94.588 Course Project

Project Title: Comparing Schemes for Network Programmability

Presented to : Professor G. Bochmann

By

Li Tang

(Student Number: 252317)

April 25, 2000

***Abstract*:**

The ability to quickly create and provide new and customized services in response to market demands will be the key factor in determining the success of the future service providers. In an effort to separate the tight coupling between network operation and services provision commonly found in the traditional networks, programmable network community pursues an approach of opening up the network to allow the deployment of third party control architectures, customized services creation and management. This project will offer a review on the programmable network research activities. Most active programmable network projects will be compared in term of their architectural model and level of programmability. The key characteristics enabling network programmability will be summarized in order to suggest a general infrastructure for programmable networks. In addition, alternative methodologies and architectures are also discussed before the project concludes with thoughts on future research and directions in programmable networks.

Table of Contents

1. Introduction

With the emergence of distributed multimedia applications over the Internet, networks nowadays are facing increasing demands for fast network service creation, customized service provision and various degree of quality of service assurance. At the same time, networks also need to evolve rapidly in response to the pace of technology advancement. Unfortunately, today's network failed to satisfy all these requirements, and its deficiency has been widely identified. [Smith et al. 1999] One of the most significant changes mandated by the 1996 Telecommunication Act require that network operators provably demonstrate capability to support third party service provisioning. [Tennenhouse et al.1997] These and many other merging trends urge the legacy network to become an open programmable environment. In an attempt to open up traditional network to allow third-party service providers to create and enhance services quickly, network programmability is added to the legacy networks. Currently, network programmability is not only an attractive feature for network operators and general users, but also a most desired architectural model for Internet service providers. There are two similar but distinctive approaches towards network programmability, namely programmable network and active network. Their key characteristics and general architectural model will be discussed in detail in this paper. Moreover, some outstanding projects will be reviewed in an effort to best demonstrate the tremendous flexibility network programmability offers.

1.1 Network programmability

Before we explore the advantages network programmability might give, let us first try to understand the concept of network programmability and theoretical support behind it. What is network programmability? It can be informally defined as the methods and ability to create and deploy tools, mechanisms and algorithms, which allow network architects to program or build new network architecture that is virtual to the physical structure. On this virtual programmable platform, network architecture and customized services are constructed dynamically by either network operators or third-party service providers through application programming interfaces to the network hardware. The key to network programmability is to break the close coupling between hardware and its controlling software. Along this vein, there are two

schools of thoughts on adding programmability to legacy network, namely programmable network approach and active network approach.

1.2 Programmable networks

Programmable network is the approach proposed by the OPENSIG community. As their name suggested, OPENSIG proposed that traditional network should open up their control and management through standardized programming interfaces. These programming interfaces allow applications and middleware to manipulate low-level network resources to construct and manage services. The objective of opening up network signaling and control is to facilitate service creation, allow customized service provision and enhance control and monitoring. All these lead to a much better support for quality of service. Programmable network is comparatively conservative in that the format of existing packet/cell format is kept. Compared to active network, packets and network switches are still behaving in the same way as that in legacy networks.

1.3 Active networks

On the other hand, active network go one step further than programmable network in the way of creating services and enhancing existing services through active packets, or capsules, as they are mostly called. Active packets are special packets that contain not only data as normal packets, but also executable programs that can be executed at visiting nodes to process the data contained in the packets. By this means, new or customized processing routines can be packetized in capsules and delivered to active nodes. Thus realize new service creation.

Proposed mainly by Active Network Research group in MIT, active network is regarded much more dynamic that programmable network as service provisioning can be set up specific enough for just one packet. With active packets, different specification of quality of service parameters can be made for particular data transmission. With active nodes, network itself contains enough intelligence to respond to various circumstances and perform accordingly. Many models of active network have been suggested. In capsule model, one extreme of active network, every packet passing within a network contains executable

code, and all nodes are able to execute the program residing in the visiting active packets. A less vigorous model, a switch-like model, is that active packets will be triggered in response to certain network conditions. Only few internal nodes have the capability of handling active packets. In either case, active packets become a powerful vehicle for service creation, transport control, network management and integration. Programmable network free us from the tight coupling between network hardware and controlling software, while active network completely changes our mindset on networking.

2. Motivation for network programmability

Whenever any new thing is ushered in with big claims, the most common response people have is to ask "why?" Today's networks are the accumulation of decades of engineering and networking experience, and they have been functioning admirably. Why do we need to change the traditional network architectures that have been fairly stable to a drastically different approach? There are many reasons behind the promotion and popularity of network programmability. This idea merged from discussions on the future direction of network systems among the broad Defense Advanced Research Projects Agency (DARPA) research community between 1994 and 1995. "Several problems with today's networks were identified: the difficulty of integrating net technologies and standards into the shared network infrastructure, poor performance due to redundant operations at several protocol layers, and difficulty accommodating new service in the existing architectural model." [Tennenhouse et al. 1997]

2.1 Deficiency of traditional network

First of all, today's telecommunication network is based on the mainframe architecture that was developed 30 or more years ago when there were far fewer users, less demands, and much simpler data transmissions. The legacy architecture assumes that the network nodes have very limited computational capabilities. In traditional networks, few powerful processors distributed within the network taking the full responsibility of service provisioning. The internal nodes are assumed to be dummy entities that dutifully perform routing based on packets' headers. On the other hand, network operators assume comprehensive control over the design, introduction, provision and management of service available to their users, because they

own the network, including both the hardware and software. This monolithic view of service provisioning process results in the close nature of network control. This intimate coupling between hardware and software makes third-party's involvement in service provision and programming very difficult, if not possible. [Lazar 1997] The success and of Internet and the popularity of all the different information transmitted on it, notably multimedia application demands openness and extensibility. Away form the monolithic view, network programmability allows networks operated by different administrations be interconnected. With network programmability added, third party service providers are able to access the system resources and participate in programming the services offered by the network.

Another strong motivation for network programmability is to minimize the standardization process for new protocols required to develop and implement end-to-end services. As commonly practiced, network protocols must be agreed upon through industry-wide standardization. "Network providers must then wait for vendor implementations and customer demand, and gradually deploy new equipment in their networks. Finally subscribers see new services offered." [Chen and Jackson, 1998] This process can take a few years. By reducing the amount of global agreement needed, both research and commercial interests will be best served. As previously described of how active network realize services, it is very obvious to see that new algorithms and protocols can be readily injected into the network without global agreement on the standards. As long as the packet knows how to process its data, the packet can take care of itself.

