# 1
# Programming Telecommunication Networks

*Aurel A. Lazar*
*Department of Electrical Engineering and*
*Center for Telecommunications Research*
*Columbia University, New York, NY 10027-6699*
*http://comet.ctr.columbia.edu/~aurel*

*Abstract*

The recent move towards market deregulation and open competition has sparked a wave of serious introspection in the telecommunications service industry. Telecom providers and operators are now required to open up their primary revenue channels to competing industries. In this paper, we examine the service structure of two major global communication networks - the Telephone Network and the Internet and explore their weaknesses and strengths.

Building upon the experience we gained during the development of the initial **xbind** prototypes, we discuss the realization of an open programable networking environment based on a new service architecture for advanced telecommunication services. Our approach to the problem stems from two angles - one conceptual, the other implementational.

In the first, we attempt to develop a service model that is open and reflects the economic market structure of the future telecommunications service industry. We believe that investigating such a model will help clarify some of the pertinent issues confronting the telecommunications service industry today as it comes of age. An engineering model for realizing the service market model is proposed as a vehicle for creating multimedia services on broadband networks. In the second, we investigate the feasibility of engineering this model from an implementation standpoint. We address some of the important issues fully aware that our work has opened new vistas that call for additional research.

We believe that our work will lead to a major shift of paradigm in research and development of service architectures for telecommunication networks. It has already found strong resonance with the members of the OPENSIG international working group and its foundations have been proposed for possible standardization by Nortel.

## 1. INTRODUCTION

The ability to rapidly create and deploy new and novel services in response to market demands will be the key factor in determining the success of the future service provider. As the high speed switching and communication infrastructure improve and bandwidth becomes a commodity, it is envisioned that the competition for product differentiation will increasingly depend on the level of sophistication, degree of flexibility and speed of deployment of services that a future provider can offer [31]. These factors in turn depend heavily on the flexibility of the software architecture in place in a provider's operational infrastructure.

The current generation of telecommunication networks is based on an architecture over 30 years in age [18]. The basic tenet behind the architecture is the implicit assumption that Customer Premises Equipment (CPE) has no computational

Building QoS into Distributed Systems A. Campbell & K. Nahrstedt (Eds.)

capabilities and limited modes of interaction. This assumption eventually translated into a design somewhat akin to a mainframe cluster model where a small number of computationally powerful processors called Service Control Points (SCPs) are distributed throughout the network and take on the responsibility of service provisioning for all connected CPEs. As in the mainframe cluster model, the CPEs themselves act solely as the user interface channelling simple user requests and responses into the system. Within the network, dedicated processors running specialized monolithic software optimized for efficiency process, co-ordinate and translate these requests and responses into the necessary data and connections that constitute the service.

The primary deficiency of this architecture is its monolithic view of the service provisioning process. The network operator assumes almost complete control over the decisions pertaining to the design, introduction and management of services since it owns the SCPs. Interfaces to the service management infrastructure are often non-existent, proprietary or narrow in scope and intimately coupled to the hardware they operate on. As a result, any third-party involvement in service programming is limited to customizing only a small set of operational parameters. Furthermore deploying new services in today's telephone networks takes up to several years primarily because the software systems with which they need to be integrated are enormously complex and prone to many cross-service interactions.

Ironically on the other hand, the capabilities of the modern computer has advanced well beyond the stifling limitations imposed by these early architectures. Modern software engineering has advanced to the point where industry standards [29] now exist for implementing platform independent distributed component based software. These and newer emerging standards allow the construction and packaging of independent software components into suites which can further be assembled via application-level frameworks to create a truly distributed information infrastructure on a global scale. While these modern software engineering aids by themselves do not solve the fundamental problems inherent in any scalable distributed system, they do provide an excellent infrastructure for dealing with problems of programmability, portability, maintainability and reusability, problems frequently faced by the telecommunications software industry.

Recent advances in distributed systems and transportable software together with increasing demand for better control of QOS in multiservice networks are driving a re-examination of network software architectures. A new opportunity exists to reconcile the perspectives of the computing and communication communities in new network architectures that support service creation, QOS control, and the joint allocation of computing and communications resources.

Since the Fall of 1994 we have been experimenting with a first generation broadband kernel called **xbind** [20], [26]. The broadband kernel is a programmable operating platform that supports the creation, deployment and management of networked multimedia services (e.g., virtual circuits, virtual paths, virtual networks, multicast, etc.) and mechanisms for efficient resource allocation (e.g., connection management, route management, admission control, QOS mapping, etc.) . The term 'kernel' is

deliberately used to draw a parallel between its role as a resource allocator and extended machine, and that of a typical operating system. The broadband kernel behaves as a resource allocator because it mediates and arbitrates between conflicting requests for resources made by various parties in the system. It functions like an extended machine because it provides a simplified means of accessing fundamental system services by abstracting away the operational complexities of provisioning these services.

Building upon the experience we gained during the development of the initial **xbind** prototypes [10], [26], we discuss the realization of an open programmable networking environment based on a new service architecture for advanced telecommunication services. Our approach to the problem stems from two angles - one conceptual, the other implementational. In the first, we attempt to develop a service model that is open and reflects the economic market structure of the future telecommunications service industry. We believe such a model will help clarify some of the pertinent issues plaguing the telecommunications service industry today as it comes of age. In the second, we investigate the feasibility of engineering this model from an implementation standpoint. We address some of the important issues fully aware that our work has opened new vistas that call for additional research.

