, not at the routers.
- Reassembly cost of active capsules in the intermediate routers may be high. If an active capsule can be processed only after all fragments are received by the active router and the active capsule is very large (which results in a lot of fragments), the time of waiting for the fragments may be very long for time-sensitive data and the overhead to reassembly the fragments and re-fragment them after processing can be high.
- If the active capsules are fragmented arbitrarily, the fragments may contain partial logical data units which increases the complexity of the active programs inside the capsules because the active programs have to be aware of the possibility of partial data due to fragmentation and have to handle the problem by themselves.

Although researchers of active networks recognize the importance of the fragmentation problem [41], the issues of fragmentation mechanisms in active networks have not been addressed in current research yet. The fragmentation scheme described in this paper represent our first attempt to address this important problem.

4.3. The proposed scheme

Our design goals of a fragmentation scheme for active networks are: (1) Fragmentation points should correspond to the natural boundaries inside the data which can

be meaningfully managed, if it is possible to do so. (2) Fragmentation policy should not divide a data into fragments unless it is necessary. (3) For the data that cannot be fragmented at the natural boundaries, the fragmentation/reassembly costs should be minimized.

The basic ideas of the proposed scheme are: (1) Instead of letting the network layer blindly fragment the application data, the application gives *hints* to the lower layer protocols on how the data can be fragmented. The application also tells the lower layer protocols how it handles the data in both fragmented and un-fragmented cases. (2) The network layer passes MTU size to the application and the application adjust its behavior to fit the network constraints whenever possible.

The ACTP Protocol (Section 3.3.4) at transport layer is a logical choice to combine and process both information from application layer and network layer. The advantages of handling the fragmentation process inside the active protocol are: (1) The fragmentation process can be done in an uniform, application-independent way. (2) By giving out hints only, the applications do not need to handle the complexity of actually fragmenting the data. (3) The active programs do not need to handle the complexity of dealing with partial data.

There are two types of hints that can be passed to the active protocol: fixed-size hint and variable-size hint. Fixed-size hint contains a single value which represents the required fragment size for the data. It is useful when the nature of the data requires that it can be divided into fixed-size fragments. For example, sound samples in an audio-conference are usually fixed-size data. Variable-size hint contains an array of breakpoints that can be applied to the data. Note that fixed-size hint is actually a special case of variable-size hint. Since we envision that there are many cases where fixed-size hint will be useful and the programming interface of fixed-size hint can be greatly simplified, we separate it out as another type of hint.

The actual fragmentation process occurs inside the active transport protocol ACTP. The fragmentation process utilizes the information from both the applications (hints) and network infrastructure (MTU). There are many possible ways to fragment the application data using the provided information. We provide an algorithm which minimizes the number of fragments generated based on the provided information. Fig. 4 present the algorithm in C-style pseudo-codes. The algorithm is optimal in terms of number of fragments (Theorem 1). The algorithm is designed based on the following constraints: (1) The algorithm tries to fill as many data sub-units (determined by the breakpoints inside the data unit) as possible into a fragment whose size is upper bounded by the maximum fragment size. (2) Partial data sub-units are not placed in the same fragment as whole data sub-units. (3) The data are fragmented at non-breakpoints only when it is impossible to fit the data into the maximum

```
//Input parameters:
//  breakpoints[] -- array of breakpoints in ascending order
//  data_size     -- data unit size
//  bc            -- number of breakpoints
//  mss           -- maximum fragment size
base = 0;
for (j = 0; j < bc; j++) {
  if ((breakpoints[j] - base) == mss) {
    Fragment(breakpoints[j], breakpoints[j]-base);
    base = breakpoints[j];
  }
  else if ((breakpoints[j] - base) > mss) {
    if ((j > 0) && (base != breakpoints[j-1])) {
      Fragment(breakpoints[j-1], breakpoints[j-1]-base);
      base = breakpoints[j-1];
      }
      if ((breakpoints[j] - base) > mss) {
        while ((breakpoints[j] - base) > mss) {
          base += mss;
          Fragment(base, mss);
        }
        if (breakpoints[j] != base) { /* still something left */
          Fragment(breakpoints[j], breakpoints[j]-base);
          base = breakpoints[j];
        }
      }
    }
}
if (base != data_size) { /* some more leftovers */
  Fragment(data_size, data_size-base);
}
```

Fig. 4. Algorithm for active fragmentation.

fragment size by using breakpoints only. (4) The data are not re-ordered within a data unit.

Theorem 1. *Algorithm given in* Fig. 4 *generates minimum number of fragments satisfying the given constraint and requirements*.

The proof is given in Appendix.

Programming interfaces. We propose the following new system calls to the socket API to provide the services described above.

- `int send_actp(int s, void *msg, int len, int hints[], void *seg_prog, VOID *UNSEG_MSG, UNSIGNED INT FLAGS)`—`send_actp()` is similar to the `send( )` system call in usual socket library. It is augmented with three more arguments, which pass the hints to the active protocol. The argument hints is an array of breakpoints. If `hints` is for fixed-size hint, a single negative integer value will be provided whose absolute value is then used by the active protocol as the proposed fragmentation size. The argument `seg_prog` and `unseg_msg` correspond to the active programs which handle the fragmented and un-fragmented data.
- `int getsockopt(int s, IPPROTO_ACTP, ACTP_MTU, void *optval, int *optlen)`—A new socket option at the active protocol layer (or *level* in terms of socket library convention) called ACTP_MTU is introduced. When the application calls `getsockopt( )` to retrieve MTU information, active protocol can perform a path-MTU discovery to retrieve this information and cache it for future use.

