A framework for programmable overlay multimedia networks

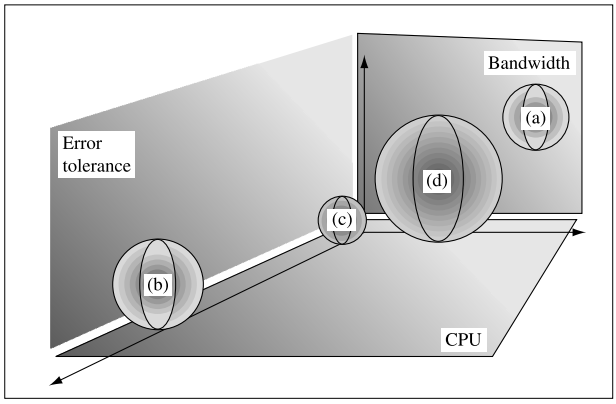
by N. R. Manohar
A. Mehra
M. H. Willebeek-LeMair
M. Naghshineh

We present an evolutionary framework for provisioning enhanced network-based multimedia distribution services to a diverse set of receivers. The framework addresses various degrees of heterogeneity among "distribution paths" from content sources to receivers, where a distribution path carries information from a content-aware multicast distribution tree from a source to one or more receivers. The heterogeneity is characterized by a spectrum of tradeoffs among receiver/client processing capabilities, network conditions, and media representations of desired content. Our goal is to provide individual receivers with the best feasible quality and representation of subscribed multimedia content by efficiently generating distribution paths suiting groups of receivers as they join a multicast. Toward this end, the framework utilizes several evolutionary measures. First, we introduce explicit awareness of receiver capabilities (e.g., bandwidth, CPU processing power) and link characteristics (e.g., loss rate, error rate) into the network. Second, we decouple media content from its representation to allow the network to generate and distribute multiple representations for the same content. Last, we model the distribution of content from sources to receivers as a relay across multiple content-aware intermediaries that cooperate to overlay a programmable multimedia network on the underlying packet-switched network. The multimedia overlay network thus formed provides sophisticated content-based programmability over the provisioning, routing, and management of media flows. Realization of such a multimedia overlay network requires extensions to existing Internet multicast and resource-provisioning technologies.

1. Introduction

Traffic in the next generation of the Internet is expected to be rich in multimedia content [1, 2], thanks to quality of service (QoS) [3, 4], mechanisms for resource reservation [5], and high-bandwidth pipes [6, 7], as well as increased expectations [8]. At the same time, the tremendous proliferation of multimedia content, multimedia formats, and multimedia-capable devices demand a rethinking of fundamental distribution mechanisms found in the current Internet [9]. The next

0018-8646/99/$5.00 © 1999 IBM



— For position only —

Figure 1

The receivers (a, b, c, d) in our example lie on an irreducible 3D operational space. The notion of the operational space of receivers is used to capture the spectrum of differences among receivers from the viewpoint of the network in a formal way. Note that this example defines a receiver to be the bundling of a client together with its link. A projection from this space results in a set of incompatible receivers being grouped together. This extreme case results because the receivers span the base vectors of the operational space. The base vectors of error tolerance (TOL), processing power (CPU), and available bandwidth (B/W) define the operational space of these receivers.

generation of multimedia networks should be capable of efficiently mitigating the interaction of multiple performance-limiting factors during the distribution of the new multimedia traffic payloads. Examples of performance-limiting factors include receiver capabilities such as access bandwidth and media processing power, and link capabilities such as loss and error rates.

Consider the following scenario, in which the same content is to be distributed to a diverse group of receivers. In particular, consider the following types of receivers: (a) network computers (NC), (b) personal computers (PC), (c) palmtops (PDA), and (d) workstations (WS). On one hand, NCs are relatively low-CPU processing nodes connected via a high-bandwidth pipe at low error rates, whereas PCs have more CPU processing power but are connected via a shoestring-bandwidth pipe at higher error rates. Similarly, workstations are typically very-high-CPU processing nodes connected via a very-high-bandwidth pipe at very low error rates, whereas palmtops are very-low-CPU processing nodes connected via a very-small-bandwidth pipe at typically high error rates.

Today's Internet is simply not capable of multicasting multimedia content to such receivers. Consider the operational space of these receivers (**Figure 1**) in terms of three primary performance-limiting factors over the performance and efficiency of end-to-end delivery:

- Available bandwidth (B/W).
- Receiver computational performance (CPU).
- Error tolerance (TOL).

Whereas the CPU axis characterizes the ability of a receiver to process a given media type, the B/W and TOL axes characterize the end-to-end transport and the media. A point on this space characterizes the set (receiver, media, and link).

These particular receivers do not lie within a linear or planar region. On one hand, in terms of increasing bandwidth, these receivers would be ordered as (c, b, a, d); on the other hand, in terms of increasing processing power, they would be ordered as (c, a, b, d). Moreover, with respect to decreasing error tolerance, they would be ordered as (c, b, d, a). A projection from this space results in a set of incompatible receivers being grouped together. Moreover, the resulting performance compromise across such grouped receivers is neither specifiable nor controllable in today's Internet. End-to-end provisioning to such widely heterogeneous receivers requires network intelligence to accommodate the differences in receiver connectivity, processing capabilities, and prevalent error rates. In general, making the network aware of such receiver disparities enables it to provision the same content in formats/representations customized for the needs and preferences of individual receivers.

For example, if audio content were to be broadcast to receiver types (a, b, c, d), the network could provision a with a relatively high-bandwidth representation of low decoding overhead and high tolerance to errors (e.g., phone-quality μ-law[1] audio); b with a relatively lower-bandwidth representation but with higher decoding overhead and lower tolerance to errors (e.g., Internet-quality audio); c with a negligible-bandwidth representation of negligible decoding overhead and high tolerance to errors (e.g., text transcripts); and d with a high-bandwidth representation regardless of decoding overhead and tolerance to errors (e.g., CD-quality audio). Thus, to efficiently accommodate the operational characteristics of diverse receiver types such as $a \cdots d$, multiple media representations of a given media content may have to be generated and distributed.

A straightforward solution would require each content source to provide the same content in each of the desired media representations. Each receiver's client software to access (i.e., decode and play back) multimedia content would have the ability to understand all of the media representations that could possibly be desired by any receiver. This solution, however, is not feasible in practice for several reasons. First, because of the rapid

[1] μ-law encoding is a form of logarithmic quantization used in telephone-quality coder/decoders (codecs).

proliferation of new and diverse information and entertainment appliances/devices, it may be practically impossible to outline the most common set of receiver characteristics that define the targeted receiver audience. As a result, and because a given content may be accessed by any number of receivers, a complete set of desired media representations for the content may not be known *a priori*. Second, even if the targeted receiver audience were to be determined in advance, it would be prohibitively expensive for a content provider to publish content in a large set of media representations. Finally, supporting every media representation in the client software would impose significantly higher resource (especially memory and/or disk) requirements on the receivers, hence increasing their cost. Moreover, this makes future representation of content in new media formats difficult if not impossible. All of these problems can be addressed by enabling the network to provide rich "content connectivity" between sources and receivers.

In this paper, we present an evolutionary framework for provisioning enhanced multimedia distribution-of-content services to a group of receivers. The framework addresses various degrees of heterogeneity among "distribution paths" from a common content source to diverse receivers, where a distribution path carries information from a content-aware multicast distribution tree between the source and one or more receivers. The goal of the framework, given the broadcast of some multimedia content, is to generate one or more distribution paths that are compatible with various receivers as they join the broadcast/multicast. The framework enables the realization of a multimedia overlay network that provides sophisticated content-based programmability over the provisioning, management, and distribution of media flows from content sources to receivers.

Toward this end, we utilize a number of evolutionary measures. First, to address receiver heterogeneity, we introduce awareness of receiver and link capabilities into the network. Second, we decouple media content from its representation and provision individual receivers with the best feasible representation that suits the receiver and the network. Third, we introduce intelligence into the network that allows it to exploit and adapt to request patterns across receivers (and groups of receivers). The above techniques support a feasible performance spectrum capable of supporting heterogeneity with regard to receiver capabilities and preferences, and performance tradeoffs among error resiliency, network bandwidth, and congestion.

The overlay requires one or more of each of the following four basic types of multimedia-aware nodes: sources, receivers, distributors, and controllers. The delivery of multimedia content originates at a source and terminates at a receiver connected by a traversal path (referred to as the distribution path) over one or more distributors. The overlay performs arbitration roles between sources and receivers, such as supplying content in a format suited to the performance capabilities of individual receivers. The construction and subsequent extension of content-distribution trees is determined by the types and numbers of receivers subscribing to a given content and representation.

To decouple content from representation—a key feature of our framework—the overlay supports content-based mechanisms to reconfigure distribution paths so as to address a wide range of receiver heterogeneity. This allows the overlay to provision content to a particular receiver by "transforming" such content from an ill-suited representation to a better-suited one. Given a core content distribution tree, the overlay constitutes a programmable network capable of autonomously generating branches from the core distribution tree. The branching criteria are based on cost criteria such as receiver capabilities (e.g., bandwidth, CPU processing power) and link capabilities (e.g., loss rate, error rate).

Our work builds upon and extends the notion of overlay concepts [10, 11] and application-level framing (ALF) [12]. In contrast to Clark's original ALF discussion, the notion of a distribution path is used to expose interior, application-level control points during the relay of a media flow as opposed to just at its ends.

The rest of the paper is structured as follows. First, we outline the goals and requirements necessary for evolving toward a next generation of programmable multimedia network services. Then, in light of these principles, we present an evolutionary perspective on high-function network elements and services. We then describe the key aspects of our framework for programmable multimedia overlay networks that satisfies the goals and requirements mentioned above. Section 5 discusses a critical component of our framework, namely, management of such overlay networks. In Section 6, we briefly describe a number of existing Internet-related technologies and necessary enhancements to them for effective realization of the proposed framework. Section 7 concludes the paper.

2. Goals and requirements

In this section we outline the key goals and requirements which must be satisfied by any framework in order to realize autonomous and polymorphic distribution of multimedia content. As argued earlier, a single representation cannot efficiently address the content requirements of truly diverse (i.e., heterogeneous) receivers. Managing diverse media representations, however, is a significant burden for designers and developers of multimedia content, applications, and tools. Receiver diversity can be transparently accommodated by ensuring independence of a particular content from its

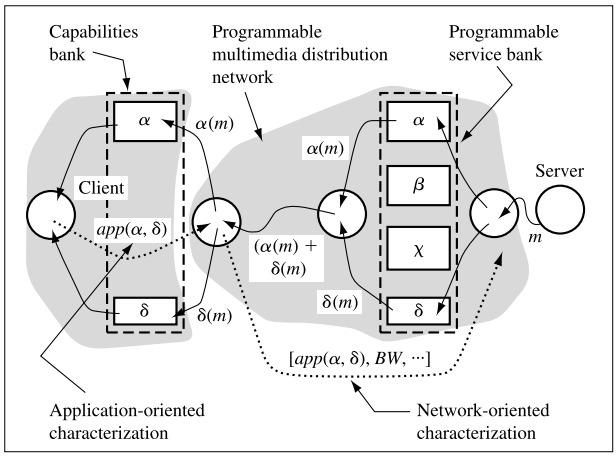


Figure 2

Generic content-adaptive client and network model. Here m denotes multimedia content. Inside the programmable multimedia distribution network, α, β, χ, and δ represent content transforms that operate on m to produce representations $\alpha(m), \cdots, \delta(m)$, respectively. In the client α and β represent content transforms capable of processing the corresponding representations of m. To determine suitability and extract an appropriate representation, the receiver characterization [denoted by *app*()] is considered by the overlay.

media representation (referred to as goal G1). Moreover, the distribution of content to receivers must be efficient and controllable (referred to as goal G2). These two goals impose two key requirements that the network must satisfy: promotion of content awareness and deployment of network services that leverage such awareness. We elaborate further on these requirements below.

Promotion of content awareness **Figure 2** illustrates the key features required of multimedia applications and the network to meet our dual goals of media independence and efficient distribution of content to heterogeneous receivers. Our approach introduces a new level of receiver-adaptive responsibilities into the network. Each receiver (i.e., client or application) is characterized by a capabilities bank, which represents the particular needs and preferences of a receiver with regard to any multimedia content. Each receiver exports a characterization of its capability bank to the overlay, which functions as a programmable multimedia distribution network. The overlay augments a particular receiver's characterization (including its processing power) with a network-oriented characterization of the receiver's attached link to the network (e.g., available bandwidth, loss rate, and error rate).

