Towards a Uniform Self-Configuring Virtual Private Network for Workstations and Clusters in Grid Computing

David Isaac Wolinsky
University of Florida
Gainesville, Florida
davidiw@ufl.edu

Yonggang Liu
University of Florida
Gainesville, Florida
yonggang@acis.ufl.edu

Renato Figueiredo
University of Florida
Gainesville, Florida
renato@acis.ufl.edu

ABSTRACT

The rising popularity of grid computing makes the issues of growth, security, and access critical in deploying and maintaining well-functioning grid systems. Overlay networks (ON) provide a framework to deal with these issues, but current techniques impose limitations and administrative burdens such as manual configuration for each new system in the grid, installation and configuration of software. Additionally, current approaches lack methods of effectively merging clusters with individual workstations, usually focusing on either the grouping of distributed clusters or a desktop/workstation Grid. The main difference between the two scenarios is that in a cluster environment all machines share a common ON router, whereas in a workstation environment each machine has ON software.

This paper presents a novel approach of self-configuring IP-based Virtual Private Networks (VPNs)[1] overlays that support dynamic, seamless addition of new resources to the grid for both cluster and workstation platforms. The approach allows for bridging physical and virtual networking in clusters, in a manner that allows dynamic configuration of IP addresses while avoiding overlay routing among nodes within the same layer 2 network. To enable these features, the ON runs on top of a Peer-To-Peer (P2P) network that provides supports a distributed data store. IP addresses are dynamically allocated by a virtual DHCP server controlled by the ON router through atomic operations on the distributed data store. This atomic operation creates a mapping of an IP address to a P2P address that can later be used by the VPN and router to determine the host of an IP address. We have prototyped this approach, demonstrating the ability to seamlessly mix both workstation and cluster based approaches into a wide-area Condor pool.

[1]Throughout this paper, the terms overlay network (ON), virtual network (VN), and virtual private network (VPN) will be used interchangeably. In this context a VPN/VN runs on top of an ON providing features such as IP and Ethernet routability to the ON.

Categories and Subject Descriptors

C.2.0 [**Data communications**]: Data communications

General Terms

Design, Standardization

Keywords

Virtual Networking, Autonomic Computing

1. INTRODUCTION

The use of overlay networks (ONs) as Virtual Private Networks (VPNs) for grid computing has been explored in previous research [8, 19, 11, 18, 12]. Through the use of ONs, owners of different clusters and workstations can merge their individual resources into a distributed grid. ONs not only provide access that would only otherwise be provided by limited public interfaces, they make the process of securing the system significantly easier. By decoupling the two networks, an administrator of a physical network need not be the same as the administrator of an ON running inside said network. This allows different security policies for the differing networks. For example, a public interface may be accessible by anyone but limit certain types of traffic, whereas an ON can be made accessible to only those with proper credentials and allow all forms of traffic.

To that end, ONs are feature rich but current approaches lack the ability to dynamically configure a grid consisting of clusters and workstations. Previous work in on this topic [9], analyzed the prospect of using dynamic host configuration protocol (DHCP) and route look up through the use of a distributed hash table (DHT) for the use of distributed virtual workstations. The contributions made in this paper are a natural continuation of that work extending the techniques to support the configuration of clusters and supporting a single, scalable ON for both clusters and workstations. Specifically, we discuss techniques for supporting a single layer 3 subnet across multiple domains using an Ethernet layer gateway without the need for an IP layer gateway. The advantage is that machines on the same layer 2 network (e.g. within a cluster) can communicate directly without routing through the overlay.

An additional focus of the paper is the handling of multiple layer 3 networks on the same physical switch with specific focus on DHCP. Layer 3 protocols like DHCP do not mix well on the same layer 2 network. When multiple DHCP servers are providing different IP subnets on the same layer 2 network, users may use join the wrong network because, by default, DHCP clients typically listen to the first DHCP server that responds to a broadcast. In other words, DHCP lacks the ability to support heterogeneous DHCP configurations [4]. A key contribution of this paper is an approach to use the DHCP protocol to automatically assign addresses over a wide-area ON, allowing the ON's DHCP infrastructure to coexist with an existing DHCP infrastructure in a single environment.

In this paper we will provide discussion and a reference implementation that evaluates the following:

- Providing scalable approaches for dynamic host configuration through the use of a P2P ON
- Simplified deployment and management of ON/VPN software
- Security and sandboxing issues
- Supporting heterogeneous DHCP servers in a layer 2 network

This paper's organization follows. Section 2 discusses the use of virtual networks in grid computing with examples and provides the necessary background to understand the framework for our contributions by reviewing previous work. Section 4 details our contributions by comparing and contrasting our cluster and workstation approaches. Section 5 analyzes different approaches that can be done to allow deployment of this software in multipurpose networks. Section 6 quantitatively evaluates the differences amongst a native network configuration, cluster model, and workstation model. Section 8 presents router scalability concerns. Section 7 concludes the paper.

