Virtualized Application Networking Infrastructure

H. Bannazadeh, A. Leon-Garcia, K. Redmond, G. Tam, A. Khan, M. Ma, S. Dani, and P. Chow

Electrical and Computer Engineering Department
University of Toronto
10 King's College Road, Toronto, ON M5S 3G4 Canada
hadi.bannazadeh@utoronto.ca

Abstract. In this paper, we present a new platform for experimenting with networked systems and distributed applications called Virtualized Application Networking Infrastructure (VANI). This infrastructure is designed as a converged communications and computing infrastructure that would facilitate operation of an open applications marketplace. VANI enables introduction of new network architectures that require in-network (hardware-accelerated) content processing and storage. We describe the VANI architecture and the resources it provides. VANI has two main planes; control and management plane, and applications plane. VANI resources are virtualized and made available to the researchers and application providers through a service-oriented control and management plane. The current VANI resources are processing, storage, networking and various software-based resources. VANI also includes a new reprogrammable hardware resource that enables experimenting with hardware-based or hardware-accelerated networking algorithms and protocols. We present performance evaluations of this reprogrammable hardware resource, and the VANI virtual networking mechanism. The results show that by using the reprogrammable hardware resource, researchers can evaluate high performance and high throughput networking algorithms as easily as evaluating software-based networking algorithms.

Keywords: Networking Testbed, Network Architecture, Service-Oriented Architecture.

1 Introduction

In the past few years, the idea of clean slate network design has been circulated in the networking community and there have been several proposals for introducing new network architectures and protocols [1,2,3]. One of the major obstacles in introducing new network architectures was and still is experimentation with proposed network architectures in a large scale environment and possibly with massive numbers of end users. To address this problem, there have been several initiatives to build large scale testbeds for networking research.

GENI [4] is one of these initiatives that tries to create a testbed by federating different testbeds such as PlanetLab [5,6] and Emulab [7] on top of a research dedicated network. GENI is still in the design and development phase, but currently it follows a slice-based architecture [8], and different testbeds would be able to connect to each other through

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 363–382, 2011.

GENI wrappers. The exact communication protocol between the GENI wrapper and the testbed is left to each testbed's control plane and currently there are a few major control planes that are trying to federate using the wrappers.

Probably among these testbeds PlanetLab [5] is the most developed. PlanetLab provides edge hosts on Internet and implements a slice-based architecture using the Linux vServer [9] technology. PlanetLab, however, does not have a clear solution for experimentation with new layer three protocols, and it's not clear how it would facilitate building high scale new routers that would need hardware-based acceleration.

In Canada, there is a research dedicated optical network called CANARIE [10] that provides light paths connecting universities and research centers across Canada. CANARIE has sponsored design and development of a User Controlled Light Path [11] (UCLP) software that enables researchers to configure CANARIE network elements through Web Services (WS) interfaces on-demand.

Another major initiative is FEDERICA [12] in Europe that is under development through federation of several research network platforms in Europe such as i2CAT in Spain and HEAnet in Ireland. FEDERICA uses WS-based UCLP software for creating on-demand virtual networks atop of involving test platforms.

Another project for experimentation with lower layer protocols and networking algorithms is NetFPGA [13]. NetFPGA is a PCI card with a Field Programmable Gate Array (FPGA), and four Gigabit Ethernet interfaces that could be used for developing networking components such as a layer three router or a hardware accelerator.

In this paper, we present a new testbed for networking experiments and networked systems. This testbed is different than the above mentioned projects in several aspects. It benefits from a novel architecture for control and management functions capable of managing various hardware-based and software-based resources. It also allows experimenting with new network architectures that require in-network content processing and storage capabilities. Moreover, it includes a new high performance and high throughput hardware resource that makes experimentation with hardware-based or hardware-accelerated networking algorithms and protocols as easy as experimentation with software-based protocols.

Our vision in designing this testbed was to develop a converged computing and communications infrastructure to support an open applications marketplace. We investigated architectural aspects of this application-oriented network and presented a proposal in [14]. We also investigated autonomic management issues and proposed an approach using virtual networks in [15].

The essential aspects to enabling the above application-oriented environment are: 1. Service-oriented application creation; 2. Infrastructure as a Services methods for configuring and scaling resources to support applications; 3. Virtualization of physical resources.

Based on this view of an application-oriented network, we began the development of a testbed that would allow university researchers and application providers to develop new networked systems and networking architectures. This testbed, Virtualized Application Networking Infrastructure (VANI), allows the creation of virtual networks of computing and communications resources. A VANI node consists of resources such as processing, storage, networking, and programmable hardware. A service-oriented

control and management plane allows VANI nodes to be interconnected into virtual networks to support applications operating in the applications plane.

In the rest of this paper, we describe the main requirements in VANI design, and its architecture and main components. Also, we explain how our design would satisfy the requirements. Moreover, we present the performance evaluations on the developed resources for this infrastructure including a virtualized reprogrammable hardware resource that enables hardware-based experimentation of networking algorithms and protocols.

2 VANI Design Requirements

Virtualized Application Networking Infrastructure (VANI) is a testbed that allows university researchers and application providers to utilize its internal resources to rapidly create and deploy networked systems, and to even experiment with new layer three protocols. Although the underlying concepts of the VANI testbed comes from our view on Application-Oriented Network [14], but networked systems running in VANI environment could follow any architecture in any networking layer. The only limitation that the researchers are facing in VANI is that their experiments should run on top of Ethernet as their layer two. Next, we describe the main requirements in designing VANI.

The VANI design follows some basic requirements (figure 1) The first requirement for VANI testbed is that it should allow experimentation for future network architectures that might not fit into the traditional layer three definitions. Currently networks are primarily responsible for delivering raw data but in future it would be possible for future network architectures to shift-up the network tasks to new functionalities that might be required by emerging applications. Among these functionalities could be the task of content-delivery in addition to data-delivery (such as the network architecture discussed in [14]) that would imply having content processing and storage functions in the infrastructure.