Deployment of network capabilities The overlay also contains an encompassing services bank of (continuous media) stream-oriented capabilities. These multimedia services are distributed throughout the overlay, and their form and function are programmable (i.e., controllable) from within the overlay. To address G1, the overlay may deploy a variety of media capabilities to achieve media polymorphism (e.g., transcoders or enhancers)[2] over selected overlay nodes. To address G2, these capabilities must be autonomously deployed on the basis of some cost criteria that best match the needs and preferences of receivers. Given a core content distribution tree, the overlay constitutes a programmable network capable of autonomously generating branches from the core distribution tree. The branching criteria incorporate both receiver capabilities (e.g., bandwidth, CPU processing power) and link capabilities (e.g., loss rate, error rate). The decision to branch off is weighted by the number of such receivers, bringing similar cost weights to the core distribution tree. Moreover, when the cost weight is substantially large, alternative representations of the broadcast are considered for distribution over the newly created branch.

By satisfying the above requirements, our framework defines a distribution network capable of "intelligently" adapting the distribution and representation of multimedia content into one that is suitable to the needs, preferences, and capabilities of its receivers. **Figure 3** uses the specific scenario of a multicast of speech content to illustrate, via an example, the sort of intelligence that can be provided on a single programmable network element (referred to as the distributor in Figure 3) within our framework. An input stream of PCM-encoded speech [13] must be recoded onto another format for some group of clients.[3] In particular, an on-line speech-to-text [14] capability could be used as an adaptation measure by the network to match the operational space of one or more receivers, as well as a multimedia service to provide a value-added text-captioning enhancement. Similarly, a text-to-speech capability (i.e., increasing the streaming requirements) could be used by the network as a media service to suit the preferences of a particular set of receivers.

Next, we present an evolutionary perspective to motivate and justify the continuing development of enhanced network elements and services, and distinguish our framework from other approaches. Subsequently we describe in detail the key features of our framework.

[2] A transcoder takes as input content in representation α and recodes such content to representation β. An enhancer takes as input content in representation α and outputs representations ($\alpha + \beta$).
[3] ITU-T Recommendation G.711 is used to compress, expand, or convert digitized PCM data among μ-law, A-law, and linear formats.

3. An evolutionary perspective

Since the arrival of the World Wide Web (WWW) [15], there has been a gradual but steady trend toward provisioning *high-function* (i.e., intelligent) network elements and increasing the functionality provided by these elements to coordinate end-to-end data transfer between a network (e.g., Web) client and server. This evolution of high-function network elements and their associated services has been driven primarily by two aspects, both relating to the way in which users perceive and utilize network services:

- The performance of content delivery as experienced by clients.
- The diversity of client devices (and their connectivity to the Internet and content servers).

It is argued that future evolution of network services will be driven by, in addition to the above aspects, the ability of network elements to provide enhanced multimedia services to any client anywhere [16]. Future network elements must be capable of transparently accommodating and adjusting to client *and* content heterogeneity.

- *Terminology*

For the ensuing discussion to be comprehensible, we first outline the terminology we adopt in describing the evolution of high-function network services. We refer to a source of network content as the server and a sink of network content as the client. A client requests delivery of content from a server, and in response content flows from the server to the client through the network. When traversing the network, the content may flow through one or more intermediary network elements that together constitute the network fabric connecting the clients and servers. Intermediaries can either be routers that relay (i.e., forward) network data onto one or more attached networks, or proxies that may perform more intelligent forwarding and transformations on the network data.

A number of clients may be communicating simultaneously with a number of servers over multiple unicast [17, 18] or multicast [19, 20] sessions. While a unicast session coordinates content transfer between a single client and server, a multicast session coordinates content transfer between a server and a set of clients wanting to receive the same content. Multicast transfers are much more efficient (in terms of server processing load and network bandwidth) than unicast transfers, if the same content is to be delivered from a server to one or more clients.

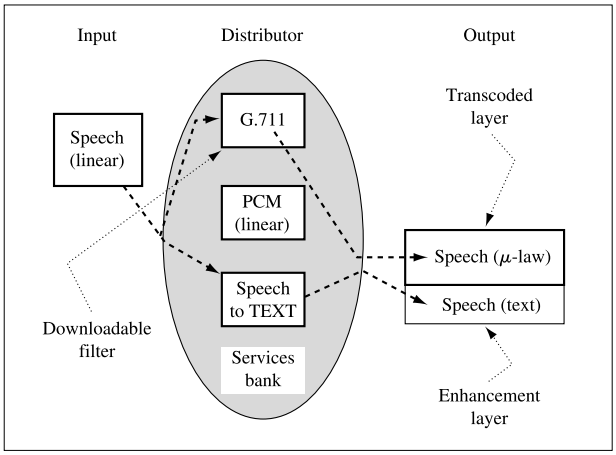


Figure 3

Dynamic creation of content and network services via deployment of network-resident capabilities. Network-resident services (i.e., capabilities) such as real-time G.711/PCM conversion capabilities could be deployed to address the "impedance matching" across heterogeneous receivers. Similarly, real-time speech-to-text capabilities could be used both for retargeting bandwidth requirements for heterogeneous receivers (a network-oriented service) and for providing text-captioning enhancements to a broadcast (a content-oriented service).

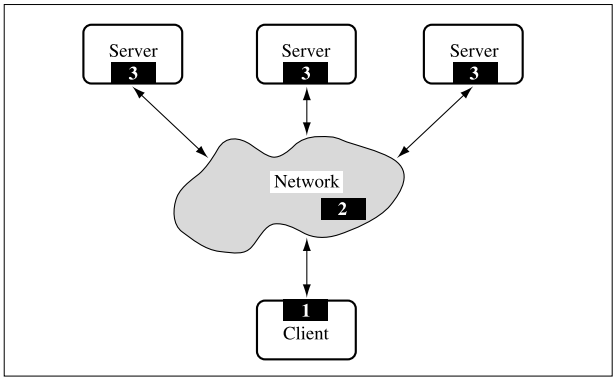


Figure 4

Placement of intelligence about network services: location 1 at the client, location 2 within the network, and location 3 at a server.

- *Location and nature of intelligence*

Figure 4 illustrates the alternative locations in which intelligence about available network services may be placed. From a client's perspective (location 1), this intelligence typically constitutes awareness of the services provided by different (possibly replicated) network servers. It may also include estimates of the end-to-end

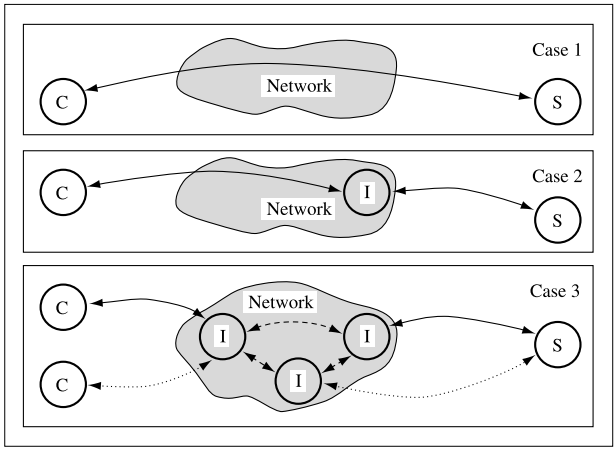


Figure 5

Variations in client–server connectivity: Client (C), Server (S), Intermediary (I).

performance the client can expect when communicating with functionally equivalent servers. Since clients and servers can be heterogeneous, this intelligence may also include knowledge of the functional disparities (in terms of supported protocols and content types) between the servers. The client can utilize this knowledge to select the most appropriate server and mechanism in order to obtain the available content.

As an alternative, this knowledge (and the associated burden) could be entirely or partially transferred to the individual servers (location 3), or could reside inside the network (location 2). In the rest of this section, we present the evolution of high-function network elements and services (i.e., placement of intelligence in location 2 in Figure 4). **Figure 5** succinctly illustrates the variations in client–server connectivity capturing this evolution. We highlight key shortcomings in supporting continuous media applications and accommodating content and client heterogeneity. We consider the following cases: no intermediary, single intermediary, multiple intermediaries, and intelligent intermediaries.

- *No intermediary*

This represents the simplest form of Internet client–server communication, characterized by a direct (logical) connection between the client and server, i.e., with no intermediary proxies (case 1 in Figure 5). Data must still flow through the set of network routers that provide the connectivity between the client and the server. There are a number of advantages of this simple configuration. No extra functionality is required in the network elements to provide connectivity between the client and server. Moreover, the client may utilize local intelligence to select one from a set of (replicated) servers with which to communicate. As mentioned above, this intelligence is usually based on the measured end-to-end performance (latency and bandwidth) between the client and the servers.

However, despite the aforementioned advantages, this configuration suffers from a number of significant drawbacks. First, it requires direct connectivity between the client and server; such connectivity is at times not possible, and either may be explicitly forbidden for security reasons or may result in poor end-to-end performance. Often the server with which a client desires to communicate resides inside the administrative domain of a company. To maintain a high degree of security, the company may not allow direct access to the server's content. In fact, the server may not even be on a network that is directly connected to the outside world, i.e., the Internet. Even with direct connectivity, performance (and hence scalability) can suffer because each client must communicate with the server for content access. This results in higher server load and wastes network bandwidth, because there is no provision for content reuse among different clients. In another situation, mobile users communicating directly with one another may experience poor end-to-end network performance because of highly lossy and unreliable wireless links. Second, the client and server must agree on the type of content delivered; i.e., the client must be able to process/display the delivered content. The burden of handling client heterogeneity lies completely on the server, which must support the necessary intelligence and incur additional processing load. This burden is most severe for multicast sessions, which must be handled by the server as multiple unicast sessions. Finally, if the servers cannot support client heterogeneity, an unnecessary burden is placed on the clients to support multiple protocol and content standards and to communicate with the "most compatible" server.

- *Single intermediary*

Several of the above-mentioned drawbacks are partially mitigated by adding a single intermediary proxy between the client and server (case 2 in Figure 5). This proxy may reside at the edge of the network, either close to a set of clients or as a front end to a set of content servers. The main role played by the proxy is to forward network data between the client and the server while often performing limited well-defined operations over such data. There are three primary benefits of using such an intermediary proxy: improved connectivity and security semantics, support for content customization, and higher performance.

Connectivity and security semantics are improved because the proxy can act as a bridge between the client

and server, providing support for indirect, authenticated, and restricted access to content servers (e.g., acting as a firewall). The presence of a proxy also provides the opportunity for content customization, i.e., application of useful protocol translation and content transcoding functions to compensate for the "impedance mismatch" resulting from protocol and client heterogeneity. An intermediary proxy can also result in higher performance (e.g., by better supporting client mobility and multicast sessions). The proxy can act as a base station connecting a mobile client to the Internet, improving performance by locally adapting to the characteristics of the wireless link to/from the client [21]. The proxy can also perform content-aware transformations such as adapting the delivery of multimedia content over wireless links via media scaling [22, 23]. Similarly, the proxy can provide some support for managing multicast sessions, freeing the server from this burden and improving overall performance.

However, a single intermediary does not provide sufficient support for large-scale high-function multimedia transfers between heterogeneous clients. Significantly greater functionality is needed to support a diverse set of networked multimedia applications. For example, while the proxy may perform limited transcoding functions, there is no support for media independence and hence for content heterogeneity. The functionality supported is typically *ad hoc*, and not sufficiently intelligent and programmable. Further, a single proxy cannot naturally capture the distributed and localized nature of multicast communication. More significantly, a single proxy provides poor scalability because of location and capacity constraints. That is, a single proxy cannot exploit the geographic locality naturally exhibited by clients, and it is unlikely to have sufficient capacity to manage and compensate for client and content heterogeneity.

- *Multiple intermediaries*

It is likely that there will be more than one intermediary proxy in the path between a client and a server (case 3 in Figure 5). Each proxy is typically dedicated to a specific function, such as a firewall or a base station, supporting a limited role based primarily on access control and physical location requirements. While there are numerous advantages of specialization of individual proxies, such a scenario results in multiple proxies with scattered, *uncoordinated* functionality.

Many multimedia applications involving large-scale content distribution naturally require and benefit from support for multicast transfers. In recent years significant attention has been paid to network and protocol support for reliable multicast sessions [24, 25]. These efforts resulted in enhanced and explicit support for scalability and reliability in multicast communication. This is achieved via explicit support for IP multicast groups, appropriate end-to-end protocol support, multiple distributed multicast routers, and local repair capabilities at designated routers [24]. While multiple network elements work together to efficiently carry multicast traffic, no support is provided for the specific requirements of heterogeneous continuous media traffic.