2. VIRTUAL NETWORKING IN THE GRID

Three important areas that concern grids are connectivity, security, and growth. Traditional Grid systems tend to aggregate clusters, where each cluster has a single head node that provides access and enforces security. However, setting this up along with connectivity between the resources and the head node as well, as handling growth, are left to the system administrator. To allow clusters to merge into a unified system, administrators would have to place all resources on mutually public networks or create complex pathways between the two domains, creating potential security issues, limiting growth, and reducing the independence of that cluster. A simple solution to this problem involves the use of middleware brokering agents, such as Globus [7], that provide secure connections between clusters by requiring that each participating cluster head node has a signed certificate and be publicly accessible to the brokering agent. Other features of the cluster's designs lie in the hands of the administrator. This approach allows many small clusters to be aggregated to form a distributed grid with one additional requirement that each site's cluster management software [1, 14, 3, 17] must have support for communicating with the brokering agent.

Not all grid scenarios require this layering approach to support multiple administrative domains provided by hosting one's own cluster. Another scenario exists where all to all connectivity between resources in the grid is a requirement. Virtual networking provides a framework to address these issues, where all-to-all virtual IP connectivity is provided while retaining isolation from the physical network to facilitate the enforcement of security in the system. The virtual networking referred to is similar to Virtual Private Networking (VPN) services (e.g. OpenVPN [22] and CiscoVPN [2]) where all participants are in the same network and thus have all to all connectivity. Bandwidth constraints, complex configuration/management, latency, and reliance on a single site are issues presented by traditional VPN configurations and led to the desire for better communication pathways provided by distributed overlay networks to support such infrastructures.

Initial works in the field of virtual networking on top of overlay networks for grid computing include solutions like Violin [11], VNET [18], ViNe [19], and IPoP [8]. The primary feature found in all virtual networking software is the support for all to all communication amongst peers in the virtual network, though their mechanisms for supporting this are different. Table 1 summarizes key differences among these approaches, which have been motivated by different assumptions about the target environment and use. In general, all these approaches share a common feature, namely native support for IP traffic, which imposes no changes to legacy applications. While each configuration may have had software requirements that imposed significant limitations, Table 1 presents only the concepts and ignores software specific dependencies. The contributions of this paper largely stem from and extends upon initial work done in the IPoP overlay described in in [8, 10], which is described in more depth in the following section.

3. IPOP - THE P2P APPROACH TO VIRTUAL NETWORKING

IPoP differentiates from related virtual network techniques in how it builds upon a Peer-to-Peer (P2P) overlay for virtual IP packet routing and for distributed hash table (DHT)-based object storage/lookup. In doing so, it removes the requirement of having any centralized hosting mechanism. Furthermore, a P2P system can provide features that allow two private-addressed machines constrained by distinct NATs (Network Address Translation devices) to communicate with each other through traversal techniques as well as through the P2P overlay without configuring any routing rules, similar to the proxying techniques used in other VNs. This section discusses the features of P2P in IPoP that reduce the complexity in configuring a VN.

In a P2P system, each node has two ways to be addressed: via the lower layer network, e.g. an IP address, and through the overlay via a P2P address. The initial IPoP architecture relied on a one-to-one mapping between virtual IP address and P2P address (the SHA-1 hash function), thus requiring only knowledge of the peer's virtual IP address to communicate with them. This approach limited each P2P system to a single IP address space as two machines cannot share the same P2P address. Furthermore, there was an issue of contention, as two users would not be able to tell if they selected the same IP address unless there was a third party providing addresses.

In a revised architecture [10], IPoP introduced two concepts that allowed greater flexibility: 1) namespaces to allow for multiple virtual IP networks to share the same P2P system and 2) the use of a method for dynamic allocation of IP addresses. Unlike the previous version, the method of determining a P2P address was changed so that instead of mapping IP addresses with a hash function, the namespace and the IP were hashed to create a key for the lookup of a P2P address.