To a large extent, Internet has been a single-service network in which all packets are processed equally. However, the growing demands or multimedia applications deeply impact people's expectations of networks and the service they provide. Besides the basic transport and control, multimedia applications also require real-time, low-delay, high bandwidth and multicasting services. However, traditional switches and routers are usually limited in their quality of service support, if any. There is little or no class differentiation and filter service in traditional network. Network programmability offers the flexibility to customize the network by allowing network operators to reconstruct the network architecture.

2.2 Lead user's pull

Increasing attention has been drawn to the deficiency mentioned above. Changes to the traditional network model have already been made in many aspects, and application specific processing and computation at internal nodes has taken place. Web proxy server and firewall are good example of user's pull on distributed and autonomous processing of the packets flowing through nodes. Going one step further than those leading applications, the advocates of network programmability suggest all the network elements should offer standardized application programming interfaces (API), which gives access to their services and resources to applications.

2.3 Technology's push

On top of all the user's push, network programmability will not be a reality without an equally strong technology pull. With all current available technologies, the concept and practice of network programmability is just a nature outcome of the advancement of today's technology. First of all, not very long ago, networks still consist of fewer high capacity processing node and whole bunch of terminals which has very limited computational capability of their own. However, it is very common to find considerably powerful computational power and sophisticated operating systems within network nodes in today's distributed network. Network programmability takes advantage of the processing capability within each network element. Inside of insisting on equal treatment from each network element on packets passing through them, network programmability allows each node to decide what to do with the packets using their own resources and intelligence. Secondly, mobile code and mobile agent technology, which realizes program mobility and remote execution, gives rise to the idea of network programmability. This is especially true in the case of network in which capsules are mobile codes. Thirdly, object-oriented approach towards software engineering and network architecture significantly impacts our way of developing software and construct networks. With function encapsulation, software components are characterized and modularized as interacting objects to facilitate software and design reusability. For instance, Xbind, one typical example of programmable network, modularizes various components in transport protocols and builds them as engines for specific purposes that can be bind later for particular type of services.

3. Advantage of network programmability

Understanding the deficiency of traditional network and the push from available technology, we then might to ask the question whether programmable network and active network will satisfy the increasing demands from both the network users and service providers. Network elements such as switches and routers have been programmable for a long time as there are always some low-level programs controlling the functionality they provide. However the programmability is totally different from the concept we are introducing here, which is far more dynamic than the programmability in legacy networks. What so new about programmable and active network is that the perpotary programmable network entities have never been able to support the dynamic, customizable and multi-client way of control. Resource allocation, protocol processing and information processing have never been so open to a range of client control from parities other than network owner. [Suzuki M. et al. 1998] As we can see later on, the fundamental philosophy behind programmable and active networks is the separation between hardware and their control. This separation promotes modularity, and therefore reusability of network components. Most importantly, the opening up of network hardware control makes it possible for third-party to setup, program, alter and enhance their service regardless of the physical architecture of network. Needless to say, this capability will be of most attraction to both Internet users and service providers. This separation between signaling business and transport business creates a vast and active service market from which Internet users will benefit the most.

4. Architectural model of adding programmability into network

After we understand the limitations of existing networks and the drive for adding programmability into legacy network, we need to look at the fundamentals that make network more programmable. "There is growing consensus that these network fundamentals are strongly associated with the deployment of new network programming environment, possibly based on network wide operating system support, that explicitly recognized service creating, deployment and management in the network infrastructure." [Campbell A et al. 1998] A programmable network is significantly different from traditional networking in that the network itself can be programmed by using a minimal set of application programming interfaces

from which network operators or service providers can optimally compose an infinite spectrum o higher level services. This is achieved by introducing computation into the existing network environment.

4.1 Introduction of computation dimension into network

In figure one, the Internet reference model consisting of application layer, transport layer, network layer and link layer is showed. These key layers can be visualized as one dimension of the networks. The other dimension is made of the functions that networks are able to perform. As we all know, there are mechanisms for transport, control and management in each layer in this model, which give vertical support along the data path in both internet connection and telecommunication networks. To just name few, RSVP and RTCP are two good examples of resource reservation and control protocols. On the other hand, there is SNMP and CMIP for network management over the transport environment. These functional supports through different layers constitute the second dimension of the network.

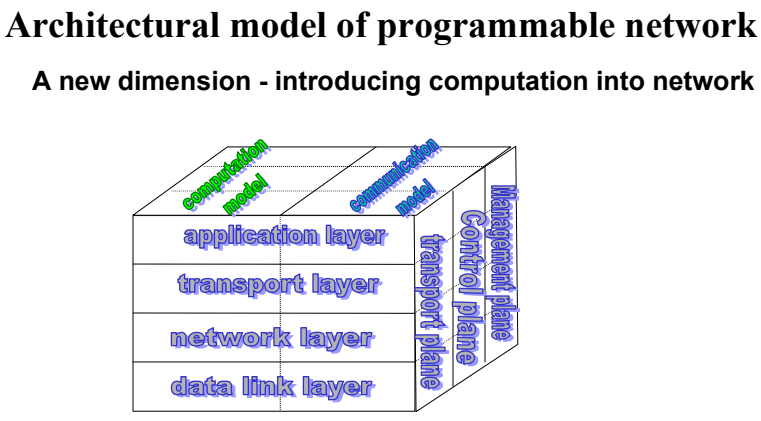

Figure one: Communication and computation model of programmable network

The philosophy behind adding network programmability is the introduction of the third dimension - computation into the existing network. This new dimension is what distinguishes programmable and active

network form the traditional network, and the contribution of this new approach. "The computation model provides programmable support across the transport, control and management planes, allowing a network architect to program individual layers (viz. Application, transport, network and link layers) in these planes." [Campbell A. et al. 1998] The introduction of computation recognizes the power and advantage processing within the network can lead us to.

4.2 Architectural model of programmable network (IEEE P1520 Standards)

Another view of the computational model can be described as in figure 2. This is the architectural model proposed by OPENSIG community and recommended and published as the IEEE P1520 Standards for Programmable Network Interfaces in 1998. [Biswas J. et al. 1998] The model consists of five well-defined levels, between which there is a standard interface. From top level, which is most obvious to the user, the five levels are End-user application level, Value-added services level, Network generic service level, virtual network device level and the last one physical element level which is most close to the hardware of the network. P1520 is still a developing reference model for opening up the closed network control and fixed algorithms specific to the hardware underneath. The objective of this model is to separate end-user application semantics from hardware and low-level software support. The functionality of each level and its required interfaces within the model will be discussed in great detail in the next section.