An example. To illustrate the fragmentation scheme, consider the case when one frame of a MPEG video is to be sent via an ACTP/IP/Ethernet network. The size of the packet is 5300 bytes, which is larger than the maximum fragment size of 1286 bytes (1286 bytes is the MTU of Ethernet minus necessary spaces for IPv6 header, ANEP header [35] and active programs). The application pass the list of hints {8, 1251, 2433, 4096, 5300} to ACTP and ACTP obtains the constraint of maximum fragment size (1286 bytes) from the network layer. The list of hints corresponds to the offsets from the beginning of the frame where a new slice of macro-blocks starts. The first fragmentation point is 1251 because the maximum possible sub-units from the beginning of the data that can fit into an 1286 bytes fragment is at 1251 bytes. The next fragmentation point is 2433 because the fragment will be too large if ACTP fragments at the next breakpoint. The next fragmentation point is 3719, which dose not correspond to any breakpoints in the hint list since data sub-unit between offset 2433 and 4096 is too large for a 1286 bytes fragment; ACTP has no choice but to fragment at the middle of the data sub-unit. Finally, 4096 and 5300 are the last two fragmentation points.

5. Usefulness: a QoS metric

Packet loss rate, link utilization, goodput, transmission latency, jitter, signal-to-noise ratio, queue length, and subjective perpetual quality are some commonly used QoS metrics for assessing the effectiveness of multimedia data transmission protocols. However, research results have shown that these parameters cannot reflect true QoS perceived by the end-user [42,43]. The problem is that currently there are two important issues for multimedia data transmission which have not been resolved satisfactorily. They are:

- System-level integration of error control and concealment techniques. The source coding, transport protocol and post-processing techniques should be designed together to achieve the optimal performance.
- Adaptable source-coding and transport-control mechanisms. In current error concealment approaches, there is a very little interaction between the source coder and the transport layer. An optimal system should be adaptable to the error characteristics of the networks and dynamically shift the burden of error concealment between the source coder and the transport layer.

In this section, a general model is developed for evaluating the application of active techniques such as transcoding [44] and active fragmentation (Section 4) to the compressed video transmission problem. Systematically, a metric called *usefulness* is developed which measures the portion of data received by the receiver that are *usable*. A fragmentation scheme is necessary for this framework. Considering active networks, packetized video data may go through a series of transformations before it reaches the destination. The objective of these transformations is to reduce the data size when congestion occurs to adapt to the network status. However, since each packet acts independently when travelling through the network, the decoder must be able to deal with packets from the same frame which are transformed differently when decoding the video frame. The active fragmentation scheme is very useful in this case.

Assume a data stream consists of n frames. The term 'frame' is used to refer to a complete data unit which can possibly be divided into smaller units. For example, an MPEG video frame can be divided into slices. The size of frame i is denoted by f_i, and the number of fragments generated from this frame is m_i. For IP fragmentation, m_i can be expressed as $\lceil f_i/M \rceil$ where M denotes the maximum fragment size (or maximum transport unit, MTU) of the underlying transport network. For ACTP fragmentation, m_i depends on both MTU and the hints provided by the application.

Each frame is assumed to be sent at time $t_{i,j}$ where i is the frame number and j is the fragment number of the frame i (i ranges from 1 to n and j ranges from 1 to m_i). If no packet

is lost and the router is not active (i.e. the router does not process the packets, it just forwards them), the total amount of data received by the destination D is $\sum_{i=1}^{n} f_i$. Assume that the router applies a transform $\Gamma(s,l(t))$ to the packet of size s at time t with the load lt). In the simplest case, the router just forwards the packet or drops it based on the load of the router (for example, the load can be derived from the queue size at time (t). Therefore, the value of $\Gamma(s,l(t))$ can either be 0 or s depending on the size of the queue. In the active router case, the possible value of $\Gamma(s,l(t))$ can be anything. Since this paper focuses on those cases where the network is congested, it makes sense to restrict the value of $\Gamma(s,l(t))$ to be between 0 and s.

Based on the above definitions, the actual amount of the data received by D is

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i}\Gamma(s(i,j),l(t_{i,j})).$$

the function $s(i,j)$ is the size of the packet received by the router and takes the value M most of the time except for the last fragment for each frame.

Not all data received by D is useful. For example, in TCP/IP protocol suite, the reassembly of the packet occurs at the IP layer. If any one of the fragments is lost, all other fragments from the same frame are useless. To model the usefulness of the received data, assume that the receiver D applies a function $U(\alpha, \beta)$ to the data fragments received for the same frame. Let α be the total data received for a particular frame. That is $\alpha_i = \sum_{j=1}^{m_i}\Gamma(s(i,j),\ l(t_{i,j}))$ for frame i. β is the frame size. For frame i, $\beta_i = f_i$. Since the importance of each frame is different, a weight W_i is associated with frame i.

Finally, the *usefulness* of the data received is defined as

$$\gamma = \frac{\sum_{i=1}^{n} W_i U(\alpha_i,\beta_i)}{\sum_{i=1}^{n} W_i f_i}.$$

For simplicity, W_i is assumed to be 1 for all i. Now, consider the following two cases.

