Private Virtual Cluster: Infrastructure and Protocol for Instant Grids

Ala Rezmerita, Tangui Morlier, Vincent Neri, and Franck Cappello

INRIA/LRI, University Paris-Sud, Orsay, France
{rezmerit, tmorlier, neri, fci}@lri.fr

Abstract. Given current complexity of Grid technologies, the lack of security of P2P systems and the rigidity of VPN technologies make sharing resources belonging to different institutions still technically difficult. We propose a new approach called "Instant Grid" (IG), which combines various Grid, P2P and VPN approaches, allowing simple deployment of applications over different administration domains. Three main requirements should be fulfilled to make Instant Grids realistic: 1) simple networking configuration (Firewall and NAT), 2) no degradation of resource security and 3) no need to re-implement existing distributed applications. In this paper, we present Private Virtual Cluster, a low-level middleware that meets these three requirements. To demonstrate its properties, we have connected with PVC a set of firewall-protected PCs and conducted experiments to evaluate the networking performance and the capability to execute unmodified MPI applications.

1 Introduction

Sharing resources in a secure way, over the Internet, is attractive for a broad range of users and communities. Audio and video over IP, file sharing, file storage and distributed computing are examples of applications concerning many communities of users. However, despite the continuous progress in Grid, P2P and VPN technologies, sharing resources over different administration domains still raises technical difficulties. Grid technologies allow sharing resources between the participants of virtual organizations [1]. Compared to previously existing technologies, Grid middleware provides tools for inter-domain security and resource management, assuming pre-existing local software and policies in every Grid site. The current trend towards the use of Services [3, 2] responds to the complexity of managing heterogeneous resources and sharing policies by providing a standard interface between the user and the resources.

However, installing Grid middleware is still complex and requires the skills of networking, security and OS experts. Moreover, providing a standard but novel interface to the users imposes, in many cases, to re-implement or to adapt the applications. P2P systems allow simple resource sharing between large communities of users. However, they exhibit two major limitations: 1) the security is very limited and generally not considered in these systems and 2) they run dedicated applications. Albeit Jxta [4] provides a communication layer to deploy and

W.E. Nagel et al. (Eds.): Euro-Par 2006, LNCS 4128, pp. 393–404, 2006.

run P2P applications, it has a major limitation by exposing only a Java interface to the application. Installing and using a VPN (Virtual Private Network), using technologies like VTun [5] or IPsec [6], allows users registered in the VPN to share their resources as if they were in a LAN. However, VPN's have their own limits: 1) installation and maintenance require OS and networking experts, administrator authorization and 2) they are static.

In fact, existing technologies restrict resource sharing to Grid and VPN experts or users of unsecured and dedicated P2P systems. This situation motivates the research presented in this paper towards a more spontaneous and dynamic Grid approach called "Instant Grid" (IG), in reference to popular "Instant Messaging" environments. Three mains requirements should be addressed in an IG environment: 1) Connectivity. Firewall and NAT settings may preclude the deployment of cross-domain applications. Moreover, the user may have no technical knowledge on how to setup correctly firewalls and NATs. Thus, an IG environment should use a set of firewall and NAT configuration and/or traversing techniques, transparent to the user and acceptable by domain administrators. 2) Security. Sharing resources across administrative boundaries should not lower the security level of the hosting sites and the shared resources, 3) Compatibility. Sharing resources should not imply specific application or runtime developments.

In this paper, we propose PVC (Private Virtual Cluster), an environment for Instant Grids. PVC design considers the following context: 1) resource sharing is established when required and 2) security is based on classical OS mechanisms (currently access rights and sandbox or virtual machines in the near future), used commonly in LAN's and clusters. PVC turns dynamically a set of resources belonging to different administration domains into a cluster where existing cluster runtime environments and applications can be run.

The next section presents the related work concerning the three issues. Section 3 describes the general architecture of PVC and gives details on the protocol implementation. The evaluation of PVC is presented in Section 4.

2 Related Works

In this section, we present the existing projects and technologies related to the three main issues of Instant Grids: connectivity, security and compatibility.

