A Virtual Network (ViNe) Architecture for Grid Computing

Maurício Tsugawa, and José A. B. Fortes

University of Florida
Dept. of Electrical and Computer Engineering, ACIS Laboratory
Gainesville, FL 32611-6200 USA
{tsugawa, fortes}@acis.ufl.edu

Abstract

This paper describes a virtual networking approach for Grids called ViNe. It enables symmetric connectivity among Grid resources and allows existing applications to run unmodified. Novel features of the ViNe architecture include: easy virtual networking administration; support for physical private networks and support for multiple independent virtual networks in the same infrastructure. The requirements of an application-friendly virtual network environment are presented and it is shown how the proposed solution meets them. Qualitative arguments are provided to justify all design decisions. Also presented is an experimental evaluation of the round-trip latencies and bandwidths achieved by a reference implementation. Measurements are reported for WAN-scenarios involving three different institutions. Under favorable conditions, ViNe bandwidths are within 90 to 100% of the available physical network bandwidth.

1. Introduction

A fundamental goal of Grid computing is to share resources, distributed across wide area networks, among users [1]. In most cases, collaboration among resources is necessary and symmetric network connectivity becomes essential for a Grid-computing environment. However, the Internet is highly asymmetric: often communication between two processes A and B can only be established if it is initiated by process A. This paper describes a general solution to the problem of symmetrically connecting resources in different administrative domains. In addition to restoring symmetry, the approach allows, with low administration overhead, the inclusion of machines and networks in distinct computational grids.

Two technologies contribute to the network asymmetry: private networks and firewalls. In order to allow clients in private networks to access servers in public networks, proxies and Network Address Translation (NAT) gateways were developed. However, due to the non-routability of private addresses, it is not possible to reach private servers from public networks. Firewalls were developed in order to protect resources against malicious attacks or bad use of network resources. Unfortunately, due to difficulties in isolating malicious applications, firewalls are often configured to block legitimate network traffic.

When a resource provider configures a computer to be part of a Grid, requirements including public network connection and firewall openings need to be met. Those requirements get more complex when resources in a given private network need to be integrated into different Grids. If the requirements cannot be met, a Grid infrastructure is usually limited to a private network or a particular sub-network [2].

A Grid-computing infrastructure, such as In-VIGO [3][4], aims to aggregate resources without imposing strict requirements on the networks to which they are connected, and also independently of the remote access mechanisms being used. For example, SSH [5], Grid Security Infrastructure (GSI) [6], Condor [7], Portable Batch System (PBS) [8] or any other mechanism can be used for remote job submission. In order to offer network environments that are customized for each computational Grid and compatible with application software expectations even in the presence of firewalls and NAT gateways between resources, the Virtual Network (ViNe) architecture must satisfy the following requirements:

1. *Symmetric communication between entities*: Internet firewalls, NAT gateways and proxies are the main barriers.

2. *Undisrupted original network environment*: ViNe must extend the existing network infrastructure without interfering with running services.

3. *Easy network configuration of resources joining the Grid*: Resource configuration requiring administrative privileges must be minimal.

1-4244-0054-6/06/$20.00 ©2006 IEEE

4. *No impact on security policies implemented in providers' domain*: Keeping the security policies of organizations untouched is essential to minimize reluctance in sharing resources through the Grid.

5. *Availability of mechanisms to automate the definition and deployment of networks*: Multiple independent and isolated virtual network environments are necessary when virtual computational Grids are created, so networks cannot be statically defined.

6. *Ability to run applications without modification*: Many applications are already available and in many cases recompiling and/or reengineering is not possible.

7. *No change in the Operating System (OS) network stack*: Resource OS changes are always problematic as they can cause applications to stop working. Also, such changes are typically not acceptable to system administrators.

8. *Undisrupted Internet infrastructure*: Changes in the core components of the Internet such as routers and DNS servers are either impractical or limit the deployment of a solution.

9. *Platform independence*: Grids are heterogeneous by nature, aggregating resources with different architectures running several OS variants.

10. *Scalability*: It is not reasonable to utilize solutions that assume a small number of machines.

To the best of our knowledge (see Section 2 and Table 1) there is not a single solution satisfying all conditions, and combining solutions is not an easy task. The ViNe architecture described in this paper addresses all the above-listed requirements. The next section reviews existing techniques used to overcome Internet connectivity limitations and to implement virtual networks. In Section 3 the ViNe architecture and a supporting design are presented. Section 4 discusses how ViNe meets the above-stated requirements, and security considerations are presented in Section 5. Section 6 describes details of a ViNe implementation and reports bandwidth measurements of ViNe performance in several scenarios. Section 7 presents conclusions and future work.

2. Previous work

Table 1 summarizes existing and proposed solutions which fall into three broad classes described below.

Solutions based on address/port translation need to deal with two problems: how to dynamically create mappings and how to enable end nodes to know about the existing mappings. Some implementations require applications to be aware of resource discovery protocols (e.g., SOCKS, DPF and GCB). Application-transparent implementations require changes in the OS kernel network stack and/or in the Internet infrastructure (e.g., IPNL, RSIP and AVES).

Solutions that abstract networking complexity and expose a new API, work only for new applications (e.g., P2P networks and the Ibis Grid Programming Environment).

Solutions based on tunneling assume that at least one host in each physical network has a public IP address without firewalls imposing limitations (e.g., VPN, VNET, VIOLIN and X-Bone).