Case 1. Traditional router and IP fragmentation. The usability function for IP fragmentation is

$$U(\alpha,\beta) = \begin{cases} \beta & \text{if } \alpha \geq \beta, \\ 0 & \text{if } \alpha < \beta. \end{cases}$$

Simply speaking, if not all of the fragments are received, none of the fragments are useful. Assume that the packets are lost with a fixed probability p. The loss of a packet could be the result of the router buffer overflow or reassembly timeout at the destination. For simplicity of analysis, let's assume the usability function to be the identity function $U(\alpha_i,\beta_i) = \alpha_i$. Then the expected lost data is

$$E_i = \min\left(\left\lceil\frac{f_i}{M}\right\rceil \times p, 1\right) \times \sum_{j=1}^{\left\lceil\frac{f_i}{M}\right\rceil}\Gamma(s(i,j),\ l(t_{i,j})).$$

and the *usefulness* of the data received is

$$\gamma_{\text{IP}} = 1 - \frac{\sum_{i=1}^{n} E_i}{\sum_{i=1}^{n} f_i}.$$

Case 2. ACTP fragmentation. Under ACTP fragmentation, the usability function is different for different types of data. For text or numeric data, the usability model is same as that of IP fragmentation. For multimedia data such as video or image, an approximate usability function is given by the following empirical equation.

$$U(\alpha,\beta) = \beta \times e^{-5((\alpha/\beta)-1)^2}.$$

A graphical representation of this equation is given in Fig. 5. The empirical equation is developed by using the evidences from the researches on user-perceived QoS [42] and our experimental data on various video/image data.

Now the expected lost data is

$$E_i = p \times \sum_{j=1}^{m_i}\Gamma(s(i,j),\ l(t_{i,j}))$$

and the *usefulness* of the data received is

$$\gamma_{\text{ACTP}} = \frac{\sum_{i=1}^{n} U(f_i - E_i, f_i)}{\sum_{i=1}^{n} f_i}.$$

Remarks:

- Our approach to the evaluation of usefulness in multimedia data is unique in the sense that we consider the data in the sub-frame level. Other work either consider the data at the packet level (each physical packet is a unit of data) or at the frame level (a frame is a data unit). For example, in [15] two active techniques (Frame-level Discard and GOP-level Discard) are investigated to address the problem of network congestion. The objective of their work is very close to ours. Frame-level Discard mechanism queues a datagram if and only if its corresponding frame can be queued in its entirety. GOP-level Discard mechanism maintains a state about

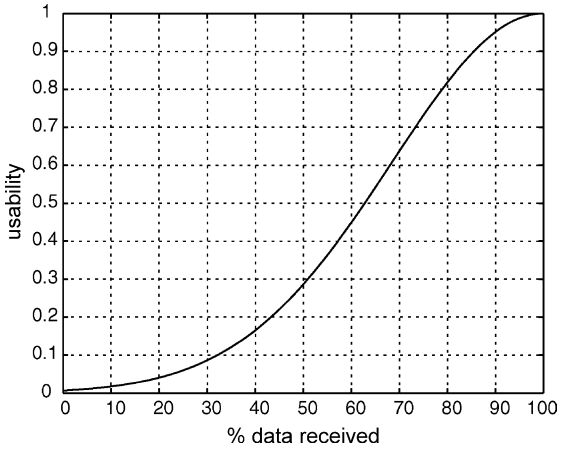

Fig. 5. Usability function for multimedia data.

Table 1
Profiles of three video clips

Clip name	Jurassic Park	Tai Chi	Lion King
Resolution	320×240	352×240	320×240
# of frames	3000	2996	3750
Total size	23,091,162	12,497,528	10,273,216
Avg. I—frame size	19,458.98 (251)	7684.57 (334)	8056.66 (313)
Avg. P—frame size	14,423.05 (750)	5632.38 (666)	4752.44 (938)
Avg. B—frame size	3695.67 (1999)	3094.44 (1996)	1317.00 (2499)
Overall avg. frame size	7696.38	4170.34	2738.85
GOP pattern	IBBPBBPBBPBB	IBBPBBPBB	IBBPBBPBBPBB

the type of frame discarded. In case an I-frame is discarded, the other frames (P-frames and B-frames) in the same group of pictures are discarded as well. Since any partially received frame contributes to the usefulness, aggressive discarding schemes such as those in [15] may result in lower application QoS. Specifically, if a frame is buffered only in its entirety, many I-frames will not be available to the applications in the period of heavy network congestion. In addition, if all the frames in a group of pictures are dropped altogether, the application will experience a very jittery video period because a group of pictures usually consists of more than 10 frames and lasts for more than 1/3 s.

- The criteria used in other research works for the evaluation of the effectiveness of MPEG video transmission algorithm are frame-level parameters such as the percentage of complete I-frame received [18,45]. We consider a partial frame to be useful if it can be repaired by some error-concealment techniques. From the experiment results, we observe that a repaired I-frame and the corresponding P-frames and B-frames in the same GOP can make the quality of the video playback very close to the case in which no data is lost.
- Usability function U is very similar to utility function in the literature. However, the usability function is applied to the packets at the system level. It is different in the sense that the function is calculated from the system point of view. Therefore the same amount of data received by the network software in OS can lead to very different usefulness level for the applications depending on the protocols used and other system parameters.

6. Experimental results

We are presenting simulation results for a part[1] of the functionality of ADNET using *ns-2* network simulator. We have extended *ns-2* to include certain functionalities described in this paper.

[1] Other results will be presented gradually in separate papers.

6.1. Test data

To perform experiment under realistic scenario, following three digital (MPEG) video clips are created from commercial video tapes.