Media awareness and client heterogeneity must be accommodated at the servers and clients via support for multiple media types and intelligent adaptation. For example, client heterogeneity may be supported via multilayered hierarchically coded media streams generated by the server; a receiving client could subscribe to one of these streams on the basis of its capabilities [26]. However, we believe that such approaches are insufficient to accommodate client and content heterogeneity in the scalable transfer of continuous media. Forcing media awareness and adaptation only at the endpoints places an undue burden on clients and servers. Furthermore, this makes it extremely difficult for them to support a variety of media types and representations, not all of which may have well-defined layered representations. Explicit support for media independence and programmability within the network promises to eliminate this burden while providing seamless integration of heterogeneous clients and content.

- *Intelligent intermediaries*

Recently there has been an explosion of interest in intelligent intermediaries within the network, as evidenced by various research efforts in active networks [10, 11, 27]. Various forms of intelligent intermediaries have been proposed. For the WWW, one or more proxy caches may serve as intelligent intermediaries, improving content delivery performance significantly by caching popular Web content [28]. These proxies can be placed and configured to exploit geographic locality and client access patterns to reduce network server load and thus improve scalability. In other forms, proxies can act as intelligent front ends [29] performing load balancing at a server farm. For mostly static multimedia data, special proxies may perform advanced content translation and distillation functions to partially support content and client heterogeneity [30]. Advanced proxies may also have the capability to perform load balancing in conjunction with content awareness and affinity [31]. Specific support for continuous media filters has also been proposed for (programmable) heterogeneous networking [32–34].

The active network proposals to date target network level programmability without being content-aware, in contrast to our proposal, which targets content-aware application-level programmability. Moreover, while supporting varying degrees of media awareness, the aforementioned approaches either apply only to static content (i.e., do not extend directly to continuous media)

or are media- and content-specific. We argue that the explosion of media types necessitates a network infrastructure that frees the clients and servers from such media dependency and the burden of managing content and client heterogeneity. This is particularly critical for continuous media because of its demanding resource requirements for processing, translation, and transmission.

That is, the next-generation network infrastructure must combine media awareness with a high degree of intelligent adaptivity in order to achieve true media independence and serve heterogeneous clients. In the next section, we present our framework for a scalable network infrastructure realizing media independence via programmable network intelligence.

4. Framework

In this section, we take a detailed look at key issues on the realization of G1 via a group of content-aware intermediaries which we call the "overlay." The G1 group decouples multimedia sources from receivers. Whereas G1 explores the mechanisms to provide media independence to heterogeneous receivers, the G2 goal is to realize efficient source-to-sink traversals through the group. To this end, in the next section, we show the realization of G2 via the ability to program this G1 group to react to network and application requirements. The resulting $G2|_{G1}$ group of intermediaries is better described as a programmable overlay for the relay of multimedia content. Next, we proceed with the specification of such overlay groups, which are hereafter called "clouds" for distinction in this discussion.

Nodes The cloud consists of four new types of high-function network elements: receivers, sources, distributors, and controllers. The cloud opens to users at well-defined points referred to as exterior nodes. Multimedia receiver nodes are exterior nodes that open the cloud to subscribers (S), whereas multimedia source nodes open the cloud to providers (P). A provider supplies content to the cloud, and a subscriber requests content from the cloud, leading naturally to a supply-and-demand model in which the cloud serves as an arbitrator for both the routing and brokering of content. The distribution of multimedia content originates at one or more sources and terminates at zero or more receivers. The distribution of multimedia content from a source to a receiver is accomplished by "relaying" such content across one or more of these interior points in the cloud. As argued earlier, multiple intermediaries are used as a way of breaking up the end-to-end distribution problem into k smaller end-to-end subproblems between interior points in the $G2|_{G1}$ cloud.

The flow of content across any two nodes inside the cloud is referred to as a media flow. The original notion of a media flow was introduced by Clark [12] to expose application-level framing at either end of an end-to-end network connection. In contrast, here a media flow represents a *first-class entity*[4] within the interior of the overlay, as opposed to just at its exterior endpoints. This notion is critical in highlighting the likelihood of exposing a media flow to network capabilities (i.e., multimedia services) at any of these interior points. These points are referred to as multimedia distribution nodes. Each distributor opens a service control point as well as a routing control point over the traversal of a media flow through the interior of the cloud.

To coordinate traversals across interior points, specialized nodes referred to as multimedia controllers implement the algorithmic functions necessary to orchestrate these distributed resources. Controllers provide a control entry point to the overlay, whereas sources and receivers realize the media entry points. There are important differences between the transport of media and control information. Whereas media connections support media flows, control connections relay the commands that manage these media flows. There are two basic approaches to implementing their transport:

1. Media information and control data are relayed over a common channel.
2. A separate channel is used to relay control data.

The first approach could be implemented by tagging and interleaving command capsules between media units. The second approach could be implemented by downloading such capsules over a separate control channel. However, the first approach is weaker because it ignores differences between the relay of media and control, requires the disruption of encoding formats, and introduces significant overhead at each node. Moreover, whereas the lifespan of a media connection is tied to that of its underlying media flow, the lifespan of a control connection is instead tied to that of its nodes. Finally, media flows are QoS-oriented, but control flows are priority-oriented. The first approach lacks the ability to prioritize the delivery of control commands, whereas the second approach provides both a QoS media channel and a priority delivery channel for commands as out-of-band data.

The components of our programmable $G2|_{G1}$ cloud are illustrated in **Figure 6**. Throughout this paper, let c_i be a controller, s_i a source, d_i a distributor, and r_i a receiver; let n_i represent any node $\{s_i, d_i, r_i\}$ in the $G2|_{G1}$ cloud. **Figure 7** illustrates some relationships among these components. On one hand, Figure 7(a) depicts a multimedia source (s_1) sending a media flow to a

[4] The notion is used in an object-oriented sense, and is intended to mean that the media flow itself is modeled as having behavior, state, and properties.

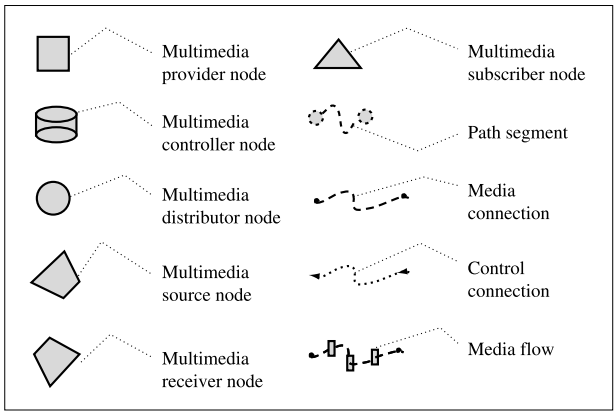


Figure 6

Basic compositional elements of the framework: multimedia nodes, path segments, media connections, control connections, and media flows. Media connections are denoted by heavy dashed lines, control connections by dotted lines. Content flows through the media connections, and control connections are used to relay management commands between nodes.

distributor (d_2). On the other hand, Figure 7(b) shows a view of the interior of the cloud, a controller (*MCN*) with control connections to the distributors (d_1, d_3, d_5, d_7) it manages. Finally, Figure 7(c) depicts a multimedia receiver (r_1) with an incoming media flow from a distributor (d_5).

Distribution paths The connection of a source to a receiver through distributors realizes an end-to-end traversal path through the cloud, referred to as a distribution path. A distribution path is said to *emit* a media representation. The distribution path is used for the relay of content m from P to S; for example, **Figure 8(a)** shows the distribution path $p[P_1, S_2, m]$. The distribution path $p[P, S, m]$ is a content-aware distributed object that describes a controllable "walk" over the overlay. These walks over the overlay typically start at some source s and terminate at some receiver r while traversing one or more distributors d_i. For example, **Figure 8(b)** illustrates the traversal of the distribution path $p[P_1, S_2, m]$ between a source and a receiver over three interior points (d_1, d_3, d_5) in the cloud. While on one hand the distribution path represents an exterior end-to-end problem, on the other hand, each segment of the distribution path also represents an interior end-to-end subproblem.

Path segments Content is relayed between any two nodes in a distribution path via a content-aware media connection referred to as a path segment. A path segment

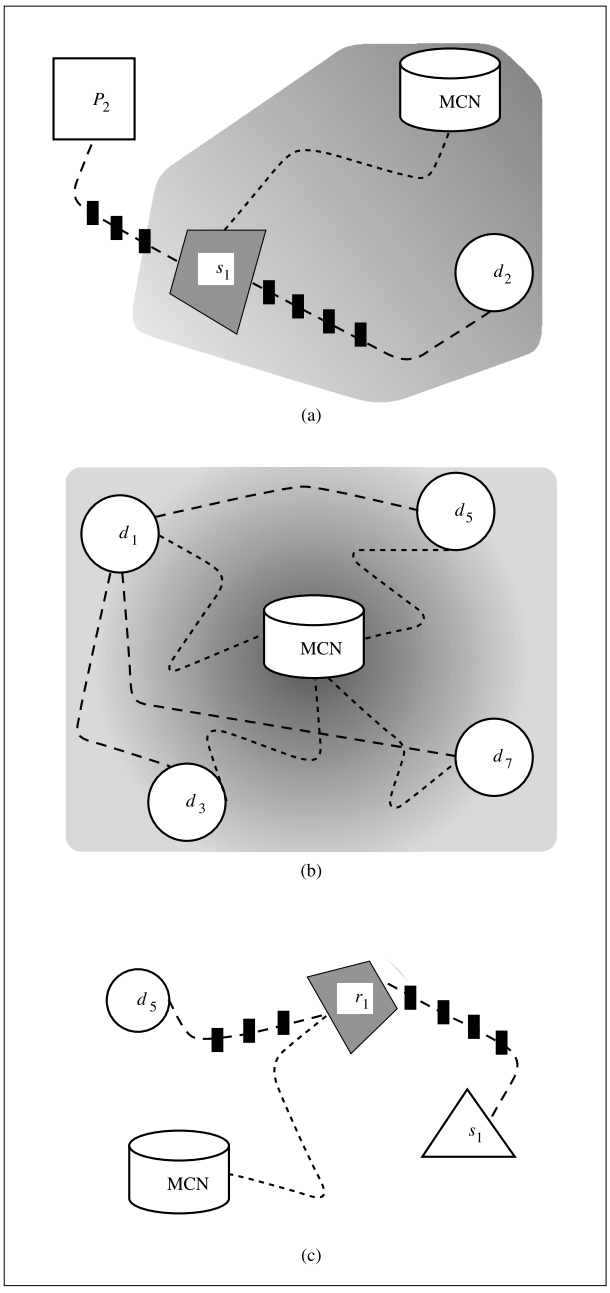


Figure 7

Sources, controllers, distributors, and receivers. (a) Sources interface with multimedia control nodes (MCN) and distributors. To interface with the overlay, a content provider relays content (represented by the small packets) to a source in the overlay. Within the overlay, media connections represent path segments with lifespan associated with that of a session (e.g., between s_1 and d_2). (b) Distributors allow the deployment of capabilities throughout the overlay. Distributors interface with other (zero or more) distributors via media connections and with one or more controllers via control connections. (c) Receivers open the overlay to subscribers. Receivers interface with one or more controllers and distributors.

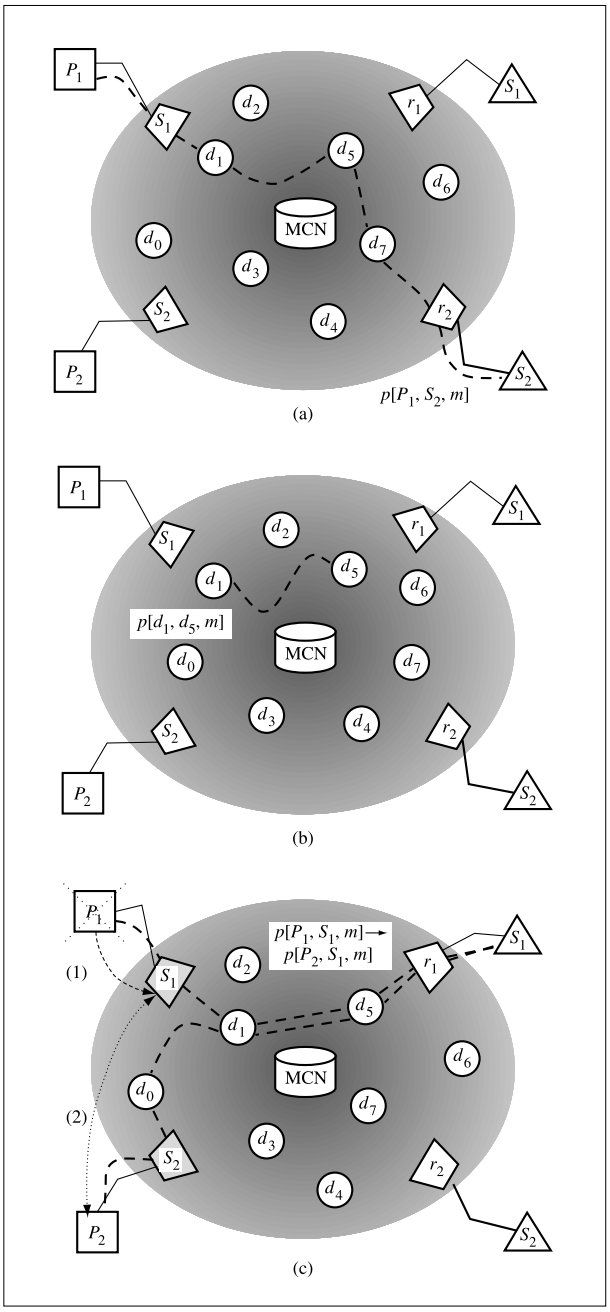


Figure 8

A distribution path (a) is used to control the distribution of content m over a packet-switched network (e.g., see $p[P_1, S_2, m]$) via the relay of content from sources to receivers through an overlay of distributors. A path segment (b) is used to influence routing during the distribution of content m over the PSN (e.g., see $p[d_1, d_5, m]$). Distributors are used as application-level intermediaries as well as routing control points. Distribution paths may be dynamically updated on the basis of network or receiver conditions. Path segments enable dynamic rerouting of distribution paths over the distributor overlay and provide the reconfiguration granularity (c) used to branch off a distribution path.