2.1. Tunneling

Application transparent solutions that do not require changes in the Internet infrastructure use tunneling techniques (which is also the case for the ViNe approach described in this paper).

The problems shared by existing tunneling-based solutions (Virtual Private Network [22], SSH tunneling, Generic Routing Encapsulation [29], etc), which make them not suitable for direct application on Grids are:

1. *No support for hosts using the same private IP address*: 16-bit private IP space (see table 2) is very popular in small networks since it is used in default configuration of cable/DSL routers and Microsoft Internet connection sharing (ICS). In large networks, the 24-bit private IP space is preferred over 20-bit space. There is a high probability that two private networks use the same private IP subnet. This means that hosts in different private subnets may have the same IP address.

2. *High administration overhead*: Every time a new subnet joins or leaves the system, each administrator of every participating subnet must be contacted to configure new tunnels. A new network configuration may take days, depending on response time of administrators.

3. *Lack of flexibility to configure independent virtual networks*: The straight-forward configuration to enable communication between hosts is to establish tunnels from one subnet to all other subnets. However, in some cases, there is need for an isolated virtual network for a subset of hosts. Configuration of multiple virtual networks requires a considerable amount of effort from administrators.

The next section details the architecture and design of ViNe and how the above-mentioned problems are solved.

3. ViNe

The primarily goal is to create virtual network environments with bi-directional communication capability between any pair of hosts. In order to support unmodified applications, the ViNe architecture is based on IP-overlay on top of the Internet. Components to be virtualized are network interfaces, routers and links between them.

Hosts participating in a ViNe are configured with one or more additional IP addresses, as detailed in subsection

3.2. The additional IP addresses – so-called virtual addresses – imply the definition of a virtual address space – an IP address space without conflicts with physical networks, as detailed in subsection 3.3. Internet routers do not handle ViNe traffic, so each physical network joining the Grid needs one host running ViNe routing software, which is called a Virtual Router (VR). Hosts are configured to direct ViNe traffic to VRs and do not need additional software. Subsection 3.5 describes how ViNe packets are routed. Inter-VR communication uses the Internet infrastructure, so if a host behind a firewall or NAT acts as a VR, its communication ability is

Table 1: Existing and proposed solutions to the Internet asymmetric-connectivity problem

	Purpose	Approach	Issues
Internet Protocol version 6 (IPv6) [9]	Solve IPv4 address shortage.	Define a larger address space	- cannot overcome firewall limitations. - slow adoption.
IP Next Layer (IPNL) [10]	Alternative to IPv6.	Extend IPv4 space by adding a layer above IP. Identify hosts based on FQDNs, globally unique IP and private IP.	- changes host network stack. - changes NAT gateways.
Address Virtualization Enabling Service (AVES) [11]	Connect hosts behind NAT gateways.	Public addresses of proxies are dynamically allocated to machines in private networks	- applications always need to contact AVES aware DNS servers. - changes NAT gateways.
Peer-to-peer (P2P) first generation [12][13]	File sharing without depending on servers.	Peers communicate directly with each other. Peers behind NAT needs to always initiate the communication.	- application specific (i.e. no socket API is exposed).
P2P second generation [14][15][16]	Self-organized overlay infrastructure	Distributed Hash Tables (DHT)	- application specific
Project JXTA [17]	Create a generic P2P infrastructure.	User-level proxies handle.firewalls/NAT	- applications need to be aware of protocols.
SOCKS [18]	Provide private-to-public and private-to-private communication.	Application layer generic proxies relay packets.	- requires application support in order to work.
Realm Specific IP (RSIP) [19]	Connect hosts behind NAT	Clients in one realm (private) lease addresses in other realm (public) from RSIP servers	- changes host network stack - changes NAT gateways
Tunneling: VPN [22], IP in IP [28], GRE [29]	Connect networks which cannot be routed in the Internet.	Encapsulate packets such that it is possible to transport transparently using the Internet.	- 1 server per network w/ public address. - high administrative overhead. - not scalable w/ nr. of networks (each network needs tunnels to all others)
Dynamic Port Forwarding (DPF) and Generic Connection Brokering (GCB) [20]	Recover Internet connectivity for distributed computing.	Dynamically set Source and Destination NAT (SNAT and DNAT) rules in firewalls or use GCB servers as proxies.	- applications need to be aware of protocols or may require recompilation/re-linking to communication libraries (dynamic linking is not always possible).
Cooperative On-Demand Opening (CODO) [21] and Semantic Firewall [2]	Deal with limitations imposed by firewalls	Communication between applications and firewalls to establish opening rules.	- applications need to be aware of protocols or may require recompilation/re-linking to communication libraries (dynamic linking is not always possible).
X-Bone [30][31]	Deploy and manage Internet overlays.	Two levels of IP encapsulation is used for each overlay.	- assumes symmetric connectivity between peers (resources). - requires control (privileged processes) in each resource (hosts and routers)
VNET [23]	Connect VMware [24] virtual machines spread over wide area to a LAN.	Layer 2 tunneling.	- assumes symmetric connectivity between hosts of virtual machines. - layer 2 broadcasts can flood the network when nr. of hosts increases. - possible VMware dependencies.
VIOLIN [25]	Create virtual isolated network environments on top of an overlay infrastructure.	Create, configure and deploy virtual entities (hosts, LANs and routers) as User Mode Linux virtual machines.	- assumes symmetric connectivity in the overlay infrastructure. - possible UML [26] dependencies.
Ibis Grid Programming Environment [27]	Environment that handles firewall, NAT, security and TCP bandwidth problems.	Runtime system abstracts lower layer communication. TCP splicing is used to cross firewalls.	- existing applications cannot run without modification. - TCP splicing may not work with all NAT

compromised. The problem is addressed using techniques described in subsection 3.4.