1. *Jurassic Park* (JP)—A typical movie with different types of scenes. The characteristics of this video clip are similar to most of the other videos in the market.
2. *Lion King* (LK)—A cartoon video. The distinguishing feature of a cartoon video is that there are many sharp edges. Compressing a cartoon is usually more difficult because the MPEG scheme works better on real-world objects.
3. *Tai Chi* (TC)—An instruction video which teaches Tai Chi Chun (a form of Chinese Martial Art). This video is a typical instruction video. The distinguishing feature of instruction videos is that usually there are no fast changing scenes and background. The changes in the video are slow because most of the scenes involve only motions of the instructor.

Table 1 gives the details of these video clips.

6.2. Comparison between ACTP and UDP

A prototype video application suite is implemented to conduct the experiments. The suite includes a video encoder/server, an active router emulator, and a video decoder with error concealment modules. Another module is included to simulate the proposed transport protocol ACTP. Experiments are conducted to determine the effect of ACTP fragmentation scheme on QoS in adaptable video-on-demand applications. Fig. 6 shows the setup of the experiments.

A 300-frame video clip from Jurassic Park is used as the input video. The experiments are conducted using frame rate of 3, 2 and 1 frames/s. For each experiment of frame rate n, a timer is set to expire at every $1/n$ second in the client program. When the timer expires, the expected frame number is increased by one and the data received for previous frames are considered late and discarded. As depicted in Fig. 6, the video data are sent from a video server running at the machine *raid4*. The video traffic goes through a simulated active

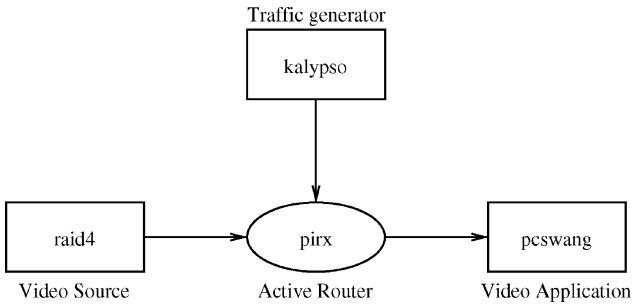

Fig. 6. Experiment setup to transmit video in RAID lab. raid4.cs.purdue.edu sends video files through pirx.cs.purdue.edu—an active router—to pcswang.cs.purdue.edu

router program running at the machine *pirx* and is received by the video client program running at the machine *pcswang*. The results of ACTP simulation are compared with traditional UDP/IP approach (Table 2).

The results show that in UDP session for 3 frames/s, the average timer expiration is 182 times, therefore in average only 118 out of the 300 frames are received and playable. In contrast, under ACTP session the timer expired 237 times on average, but there are 287 playable frames on average because most of the frame are partially received when the timer expired and most of these frames can be made playable using simple error concealment techniques in the client program. By conducting real experiment, we see that the ACTP session provides a very smooth playback, although there is some occasional block noise during the playback due to the lost slices. In contrast, regular playback session is very jittery because a lot of frames are not available. A demo is available to visually see the impact of ACTP vs. non-ACTP approach on video quality. Extractions of some video frames of ACTP and UDP sessions are shown in Fig. 7. By visually comparing the two figures, the difference in the user-perceived QoS between the two approaches is very apparent. Usefulness, γ, with varying packet loss probability for clip JP is given in Fig. 8. Usefulness of ACTP session is much better than that of IP session.

6.3. Comparison between ACTP Fragmentation and IP Fragmentation

In this experiment, we compare three different scenarios of video transmissions: (1) sending the video one frame at

Table 2
Comparison of ACTP and UDP video session

	Frame rate	Timer expired	Playable frame	Γ	Perceptual quality
ACTP	3	237	287	0.80	Good
	2	248	290	0.80	Good
	1	235	286	0.83	Good
UDP	3	182	118	0.17	Bad
	2	187	113	0.16	Bad
	1	185	115	0.16	Bad

(a) IP Sequence

(b) ACTP Sequence

Fig. 7. Sequence of video clips using IP and ACTP fragmentation. The ACTP fragmentation maintains same quality among all clips where as the IP fragmentation suffers from severe QoS degradation in d006.jpg, d007.jpg, d008.jpg, d010.jpg, d011.jpg, d015.jpg, d016.jpg, and d017.jpg clips.

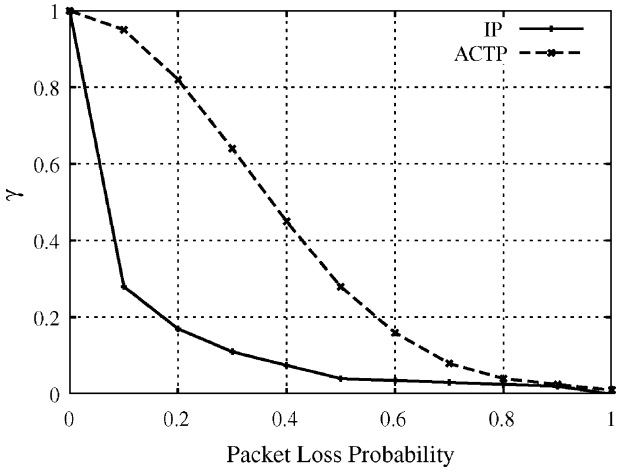

Fig. 8. Usefulness for clip Jurassic Park with varying packet loss probability.