	Violin	VNET	ViNe	IPoP
Connectivity	Virtual hosts connect individual machines to the virtual networks. Optionally virtual hosts can connect to virtual switches at a layer 2, usually deployed per site. For layer 3, switches and hosts connect to routers, routers act as proxies. Sites are typically allocated an IP address space.	Each site runs a proxy providing Ethernet bridging to other proxies. Virtual machine hosts run retrieve packets and forwards them to the local proxy. Proxies routes are statically configured.	Each site has one router which is allocated a single address space. Each site must be connected to at least one other site, but allows for proxying over each other. No software required on individual hosts.	Each machine runs VPN software with a dynamic IP address in a common virtual address space. No proxying software required beyond the VPN stack running on each host.
Routing Paths	It is not described how sites are allocated address space or determine pathway to other addresses; routing tables are possibly statically defined.	Broadcast for discovery. Bridging learns paths after initial discovery. Virtual network packets are routed between VNET proxies. Various topologies possible, but need to be explicitly configured.	Packets in specific address are sent to the owner for that address space. Virtual network packets are routed between ViNe routers. Various topologies possible, but need to be explicitly configured, including NAT traversal.	Mapping stored in DHT resolves virtual IP address to P2P address. Virtual network packets are routed by VPN endpoints over shortcut optimizing, self-configuring structured P2P topology with STUN-based NAT traversal.
Growth	Requires software at each host and configuration specifying local switch and routers. Potentially limited by address space given per site and address space collisions.	Each machine must be configured with software that connects it to the proxy. Ethernet address conflicts need to be resolved.	Each machine must be configured by an administrator for regular and ViNe network access. Potentially limited by address space given per site and address space collisions.	Requires the IPOP software stack at each host.
Security	None mentioned, potentially through the use of SSH Tunnels.	Mentions use of SSL and SSH Tunnels.	Supports encrypted tunnels between ViNe routers.	Supports encrypted P2P links and end-to-end VPN tunnels (unpublished work).
Migration	Migration possible; requires reconfiguration of switches.	Layer 3 migration, product of layer 2 virtualization.	Not discussed; migration to a different domain may require forwarding from home domain.	Migration possible; routes self-configure without user intervention, product of the P2P overlay.

Table 1: Virtual Network Comparison

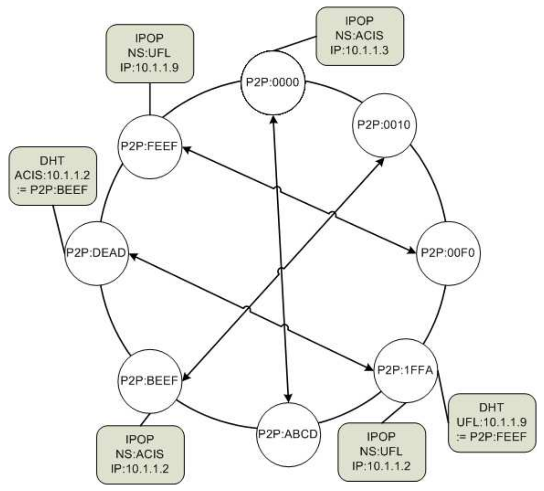

Figure 1: IPOP - IP Virtual Network with IP over P2P. Some P2P nodes have IPOP, others have DHT, while all support connecting to the P2P system.

Supporting dynamic address allocation requires access to a distributed data share that supports atomic and idempotent writes. To this end, IPoP uses a DHT provided by the underlying P2P system. A DHT is similar to a hash table: a database where data is stored in a (key, value) format. During address allocation, a machine attempts to perform an atomic write where the DHT key is the namespace and requested IP address and the value is its P2P address. The DHT is also used to store information about the namespaces that is used during DHCP configuration, such as the valid address range, lease time, and reserved IP addresses. Figure 1 presents an example IPoP deployment that illustrates this approach.

Furthermore, because IPoP uses the P2P address for message routing, which is decoupled from the physical network address of a host, virtual IP address migration happens transparently to applications (even across domains), and connectivity can be established as quickly as it takes for P2P links to be reformed. For example, when a user disconnects from a P2P system, they disconnect from other endpoints and no packets can be routed to them. When they connect at a later time or in another location those links are recreated and they are routable at the same P2P address.

As mentioned in Table 1, IPoP handles only layer 3 virtualization and has limited interaction with layer 2. As such, prior to the techniques described in this paper, IPoP could only support a one-to-one mapping of IPoP instance to host.

4. SELF-CONFIGURING, LOW OVERHEAD VIRTUAL NETWORKING

As the previous sections presented, the work of making virtual networking self-configuring with low overhead has not yet been completed. In this section, we focus on the following provisions:

- Allow resources in the same physical network direct layer 2 physical network communication
- Automatic configuration of a cluster requiring only a single virtual router for the entire cluster
- Allowing a single layer 3 subnet to be shared amongst all peers in a wide-area virtual network

Our solution space for the three provisions relies on manipulating the following features of a layer 2 / 3 (Ethernet / IP) network:

- Dynamic IP address and hostname allocation for a cluster via Dynamic Host Configuration Protocol (DHCP) and Domain Name Services (DNS)
- Resource address discovery via DHT
- Ethernet based address discovery via Address Resolution Protocol (ARP)
- Virtualizing Ethernet routing

The following two sections discuss the protocols and how they can be applied to to provide self-configuring, low overhead virtual networking. Afterwards we discuss how these techniques can be deployed in a single workstation or for an entire cluster. Finally, we compare this implementation to that of other popular virtual networking technologies.