$p[n_1, n_2, m]$ is a content-aware distributed object between two nodes $n_1 \& n_2$ in the G1 cloud that is said to emit a representation [e.g., $\alpha(m)$] of a media flow associated with some content m. This is represented as

$$p[n_1, n_2, m] \rightarrow \alpha(m), \tag{1}$$

where $\alpha(m)$ is one possible content representation [e.g., $\alpha(m)$, $\beta(m)$, $\cdots$] for content m. A distribution path is composed of one or more path segments [for example, Figure 8(a) shows the distribution path $p[P_1, S_2, m]$, and Figure 8(b) shows one of the constituent path segments (i.e., $p[d_1, d_5, m]$) over which the distribution path $p[P_1, S_2, m]$ is realized]. Path segments enable dynamic rerouting of distribution paths over the distributor overlay and provide the reconfiguration granularity used to branch off (i.e., reuse) a distribution path. For example, **Figure 8(c)** shows reuse and reconfiguration of path segments.

Next, we formally define the relationships among nodes, distribution paths, and path segments.

Object management services A distribution path can be represented as an ordered tuple that lists nodes on a traversal ordered from provider to subscriber. Equivalently,

$$p[P, S, m] = (P, s, \{d\}^+, r, S). \tag{2}$$

Now, we refer to the predecessor $\text{pre}(n_i)$ of a node $n_i \in p[P, S, m]$ as simply n_{i-1} and to its successor $\text{suc}(n_i)$ as simply n_{i+1}. For example, if $p[P, S, m]$ is a distribution path, then $p[n_i, n_{i+1}, m]$ is its ith underlying path segment. For convenience, we define the term "core distribution tree," $CDT'(m)$, as the set of distributors in a distribution path (i.e., $\{d\}^+$).

A path segment $p[n_i, n_{i+1}, m]$ is controlled by its spanning node n_i. To reflect this asymmetry, we classify path segments as incoming or outgoing. At node n_i, incoming path segment $\text{in}(pid, n_i)$ of node n_i in a distribution path is denoted by $p[\text{pre}(n_i), n_i, pid]$. Similarly, the outgoing segment $\text{out}(pid, n_i)$ of node n_i in a distribution path pid is denoted by $p[n_i, \text{suc}(n_i), pid]$. The path identifier pid is used to "tie in" (within a node) incoming segments to outgoing segments by associating segment identifiers sharing the same path identifiers.[5]

Object directory services (e.g., object request brokers [35]) can be used to track these distributed objects, nodes, and their relationships. On one hand, to track a distribution path $p[P, S, m]$, the controller creates a globally unique identifier $pid(p[P, S, m])$ to each distribution path. This identifier is referred to as the path identifier. An object proxy [35, 36] of the path identifier is made available to every node on its corresponding

[5] It is possible that one or more (e.g., m) incoming segments map to one or more (e.g., n) outgoing segments for a given path identifier pid. In this case, the node n_i acts as an $m \cdot n$ concentrator/multiplexor for content m.

distribution path. On the other hand, to track a path segment, each node in a distribution path associates a unique identifier $sid(p[n_i, n_{i+1}, m])$ with each outgoing path segment. This identifier is referred to as the segment identifier. The segment identifier is a distributed object known only by nodes n_i and n_{i+1} at either side of the corresponding path segment.

Path and segment identifiers allow the representation of the relationships between path segments and distribution paths as follows. First, a distribution path can be composed of one or more path segments. To represent this relationship, it is necessary to determine, given a path identifier, the segment identifiers associated with it. To this end, each path segment in a distribution path is associated with the same path identifier. Second, a path segment can be associated with zero or more distribution paths. To represent this relationship at a node, it is necessary to determine, given a segment identifier, all path identifiers associated with it. At the controller, given a path identifier, it is necessary to determine all nodes found along the corresponding distribution path. This way, only nodes in a distribution path must be tracked by the controller.

Capabilities The path segment realizes a content-aware connection used to expose a media flow to application-level processing at either of its ends. As content is relayed through the cloud, distributors are capable of exposing an incoming media flow to application-level capabilities (e.g., multimedia services). A capability is defined as an application-level filter $f(\,)$ that takes as input a media flow having content representation $\alpha(m)$ and outputs a media flow in content representation $\beta(m)$. Equivalently,

$$f[\alpha(m), t] = [\beta(m), t - t_d], \tag{3}$$

where t_d is an upper bound on the aforementioned application-level processing overheads. Examples of capabilities are encryption of a media flow, multiplexing of multiple incoming media flows into a single outgoing media flow, content-based filtering, content translation, etc. Capabilities such as adaptive filtering can be used to adapt the incoming media flow to the requirements of outgoing path segment(s) [22, 23, 37].

Assuming that it is possible to provide on demand any capability at any distributor, the concatenation of path segments (i.e., the cascading of distributor capabilities) appears unnecessary. This fails to take into account, however, that the path segment abstraction is used not only to satisfy G1 but to satisfy G2 as well. On one hand, because resources are finite, cascading capabilities across distributors are needed to span distribution paths capable of emitting representations that would otherwise not be possible because of resource allocation constraints. On the other hand, as discussed later, each distributor also serves as a control point over the routing of a media flow. Consequently, the cascading of capabilities provides the means to introduce multiple route control points over the routing of a media flow over the overlay. Moreover, the cascading of capabilities increases the opportunity of reusing path segments.

5. Management of the overlay

In this section, we explore three important issues on the management of the overlay: global management of capabilities, end-to-end management of distribution paths, and subscription management.

- *Capability management*

Clearly, a management authority is needed to manage capabilities throughout the overlay. To facilitate the discussion, controllers are entrusted with the management of (subsets of) distributors and their capabilities.[6] In terms of our algorithmic needs, distributors can be represented in terms of a combination of application and network state. To support capability management, it is necessary to track each distributor in terms of its currently loaded capabilities (C_i) and their respective capacities ($|C_i|$). Moreover, to support application-level routing across distributors, it is necessary to differentiate among distributors, for example, in terms of their total capacity $T_{\text{allot}}(d)$ and their utilization state $U(d)$. To this end, a distributor d having k capabilities and total capacity T_{allot} is modeled as a tuple of the form

$$(T_{\text{allot}}, \{C_i, |C_i|\}^k) \qquad \text{where } T_{\text{allot}}(d) = \sum_{i=0}^{k} |C_i|. \tag{4}$$

Route control versus service insertion points Will more than one distributor be needed for the purposes of transforming content into a suitable representation? On one hand, because resources are finite, the cascading of capabilities across distributors allows the generation of representations that will otherwise not be available. Furthermore, each distributor serves as an explicit control point over the routing of a media flow. Moreover, multiple distributors can be used as application-level relays (routers) by routing a media flow via a "pass-through" capability at each such distributor. Each such distributor increases the effectiveness of application-level routing over the overlay, since each introduces an additional control point for the routing of media flows through the underlying packet-switched networks. However, each distributor introduces significant application-level overhead to the end-to-end delay between sources and

[6] This does not imply that the capability management authority is centralized nor that distributors lose their autonomy to such authority. It is best to think of such authority as a recommendation system.

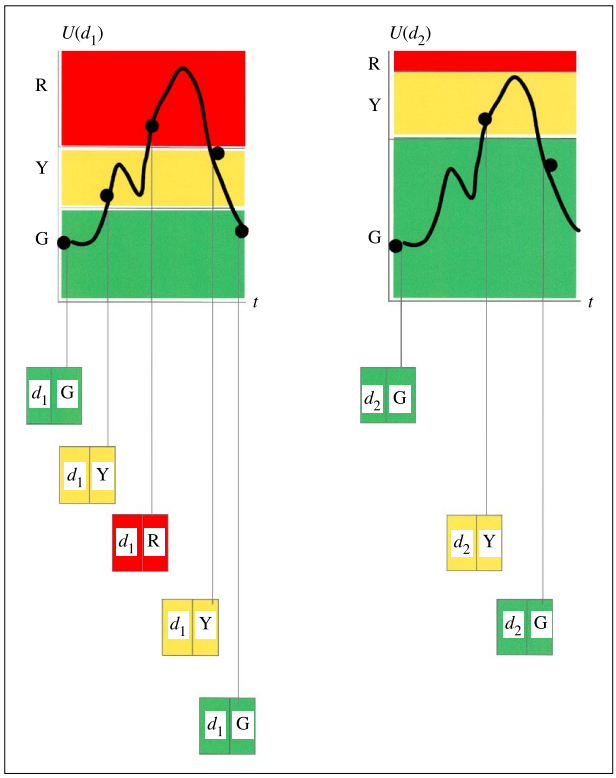


Figure 9

The controller must track the utilization state $U(d)$ of its distributors to create and route path segments. The figure shows plots of the overall utilization $U(d)[t]$ over time t for two distributors. A watermark scheme is used to relax the need for measurement reports as well as to handle heterogeneity among distributors. The measurement reports sent by each distributor to its controller are shown for each case. Whereas, on the left, distributor d_1 relied on conservative watermarks, distributor d_2 on the right relies on an aggressive green zone.

Table 1 Integrated routing lookup table. The controller maintains the state of network conditions between distributors and of application-level processing costs within each distributor. Unlike traditional network elements, distributors introduce content processing overheads that must be accounted for during the routing of path segments and distribution paths.

$\alpha(m)$	n_1	n_2	n_3
n_1	(0, 1)	$rtt(n_1, n_2)$, c_{12}	$rtt(n_1, n_3)$, c_{13}
n_2	$rtt(n_2, n_1)$, c_{21}	(0, 1)	$rtt(n_2, n_3)$, c_{23}
n_3	$rtt(n_3, n_1)$, c_{31}	$rtt(n_3, n_2)$, c_{32}	(0, 1)

receivers (i.e., when compared to a traditional network router). Clearly, a tradeoff exists between the number of distributors used for routing control and the cumulative application-level overhead (i.e., $\Sigma t_d \ \forall \ n_i \in p[P, S, m]$) introduced by such distributors.

The overlay also opens new opportunities in the integration of route and service control. Multiple distribution paths sharing a common link may benefit from network (as opposed to client-based) integration. Our approach differs significantly from that of a low-level packet flow multiplexor in that our multimedia concentrator capability creates the opportunity to introduce awareness of receiver/subscriber capabilities. For example, the concentrator could rely on an earliest-deadline-first scheduler to remove media frames that end up being simply overhead to a particular class of subscribers. This approach results in several benefits. First, it shifts media integration from the subscriber to the overlay, thus easing the tasks of "low-CPU-powered" subscribers. Second, it induces common routing behavior for multiplexed media flows that, after all, are to be provisioned to the same receiver. On the down side, the concentrator capability reduces the ability of the overlay to provision under scarce available bandwidth. A concentrator imposes a harder constraint over admission control—finding k small holes across k links is easier than to find one large enough on one link.

In a more subtle point, the capabilities available on a distributor also provide the overlay with a way of indirectly biasing the traffic on the underlying packet-switched networks (heretofore referred to as the path inducement effect of the overlay). For example, if a distributor were exclusively allocated to a specific capability (for example, a flow decryption distributor), such a distributor would then be capable of a significant path inducement effect on the routing of most media flows requiring such services. The routing of such media flows would likely consider and/or reuse path segments through this distributor. On the other hand, if the distributor were to have too many capabilities, its path inducement effect would be decreased. Thus, choosing capabilities for a distributor is a difficult task. For this reason, means to dynamically reconfigure capabilities on a distributor become desirable.

