# Virtual Active Networks – Safe and Flexible Environments for Customer-Managed Services

Marcus Brunner[1] and Rolf Stadler[2]

[1]Computer Engineering and Networks Laboratory, TIK
Swiss Federal Institute of Technology Zurich (ETH)
Gloriastr. 35, CH-8092 Zurich, Switzerland
E-Mail: brunner@tik.ee.ethz.ch

[2]Center for Telecommunications Research (CTR)
and Department of Electrical Engineering
Columbia University, New York, NY 10027-6699
E-Mail: stadler@ctr.columbia.edu

**Abstract.** Recent research has demonstrated the benefits of active networks: customized network services can easily be built and modified, packet streams can be processed inside the network, etc. This paper addresses the question how the benefits of active networking can be exploited in a telecom environment, where a large number of customers must share a common network infrastructure. We introduce a framework that allows customers to deploy and manage their own active services in a provider domain. The key concept in this framework is the *Virtual Active Network (VAN)*. A VAN is a generic service, offered by the provider to the customer. From the customer's point of view, a VAN represents an environment on which the customer can install, run and manage active network services, without further interaction with the provider. From the provider's perspective, the VAN serves as the entity for partitioning the provider's resources and isolating customers from one another in virtual environments. We describe how the VAN concept, VAN management, and customer service management is realized on ANET, an active networking testbed.

**Keywords.** Management of Active Networks, Service Provisioning, Service Management, Active Network Testbeds

## 1 Introduction

In today's telecom environments, customers and service providers typically interact via two interfaces--the management interface and the service interface. The management interface, based on standardized management protocols, is generally used for service provisioning and service management. The customer accesses the service through the service interface. Both interfaces are service specific; a provider has to support different service and management interfaces for each service offered, which makes the introduction of new services dependent on standardization, and therefore time consuming and expensive.

The introduction of active networking technology in telecom environments, characterized by the use of active networking nodes as network elements, will change the role of these interfaces in two ways. First, using active networking nodes enables the definition of a generic service interface for network services, based on the concept of *active*

R. Stadler and B. Stiller (Eds.): DSOM'99, LNCS 1700, pp. 195-210, 1999.

*packets* (see Section 2). In addition, this service interface can be used by the customer for service management interactions, i.e., for operations related to the installation, supervision, upgrading and removal of a specific service. Therefore, service management operations can be performed by a customer without interaction with the provider's management system. Second, the management interface, i.e., the interface through which the customer and the provider management systems cooperate, can be restricted to the task of service provisioning. Similar to the service interface, the management interface can be kept generic; it relates to a generic service abstraction that allows for installing and running a large class of network services.

The introduction of active networking technology, where network nodes perform customized processing of packets, will change both types of interactions described above. The granularity of a service abstraction and its related control capabilities can be chosen, ranging from a very limited, constrained service model to a very detailed one. The key concept that provides this capability is the *Virtual Active Network (VAN)*, which is the focus of this paper. In the same way as an active network can be understood as a generalization of a traditional network, a VAN can be seen as a generalization of a traditional Virtual Private Network (VPN). Similar to a VPN, a VAN can be used by a customer to run active networking services, using a provider's networking resources. In contrast to a traditional VPN, however, a VAN gives a customer a much higher degree of controllability. Further, the VAN concept supports rapid installation and upgrade of customer-specific active network services in a telecom environment.

The paper is organized as follows. Section 2 briefly reviews the concept of active networking. Section 3 outlines our framework for customer-provider interaction in an active telecom environment, i.e., an environment that is based on active networking technology. Section 4 introduces the VAN concept, and Section 5 gives aspects of VAN provisioning. Section 6 describes an active networking platform we have built for experimenting with active networking concepts and shows how VAN provisioning and customer-controlled service management is realized on our platform. Section 7 surveys current efforts on active technologies for network management. Section 8 summarizes the contributions of this paper and gives an outlook on further work.

## 2 The Concept of an Active Network

The processing of packets (or cells) inside traditional networks is limited to operations on the packet headers, primarily for routing purposes. Active networks break with this tradition by letting the network perform customized computation on entire packets, including their payloads. As a consequence, the active network approach opens up the possibilities of (1) computation on user data inside the network and (2) tailoring of the packet processing functions in network nodes according to service-specific requirements.

Active networks transport *active packets* [1] (also called capsules [2]). Active packets carry programs, in addition to data. A network node executes such a program, which possibly modifies the nodes' state and possibly generates further active packets to be sent over the outgoing links. Specifically, an active packet can include a program that modifies or replaces the nodes' packet processing function.

Similar to traditional networks, an active network consists of active network nodes, which are connected via links. In addition to transmission bandwidth, the key resources of an active network node include memory and CPU resources for processing active packets.