The second main requirement was to allow researchers to experiment with new layer three protocols (as in the traditional definition of L3) instead of the current Internet

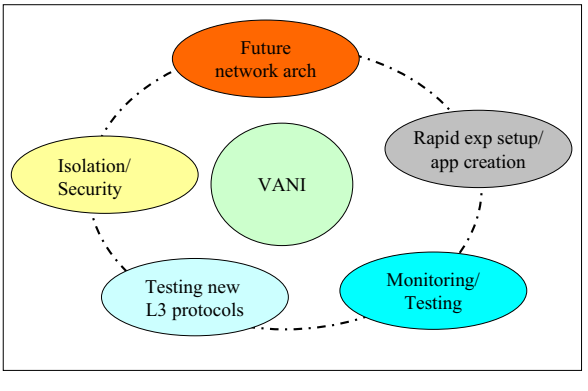

Fig. 1. VANI design requirements

Protocol. To do so, we designed the testbed assuming that everything above layer two could be redesigned and experimented with, and we chose Ethernet protocol as the basis of our layer two design.

Another main requirement in the testbed is to be able to setup experiments or create new applications rapidly using already developed and ready to use components that could be accessed through open interfaces. These components could be the virtualized resources such as processing, low-latency hardware processing, and accelerator nodes, or software components such as event processors that are used in many experiments for data gathering and analysis. This requirement could be satisfied through the use of the SOA technologies and standards that could allow flexible and dynamic composition of reusable service components.

The fourth main requirement was to provide an isolated and secure environment for researchers to carry on their experiments and develop their networked applications. This requirement has to be satisfied at different levels such as traffic separation, bandwidth allocations, storage access, secure access to the physical resources, and isolation between different physical resources. The fifth main requirement was the monitoring and debugging mechanisms. In our design, we envisioned powerful complex event processing components that could be customized to gather and analyze test and debugging data for each experiment separately as well as for the testbed itself.

2.1 VANI Architecture

Based on these main requirements, we designed a two plane architecture for our platform: control and management plane (VANI-CMP) and applications plane (VANI-AP).

VANI-CMP is responsible for virtualizing physical resources and allocating them to the researchers and application providers. On the other hand, researchers deploy their applications and experiments in the VANI applications plane (VANI-AP). Applications operating in the applications plane can have their own architecture inside an applications plane slice that is created by VANI-CMP.

For example, an experiment/application could be a new layer three protocol that covers OSI layer three and four functions, could replace TCP/IP layer, or could be

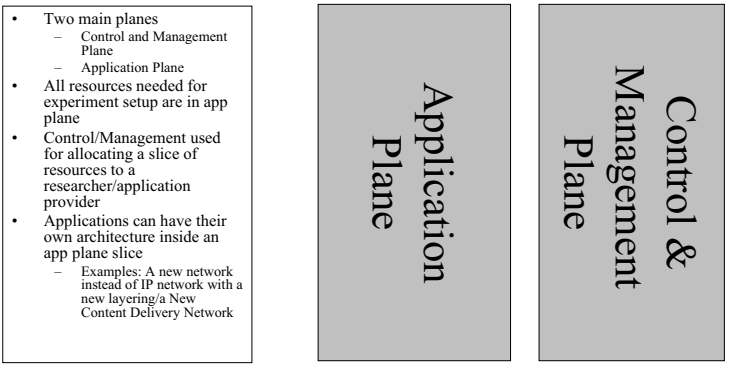

Fig. 2. VANI architecture

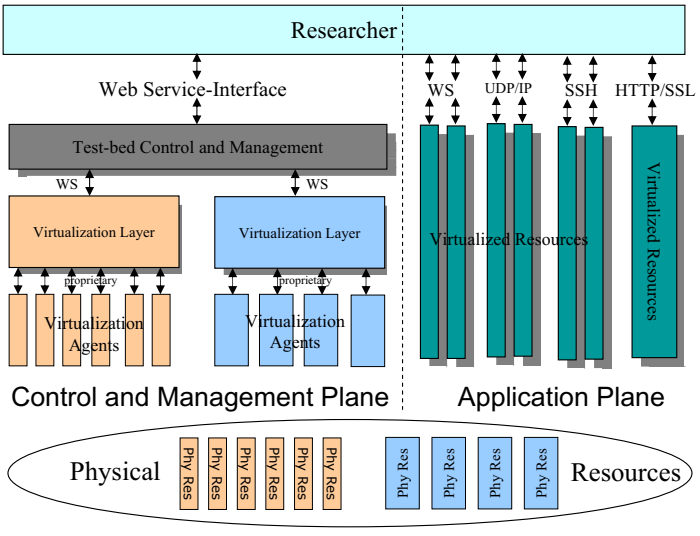

Fig. 3. Researcher interaction with VANI planes

a new content delivery network. Figure 2 shows this architecture including its two planes.

All virtualized resources and service components that can be used by researchers for creating an application reside in the applications plane. Researchers can ask for these resources through the testbed control and management plane and then they can directly connect to the virtualized resource in the applications plane through any resource specific protocol such as HTTP, UDP/IP, or ssh.

For example, a user can ask for uploading or downloading of a file to the storage service through the control plane, and then if permitted by the control plane, it has to directly contact the storage file service using HTTP/TLS connection and download or upload its files.

VANI control and management plane (VANI-CMP) is responsible for allocating testbeds resources to the researchers. Researchers ask VANI-CMP for a resource using VANI-CMP's Web Service interface. WS interface is chosen due its universal acceptance for SOA, and the abundance of available tools for orchestrating and creating new applications using independent Web Services.

After receiving the requests for resources from a researcher, VANI-CMP authenticates the researcher and authorizes its request and then sends the request to the resource virtualization layer. The resource virtualization layer is the layer which abstracts a physical resource and offers it as a service to the control and management layer. If the allocation is successful, VANI-CMP records the allocation, and replies back to the researcher with a successful return result.

VANI-CMP also programs and releases the resource whenever an authorized researcher wants to do so. Figure 3 depicts the logical view of the VANI testbed and how a researcher interacts with VANI planes.

2.2 Current Physical Resources in VANI (VANIv1 Resources)

Currently, several physical resources have been virtualized and made available to VANI users. In [16], the design and development details of these resources have been presented, and here, we briefly overview these resources and type of functionalities that they can offer to researchers.