3.1. Overall Architecture

Figure 1 illustrates the overall architecture of ViNe. For each physical network joining the Grid, a virtual address space, to identify new virtual hosts, is allocated. A VR is configured to manage that virtual address space. Hosts are configured with new network interface(s), to be used for the ViNe traffic. VR and hosts "physical" setup for ViNe operation is a "one-time" process. Virtual networks deployments and destructions are handled dynamically by ViNe without administrative intervention. VRs are connected together through tunnels in the physical space.

Since VRs use the Internet routing infrastructure to tunnel packets, there are as many virtual links (dashed lines in Figure 1) as routes supported by the Internet. Communication problems to/from limited-VRs (i.e., VRs behind firewalls and/or NAT gateways) are addressed in the next subsections.

3.2. Hosts: Virtual Network Interface

Hosts may have their connectivity limited by firewalls and/or NAT gateways, and the use of real network identifiers is not possible when symmetric communication or unique addresses are required. New network identification (i.e., IP address) is necessary in each participating host – this is achieved by properly configuring a virtual network interface.

There are several ways to virtualize a network interface. One method is to intercept packets before the physical routing infrastructure is reached. Intercepted packets can then be modified and routed according to virtual networking needs. The best place to intercept packets is the OS kernel network stack; however kernel programming would make the software hard to port. Another idea is to intercept OS networking calls and modify the behavior of those calls for ViNe activity. This approach can potentially degrade the performance of the whole machine and also has portability issues. The use of communication libraries may require application reengineering.

A complete virtual network interface card (NIC) can also be implemented. Virtual NICs are software components that emulate hardware NICs, and in general they are implemented as OS kernel modules. The use of universal devices such as TUN or TAP drivers [33] is a possibility, but this requires the installation of TUN/TAP packages in all hosts.

The alternative of choice for ViNe is that of using IP aliasing, which is the capability of binding multiple IP addresses in one physical NIC. The configuration of a static route for the virtual address space would redirect ViNe packets to VRs. In this way, hosts do not require any additional software installation in order to be ready for virtual networking.

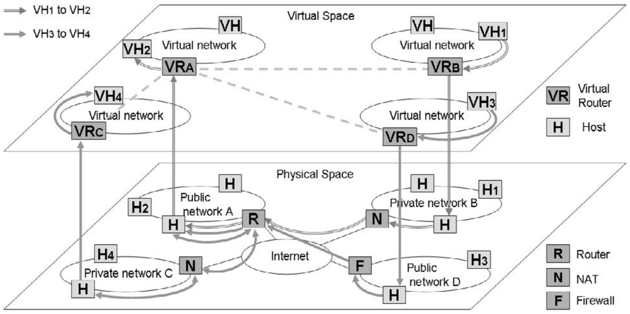


Figure 1: ViNe architecture – in each network in physical space, a machine (physical or virtual) is allocated to be a router in virtual space. ViNe traffic is directed to virtual routers which deliver packets using the Internet routing infrastructure.

3.3. Virtual Address Space

It is possible to virtualize the whole IP address space, making independent virtual 32-bit address spaces available for each application. However, resources should also be able to access the regular Internet services even if they are participating in one or more virtual networks. In order not to disturb the Internet, ViNe uses the space reserved for private addressing [32], which is shown in Table 2.

The use of the private addresses as virtual identities could be a problem when configuring a network that is already private. In order to avoid overlapping address spaces between networks, a global address space, partitioned into non-overlapping spaces, is needed.

Deciding which private space to use is difficult, as typically the resources joining a Grid belong to (physical) private networks that are already deployed. The less-frequently-used 20-bit space is a good candidate for virtual identifiers. However, the architecture must allow organizations utilizing the 20-bit private space to also join ViNe. This can be done by making VRs mirror the 20-bit space in a region of the 24-bit space. For example, the 172.16.0.0 – 172.31.255.255 range can be mirrored in the 10.116.0.0 – 10.131.255.255 range. VRs will deliver packets destined to 172.17.0.10 and 10.117.0.10 to the same machine. Organizations having conflicts with the 20-bit space would need a VR configured to work with a 20-

Table 2: Private IP address space

10.0.0.0 – 10.255.255.255	24-bit
172.16.0.0 – 172.31.255.255	20-bit
192.168.0.0 – 192.168.255.255	16-bit

bit range mirrored in some region of the 24-bit space, and use that space for virtual networking. It is unlikely for an organization to both use the 20-bit and 24-bit private spaces in their entirety.

The ViNe approach allocates a portion of the 20-bit space (using mirroring if necessary) for each organization joining the Grid. For example, 4096 organizations can be supported if the space is partitioned into blocks of up to 255 addresses. Blocks do not need to be of equal size, so organizations with more resources can receive larger partitions. Multiple networks can be defined, and participating entities are configured per host, i.e. each host in an organization is allowed to be connected to a different network. The same block can be allocated to different organizations if they participate in different networks or if it is possible to establish that they are not active simultaneously.

3.4. Firewalls and NAT

When it is not possible for an organization to configure one publicly accessible machine as a VR, the VR is not able to receive packets directly from peer VRs. However, sending packets (i.e., initiating the communication) is in general possible. Once a packet is sent, receiving the response packet is always possible. ViNe takes advantage of this fact to overcome connectivity problems.