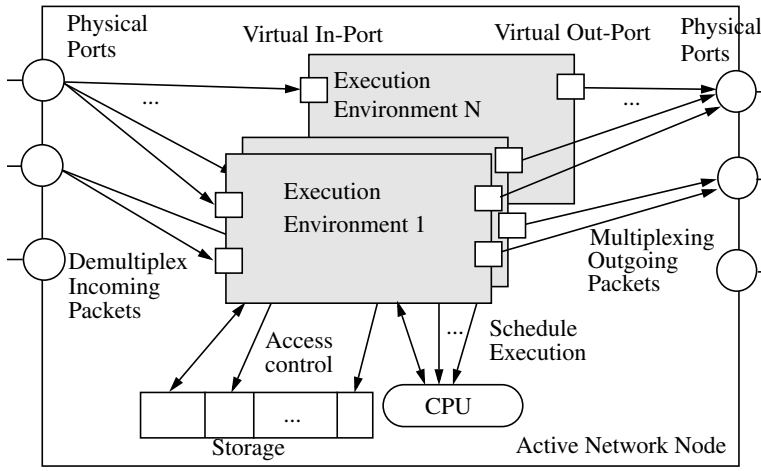



**Figure 1:** Active Node Architecture

Figure 1 gives a model of an active network node. The basic functions of such a node are (1) the control of the incoming packets, (2) the control of the outgoing packets, (3) packet processing, and (4) memory access. An active network node runs several Execution Environment in parallel. Each active packet arriving at a physical port contains an identifier of the Execution Environment that will process this packet.

Figure 1 also illustrates that an active network node can be seen as a generalization of traditional network node, such as an IP router. An IP router is limited in the sense that there is only one Execution Environment and a single set of pre-installed functions for packet processing.

A different view of an active network node is given in Figure 5, which stresses operating system aspects. This figure is specialized for a provider-customer environment where each customer service is run in a separate Execution Environment.

## 3 Customer-Provider Interactions in Active Telecom Environments

### 3.1 Interactions in Traditional Environments

Figure 2 shows the interaction taking place between a customer domain and a provider domain for the purpose of service provisioning, service delivery, and service management. Depending on the type of service, customers and providers interact in two fundamentally different ways. The first way is characterized by a provider offering functionality in its domain through a service interface. A typical example of such a service is a virtual network, e.g., a Virtual Path (VP)-based service, which is purchased by a customer with geographically separated premises networks in order to construct a

company-wide enterprise network. In the provisioning phase, the customer negotiates the connectivity and resources of the service, i.e., the virtual network topology and the quality of service (QoS) requirements. (A customer may request a set of constant bit rate (CBR) VPs of a certain bandwidth with delay and loss bounds.) This interaction takes place via the management interface, which interconnects the customer's and provider's management systems, and generally includes communication between human operators on both sides. The provisioned service can be accessed via the service interface, which corresponds to the user-network interface (UNI) in our example.

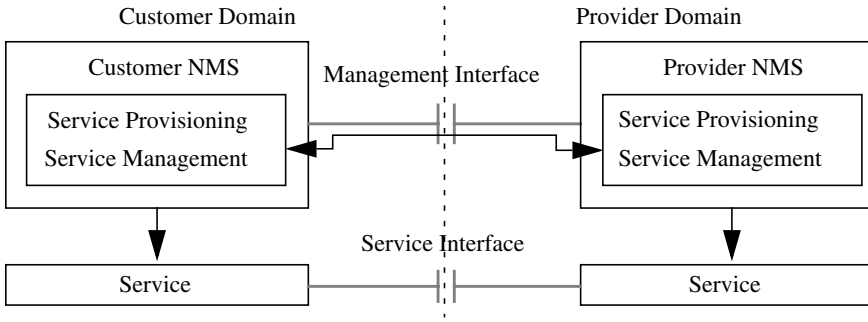



**Figure 2:** Management Interaction in a Traditional Telecom Environment

The second way of interaction shown in Figure 2 relates to the case where a customer outsources control and management of a specific service to a provider, which installs and runs the service in the customer domain. In this case, the customer gives the provider access to its networking elements via management interfaces. The customer is not involved in service installation, upgrade, and management, but can concentrate on its core business instead. The provider customizes the service according to the customer's requirements.

### 3.2 Our Framework for Active Telecom Environments

In an active networking environment, the above described two ways of interactions between a customer and a provider can be realized in a much more flexible way with respect to service abstractions and control capabilities for the customer in the provider's domain and vice versa. In the following, we outline our framework for interaction in an active networking environment, which we have first proposed in [3].

Figure 3 shows the interaction between a customer domain and a provider domain for service provisioning, service delivery and service management in our framework. When comparing Figure 3 with Figure 2, the key differences between a traditional environment and an active telecom environment that follows our approach become clear. We propose that the provisioning of a specific (active) network service X is split into two different operations--a) the *provisioning of a generic service*, which we call the VAN service, is performed via the cooperative VAN provisioning interface; and b) the *installation of the specific service* X is performed via the generic service interface, without further interaction with the provider. The same interface is used by the customer for managing service X during its lifetime.