Starting from the bottom, we can understand the physical element level as the actual network elements. Depending on the type of networks we are focusing on, this level includes elements such as ATM switches, IP routers or circus-switches elements. This level is the abstraction of the network hardware. Virtual network device level, the second level above physical element level, is regarded as a collection of logical representation of the certain state variables of the objects in the first level. It is called virtual as it is the logical abstractions representing the access to the resources in the physical element level below, such as routing table, processing queue and transmission channels. Node Operating system (NodeOS) as in ANTS project and Node kernel as in SmartPacket projects are two very good examples of the realization of this level. Like a local operating system, this level offers low-level software handlers for local resource access, transport control, connection control etc.

Basic services and core functions of the network are captured in the network generic services level. As the name suggested, this level is considered generic as it contains core network algorithms and the fundamental functions of the network. For example, algorithms for virtual path/virtual connection, routing, service scheduling, resource reservation and policy monitoring and control are among the most commonly found components in this level. As they provide the basic services of the network, this level will be very similar from network to network. The interface between network generic services level and virtual network device level consists of primitives for the purpose of directly access and manipulate local network resources. Access to the file system, query routing table and schedule data transmission are few examples of the primitives offered in this interface. In contrast of the traditional proprietary approach as most commonly found in legacy networks, this interface exposes the access and control over the hardware, which allows third-party service providers to manipulate the state and resource of internal nodes, thus compose new services or deploy customized algorithms easily.

At the value-added service level, the entities are end-to-end algorithms that add value to services provided by the third and lower-levels by means of user-oriented features and capabilities, such as real-time stream management, synchronization of multimedia streams, and other capabilities beneficial to value-added service providers and end users. The interface between this level and the level below realizes the separation between software abstraction and implementation. This interface allows the upper level establish network connections and requests for transport services without knowing the real setup and signaling procedures of the tasks. In other words, all the details of hardware and procedure implementation are hidden by this interface. This devoice between interface and implementation offers great benefit; as long as the underneath network provides this interface, third-party service providers are able to access the basic network resources to enhance network services. When active capsules arrive at the network element, the programs contained in the active packets start executing by calling functions offered in this interface to complete the tasks.

Various user applications are contained in the end-user application level which is the top one in the reference level. User orientated applications such as IP telephony, telephone conference, video on demand and one line shopping application are typical examples of this level. The revolutionary concept of P1520 reference model lies in the interface between this level and value-added-services level. Like a toolkit, this interface provides a rich set of application programming interfaces that can be used by third-party services providers and network operators to compose and write new end-user applications. For example, in a tele-conference example, this interface will provide facilities to negotiate quality of service parameters, mechanisms to monitor quality of video streams and notification procedures when quality of service violations are detected. All in one words, third-party service providers and network administrators are able to program the network through this interface.

Architectural model of programmable network

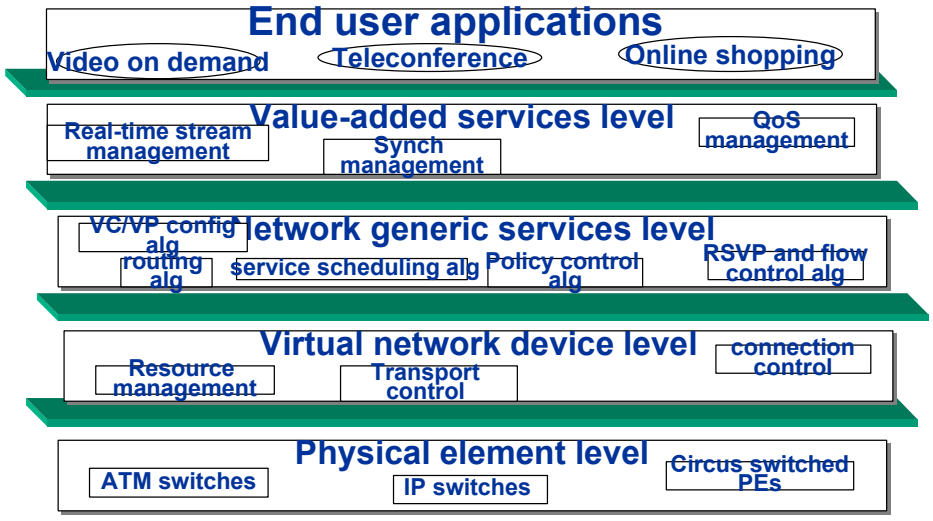

Figure Two: Architectural model of Programmable network (IEEE P1520)

Many projects on programmable network and active networks will be reviewed in the later section. As we can observe later on, some models used in the projects can be clearly mapped into this reference model,

while others are not so obvious. However, the fundamental of all the projects that will be reviewed is that they somehow offer an interface to network resources that used to be highly restricted to only the hardware owners. This platform hides the diversity of the network so new network architecture and services can be built and programmed easily in a heterogeneous network environment. Advantages of this network transparency can be easily observed in the following projects.

5. Comparing schemes of programmable networks

After discussing on the generalized model for adding programmability into traditional network, I would like to give an overview on a number of research projects in which either programmable network or active networks. Not meant to be exhaustive, this survey is to review the unique contribution of some significant research projects in an effort to demonstrate the benefit and flexibility network programmability offers in a variety of network service domains. Campbell et al. have suggested a very comprehensive comparison framework under which each research project can be examined. I would like to use their framework to make comparison among research projects in active and programmable networks. Following is the key characteristics that I will compare each project on;

- Application domain, for which area of network services the project is developed. For example, Xbind project developed in University of Columbia modularized the functions of transport level, and regards data transmission process as doing competitive business in an open market at the transport level. [Lazar. A. 1998] On the other hand, active packets are used in Smartpackets project to collect and process data for the purpose of network management and performance monitoring. Design and architectural structure of the project can vary considerably because of the targeted application domains in focus. A wide spectrum of application domain is covered by the projects reviewed from Internet multimedia software composition, to quality of service management for real time application, to network fault management, to service enhancement.

- Architectural model, which differs from project to project, thus shapes the functionality and performance of the network will be discussed. Using IEEE P1520 standard for programmable

interfaces, the mapping between the model adopted by each project and the IEEE P1520 standards for programmable interfaces will be discussed in detail with special attention on how each project realizes the separation between hardware and control, and the provisioning of programming APIs.