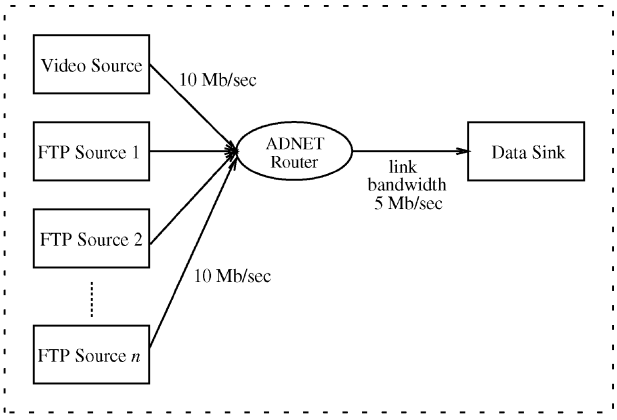

Fig. 9. Simulation setup for ADNET experiments. The ADNET router is the active router. FTP applications create background traffic.

a time at the rate of 30 frames/s under IP fragmentation, (2) sending the video at the same rate under ACTP fragmentation, (3) sending the video at the same rate under ACTP fragmentation and with active program in each packet (50 bytes).

The active program queries the current status of the system queue. If addition of the packet causes the queue to drop packets, the packet is handed over to the active execution environment and *transformed* to 80% of its original size by sacrificing the quality of the video frame. The transformation causes a delay of 5 ms for the packet.

The configuration of the experiments is shown in Fig. 9. The bottleneck link is configured at 5 MB/s. This value is large enough for the video source to send the video at 30 frames/s without any loss when no competing traffic source exists. The experiments are repeated for the configuration of 1–5 competing traffic sources (FTP sources).

Figs. 10–12 show the results of the experiments. Byte loss rates for the three scenarios are almost the same (Fig. 10). Since in the ADNET routers the queuing discipline is DRR, it is expected to be fair to all traffic flows. However, the packet loss rates for ACTP fragmentation with active transformation is less than that for the other cases (Fig. 11). By reducing the size of the packet, the packet has a higher chance of surviving and the loss rates are reduced by around 10%. The most exciting results are that the QoS observed by the applications (usefulness, γ) improves significantly for ACTP fragmentation, and even more for ACTP fragmentation with transformation under heavy network load.

6.4. Overhead of the fragmentation scheme

The purpose of this experiment is to get an idea of how many extra bytes are needed in ACTP fragmentation. We choose the maximum fragment size to be 1286, which is the MTU of Ethernet minus the spaces for IPv6 Header, ANEP header and an active program of 100 bytes. We pass a hint list consists of the offsets from the beginning of a video frame which correspond to the starting points of the slices inside the video frame. Slices are the largest meaningful sub-units inside a video frame. The application and the active programs know how to deal with slices directly. It makes sense to fragment the video frames at the boundary of slices to reduce the complexity of active programs. Thus the processing of the fragments is optimized and can be done independently.

The results are given in Figs. 13 and 14. The result shows that the average overhead ranges from 5.66% (clip JP, MFS = 1236) to 0.40% (clip TC, MFS = 1436) of the total data size, and the maximum overhead for a single frame range from 27.77 to 2.68%. The worst case occurs when the video frame consists of a lot of slices, which are slightly larger than the maximum fragment size. From these experimental results, we see that the proposed scheme is performing well, with low overhead, under the realistic experiment scenario.

7. Conclusions and future works

The paper investigated techniques to solve multimedia data transmission problem using active network paradigm. We developed an adaptable network architecture which allows different QoS provision schemes such as active networks, integrated services, and differentiated services to

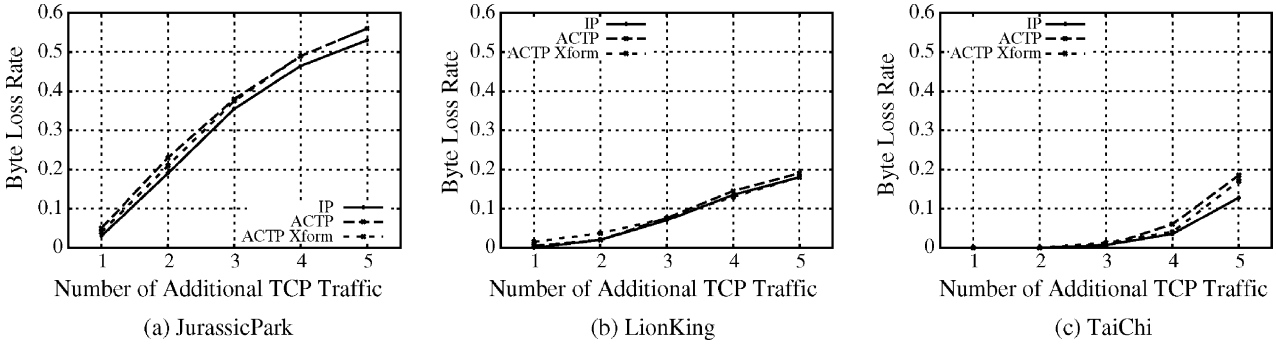

Fig. 10. Byte loss rates under various fragmentation schemes and video files. Byte loss rates for the three scenarios are almost same. Jurassic park experiences much higher loss than other two videos.