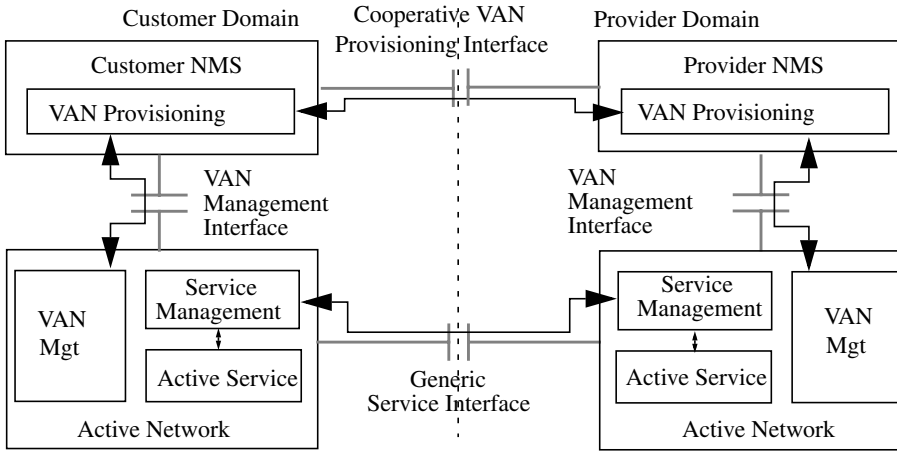



**Figure 3:** Management Interaction in an Active Telecom Environment that follows the VAN concept

The *generic service interface* for accessing network services is enabled through the concept of active packets. It is used for data transport, as well as for all operations related to the installation, supervision, upgrading and removal of a specific service X. (See Section 6.2 for an example of service installation and supervision through this interface).

The *cooperative VAN provisioning interface*, i.e., the interface through which the customer and the provider management systems cooperate, is restricted to the task of VAN provisioning. VAN provisioning includes negotiating the Virtual Active Network topology and the resources allocated to the VAN.

Within a domain, a VAN is created, changed, and monitored by the VAN provisioning system, which interacts via the *VAN management interface* with the active networking platform. This interface can be realized using management protocols, such as SNMP or CMIP, or--by exploiting active networking technology--it can be realized using a similar interface as the generic service interface. In the latter case, VAN provisioning via the VAN management interface uses the same techniques as service provisioning via the generic service interface. (See Section 6.1 for a specific example.) The (local) VAN management interface for an active network node is given in Section 6.

## 4 The Virtual Active Network (VAN)

A *Virtual Active Network (VAN)* can be described as a graph of virtual active nodes interconnected by Virtual Links. Virtual active nodes are also called *Execution Environments (EEs),* following the terminology of the AN working group [1]. A virtual active node has resources attached to it in form of processing and memory resources, provided by the underlying active networking platform. Similarly, a virtual active link has bandwidth allocated to it. We envision that a single (physical) active node can run several virtual active nodes belonging to different VANs, and a single (physical) network link can support several Virtual Links for different VANs. (The term Virtual

Active Network, as defined in this paper, is also used by other authors with a different meaning [4].)

Figure 4 shows an active network with five nodes in a provider domain. On this network, two VANs have been installed, one for customer 1 and one for customer 2. The figure also shows the Management VAN, interconnecting the Management EEs, which are used by the provider for VAN provisioning and supervision (see Section 5).

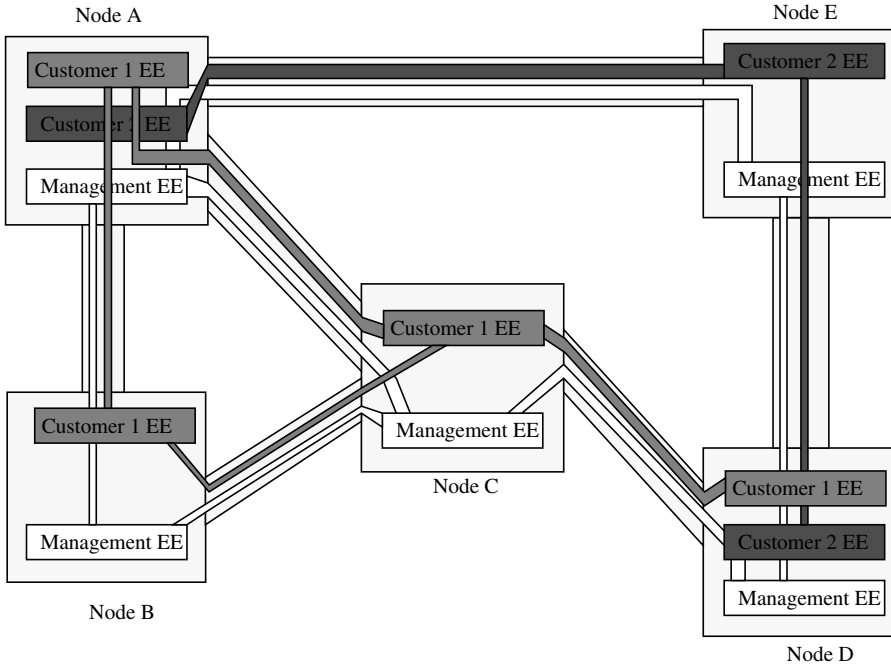



**Figure 4:** An active network with nodes A-E, on which VANs for customers 1 and 2 have been installed.

What problem exactly does the VAN concept address and what are its benefits? First, the VAN provides a *generic service abstraction* for an active telecom environment. From a provider's point of view, the VAN is the entity according to which active network resources are partitioned and according to which the customers, using the provider's infrastructure, must be isolated from one another. The VAN is further the (only) object that is shared between provider and customer, and it is the object of negotiation between the two parties. Specifically, the provider is not concerned about which specific service(s) a customer is running on its VAN. The task of the provider is solely to monitor and police the use of resources on the VAN level and to ensure that the QoS, as agreed upon between customer and provider, can be guaranteed.