When direct delivery of packets is not possible, an intermediate VR, connected to the public network, is assigned as a queue server where packets destined to limited-VRs are forwarded. Limited-VRs establish communication channels to the VR assigned as a queue server (hereon called a queue-VR). A communication channel can be a TCP connection initiated by a limited-VR and kept alive. When a packet is routed to the queue-VR, it can be forwarded immediately to the destination VR through the established TCP channel. Packets only need to be queued when the TCP channel is closed. Packets are kept in the queue until the TCP channel between the queue-VR and the limited-VR is re-established. With this approach, it is possible to have all VRs communicating with each other.

3.5 Virtual Routing Infrastructure

VRs are responsible for relaying ViNe packets. A packet destined to ViNe is directed to the VR (in the source node network), which examines the destination address and forwards it to the VR capable of delivering it.

VRs are required to have access to the Internet but they do not need to be in the public network. VR-to-VR communication uses the Internet routing infrastructure, and it is possible to assume that VRs are fully connected (see Section 3.4).

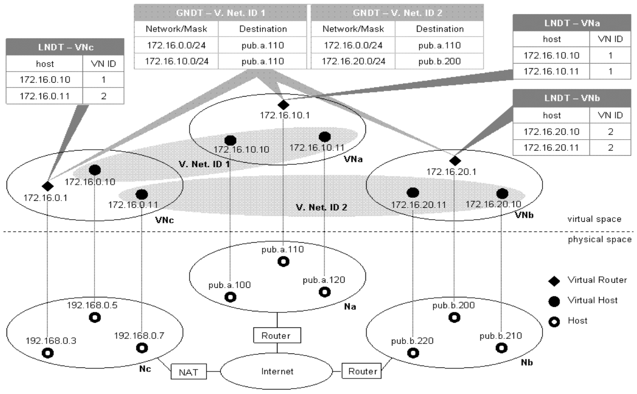
VRs maintain a set of routing tables: one Local Network Description Table (LNDT) and a Global Network Description Tables (GNDTs) for each active ViNe.

The LNDT stores information about hosts connected to the same physical network of a VR – an entry indicates which network a host is participating in. GNDT stores information about the structure of a virtual network – an entry indicates the physical IP address of the destination VR to where a packet needs to be forwarded if the destination address falls in the listed range. When a host-generated packet destined to a particular network reaches the VR, first the virtual network ID is verified in LNDT. Then using the corresponding GNDT the packet is tunneled to the destination VR where it will be delivered.

Figure 2 illustrates an example where two networks are defined. Virtual Network 1 (V.Net.ID1) connects participating hosts (represented by circles) in sub-networks VNa and VNc while Virtual Network 2 (V.Net.ID2) connects participating hosts in sub-networks VNb and VNc. All VRs (shown as diamond shapes) receive a copy of the GNDTs of every ViNe deployed (GNDT-V.Net.ID1 and GNDT-V.Net.ID2 in this example). When a packet is sent from 172.16.0.10 to 172.16.10.11, LNDT-VNc is checked to find out that the source host (172.16.0.10) is a member of Virtual Network 1. Then 172.16.0.1 (the Virtual Router handling this packet) consults GNDT-V.Net.ID1 and forwards the packet to pub.a.110. Finally, pub.a.110 delivers the packet.

Packets destined to 172.16.0.0/24 (which has a Virtual Router behind a NAT gateway in physical space) are forwarded to pub.a.110. 192.168.0.3, the VR of VNc, opens a TCP channel to pub.a.110 from where it will receive all messages destined to VNc.

If a host in VNa tries to communicate to a host in VNb,



LNDT – VNc	
host	VN ID
172.16.0.10	1
172.16.0.11	2

GNDT – V. Net. ID 1	
Network/Mask	Destination
172.16.0.0/24	pub.a.110
172.16.10.0/24	pub.a.110

GNDT – V. Net. ID 2	
Network/Mask	Destination
172.16.0.0/24	pub.a.110
172.16.20.0/24	pub.b.200

LNDT – VNa	
host	VN ID
172.16.10.10	1
172.16.10.11	1

LNDT – VNb	
host	VN ID
172.16.20.10	2
172.16.20.11	2

Figure 2: Virtual routing example - for each physical network (Na, Nb and Nc), a partition of the virtual network space (VNa, VNb and VNc) is allocated. All Virtual Routers receive copies of GNDTs. Each Virtual Router maintains the LNDT corresponding to its subspace.

packets will be dropped by VRs, as there is no matching entry in GNDT-V.Net.ID1 and GNDT-V.Net.ID2.

3.6. Multiple Isolated Virtual Networks

VRs act as routers and firewalls in virtual network space, and following the rules defined by routing tables it can forward, deliver or drop packets. This functionality is essential to support isolated virtual networks.

The information in the LNDT enables a VR to support the participation of hosts in different virtual networks. GNDTs define the structures of independent virtual networks.

When a VR receives a packet for routing, the source host address is verified – if it is not listed in LNDT the packet is immediately dropped. The corresponding LNDT entry will point to the GNDT that should be used for routing. The packet is then forwarded to the VR described in the GNDT. The destination VR verifies whether the destination host of the packet is in its LNDT, and whether the destination host is part of the ViNe.