2.1 Connectivity, Security

One of the most popular projects providing connectivity among peers in different administration domains is JXTA [4]. Based on proxy technologies, JXTA proposes two communication approaches: a rendezvous and a pipe binding protocol. The two methods use a relay to forward messages between peers which results in significant communication overhead. To provide secure communication between peers, JXTA uses a virtual transport layer based on TLS (Transport Layer Security). The difficulty of installing and configuring the proxy limits the usage of JXTA in the Instant Grid context.

Ibis [7] is another project providing NAT and Firewalls traversing techniques to connect resources in different administration domains. Several approaches are successively proposed to bypass Firewall/NAT: a direct connection, a simultaneous TCP SYN connection and a proxy connection. For user identification and secure communication a standard SSL/TLS infrastructure, performing data encryption and peer authentication over a socket connection.

CODO [8] provides end-to-end connectivity for distributed applications over firewalls/NAT protected domains in a secure way. It consists in firewall agents (FAs), placed on the firewall machines and client libraries (CLs) linked with the application. The FA communicates with CL to dynamically add and delete rules needed to establish direct connections. Authentication and security are based on X.509 certificates. The major limitation of CODO is that it currently supports only firewalls based on Netfilter and it assumes the installation on firewalls of the FAs that is not always possible.

Another simple and practical NAT traversal technique is UDP/TCP hole punching [9]. This technique enables two clients behind NAT, to set up a direct peer-to-peer UDP/TCP session with the help of a rendezvous server. Following the statistics given in the article describing this technique, about 82% of the NATs support hole punching for UDP, and about 64% for TCP streams.

With the popularity of DSL network, the use of NAT increases dramatically. Unfortunately, NAT imposes another limitation for the direct connection of peers. Two projects propose NAT discovery and bypassing techniques. The first one is STUN RFC [11]. This standard describes the techniques to discover the NAT type and an UDP protocol to traverse it. The standard classifies NATs in four classes and the traversing technique works for three of them.

The second one is UPnP [10]. The UPnP Forum proposes an API for communications with the NAT device allowing opening firewall ports for direct connection. This technique is particularly suitable for the objectives and constraints of Instant Grids. Obviously, this method does not work with the NAT devices that are not UPnP compatible or if the administrator does not enable it.

2.2 Compatibility, Virtualization

To the best of our knowledge, only JXTA provides compatibility and a virtualization layer in addition to connectivity and security. Its approach is based on unique IDs, by which the network resources can be addressed independently of their physical address.

Several projects allow the creation of a virtual cluster from independently administered domains through machine and network virtualization. A VioCluster [12] logically moves machines between virtual domains, allowing a cluster to dynamically grow and shrink based on resource demand. Network virtualization in VioCluster is made by a hybrid version of VIOLIN [13] which gives to a machine the ability to connect to the private network.

Cluster-On-Demand(COD) [14] shares the same objective. COD was inspired by Oceano [15]. Its main difference is its dynamic resource management between multiple clusters by reinstalling the base OS on resources. The VNET [16] is a

virtual private network tool implementing a virtual local area network over a wide area, for virtual machines in Grids. VNET is a simple proxy scheme that works entirely at user level and uses the Layer Two Tunneling Protocol (L2TP).

3 General Principles of Private Virtual Cluster

The objective of Private Virtual Cluster (PVC) is to provide, in a transparent way, an execution environment for existing cluster applications over nodes distributed on the Internet. The main difference between PVC and VPN is its capability to dynamically connect firewall protected nodes, without any intervention of domain administrators and without breaking the security rules of the domains hosting the nodes. Compared to other projects presented in the related work section, PVC provides a fully integrated environment.

PVC itself is a distributed system working as a) a daemon process (*peer*) running on each participating host and b) a brokering service. The role of each local peer daemon is to establish a secure direct connection between the local peer and the other participating peers, subsequently leaving the connection control to the application. The role of the brokering service is to help establishing these connections by 1) collecting and advertising the peer connection requests and 2) tunneling some communications between peers when direct connections are not established, 3) translating network addresses from virtual to real and 4) transporting security negotiation messages. Typically, the brokering service may also help with failure detection, although this feature is not yet implemented.