4.1 Self-configuration and Packet Routing Provided by DHT/DHCP

As defined in the RFCs [4] and [15], DHCP provides for automatic allocation of IP addresses and additional network configuration. DHCP is a widely used protocol for network configuration, and various implementations of clients and servers exist. The steps in DHCP follow are shown in Figure 2 and are detailed in the following:

- Client sends Discover packet requesting address.
- Server receives the packet, allocates an address, and sends an Offer of the address and other network configuration.
- Client receives and acknowledges the Offer by sending a Request message to accept the offer.
- Server receives Request message and returns an Ack message containing the same details as the Offer.

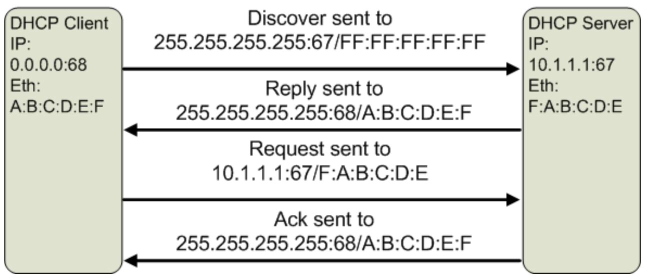

Figure 2: DHCP Client/Server Interaction.

For virtual networks where each site has its own layer 3 subnet, each site may host its own DHCP server that provides addresses in a network space that is unique to that site. To provide unified DHCP support for an entire wide-area virtual network system would require either Ethernet virtualization (so that a single DHCP server could provide address allocation for the entire network) or a distributed DHCP algorithm.

In [10], we presented a distributed DHCP algorithm. Instead of communicating with a DHCP server, a DHCP client talks with a DHCP/DHT proxy that attempts to write IP:P2P address values to the DHT. If it is successful, the proxy replies with an offer or acknowledgment as necessary. Furthermore, the DHT supports multiple DHCP configurations determined by the abstraction of a namespace. Each DHCP configuration data is stored inside the DHT is associated with a string corresponding to namespace, and as such a client / proxy need only know which namespace to run through a DHCP process and obtain an IP address to communicate inside that network.

When the virtual networking software receives a packet, its knowledge should be limited to source and destination IP of the packet and the DHCP namespace. The virtual network then queries the DHT for the namespace:destination IP. The return result will be either a P2P address or null. If it is null, the packet is dropped, otherwise the packet is forwarded to the P2P address.

Many legacy grid applications require fully qualified domain names, and providing this in a self-configuring system presents a challenge. To this extent, the virtual networking software should host a virtual DNS. The DNS server 'runs' on a reserved IP address, such as the lowest IP in the address space. When the virtual networking software receives a packet destined for that address to the DNS port (53), it passes the packet to the virtual DNS server that handles the translation between name and address. If the actual names are unimportant, then they can be based upon the IP address such that a virtual IP address like 10.5.123.44 would translate to a virtual name such as C005123044 for an address space of 10.0.0.0/8. If names need to be unique, they can be stored in the DHT, though schemes of how the exact name may be application-dependent. The machines learn about the DNS server from the DHCP configuration where part of the configuration variables include the address of a nameserver.

4.2 Virtualizing Ethernet Communication

DHCP only provides the mechanism for layer 3 connectivity, but how does a packet get to the router? In systems like ViNe, this was accomplished by using unique IP address spaces at each site. This approach allows all resources to communicate directly with each other and packets outside that subnet are sent to the ViNe router for forwarding. Previous sections have stated the disadvantages of not having a single address space; though the approach of using a layer 3 router is appealing due to simplified configuration and routing, it limits a virtual network to placing each machine in its own subnet and forwarding all packets to the router. There is another alternative, which is considered in this paper: implementing a virtual bridge that routes only layer 3 IP packets.

In order to implement a virtual bridge, the virtual networking router must handle Address Resolution Protocols (ARP). An ARP packet is used on a layer 2 network to discover the layer 2 address of a machine given its layer 3 address, as shown in Figure 3. The approach is essen-

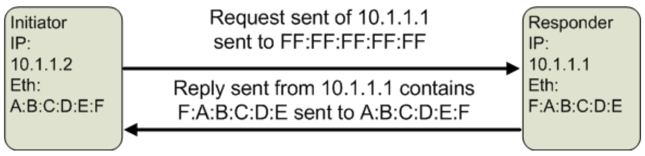

Figure 3: ARP Request/Reply Interaction.

tially as follows. The VN router responds to ARPs that are destined to virtual network hosts with an unallocated Ethernet address (such as FE:FD:00:00:00:00). The Ethernet switch fabric device will recognize that the machine hosting the virtual network must be bridging the machine to another switch, and will then forward all packets sent to FE:FD:00:00:00:00 to the machine hosting the virtual network router.