Second, from a customer's perspective, the VAN concept allows for installation and *management of active network services, without interaction with the provider.* (As mentioned above, all interactions between customer and provider relate strictly to the VAN.) The customer can run a large variety of active network services on the VAN.

These services are only restricted by the specific Execution Environment(s) the VAN supports. (Developing Execution Environments for active networks is currently subject of intensive research. In our work, we base on the current state of the AN working group [8]). Note that the high degree of customer control over services running in a provider's domain, which a VAN provides, is virtually impossible to realize with today's "traditional" networking technology.

Third, --exploiting a general benefit of active networking-- the VAN concept enables *rapid deployment of new network services*. Deploying and upgrading network services is difficult and time consuming in today's networks, due to the closed, integrated architecture of network nodes. With the concept of a VAN, which divides the active network resources into partitions for different customers, the installation of any customer-specific service becomes feasible, and, as explained before, it can be accomplished by the customer alone, without interaction with the VAN provider.

Lastly, customers can run a mix of different network services on a single VAN. This allows customers to perform *dynamic re-allocation of VAN resources* to the various services, according to their own control objectives and traffic characteristics--again, without interaction with the VAN provider.

As mentioned before, the VAN concept can be compared to that of a Virtual Path (VP)-based Virtual Private Network (VPN). Similar to a VAN, a VP-based VPN provides customers with a service abstraction, on which they can run their own services, such an IP-based data or real-time services. Since a VP is a simple abstraction, a customer's ability to control traffic inside the provider's domain is very limited. (See [19] for a discussion of this point.) A VAN, on the other hand, is a much more complex abstraction than a VP, and, consequently, gives customers extensive control capabilities inside the provider's domain. In a similar way as dynamic bandwidth provisioning can be performed in a VP-based VPN [19], we envision that VAN resources can be re-negotiated during the life-time of a particular VAN via the VAN management interface shown in Figure 3.

## 5 VAN Provisioning and Supervision

Figure 5 gives an operating system point of view of an active network node in the provider's environment. A node operating system layer configures and provides access to the node's resources, such as links, processing and memory resources. This layer runs the Execution Environments, separates them from each other, and polices the use of the resources consumed by each Execution Environment.

Figure 5 specifically shows the case where a provider offers Virtual Active Networks to several customers (Customer 1, Customer 2, ..., Customer N). Each customer runs its service in a separate Execution Environment, which corresponds to a Virtual Node of the VAN.

A privileged Execution Environment, the *Management EE,* runs the provider's VAN management system, which creates the customer EEs and is able to modify, monitor and terminate them. The Management EE is accessed by the provider's VAN provisioning system via the local VAN management interface, which is described in Section 6.

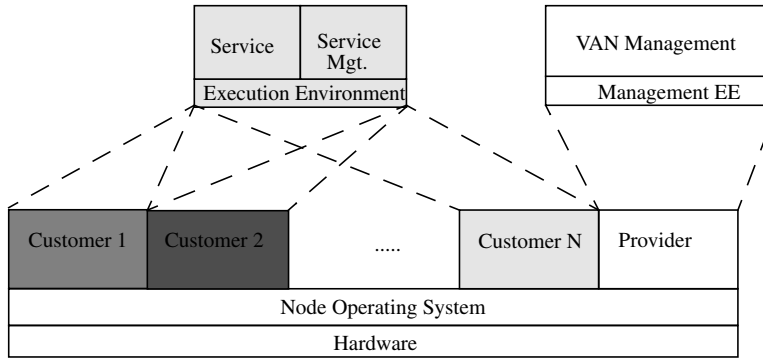



**Figure 5:** Architecture of an Active Network Node in a Telecom Environment

At the beginning of the VAN provisioning phase, the customer and the provider negotiate the topology and the resource requirements for the customer's VAN. The provider's VAN provisioning system then maps the VAN topology onto its active networking platform and interacts with some of the active nodes via the VAN management interface to create this VAN. By accessing the Management EEs on the active nodes, the provider sets up EEs for the customer and interconnects them via virtual links, according to the VAN topology that has been negotiated with this particular customer.

Note that the design given in Figure 5 is compliant with the architecture of an active network node developed by the AN Working Group [1]. The difference between Figure 5 and [8] is that we explicitly assign each Execution Environment to a particular VAN, i.e., to a particular customer.

## 6 Realizing VAN Provisioning and Management on the ANET Platform

We have built an active networking platform, in order to test, evaluate and demonstrate active networking services and management concepts. The core of this platform consists of a cluster of Ultra-SPARCs, interconnected via an Ethernet and an ATM LAN. Each active network node runs on a separate workstation. On top of this infrastructure, we have implemented traffic generators, traffic monitors, a VAN management system, and a service management system.

All software components of our platform are written in Java. We chose Java because of its strengths as a prototyping language for networking environments, and because Java directly supports the realization of active packets through the concept of mobile code. Additional Java features which we take advantage of include object serialization, thread support, and safe memory access achieved by the type-safety of the language. In our implementation, an active network node is implemented entirely in software, which gives us the flexibility of experimenting with different designs. Achieving high performance, such as realizing a high throughput of packets on a network node, is not the focus of our work on the ANET platform. (Instead, we plan to realize the VAN concept, once it has been fully studied on the ANET platform, on a high-performance

active node, which is currently under development in our lab at ETH Zurich [5].)