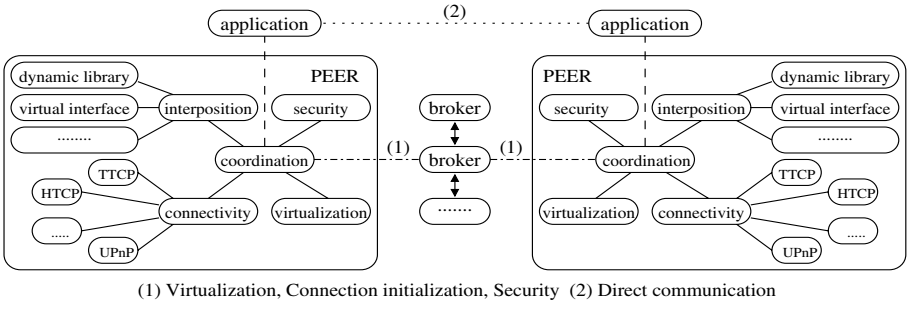

Fig. 1. PVC architecture

Figure 1 presents the modular architecture of PVC. The peer daemon encapsulates five modules for: 1) operation coordination, 2) communication interposition, 3) network virtualization for the application, 4) security checking and 5) peer-to-peer direct connection establishment. The modular architecture offers the possibility to extend and adapt each module to fit with the target environment. The brokering service is implemented as a set of replicated nodes, connected to the Internet and accept inbound communications from PVC peers.

All daemon modules are coordinated locally by the coordinator, which also participates in the global coordination of a PVC deployment. The coordinator

runs a workflow through the four other modules to establish the direct connection between the local peer and distant ones. The coordinator also exchanges messages with the brokering service to implement the global coordination.

The interposition module intercepts the application connection requests and transfers them to the coordination module. It may be implemented in various ways (network calls overloading, virtual network interface) offering high adaptability to the system configuration. The intercepted requests are routed on a virtual network simulated by the PVC virtualization module, which features its own IP range and domain name service.

In this virtual network, the PVC security module checks the respect of pre-existing security policies and authenticates the virtual cluster participants. Different security standard and specific methods may be adapted to the PVC architecture (SSL certificates, standard security challenges, etc.). The connectivity module transparently helps the cluster application to establish direct connections between virtual cluster nodes (peers). Like all the other modules, a variety of techniques can be used depending on participants host configuration as well as its local network environment (firewall, NAT). Standard (UPnP) and original mechanisms (Traversing-TCP, TCP Hole Punching) may be used to establish direct connection between peers. In the following parts, we will focus three key modules: virtualization, security and connectivity.

3.1 Domain Virtualization

One of the PVC objectives is to allow the execution of cluster applications without any modification. Cluster applications generally use the socket model as interface with the communication network. Following this constraint we have chosen to use a domain virtualization at IP level. The virtualization layer establishes an IP domain over resources belonging to different administration domains having public or private (possibly conflicting) addresses. Like in a VPN, an overlay network featuring virtual IP addresses is built on top of the actual network.

To avoid the conflict between real and virtual networks used by the resource, we use a specific IP class defined by a RFC [17] for experimental purposes (class E ranging between 240.0.0.1 and 255.255.255.254). The use of these IP addresses guaranties that no real machine uses them (such addresses are actually not routed on the Internet). A virtual DNS, configurable by the PVC members, is associated with this experimental IP class.

3.2 Security Policy in PVC

The main objective of the security mechanism is to fulfill the security policy of every local domain and enforce a cross-domain security policy. Two security levels are implemented: 1) local to the administration domain and 2) between domains. The intra and inter-domain security policies could be configured by local system administrator who could also define the global policy. When a connection is requested by the application, the local peer first checks that it can accept inbound and outbound communications with other peers outside the

administration domain, according to the local policy. Then, it checks that IP addresses of external peers and the ports to be used are granted by the global policy.

Without a strong access control mechanism, someone may take advantage of the brokering service and pretend to take part of the virtual cluster. To avoid this, every virtual cluster has a master peer (a peer managed by the virtual cluster administrator) implementing the global security policy. Only the master peer can dynamically register new hosts. Before opening the connection, every peer checks that the other peer belongs to the same virtual cluster. The cross authentication is performed using master information and crossing the brokering service. A key point in the design is that the security protocol does not need to trust the brokering service. This protocol ensures that: 1) only the participating hosts of a cluster can be connected to each other and 2) only trusted connections are returned to the cluster application.