We begin in Section 2 by proposing a simple classification scheme for services. We use this scheme to examine the service structure of two prominent networks - the Telephone Network and the Internet. In Section 3, we introduce the basis of an economic model for describing future telecommunication services based on market forces. We believe this will be the end result of a natural evolution of the industry given the current trends in deregulation and open competition. In Section 4 we propose a service architecture based on the principles of open APIs that closely parallels our economic model and briefly list out some of its principle components. In section 5 we present the service creation model. Quality of service, performance and scaling issues are dealt with in section 6. Implementation considerations are presented in section 7.

## 2. SERVICE ARCHITECTURES: A TAXONOMY OF THE STATE-OF-THE-ART

A service architecture defines the structure and mode of operation of a facility that offers a service. There are two basic services in most communication infrastructures:

1. *Basic Communication Services*, which focus on the mechanisms and the interactions between network entities so as to enable the basic communication process; and
2. *Content Services*, which focus on the means of access, presentation and organization of content resources in the network to facilitate communication.

A third category of services deals with value-added enhancements over the basic communication services. Conceptually these services lie between the two basic categories in the sense that their utility is dependent on the functioning of the basic communication service yet by themselves are not considered as such. These services are exemplified by the Advanced Intelligent Network (AIN) [8] services and range

from convenience features like call forwarding and caller ID, to more complex services like mass calling.

## 2.1. Service Requirements - the Need for Domains

The conflicting forces of market demand for both flexibility, accountability and robustness makes the task of architecting an open service model a difficult technical challenge. Basic communication services lie at the base of the service spectrum supporting the fundamental mechanism for information exchange upon which the other service categories are derived. Thus, the primary requirement of basic communication services is one of robustness, high availability and low failure rate. Content services, on the other hand lie at the opposite end of the spectrum. Market forces demand these to be highly customized, user oriented and easy to deploy.

Hence, as we ascend the hierarchy of services from the basic communication services to sophisticated content-driven end-user services, the need for programmability also rises. The difference in requirements for these service categories often result in different technical solutions being employed for addressing the issues pertinent to each category. In effect these enforce a natural domain-like separation between network concerns, service concerns and user concerns. A similar view is reflected in the Telecommunications Information Networking Architecture Consortium (TINAC) stakeholder domains [6] which classify TINAC services into network provider domains, service provider domains and user domains.

## 2.2. Service Models - a Question of APIs?

The service architectures of the Internet and the Telephone Network differ in several aspects. In the Telephone Network, historical assumptions about CPE capabilities has led to a two tiered architecture consisting of a user domain and a network domain. (The service architecture of ATM Networks is closely related to the one of the Telephone Network.) Users lie at the periphery of the system and access network services via a thin interface known as the User Network Interface (UNI) [9] while within the network domain, interconnection is achieved via a complex Network-Node-Interface (NNI). In this model, there is little distinction between network provider and service provider since most useful services (in particular, IN types) require the intimate network support available only through an NNI. In this sense, although there is a clear separation between bearer services and AIN services in a technical and engineering sense, in reality until only recently, the administrative, operational and business concerns of an enterprise often make it more lucrative to merge network and service provider roles as one. For instance, in the past, protective regulatory structures had always afforded network operators the luxury of using AIN services as a value-added component to enhance the marketing and sales of plain old bearer services.

The recent 1996 Telecommunications Act however, has demolished to a considerable extent many of these protectionistic measures, paving the way for freer competition at all levels in the market. One of the more significant changes mandated by the new regulations effectively require that network operators provably demonstrate capability to support third party service provisioning. These and other emerging trends are

indicative of the need for a serious re-examination of current telecommunication service models (and the ensuing business practices) or risk serious financial consequences.

The intimate coupling of the communications and service architecture makes the introduction of new services, particularly those that do not conform to the traditional point-to-point connection paradigms clumsy and restrictive because:

1. The interface between the network and the service implementor (which is responsible for basic communication services like connection setup procedures) is rigidly defined and cannot be replaced, modified nor supplemented - all services must be implemented in terms of these.
2. The interface between individual services is defined by the rather restrictive Intelligent Network Access Point (INAP) protocol which all conceivable service-level signalling procedures and semantics must map into.

An even worst situation presents itself if we examine the boundary between the network and the user. The potential diversity and flexibility requirements of user level services and applications far exceed that of typical AIN services demanding all the more an open environment for design, installation and operation. The simple UNI is not extendible and was never designed for these requirements.

In the Internet, the lack of a UNI implies a conceptually flat service structure where there is no strict distinction between network provider, content provider and user [7]. In practice domain-like separation (usually for security and administrative reasons) is typically imposed through artificial means (like addressing structure and naming). In this model, any user can also be:

1. A network provider by achieving physical connectivity with the Internet and offering connectivity services to other users.
2. A content provider by making available content for public access and advertising its availability through a directory service.

In terms of APIs for content provisioning, the Internet far excels the Telephone or ATM Networks. The Java virtual machine [29] in effect can be seen as a freely extendible API for content services whose basic parameters are programs instead of simple types. Even at the basic communication services level, IP options provide, in principle, a primitive API for influencing basic packet routing policies. In reality, most of these APIs are usually not fully implemented or supported. For example, the source-routing option of IPv4 is an example of an API which, if fully supported by all IP-capable hosts, would provide elegant solution to the problem of host mobility without the need for tunnelling.

In summary, the service model of the Telephone and ATM Networks is one of provider and customer. This is a clear demarcation in terms of the technology employed (as reflected by the APIs) and responsibility between the two distinct roles. In the Internet this distinction is less clear and a peer-to-peer model is perhaps a more accurate analogy since users and providers both run on essentially the same technology. The

obvious advantage of a peer-to-peer service model is one of flexibility since there are no technical barriers from preventing any user from setting up his/her own service. On the downside, security, policy and standards are harder to enforce since they require cooperation from a much larger community.