The complexity of the software we have built to date--active network node, provider management system, and service management system--is in the order of 400 Java classes with 30'000 lines of code.

In the following, we describe the realization of an active network node, as outlined in Figure 5, on ANET. The in-bound ports of the active node deliver the incoming packets to the appropriate Execution Environments, identified by a multiplexing identifier in the packet header or in underlying protocol headers. The node operating system schedules the Execution Environments, taking into account the processing resources allocated to each of the Execution Environments. Packet schedulers on the outgoing ports multiplex and transmit packets produced by the Execution Environments to neighboring nodes.

The provider VAN management system, which performs VAN provisioning and supervision, interacts with customers over the cooperate VAN provisioning interface, and with the active network nodes via the VAN management interface. On ANET, we have implemented the VAN provisioning system on a VAN management station. This management station performs VAN management operations by sending active packets via the Management VAN to particular Management EEs in the network. These active packets, which are executed in the Management EEs, perform the management operations on the active nodes via the local VAN management interface.

Table 1 lists the most important functions of the *local VAN management interface* as implemented in the ANET prototype. The functions are described using a Java-like notation with input parameters in brackets and return values in front of the function name.

| |
|---|
| `vin_portid install_Virtual_InPort (in_portid, eeid, inmid);` |
| `vout_portid install_Virtual_OutPort (out_portid, eeid, outmid, bandwidth);` |
| `void remove_VirtualPort (vportid);` |
| `eeid install_EE (ee, cpu_resource, memory);` |
| `void remove_EE (eeid);` |
| `cut_through_id install_ThroughLink (in_portid,inmid,out_portid,outmid,qlen,bandwidth)` |
| `void remove_ThroughLink(cut_through_id);` |

**Table 1:** Local VAN Management Interface of an ANET active node

The *install_Virtual_InPort* function creates a *Virtual In-Port* for the Execution Environment identified by *eeid*. It further configures a physical port identified by the *in_portid* to dispatch incoming packets with multiplexing identifier *inmid* to the just created Virtual In-Port.

The *install_Virtual_OutPort* function creates a *Virtual Out-Port* for the Execution Environment identified by *eeid*, and multiplexes outgoing packets leaving the Execu-

tion Environment over the just created Virtual Out-Port with multiplexing identifier *outmid*. The outgoing stream of packets is constrained by the amount of *bandwidth* reserved with this function. Any virtual port can be removed in case the VAN topology changes.

The *install_EE* function installs a new Execution Environment given in the parameter *ee*. The node operating system is configured to give the Execution Environment a portion of *cpu_resource* and *memory* and to police it to not overconsume the reserved resources. An Execution Environment identifier is returned for further use.

The *install_ThroughLink* function installs a cut-through path from the physical port with identifier *in_portid* to the physical port identified by *out_portid*. Packets arriving at *in_portid* tagged by the multiplexing identifier *inmid* travel along the cut-through path and are not executed in an Execution Environment. They are switched to the specified outgoing link, where they are sent with multiplexing identifier *outmid*. The node operating system has to know what amount of *bandwidth* is expected along the cut-through path to configure the packet scheduler accordingly. Further, a buffer with length *qlen* is created to store the packets in case they have to wait some time until the packet scheduler serves them.

The local VAN management interface includes functions that are not shown in Table 1. Most of them relate to monitoring operations, such as gathering statistics about the state of the node operating system, the resource consumption by the customer Execution Environments, virtual ports, and cut-through links.

Similar to the VAN management station, we have implemented the (customer-operated) service management system on a service management station. The service management station sends active packets into the VAN to perform management operations, such as the installation of a particular service and the supervision of the service.

In the following, we illustrate some of the design principles and capabilities of the ANET platform, by describing a series of demonstrations we can perform.

### 6.1 Demonstrating VAN Provisioning and Supervision

Figure 6 shows the situation at the start of the demonstration. The provider network

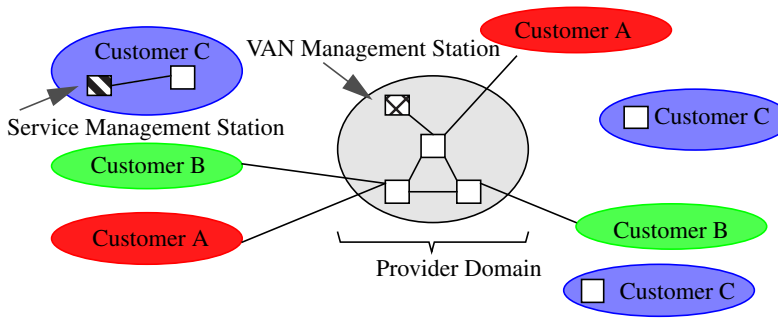
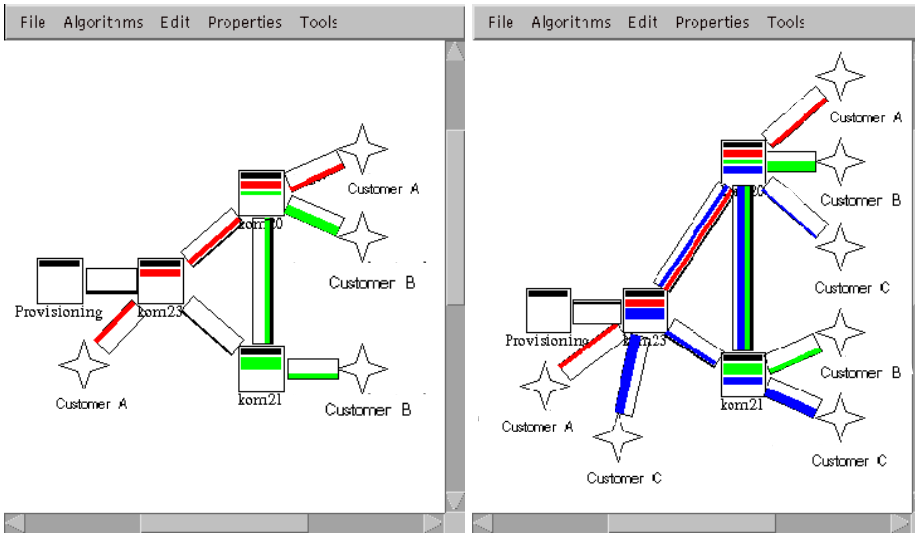