- Network technology, "which implicitly limits the programmability that can be delivered to higher levels." [Campbell et al. 1998] The characteristics and nature of the topology and technology used in the network deeply impact the degree of programmability a system can offer. IP based networks like the Internet are known for their scalability and resilience to partial failures. Their reliance on data gram forwarding on a hop-by-hop basis makes them ideal for the transport of short control messages with little or no call holding times. Connection-oriented broadband networks such as ATM permit a much higher degree of predictability because of its inherent resource partitioning capability. Therefore, ATM networks are highly suited for transport of streams with quality of services requirements. [Lazar 1997] There will be little application value if we fail to understand the nature of the network technology on which the project is carried out.

- Level of programmability of each project will also be compared in terms of the granularity they offer to introduce and compose new services into the network infrastructure. The technology such as mobile agent and mobile code technology, that each project deploys to achieve programmability is highlighted and the programming languages used are briefed through contrast the implementation details. Software engineering strategies and a number of other networking methodologies such as object-orientated modeling also directly influence level programmability, and therefore will be examined in each project.

- Flexibility of the project which indicates how easy the network subsystems can be adapted to a wide variety of tasks. As one of the main objective of adding network programmability is to improve the network operated by different administrative entities to be interconnected, thus minimize the standardization required for end to end services, the mechanisms that support scaling and interoperability are the focus of this section.

- Security issues such as access control, application authentication and authorization, resource reservation and management are of greatest concerns in programmable and active network research community. As users are now able to compose new or customized services using application programming interfaces offered by the programmable network, or dispatch active packets to manipulate data and state variable in visited routers and switches in active network, more and more attention is given to the mechanisms to ensure overall network reliability and data security. How each project is handling this issue at different levels are evaluated.

Some significant contribution from each project to the understanding towards a dynamically programmable network is also highlighted.

5.1 Xbind – Programmable multimedia networks

Application domain

Developed in the Comet research group in Columbia University, Xbind is a research project exploring the architecture, management and behavior of programmable network environment. The programmable environment is most often referred in their publications as a broadband kernel. [Lazar 1997], [Huard F et al 1998]. The term "kernel" is deliberately used "to draw a parallel between Xbind's role as resource allocator and extended machine and that of a typical operating system. The broadband kernel behaves as a resource allocator in that it arbitrates between various parities conflicting requests for system resources. It functions like an extended machine in providing a simplified way to access fundamental system services by abstracting away the operational complexities of service provision. " [Campbell et al 1997] The objective of the project is to develop an open operating platform which give comprehensive support to service creation, new technology deployment, quality of service management and control. The targeted application domain is service creation for multimedia telecommunications.

Architectural model

The architectural model of Xbind project considerably resembles the reference model proposed in IEEE P1520 as discussed in earlier section. Their networking design model – Extended Reference Model (XRM) is illustrated in figure 3. The physical element level and virtual network device level in the IEEE P1520 reference model combining contribute to the broadband network in XRM model, which consists of ATM switches, communication equipments and multimedia end-devices together with their logical representations. Again, the functionality and entities in network generic services level and value-added services level in P1520 standards make up the second layer in XRM – the multimedia programming model.

Xbind research community strongly believe that the flexibility of service creation lies in the well defined set of APIs. There are two set of APIs offered between the three layers. The first set – the one between broadband network and multimedia network abstract the states of local multimedia resources such as devices, switches, links and their processing and transmitting capacities. These resources are abstracted as autonomous software objects with open interfaces which allow them to be combined, plugged in and reassembled to form a complete services. These interfaces – called Binding Interface Base (BIB) are regarded as the most basic building blocks of the broadband network.

The second set of APIs lies between multimedia network layer and the top layer. This set of APIs are collection of abstractions of end to end service algorithms including communication service such as connection setup, routing, transport service such as transport stack management, quality of service mapping service such as admission control and resource allocation. Similar as BIB, these service abstractions – called Broadband Network Services (BNS) also provide open interfaces for service creation, management, programming and operation through which application clients are able to control and manipulate the service. These BNS are independent entities that can be bind together to compose high-level services according to the requirements of the consumer applications.

Network technology

The broad bandwidth and connection-oriented nature of ATM network offers tremendous advantage for stream transportation of multimedia communication. However, because of ATM's sophisticated signaling components, it is very complex to use ATM for media stream transport. Considering this shortcoming of TAM technology, Xbind uses IP for inter-object for inter-object communication (signaling) while ATM for media stream transport.

Level of programmability

Xbind network architecture offers a very dynamic programmable environment at transport level. As both the low-level network resources and high-level services are abstracted into programmable entities with open interfaces for controlling and binding, it is easy to visualize a network service as a set of independent but interconnected logical entities interacting with each other to achieve one tasks. Depending on the client request, services are easily customized through the extended interfaces they support. Quality of services parameters are easily modified and monitored. For example, the process of service creation in Xbind can be summarized into the following steps:

- When service request is received, create a service skeleton indicating the source and destination end points.
- Map the skeleton into the appropriate name and resource space, i.e. reserve resources for this application both on the end nodes, and the internal routing entities.
- Bind appropriate transport protocol to the skeleton.
- Bind the allocated resources to the application.
- Bind application to the service management system to make it a managed service.

[Campbell et al. 1997]

Xbind is based on the industry standard distributed object-oriented platform – CORBA. All service controllers are modeled as objects interacting through object invocations defined using the Interface Definition Language (IDL) in CORBA. The APIs are all developed as CORBA interfaces abstracting the lower-level resources. Above analyses demonstrated the object-oriented design of the Xbind architecture.

Security and performance

There is little mentioning of network security in Xbind project. Chan et al. have evaluated performance issues using Xbind architecture with techniques of application caching, aggregation of connection requests and parallel connection setup. [Chan et al. 1999] Their measurement statistics show that Xbind implementation of signaling system outperforms other traditional system by a very significant margin. [Chan et al. 1999]

Project significance

As one the earliest project exploring the possibility and architectural issues of programmability networks, Xbind provides a sophisticated model for open programming network environment. Its influence can be observed from the similarity between the standardized programmable interface model specified in IEEE P1520 and the XEM used in the project. Together with the overall infrastructure Xbind suggested on abstracting network services and resource as bindable building blocks that can be reconstructed in response to users' request and network circumstances, Xbind also offers a comprehensive application developing toolkit to compose network services. Quite a few Internet service applications have been built using the toolkit, including an Internet teleconference application, management application for multimedia multicasting on the Internet, [Aurrecoechea, 1998] video conferencing control application [Lazar et al, 1996] to just name a few. All in one word, as a pioneer in the area of programmable network, Xbind and its research community not only offer a well-defined infrastructure to open the control of network by abstracting network resources and services, but also develop a powerful toolkit including a set of methodologies and techniques to modulate service creation process.