Dynamic reconfiguration Because we may not know in advance which capabilities are most needed, the ability to dynamically reconfigure capabilities in a distributor is desirable. In particular, we would like to

1. Monitor and forecast capability usage.
2. Pre-load critical capabilities into distributors.
3. Dynamically deploy the remaining capabilities throughout the overlay.

Our approach is to pre-load popular capabilities into distributors and pre-allocate spare capabilities *and*

Table 2 Capability lookup table. To manage path segments and content representations $[\alpha(m), \cdots]$, the controller must track available capabilities $f[\alpha(m)$ to $\beta(m)]$ within the overlay, in addition to their requirements $b[\alpha(m)]$, location (e.g., n_j), and utilization state (shown in Figure 10). Note that in this example not all capabilities are found at every node.

m	$\alpha(m)$	$\beta(m)$	$\gamma(m)$
$\alpha(m)$	—	—	$f(\gamma \rightarrow \alpha)@b(\alpha)@(n_1, n_4)$
$\beta(m)$	$f(\alpha \rightarrow \beta)@b(\beta)@(n_2)$	—	—
$\gamma(m)$	—	$f(\beta \rightarrow \gamma)@b(\gamma)@(n_3, n_4)$	—

capacities through the overlay so as to make room for dynamic downloading of less popular capabilities. Spare and unused capacities provide controllers with a highly available resource pool (as a *swap space* for the overlay) that would be used, for example, to swap less popular capabilities for more popular ones. Similarly, the capacity of a capability would be dynamically reconfigured, for example on the basis of demand for a multimedia service. On the basis of overlay performance (e.g., the response of the overlay to demand), a controller could then recommend that a particular distributor update its capability banks and, if necessary, download new capabilities. One approach to implement the downloading of capabilities (i.e., software) across heterogeneous distributors would be via the use of platform-independent program capsules (e.g., Java** [38] program capsules). In contrast to other proposed active networks such as Mobiware [33], our mechanism targets the aggregate long-term performance of the overlay—capabilities are downloaded as needed in response to the performance of the overlay as opposed to the needs of a particular media flow (i.e., steady vs. transient response). On one hand, our approach enables the overlay to deploy and optimize capabilities based on criteria such as critical mass. On the other hand, our approach increases the response time of the overlay to adaptation of per-flow needs.

Measurements approach To branch new path segments from a given distributor, the controller determines whether it is possible to generate a particular representation (i.e., as needed for service provisioning) at a distributor (i.e., as needed by routing control). For this reason, the capacity and availability of resources at each distributor are tracked by the controller. It is important, however, that the overhead of such tracking be low. If the updates are too frequent, too large an overhead is incurred. On the other hand, if the updates are too infrequent, the controller could end up relying on an incorrect assessment of the state of its distributors when constructing distribution paths across the overlay. Elsewhere [39], we presented a robust statistical process-control framework to derive reliable long-term measurements among widely distributed resources. Our approach consists in relaxing the need for periodic measurements by relying on the use of a special kind of watermark based on well-known statistical process control principles [40]. Watermarks are used to remove heterogeneity across distributors and to reliably measure the utilization state $U(d)$ of a distributor. High and low watermarks are used to bound a critical region of capacity on the distributor (for example, (low, high) = [84%, 93%]). The region below the low watermark is referred to as the "green zone," the region between watermarks is referred to as the "yellow zone," and the region above the high watermark is referred to as the "red zone." Consequently, a distributor is said to be in a red, yellow, or green state by the controller in terms of its reported overall available capacity. The distributor needs only to report long-term changes around its watermarks. Moreover, the actual definition of a watermark needs to be known only by a distributor. This way, watermarks serve a dual functionality. **Figure 9** illustrates the use of this scheme across two distributors d_1 and d_2. The approach is summarized as follows.

1. When a distributor capability is green, the controller may freely allocate new paths to this capability.
2. When a capability at a distributor is yellow or red, the controller does not allocate the capability to new configuration requests.
3. A distributor considers new configuration requests only when green. Note that race conditions are possible. Specifically, a controller may believe a distributor to be green while the distributor is no longer green. In such case, depending on its own admission controls, a distributor may or may not accept such requests. The region between low and high watermarks provides resiliency for the delay between distributor reports and the generation of new paths.

Illustration To support configuration of capabilities across the overlay, controllers must also be capable of assessing the state of the overlay. In contrast to traditional routers, distributors require accounting for application-level costs during routing over the overlay. New ways are needed to represent and integrate application and network costs into the rerouting of distribution paths across the overlay. **Table 1** illustrates one approach for the

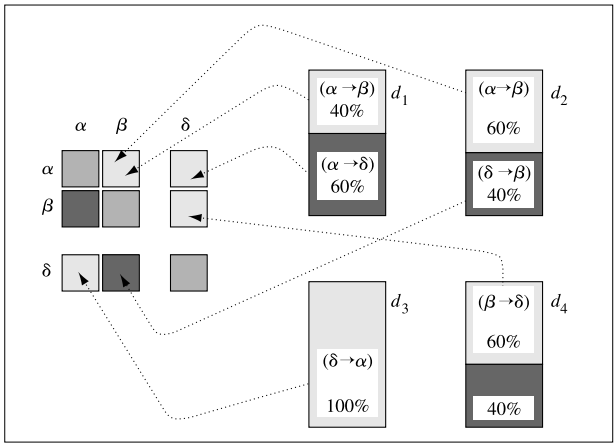


Figure 10

The controller must track the state of its distributors. At each distributor, each capability [such as $(\alpha \rightarrow \beta)$, $(\beta \rightarrow \delta)$] is associated with a capacity (e.g., 40%, 60%); in turn, each capacity is associated with a utilization state (green, yellow, and red) used to drive admission controls. At the controller, to avoid per-capability measurement overhead, the controller can rely solely on the overall state of a distributor $U(d)$ and negotiate admission on demand to any particular distributor capability.

integration and representation of application and network costs for content representation $\alpha(m)$. The (ith, jth) entry on the table contains the tuple $[rtt_{ij}, c_{ij}]$ which is associated with path segment $p[n_i, n_j, m] \rightarrow \alpha(m)$. Whereas rtt_{ij} equals the round-trip time between n_i and n_j, c_{ij} represents a weight factor associated with such a path segment. On one hand, rtt_{ij} represents the network cost of a path segment. On the other hand, c_{ij} represents application-level processing costs associated with such a path segment. Since either cost factor can be arbitrarily large, routing across such a path segment must consider both cost factors.

Table 2 illustrates, via an example, the tracking of routing information (for content m). The table enables the controller to manage content m (e.g., allocate, reuse, destroy, adapt) across the overlay. The (ith, jth) entry in this table contains the tuple $[f(m_1 \rightarrow m_2)@b(m_2)@(n_i \cdots n_j)]$. Whereas $f(m_1 \rightarrow m_2)$ represents a multimedia capability taking as input content in representation m_1 and transforming it to representation m_2, $b(m_2)$ represents the requirements of such a multimedia capability, and $(n_i \cdots n_j)$ represents the set or *group* of distributors possessing such a capability. Because membership on this set is dynamic, each tuple can be visualized as a process group provisioning the corresponding representation (e.g., m_2).

Figure 10 illustrates, via an example, the tracking of four different distributors by a controller. The example shows one particular content type m (e.g., audio) and three known representations $[\alpha(m), \beta(m), \gamma(m)]$ (e.g., PCM, H.323, and text). As stated earlier, the ability to provision a representation is referred to as a network capability (or multimedia service). In this example, the network capabilities $(\alpha \rightarrow \beta, \alpha \rightarrow \gamma, \cdots)$ are known to be deployed on the overlay, as shown in Figure 10. In this particular example, distributor d_1 is configured 40% for capability $(\alpha \rightarrow \beta)$, with its remaining 60% reserved for capability $(\alpha \rightarrow \gamma)$. Similarly, distributor d_2 is configured 60% for capability $(\alpha \rightarrow \beta)$, but the remaining 40% is reserved as spare capacity for on-demand deployment of a new capability. Finally, distributor d_3 is configured 100% as spare capacity reserved for on-demand deployment of new capabilities.

Figure 10 also highlights several important issues on the management of capabilities. First, the allotment of a distributor can be spread across to zero or more capabilities. For example, the d_3 allotment is fully dedicated to the capability $(\delta \rightarrow \alpha)$, whereas both d_1 and d_2 provide partial allotment for the capability $(\alpha \rightarrow \beta)$ and dedicate the rest to other capabilities $[(\alpha \rightarrow \delta)$ and $(\delta \rightarrow \beta)]$, respectively. Second, part of a distributor's allotment could be set aside by the controller for later use (for example, 40% of the d_4 allotment is set aside as spare capacity for use by the overlay).

- *Management of distribution paths*

In this section, we discuss issues on the generation of distribution paths and the branching of path segments. A distribution path is said to be feasible if there exists some distributor along the path capable of emitting the requested content representation. The following algorithm determines whether a feasible distribution path exists. The algorithm attempts to find the availability of the requested representation as close as possible to the requesting receiver.

Algorithm 1 Given a core distribution tree $CDT'(m) \rightarrow \beta(m)$ provisioning representation $\beta(m)$ of content m, determine whether placement of a provisioning request R from subscriber S at receiver r for representation $\alpha(m)$ results in a branch of the core distribution tree $CDT'(m)$.

1. If $\alpha(m)$ is already provisioned at receiver r (for example, as needed for some other subscriber S' or for caching purposes), the controller requests r to multicast $\alpha(m)$ to S as well.
2. Otherwise, find and rank the set of distributors already provisioning $\alpha(m)$ according to some cost criteria such as estimated delay, utilization state, or capacity.
3. If the above set is not empty, find the closest distributor d_i willing to create the path segment $p[d_i, r, m] \rightarrow \alpha(m)$.

4. Otherwise, delay this request until critical mass is created according to some cost criteria such as the number of similarly pending requests.
5. Find and rank the set distributors in $CDT'(m)$ provisioning representation $\beta(m)$ according to some cost criteria such as estimated delay, utilization state, or capacity.
6. Find the closest distributor d_j willing to apply network capability $\beta(m)$ to $\alpha(m)$ and branch out the path segment $p[d_j, r, m] \rightarrow \alpha(m)$.

Otherwise, if no distributor d_i is found, $\alpha(m)$ cannot be provisioned at the current time by the overlay. In such a case, it is desirable to wait and determine whether a critical mass of $\alpha(m)$ requests exists to construct a new core distribution tree $CDT''(m)$ capable of emitting $\alpha(m)$, and then apply Algorithm 1 to the critical mass of requests. It is also desirable in such a case to migrate the $\alpha(m)$ branches out into the new core distribution tree $CDT''(m)$.

Once the need for a new distribution path is established, the controller instructs each node n_{i-1} along a distribution path $p[P, S, m]$ to set up and reserve an outgoing path segment to its successor n_i. After reserving outgoing path segments, the controller configures individual path segments into one distribution path (pointed to by *pid*) as follows. At each node n_i along a distribution path $p[P, S, m]$, the set of incoming path segments configured for *pid* are associated with the set of outgoing path segments configured for *pid*, thus creating the end-to-end distribution path $p[P, S, m]$. Each node associates the resulting coupling to a path identifier $pid(p[P, S, m])$.

Algorithm 2 A controller orchestrates an end-to-end distribution path in a manner analogous to that of the two-phase swipe used in RSVP [5]:

1. In the forward swipe,
   (a) The controller "requests" each node n_i in a distribution path to establish an outgoing path segment to its designated successor n_{i+1} (or successors when constructing a multicast tree).
   (b) At each node, network resources (e.g., sockets, bandwidth, buffers) are reserved and associated with the distribution path pointed to by *pid*.
   (c) To connect a source to a distributor, the controller instructs the source to connect to its distributor(s).
   (d) Once connected, the successor connects to its own successor(s).

   This process is repeated until the subscriber is reached, at which point the distribution path physically exists for the first time.

Table 3 Adaptive control primitives for media flows.

Node	*Command*
Distributor	*adapt*(n_i, n_{i+1}, *pid*, $\alpha(m)$)
	back-pressure(n_i, n_{i-1}, *pid*, m)

2. Once the distribution path $p[P, S, m]$ is physically established, a multimedia flow can be started over it. The backward swipe accomplishes this task.
   (a) To enable the flow of media between a source(s) and receiver(s), the controller starts the backward swipe by signaling receiver(s).
   (b) The receiver then signals its predecessor(s).
   (c) The predecessor then signals its predecessor(s), and so on.

   This process is repeated until it reaches the source(s), at which point the distribution path starts relaying content for the first time.