**Figure 6:** Situation at the Start of the Demonstration

consists of three active network nodes. The provider VAN management station is con-

nected to one of these nodes. Three customers are involved in the scenario. Customer A and B have two (active) customer premises networks each, and customer C has three. The Virtual Active Networks for customer A and customer B have been set up by the VAN management system. The view of the provider VAN management station at this point in the demonstration is displayed in Figure 7a. It shows the VANs for customers A and B and the management VAN. Further, customer premises networks are represented by a star labeled customer A, B, and C.

The VAN provisioning capability is demonstrated by setting up a new VAN for customer C. The process of provisioning a VAN includes the installation of Execution Environments on all three nodes, connecting the Execution Environments by setting up Virtual Links between them and connecting one end of a Virtual Link to the customer premises. The view of the provider management system after the VAN for customer C has been provisioned is shown in Figure 7b.




(a) (b)

**Figure 7:** VAN provisioning as displayed on the provider's VAN management station: **before (a)** and **after (b)** provisioning a new VAN.

The service management system in the domain of customer C has now the view of a VAN spanning over the provider domain and all domains of customer C. Figure 8 shows this view, which includes seven nodes. Three of them are in the provider domain and four in the domain of customer C. Note that, on this level of abstraction, nodes in the customer domain do not differ from those in the provider domain in terms of service management capabilities. The service management system has the view of a single active network, on which services can now be installed and supervised.

Figure 8 shows a window from the service management station of customer C. It includes a snapshot of the Execution Environment of a VAN node. In our current implementation, the service management system can display the configuration and the

state of Execution Environments in VAN nodes. The figure shows the buffers of the CPU scheduler, the memory, the in-bound, and the out-bound links. Three buffers are associated with the CPU scheduler: the default buffer for (active) packets, a second buffer for packets that belong to service management functions (e.g., filters for detecting specific events), and a third buffer for packets of a mechanism that routes the packets of the service management system. This is the basic configuration of an active network node, after the VAN has been set up by the provider and the service management system has been initialized by the customer. At this point, the service management system is ready to install specific network services and service management functions on the VAN.

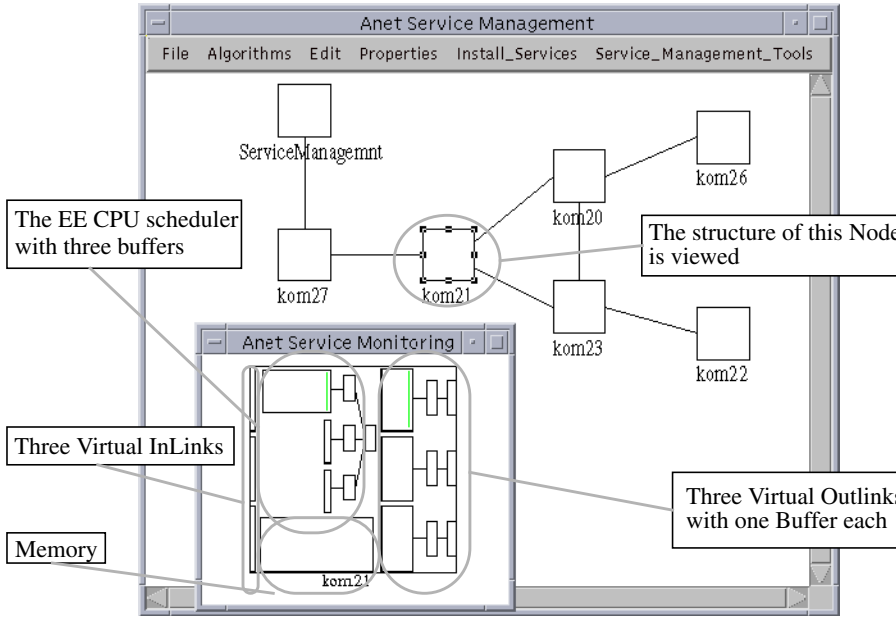



**Figure 8:** View of the Service Management Station after Provisioning a VAN for a Customer

### 6.2 Customer-Controlled Service Installation, Upgrade, and Supervision

On the ANET platform, we can demonstrate the installation, upgrade, and supervision of an IP service. Installing an IP service on a active network node is achieved by configuring a virtual router inside the node's Execution Environment. The service management system sends a sequence of active packets to the Execution Environment. Processing these packets results in installing an IP routing table, creating output buffers for the virtual out-bound links, setting up packet schedulers that operate on these buffers, installing function code for routing and management operations, configuring service-specific control parameters and management parameters, etc. After that, the IP service is initialized, which includes starting the routing protocol.