In VANI all physical resources are virtualized. Through virtualization, we separate applications from their underlying physical resources. To do so, we developed a virtualization layer and virtualization agents for each physical resource as shown in figure 4. The task of the virtualization layer is to coordinate the system wide virtualization of a resource and to expose the resource as a service component with Web Service interface to the rest of the system, and the agents task is to launch or destroy the virtual resources on top of each physical resource.

The first physical resource that we have virtualized is the reprogrammable hardware resource. To develop this resource we have used BEE2 boards [17]. Each BEE2 board has four high-end Xilinx Field Programmable Gate Arrays (FPGA) each connected to four 10GE interfaces. We have virtualized all four FPGAs in a BEE2 board so that a researcher could ask for one or more FPGAs and program it as s/he likes.

Researchers can ask for an FPGA through the control plane and then program it, configure it, or release it. They also have access to the libraries for controlling the 10 GE interfaces and some other commonly used hardware blocks such as DDR2 memory modules. After programming an FPGA, a researcher can directly connect to the FPGA through the 10GE interfaces according to whatever protocol designed for that FPGA. For example, a researcher can use one FPGA or all four FPGAs to develop a layer three router with 4x10GE ports or 16x10GE ports, or a content-based routers that routes packets based on the packets payload rather than their headers. We present the performance evaluation results for this hardware resource in the performance evaluation section of this paper.

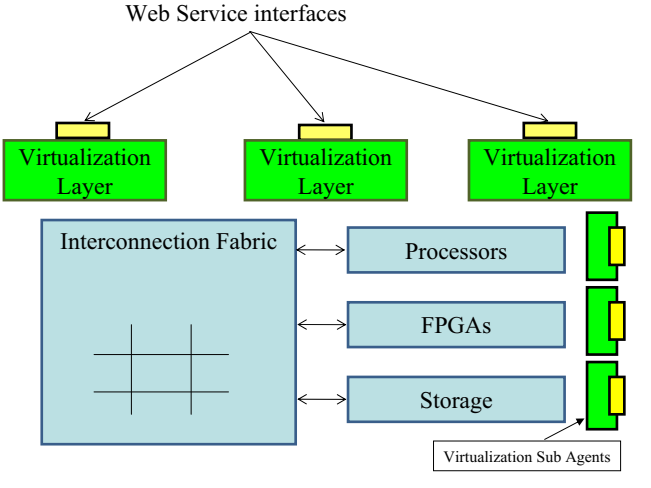

Fig. 4. Virtualizing physical resources in VANI

Another physical resource in the VANI testbed is the processing resource. The processing service is developed based on Linux vServer [9] technology. Linux vServer is an OS-level virtualization software that creates a virtual processing node on top of a Linux kernel. Researchers are able to get a processing resource through VANI-CMP, and release it whenever they wish to do so. Once a virtual processing node is allocated, the researcher can directly ssh to the node. Researchers are also able to program the virtual processing node with a specific image, create an image of their own, and save it on the storage service, and share it with others or program other virtual nodes with that image.

We have also virtualized the internal fabric of the testbed for creating virtual networks. The internal fabric consists of a set of high capacity Ethernet switches that are able to isolate traffic between different applications and experiments by creating separate virtual LANs. Moreover, it allows different experiments to intercommunicate by creating shared virtual LANs that all have access to. This resource, together with the processing resource, enable VANI to guarantee the bandwidth for an experiment. Later in the bandwidth guarantee section, we will discuss this feature in more detail.

The gateway and bridge resource is another developed resource that enables communication between different VANI nodes. If one of the resources in VANI needs to be accessible from the Internet or from a resource in another VANI node, it can ask for a public address through the gateway service and get an address for duration that the external access is needed. The researcher can release the public address when it is no longer needed.

Th bridge service is used for experiment involving new layer three protocols on top of Ethernet network. Using the bridge service, a researcher can send and receive layer two Ethernet frames to any other VANI node, and hence, would be able to develop and test new layer three protocols over a wide area network. This functionality would only be available if the VANI nodes are connected using a wide area Ethernet network. We will discuss this case later in more detail.

Another physical resource developed for VANI is the storage resource. Storage resource is implemented on a set of distributed file servers that emulates one big storage server. Researchers are able to connect to the storage service through VANI-CMP and then directly connect to a file server for uploading and downloading files. All the direct communications to the file servers for uploading and downloading files are done over a secure HTTP/TLS connection. Researchers can use this service to store images for programming other resources such as processing resource, and reprogrammable hardware resource, and they can also share file with other researchers through this service.

2.3 Example: Requesting a Resource in VANI

Figure 5 shows a sample message exchange scenario between a researcher, the VANI control and management plane and physical resources inside a VANI node. A researcher starts requesting for a resource by invoking the getResource operation of the VANI-CMP WS interfaces. In that request, the researcher includes the type of resource, the duration and number of required resources.

VANI-CMP authenticates and authorizes the request and forwards the request to the resource. All resources in the testbed expose their operations to VANI-CMP through a

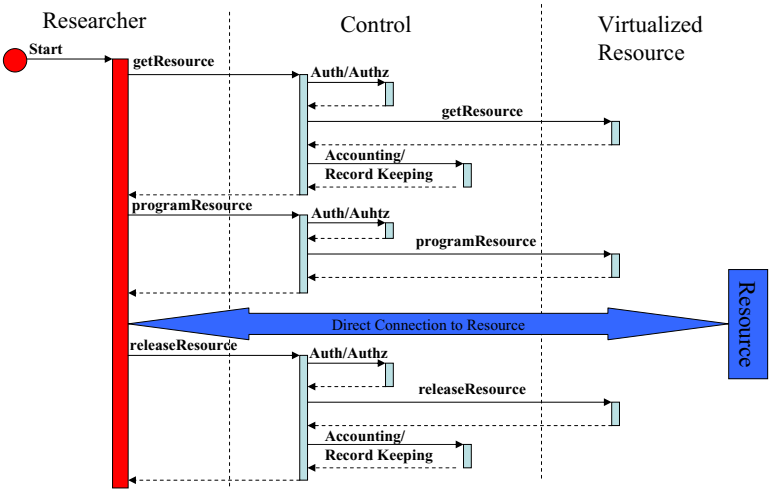

Fig. 5. A sample interaction between a researcher and VANI

generic WSDL interface. This makes it possible to easily extend the types of resources and services in the testbed without changing the control and management software.