The tear-down of a distribution path is asymmetrical (based on a second-chance-to-live principle [41]) to its creation because we would like to foster the reuse of path segments. A node sends a tear-down request to its predecessor to indicate that it is no longer interested in an incoming path segment. The predecessor signals its own predecessor and decreases its reference count for this path segment. If this path segment is no longer in use, a tear-down timer is initiated, causing the tear-down of the path segment to be delayed. This provides an opportunity for the reuse of this path segment by other media flows. If the timer expires and the path segment is still not being used, the path segment is destroyed and data structures are updated.

In addition to the above setup functions, each node also implements path segment monitoring and maintenance tasks (see **Table 3**). Each node n_i in a distribution path estimates the performance of an outgoing path segment $p[n_i, n_{i+1}, m]$ with respect to the requirements of its corresponding distribution path. An assessment that indicates a need for remedial action at the node triggers fine-tuning of the media resolution on the outgoing path segment (i.e., $p[n_i, n_{i+1}, m]$). If the assessment indicates that remedial action is needed at a predecessor, back-pressure is applied to request predecessors to adapt to the conditions of the outgoing path segment $p[n_{i-1}, n_i, m]$. It is easy to see that each receiver oversees the dynamic reconfiguration for all of the distribution paths leading to it. When a receiver hands off to another receiver, the receiver also transfers the ownership of the distribution path.

Illustration **Figure 11** depicts the realization of a distribution path in response to a subscription request.

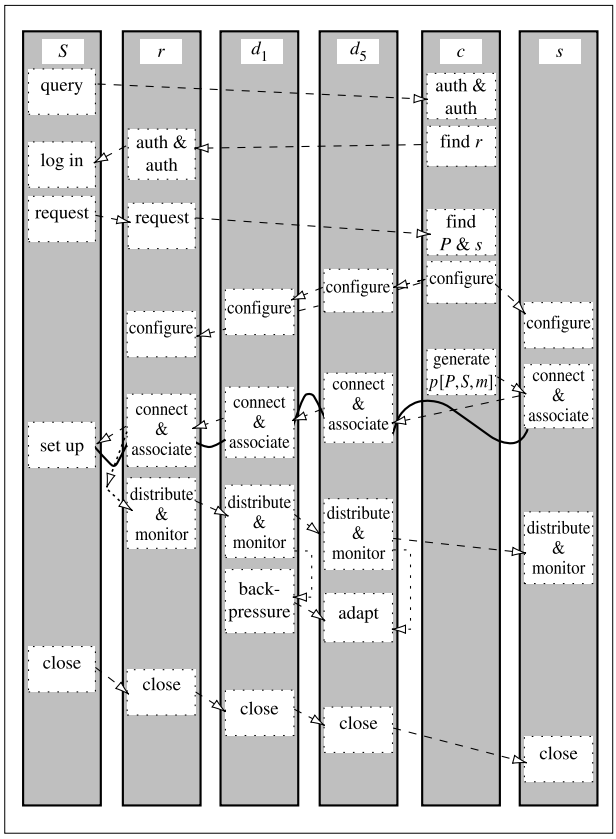


Figure 11

To span a distribution path, overlay nodes perform several control tasks and exchange control messages. A subscriber S is first authenticated, authorized, and assigned to a receiver r. The controller instructs nodes to set up outgoing path segments, causing admission controls to be applied and resources to be reserved. The controller then couples incoming and outgoing path segments into one distribution path. Once the distribution path $p[P, S, m]$ is physically established, a media flow may be started over it. The process is enabled by the receiver, which requests its predecessor(s), which then request their own predecessor(s), and so on, until it reaches the source(s), at which point the distribution path may begin relaying content for the first time. When tearing down a distribution path, a node may choose not to release an outgoing path segment until an associated timer expires in order to encourage its reuse by other distribution paths.

Table 4 Content distribution primitives for handling media flows.

Node	*Command*
Source	$connect(p[n_i, n_{i+1}, m], pid)$
Distributor	$associate(cid_{in}, cid_{out}, pid)$
Receiver	$distribute(n_i, pid)$
	$close(n_i, n_{i+1}, pid)$

In order to respond to a subscription request and span a distribution path, overlay nodes perform several control tasks and exchange control messages with other nodes. A subscriber S is first authenticated, authorized, and assigned to a receiver r. Although each node is independently addressable, only the controller must know their addresses. The controller is aware of its distributors, receivers, and sources, as well as of other controllers in the overlay. The controller uses the *configure*( ) command to instruct a node n_{i-1} along a distribution path $p[P, S, m]$ to set up an outgoing path segment to its successor n_i. Admission controls are applied, and resources (sockets, buffers, etc.) are reserved at each node. After reserving an outgoing path segment, the controller uses the *connect*( ) command to integrate individual path segments into one distribution path (pointed to by *pid*). Every media node implements the *associate*( ) command, used to concatenate an incoming path segment to an outgoing path segment and associate the resulting coupling with a path identifier $pid(p[P, S, m])$. At each node n_i along a distribution path $p[P, S, m]$, the set of incoming path segments configured for *pid* is associated with the set of outgoing path segments configured for *pid*, thus creating the end-to-end distribution path $p[P, S, m]$. It is possible (and desirable) for path segments to be reused across distribution paths. Once the distribution path $p[P, S, m]$ is physically established, a multimedia flow can be started over it. A node n_i in a distribution path uses the *distribute*( ) command to request the relay of multimedia content to its predecessor. The process is started by the receiver, which sends a distribute( ) command to its predecessor(s), which execute the command and then issue distribute( ) commands to its predecessor(s). The process repeats until it reaches the source(s), at which point the distribution path *starts relaying content for the first time*. Last, the *close*( ) command is used by a node n_i to tear down an outgoing path segment $p[n_i, n_{i+1}, m]$. A node may choose not to immediately release an outgoing path segment until an associated timer expires in order to encourage its reuse by other distribution paths. **Table 4** summarizes these setup and control primitives. **Figure 12** depicts the scope of these primitives with respect to nodes in a distribution path.

- *Subscription management*

Service provisioning is found in utility models, and its applicability to the Internet has recently been the focus of increasing research [2]. The overlay represents an open distribution network (much like the Advanced Intelligent Network [42]) being shared by independent and, possibly, competing content providers over which content is routed in the form of media flows to subscribers in an efficient and accountable manner. To provision subscription

requests, the overlay must perform several new application-level tasks currently not found in today's end-to-end routing of multimedia subscriptions to multimedia servers. In particular, the overlay must now perform several tasks (**Table 5**) involving different amounts of application-level negotiation of resources and services distributed throughout the overlay, where each such negotiation is possibly performed according to different optimization policies. In particular, the overlay performs the following tasks:

- Negotiate with candidate content providers for a request and determine their suitability.
- Select a binding to a suitable provider $x = P$ according to some cost criteria.
- Explore possible distribution paths so as to span a feasible distribution path from P to S through r—suited to the capabilities of S while attempting to make best use/reuse of existing overlay resources.

To subscribe to multimedia services, a subscriber S places a query to the controller requesting a local point of entry to the overlay. The controller responds by determining a suitable receiver r for S. The entry point to the overlay is simply the address of any such receiver. Note that whereas the address of the controller is well-known to subscribers, the addresses of other nodes (such as sources and receivers) need not be known by subscribers. Conventional authentication and authorization techniques could be enforced by the controller to entitle the subscriber to global overlay services. Additionally, individual nodes (e.g., receivers and sources) could also request the authentication and authorization of a subscriber to enforce access controls. Successfully authenticated and authorized subscribers are referred to as trusted subscribers.

Once a subscriber S is considered trusted, it places a provisioning request R (e.g., for content characterized by m at a cost of up to c^* units per minute) to its receiver r to request content from the overlay. Because subscribers are likely to be heterogeneous, a receiver must address the "impedance mismatch"[7] between subscribers and network capabilities. To do so, a subscriber characterizes its capabilities to its (current) receiver. For example, recall that in our generic content-adaptive application model (shown in Figure 2), each subscriber/client was characterized in terms of a capabilities bank, used by the overlay to select and configure those multimedia services found to be suitable for the capabilities of this subscriber. In our model, subscriber characterization is used by a receiver to augment a subscriber-oriented request into a

[7] The term *impedance mismatch* is borrowed from engineering circuit analysis [43] to denote the incompatibility between interfaces of two or more "plug-in" components.

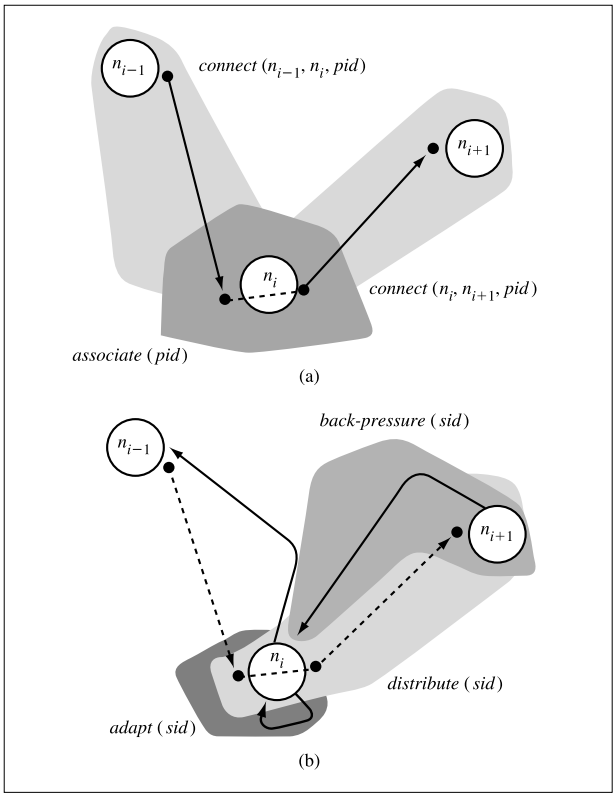


Figure 12

Scope control and media primitives. Part (a) shows how an incoming path segment is associated with an outgoing path segment at a node m_i. The association is requested by the controller for a particular path identifier *pid*. Part (b) shows the scope of distribution of a media flow. Adaptation measures over a media flow take place at a node. When adaptation at a node is found to be inefficient, a back-pressure request to the predecessor is sent, to request upstream adaptation.

Table 5 Handling of a provisioning request $p[x, S, m]$ by the controller with the ultimate goal of producing a binding between some source $x = P$ and the requesting receiver $r(S)$.

Step 1	Bind provider ($x = P$) to S for provisioning of content m.
Step 2	Determine suitable content representation $\alpha(m)$ for content m.
Step 3	Generate feasible distribution path $p[P, S, m] \rightarrow \alpha(m)$.
Step 4	Secure the distribution path.
Step 5	Create record of transaction.
Step 6	Enable media flow to path $p[P, S, m]$.

network-oriented request $p[x, S, m]$ (where x may be either known or unknown). The receiver then places such a provisioning request $p(x, S, m)$ (on behalf of S) to some controller on the overlay. The controller finds a suitable provider $x = P$ and generates a feasible distribution path from P to S through r—suited to the capabilities of S.

In a large network, subscribers necessitate a way to discover content of interest. One approach is to allow subscribers to explicitly choose providers; another approach is to defer such a task to the overlay. If the number of "channels" grows too large, the latter mechanism becomes more attractive. Regardless, we assume that the controller owns the responsibility of binding a subscriber to a provider. Receivers then forward subscription interests to a controller, which explores, negotiates, and secures subscription offers from different sources. This process involves exploring the feasibility of end-to-end application constraints (e.g., cost constraints and QoS parameters) as well as finding matches for subscription interests (e.g., keywords and content features).

In general, it is desirable for sources and receivers to regard the interior of the overlay solely as a distribution network, since the overlay is open to competing providers as well as subscribers. Therefore, means to secure the relay of content across the interior of the overlay (such as an encrypted media tunnel between sources and receivers) are desirable. Moreover, because of differences between subscribers, security mechanisms must address differences among providers as well as among subscribers. When required by subscribers, encryption measures could be relaxed from those associated with a shared channel with discouragement measures against casual content snooping. For example, when provisioning subscribers having reduced computational capabilities, selective encryption of a media flow (e.g., the encryption of critical media frames such as I-frames) could be used to trade off the robustness of content privatization against its corresponding computational decryption overhead.

In general, service provisioning requires a means of charging users. The construction of an information economy [44] that would encourage usage, generate revenue, and avoid chaos is a complex problem. Aside from asymptotic behavior, it is even unclear whether we should charge for connection time, contracted bandwidth, or rely on application-defined costs (e.g., per-capability service fees). Regardless of the charging scheme used, the framework must provide the ability to monitor usage throughout the overlay. However, tracking and monitoring are expensive functions that affect the scalability, security, and fault tolerance of the overlay. Centralization of billing is weak in terms of scalability and fault tolerance, but in general it is better for security. Distributed billing is better in terms of scalability and fault tolerance, but weaker in security. To this end, upon securing a distribution path, the receiver triggers the metering (i.e., usage tracking) of the path segment to a subscriber. Once a receiver terminates a subscriber connection, the receiver forwards the usage record to the source node(s) associated with the corresponding path segments. This is a fair model for the billing of actually delivered content when accounting for subscriber disconnection and failures.