## 3. MOVING BEYOND THE STATE-OF-ART: A LAYERED MARKET MODEL

In the previous section, we examined two primary service models and their respective trade-offs. We note that a key factor influencing the service model of a network depends to a large extent on the types and levels of APIs available. In fact, we believe that APIs primarily reflect the flexibility and maturity of a service architecture and good APIs at the service level are the macroscopic counterparts of good coding practices at the implementation level.

In this section, we propose a 3 layered API based service model inspired by the principle of the open market which we believe better reflects the operating structure of the future communication services industry. The basic tenet of the model is the premise that the future telecommunication service industry will operate within an open market where sufficient alternatives exist to allow services to be traded as commodities. This premise is not unreasonable given the recent trend towards deregulation and the general move towards streamlined horizontal market structures.

The model is divided into 3 layers with each layer representing a market. At the lowest layer, is a hardware market where numerous equipment manufacturers and vendors offer hardware and firmware solutions for building the basic communication infrastructure. The customers of this market are typically network carriers, third party software developers who specialize in developing software for service providers and a handful of service providers themselves. The APIs provided by vendors in this market allow their users to write basic communication services and the associated middleware components. In the second layer is a middleware service market where carriers, software developers and middleware service providers offer middleware service products to customers who are in the user service provisioning business. The APIs provided in this market are sufficiently high to allow development of any consumer level service. Finally at the highest layer is the consumer services market where consumer service providers compete to bundle, integrate and customize their wares in the most appealing form for the mass market. Within each market there may exist brokers (as rightly recognized by the TINAC stakeholder domain model [30]) whose role is to mediate dealings between buyers and sellers who, because of regulatory or business policies, cannot transact directly. The model is shown in Figure 1.

The service model just described falls somewhere between the Internet's peer-to-peer model and the Telephone Network's strict provider-customer model. It allows, in principle, the cooperation of any number of entities in the network for realizing a common service as well as the competition among services for network resources. As will be shown in the next section, the corresponding engineering model can be parametrized in such a way that the basic characteristics of the peer to peer model as well as the characteristics of the provider and costumer model can be recovered.

Within each layer (or market), players are free to enter and buy, sell or rebundle each other's services. Across layers, the relationship takes on more of the form of provider-customer. Once again, APIs play the crucial role of defining market boundaries.

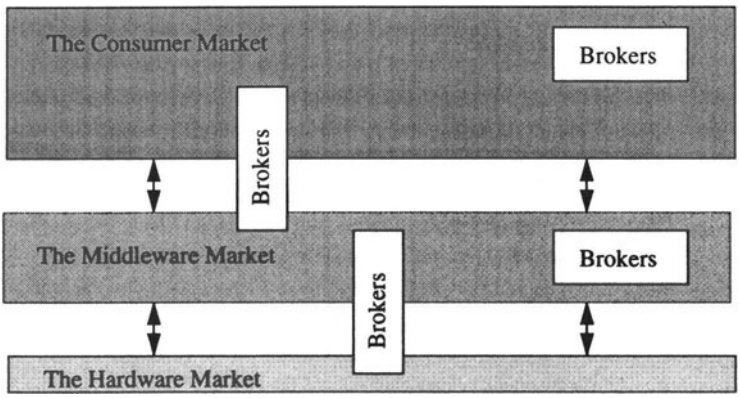


Fig. 1. The Layered Market Model

## 4. REALIZING THE SERVICE MARKETPLACE

The layered market model outlined in the previous section is merely an economic model reflecting the author's vision of the future telecommunications service industry. In the remaining sections of this paper, we focus on designing and realizing a novel service architecture in the technical sense which closely reflects the philosophy of the layered market model. The architecture we describe is targeted towards multimedia services as opposed to the economics of the general market model just presented. We begin by briefly describing the Extended Reference Model (XRM) [17] for multimedia networks and its decomposition into 3 submodels [20].

### 4.1. The Extended Reference Model

The XRM models the communications architecture of networking and multimedia computing platforms. It consists of 3 components called, the *Broadband Network* (the R-model), the *Multimedia Network* (the G-model) and the *Services and Applications Network* (the B-model, see Figure 2). The broadband network is defined as the physical network that consists of switching and communication equipment and multimedia end-devices. Upon this physical infrastructure, resides the multimedia network whose primary function is to provide the middleware support needed to realize services with end-to-end QOS guarantees over the physical media-unaware network. This is achieved by deriving from the broadband network a set of QOS abstractions. The latter jointly define the resource management and control space.

The process of service creation calls for resource reservation and distributed state manipulation algorithms. From this perspective, the multimedia network provides a *programming model* that allows service behavior to be specified and executed. Service

abstractions represent the states of a service created using algorithms native to the multimedia network. These abstractions are used by the services and applications network for managing and creating new services through dynamic composition and binding.