Inter-VR communication is used not only for forwarding packets but also to exchange routing information. When a new virtual network is defined or when structural changes are necessary in the already existing virtual network, VRs exchange information to update LNDTs and GNDTs, without the need for local administrator intervention. For example, when a group of computers are needed to execute a parallel job, the Grid middleware can contact a VR to initiate the process of creating a new ViNe. When the job is done, the ViNe can be removed from the system. Details on inter-VR messaging are beyond the scope of this paper.

3.7. ViNe Address Allocation

The virtual address space allocation for each joining organization follows the Internet model. As such, ViNe requires an entity playing the role of the Internet Assigned Numbers Authority (IANA).

Inter-VR communication is secured based on the Public Key Infrastructure (PKI) model, and VRs are identified by their public keys. Only VRs with a valid certificate (signed by a trusted Certificate Authority) is allowed to join ViNe. Since VRs can be identified and authenticated, address assignment can be fully automated. The virtual address space allocation service can be implemented as a centralized service or utilizing distributed mechanisms.

4 Analysis

This section discusses the ViNe architecture in regard to all requirements listed in Section 1.

1. *Symmetric end-to-end communication*: The presented routing infrastructure, based on VRs, offers end-to-end connectivity between hosts by defining queue-VRs capable of queuing packets, when target VRs cannot be reached directly. The implication is that at least one VR needs to be placed on public network, and that hosts that depend on limited-VRs will potentially experience lower network performance than those connected to public VRs.

2. *No interference with Internet services*: In order not to disrupt the original Internet services in hosts joining ViNe, the 20-bit private address space (mirrored if necessary) is used to provide global virtual identities. Packets destined to the virtual range are redirected to VRs for routing in virtual space. Since redirection is done by defining static routes in hosts, there is no performance penalty for the regular Internet traffic. Also, there is no overhead in local communication between hosts.

3. *Easy configuration*: Configuring a network to be part of a ViNe is equivalent to configuring a VR. A VR is a machine – which could be physical or virtual, dedicated or non-dedicated - placed in participating networks with Grid resources. Configuration of routing software can be made very simple, as creation and management of ViNe are automatically done through communication between VRs. Virtual machine technology can further help on deployment of VRs, allowing transfer and instantiation of pre-configured VRs into target networks. Configuration of hosts is even simpler. Hosts need a one-time very simple system administrator intervention: configuration of a virtual interface (IP aliasing) and definition of a static route to the ViNe address space.

4. *Unchanged local security policies*: The presented architecture leaves the Internet traffic untouched. There is no need to change security policies already in place. However, since the architecture adds a new network interface (private IP address for ViNe) in each participating host, network firewalls and/or host firewalls need to allow traffic from/to the ViNe private address space. The policies defined for the original LAN can be applied to the ViNe address space. ViNe traffic is only visible in the ViNe space, since the Internet does not route private addresses.

5. *Automated creation and maintenance of virtual networks*: The presented architecture supports multiple independent and isolated virtual networks, and the definition, deployment and maintenance of ViNe can be fully automated. Resource allocation is under control of Grid middleware, so ViNe creation is triggered by the middleware, possibly (but not necessarily) in response to requests from hosts to join a virtual network.

6. *Existing applications run without modification*: From the applications' perspective, ViNe and physical networks are indistinguishable. It is even possible to aggregate machines in different private networks into a cluster and

run parallel applications, for example based on MPI, without recompilation or reengineering of software.

7. *Unchanged OS network stack*: The host configuration for ViNe does not involve any additional software installation.

8. *Unmodified Internet infrastructure*: The presented approach does not require any change to the Internet services. The unmodified Internet routing infrastructure is used to tunnel ViNe traffic between VRs. No other Internet service is required. When an Internet service (e.g., DNS) is needed in a ViNe, it needs to be configured in virtual space. Those services can be easily integrated since no software modification is required.

9. *Platform independence*: The ViNe approach does not require software installation on participating hosts. Running OSes are only required to support the definition of multiple IP addresses per NIC and configuration of static routes (which are supported by most modern OSes). This fact makes the ViNe approach virtually compatible to any platform.

10. *Scalability*: The maximum number of hosts simultaneously connected to a ViNe is limited due to the use of the 20-bit private IP space - not an architectural limit. The real limit is much larger considering the fact that two networks can share the same virtual address partition if it can be established that they are not active at the same time. It is even possible to have networks sharing addresses that could be active at the same time if they participate in different virtual networks. Looking forward to the possible adoption of IPv6, firewalls are likely to remain a problem with regard to asymmetric connectivity. Since the ViNe architecture is not tied to IPv4, the implementation described in Section 6 can be easily modified to support IPv6, in which case a much larger number of hosts can be supported.

5. Security

As ViNe creates connectivity between hosts without links in the physical space, many security issues can be raised, especially since connections might have been limited partially due to security concerns. The following are the main questions that need to be considered:

1. *Are security holes created in the physical network?* No. The approach requires one host in the joining network to work as a VR. This host does not need to have a public IP address and firewall rules need not be changed to allow incoming connections. All original security policies are in effect after enabling virtual networking. The difference is that communication with new hosts will be possible in virtual space, but that was the intent when the organization decided to have some of its resources join ViNe and the Grid.

2. *Can attackers in the Internet access hosts through virtual addresses?* No. Private IP addresses are used as virtual identifiers. Internet routers are configured to not route such addresses. The attack would only be possible if the organization's physical router gets compromised.

3. *Can hosts in the Internet be accessed?* Yes, but access to the Internet happens in physical space, following original security policies. VRs do not route packets to the Internet.