In the current implementation, each host connected to PVC has its own private and public key. Every participant to a virtual cluster knows the public key of its master before connecting to PVC infrastructure. The master peer registers the participation of a new peer, asking the brokering service to store its public key previously encoded with the master's private key. During the establishment of the connection, both peers obtain the other side's public key from the brokering service and decode the received message with the master's public key. This mechanism ensures that only the master registers other participants on the brokering service.

The peer's mutual authentication consist in a classical security challenge-response: the client generates a cryptographically random string M, encrypts it with server public key and sends it to the server; the server decrypts the message with its private key, encrypts the obtained value using its private key and returns the result, Es(M), to the client; the client decrypts Es(M) using the server's public key, obtaining Ds(Es(M)); if that value is equal to the original M, the client is satisfied of the server's identity. Similarly, the server picks a random string L, encrypts it and sends it to the client, which returns Ec(L) to the server. The server checks that Dc(Ec(L)) equals L and it thereby satisfied with the client's identity. The security is implemented using OpenSSL Crypto library [23]. The experimental results are presented in Section 4, where we discuss the overhead of PVC in secure connection establishment.

3.3 Inter-domain Connectivity Techniques

As the major objective of PVC is to establish direct connections between distributed peers, the connectivity module can host several connection protocols. In the current implementation, we have integrated three techniques in PVC. In this section, we present the integration of these techniques.

Integration of a Firewall configuration protocol. In the last two years, UPnP project became very popular. The principal vendors of domestic network devices incorporated UPnP in their routers. Using UPnP, a PVC peer can communicate with the router and can open the ports for direct connections.

To guarantee the safety of the local area network, the port forwarding rules must be erased from the router when they are no longer needed. If during the connection initialization, the authentication fails, or if the peer detects the end of the connection, it erases immediately the target rule from the router. If the PVC peer fails before the end of the connection, the application that starts PVC, running on the same peer re-launches it. A security issue may occur if the host running PVC fails before the end of the connection, and another host takes its private address. In this case, all the rules related to the host should be removed from the router. We use a distributed architecture to detect node failure and handle firewall rule deletion. Every node of a domain runs a monitoring daemon. These daemons periodically check the rules present on the firewall and ping the corresponding host so that they may delete the rules related to a faulty machine.

TCP hole punching. Widely used for applications such as online gaming and voice over IP, TCP Hole Punching allows connection establishment between two hosts behind NATs in different administration domains.

Both clients establish a connection with the broker that observes the public addresses (given by NAT) and private addresses of the clients and shares this information between the peers. After this exchange, the clients try to connect to each other's NAT devices directly on the translated ports. If NAT devices use the previously created translation states then a direct connection is possible.

The advantage of using this method is that it does not require special privileges or specific network topology information. However, this technique does not work with all type of NATs as their behavior is not standardized.

A novel technique: Traversing-TCP. Traversing TCP (TTCP) is derived from the TCP protocol and it works with firewalls that are not running stateful packet inspection.

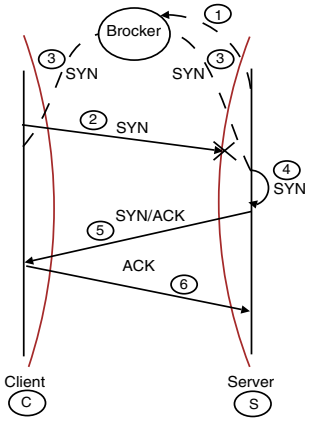

Fig. 2. Traversing-TCP

It essentially consists in 1) transporting, using the PVC Broker, the initiating TCP packet (SYN) blocked by the firewalls or NAT on the server side and 2) injecting the packet in the server IP stack.

Figure 2 presents in details the TTCP technique. Plain lines show the packets corresponding to the TCP standard. Dashed lines correspond to specific Traversing-TCP messages. A TTCP connection behaves as follows:

Definitions: Server node: S, Broker: B, Client node: C, initializing packet: SYN

1. The peer on S connects to B and waits for new connection demand;
2. C sends SYN to S. It opens the Firewall of C but it is stopped by the Firewall of S;
3. The peer on C sends the SYN packet information to B. B forwards it to the peer on S;

4. The peer on S injects the SYN packet to the IP stack on S;
5. To this SYN packet S replies with a SYN/ACK packet. The SYN/ACK packet opens Firewall on S and is accepted by the Firewall on C (previously opened);
6. The initialization of the TCP connection ends with an ACK packet from C to S: the TCP connection is established.