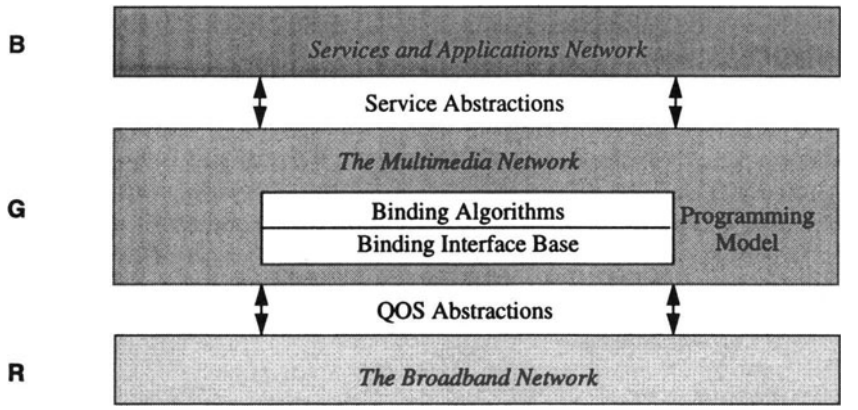


Fig. 2. Overview of the RGB decomposition of the XRM

## 4.2. The Power of APIs in the XRM

As was mentioned previously, the functionality and indeed the level of sophistication of a service architecture is chiefly characterized by the APIs it offers. In the XRM, 2 types of APIs are available for building services. These are represented by QOS and service abstractions that lie at the interfaces between the R- and G- (also denoted R||G), and G- and B-models (G||B), respectively.

The primary power of these APIs is the tremendous flexibility available for service providers and users alike to mold the structure of the network in a way that reflects economic policies and business practices. By this we mean that these various levels of APIs allow different parties or stakeholders to influence the partitioning of resources to carve out natural market niches or even create wholly new markets. In other words, unlike the service architectures of the Telephone Network or the Internet which tend to fall along the lines of 'all or nothing', we envision a future service marketplace to be rich in choices, variations and sophistication.

### *R||G Interface APIs: QOS Abstractions*

R||G interface APIs abstract the states of local multimedia resources (both logical and physical) in the broadband network. These resources represent the devices, switches, links, processors and their respective capacities. Resources representing name spaces (e.g., VP/VC Identifiers) are also modeled. In the XRM, R||G interface APIs are implemented as a collection of distributed autonomous software entities with open interfaces. The open interfaces allow the states of these resources to be monitored,

controlled and managed remotely. More importantly, they allow these resources to be treated as independent pluggable components and services to be built by cleverly interconnecting combinations of them.

From the perspective of the G-model, the R||G QOS abstractions appear as a collection of interfaces called the Binding Interface Base (BIB) [19]. Because the APIs in the BIB are seen as basic building blocks of a multimedia network, they are key to several important new initiatives including open signalling (see OPENSIG [25]) and multimedia integration frameworks (such as DMIF: http://drogo.cselt.it/mpeg/mpeg.htm).

We have recently announced completion of the draft specification of the BIB [1] at the OPENSIG workshop in Cambridge. The draft document and associated IDL templates have been made available to the community for comments and improvement. Also at the recent ISO MPEG meeting at Bristol, a proposal for using the BIB as an interface framework for the "DMIF Network and Media Dependent Parts" has been put forward by Nortel [3], [4]. It is expected that this work along with the rest of the MPEG-4 standards will reach international standard status by 1999.

### *G||B Interface APIs: Service Abstractions*

In contrast to the R||G interface APIs, the G-model APIs provide access to the basic resource allocation and management algorithms of the multimedia network. These include algorithms for:

1. Communication services such as connection management, routing and admission control.
2. Device management services.
3. Transport level services such transport monitoring, protocol stack management.
4. QOS mapping services.

These algorithms operate on the BIB with the goal of implementing a set of rudimentary communication services that allow the creation of simple point-to-point connectivity with guaranteed service characteristics. Collectively, these services are termed the *Broadband Kernel Services* and are likened to be networking counterparts of low-level operating system services (e.g., memory management, file system management, etc.).

In a similar manner that BIB interfaces are assembled to create broadband kernel services, broadband kernel services can themselves be assembled together to compose even higher level services. An especially useful class of such services are *Network Services* which consists of:

1. Virtual Circuit Services
2. Virtual Path Services
3. Virtual Network Services
4. Multicast Services

These services allow the construction of complex connectivity graphs in the network with associated transport and management facilities. The collection of network services at the GIIB interface is known as the Service Interface Base (SIB). Similar in concept to the BIB, the SIB defines a set of independent interworkable services that may be assembled to create yet higher consumer level services.

In the fall of 1996, we completed an implementation of the broadband kernel called **xbind** 2.0 [10]. The system supports a simplified BIB, prototypes of all broadband kernel services and a simple teleconferencing service with QOS renegotiation capabilities.

The system has been installed on a testbed of 4 ATM switches and interconnects the Columbia University distance learning program, the Columbia Video Network, to a wide area ATM testbed. In terms of platforms, **xbind** 2.0 has been ported to SunOS, Solaris, HP-UX and Windows 95/NT operating systems and Fore ASX100/200, NEC Model 5 and ATML Virata 1 switches. The system also supports multimedia devices ranging from high-end workstation based JPEG video cards to entry-level real-time MPEG-1 encoders on the PC. **xbind** 2.0, which is CORBA 2.0 compliant [23], [24], stands at roughly 30,000 lines of C++ and Java code. Access to the switch hardware has been obtained through cooperative agreements with the vendors.

## 5. THE SERVICE CREATION MODEL

In accordance with the model described in the previous section, network services are obtained by invoking distributed algorithms in the space created by the BIB objects. For an object oriented software implementation this requires modeling the service interfaces as well as providing a service creation model. Both will be described next.

### 5.1. Modeling Service Interfaces

High level services (and conceptually all services) are composed of an algorithmic component and a data component. The algorithmic component expresses the execution logic of the service instance while the data portion is an abstraction of its state. In order for services to interact with each other, several types of service interfaces are also defined. These interfaces reflect the roles that a service might play in the process of its execution. Typically these include creation, operation, management and programming.