6. Realization

In this section, we motivate extensions to existing technologies that must be developed in order to implement the proposed framework. In particular, we focus on networking protocols such as IP multicast, the experimental multicast backbone (MBONE), receiver-driven layered multicast (RLM), and Resource reSerVation Protocol (RSVP). We briefly describe each of these technologies and highlight their shortcomings in the context of building programmable multimedia overlay networks.

- *Existing technologies*

IP multicast [19, 20] provides the underlying network support for the efficient delivery of datagrams to groups of receiving hosts (i.e., receivers) referred to as multicast (host) groups. Host groups are identified by multicast group addresses. A datagram sent to a multicast group address is forwarded to all of its members by relaying the datagram over a set of routers capable of processing multicast addresses (i.e., multicast routers). Multicast routers that interface with one or more hosts are referred to as edge routers, whereas multicast routers that interface only with other routers are referred to as interior routers. Since individual receivers in a host group may be distributed across the whole Internet, it is likely that forwarding may traverse multiple multicast routers. To support efficient forwarding of multicast datagrams, IP multicast specifies two key functions: a group membership protocol and a multicast routing protocol. IP multicast specifies the Internet Group Management Protocol (IGMP) [45] to provide lightweight group monitoring at multicast edge routers. IGMP tracks multicast group addresses subscribed to by hosts (i.e., receivers) on the same network as an edge router. IP multicast specifies its routing function through distributed routing algorithms such as the Distance Vector Multicast Routing Protocol (DVMRP) [46] and Protocol Independent Multicast [47, 48] so that multicast datagrams sent to a group can be forwarded to the group members as efficiently as possible. A limitation of IP multicast routing is that it is not content-aware, since routing and group functions are driven only on network indicators (e.g., time-to-live counts).

MBONE [49] refers to the experimental "multicast backbone" that has been implemented on today's Internet by selectively deploying IP multicast technologies in routers. Since not all routers in the Internet are multicast-capable, MBONE utilizes "tunnels" to relay multicast datagrams across a cloud of non-multicast-capable routers. Upon entering a tunnel, IP-multicast packets are encapsulated as unicast IP packets (using the "IP-over-IP" [50] protocol). This encapsulation is removed upon exit from the tunnel, and the packet continues to be forwarded as a multicast datagram. MBONE provides user discovery and subscription of multicast groups via the Session Description Protocol (SDP) [51], which supports operations such as lookup and join (i.e., bind/subscribe) on the active multicast groups. IP multicast focuses solely on efficient forwarding of *identical* content to heterogeneous hosts. However, as discussed earlier, because of receiver heterogeneity and network congestion, it is highly desirable to adapt content delivery to match receiver capabilities and the network performance experienced by each receiver.

Receiver-driven Layered Multicast (*RLM*) [26] is a set of enhancements to IP multicast that transparently (to the network) addresses heterogeneity of receivers and the variability in network performance. With RLM, at a media source a layered encoder splits a given media stream m into multiple media layers $(l_1, \cdots, l_j)$, where l_1 is the base layer and the rest are enhancement layers [52]. These media layers are transmitted to separate multicast groups, each associated with the same multicast session. An RLM receiver interested in receiving a media stream transparently joins one or more of these multicast groups, depending on its requirements and the prevailing network congestion (as measured by packet loss). That is, each layer is given its own multicast group, say $g(m, l_i)$. A session name, say $s(m)$, is used to track the mapping of m into its multiple layered encoding process groups $g(m, l_i)$. RLM receivers seeking $s(m)$ are automatically subscribed to the base layer and to zero or more enhancement layers on the basis of a network-oriented performance indicator. Thus, a receiver may be a member of multiple multicast groups for the same multicast session. Since RLM relies primarily on network congestion/packet loss to determine the operational capabilities/characteristics of the receiver, it lacks explicit awareness of receiver capabilities. The receiver's layering decoder automatically joins and drops the enhancement multicast groups associated with the multicast session. Whereas increases in congestion/packet loss cause enhancement layers to be dropped, decreases in congestion cause enhancement layers to be added.

RSVP [5, 53, 54] is a receiver-initiated end-to-end signaling protocol that provides network-level signaling for reserving resources in end systems and network routers on the path between the sender and one or more receivers. With RSVP, reservation "soft states" are installed at all of the nodes participating in a reservation for an end-to-end flow, and maintained via periodic refreshes. RSVP supports resource reservation for unicast as well as multicast flows, and the primary model corresponds to one reservation per end-to-end application flow.

- *Necessary enhancements*

Realizing our framework requires a number of enhancements and extensions to the above-mentioned technologies, none of which is content-aware. Our network design philosophy was to introduce application-level intelligence (i.e., content and receiver awareness) into the network to promote a more efficient control over the routing of shared media flows. From the viewpoint of receivers, our goals were twofold:

- To enable receivers to subscribe to content with the best possible quality suiting their capabilities and preferences.
- To enable sources to provision content without *a priori* knowledge of the capabilities of the actual receiver audience.

We argue that while the existing technologies described above are indispensable, they do not provide sufficient support to meet our goals. We also provide a glimpse into key extensions that we are developing to embed content awareness into the existing network infrastructure.

As mentioned, IP multicast provides efficient forwarding of the same content to multiple receivers, while MBONE provides the necessary connectivity among multicast islands. However, neither of them is content-aware. At first glance, it may seem that RLM addresses the goals mentioned above. However, this is not the case, as is next explained. By augmenting multicast groups with the notion of optional receiver membership into one or more enhancement layers, RLM retrofits a *limited* degree of media awareness into multicast sessions without modifying the underlying IP multicast infrastructure. However, for the same reason it fails to satisfy the two goals above.

RLM fails in the first goal in that it relies inherently on a network-oriented characterization (i.e., a congestion signal) to approximate receiver capabilities and preferences. Thus, RLM effectively maps the entire range of receiver capabilities (such as network bandwidth, processing power, and error tolerance) into a single measure of available network bandwidth. This constitutes a loss of capability information about the receiver audience and violates goal G1 outlined earlier. RLM also does not satisfy the second goal. To generate an appropriate layered encoding, RLM implicitly requires each media source to determine *a priori* the entire serviceable operational space (in terms of receiver

capabilities and preferences) of the expected receiver audience. Such an approach is tractable only if relatively few discernible receiver operational points exist and can easily be determined by all media sources.

However, as argued in Section 1 and illustrated in Figure 1, we believe that this will not continue to be the case, given the advent of new client devices with limited processing power, such as thin clients and a variety of consumer hand-held devices, the increasing use of wireless and mobile network access, and an ever-increasing disparity in available processing power and network bandwidth for different clients. Moreover, the RLM approach requires that a feasible layered representation of media content be produced for each media type. However, not all media types may yield layered-encoding schemes capable of spanning all serviceable operational points of interest. As heterogeneity becomes prevalent, the number of discernible operational points in terms of receiver capabilities and preferences will increase. The operational space spanned by such extensive receiver heterogeneity must be efficiently managed for rapid proliferation of multimedia content. We believe that introducing explicit awareness of receiver capabilities and preferences into the network greatly facilitates application-level routing decisions to engineer available network resources in order to differentiate and match receiver heterogeneity and preferences. Such awareness can be harnessed to enhance content delivery while provisioning network resources for a variety of heterogeneous receivers.

As mentioned earlier, scalable provisioning of content-aware network resources involves the dynamic setup, management, and aggregation of end-to-end media flows. For several reasons, RSVP, the primary resource reservation signaling protocol for the Internet, is not immediately suitable for this purpose. First, RSVP is geared toward end-to-end network flows with no explicit awareness of media content. Second, RSVP provides a receiver-initiated mode of resource reservation in which a receiver communicates with the nearest previous-hop router. Finally, RSVP assumes a tighter coupling between the network view of a media source (i.e., the traversing of RSVP "path" messages) and the network view of a receiver (the traversing of RSVP "resv" messages). These reasons necessitate extensions to RSVP before it can be deployed to realize our framework, for which key abstractions (i.e., path segments, distribution paths, content distributors, and content distribution controllers) constitute application-level entities controlling media flows within the network.

- *Key extensions*

We are currently developing key extensions to the IP multicast, MBONE, RLM, and RSVP technologies in order to facilitate realization of our framework. These extensions introduce content awareness and application-state management into multicast routing and network resource provisioning. Two key extensions are content-aware multicast group management and multicast session directory services for discovery and subscription. Developing these extensions involves

1. Extending IP multicast semantics to associate content representation with multicast sessions.
2. Grouping together receivers with homogeneous characteristics and preferences for more efficient content delivery.
3. Defining new group membership policies based on content representation and receiver capabilities.

Details of the extensions and implementation of our framework are beyond the scope of this paper; they are the subject of forthcoming papers.

7. Concluding remarks

We have presented an evolutionary framework for multimedia distribution services that efficiently addresses the needs of groups of heterogeneous receivers. The framework integrates several evolutionary measures. First, we have introduced receiver and content awareness into the network. Second, we have modeled the distribution of content between sender and receiver as a relay across multiple intermediaries. Finally, we have exposed media flows to content-aware network services. We propose the realization of these measures by overlaying a programmable multimedia network over the packet-switched network. We have modeled the overlay as a content-aware active network and illustrated the benefits of content-aware programmability over the routing and management of media flows.

The framework addresses various degrees of heterogeneity among "distribution paths" to receivers. The distribution path realizes a content-aware connection between a distribution tree and one or more receivers. The framework enables efficient generation of distribution paths to dynamically adapt to groups of receivers, thereby creating a programmable network capable of autonomously generating branches out of the core distribution tree of a multicast.

Our approach treats media flows as first-class entities within the interior of the overlay, in contrast to just at the exterior of the overlay. This allows the overlay to exploit application-level management of distribution paths and overlay network services to influence the routing of multimedia payloads on the underlying packet-switched network. In summary, our framework provides a vehicle for retrofitting today's packet-switched networks into the next generation of intelligent and programmable multimedia networks.

Acknowledgments

The authors thank F. Hendriks, J. Von Kaenel, L. Lumelsky, the *e*-Media Technology Group at the IBM Thomas J. Watson Research Center, and the anonymous reviewers of this paper for feedback on an earlier draft of this manuscript. An earlier draft of this manuscript was circulated at the IBM Thomas J. Watson Research Center from January through May 1998.