Upgrading the IP service to an IP service supporting several traffic classes with different QoS requirements is accomplished in our system by reconfiguring the virtual routers inside the Execution Environments. The service management system sends an

active packet to each Execution Environment of the VAN. The processing of this packet results in upgrading the packet classifier (to detect the class of a packet), setting up buffers for each traffic class, and substituting the packet scheduler with a scheduler for multi-class traffic.

Figure 9 shows the structure of the Execution Environment after installing an IP service and upgrading it to a multi-class IP service. Compared to Figure 8, the structure of the output-buffers has changed to contain two buffer partitions, and the CPU scheduler part has grown by two additional components, one for the IP routing protocol and one for the management of the multi-class IP service.

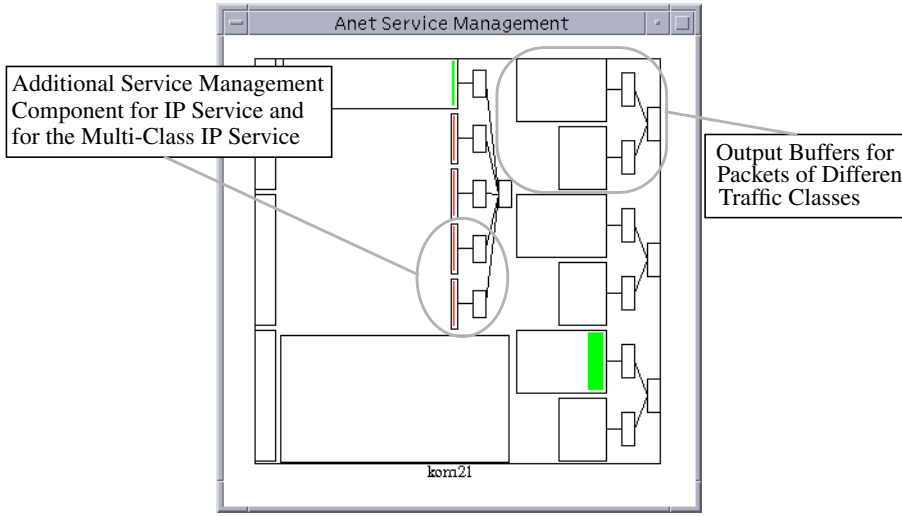



**Figure 9:** Node Structure after Installing an IP Service and Upgrading to a Multi-Class IP Service

In our implementation, the service management system can change the partitioning of the output buffers, by sending active packets to the virtual routers installed on the ANET platform. Further, it can monitor the buffer usage, by configuring the active management component to send packets back to the service management station in regular time intervals. This way, we can perform service management operations the same way as a customer would do while managing its IP service on its own premises.

## 7 Related Work

The networking community is currently putting substantial efforts into investigating the active networking approach. Additional motivation for this work and ongoing projects can be found in [2]. Two areas of current research are (1) designing Execution Environments for active network nodes ([6], [7], [4]), and (2) extracting application-specific functionality to be integrated into the network layer, such as, application-specific packet filtering functions and application-specific packet routing ([8], [9], [10]). An architectural framework for active networks is being developed by the AN Working Group [1].

One part of the Netscript project [11][4] deals with management of active networks. In that project, a platform for programming network services is being built. These services can be automatically instrumented for management purposes, and corresponding MIBs can be generated. During operation, services can be managed through those MIBs. Contrary to the Netscript project, our work leaves open the question of service instrumentation in an active network environment, but it focuses on a flexible framework for supporting interactions between customers and providers.

Research approaches in the area of *programmable networks* focus on developing interfaces that facilitate flexible service creation and resource partitioning in a telecom environment. This work generally centers around a programmable control plane for broadband networks [12][13][14]. While the current research in programmable networks clearly facilitates service instrumentation, it does not pursue service management aspects and does not deal with customer-provider interaction, as this paper does.

The Genesis project [15] brings up the notion of *virtual programmable networks* as the entity to bind resources to it, and they describe the life-cycle to install such a virtual programmable network. First, the Genesis approach is derived from the programmable networking (build on top of switchlets [12]), which concentrate the virtualization to the control plane. Our paper also includes the data path, which makes the network active on the data, control, and management plane. Second, the described life-cycle virtual networks, does not include any customer-provider interaction, as our paper does.

During our work we have identified several requirements to be met by an active networking platform for an effective realization of our management framework and the VAN concept. They include:

- Active packets from different Execution Environments have to be multiplexed onto a physical link to enable the abstraction of Virtual Links.
- Resources, such as CPU-cycles, memory, and link bandwidth have to be shared among different Execution Environments.
- An active network node has to prevent a customer from consuming more resources as he is entitled to.
- Access to memory has to be protected against unauthorized read and write.