The service creation interface is akin a constructor of a class [21]. It is the primary entry point of execution or instantiation of a service and is called by the server once the service template has been completely downloaded. The service operation interface defines the operational functionality of the service and is usually the primary interface through which services interact. The programming interface allows manipulation of the service logic to be performed while a service is in execution. Finally the service management interface allows for monitoring of service states and manipulation of service parameters. Figure 3 illustrates.

Recall from the previous section that the service creation process typically requires support for a number of middleware services. The bubble in Figure 3 represents their

invocation points embedded in the service script.

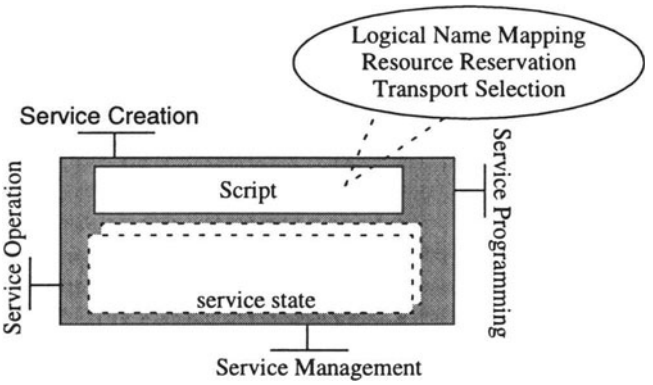


Fig. 3. Interfaces for a G- model multimedia service

Although there might exist services with little or no state, the focus of our work is on those that require state keeping since these are typically more complex and harder to scale. Service states exist in two forms:

1. Local states - which are of purely local significance
2. Distributed states - which are states associated with other services and may be geographically dispersed.

An example of a local state for a teleconference service might be the number of current users while a distributed state might be the list of all switches through which a conference connection passes through.

Because most multimedia services usually require the assistance of one or more broadband kernel services, the state space of the service is usually distributed. Conceptually, distributed states can be viewed as a general directed graph where the nodes represent constituent 'sub'-states and links represent the relation or nature of the association between nodes.

## 5.2. The Process of Service Creation: A Simple Example

The service creation process in broadband networks includes the creation of a service skeleton for an application, the mapping of the skeleton into the appropriate name and resource space, the association (or binding) to the application of a media transport protocol, the binding of the transport application to resources, creating a network service, and finally, the binding of the service management system to the network service, thereby creating a managed service.

As an example, consider a simple teleconferencing service which provides point-to-point video and audio connectivity services. When a request for a conference session

is received, a capability check must be performed at source and destination end-points to ensure that the appropriate multimedia devices exist and can support the requested media format. Next the appropriate end-to-end network resources are reserved so that a connection is established between source and destination nodes. This involves mapping the end points which may be specified in some logical names into physical nodes in the network as well as determining the route that a connection must take. Finally, transport selection and binding is performed so that both end-devices are associated with the correct network terminations.

Although the above model is realizable for simple services, the complexity of state management grows exponentially with the number of sub-services employed. As a result, complex services require additional support for state aggregation. This is covered in the next section.

## 5.3. Managing Complex Services

Our approach for making a service manageable introduces a new type of controller in the service control system: a dynamic object that is created for each new session. This object, which represents a service session (or service instance), makes itself accessible to the management system by registering with a management object, the service aggregator.

For each type of service there is a service factory which handles the requests for new service sessions and coordinates the service creation/instantiation process. For example, a service factory for unicast VCs receives requests for a new VC and contacts the necessary VC connection managers and routers in order to set up the new VC. The service factory object abstracts the specific scheme used for the service creation process.

The service instance represents the status and capabilities of an existing service session. Every service instance exports two types of interfaces. One is accessed by the service user and represents the control capabilities (status, modify, terminate) a user has once the service session is created; the other is accessed by the management system and represents the management capabilities.

The service aggregator has two purposes. First, it provides a single point of access for the service manager to monitor and control the existing service sessions of a particular service type. Second, it provides aggregated or abstracted views of the service instances and allows the manager to manipulate sets of service instances.

An important aspect of our model is that the designer of a service has several degrees of freedom when developing service management functionality. The choices include the selection of the state information and functionality of the service instances, the amount of information kept in the service aggregators and the consistency requirements for the management data in the aggregators. All these factors influence the *cost* of managing the service in terms of design complexity during the development phase and in terms of processing and computational resources needed during the operational phase.

# 6. QUALITY OF SERVICE, PERFORMANCE AND SCALING ISSUES

In the previous section we dealt with the art of service creation. We will focus now on issues related to quality of service, performance and scaling.

## 6.1. Cell Level QOS

In order to control the QOS at each contention point in the network a QOS abstraction on the object level is needed. Mapping of the states of the hardware into the software domain can be achieved with the QOS-extension of GSMP (a protocol proposed by Ipsilon Networks [22] and now an Internet RFC) that we proposed and refer to as *qGSMP* [2]. The extension provides a number of key features, including means to specify QOS constraints, select scheduling and buffer management policies, and transfer of schedulable regions [15], [16].

Apart from semantic changes to two fields associated with the connection management messages, qGSMP does not alter the original GSMP messages. Moreover, the changes to these two fields are such that qGSMP is backward compatible with GSMP version 1.0.

The extensions provided by qGSMP focus on controlling the output multiplexers and retrieving schedulable region estimates. They provide means for selecting scheduling, buffer management and schedulable region estimation algorithms, setting traffic parameters and QOS constraints, and collecting QOS-related measurements. The general architecture of a switching environment using qGSMP is available on the web (http://comet.ctr.columbia.edu/specs/specs.html).