5.2 ANTS - Active Network Transport Systems

Application domain

Active Network Transport System, as often referred to as ANTS, is a very good example of active network. Developed in MIT, ANTS provides a node runtime and protocol programming model that allows network users to select and customize the processing of their capsules, thus tailor or create network services to best

suit their individual needs. Aiming at the same objective, ANTS takes a step further than Xbind in that the format of packets passing in the network and the behavior of internal nodes are changed to add extensibility and flexibility of the network. As mentioned before, service routines are injected into the network by the delivery of active packets – capsules, as they are mostly called. Upon arrival at the active node, these

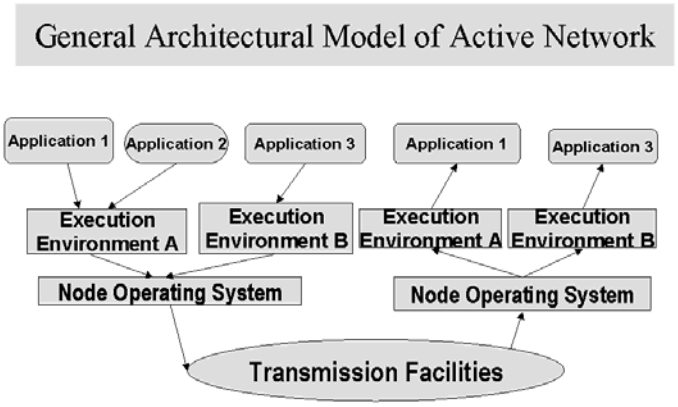

Figure 3: Architectural model for active network (ANTS)

routines executables – often referred to as protocols, will be cached into the execution environment within active nodes. Following packets containing data and protocol identification will be processed by the corresponding protocol procedure residing in the internal nodes before they are dispatched on the outgoing links. From this brief description on operation of ANTS, we can see that ANTS is a very dynamic approach to service creation as one protocol can be setup to support a single data stream, and change of protocol is as easy as pack and send an additional active packet.

Architectural model

As shown in figure 4, the architectural model is ANTS is very similar as the IEEE 1520 reference model. Network users (in this case, end-user applications) obtain network services by interacting with the application programming interfaces offered at internal nodes. The collection of APIs is most often referred to as execution environments in active network literature. Similar as the interface between multimedia

network and physical network described in the Xbind project, the execution environments is a collection of APIs that give access to the functions and resources of the internal node. "Execution environment is responsible for all aspects of the user-to-network interface, including the syntax and semantics of the packets the user submits, the nature of the programming model and the abstractions supported, and the addressing and naming facilities." [Smith J. et al. 1999]

To maximize network flexibility, it is desired that active node has the capacity to support multiple execution environments. This strategy increases the network's adaptability and extendibility. However, a single node becomes a shared resource among various execution environments. To ensure the efficient resource sharing and secure system access, it is required that all execution environments interact with the node's operating system to obtain system resource. Besides source allocation and access control, operation system in active node is also responsible for enforcing security policies that govern resource use. As normal IP node, it also keeps routing table information and mechanisms to forward packets out onto transmission facilities. The interface between execution environment and node operating system is a collection of primitives that are responsible for environment query, storage management and control operations. There are all together 10 primitives defined in ANTS operating system, nevertheless, the small number of primitives are proven to be sufficient to express a rich set of processing routines in the many applications developed using ANTS toolkit.

Network technology

Java is the sole programming language used in ANTS because of its comprehensive security facilitates and light-weight process nature. Java Virtual Machine and the sandboxing model for loading code and classes blend really well with ANTS architecture. Mobile code technology is also used as all the executable contents contained in capsules are mobile codes. ANTS is developed to run over the IP network such as Internet.

Level of programmability

The programmability that ANTS offers is very dynamic. Because active packets are the vehicle for service creation and enhancement, programming the network and tailor the services it provides becomes dynamically injecting routines and processing command as required. This flexibility is achieved, however through a significant change in the network infrastructure. As mentioned before, both packet format and internal node behavior are changed in ANTS. The architecture consists of three key components, namely capsule, active node and code distributed system. Different from traditional packet, capsule packets contain its own header specifying protocol identifier and processing routines following ordinary IP header, so active node can identify the capsule boundary and cache the routine for data processing. When packets arrive at the active node's incoming link, its operating system will check the packet type together with security and integrity check to decide whether it is an active or ordinary data packet. If the packet is the usual data packet, it is forwarded to logical links just as in traditional IP networks. However, if the packet is identified as active according to its header, the node operating system will send them to appropriate execution environment if they are proven to be trustworthy and correct. A code distribution system ensures that the processing descriptions are automatically transferred through active packets to nodes that require them.

Security and Performance

Because of the use of Java as its programming language, ANTS obtains many security facilities and already-implemented security check for free. There are some mechanisms at each level of ANTS architecture to ensure data and node security. At active node, Time-to-live (TTL) scheme is used to ensure that the termination of each capsule is bounded by a fix time interval. Active node will discard the capsule if its TTL has reached the limit. In this way, active capsules are limited in their capabilities of reserving resource. Active node has full authority to decide whether it will execute the capsule code, depending on the available resources and trustworthiness of the packet. When active node chooses not to execute the service program, it will perform default IP-like packet forwarding on these denied capsules. In this way, active node is protected form unwanted interaction with the active packet. Within the execution environment, ANTS uses protocol as unit of protection to prevent one protocol interfering with the state of

another. All execution environments reserve resources through node operating system to keep the integrity of active node state and consistent resource allocation.

Because of the programming language and dynamic downloading and binding services at runtime, performance is a big concern of ANTS. The tremendous flexibility comes with a price. ANTS, or active network over all introduces three types of overhead: code shipping, additional header fields and code storage. [Banchs et al. 1998] One performance measurement shows "less than an 8 percent decrease in throughput, and a small increase in latency that corresponds to 20 percent for 512 -byte packets." [Wetherall D. et al, 1998] Another experiment show ANTS “achieves most of the throughput of a Java relay, adding approximately 35 percent overhead of 512 byte capsules.” [Wetheral et al. 1999b] Interesting questions have been raised on many design decision of ANTS such as strategic points to place active nodes, and overall improvement on performance for specific applications rather than entire network.