In order to take advantage of direct communication to machines on the same physical network, the virtual network router must know how to differentiate ARP requests for remote hosts (which it replies to) from ARP requests for local hosts (which it ignores). This can be handled by the virtual network's DHCP and router combined — not only does the DHCP handle configuring the client, it self-configures the router as well. The cases when not to reply to ARP are:

- After obtaining DHCP configuration from the DHT, the virtual router is aware of the IP address space, and only serves within that space.
- During DHCP, usually after a machine receives an Offer and prior to it sending a Request, it will send an ARP to the network making sure the offered address is unallocated. Therefore it is imperative that the virtual router take note of this as soon as an address has been allocated by a virtual DHCP server.
- The requested host is within our physical layer 2 network.

Handling migration becomes more complex than in previous IPOP scenarios. ARP requests, however, are sent out frequently enough to handle a detection of a moved node relatively quickly; to take advantage of this, the router should occassionally send out ARP requests to the local clients.

This approach is not limited to layer 2. In order to support full virtual bridging, the virtual networking router could store Ethernet addresses in the DHT as well. In the DHT, the Ethernet addresses would have to be allocated first come, first serve, though potentially the administrator could be notified if there is a collision. So now when an ARP packet comes in, a lookup is performed on the DHT and the router now needs to keep a mapping between all remote sites Ethernet, IP, and P2P addresses. The downside to this approach is that machines are still required to use DHCP in order to setup communication channels between the client and the router, otherwise undesirable broadcasting of packets will be required to discover the location of an Ethernet address in the system.

4.3 Deploying in Clusters and on Workstations

Previous virtual networking technologies used packet capturing techniques at the router / switch machine. An alternative approach first used by IPoP and later by a revised ViNe approach is the TAP device [13], or virtual Ethernet device. TAP devices are available for many popular operating systems including Windows, Linux, and MAC OS/X.

A TAP interface is similar to other I/O devices, in that it allows operations such as reads and writes from user-level processes. Packets that are available for reading from the TAP device arrive by being inserted into the networking stack whether from a local socket or via a bridge. Packets that are written to the TAP device enter the network stack and are handled by a user's application via a socket or redirected over a bridge.

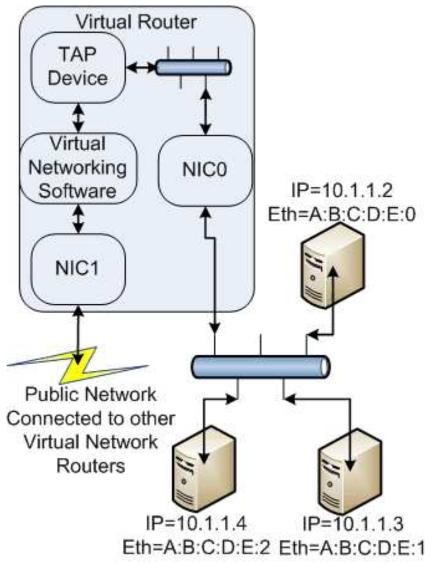

Figure 4: TAP deployed for a cluster based virtual router.

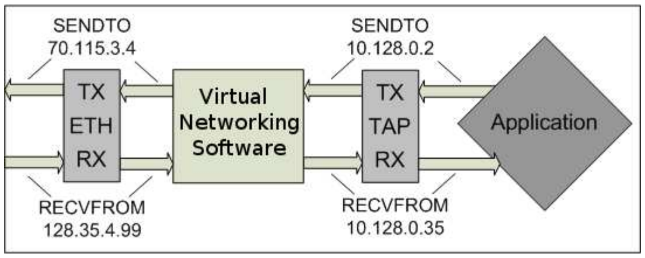

Figure 5: TAP deployed on a workstation based virtual router.

The configuration for a cluster is shown in Figure 4, where a tap device is bridged with the physical network. Figure 5 presents the configuration for a workstation environment. In the latter case the networking stack uses sockets and the TAP device receives an IP address. The techniques presented in this paper can be used to configure either environment.

4.4 Comparing and Contrasting to Other Virtual Networks

Taking into consideration that our approach can be applied to layer 2 or layer 3 networks, the significant difference between our approach and VNET or Violin is scalability of routers / switches in the overlay. Where VNET and Violin must broadcast packets for Ethernet address location, our approach places the mapping in a well-defined location. The broadcasting issue also makes the deployment of a DHCP server difficult in such solutions as the Discover messages would have to be broadcast over the entire network. Because this approach relies on an abstracted address (the P2P address) and like VNET has support for layer 2, they both inherently support migration of IP addresses.

In terms of layer 3 connectivity, Violin and ViNe provide similar features — that is each router provides connectivity for an individual subnet — though after that their approaches are markedly different. Violin requires that each machine have a virtual host configuration that connects to a virtual switch which connects to a virtual router, though a virtual host can also act as a virtual router bypassing the switch allowing support for clusters and workstations. Violin does not appear to allow direct communication between peers on a physical network due to the virtual host requirement, though ViNe does allow direct communication on a physical network. Since ViNe and Violin use different subnets for each site in the network, each site can provide its own DHCP services without worrying about global conflicts. The problem using subnets though is that the more subnets that are used, the greater chance of address conflicts with other private address spaces that are not part of the virtual network, wasted addresses, and more static information being stored at the router node. Migration is possible in both approaches but the picture is not so clear. In ViNe, the entire network would have to follow the migrating address, furthermore, neither approaches discuss events that determine exiting the network and successive re-entering.