4. *How secure are VRs?* VRs are exposed to the same level of security as any host connected to the same physical network. VRs are assumed to be protected by physical firewalls. In virtual space, VRs drop packets that have its address as their destination, making it nearly impossible to be compromised in virtual space.

5. *How secure are VR daemon processes?* The first implementation runs a daemon listening to a TCP and/or UDP connections in port ACIS (2247). Regular inter-VR communication utilizes UDP messages, while limited-VRs use TCP channels. There are many ways to secure this communication: at lower layers, it is possible to configure VRs to only accept packets from IP addresses belonging to other VRs. At the application layer, PKI and digital signatures can be used to authenticate packets. In fact, all inter-VR messages, including encapsulated packets, are authenticated using keyed-hashing message authentication code (HMAC). Note that this is all done in physical space.

6. *Are tunnels encrypted?* No, however applications that need end-to-end confidentiality can use traditional mechanisms to secure their communication (e.g., SSL/TLS). Due to performance reasons, the current reference implementation provides communication channels between peers, not "secure channels". However, all VR-to-VR communication is authenticated using HMAC mechanism.

7. *Can network attacks be initiated in virtual space?* Only if an attacker has access to a host that is a member of a virtual network. As virtual space cannot be reached from physical space, an attack on virtual space can only happen if a node in physical space is compromised. There is the potential problem of not having a common security policy shared among organizations, but the presented architecture can restrict networks to only participate in certain virtual networks. So, organizations can specify to which physical networks they accept to communicate.

6. Implementation and Evaluation

A reference VR software has been implemented in Java, with lower level networking handled by C code. Packets are captured within the netfilter infrastructure [34], and copied from kernel to user space for processing. Packet injection is done by making use of libnet [35] library. A much efficient implementation would be in the form of an

OS-kernel module but, initial research and assessment of the ViNe approach and features are more conveniently done through a user-level implementation.

Hosts willing to join ViNe are not required to install any ViNe-related software. A one time administrative intervention is needed to configure the virtual network interface and the routes to virtual space. Binding a second IP address to a network card is accomplished by running "ifconfig" command in most of UNIX systems, while Windows allows defining up to 5 additional IP addresses in "advanced properties" of network interfaces. Static routes are configured by making use of the "route" command in most OSes.

The experimental setup has hosts in the University of Florida (UF), Purdue University (PU) and Northwestern University (NWU). Figure 3 shows the measured TCP round-trip latency and unidirectional throughput of the physical links involved. TCP round-trip latency and unidirectional throughput between end-hosts were measured using netperf [36] software.

There are three sources of overhead in the current prototype. One is the insertion of VRs into the network (increasing at least by two the number of hops between a pair of hosts), the second is the user-level software in the VR and the third is tunneling. Of these, as it will become clear in Section 6.2, the first two are the dominant overheads. VR performance can be best evaluated when VRs are exposed to a good physical networking environment (i.e., LAN) where VR throughput, not network bandwidth (1 Gbit/s), determines communication bandwidth. VR performance evaluation in a LAN is presented in the next section, followed by a discussion in Section 6.2 of the ViNe evaluation in a WAN environment.

6.1. Virtual Router Performance

Figure 4 shows the TCP throughput and round-trip latency between two hosts in different private networks routed by VRs connected in a LAN for different CPU configurations. The VR software is a multi-threaded program, so it performs better in multi-processor environments and when Hyper-Threading (HT) technology is used. TCP throughput increases by approximately 50% when HMAC is disabled. The current user-level implementation of a VR on a Hyper-Threaded dual Xeon with HMAC disabled can deliver a bandwidth of 80 Mbit/s while introducing only a round-trip latency of 1 ms.

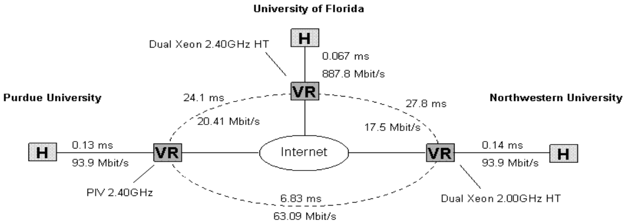


Figure 3: Experimental setup. Six different cases are considered depending on where firewalls are inserted in relation to the virtual routers (See Section 6.2).

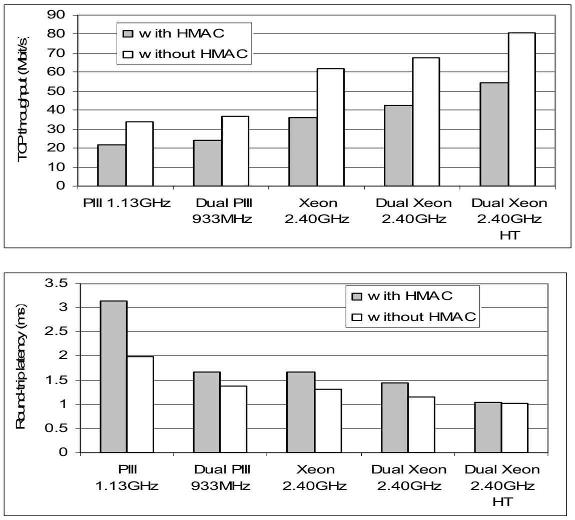


Figure 4: TCP throughput and round-trip latency for different CPU configurations, in a LAN environment (1 Gbit/s)

6.2. ViNe Performance

To evaluate the ViNe performance in WAN environments the following scenarios were considered:

Case 1: all VRs have access to the public network, without firewall limitations.