**Trademark or registered trademark of Sun Microsystems, Inc.

References

1. B. Leiner, V. Cerf, D. Clark, R. Kahn, L. Kleinrock, D. Lynch, J. Postel, L. Roberts, and S. Wolf, "The Past and Future History of the Internet," *Commun. ACM* **40,** No. 2, 102–108 (February 1997).
2. J. Birnbaum, "Pervasive Information Systems," *Commun. ACM* **40,** No. 2, 40–41 (February 1997).
3. C. Aurrecoechea, A. Campbell, and L. Hauw, "A Survey of QoS Architectures," *Multimedia Syst. J.* **6,** No. 3, 138–151 (May 1998).
4. H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks," *Proc. IEEE* **83,** No. 10, 1374–1396 (1995).
5. L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A New Resource Reservation Protocol," *IEEE Network Magazine* **7,** 8–18 (September 1993).
6. J. Jamison and R. Wilder, "vBNS: The Internet Fast Lane for Research and Education," *IEEE Communications Magazine* **35,** No. 1, 60–63 (January 1997).
7. C. Song, L. Cunningham, and R. Wilder, "Quality of Service Development in the vBNS," *IEEE Communications Magazine* **36,** No. 5, 54–60 (May 1998).
8. M. Luker, "NSF's New Program for High-Performance Internet Connections," *Commun. ACM* **39,** No. 10, 27–28 (October 1996).
9. M. Naghshineh and M. Willebeek-LeMair, "End-to-End QoS Provisioning in Multimedia Wireless/Mobile Networks Using an Adaptive Framework," *IEEE Communications Magazine* **35,** No. 12, 72–81 (November 1997).
10. D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden, "A Survey of Active Network Research," *IEEE Communications Magazine* **35,** No. 1, 80–86 (January 1997).
11. D. L. Tennenhouse and D. J. Wetherall, "Towards an Active Network Architecture," *Computer Commun. Rev.* (originally published in *Multimedia Computing and Networking '96*) **26,** No. 2, 5–18 (April 1996).
12. D. D. Clark and D. L. Tennenhouse, "Architectural Considerations for a New Generation of Protocols," *Proceedings of the ACM Symposium on Communications Architectures & Protocols* (*ACM SIGCOMM '90*), September 1990, pp. 200–208.
13. Michael Villeret, P. A. Deschenes, and H. Stephene, "A New Digital Technique for Implementation of Any Continuous PCM Companding Law," *Proceedings of the IEEE International Conference on Communications*, Vol. 1, 1973, pp. 11.12–11.17.
14. A. I. Rudnicky, A. G. Hauptmann, and K. F. Lee, "Survey of Current Speech Technology," *Commun. ACM* **37,** No. 3, 52–57 (March 1994).
15. T. Berners-Lee, R. Cailliau, A. Luotonen, H. Frystyk Nielsen, and A. Secret, "The World-Wide Web," *Commun. ACM* **37,** No. 8, 76–82 (August 1994).
16. David Clark and Joseph Pasquale, "Strategic Directions in Networks and Telecommunications," *ACM Computing Surv.* **28,** No. 4, 679–690 (December 1996).
17. V. G. Cerf and R. E. Kahn, "A Protocol for Packet Network Interconnections," *IEEE Trans. Commun.* **5,** 627–641 (May 1974).
18. W. Stallings, *TCP/IP Illustrated*, Vol. 1, Addison-Wesley Publishing Co., Reading, MA, 1994.
19. S. Deering, "Host Extensions for IP Multicasting," *Request for Comments RFC 1112*, Internet Engineering Task Force (IETF), 1895 Preston White Dr., Suite 100, Reston, VA 22091, August 1989.
20. S. E. Deering and D. R. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs," *ACM Trans. Computer Syst.* **8,** No. 2, 85–110 (May 1990).
21. B. C. Housel and D. B. Lindquist, "WebExpress: A System for Optimizing Web Browsing in a Wireless Environment," *Proceedings of the Second Annual International Conference on Mobile Computing and Networking*, September 1996, pp. 108–116.
22. D. Hoffman, M. Speer, and G. Fernando, "Network Support for Dynamically Scaled Multimedia Data Streams," *Proceedings of the Fourth International Workshop on Network and Operating System Support for Digital Audio and Video* (*NOSSDAV '93*), May 1993, pp. 251–262.
23. L. Delgrossi, C. Halstrick, D. Hehmann, R. Herrtwich, J. Sandvoss, O. Krone, and C. Vogt, "Media Scaling for Audiovisual Communication with the Heidelberg Transport System," *Proceedings of ACM Multimedia*, November 1993, pp. 99–104.
24. C. Diot, W. Dabbous, and J. Crowcroft, "Multipoint Communication: A Survey of Protocols, Functions, and Mechanisms," *IEEE J. Selected Areas of Commun.* **15,** No. 3, 277–290 (April 1997).
25. S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Light-Weight Session and Application Level Framing," *IEEE/ACM Trans. Networking* **5,** No. 6, 784–803 (December 1997).
26. S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-Driven Layered Multicast," *Proceedings of ACM SIGCOMM '96*, August 1996, pp. 117–130.
27. D. Scott Alexander, M. Shaw, S. M. Nettles, and J. M. Smith, "Active Bridging," *Proceedings of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 1997, pp. 101–111.
28. M. Abrams, C. Standridge, G. Abdulla, S. Williams, and E. A. Fox, "Caching Proxies: Limitations and Potentials," *Proceedings of the Fourth International World-Wide-Web Conference*, December 1995, pp. 119–133.
29. G. D. H. Hunt, G. S. Goldszmidt, R. P. King, and R. Mukherjee, "Network Dispatcher: A Connection Router for Scalable Internet Services," *Computer Networks and ISDN Syst*. **30,** 347–357 (April 1998).
30. A. Fox, S. D. Gribble, E. Brewer, and E. Amir, "Adapting to Network and Client Variability via On-Demand Dynamic Distillation," *Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems* (*ASPLOS*), October 1996, pp. 160–170.
31. A. Fox, S. D. Gribble, Y. Chawathe, E. Brewer, and P. Gauthier, "Cluster-Based Scalable Network Services," *Proceedings of the Sixteenth ACM Symposium on Operating Systems Principles*, October 1997, pp. 78–91.
32. N. Yeadon, F. Garcia, D. Hutchinson, and D. Shepherd, "Continuous Media Filters for Heterogeneous Internetworking," *Multimedia Computing and Networking* (*MMCN'96*), *Proc. SPIE* **2667,** 246–257 (January 1996).
33. A. Campbell, A. Lazar, H. Schulzrinne, and R. Stadler, "Building Open Programmable Multimedia Networks," *IEEE Multimedia* **4,** No. 1, 77–82 (January–March 1997).

34. J. Pasquale, G. Polyzos, E. Anderson, and V. Kompella, "Filter Propagation in Dissemination Trees: Trading Off Bandwidth and Processing in Continuous Media Networks," *Proceedings of the Fourth International Workshop on Network and Operating System Support for Digital Audio and Video* (*NOSSDAV '93*), May 1993, pp. 269–278.
35. T. J. Mowbray and W. A. Ruh, *Inside Corba: Distributed Object Standards and Applications* (Addison-Wesley Object Technology Series), Addison-Wesley Publishing Co., Reading, MA, 1997.
36. A. Birrell, G. Nelson, S. Owicki, and E. Wobber, "Network Objects," *Proceedings of the 14th ACM Symposium on Operating Systems Principles*, December 1993, pp. 217–230.
37. A. Balachandran, A. Campbell, and M. E. Kounavis, "Active Filters: Delivering Scaled Media to Mobile Devices," *Proceedings of the Seventh IEEE International Workshop on Network and Operating Systems Support for Digital Audio and Video*, May 1997, pp. 125–134.
38. K. Arnold and J. Gosling, *The Java Programming Language*, Addison-Wesley Publishing Co., Reading, MA, 1996.
39. N. Manohar, M. Willebeek-LeMair, and A. Prakash, "Applying Statistical Process Control to the Adaptive Rate Control Problem," *Proceedings of IS&T SPIE's 10th Annual Symposium on Multimedia Computing and Networking*, San Jose, CA, January 1998, pp. 42–56.
40. D. C. Montgomery, *Introduction to Statistical Quality Control*, chapter entitled "Other Statistical Process Control Techniques," John Wiley & Sons, Inc., New York, 1990, pp. 313–357.
41. A. Agarwal, J. Hennessy, and M. Horowitz, "Cache Performance of Operating System and Multiprogramming Workloads," *ACM Trans. Computer Syst.* **6,** No. 4, 393–431 (November 1988).
42. M. P. Kaplan, L. Gabuzda, and N. Shah, in *The Intelligent Network Standards: Their Application to Services* (McGraw-Hill Series on Telecommunications), I. Faynberg, Ed., McGraw-Hill Book Co., Inc., New York, 1996.
43. J. W. H. Hyat and J. Kemmerly, *Engineering Circuit Analysis*, McGraw-Hill Book Co., Inc., New York, 1978.
44. J. K. MacKie-Mason, S. Shenker, and H. Varian, "Service Architecture and Content Provision: The Network Provider as Editor," *Telecommun. Policy* **20,** 203–217 (1996).
45. W. Fenner, "Internet Group Management Protocol," Version 2, *Request for Comments RFC 2236*, Internet Engineering Task Force (IETF), 1895 Preston White Dr., Suite 100, Reston, VA 22091, November 1997.
46. S. E. Deering, D. Waitzman, and C. Partridge, "Distance Vector Multicast Routing Protocol," *Request for Comments RFC 1075*, Internet Engineering Task Force (IETF), 1895 Preston White Dr., Suite 100, Reston, VA 22091, November 1988.
47. S. Deering, D. L. Estrin, D. Farinacci, Ching-Gung Liu, Van Jacobson, and L. Wei, "The PIM Architecture for Wide-Area Multicast Routing," *IEEE/ACM Trans. Networking* **4,** No. 2, 153–162 (April 1996).
48. D. Estrin, D. Farinacci, A. Helmi, D. Thaler, S. Deeri, P. Sharma, and L. Wei, "Protocol Independent Multicast—Sparse Mode (PIM–SM): Protocol Specification," *Request for Comments RFC 2362*, Internet Engineering Task Force (IETF), 1895 Preston White Dr., Suite 100, Reston, VA 22091, June 1998.
49. H. Eriksson, "MBONE: The Multicast Backbone," *Commun. ACM* **37,** No. 8, 54–60 (August 1994).
50. C. Perkins, "IP Encapsulation Within IP," *Request for Comments RFC 2003*, Internet Engineering Task Force (IETF), 1895 Preston White Dr., Suite 100, Reston, VA 22091, October 1996.
51. M. Handley and V. Jacobson, "SDP: Session Description Protocol," *Request for Comments RFC 2327*, Internet Engineering Task Force (IETF), 1895 Preston White Dr., Suite 100, Reston, VA 22091, April 1998.
52. S. McCanne, M. Vetterli, and V. Jacobson, "Low Complexity Video Coding for Receiver-Driven Layered Multicast," *IEEE J. Selected Areas of Commun.* **15,** No. 6, 983–1001 (August 1997).
53. R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource Reservation Protocol (RSVP)—Version 1 Functional Specification," *Request for Comments RFC 2205*, Internet Engineering Task Force (IETF), 1895 Preston White Dr., Suite 100, Reston, VA 22091, September 1997.
54. J. Wroclawski, "The Use of RSVP with IETF Integrated Services," *Request for Comments RFC 2210*, Internet Engineering Task Force (IETF), 1895 Preston White Dr., Suite 100, Reston, VA 22091, September 1997.

Received June 10, 1998; accepted for publication July 15, 1999

Nelson R. Manohar *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598 (nelsonr@watson.ibm.com).* Dr. Manohar received the B.S. degree in computer engineering from the University of Puerto Rico at Mayaguez in 1987, the M.S. degree in computer engineering from the University of Wisconsin at Madison in 1988, and the M.S.E. in industrial and operations engineering and Ph.D. in computer science and engineering from the University of Michigan at Ann Arbor in 1992 and 1997, respectively. In 1997 Dr. Manohar joined the IBM Thomas J. Watson Research Center, where he is currently a Research Staff Member in the *e*-Media Technology group. From 1988 through 1991, Dr. Manohar worked at the AT&T Bell Laboratories, Naperville, IL, as a Member of Technical Staff on the Advanced Intelligent Network. His primary research interests are in exploratory research of applications and infrastructure for the next-generation Internet, where he has worked and led the filing of several patents on end-to-end resource management for global ubiquitous media services and asynchronous collaborative multimedia systems.

Ashish Mehra *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598 (mehraa@watson.ibm.com).* Dr. Mehra received the B.Tech. (Bachelor of Technology) degree in electrical engineering from the Indian Institute of Technology at Kanpur, India, in 1989, and the M.S.E. and Ph.D. degrees in computer science and engineering from the University of Michigan in 1992 and 1997, respectively. He is currently a Research Staff Member in the Server and Enterprise Networking group at the IBM Thomas J. Watson Research Center. His primary research interests are in server operating system and networking support for application quality of service requirements, network services for intelligent multimedia transport, network security, and code mobility.

Marc H. Willebeek-LeMair Dr. Willebeek-LeMair is no longer with IBM; his address was not available at the time of publication. Dr. Willebeek-LeMair received the B.S. degree in computer and electrical engineering from George Mason University, Fairfax, Virginia, in 1985, and M.S. and Ph.D. degrees from the School of Electrical Engineering at Cornell University, Ithaca, New York, in 1988 and 1990, respectively. At the time this paper was written, he was with the

Multimedia Networking group at the IBM Thomas J. Watson Research Center, specializing in the development of networked multimedia systems. Dr. Willebeek-LeMair joined IBM in 1990 as a Research Staff Member in the Research Division High Bandwidth Systems Laboratory. His research interests include real-time networked applications such as desktop videoconferencing and audio/video streaming, high-bandwidth communications, computer architecture, parallel processing, and interconnection networks. Dr. Willebeek-LeMair is a member of the IEEE Computer Society, the IEEE Communications Society, Alpha Xi, and Eta Kappa Nu.

Mahmoud Naghshineh *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598 (mahmoud@us.ibm.com).* Dr. Naghshineh currently manages the Pervasive and Embedded Networking Department. He has been working at IBM since 1988 on a variety of research and development projects dealing with design and analysis of local area networks, communication protocols, and fast packet-switched/broad-band networks, TCP/IP networking, quality of service and congestion control, wireless and mobile ATM, wireless radio and infrared access broad-band and local-area networks. He received his doctoral degree from Columbia University. Dr. Naghshineh has served as a member of technical program committee, session organizer, and chairperson for many IEEE conferences and workshops, and as editor of a number of ACM and IEEE journals and magazines. He is also the editor-in-chief of the IEEE personal communications magazine starting in the year 2000. He is currently an adjunct faculty member of the Department of Electrical Engineering at Columbia University, where he teaches a course on wireless/mobile communications and networking.