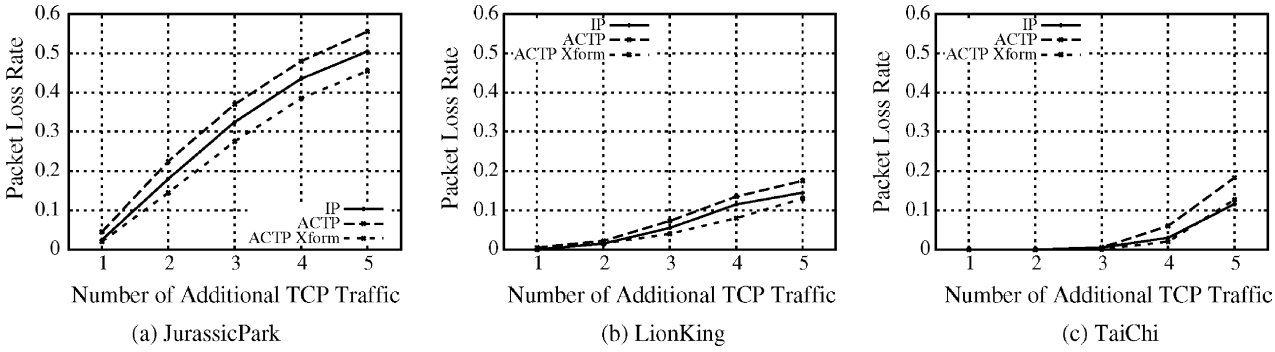

Fig. 11. Packet loss rates under various fragmentation schemes and video files. The packet loss rates for ACTP fragmentation with active transformation is reduced by 10% than the other schemes. Because it reduces the size of the packet, which increases the chance of surviving the packet.

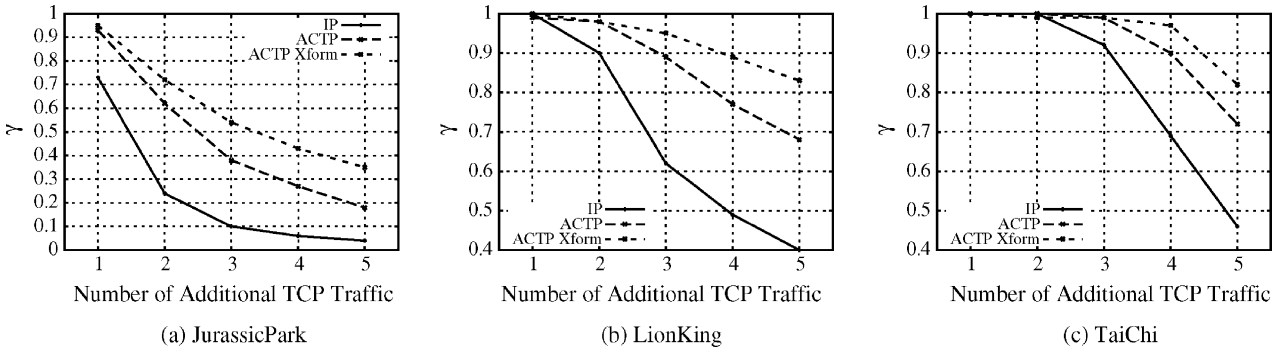

Fig. 12. Usefulness γ, for various fragmentation schemes and video files. ACTP fragmentation with transformation has the highest usefulness for all three experiments. Additional TCP traffic makes the network congested and lowers the usefulness.

co-exist. Thus, the proposed architecture can be used in monitoring and controlling all types of incoming traffic inside a network domain. This architecture enables both the application programs and the networks to adapt and perform the tasks of traffic management and control together.

A fragmentation scheme is proposed to address the unique need of active networks and utilizes the special properties the new infrastructure provides. In order to provide a better assessment of QoS received by the applications, a new metric to quantify the user-perceived QoS is introduced. It is shown to provide a more accurate assessment of the user-perceived QoS than other commonly used measures.

In our future work, we plan to conduct rigorous experiments on interoperability of the IntServ and DiffServ with active paradigm. The long-term goal is to integrate the DiffServ architecture into the active network architecture as a special case. It may even be possible to integrate the IntServ architecture to create a unified traffic class.

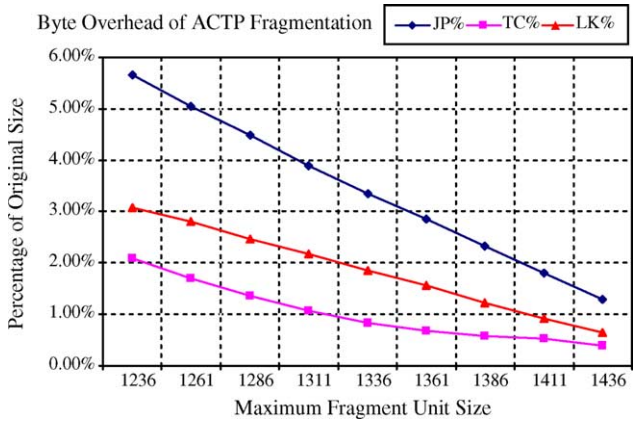

Fig. 13. Average byte overhead of ACTP fragmentation for three video files. The overhead is very low for 1435 fragment size.