The resource responds back to the control plane request with a success result, and a Universally Unique IDentifier (UUID) for the resource. The control plane stores this returned UUID and passes it to the researcher. The researcher can program the resource identified by returned UUID, and release it at a later time.

In the next section, we delve into the control and management design and we describe its main functionalities in detail.

3 VANI Control and Management Plane (VANI-CMP)

VANI-CMP is responsible for performing Authentication Authorization Accounting (AAA) operations and allocates resources to the researchers and application providers. In addition, it performs user management functions, and stores and manages the testbed configuration data. It also has a registry for all services and resources that can be used by researchers for creating a new application or experiment setup. Researchers can register new types of resources in this registry, and make them available for use by other researchers.

VANI-CMP is designed based on service-oriented design concepts and developed using SOA technologies. VANI-CMP is developed in Business Processes Execution Language (BPEL) [18] and deployed on an Enterprise Service Bus (ESB) [19]. Similar to other virtualized resources and services in the testbed, all internal components and functions of VANI-CMP have also been developed as independent service components, and are accessed through Web Services interfaces.

The use of ESB and Web Services enables VANI-CMP to be easily extended in functionality and accessed through other types of interfaces in the future. This design choice

also enables independent development, testing, and redeployment of internal functions of VANI-CMP such as AAA operation, configuration management, etc. Moreover, the use of BPEL language for VANI-CMP enables a high level description of the VANI control and management operations. This enables rapid and easy modifications of the control and management logic.

In the next subsections, we examine each of the functionalities of the control and management plane and we describe the design steps and interfaces of each of the modules.

3.1 User Management

Three concepts are used to manage users in VANI: application plans, service levels, and plan administrator levels. Application plans are used to show different experiments and to organize resources and resource usage in each experiment. When booking a resource, the researcher must specify which plan (experiment) the resource is being booked on. Any researcher belongs to a service level which governs what control operations s/he is allowed to call and also how much of each resource s/he is allowed to book. Custom service levels may be designed for specific users in order to maintain flexibility. Lastly, plan administrator levels are used to govern access to certain resources. Resource users will be granted specific levels of access defining their ability to release, program, save, etc.

3.2 Authentication Authorization Accounting

The control software is responsible for handling authentication of users. All operations in the control plane require users to provide credentials. Currently, credentials are in the form of a user name and password combination however the implementation allows this to be easily changed. On every call to the control software, the user is authenticated and a check is made to ensure that the user has the rights to execute the requested operation. In addition to authentication, the control software is responsible for authorizing access to resources. Every access to a resource consists of two checks, ensuring the resource belongs to the user, and the user has the rights to manipulate the resource as requested.

In order to prevent outsiders from directly accessing resources and bypassing the control plane, all requests to resources require credentials known only to the control plane. This credential is generated when resources are initialized.

The control software keeps a record every time a resource is booked or released. This keeps an account of which resource was used by which user (on which plan) and for how long as well as all resources currently in use. Resources are identified by a UUID generated by the resource and passed back through the control plane.