TTCP works under the following device configurations: 1) The firewall must authorize the outgoing packets and must accept all packets from established connections; 2) Following the [11] classification, TTCP should work with all NAT's, except symmetric NAT (which maps a port to a quadruplet: the internal host-port and external host-port).

Note that RST packets sent as rejection notification are also captured by a PVC client peer and not forwarded to the client IP stack. Following our experience with the DSL-Lab platform (cf. the evaluation section) and related work [18], these requirements fit many professional and domestic configurations.

After connection establishment, the communication can continue following classical TCP operations. The communication between the two peers is direct, bypassing the broker and ensuring high communication performance.

4 Performance Evaluation

In this paper, we focus on the performance evaluation of the whole workflow for establishing a virtual cluster. We measure the overhead of PVC and demonstrate its capabilities by running unmodified MPI applications deployed over a set of firewall protected PCs, connected to the Internet by ADSL connections.

4.1 Experimental Protocol

PVC was designed to have a minimal overhead for TCP communications. In our first experiments, we demonstrate this property with two types of tests: the first one compares network performance with/without PVC, using NetPerf [19]. The second one evaluates the overhead of PVC for establishing a connection.

The evaluation test for network performance was performed on a local PC cluster with three different Ethernet networks: 1Gbps, 100Mbps, 10Mbps. We have used standard PCs with BroadCom TG3 Ethernet interface connected using Netgear EN106 10Mbps Ethernet switch, a Netgear FS105 100Mbps Ethernet switch and a D-LINK D65-1216T Gigabit Ethernet switch.

To evaluate the system overhead of establishing a connection, we used the DSL-Lab [22] platform: a set of resources connected to the Internet by a DSL network. The same platform was used for the second type of experiments, which consisted of the execution of real cluster applications. We ran the NAS benchmarks [20], the MPIPOV program and the scientific application DOT [21] to evaluate the capability of PVC to establish all the connections required by a complex distributed environment like the MPICH runtime environment.

4.2 Evaluation Results

Bandwidth Overhead. Figure 3 presents the bandwidth (in Mbps) of PVC (with/without firewall) and the reference (without PVC), using NetPerf on 10Base-T, 100Base-T and 1000Base-T Ethernet networks. To simulate the firewall on both sides we used Linux Netfilter Iptables.

	1Gbps	100Mbps	10Mbps
Reference (without PVC)	715 (5)	94.0 (0.4)	6.9 (0.1)
PVC without firewall	720 (5)	94.0 (0.4)	6.9 (0.1)
PVC with firewall	717 (3)	94.8 (0.5)	6.9 (0.1)

Fig. 3. Bandwidth of PVC and reference, as measured by NetPerf, on three ethernet networks, in Mbps. Values in parenthesis are standard deviations.

Figure 3 demonstrates that the network rates computed by NetPerf are statistically similar. The difference between the two series of measurements is lower than the standard deviation for all the tests. We can conclude that PVC does not reduce the available bandwidth: once the connection is established, PVC does not interact with the application any more, leaving the network rate unaltered.

Connection overhead. For the establishment of direct connections between the peers, PVC uses the TCP-Traversing, TCP hole punching techniques or the firewall configuration protocol UPnP. To compare these three methods, we evaluated their overhead using a specific test suite. In our test suite, a client makes 1000 consecutive connections to a server and then tears them down.

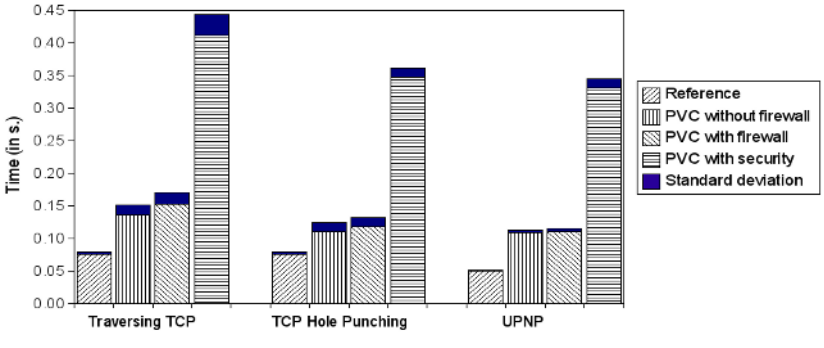

Fig. 4. Overhead of Private Virtual Cluster

Figure 4 shows the mean costs for the TCP connection establishment. In the presence of firewall the overhead of PVC using Traversing-TCP resp. (TCP Hole punching) technique is 76ms (42ms) and 60ms (40ms) without firewall. The overhead observed for PVC with UPnP is the same in both cases and is about 60ms. This overhead encompasses the costs of interception of application connection attempt and communication with the broker.