The design of qGSMP follows the spirit of GSMP in the sense that it provides standard interfaces for development of switch-control software. By providing generic descriptions for expressing QOS-semantics, the controllability and programmability of the switch control software is significantly enhanced, allowing possibly much better resource utilization.

Our implementation of qGSMP is supported by Advanced Telecommunications Modules Limited (ATML) switching technology. We are currently also negotiating the implementation on switches built by AVIDIA Systems. We also hope to be able to install the qGSMP interface on the switches provided by Washington University under NSF support. Finding efficient real-time algorithms for estimating the schedulable region as well as integrating the complete qGSMP interface into the **xbind** broadband kernel represents a key research priority. Having access to the schedulable region will create an extraordinary opportunity for experimentation with resource partitioning algorithms that guarantee QOS. We have devised such algorithms based on game theoretic models in the past and believe that these can be readily implemented on our operational network.

## 6.2. Performance of the Service Delivery System

The service creation methodology described in section 5 focussed on the efficient

realization of services in terms of the number of states and the flexibility in accessing them. The design trade-offs that our architectural model is enabling pertains to the location and the degree of cooperation among various entities that the service creation process is based upon. Service creation can be executed at the periphery of the network, as in the case of the Internet, or in the network itself, as is the case in the Telephone and ATM Networks. In the framework of our architecture there is complete freedom in locating the objects participating in the creation process. In addition, there might be multiple providers offering services at various level of abstraction and thereby a parametrization between the 'all or nothing' capabilities of the Internet and Telephone Network is made possible.

Modern approaches for provisioning connectivity services can be found in TINA [30]. In these approaches, controllers run on general purpose distributed computing platforms, and interact through local or remote invocations. Such approaches have the advantage that they allow signaling activities to be expressed in terms of high-level operations, instead of low level mechanisms.

We have designed and developed a connection management architecture for an ATM network based on the CORBA. In this architecture, controllers are modeled as objects interacting through (remote or local) object invocations defined using the Interface Definition Language (IDL). With the objective of supporting a large number of users, it is essential that the connection management system exhibit high performance. In [11], the elements of an approach for realizing a high performance connection manager with high *call throughput* and low *call setup delay* is presented.

An efficient design of the connection manager has to take into account that the most expensive operations in a distributed environment are remote object invocations. In particular,

- the vast majority of the remote operations during connection setup have small arguments,
- remote calls contribute the bulk of the latency in call processing,
- most computations are executed in the communication layer.

In order to design a high-performance connection management system, the following rules are followed:

- *Parallelization of the Object Call Request Execution* - design the system to run with a maximum amount of parallelization so that processors can be kept busy as much as possible,
- *Caching of Network States* - minimize the number of remote procedure calls through aggressive caching of network states (network resources cache include name space and bandwidth),
- *Aggregate Access to Remote Objects* - aggregate access requests to remote objects as much as possible.

A first version of the connection manager described has been implemented. Running on a cluster of SUN Sparc 10s, initial results indicate that the system can support a throughput of about 100 calls/sec (or 360,000 calls/hour) with average latency of 200ms. The intrinsic 16ms call set up time is substantially shorter then the latency data published by the ATM Forum [5].

The caching principle proposed leads to interesting performance questions. How much resource should be cached or, equivalently, how should resources be partitioned? Resources here refer to both name space, bandwidth and buffer space. There are essentially two main categories of algorithms that can be designed here. A competitive scheme among connection managers or a cooperative scheme that can be managed by a separate entity that is independent of the connection managers or switch controllers. (A competitive scheme is distributed and works with local information. A cooperative scheme tends to be centralized and works with global information.)

We believe that understanding the fundamentals of allocating the resources involved, in particular the distributed allocation of the VCI space, will allow us the tuning of the key parameters resulting in a further 2 to 3 times performance improvement. This level of performance will bring us in the range of the current STP processors. Note that the other option for improving the performance of the service creation process is possible via the engineering of the RPC. However, this is not an option to us as we do not have access to the source code.

## 6.3. Scaling Issues

The implementation of the service architecture on the **xbind** platform will support experimentation with small networks. Scaling our real-time environment is fundamentally limited by costs, however. To experiment with the scaling properties of our architecture, we have realized a high-performance emulation platform, based on a parallel simulator. We intend to use this platform to study scaling aspects of the architecture and of the various telecommunication services.

The platform we have built over the last two years allows us to closely approximate the functional and dynamic behavior of network control systems on the call level [12], [13], [14]. By providing support for real-time visualization and interactive emulation, it can be used to study telecommunications systems in various scenarios, such as different load patterns, network sizes and management operations. The current implementation runs on the IBM SP2 parallel processor at the Cornell Theory Center, which is connected to a graphics workstation in our laboratory at Columbia University via an ATM link.

The emulated system and emulation support modules consist of a set of objects that communicate by exchanging messages, using functions provided by the simulation kernel. The emulated system module represents the prototype system under evaluation. Generally, each controller is implemented as a C++ object. Objects that interact with the parallel simulation kernel require minimal knowledge about the kernel--mainly how to send and receive messages. Therefore, the design of the emulated system follows the same rules as the design of controllers that run on a real network platform.

The major difference is in how the interaction among controllers is realized. In the emulated system, interaction is performed by the simulation kernel. In a broadband network environment, the exchange of messages is provided by a signalling system.