High-performance platforms under development today lack at least one of these requirements. The ANN project [5] follows the approach of trusted code. The installation of code into a network node is performed by loading code from a trusted code server, which prohibits a customer from introducing any code into the node. Further, the CPU-cycles and link bandwidth are equally shared between different classes of flows, which restricts customizable resource allocation. Finally, multiplexing is achieved by using ATM on the physical link, which relieves the node operating system to perform the multiplexing.

The Switchware project [16] takes a language based approach. The memory access is controlled via a type-checked Programming Language for Active Networks (PLAN) [17]. Switchware does not support resource partitioning. Multiplexing is implicitly build into the active packet carrying the code to evaluate on intermediate active nodes. The executing code calls in a controlled way previously installed routines on the network node.

## 8 Discussion and Further Work

Rapid deployment of new services on an network infrastructure is the main driving force behind active networking research. In this work, we specifically focused on service provisioning and management in an active telecom environment and we showed the promising potential that active networking opens up in this area.

We identified the concept of a Virtual Active Network as the key abstraction in our framework. The VAN

- defines the object and level of customer-provider interaction,
- provides a basis for the provider to manage the resource consumption of customers in a safe way, and
- allows a customer to install, upgrade, and supervise a customer-specific service in a VAN spanning both the customer and provider domains.

We illustrated the above properties of a VAN by describing a series of demonstrations conducted on the ANET active networking platform.

We believe that our work opens up the way for further research. Obviously, efficient and flexible resource partitioning mechanisms need to be developed for sharing of CPU, memory, and link bandwidth in an active networking environment. Also worth pursuing are toolkits to support the design and implementation of active services. We have experienced that service management functionality is often similar for different active services. This similarity lets us conclude that a set of generic components can be build that will facilitate the realization of active services and their management applications.

## References

1. AN Architecture Working Group, “Architectural Framework for Active Networks,” K. Calvert (editor), 1998.
2. D. Tennenhouse, J. Smith, W. Sincoskie, D. Weatherall, G. Minden, “A Survey of Active Network Research,” IEEE Communications Magazine, Vol. 35(1), 1997.
3. M. Brunner, R. Stadler, “The impact of active networking technology on service management in a telecom environment,” IFIP/IEEE International Symposium on Integrated Network Management (IM ‘99), Boston, MA, May 10-14, 1999.
4. S. Da Silva, Y. Yemini, D. Florissi, “The Netscript Project,” ICC Workshop on Active Networking and Programmable Networks, Atlanta, 1998.
5. D. Decasper, G. Parulkar, S. Choi, J. DeHart, T. Wolf, B. Plattner, “A Scalable, High Performance Active Network Node,” IEEE Network, Vol. 13(1), 1999.
6. D. Weatherall, J. Guttag, D. Tennenhouse, “ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols,” IEEE Conference on Open Architecture and Network Programming (OPENARCH’98), San Francisco, USA, April 1998.
7. J. Smith, D. Farber, C. Gunter, S. Nettles, D. Feldmeier, W. Sincoskie, “SwitchWare: Accelerating Network Evolution,” Technical Report MC-CIS-96-38, CIS Department, University of Pennsylvania, May 1996.
8. S. Bhattacharjee, K. Calvert, E. Zegura, “An Architecture for Active Networking,” Proceedings of High Performance Networking (HPN’97), 1997.

9. U. Legedza, J. Guttag, “Using Network-level Support to Improve Cache Routing,” 3rd International WWW Caching Workshop, Manchester, England, June 1998.
10. S. Bhattacharjee, K. Calvert, E. Zegura, “Self-organizing wide-area network caches,” IEEE INFOCOM, 1998.
11. Y. Yemini, S. da Silva, “Towards Programmable Networks,” IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM’96), L’Aquila, Italy, 1996.
12. J. van der Merwe, I. Leslie, “Switchlets a Dynamic Virtual ATM Networks,” Fifth IFIP/IEEE International Symposium on Integrated Network Management (IM’97), San Diego, California, U.S.A., May, 1997, pp. 355-368.
13. A. Lazar, K. Lim, F. Marconcini, “Realizing a Foundation for Programmability of ATM Networks with the Binding Architecture,” IEEE Journal of Selected Areas in Communications, Vol. 14(7), September 1996.
14. J. Biswas, A. Lazar, J. Huard, K. Lim, S. Mahjoub, L. Pau, M. Suzuki, S. Torstensson, W. Wang, S. Weinstein, “The IEEE P1520 Standards Initiative for Programmable Network Interfaces,” IEEE Communications Magazine, Vol. 36(10), 1998.
15. A. Campbell, M. Kounavis, D. Villela, H. De Meer, K. Miki, J. Vicente, “The Genesis Kernel: A Virtual Network Operating System for Spawning Network Architectures,” IEEE Conference on Open Architecture and Network Programming, (OPENARCH’99), 1999.
16. S. Alexander, W. Arbaugh, M. Hicks, P. Kakkar, A. Keromytis, J. Moore, C. Gunter, S. Nettles, J. Smith, “The Switchware Active Network Architecture,” IEEE Network, Vol. 12(3), May/June 1998.
17. M. Hicks, P. Kakkar, J. Moore, C. Gunter, S. Nettles, “Network Programming with PLAN,” IEEE Workshop on Internet Programming Languages, May 1998.