5. INTEGRATION WITH EXISTING NETWORKS

Ideally, the virtual networking software would be configured in such a way that all traffic would occur on an isolated layer 2 network. By doing this and making the router an Ethernet router as opposed to an IP layer fully isolates the machines to the virtual network (VN), effectively sandboxing the environment. There are two methods to achieve this isolation: either through the use of Virtual LANs (VLANs) or unique hardware switches for the VN.

Virtual LAN technology has been widely adopted in enterprise networks to support multiple independent layer 2 networks on the same switch. Configuration of a VLAN is usually done on a per network port basis. To support VLAN across switches, VLAN-specific information is added to Ethernet packets. A typical Ethernet packet has a source and destination address, protocol field, and a checksum at the end of the packet. The VLAN addition specifies a protocol type, an additional two bytes to specify the VLAN ID, two more bytes specifying the layer 3 data protocol. This is the preferred way to deploy a VN inside a LAN, though it has limitations: it requires an administrator to configure the switch; VLAN functionality is only available on managed switches which tend to be considerably more expensive than unmanaged one; and inability to allow the host to configure its own networking rules.

If the physical network does not support VLANs, there are some features that need to be addressed such as isolation and preventing interruption of normal network behavior. In particular, DHCP does not naturally support multiple servers supporting heterogeneous network configurations on the same layer 2 network. This becomes relevant to a VN when deploying in a cluster where an existing DHCP server is configured, and where a separate VLAN for the VN cannot be configured.

In such cases, virtual machines which are increasingly used in grid computing [6] to provide secure, sandboxed environments [16] can be leveraged. There are two different approaches to configuring a VN for VMs: (1) have each VM contain the VN software [21] or (2) bridge the VM to the physical network and connect that to the VN router. Having the VN software in the VM requires that the VM have a connection to the public network, removing some of the isolation benefits from the VM sandbox. Here we focus on the latter approach of bridging the networks. While VM traffic is confined to the VN, the physical machine may need access to the public network.

In this section, we describe techniques that improve isolation for VNs. Without using a VM, these solutions can be circumvented by root exploits and, as such, they are only useful for service separation layer 3 broadcast services like DHCP. Solutions that we have investigated so far include:

- VLAN provided by the host computer
- Packet forwarding through the use of EBTables
- Customization of DHCP processes

5.1 VLAN

Many operating systems, network card drivers, and VM managers allow the configuration of VLAN on a regular network card. A VN can leverage this by having all members in the VN be configured to be on the same VLAN. This approach has a significant limitation, namely, it does not work with managed switches which drop the packets as they do not know of the VLAN. This method does not provide isolation unless the VN is being accessed by a VM, in which case, the host may configure the VM's network card to run on a VLAN.

5.2 Packet Filtering

One thought was to have IPTables, a network stack management utility found in Linux and other Unix-like operating systems. As it turns out, raw sockets skip past IPTables thus any attempts to use IPTables directly fails. There is another approach that still allows a machine to manipulate raw sockets. It involves creating a software "bridge" and using EBTables, a networking filtering package for software bridges. The steps to do this are: create a bridge, bridge the VN interface to it, and optionally rename the bridge to the VN interface. At this point, the bridge will be used as the network interface and the original network interface will be a bridge to the physical layer 2 network. In the case of a VM, the EBTables would be configured on the virtual device used to communicate with the VM (e.g. in VMware these are the "vmnet" devices).

An attractive feature of EBTables is that it can review IP-level packets. This allows for a simple approach of providing DHCP separation. By detecting DHCP packets, they can be forwarded to a different port than the RFC-specified 67 and 68. Of course, the virtual DHCP server would have to be configured to use these ports as well.

5.3 Changes to DHCP Infrastructure to Support Multiple Layer 3 Networks

Some DHCP client distributions, such as dhclient, support specifying the DHCP port numbers. This approach removes the use of an additional two ports from use in the

client / server	m1		m2		m3		m4		average	
	h2h	v2v	h2h	v2v	h2h	v2v	h2h	v2v	h2h	v2v
m1	N/A	N/A	700	109	940	192	940	183	860	161.33
m2	576	111	N/A	N/A	622	83	566	105	588	99.67
m3	928	162	701	87	N/A	N/A	939	140	856	129.67
m4	942	202	701	98.3	942	136	N/A	N/A	861.67	145.43
average	815.33	158.33	700.67	98.1	834.67	137	815	142.67	791.42	134.03

Table 2: Iperf bandwidth (Mb/s). First column is Host to Host (h2h) bandwidth and the second is virtual to virtual (v2v) IPOP bandwidth. Machines act as clients in rows and servers in columns.