5.3 SmartPacket

Application domain

SmartPackets project in BBN is funded by the US Defense Advanced Research Project Agency for the goal of exploring the possibility of applying active network technology in IP network management. Today’s common network management strategy is that there are few powerful central processor working as managing centers. The managing centers routinely poll information from managed nodes. Three deficiencies are associated with this strategy: first on all, the polling takes place during certain interval, which makes the network slow to response to faults; secondly, polling requires wide bandwidth. With network grows exponentially, this process does not scale well with the expansion of network. Thirdly, huge chunk of data is transmitted through the network, but little is actually of any relevance after processing. The overall network utilization and efficiency suffer from this large amount to date transportation. SmartPackets project aims to solve these problems in IP network management by using active network technology. Active packets are injected into the network element closer to the managed node to filter out and process information before transmitting. [Kulkarni et al. 1998]

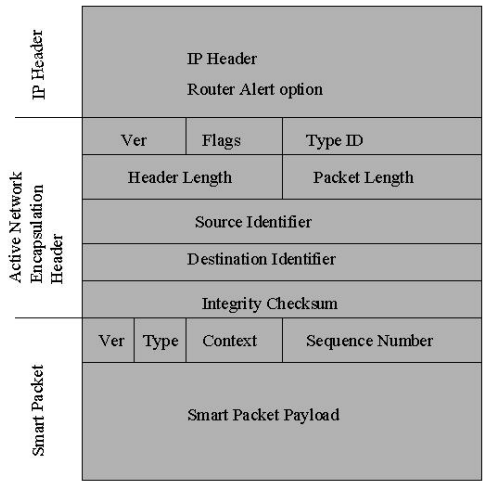

Figure 4: Active packet format in SmartPackets Project

Architectural model

SmartPackets' architecture is made of the following components. The first and most interesting component is the format of the smart packet. The format of a typical active message is shown in figure five [B. Schwartz et al. 1999]. Besides the IP header we expect in any packets, there is also another header called Active Network Encapsulation Header, which specifies the message type. There are four types of packet in SmartPackets project, namely program packet (active packet), data packet or error packet and message packet. Program packets contain executables in its payload, while data packets are very much similar as ordinary IP packet.

Similar as in ANTS, there is also an execution environment required in active node to receive smart packets and execute the code they carry. As the purpose of Smartpacket project is for network management and performance monitoring, the primitives offered by the execution environment is just a subset of the interface provided node operating system in ANTS. No access to the file system is allowed. Smart packets

are strictly forbidden to manipulate or change state variables in the hosting node, either are they allowed to plant global network state flags in the visited node. This characteristic, as a result of the application requirement, secures the consistence of the network.

Network technology

SmartPackets project is designed and implemented for IP network management. Similar as in Xbind and ANTS, physical network elements are abstracted into managed objects. These objects offer low-level APIs, a set of operating primitives that provide sufficient functionality and resources to execute bytecodes carried in the active packets. "Smart packets are encapsulated with an Active Networks Encapsulation Protocol (ANEP) developed for the DARPA Active Networks Program to facilitate portability and interoperability with other Active Networks projects. " [Schwartz et al. 1999] Similar as in ANTS, mobile code technology and distributed delegation are used to achieve scalable and flexible network management.

Two programming languages are used in the implementation of SmartPacket project. Sprocket, a programming language much like C, is used for high-level application implementation. Taking into account of performance and security, Sprocket is compiled into a much efficient language called Spanner, which is a machine independent assembly language with high efficiency. [Schwartz et al. 1998] All smart packets carry Spanner assembly codes that are ready to run at the virtual machine provided at active nodes.

Level of programmability

SmartPackets project makes the managed nodes programmable. After active packets deliver data processing routines, managed nodes are able to filter out redundant data, perform simple statistical computation on the data passing through. These nodes do not need to be active initially. When network managing center detects increasing in network scale, they can active the internal nodes to make them share some of the data processing responsibilities. This process is highly programmable, as first of all, the information returned to the managing center can be easily tailored. Secondly management rules can be sent to managed nodes to detect problem and handle faults as soon as they occur. This proactive approach to

network management is revolutionary compared to the passive strategy of polling the information when errors occur.

Security and Performance

Security is always a big concern for active networks. This is also the challenge faced by SmartPackets project. Security is ensured at the network element level as smart packets are executed within virtual machine that only interacting with the node operating system. Also smart packet daemon running at the node are responsible for checking the authentication and data integrity of the packet before it is admitted into the virtual machine. A cryptographic hashing function is used to encrypt the code and each smart packet carries an authenticator identifying the entity that originated the packet. Packets are discarded if authentication fails.

5.4 Mobile Intelligent Network

Application domain

Breugst et al. in Technical University of Berlin, Germany proposed a mobile agent based active network architecture to transform today's telecommunication Intelligent Network (IN) to Active Intelligent Network [Breugst M. et al. 1998]. The foundation of this architecture is based on two enabling technologies: Mobile Agent and Programmable Switches. Current Intelligent Networks provide not only connection and switching services for communicating nodes but also integrated services such as Call Forward, Call Screen, Automatic Call back, etc. Connection and switching functions are performed by Service Switching Points (SSP's) and the additional intelligent services are performed by a centralized functional unit call Service Control Point (SCP). Typically, a SCP is responsible for a collection of SSP's. Because SSP's can be from different vendors, a standard interface call SS7 is necessary between SSP's and a SCP. A mobile agent based active network moves intelligent service functions towards the SSP's eliminating the needs excessive communication between the SCP and SSP. The use of mobile agent and programmable switches make it possible for automated, flexible and customized provision of services in a highly distributed way.

Architectural model

Mobile Intelligent Network architecture offers a platform allowing mobile agents to migrate along with user's calls. The mobile agents are called Intelligent Network Service agents in their project. These Intelligent Network Service agents are time-dependent, i.e. installed for a limited time duration, and location dependent, i.e. installed at dedicated switches or end user systems. The intelligence inside the service agents enables a distributed configuration and provision of service intelligence into the telecommunication network. On the other hand, each Intelligent Network Element (SSP's, SCP's, communicating devices such as phones and computers), provides an agency in which service agents are able to execute. Called by a different term, the agencies are functioning in the same way as the executation environment to the agent. To be more precise, they provide sufficient resources for the execution of program carried by the agents. Collectively, these agencies form a distributed agent environment. Thus service agents can easily migrate in this "harmonized" service environment from the service provider system to the most appropriate network node.