Case 2: as Case 1 but PU has a limited-VR which opens a TCP channel to UF queue-VR.

Case 3: as Case 1 but PU has a limited-VR which opens a TCP channel to NWU queue-VR.

Case 4: PU and NWU have limited-VRs which open TCP channels to UF queue-VR.

Case 5: PU and UF have limited-VRs which open TCP channels to the NWU queue-VR.

Table 3 summarizes the results. Case 0 represents the available physical performance. Cases 1 to 5 represent measurements with HMAC and Case 1a represents a measurement without HMAC.

Case 1 is the most favorable scenario when all VRs are connected to the public network. Insertion of the VRs in the host-to-host route has a small impact on latency – it increases by about 1~1.5 ms. Also it was possible to push TCP packets at nearly the available throughput for UF-PU and UF-NWU communication. This exemplifies the case when network speed, not VR throughput, limits

Table 3: Average round-trip latency (rtl – ms) and throughput (tp – Mbit/s) measurements for different cases of the WAN setup in Fig. 4 involving U. Florida (UF), Northwestern U. (NWU) and Purdue U. (PU)

	UF ⇔ PU		UF ⇔ NWU		NWU ⇔ PU	
	rtl	tp	rtl	tp	rtl	tp
0	24.1	20.4	27.8	17.5	6.82	63.1
1	25.5	19.8	29.3	15.7	8.26	27.7
1a	25.6	20.4	29.1	15.7	8.13	39.1
2	27.0	15.3	29.3	15.8	31.1	12.0
3	32.5	14.1	29.5	15.5	8.11	27.1
4	25.6	19.6	29.3	15.6	54.9	6.46
5	37.5	11.8	29.8	15.2	8.32	26.9

communication rates. The performance degradation in NWU-PU communication exemplifies the case when VR throughput, not network speed, determines bandwidth. By referring to Figure 4 one can realize that the measured bandwidth correspond to the processing limit of the machine acting as VR in PU. With more processing power (or lighter VR processing), it would be possible to achieve closer to the available throughput. This is also confirmed by Case 1a. It was possible to improve the throughput by approximately 50% when HMAC computation was disabled. For Case 1a, a UDP-datagram bandwidth of 90 Mbit/s, matching the available physical performance, was measured (UDP measurements are not shown in Table 3).

As mentioned in Section 3.4, VRs behind a firewall must contact a queue-VR to retrieve packets. Cases 2 and 3 illustrate how the allocation of the queue affects performance. Case 2 assigns UF as the queue-VR for PU, while in Case 3, NWU VR is the one assigned. Case 2 shows poor performance because, in addition to the additional hops and queue-VR overhead, communication is re-routed through the slowest of VR-to-VR routes (i.e. UF-to-NWU). The Case 2 throughput of 12 Mbit/s between PU and NWU VRs is not the result of VR overheads degrading the physically available bandwidth of 63.1 Mbit/s between PU and NWU, but the effects of the lowest of the bandwidths between PU and UF and UF and NWU (i.e. 17.5 Mbit/s). In contrast, since PU has better connectivity to NWU than to UF, using the NWU VR as queue-VR enables Case 3 to achieve performance close to that of Case 1 for all communication.

Cases 4 and 5 consider the scenario where only one VR is publicly accessible. In this setup, allocating the NWU VR as the queue-VR proved to be the best. Only UF-PU communication degraded but it still exhibited reasonable performance.

7. Conclusions

This paper described the ViNe approach to the implementation of virtual networks that eliminate asymmetric-communication problems in the Internet. Among other advantages, ViNe can be implemented with little administrative overheads and can be used to execute existing unmodified applications. A network administrator at an institution wishing to make resources available to the Grid needs only to do a one-time set-up of Virtual Router (which could be a cloned virtual machine or an existing physical machine) and a very simple one-time configuration of each resource's network interface. ViNe utilizing these resources can be automatically created by middleware with routing information being exchanged by VRs as needed. This is in contrast with other approaches reviewed in this paper which either require repeated extensive administrators' interventions, resource OS changes, application redesign/recompiling or changes in the standard Internet infrastructure.

Virtualization overheads of the implementation were quantified using a reference implementation of VRs. This implementation does not focus on performance, as the goal is to validate the design of ViNe. An implementation of VR software as a kernel module (similar to firewall software) or even as a firewall module is expected to offer performance similar to existing routers and firewalls as the ViNe processing requirement (packet header inspection, checksum calculation, tunneling, etc) is closely related. The measured performance confirms the potential suitability of ViNe for many Grid applications with moderate communication requirements. Currently, extensions to the prototype are in progress to improve its performance, to include all features of the design, and to fully evaluate the approach in In-VIGO environment with real applications and users. Virtual router improvements include using faster machines and/or kernel-level implementations. Such improvements can potentially make the ViNe approach also viable for data-intensive Grid applications that use high-performance networks.

8. Acknowledgements

This project is supported in part by the National Science Foundation under Grants No. EIA-9975275, EIA-0224442, ACI-0219925, EEC-0228390, EIA-0107686 and EIA-0131886; NSF Middleware Initiative (NMI) collaborative grants ANI-0301108/ANI-0222828, SCI-0438246; and by the Army Research Office Defense University Research Initiative in Nanotechnology. The authors also acknowledge two SUR grants from IBM and gifts from VMware Corporation and Cyberguard. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, Army Research Office, IBM, or VMware. The authors would also like to thank A. Sundararaj and Dr. P. Dinda of Northwestern University and R. Kennell and Dr.