3.3 Resource Allocation

Resources are booked through the control plane whether the user is a researcher or an application provider building a resource on top of another. Users provide their credentials and specify which resource they wish to book (on which VANI node) and the plan

```
<xsd:element name="getRequestGenericContents">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="internalIP" type="xsd:string"></xsd:element>
        <xsd:element name="uuid" type="xsd:string"></xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

Fig. 6. A sample schema for generic XML content in a getRequest response message

to which the resource will belong. The control plane ensures the user is allowed to book the resource and determines the location (WSDL address) of the resource in the network. A getResource request is then made to the resource. The resource does not know who is requesting the resource as this information is hidden by the control software. If successful, the resource will return a UUID identifying the resource as well as any other relevant data which is then passed back to the user. The UUID is used by the control plane for accounting purposes.

3.4 Generic Resources/Registration

New resources can be made available dynamically in the control plane through a registration operation. The new resource must consist of a unique name, a service name, a port name, one or more WSDL addresses, and optionally a JNLP address for the resources GUI. The service and port name are used to create an end point reference which is assigned to the partner link when the resource is to be accessed. The resource may have multiple WSDL addresses if there are different instances of the resource on different VANI nodes. The control software will select the appropriate address depending on which node the user is attempting to access. Lastly, a JNLP address may be included which allows resource creators to design and deploy their own GUI using Java web start technology [20].

In order for resource creators to dynamically add new resources to the control plane, it is necessary to use a generic WSDL interface for all resources. The main objective with the generic interface is to provide a template that makes creating resources easy while providing flexibility. This is accomplished by providing a number of operations, messages that are common between many resources such as get, release, and program. To maintain flexibility, each operation contains an optional XML string which can be used to customize data that is passed in and out (figure 6). Furthermore a generic operation is included in the WSDL which can be used to include operations not already included in the template.

4 Security in VANI

One of the basic requirements in VANI design was to make sure the experiments are done in a secure and isolated environment from the other applications and experiments. To create this secure environment we have to consider security issues in various parts of the system architecture.

The first part is to secure the communications between the researchers and VANI-CMP. In VANI all communications between these two entities are encrypted using secure SSL connections and WS-security specification. To do so, each researcher has to share his/her public key with VANI (and vice versa). On top of that VANI-CMP authenticates the researchers and application providers using the credentials provided in all transactions, and then, authorizes the researcher's access level to the resource.

The second part is the communications between the resources and VANI-CMP. These communications have also been encrypted. Moreover, credentials only known to the resource and VANI-CMP are included in all communications from VANI-CMP to the resources.

All internal traffic within one experiment is separated from other experiments using tagged Ethernet VLANs. By proper configuration of the testbed internal fabric resource, we are able to isolate these tagged VLANs from each other. This case is discussed in more detail in the bandwidth guarantee section.

Communications inside the applications plane, internal to one experiment, or coming to and from that experiment could be encrypted or not depending on the experiment, and therefore it is outside of the scope of the VANI design. This allows researchers to freely design and develop new encryption and decryption algorithms in different layers inside their application plane slice.

5 Bandwidth Guarantee in VANI

In order to make sure that one experiment cannot undermine another experiment's capability to send and receive traffic, we need to have a bandwidth guarantee mechanism in place. Likewise, for communications between different VANI nodes, there should be a rate guarantee in place so that a distributed experiment could have a guaranteed access to the available bandwidth.

Since all communication in VANI is carried over the VLAN tagged Ethernet frames, an Ethernet rate limiting mechanism in processing nodes has been developed. By doing so, we limit the rate in which each virtual processing node sends and receives traffic from/to another virtual processing nodes inside a VANI node.

Also the gateway and bridge service controls the rate in which an experiment sends/receives traffic to/from the VANI wide area network. The wide area network that is used to connect the VANI nodes would be a research-dedicated network like CANARIE [10] that can guarantee the aggregated traffic to/from the VANI nodes. If the wide are network was able to provide dynamic and on-demand bandwidth allocation, VANI would be able to use this functionality whenever an experiment asks for sending/receiving traffic to/from the wide area network. VANI nodes could also be connected to the public Internet network, however, bandwidth could not be guaranteed for the experiments in this case.

To request a bandwidth guarantee in VANI, a researcher can specify the bandwidth requirements of a virtual processing node in the resource get request. Likewise, a bandwidth requirement can be specified when access to the VANI wide are network is requested. The virtualization layer in VANI control and management plane makes sure that the specified requirements are met when allocating virtual resources to the experiment.

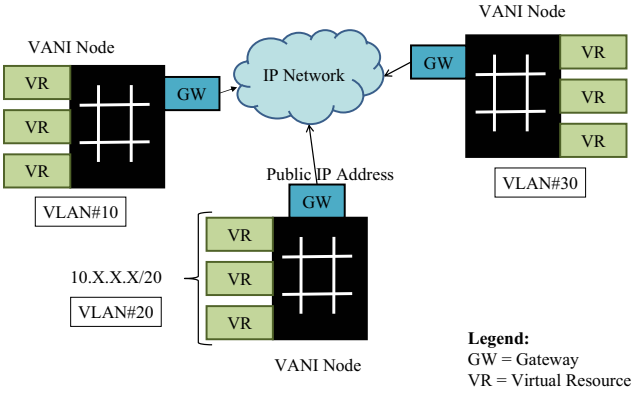

Fig. 7. Connecting VANI nodes in IP layer

5.1 Interconnecting VANI Nodes in IP Layer

Figure 7 shows how we can set up an experiment or create a distributed application across a wide area IP network. In this setting, all resources inside an experiment in a VANI node get a local IP address in the range of 10.X.X.X. All resource could send traffic to the wide are network using the NAT functionality implemented in the gateway service (shown as GW in figure 7). It is possible to put multiple gateways in place and direct outgoing traffic to different gateways to avoid bottlenecks in the system.