Network technology

Service logic and data can be implemented using CORBA based objects, i.e. each IN service capability/feature may be implemented by an appropriate object within a distributed processing environment (DPE). Since Mobile Agents can be regarded as extensions of plain CORBA objects, these service objects are implemented as mobile service agents residing in an agency which provides, besides the basic agent lifecycle management and mobility services, other specific services such as switch/resource control interfaces. Correspondingly, a mobile service agent also comprises two parts – application part and core agent part. Application section of the agent deals with the provision of real services. Service control logic and data together with the procedures need to be carried out at external switch through their control interfaces are the key elements consisted in this part. Depending on the application, this part also includes functionality for service subscription, customization and maintenance. The second part is the core agent component. This section of code make the agent behave in autonomous way on behalf of the application Agent lifecycle management, migration pattern and mobility support are also included in this section within the agent.

Level of programmability

The proposed architecture can be easily mapped to the discrete approach of active network architecture. “A practical vision of implementation is to consider the capsules as mobile agents, where an end system initiating a communication with another end system may initially download a customized call model and corresponding protocol stacks inband into all relevant open switches along the designated network route. This means that each application can program the network individually for the time of service provision. In this way, ultimate flexibility in network programming and service provision can be achieved. “ [Breugst et al. 1998]

5.5 Liquid Software

Application Domain

Joust project [Hartman et al. 1999] developed at University of Arizona is also funded by DARPA. Equivalent as the executable mobile codes, the term liquid software is used to referred to low-level communication oriented executables that easily flows from machine to machine. As illustrated in the ANTS projects, execution environments request resources form the internal node by interacting with the nodes’ local operating system. “General-purpose OSs, however, are typically designed for computation, not communication, and provide neither the performance nor the services required by active networks.” [Hartman et al. 1999] Because of the communication nature of active networks, special operating system is needed to support real-time stream delivery, multicasting and quality of service monitoring and control.

Architectural model

In the aim of avoiding the limitations for the general-purpose operating and runtime systems, Joust tries to build a platform that works as an operating system on top of the physical network element, however, offers more support for data communication and scheduling. The architectural model in Joust consists of three key components:

- A communication oriented, Java-based, highly configurable operating system called Scout. Scout consists of many modules. Modules is regarded as the unit of programmable entity that provide

well-defined and independent functionality. Modules can be configured and composed into Scouts.

- A JVM implementation that gives high-performance and support for mobile codes' execution.
- A JIT compiler that is able to translate Java bytecode into native instructions.

This architecture model is very similar as the model used in ANTS project if we map the modules as the primitives offered by the hardware, and Scout operating system as the interfaces between network elements and their logic abstraction. Again, the Java Virtual Machine provides the environment for the liquid software to execute.

Network technology

Mobile code technology is used extensively in this project. Java is the programming language for the development of high-level application interface, as well as the communication-oriented operating system. However, there is a JIT compiler which translates Java code into native instruction to the physical element. This translation and just –in-time compiling are believed to improve overall system performance. Joust project does not implement its own security mechanism. Because of the use of Java Virtual Machine, system loading safety and data integrity are ensured to a certain extend.

Level of programmability

Similar as the service engine object in Xbind, modules in Joust are the basic unit of service functionality. They can be easily combined and configured to form a new Scout – an platform in which liquid software (mobile codes) can run. New Scout can be easily configured and same Scout easily tailored by adding, removing and modifying the modules to support different applications. Because of this configurable nature of Scouts in Joust, the programmability level of this project is also very dynamic.

5.6 SwitchWare

Application domain

Similar as the intension for ANTS, SwitchWare project [Alexander et al. 1998a] aims at providing flexibility and programmability into the IP network without scarifying security and performance.

Developed at University of Pennsylvania, the project make it their objective to design and develop a network that is quick in response of introduction of new protocols to the network.

Architectural model

From the above comparison, we can see that all active networks are constructed in a very similar way as active packets traversing the network, executing the executable content it carries at active nodes' execution environment to realize services. Similar architecture is applied in SwitchWare project as well. There are three components in SwitchWare architecture: first of all, there are active packets that keep similar format as in SmarkPackets. Active packets are also the vehicle of service delivery in this project. Secondly, the active extension forms the second layer of the network. The extensions can be dynamically loaded, but they are statically executed just at decided switches and routers. They provide the core functionality of the service. The bottom layer of the model again is the physical elements which provide the foundation on which the other two layers are built.

Network technology

Mobile code technology is used for the flexibility of active packet in SwitchWare project. Programming language Caml is used rather than Java because of the following features Caml offers. First of all, Caml bytecodes are dynamically loadable and machine independent, thus mobility of the executable is supported. Secondly, simple as its implementation, Caml is believed to be more efficient that Java. The functionality Caml offers is rather limited, but it is proven to be adequate for the purpose needed by SwitchWare project. Not using Java virtual machine and class loader, Caml faces the serious security questions. A complicated security system is implemented to ensure the system integrity. Security facilities includes public facility which is available to all applications, authenticated facilities, which is a check on user' identity, and verified facilities which are implemented using cryptographic algorithms and barriers to check code content consistency. All three facilities are developed at different level in SwitchWare project. [Alexander et al 1998b]

Level of programmability

SwitchWare also offers a high level of programmability and flexibility with the combined use of active packets, dynamic extension and static extensions. The core functionality or processing of a service is download to the switches as extensions. Data together with the function calls of how the data are to be processed arrive at the extensions as active packets. Active packets invoke the extension, thus provide the customized services required by this particular application. Conversely, extensions are able to send and receive active packets to achieve remote communication. In this manner, supporting a new protocol or tailor services to one application are made very easy.

6. Conclusion

From the above comparison details, it is very obvious to see that to add programmability into the legacy network, we have to somehow open up the software control of the network element that used to be tightly couple with the physical hardware. No matter how their implementation differs, opening up the network control is the key theme that we observe in all the implemented projects. Aiming at the same goal, programmable network and active network are two related, but very distinctive approach toward network programmability. Programmable network keeps the legacy networking structure, while insists on the provision of a set of general-purpose API at the network level. The rich set of APIs can be used to combined and recompose new network services. On the other hand, active network completely change the packet format and node behavior in the traditional network, and make the packets the vehicle for service delivery and provisioning. Active network is more dynamic than programmable network in that the service and control can be tailored to one single packet.

In spite of all the claims made on network programmability, it is still a technology that far from maturity. More serious work is needed in the area of ensure security. From the brief survey of the above projects, security is a issue that discussed intensively in all projects, however, none of the project implements a comprehensive set of security mechanisms that are inmute to attacks from active packets, hostle nodes and other frauds. Most of the projects count on Java's security check to ensure data integrity. However, as we all know, more security facilities are needed to ensure the overall reliability of the network. Performance

is another issue on which quite a few research have been addressing. [Vanecke et al. 1999] [Chan et al. 1999] All in one word, programmable network and active network are two very promising approaches towards a more flexible and responsive network even in their infancy. More research and work are needed in the area of performance measurement and improvement, infrastructure analysis and security assurance.