Security overhead. In the current version of PVC, security is implemented using OpenSSL Crypto library [23]. The first step of the verification (membership to the same virtual cluster) is coupled with the resolution of the virtual name, avoiding the substantial increase of overhead. The second step of the security protocol, which is done after the connection establishment, increases significantly the overhead. Figure 4 shows the mean costs of authentication in PVC during connection establishment.

Since PVC intervenes only at the beginning of the end-to-end communication, the observed overhead remains reasonable in the context of the distributed applications.

Running MPI applications. We successfully ran several of the NAS benchmarks class A (EP, FT, CG and BT) on the PVC architecture. However, due to the network performance between the nodes of the DSL platform, FT, CG and BT do not scale with the number of nodes. Only EP with its low communication to computation ratio is scalable.

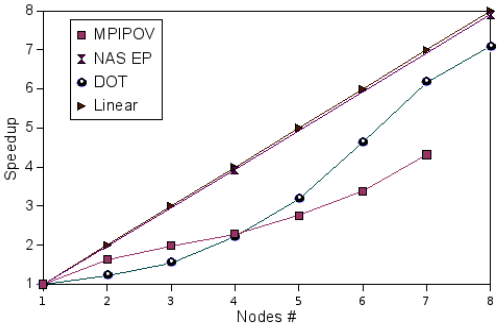

Fig. 5. Speedup of MPI applications with PVC over a set of DSL connected nodes

Figure 5 presents the performance of EP according to the number of nodes in the DSL platform. The speedup increases almost linearly with the number of nodes. The results of the NAS benchmarks demonstrate that PVC successfully transforms a set of nodes connected to the Internet through Firewall and NAT into a virtual cluster where MPI runtime environments and applications can be executed without modification.

The purpose of the DOT [21] program is to compute electrostatic potential energy between charged molecules. It operates in a master/slave mode. During the computation, the amount of data communication is low, but at the beginning and at the end, some large arrays must be communicated. Figure 5 presents the speedup for the computation of the example provided with the DOT distribution, using from one to eight workers. The master is running on the first node.

The MPIPOV test measures the execution time for the computation of a graphical rendering application parallelized with MPI. MPIPOV uses a master-worker algorithm. Compared to the NAS EP, MPIPOV requires more

communications of image rendering parameters and results. We perform the test by splitting the image in 32 sub-images. Figure 5 presents the speedup for the computation of the same image using from one to seven workers. The master is running on a separated node.

Over all applications, the speedup evolves in a non-linear way. Obviously, the scalability of the MPI application performance on DSL networks depends on the communication to computation ratio of the application and the individual performance of the heterogeneous platform components. We did not tune the application in order to improve the performance since the purpose of the experiments is only to demonstrate the capability of PVC to run unmodified MPI runtime environments and unmodified, non-trivial MPI applications. However, even without tuning, the test demonstrates that the ADSL platform can provide significant speedups for some non-trivial MPI applications.

5 Conclusion

In this paper, we have presented and evaluated the performance of a lightweight middleware called PVC (Private Virtual Cluster) designed to dynamically establish virtual clusters over resources connected by the Internet and protected by firewalls and NAT. PVC derives from a mix of Grid, P2P and VPN concepts. It features three main properties required for "Instant Grids": 1) a security model that does not reduce the security level of the domains and resources to connect, 2) the capability to run cluster applications and runtime environments without modification, and 3) negligible communication overhead.