The simulation kernel controls the execution of these objects and ensures that messages are processed in the correct order. It is realized as a parallel discrete event simulation (PDES) system, using a window-based synchronization protocol. In order to support real-time visualization and interactive control of the emulated system, the kernel controls the progression of the simulation time, constraining it by the progression of the processor time. The module for real-time visualization and interactive control contains a graphical interface which provides 3-D visual abstractions of the system state.

Both the emulated system and the simulation kernel (also coded in C++) run on an SP2 parallel processor located at the Cornell Theory Center (CTC) in Ithaca, New York. The real-time visualization and interactive control module resides on an SGI Indigo2 workstation at Columbia University. It is written using Open Inventor, a 3D graphics tool kit based on Open GL, and runs as a UNIX process that communicates with the emulation support module via *UNIX System V* shared memory. The emulation support module is distributed on the two machines. These machines communicate through NYNET, an ATM network that connects several research laboratories in New York State. For more details as well as for download of the source code, the reader is directed to the URL: http://comet.ctr.columbia.edu/software.

## 7. IMPLEMENTATION ISSUES

Object-oriented methodologies and technologies are essential to us for developing service control and management systems. The basic elements of an object model, namely, concurrent objects interacting via messages, can be used to model controllers and their interactions in a distributed system. Such a model can be executed in a PDES (parallel discrete event simulation) environment (such as our emulation platform), as well as on a CORBA-based platform (such as **xbind**).

Our requirement is that the implementation design of the service control and management systems can be done in such a way that their code runs on both platforms. Also, the GUI enabling service management functionality by an operator must run on both platforms.

The servers or controllers represent perhaps the most complex implementation challenge in the model. The primary difficulty comes from the inherent flexibility a server must support - the ability to load any service script, instantiate, externalize transport and store its code and state and resolve all references to calls for broadband kernel services. The major implementation language being used is Java because of the easy availability of its virtual machine and class loader facility. The API stubs to the broadband kernel services would be implemented to make CORBA calls (or alternatively Java RMI calls) but would themselves be Java classes. Several Java based ORBs and interworking products provide this capability. Moreover, the Sunsoft JDK 1.1 release includes facilities for serializing [27] and externalizing Java object graphs.

These facilities will be used to implement the basic facilities component in the servers. It is anticipated that the B-model services will be readily implementable in Java using the Java beans [28] component framework. This is because these services typically do not have to deal with hardware or I/O which are usually lacking in Java APIs.

Consequently, there is a wide range of possible designs for building management systems for a particular service, from light-weight, low-cost systems with minimal functionality to heavy, high-cost systems with rich functionality. More precisely, we believe that a trade-off analysis must be made, balancing various factors in order to achieve the best combination between low cost, good scalability and rich functionality of the management system for a specific service and a particular environment. There are strong reasons to engineer configurable systems which can be customized at the time of service deployment. Moreover, it may be necessary to allow for dynamic reconfiguration of a system during run-time, in response to changing needs and the availability of resources allocated to management tasks.

## 8. CONCLUDING REMARKS

The major theme of this paper is the realization of open programmable telecommunication networks. Specifically we described:

- a new architectural foundation for the creation, deployment and management of future telecommunications services based on the paradigm of open programmable networking;
- open Application Programming Interfaces (APIs) for service creation, quality of service control and the joint allocation of computing and communication resources;
- the need to investigate scaling issues through the emulation of complex service scenarios arising in large scale broadband networks;
- a methodology for evaluating the performance of the proposed service delivery system; and
- recommendations for a standard set of open APIs and advanced broadband kernel technology to the research community pursuing open programmable networking.

Due to space limitations, we have not discussed other key issues regarding the design and implementation of a service architecture for broadband networks. In particular, the extension of the service architecture to wireless environments, problems of security, problems of software reliability and more generally problems of verification and testing of the proposed architecture have not been mentioned at all.

Concluding, we believe that building programmable telecommunication networks is one of the key research challenges that faces the networking community as we move towards the new millennium. To address this challenge we have initiated a number of new projects and international forums (OPENSIG and OPENARCH, http:// comet.ctr.columbia.edu/openarch) to promote the ideas which drive our research. We presented a number of research topics associated with service creation, quality of

service, performance and scaling. Our work is backed by two software platforms currently in various phases of development. The **xbind** 1.0 platform is currently being used by a number of universities and industrial research laboratories worldwide. (See, among others, the list of participants in [25]).

## 9. ACKNOWLEDGMENTS

The author would like to thank Koon-Seng Lim and the other members of the COMET Group, for extensive discussions on these and related issues. The research presented here was supported by the Air Force Rome Laboratory, CSELT, HP, ISS, NEC, NTT, NYNEX, ORL, Sun Microsystems and Toshiba.

## 10. REFERENCES

[1] C. M. Adam, M.C. Chan, A. A. Lazar, J.-F. Huard and K. S. Lim, "The Binding Interface Base Specification Revision 2.0", OPENSIG Workshop on Open Signalling for ATM, Internet and Mobile Networks, The Moller Centre, Cambridge, UK, April 17-18, 1997; also *CTR Technical Report #475-97-09*, Center for Telecommunications Research, Columbia University, New York, April 16, 1997, available under URL: http://comet.ctr.columbia.edu/publications/standard_contributions/BIB.html.

[2] C. M. Adam, A. A. Lazar and M. Nandikesan, "QOS Extension to GSMP", OPENSIG Workshop on Open Signalling for ATM, Internet and Mobile Networks, The Moller Centre, Cambridge, UK, April 17-18, 1997; *CTR Technical Report #471-97-05*, Center for Telecommunications Research, Columbia University, New York, April 16, 1997, available under URL http://comet.ctr.columbia.edu/xbind/qGSMP.