Reference:

1. [Alexander et al.1998a] Alexander D., Arbaugh W., Hicks M., Kakkar P., Keromytis A., Moore J., Gunter C., and Nettles S *The SwitchWare Active Network Architecture* **IEEE Networks** Vol.11. No.2 May/June 1998 P29 – 36
2. [Alexander et al. 1998b] *A Secure Active Network Environment Architecture* **IEEE Network** Vol.11. No. 2 May/June 1998
3. [Aurrecoechea et al. 1998] Aurrecoechea C., Lazar A., and Stadier R. *Opening Network Services for Management* **Open Architectures and Network Programming IEEE 1998** P61-71.
4. [Banchs et al. 1998] Banchs A., Effelsberg W., Tschudin C. and Turau V. *Multicasting Multimedia Streams with Active Networks* **IEEE First International Conference on Open Architectures and Network Programming** 1998 P150-159.
5. [Biswas et. al 1998] Biswas J., Lazar A., Mahjoub S., Pau L. Suzuki M. Torstensson. S, Wang W. and Wsinstein S. *The IEEE P1520 Standards Initiative for Programmable Network Interface* **IEEE Communications Magazine**, Oct. 1998 P 65-71.
6. [Breugst et al. 1998] Breugst M., and Magedanz T. *Mobile Agents – Enabling Technology for Active Intelligent Network Implementation* **IEEE Networks** Vol.11 No.2 May/June 1998 P53 - 60
7. [Calderon et al. 1998] Calderon M., Sedano M., Azcorra A. and Alonso C. *Active Network Support for Multicast Applications* **IEEE Networks** Vol.11 No.2 May/June 1998 P46-52.
8. [Campbell.et al 1997 a] Campbell A., Lazar A., Schulzrinne H., and Stadler R *Building Open Programmable Multimedia Networks* **IEEE Multimedia** Jan/Mar. 1997 P77-82.
9. [Campbell.et al 1998 a] Campbell A., Lazar A., Schulzinne H., and Stadler R. Building Open Programmable Multimedia Networks **Computer Communication Journal** Vol. 21 No. 8 June 1998 P758-770.
10. [Campbell.et al 1998 b] Campbell A., DeMeer H., Kounavis M., Miki K., Vicente J., and Villela D. *A Survey of Programmable Networks* **ACM SIGCOMM Computer Communication Review** Vol. 29 No. 2 April 1999. P7 – 23.
11. [Chan et al. 1999] Chan M. and Lazar A. *Designing a CORBA-based High Performance Open Programmable Signaling System for ATM Switching Platforms* **IEEE Journal on Selected Areas in Communications** Vol. 17 No. 9 Sept. 1999 P1537-1548.
12. [Chen and Jackson, 1998] Chen T. and Jackson A. *Active and Programmable Networks* **IEEE Network** May/June 1998 P10-11.
13. [Ghonaimy 1999] Ghonaimy M. *New Generation Internet and the Evolution Towards Active and Programmable Networks*, Preceedings of **16th National Radio Science Conference, NRSC'99**
14. [Hartman et al. 1999] Hartman J., Bigot P., Bridges P., Montz B., Piltz R., Spatscheck O., Proebsting T., Perterson L. and Bavier A Joust: A Platform for Liquid Software **IEEE Computer** 1999 P50 – 56.
15. [Huard and Lazar 1998] Huard F. and Lazar A., *A Programmable Transport Architecture with QoS Guarantees* **IEEE Communications Magazine** Vol. 30 No.10 Oct. 1998 P 54 – 62.
16. [Kulkarni et al. 1998] Kulkarni A., Minden G., Hill R., Wijata Y., Gopinath A, Sheth S., Wahhab F., and Nagarajan A. *Implementation of a Prototype Active Network* **IEEE First International Conference on Open Architectures and Network Programming** 1998 P130-143.
17. [Lazar 1997] Lazar, A. *Programming Telecommunication Networks* **IEEE Networks** Vol. 11 No.5 Sept. – Oct. 1997 P 8 –18.
18. [Lazar et al. 1996] Lazar A., Koon-Seng L. and Marconcini. F. *Realizing a Foundation for Programmability of ATM Networks with the Binding Architecture* **Selected Areas in Communications, IEEE Journal.** Vol.14 No. 7 Sept. 1996 P 1214-1227.
19. [Schwartz et al. 1999] Schwartz B., Jackson A., Strayer T., Zhou W., Rockwell R. and Partridge C. *Smart Packets for Active Network* **IEEE Second International Conference on Open Architectures and Network Programming** 1999 P90-97
20. [Suzuki et al. 1998] Suzuki M., Wang W., and Weinstein S. *Programmable Networks* **IEEE Communications Magazine Oct 1998** P40 –41.
21. [Tennenhouse et al 1997] Tennenhouse D., Smith J., Sincoskie D., Wetheral D. and Minden G. *A Survey of Active Networks Research* **IEEE Communication Magazine** Vol. 35 Jan. 1997 P80-86.

22. [Vanecek et al. 1999] Vanecek G., Mihai N., Vidovic N., and Vrsalovic D. *Enabling Hybird Services in Emerging Data Networks*. **IEEE Second International Conference on Open Architectures and Network Programming** 1997 P 102 - 109
23. [Wetheral et al. 1998c] Wetheral D., Guttag J. and Tennenhouse D. *ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols*. **IEEE Proc. IEEE OPENARCH'98** 1998 P117 – 129.
24. [Wetheral et al. 1998a]Wetheral D., Legedza U. and Guttag J. *Introducing Net Internet Services: Why and How* **IEEE Networks** Vol.11 No. 2 May/June 1998 P12-19.
25. [Wetheral et al. 1999b] Wetheral D., Guttag J and Tennenhouse D. *ANTS: Network Services Without the Red Tape* **IEEE Networks** Vol. 11 No.2 May/June 1998 P42-48.
26. Details on Xbind http://comet.ctr.columbia.edu/xbind
27. OpenSig documentation http://comet.ctr.columbia.edu/opensig/
28. ANTS: http://www.sds.lcs.mit.edu/activeware
29. Liquid Software: ftp://ftp.cs.arizona.edu/xkernel/Papers
30. Smartpackets: http://www.net-tech.bbn.com/smtpkts/smtpkts-index.html