client / server	m1		m2		m3		m4		average	
	h2h	v2v	h2h	v2v	h2h	v2v	h2h	v2v	h2h	v2v
m1	N/A	N/A	0.48	0.94	0.5	0.7	0.26	0.51	0.42	0.72
m2	0.32	0.82	N/A	N/A	0.38	0.73	0.14	0.57	0.28	0.71
m3	0.49	0.78	0.51	0.96	N/A	N/A	0.12	0.48	0.38	0.71
m4	0.27	0.53	0.3	0.7	0.22	0.55	N/A	N/A	0.26	0.58
average	0.36	0.71	0.43	0.87	0.37	0.65	0.18	0.52	0.33	0.69

Table 3: Ping latency (ms). First column is Host to Host (h2h) latency and the second is virtual to virtual (v2v) IPOP latency. Machines act as clients in rows and servers in columns.

virtual network and suffers from scalability problems if too many distinct DHCP systems are running in the same layer 2 network.

A solution that may scale better would be by allowing a DHCP client to have a configurable field where it can specify a DHCP namespace or configuration that it is seeking. Then only DHCP systems that support that namespace would respond. This can be achieved by manipulation of the magic cookie field in DHCP packets. The downside of this approach is that it requires a "patched" DHCP client that connects to the virtual network.

6. EVALUATION

To evaluate the contributions of this paper, we have added router features to IPoP. For evaluation, we focus on overheads that are introduced when single machines communicate over an external VN router as opposed to a local VN connection. The machines were selected to represent a variety of hardware configurations dated from 2003 to 2007, though the emphasis in this section is not on absolute numbers but the relative performance for the test environments. Each machine is equipped with a gigabit network card connected to the same gigabit switch.

First we establish baseline results for both host-to-host (h2h) and for virtual network (v2v) performance when the VPN router runs on both endpoints. All pair-wise combinations of machines $m1..4$ are considered. Results are provided in Table 2 for bandwidth and Table 3 for latency. Latency was measured by running ping for 30 seconds. Bandwidth was measured by running iperf for 30 seconds. As was expected, the h2h performance is significantly better than v2v, reflecting the main motivation in this paper to avoid v2v overheads in communications within a cluster.

We now look at the performance of the virtual network when two endpoints communicate through VPN routers residing on separate machines. In this experiment, machines $m3$ and $m4$ are the endpoints, and $m1$ and $m2$ host the VPN routers. The router machines were selected as the two best-performing machines from the base line results. Results are presented in Tables 4 and 5. Comparing latency between the approaches where the router runs on the end-

client / server	m3 / m1			m2 / m4		
	meas	exp	phys	meas	exp	phys
m3 / m1	N/A	N/A	N/A	219	183	701
m2 / m4	219	202	576	N/A	N/A	N/A

Table 4: Bandwidth in router configuration (Mb/s). Values shown are: meas (measured via iperf), exp (expected value calculated as the minimum between routers and host and routers), phys (minimum of host to host in path). exp and phys are derived from table 2. x / y means that y acts as a router for x.

client / server	m3 / m1			m2 / m4		
	meas	exp	phys	meas	exp	phys
m3 / m1	N/A	N/A	N/A	0.94	1.17	0.91
m2 / m4	1.13	1.51	1.26	N/A	N/A	N/A

Table 5: Latency in router configuration (ms). Values shown are: meas (measured via ping), exp (expected value calculated as ping latency h2h for client to router, v2v for router to router, and then h2h for router to client), , and phys (latency between hosts in the path). exp and phys are derived from table 3. x / y means that y acts as a router for x.

points versus remotely, the latter is comparable to the physical latency between the router and the client / server. In terms of bandwidth, the approach with external routers improve performance compared to h2h. By allowing the router to focus on routing as opposed to data processing in a user level application and sending the data over a VN removes some of the overheads that reflect on bandwidth. However, a user-level router may not scale up to match the speed of the network, and multiple routers may help in improving cross site bandwidth.

7. IMPROVING PERFORMANCE WITH ADDITIONAL ROUTERS

As shown in the evaluation, the virtual network may not be able to take full advantage of the available physical bandwidth. One feature we were unable to evaluate was how well bandwidth and latency scale in a cluster environment where potentially hundreds of machines would be using a single virtual router. Two potential approaches are increasing the thread count in the VN software or increasing the number of VNs at each site. The thread approach was discussed in [20] and shown not to have any improvements. As such, this section focuses on techniques to support multiple routers in the same network.

Before deploying more than one router, the network should be checked to determine the benefits of additional routers. Certain criteria include maximum bandwidth and latency issues, so there should be a tool that determines if a single virtual router cannot handle these requirements. The tool should also be capable of determining the proper number of routers to achieve maximum capacity.

The second issue is how to deploy the virtual routers. To determine this, an evaluation on whether or not multiple virtual routers on the same machine would be beneficial. Different hardware configurations may produce different results, specifically, multicore and multi-network card machines may be be able to take advantage of multiple virtual routers.