On the other hand, if a resource needs to be accessible from the wide area network, the researcher can ask the gateway service for a public address/name, and the gateway service redirects all traffic to that public address to the resource's internal IP address/VLAN.

5.2 Interconnecting VANI Nodes in Ethernet Layer

Figure 8 shows an Ethernet connected VANI. Ethernet connected VANIs use the bridge service instead of the gateway service to interconnect. Inside a VANI node, all resources

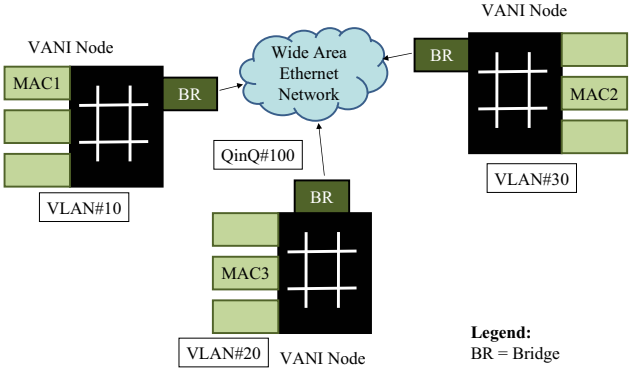

Fig. 8. Connecting VANI nodes in Ethernet layer

in an experiment communicate using a specific VLAN which is unique to the VANI node. If an experiment needs to operate across multiple VANI nodes (for instance, to test a new layer three protocol), the VANI wide area network has to be able to transfer Ethernet frames. In this case, a unique Q-in-Q tag [21] would be assigned to the experiment. The bridge service would be used to re-frame the internal tagged Ethernet frames to the wide are Q-in-Q frames and the destination bridge would do the reverse operation, and deliver the Ethernet frames to the destination MAC/VLAN in the destination VANI node.

Since Q-in-Q tagged Ethernet frames might not be available in a wide area network, we are able to define public MACs that can be used for redirecting traffic to an internal MAC/VLAN by the bridge service. This functionality would enable any other Ethernet-based experiment to send Ethernet frames to a resource in another experiment through the bridge service.

5.3 Experimentation with L3 Protocols

Figure 9 shows how the testbed could be used to test a new layer three protocol in a a large scale and distributed environment using proxy nodes. In this setting, the new L3 protocol is tunneled within IP payload to a resource inside a VANI node, and then that resource strips off the IP header and feed the new L3 packet over the VANI wide are Ethernet network.

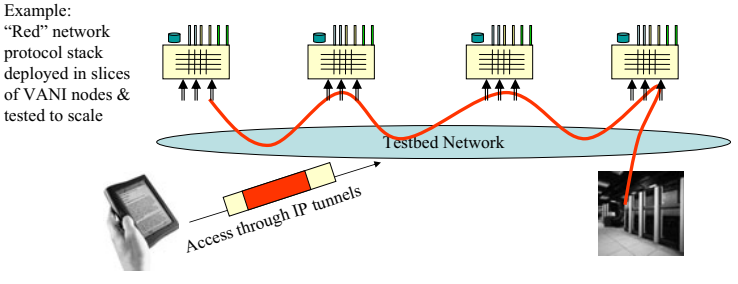

Fig. 9. Large scale experimentation with new L3 protocols

6 SW-Based Resources in VANI

One of the main contributions in our testbed control and management plane is that we could encapsulate any software or hardware resource in our testbed as a service. To do so, the resource can be virtualized, and abstracted as a service component that follows a generic resource WSDL template. Then it can be registered into the control plane and made available to other researchers. Details on how this task can be accomplished have been discussed in the control and management plane section in this paper.

Examples of such resources as a service are any hardware function or resource that could be reused in different applications and experiments such as hardware accelerators for encryption, decryption, content conversion, and content compression/decompression.

Also other reconfigurable hardware modules such as NetFPGA could be virtualized and offered to the researchers on an on-demand basis.

Other types of processing nodes could also be offered to the researchers as a resource. For example, Amazon Elastic Computing Cloud (EC2) nodes [22], GENI virtual processing nodes, VMWare-based virtualized processing nodes [23], or Graphics Processing Units (GPUs) could be controlled and managed by VANI-CMP.

Moreover, software services such as database service, BPEL orchestrator engine and Complex Event Processing (CEP) engine, could be developed and/or deployed on top of current virtual resources and made available to the researchers through VANI-CMP. Currently, we have developed and deployed several software-based resources as service components in VANI including a database service, BPEL orchestrator engine, and a sensor service.

7 Federation with GENI

GENI is an initiative to create a large scale experiment through federation between different testbeds. Federation in GENI is done using GENI wrappers. A GENI wrapper is developed for each testbed and testbeds could connect to each other through them. In VANI, we developed a wrapper for control and management plane, and through that we invoke GENI wrapper operations to get a node on any GENI testbed. We tested our wrapper with PlanetLab GENI wrapper and managed to obtain a PlanetLab processing node through our VANI-CMP.

In VANI, researchers are able to get a PlanetLab processing resources using VANI generic resource template. Since PlanetLab does not support storage service, and also does not support other VANI requirements such as processing and bandwidth requirements, access to PlanetLab processing resources would not support these functionalities. Figure 10, shows the structure of interconnection between VANI and PlanetLab

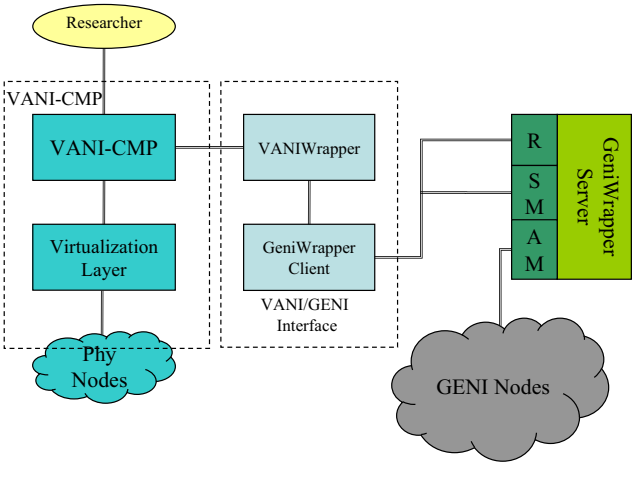

Fig. 10. Connecting VANI to GENI

through the GENI wrappers. Currently, we are in the development phase of offering VANI resources to GENI researchers through the VANI wrapper.

8 A VANI Node

A VANI node is composed of the resources described in this paper, their corresponding virtualization software, control and management software, and the storage service. A VANI node can be totally deployed on a computer cluster composed of normal computing blades, and manageable Ethernet networking elements. The basic resources in a VANI node are the processing resource, the storage service, and the fabric service for the network virtualization that are deployed on a computer cluster.

All other resources and the control and management software are deployed on these basic services. In addition, all other software-based resources, and the virtualization layer for resources like reconfigurable hardware resource, and the VANI wrapper for connecting to GENI testbeds are also deployed on these basic resources.

The only elements that cannot be found in a normal computer cluster are the reconfigurable hardware resources, the gateway and bridge services, and required 10GE Ethernet switches. These resources are also co-located with the computing cluster to provide the WAN connectivity and to enable running experimentation with the reconfigurable hardware resource.

9 Performance Evaluations