PVC is itself a distributed system running as coordinated peer daemons executed on volunteer participants. Its architecture is modular and uses a set of adaptable modules for its main functions: coordination, security, connectivity, virtualization, interposition. Modules can be extended or modified to fit with the target environment (e.g. interposition by virtual network interface or by shared libraries, firewall-traversing protocols based on UPnP, TCP hole punching or other.). We have detailed two important mechanisms: the security model and an original traversing technique called "Traversing TCP". TTCP allows establishing direct connections between resources protected in different administration domains, except if the communicating resources are protected by a firewall running state-full packet inspection.

Our performance evaluation demonstrates a moderate overhead (60 ms) for the connection establishment, and a negligible bandwidth and latency reduction compared to standard TCP communication. By establishing a virtual cluster, at the IP level, with negligible communication overhead, PVC can be used to deploy and run unmodified cluster applications and runtime environments. We demonstrate this capability by running the MPI version of the NAS benchmarks and the POV-Ray program on a set of PCs connected to the Internet by protected DSL connections. Altogether, PVC features a set of characteristics allowing non-OS and network specialists to deploy and run existing cluster applications over multiple administration domains, with minimal performance overhead.

References

[1] Ian Foster and Carl Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[2] M. Humphrey et all. An early evaluation of WSRF and WS-notification via WSRF.net. In *Proceedings of the 5th IEEE/ACMInternational Workshop on Grid Computing*, pages 172-181, Washington, DC, USA, 2004. IEEE Computer Society.

[3] S. Tuecke, K. Czajkowski, and I. Foster. Open Grid Services Infrastructure (OGSI) version 1.0. *Global Grid Forum*, 2003.

[4] Li Gong. JXTA: A network programming environment.*IEEE IC*, 5(3):88-95, 2001.

[5] M. Krasnyansky. Virtual tunnels over tcp/ip networks.*http://vtun.sourceforge.net/*.

[6] R. Thayer, N. Doraswamy, and R. Glenn. Rfc 2411 - ip security document roadmap. USA, 1998. RFC Editor.

[7] A. Denis et all. Wide-area communication for grids: An integrated solution to connectivity, performance and security problems. In *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing*, pages 97-106, Washington, DC, USA, 2004. IEEE Computer Society.

[8] S. Son, B. Allcock, and M. Livny. CODO: Firewall traversal by cooperative on-demand opening. In *Proceedings of the 14th IEEE International Symposium on High Performance Distributed Computing*, Washington, DC, USA, 2005. IEEE CS.

[9] P. Srisuresh, B. Ford and D. Kegel. Peer-to-peer communication across NATs. *USENIX Annual Technical Conference*, 2005.

[10] http://www.upnp.org/standardizeddcps/.

[11] C. Huitema J. Rosenberg, J.Weinberger and R.Mahy. Rfc 3489 - STUN - simple traversal of UDP through NATs. USA, March 2003. RFC Editor.

[12] P. Ruth, P. McGachey, X. Jiang, and D. Xu. VioCluster: Virtualization for dynamic computational domains. *IEEE IC on Cluster Computing (Cluster 2005)*, 2005.

[13] X. Jiang and D. Xu. Violin: Virtual internetworking on overlay infrastructure. Technical report, Purdue University, 2003.

[14] J. Chase et all. Dynamic virtual clusters in a grid site manager. *The 12th International Symposium on High Performance Distributed Computing)*, 2003.

[15] K. Appleby et all. Oceano-SLA based management of a computing utility. *In Proc. 7th IFIP/IEEE International Symposium on Integrated Network Management*, 2001.

[16] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, and B. Palter. Layer two tunneling protocol l2tp. USA, August 1999. RFC Editor.

[17] J. Reynolds and J. Postel. Rfc 1340 - assigned numbers. USA, 1992. RFC Editor.

[18] A. Wool. A quantitative study of firewall configuration errors. *IEEE Computer*, volume 37, number 6, pages 62-67, 2004.

[19] R. Jones. Netperf: http://netperf.org/, 1999.

[20] D. H. Bailey et all. The NAS parallel benchmarks: summary and preliminary results. In *Proceedings of the 1991 ACM/IEEE conference on Supercomputing*, pages 158-165, NY, USA, 1991. ACM Press.

[21] LF Ten Eyck, J Mandell, VA Roberts, and ME Pique. Surveying molecular interactions with DOT. *Proc. ACM/IEEE SC 1995 Conference*, 1995.

[22] http://www.lri.fr/ rezmerit/dsllab/

[23] http://www.openssl.org/