The final piece to the puzzle is having the virtual routers work together. In the architecture specified in this paper, the system assumes a single virtual router for the entire layer 2 network. To provide a self-configuring multirouter infrastructure these issues arise:

- how do routers discover each other - perhaps through broadcast / multicast?
- how to provision DHCP, DNS, and other services?
- how to communicate state changes between routers?
- how to load-balance?

A simplistic approach would be to have routers rank themselves in order of their uptime followed by Ethernet address. Each router in the system would synchronize with the others to determine the handling of DHCP and thus assignment of machine to router. For example, in a 5-router system, the 5th router takes responsibility for every 5th machine that enters the network. This approach suffers from inability to handle dynamic load balancing though. Beyond this, we have not examined the topic further and is left as future research.

8. CONCLUSIONS

This paper presents an approach to support self-configuring virtual networks that can aggregate both single workstations with co-located VN interface and clusters with a VN router for the entire cluster. At the heart of this lies a DHT/DHCP proxy that not only allocates addresses but also automatically configures the virtual router to be aware of which machines are in the network. The approach allows for either physical or virtual machines in different physical networks to communicate with each other over virtual links. The techniques can be applied to virtual networking technologies like VNET and Violin to improve scalability. Furthermore, we discussed issues such as security and network separation emphasizing that the use of virtual machines or VLANs.

For a reference implementation, we implemented the proposed features in IPoP, which is a P2P overlay supporting a DHT. Outside the contexts of this paper, we have successfully tested the functionality of the DHCP/DHT proxy by connecting nodes behind the router to a wide-area cluster/desktop Condor pool (the Archer computer architecture grid [5]). In our evaluation, we also showed the benefit of avoiding virtual networking overhead when two nodes are on the same physical network, and the differences in latency and bandwidth between having a single virtual router per site and having a router co-located with a workstation resource.

9. ACKNOWLEDGMENTS

Effort sponsored by the NSF under grants OCI-0721867 and CNS-0751112. 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 NSF.

10. REFERENCES

[1] Altair. Pbs pro, March 2007.
[2] Cisco. Cisco vpn, March 2007.
[3] P. Computing. Load sharing facility, lsf, March 2007.
[4] R. Droms. *RFC 2131 Dynamic Host Configuration Protocol*, March 1997.
[5] R. Figueiredo and et al. Archer: A community distributed computing infrastructure for computer architecture research and education. In *CollaborateCom*, November 2008.
[6] R. J. Figueiredo, P. A. Dinda, and J. A. B. Fortes. A case for grid computing on virtual machines. In *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, Washington, DC, USA, 2003. IEEE Computer Society.
[7] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *Int. J. High Perform. Comput. Appl.*, 15(3):200–222, August 2001.
[8] A. Ganguly and et al. Ip over p2p: Enabling self-configuring virtual ip networks for grid computing. In *IPDPS*, Apr.
[9] A. Ganguly, D. Wolinsky, P. O. Boykin, and R. Figueiredo. Decentralized dynamic host configuration in wide-area overlay networks of virtual workstations. In *PCGrid 2007*.
[10] A. Ganguly, D. I. Wolinsky, P. O. Boykin, and R. J. Figueiredo. Decentralized dynamic host configuration in wide-area overlay networks of virtual workstations. In *WORKSHOP ON LARGE-SCALE, VOLATILE DESKTOP GRIDS*, March 2007.
[11] X. Jiang and D. Xu. Violin: Virtual internetworking on overlay infrastructure. In *IPDPS '03*.
[12] D. Joseph and et al. Ocala: an architecture for supporting legacy applications over overlays. In *NSDI'06*.
[13] M. Krasnyansky. Universal tun/tap device driver, March 2007.
[14] M. Livny, J. Basney, R. Raman, and T. Tannenbaum. Mechanisms for high throughput computing. *SPEEDUP Journal*, 11(1), June 1997.
[15] R. D. S. Alexander. *RFC 2132 DHCP Options and BOOTP Vendor Extensions*.
[16] S. Santhanam, P. Elango, A. A. Dusseau, and M. Livny. Deploying virtual machines as sandboxes for the grid. In *WORLDS*, 2005.
[17] Sun. gridengine, March 2007.
[18] A. I. Sundararaj and P. A. Dinda. Towards virtual networks for virtual machine grid computing. In *VM'04*.
[19] M. Tsugawa and J. Fortes. A virtual network (vine) architecture for grid computing. *IPDPS 2006*.
[20] M. Tsugawa and J. A. Fortes. Characterizing user-level network virtualization: Performance, overheads, and limits. In *e-Science '08*.
[21] D. I. Wolinsky and et al. On the design of virtual machine sandboxes for distributed computing in wide area overlays of virtual workstations. In *VTDC*, 2006.
[22] J. Yonan. Openvpn, March 2007.