Up to now, we presented the VANI architecture and we discussed different aspects of its design. To find if the currently developed resources can meet VANI design requirements, we performed several experiments on those resources. In this section, we present performance measurements on two key physical resources that have been virtualized and offered to the researchers in VANI. The first one is the reprogrammable hardware resource, and the next one is the processing resource. Our main focus in this part would be to see if we could guarantee the promised quality of service to the researchers that use these resources in their experiment.

9.1 Reprogrammable Hardware Resource

By introducing a virtualized and reprogrammable hardware resource in VANI, we enable researchers to test new networking algorithms and protocols using high performance and high throughput hardware resources. To do so, we virtualized BEE2 boards developed in the University of California at Berkeley. A BEE2 board consists of one controlling FPGA, and four high capacity Xilinx Vertex-II FPGAs (figure 11) that can be programmed by users. Each FPGA has four 10GE interfaces, and 4 GB of memory.

In VANI, a researcher can get a set of FPGAs on a BEE2 board, and can ask for on-board inter-chip communication channels which can carry up to 5 GigaBytes per second (GBps). The detailed design of BEE2 virtualization system and introducing it as a resource in VANI can be found in [16]. Here, we present the performance measurements on this resource. The parameters of interest are the programming time of the

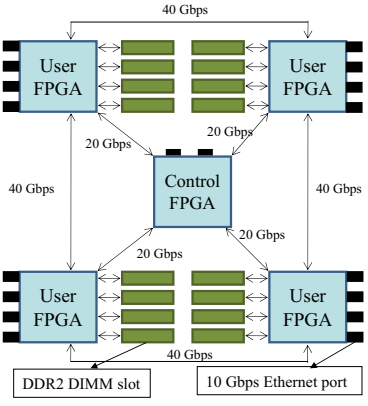

Fig. 11. Reprogrammable Hardware (BEE2 Board)

FPGAs through the virtualization software as well as the speed with FPGAs can send and receive data.

The first parameter is the time in which a researcher can program an FPGA through the testbed control plane. Also, we would like to know how this time would change if four researchers want to program all four FPGAs concurrently. To do so, we developed a bitstream that initializes all 10GE interfaces on the FPGAs and starts sending a burst of UDP/IP packets on one of its 10GE interfaces, and we programmed FPGAs through VAN-CMP using the generated bistream for several times. Table 1 shows the average maximum programming time that programming one, two, three, and four FPGAs take. As can be seen, it only takes 30 seconds on average to program an FPGA in the case where all four FPGAs are programmed concurrently, and this time is around 11 seconds if only one FPGA is programmed at a time.

This fast programming time allows a researcher to get an FPGA with four 10GE interfaces in less than a minute, and to run an experiment and return the FPGA back to the VANI resource pool as soon as it's not required.

The next experiment that we performed is to measure the speed with which the FPGAs can send and receive traffic. To do so, we developed a traffic generator using Verilog hardware description language, and we started sending traffic from one 10GE interface to another 10GE interface on the same FPGA, and we recorded the maximum bandwidth that we could receive in the hardware resource. We also compared this with the traffic statistics gathered by the Ethernet switch connected to the FPGA. We repeated this experiment several times and were able to send and receive Ethernet frames to the rate of 1GBps, which is equal to 8Gbps. The reason that we could not send more traffic is the 8/10 bit encoding mechanism for 10GE-CX4 interfaces, and 8Gbps is the

Table 1. Average maximum FPGA programming time

FPGAs	1	2	3	4
Programming Time (s)	11	17	24	30

maximum achievable traffic rate per port on a BEE2 board. In our measurements, this rate did not change if all ports started sending and receiving traffic at the same time since separate internal modules are controlling each port. This experiment shows that one FPGA alone can send and receive 32Gbps traffic. If a researcher get all four FPGAs on a BEE2 Board it is possible to send/receive traffic in the rate of 4x32=128Gbps.

We have used this reprogrammable resource in developing the high capacity gateway and bridge service for VANI, and we have developed a bandwidth control mechanism on this resource that controls and guarantees the rate at which one experiment could send and receive traffic to/from a wide are network. In the future, we will present our design for the gateway and bridge service, and we will present our performance measurements for this service as well.

9.2 Processing Service and Network Virtualization

Another main physical resource that we have virtualized is the processing service that uses Linux vServer software. There have been studies on processing virtualization techniques [24], and also specifically on Linux vServer [9]. Linux vServer performance evaluations show that this virtualization module has a very low overhead on overall system performance.

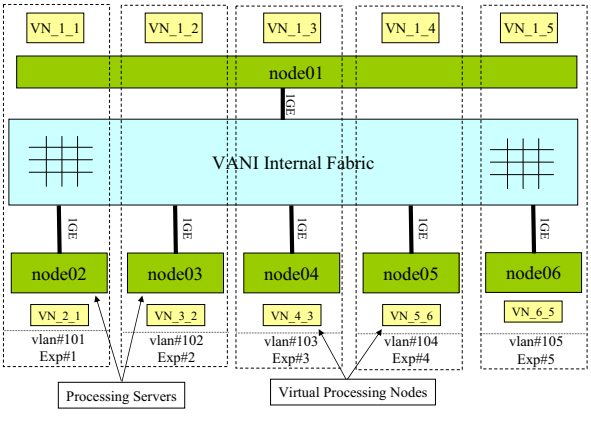

Fig. 12. Traffic measurement experiment topology

However, since we are also doing network virtualization in addition to the processing virtualization, we conducted two more experiments that were necessary to show that virtual processing nodes can have guaranteed access to the VANI network.

In our experiment, we virtualized cluster blades with dual Xen 1530 CPUs and 2GB of RAM and one 1GE interface. The Linux kernel version that we used was 2.6.16, and we used vServer 2.3.2. patch. The developed virtualization layer allows up to ten virtual nodes on a physical node. For this experiment, we initialized and launched 5 virtual nodes on a node named node01. We also launched 5 other virtual processing nodes on five separate servers with same capabilities described for node01. These nodes are

Table 2. UDP/TCP traffic measurements in MBytes per second (MBps)

node01 from/to	UDP	UDP (rl)	TCP	TCP(rl)
node02 (12.50MBps)	24.5/24.3	12.4/12.4	15∼35/24.7	12.3/12.3
node03 (18.75MBps)	24.5/24.3	18.8/18.8	15∼35/24.3	18.4/18.4
node04 (25.00MBps)	24.5/24.3	25.3/25.3	15∼35/24.1	24.8/24.6
node05 (31.25Mbps)	24.5/24.3	31.7/31.6	15∼35/22.1	31.3/31.1
node06 (31.25Mbps)	24.5/24.3	31.7/31.6	15∼35/23.2	31.3/31.1

named node02 to node06. Each of the virtual nodes in node01 belongs to an experiment that includes one other virtual node running on one of the other nodes. The topology and VLAN tags for experiments are shown in figure 12.

In this experiment, we measured the UDP and TCP traffic rate that each virtual node in an experiment could send and receive in different cases. The first case is to find out the maximum achievable rate when no limit is placed on the traffic rate and only one experiment is active. This rate is 122MB per second (MBps) for both UDP and TCP traffic which is equal to 976Mbit per second (Mbps). Table 2 show the achievable rate in different cases when all experiments are active and send as fast as they can. Since all experiments running on node01 try to send and receive on one 1Gbps Ethernet link concurrently, they get a different share of this available traffic in different cases.