[3] V. Balabanian, "Binding Interface Base Operations", MPEG97/M2015, ISO/IEC JTC1/SC29/WG11.

[4] V. Balabanian and F. Cuervo, "Proposal of an Interface Framework for Control of Bindings between DMIF Network and Media Dependent Parts," MPEG97/M2014, ISO/IEC JTC1/SC29/WG11.

[5] A. Battou, "Connection Establishment Latency: Measured Results," ATM Forum Document, ATM_Forum/96-1472, October 1996.

[6] H. Berndt, R. Minerva, "Service Architecture version 2.0", TINA Baseline Document, TB_MDC.012_2.0_94, TINAC, March 1995.

[7] R. Braden, D. Clark and S. Shenker, "Integrated Services in the Internet Architecture: An Overview", IETF RFC 1633, June 1994.

[8] CCITT Recommendation I.312/Q.1201, Principles of Intelligent Network Architecture, CCITT, April 1992.

[9] CCITT Recommendation Q.700, Introduction to CCITT Signalling System No. 7, Fascicle V1.7, Nov. 1988.

[10] M. C. Chan, J.-F. Huard A. A. Lazar and K. S. Lim, "On Realizing a Broad-

band Kernel for Multimedia Networks", In Proceedings of the Third COST 237 Workshop on Multimedia Telecommunications and Applications, Barcelona, Spain, November 25-27, 1996.

[11] M. C. Chan, A. A. Lazar, "Connection Services on the **xbind** Broadband Kernel," OPENSIG Workshop on Open Signalling for ATM, Internet and Mobile Networks, The Moller Centre, Cambridge, UK, April 17-18, 1997.

[12] M.C. Chan, G. Pacifici and R.Stadler,"A Platform for Real-Time Visualization and Interactive Simulation of Large Multimedia Networks," Fourth IEEE International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS 96), (Honolulu, Hawaii), April 1996.

[13] M.C. Chan, G. Pacifici and R. Stadler, "Managing Multimedia Network Services," Journal of Network and Systems Management, 1997, to appear.

[14] M.C. Chan, G. Pacifici and R. Stadler, "Real-Time Emulation and Visualization of Large Multimedia Networks," in Proceedings of the ACM Multimedia, Demonstrations Program, (San Francisco, CA), November 1995.

[15] J. M. Hyman, A.A. Lazar and G. Pacifici, "Real-Time Scheduling with Quality of Service Constraints", IEEE Journal on Selected Areas in Communications, Vol. 9, September 1991, pp. 1052-1063.

[16] J. M. Hyman, A.A. Lazar and G. Pacifici, "A Separation Principle between Scheduling and Admission Control for Broadband Switching", IEEE Journal on Selected Areas in Communications, Vol. 11, May 1993, pp. 605-616.

[17] Lazar, A.A., "A Real-Time Control, Management and Information Transport Architecture for Broadband Networks", Proceedings of the 1992 International Zurich Seminar on Digital Communications, March 16-19, 1992, pp. 281-296.

[18] Lazar, A.A., "Challenges in Multimedia Networking", Proceedings of the International Hi-Tech Forum, Osaka `94, Osaka, Japan, February 24-25, 1994, pp. 24-33.

[19] Lazar, A.A., Bhonsle, S. and Lim, K.S., "A Binding Architecture for Multimedia Networks", *Journal of Parallel and Distributed Computing*, Vol. 30, No. 2, Nov. 1995, pp. 204-216.

[20] Lazar, A.A., Lim, K.S. and Marconcini, F., "Realizing a Foundation for Programmability of ATM Networks with the Binding Architecture", *IEEE Journal of Selected Areas in Communications*, Special Issue on Distributed Multimedia Systems, Vol. 14, No. 7, September 1996.

[21] Lazar, A.A., Lim, K.-S., "Programmability and Service Creation for Multimedia Networks,", Workshop on Multimedia and Collaborative Environments, Fifth IEEE International Symposium on High Performance Distributed Computing (HPDC-5), Syracuse, NY, August 6-9, 1996, pp. 217-223.

[22] P. Newman, R. Hinden, E. Hoffman, F. C. Liaw, T. Lyon, and G. Minshall, "General Switch Management Protocol Specification", Ipsilon Networks, Inc., Palo Alto, CA, March 1996.

[23] Object Management Group (OMG), The Common Object Request Broker: Architecture and Specification, Rev. 1.2, Dec. 1993.

[24] OMG, The Common Object Request Broker: Architecture and Specification, Revision 1.1, OMG Document No. 91.12.1.

[25] OPENSIG, http://comet.ctr.columbia.edu/opensig

[26] Project **xbind**, http://comet.ctr.columbia.edu/xbind.

[27] Sun Microsystems Inc., Java Object Serialization Specification - Release 1.2, Dec. 1996, available at http://chatsubo.javasoft.com/current/doc/serial-spec/serial-spec.ps

[28] Sun Microsystems Inc., JavaBeans 1.0 API Specification, Dec. 1996, available at http://splash.javasoft.com/beans/beans.100A.ps

[29] Sun Microsystems Inc., "The Java Language Environment", *White Paper*, Oct. 1995.

[30] TINA-C, Service Architecture Version 2.0, Document No. TB_MDC.012_2.0_94, March 1995.

[31] K. J. Willets and E. K. Adams, "Managing a Broadband Environment: You Can't Buck the Market", IEEE Comm. Mag., Dec. 1996.