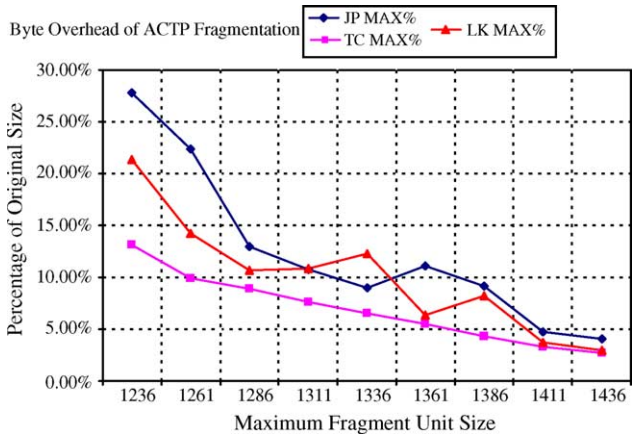

Fig. 14. Maximum overhead of ACTP fragmentation. Even though the maximum overhead is very high for 1235, it is less than 35% for 1435 fragment size.

Appendix

Proof. (Theorem 1) The breakpoint preceding a data sub-unit with size, T, larger than the maximum fragment size M is a fragmentation point. Since no partial data sub-unit is combined with a whole data sub-unit, the next breakpoint is also a fragmentation point. In this case, at least $\lceil T/M \rceil$ fragments are needed and the algorithm produces exactly $\lceil T/M \rceil$ fragments.

Following the above reasoning, the only case we need to consider is where all the data sub-units have sizes which are less than the maximum fragment size. Let the algorithm's solution A produces a sequence of fragmentation points $a_1,a_2,\ldots,a_n$. Assume that solution A is not optimal. Let S be the optimal solution, with the sequence of fragmentation points $s_1,s_2,\ldots,s_m$. $m<n$ since S is optimal. Now, $s_1<a_1$ because the algorithm requires that a_1 contains the maximum possible breakpoints that can be included. $s_2 \leq a_2$ because (a_1,a_2) contains the maximum possible breakpoints that can be included, and (s_1,a_2) contains at least one more breakpoint than (a_1,a_2). Following similar arguments, we conclude that $s_i \leq a_i$ for all $1 \leq i \leq m$. Now consider the last fragment of solution S. It starts from offset s_m to the end of the data. Since $s_m \leq a_m$ and $m<n$, there is a $k \leq n$ such that $s_m < a_k$. Then from the requirement of the algorithm there are at least two fragments between s_m and the end of the data because it require at least two fragments between a_m and the end of the data. But this is a contradiction because by assumption s_m is the last fragmentation points. Therefore, the given algorithm produces minimum number of fragments. □

References

[1] P.W. Agnew, A.S. Kellerman, Distributed Multimedia: Technologies, Applications, and Opportunities in the Digital Information Industry, ACM Press, 1996.

[2] S. Li, Quality of service control for distributed multimedia systems, PhD Thesis, Department of Computer Science, Purdue University, December 1997.

[3] O. Spaniol, J. Meggers, Active networks nodes for adaptive multimedia communication, in: Fifth International Conference on Intelligence in Networks (SMARTNET 99), 1999.

[4] M. Uruena, D. Larrabeiti, M. Calderon, A. Azcorra, J.E. Kristensen, L.K. Kristensen, E. Exposito, D. Garduno, M. Diaz, An active network approach to support multimedia relays, in: Joint International Workshop on Interactive Distributed Multimedia Systems/Protocols for Multimedia Systems (IDMS-PROMS), 2002.

[5] Y. Li, L. Wolf, Adaptive resource management in active network nodes, in: Proceedings of IEEE Symposium on Computers and Communications (ISCC'2003), 2003.

[6] D.J. Wetherall, J. Guttag, D.L. Tennenhouse, ANTS: a toolkit for building and dynamically deploying network protocols, in: Proceedings of IEEE OPENARCH'98, San Francisco, California, 1998.

[7] M. Hicks, P. Kakkar, J.T. Moore, C.A. Gunter, S. Nettles, PLAN: a packet language for active networks, in: Proceedings of the Third ACM SIGPLAN International Conference on Functional Programming Languages, 1998, pp. 86–93.

[8] M. Hicks, J.T. Moore, D. Wetherall, S. Nettles, Experiences with capsule-based active networking, in: Proceedings DARPA Active Networks Conference and Exposition, San Francisco, California, 2002, pp. 16–24.

[9] D.L. Tennenhouse, D.J. Wetherall, Towards an active network architecture, in: Proceedings DARPA Active Networks Conference and Exposition, San Francisco, California, 2002, pp. 2–15.

[10] T. Wolf, J. Turner, Design issues for high performance active routers, IEEE Journal on Selected Areas of Communications—Special Issue on Active and Programmable Networks 19 (3) (2001) 404–409.

[11] T. Fuhrmann, T. Harbaum, M. Schoeller, M. Zitterbart, AMnet 2.0: an improved architecture for programmable networks, in: Proceedings Fourth Annual International Working Conference on Active Networks, Zurich, Switzerland, 2002.

[12] S. Karlin, L. Peterson, VERA: an extensible router architecture, Computer Networks 38 (3) (2002) 277–293.

[13] F. Kuhns, J. DeHart, A. Kantawala, R. Keller, J. Lockwood, P. Pappu, D. Richards, D. Taylor, J. Parwatikar, E. Spitznagel, J. Turner, K. Wong, Design of a high performance dynamically extensible router, in: Proceedings DARPA Active Networks Conference and Exposition, San Francisco, California, 2002, pp. 42–64.

[14] S. Bhattacharjee, K.L. Calvert, E.W. Zegura, An architecture for active networking, in: High Performance Networking (HPN'97), White Plains, New York, 1997.

[15] S. Bhattacharjee, K.L. Calvert, E.W. Zegura, Congestion control and caching in CANES, in: Proceedings of ICC'98, Atlanta, GA, 1998.