In table 2, we show the maximum traffic rate in MBps between a virtual node on node01 and its corresponding virtual node on node02 to node06. The UDP and TCP columns show the maximum rate when all virtual nodes in all experiments send and receive UDP or TCP traffic, concurrently, without any rate limit mechanism in place. As it can be seen, because of the massive packet loss in this case, TCP cannot achieve a stable rate, and its rate changes from 15 to 35 MBps. These measurements prove the need for a rate limiting mechanism when different experiments want to run on a shared virtualized infrastructure.

The columns with (rl) show measurements when we limit the send and receive rate in experiments to (12.5), (18.75), (25), (31.25), and (31.25) MBps respectively, totaling to 118.75 MBps (950 Mbps). As can be seen, using the rate limit functionality we could achieve the bandwidth guarantee requirements (with maximum 1% deviation from the target rate) in a VANI node. Another case that we have studied is the case where all virtual nodes in one experiment start sending traffic to one virtual node concurrently. This would result in congestion on the shared link that is serving the destination virtual node. To solve this problem, we have developed a novel traffic control mechanism that we will present in a separate paper in future.

10 Conclusion and Future Work

Virtualized Application Networking Infrastructure (VANI) is a converged communications and computing network that facilitates the realization of an open applications marketplace using a service-oriented control and management plane capable of managing hardware-based and software-based resources.

The architecture of VANI is designed to allow rapid application creation and experiment setup using service-oriented approaches. VANI utilizes virtualized commodity physical resources such as processing, storage, and networking resources. It also includes reprogrammable hardware resources used for development and deployment of high scale and high throughput networking algorithms and protocols.

VANI is designed to enable experimentation with architectures and applications that provide responsiveness and quality of service by having processing, storage, and hardware acceleration resources in all its nodes. Example applications that are video streaming applications, new content delivery networks, as well as power-aware and green networking architectures. In addition, applications that require high performance computing and networking can benefit from VANI's reprogrammable hardware resource. This resource can be reprogrammed in a short time to run hardware-based networking algorithms and protocols, and can send and receive traffic rates up to 128Gbps. Currently, we are working on development of a novel green networked system on VANI. We are also in the process of designing novel functionalities into the VANI control and management plane to automate application creation and deployment.

References

1. Bellovin, S.M., Clark, D.D., Perrig, A., Song, D.: A Clean-Slate Design for the Next-Generation Secure Internet (2005), http://sparrow.ece.cmu.edu/group/pub/bellovin_clark_perrig_song_nextGenInternet.pdf
2. Stanford University Clean Slate Design For Internet: An Interdisciplinary Research Program, http://cleanslate.stanford.edu
3. 100x100 project, http://100x100network.org
4. GENI System Overview (September 2008), http://www.geni.net
5. Peterson, L.: PlanetLab: A Blueprint for Introducing Disruptive Technology into the Internet (January 2004), http://www.planet-lab.org
6. PlanetLab GENI Control Framework Overview (January 2009), http://www.geni.net
7. Hibler, M., Ricci, R., Stoller, L., Duerig, J., Guruprasad, S., Stack, T., Webb, K., Lepreau, J.: Large-scale Virtualization in the Emulab Network Testbed. In: Proceedings of the 2008 USENIX Annual Technical Conference, pp. 113–128 (June 2008)
8. GENI Control Framework Requirements (January 2009), http://www.geni.net
9. Fiuczynski, M.E., Pötzl, H.: Linux-VServer, Resource Efficient OS-Level Virtualization (June 2007), http://ols.108.redhat.com/2007/Reprints/potzl-Reprint.pdf
10. CANARIE Inc. CANARIE: Canadian Network for the Advancement of Research, Industry and Education, http://www.canarie.ca
11. Grasa, E., et al.: UCLPv2: A Network Virtualization Framework Built on Web Services. IEEE Communications Magazine 46(3), 126–134 (2008)
12. Szegedi, P., Figuerola, S., Campanella, M., Maglaris, V., Cervello-Pastor, C.: With evolution for revolution: managing FEDERICA for future Internet research. IEEE Communications Magazine 47(7), 34–39 (2009)
13. Gibb, G., Lockwood, J.W., Naous, J., Hartke, P., McKeown, N.: NetFPGA: An Open Platform for Teaching How to Build Gigabit-Rate Network Switches and Routers. Trans. on Education 51(3), 364–369 (2008)

14. Bannazadeh, H., Leon-Garcia, A.: On the Emergence of an Application-Oriented Network Architecture. In: Proc. of IEEE Int. Conf. on Service-Oriented Computing and Applications, SOCA 2007, Newport Beach, California, pp. 47–54 (June 2007)
15. Farha, R., Leon-Garcia, A.: Blueprint for an Autonomic Service Architecture. In: International Conference on Autonomic and Autonomous Systems, ICAS 2006, Silicon Valley, CA (July 2006)
16. Redmond, K., Bannazadeh, H., Leon-Garcia, A., Chow, P.: Development of a Virtualized Application Networking Infrastructure Node. In: Proceedings of the 3rd IEEE Workshop on Enabling the Future Service-Oriented Internet, Honolulu, Hawaii (December 2009)
17. Chang, C., Wawrzynek, J., Brodersen, R.W.: BEE2: a high-end reconfigurable computing system. IEEE Design and Test of Computers 22(2), 114–125 (2005)
18. Mathew, B., Sarang, P., Juric, M.: Business Process Execution Language for Web Services BPEL and BPEL4WS. Packt Publishing, Birmingham (2006)
19. Sun Microsystems Inc.: OpenESB: The Open Enterprise Service Bus, `http://open-esb.dev.java.net`
20. Sun Microsystems Inc.: Java Web Start Technologies, `http://java.sun.com/javase/technologies/desktop/javawebstart`
21. IEEE 802.1ad-2005, Virtual Bridged Local Area Networks Amendment 4: Provider Bridges (2006), `http://standards.ieee.org`
22. Murty, J.: Programming Amazon Web Services: S3, EC2, SQS, FPS, and SimpleDB. O'Reilly Media Inc., California (2008)
23. Inc VMWare. VMware: A Virtual Computing Environment (2001), `http://www.vmware.com`
24. Padala, P., Zhu, X., Wang, Z., Singhal, S., Shin, K.G.: Performance Evaluation of Virtualization Technologies for Server Consolidation (2007), `http://www.hpl.hp.com/techreports/2007/HPL-2007-59R1.html`