S. Goasguen of Purdue University for providing the test environment.

9. References

[1] I. Foster, C. Kesselman and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International J. Supercomputer Applications*, 15(3), 2001.

[2] M. Surridge and C. Upstill. Grid Security: Lessons for Peer-to-Peer Systems. In *Proc. 3rd IEEE Conference on P2P Computing*, pages 2-6, Sep. 2003.

[3] S. Adabala, V. Chadha, P. Chawla, R. Figueiredo, J. Fortes, I. Krsul, A. Matsunaga, M. Tsugawa, J. Zhang, M. Zhao, L. Zhu, and X. Zhu. From Virtualized Resources to Virtual Computing Grids: The In-VIGO System. *Future Generation Computer Systems*, 21(6):896-909, Jun. 2005.

[4] J. Fortes, R. Figueiredo and M. Lundstrom. Virtual Computing Infrastructures for Nanoelectronics Simulation. *Proceedings of the IEEE*, 93(10):1839-1847, Oct. 2005.

[5] C. Lonvick, editor. SSH Protocol Architecture. Internet-Draft, Dec. 2004.

[6] R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, and V. Welch. Design and deployment of a national scale authentication infrastructure. *IEEE Computer*, 33(12):60-66, 2000.

[7] M. Litzkow, M. Livny, and M. Mutka. Condor – A Hunter of Idle Workstations. In *Proc. 8th International Conference on Distributed Computing Systems*, pages 104-111, 1988.

[8] R. Henderson. Job Scheduling Under the Portable Batch System. In *Proc. Workshop on Job Scheduling Strategies for Parallel Processing*, pages 279-294, 1995.

[9] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC2460, Dec. 1998.

[10] P. Francis and R. Gummadi. IPNL: A NAT-Extended Internet Architecture. In *Proc. of the ACM SIGCOMM*, 2001, Aug. 2001.

[11] T. S. Eugene Ng, I. Stroica and H. Zhang. A Waypoint Service Approach to Connect Heterogeneous Internet Address Spaces. In *Proc. USENIX 2001*, pages 319-332, June. 2001.

[12] T. Klingberg and R. Manfredi. Rfc-gnutella, http://gnutella.sourceforge.net, June. 2002.

[13] Homepage, http://www.kazaa.com, Sep. 2005.

[14] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of the ACM SIGCOMM 2001*, Aug. 2001.

[15] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Techinical Report CSD-01-1141, U. C. Berkeley, Apr. 2001.

[16] Homepage, http://brunet.ee.ucla.edu/brunet/, Jan. 2006.

[17] L. Gong. Project JXTA: A Technology Overview. Sun Microsystems, Oct. 2002.

[18] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas and L. Jones. SOCKS protocol version 5. RFC1928, Mar. 1996.

[19] M. Borella, J. Lo, D. Grabelsky and G. Montenegro. Realm Specific IP: Framework. RFC3102, Jul. 2000.

[20] S. Son and M. Livny. Recovering Internet Symmetry in Distributed Computing. In *Proc. of the 3rd International Symposium on Cluster Computing and the Grid*, May 2003.

[21] S. Son, B. Allcock and M. Livny. CODO: Firewall Traversal by Cooperative On-Demand Opening. In *Proc. of 14th IEEE International Symposium on High Performance Distributed Computing*, Jun. 2005.

[22] B. Gleeson, A. Lin, J. Heinanen, G. Armitage and A. Malis. A framework for IP-based virtual private networks. RFC2764, Feb. 2000.

[23] A. Sundararaj and P. Dinda. Towards Virtual Networks for Virtual Machine Grid Computing. In *Proc. of the 3rd USENIX Virtual Machine Research and Technology Symposium*, May 2004.

[24] Homepage, http://www.vmware.com, Jan. 2005.

[25] P. Ruth, X. Jiang, D. Xu and S. Goasguen. Towards Virtual Distributed Environments in a Shared Infrastructure. *IEEE Computer*, 38(5):63-69, 2005.

[26] Homepage, http://user-mode-linux.sourceforge.net/, Jan. 2005.

[27] A. Denis, O. Aumage, R. Hofman, K. Verstoep, T. Kielmann and H. Bal. Wide-Area Communication for Grids: An Integrated Solution to Connectivity, Performance and Security Problems. In *Proc. of 13th IEEE International Symposium on High Performance Distributed Computing*, Jun. 2004.

[28] W. Simpson, "IP in IP Tunneling", RFC1853, Oct. 1995.

[29] D. Farinacci, T. Li, S. Hanks, D. Meyer and P. Traina. Generic Routing Encapsulation (GRE). RFC2784, Mar. 2000.

[30] J. Touch and S. Hotz. The X-Bone. In *Proc. of Global Internet Mini-Conference at Globecom*, Nov. 1998.

[31] J. Touch. Dynamic Internet Overlay Deployment and Management Using the X-Bone. In *Proc. of International Conference on Network Protocols*, Nov. 2000.

[32] Y. Rekhter, B. Moskowitz, D. Karrengerg, G. de Groot and E. Lear. Address Allocation for Private Internets. RFC1918, Feb. 1996.

[33] Homepage, http://vtun.sourceforge.net, Jan. 2006.

[34] Homepage, http://www.netfilter.org, Jan. 2006.

[35] Homepage, http://www.packetfactory.net/libnet/, Jan. 2006.

[36] Homepage, http://www.netperf.org/netperf/, Jan. 2006.