[16] N. Prabhavalkar, M. Parashar, Controlling unresponsive connections in an active network architecture, International Journal of Network Management 13 (4) (2003) 289–305.

[17] S. Li, B. Bhargava, Active gateway: a facility for video conferencing traffic control, in: Proceedings of COMPSAC'97, Washington, DC, IEEE, 1997, pp. 308–311.

[18] S. McCanne, V. Jacobson, Receiver-driven layered multicast, in: Proceedings of the ACM SIGCOMM, Stanford, CA, 1996.

[19] S. Kang, H.Y. Youn, Y. Lee, D. Lee, M. Kim, The active traffic control mechanism for layered multimedia multicast in active network, in: Proceedings Eighth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, San Francisco, CA, 2000.

[20] D.S. Alexander, W.A. Arbaugh, M.W. Hicks, P. Kakkar, A.D. Keromytis, J.T. Moore, C.A. Gunter, S.M. Nettles, J.M. Smith, The switchware active network architecture, IEEE Network Magazine 12 (3) (1998) 29–36.

[21] S.L. Murphy, E.T. Lewis, R.N. Watson, Secure active network prototypes, in: Proceedings DARPA Active Networks Conference and Exposition, San Francisco, CA, 2002, pp. 166-181.

[22] B. Braden, D. Clark, S. Shenker, Integrated Services in the Internet Architecture: An Overview, RFC 1633.

[23] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, An Architecture for Differentiated Services, RFC 2475.

[24] K.L. Calvert, S. Bhattacharjee, E. Zegura, J. Sterbenz, Directions in active networks, IEEE Communications Magazine 36 (10) (1998) 72–78.

[25] D.S. Alexander, W.A. Arbaugh, A.D. Keromytis, J.M. Smith, A secure active network environment architecture: realization in switchware, IEEE Network Magazine 12 (3) (1998) 37–45.

[26] S. Kent, R. Atkinson, Security Architecture for the Internet Protocol, RFC 2401.

[27] J.-Y. L. Boudec, Some properties of variable length packet shapers, in: Proceedings SIGMETRICS/Performance, 2001, pp. 175-183.

[28] A. Habib, B. Bhargava, Network tomography-based unresponsive flow detection and control, in: Proceedings IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS '03), 2003, pp. 258-264.

[29] P. Tullmann, M. Hibler, J. Lepreau, Janos: A java-oriented os for active network nodes, in: Proceedings DARPA Active Networks Conference and Exposition, San Francisco, CA, 2002, pp. 117-129.
[30] N. Shalaby, Y. Gottlieb, L. Peterson, M. Wawrzoniak, Snow on Silk: A NodeOS in the Linux kernel, in: Proceedings Fourth Annual International Working Conference on Active Networks, Lecture Notes in Computer Science, Springer, Zürich, Switzerland, 2002, pp. 1-19.
[31] B. Schwartz, A. Jackson, T. Strayer, W. Zhou, R. Rockwell, C. Partridge, Smart packets: applying active networks to network management, ACM Transactions on Computer Systems (TOCS) 18 (1) (2000) 67–88.
[32] P. Pappu, T. Wolf, Scheduling processing resources in programmable routers, in: Proceedings of IEEE Infocom, New York, New York, 2002.
[33] S. Kent, R. Atkinson, IP Authentication Header, RFC 2402.
[34] S. Kent, R. Atkinson, IP Encapsulating Security Payload (ESP), RFC 2406.
[35] D.S. Alexander, B. Braden, C.A. Gunter, A.W. Jackson, A.D. Keromytis, G.J. Minden, D. Wetherall, Active network encapsulation protocol (anep).
[36] D.L. Tennenhouse, J.M. Smith, W.D. Sincoskie, D.J. Wetherall, G.J. Minden, A Survey of Active Network Research, IEEE Communications Magazine 35 (1) (1997) 80–86.
[37] D. Grimm, Evaluation of IPv6 fragmentation for RTP streaming video, in: Australian Telecommunications, Networks and Applications Conference (ATNAC '03), 2003.
[38] D.D. Clark, D.L. Tennenhouse, Architectural considerations for a new generation of protocols, in: Proceedings of the ACM SIGCOMM, 1990.
[39] T. Braun, C. Diot, Protocol Implementation using integrated layer processing, in: Proceedings of the ACM SIGCOMM, 1995.
[40] F. Li, I. Nikolaidis, Trace-adaptive fragmentation for periodic broadcast of vbr video, in: Proceedings of Ninth International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '99), 1999.
[41] J.B. Evans, G.J. Minden, Active Networking Archive, URL http://www.ittc.ukans.edu/Projects/ActiveNets (1999).
[42] C.J. Van den Branden Lambrecht, O. Verscheure, Perceptual quality measure using a spatio-temporal model of the human visual system, in: Proceedings of the SPIE 96, San Jose, CA, 1996.
[43] O. Verscheure, X. Garcia, G. Karlsson, J.-P. Hubaux, User-Oriented QoS in Packet Video Delivery, IEEE Network Magazine 12 (6) (1998) 12–21.
[44] E. Amir, S. McCanne, H. Zhang, An Application level video gateway, in: ACM Multimedia'95, San Francisco, CA, 1995.
[45] J.-C. Bolot, T. Turletti, I. Wakeman, Scalable feedback control for multicast video distribution in the internet, in: Proceedings of the ACM SIGCOMM, London, England, 1994.