Efficient Message Passing on Multi-Clusters: An IPv6 Extension to Open MPI

Christian Kauhaus Adrian Knoth Thomas Peiselt Dietmar Fey
{kauhaus,adi,peiselt,fey}@cs.uni-jena.de
Institute of Computer Science, Friedrich-Schiller-Universität Jena, Germany

Abstract

At our university, different institutes have installed their own cluster computers. Connecting several of these clusters to perform distributed high-performance computing requires message passing spanning heterogeneous network structures. One problem is that private IPv4 addresses inside clusters, although common and suitable for internal communication, preclude end-to-end connectivity. To establish multi-cluster message passing in such a context, we propose to use MPI over IPv6. In this article, we present our IPv6 extension to Open MPI, which is able to cope with mixed IPv4/IPv6 environments and delivers high performance levels.

1 Introduction

The increased use of cluster computing for scientific simulation has led over the years to the installation of independent, small-to-medium-sized cluster computers throughout the departments of our university. Because of our high bandwidth campus network with Gigabit-Ethernet-like performance, distributed high-performance computing spanning several department clusters seems to be within reach. This would allow programs using MPI to run on multiple clusters in parallel for very demanding jobs.

Transparent message passing between multiple clusters requires end-to-end connectivity of some kind throughout all nodes. However, the typical networking architecture of clusters at our university and similar places fails to provide such connectivity. Due to the shortage of IPv4 addresses, compute nodes connected to the clusters' internal Ethernet networks get only private addresses [1] and are thus not reachable from the outside. This fundamental problem gave rise to several different gatewaying and proxying solutions [2]. Most of these perform message routing at a higher level than ordinary network routing, e. g. with user space daemons. This complicates the setup and management of multi-clusters more than necessary and also affects performance.

As a solution, we propose the use of message passing over IPv6 as a very promising alternative to current approaches. We present an IPv6 [3] extension to the Open MPI library [4] and our first experiences with multi-cluster setups using Open MPI over IPv6. Due to the much larger address space, every node is now allocated a globally unique IPv6 address. This re-establishes the end-to-end principle and moves routing back to the place where it is done best, namely to the TCP/IP stack of the operating system or even specialised hardware. Since no extra daemons are required anymore, the setup of multi-cluster message passing is simplified. Additionally, our Open MPI extension is fast enough to provide near-wire-speed throughput and low latencies even for cluster-to-cluster communication. This has been made possible by recent IPv6 implementations like those found in the Linux 2.6 kernel series, which offer competitive performance.

This paper is organised as follows. In Section 2, we introduce related work on multi-cluster message passing and the general architecture of Open MPI. Section 3 explores how the use of IPv6 simplifies the overall setup and how Open MPI needs to be extended to accomplish this. Section 4 presents performance measurements and Section 5 concludes with a summary.

2 Background

2.1 Related Work

Recent advances in Grid technology provide solutions for important infrastructure problems in distributed computing, like (co-)scheduling, job launch, file access etc. Some packages, for example the Globus Toolkit, include Grid-enabled MPI libraries like MPICH-G2 [5]. However, these assume full IPv4 connectivity, that means public IPv4 addresses on all participating nodes, and hence do not consider routing disruptions as found in our scenario. To deal with such problems, several solutions have been proposed.

Some systems realise message passing over disconnected networks with user space daemons. Examples include PACX, Stampi or MPICH/Madeleine [6–8]. All of these divide their universe into cells with inner connectivity, and connect multiple cells using a gateway or proxy. Since the configuration of cell boundaries and the startup of daemons require manual intervention, these solutions cannot be considered optimal. Also, the obtainable performance levels of user space daemons are generally not adequate for Gigabit networking [9]. Reasons for this include that one additional kernel–user–kernel round trip is necessary and that a daemon cannot access NIC interrupts.

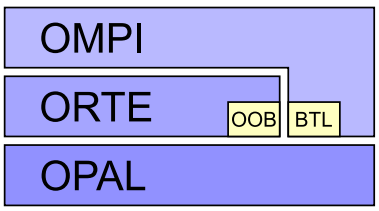

Fig. 1
Open MPI's architecture consists of the three main modules OMPI, ORTE, and OPAL.

VPN-based solutions yield improved performance at least for kernel-based implementations, but also face management difficulties [10]. Problems include that administrators need to coordinate private address spaces for all clusters to prevent overlaps. Connecting multiple clusters via VPNs is still a viable solution in some cases.

The research works most similar to ours are the IPv6 extension to PVM [11] and the MPICH/IPv6 experiments [12]. Replacing all network calls with their IPv6 equivalents, as seen in these two cases, breaks backward compatibility with existing IPv4 installations. Although this approach is adequate for a proof of concept, it is not sufficient for software intended to run in production environments.

2.2 Architecture of Open MPI

The main reason why we chose Open MPI is its carefully designed Modular Component Architecture (MCA), which breaks all of the functionality into narrowly scoped modules that can be modified independently [13]. Our implementation is based on Open MPI 1.2, which itself already supports multi-protocol communication and thus allows to include support for both TCP/IPv4 and TCP/IPv6 without major architectural changes.

To identify the relevant parts for our IPv6 extension, we have to consider all three layers of the Open MPI architecture, as shown in Figure 1: the *Open Portable Access Layer* (OPAL), the *Open Run-Time Environment* (ORTE) and the *Open MPI* functionality (OMPI). OPAL is a collection of supporting functions providing hardware and OS abstractions. ORTE is a generic run-time layer providing infrastructure for process management and I/O-handling via its own communication channel, the *Out-of-Band Communication* (OOB). An important use of the OOB is the *wire-up,* which is the initial establishment of OOB channels between all participating processes. OMPI contains its own high performance communication, the *Byte Transfer Layer* (BTL). The BTL is independent of the OOB, since BTL selects the "best" network transport in case there are several (i. e. Myrinet, Infiniband), while OOB always opens a standard TCP connection. If there are multiple network paths between any two nodes, Open MPI optionally uses striping to maximise throughput. With striping, Open MPI fragments big messages and sends them in parallel through several interfaces.

3 Networking Implications

Message passing for distributed high-performance computing can be set up by using IPv6 with significantly lower administrative overhead compared to other current solutions. Once IPv6 has been deployed in a campus network, a suitably extended MPI implementation is able to run on a large variety of distributed setups without requiring manual configuration. The main goal for our IPv6 extension to Open MPI is to make it "just work" regardless of the network context. To achieve this, modifications on all three layers are necessary: for OPAL, the interface handling code has to be adapted; ORTE and OMPI require enhancements to their respective TCP implementations, which are the OOB/TCP and BTL/TCP components. The last two are described in the following.

3.1 Out-of-Band Communication and Wire-Up

During wire-up, before any MPI communication takes place, all processes must establish an out-of-band channel to exchange run-time information with ORTE's *General Purpose Registry* (GPR). The first running process is called *Head Node Process* (HNP). As the process launching mechanism is user-configurable, and hence Open MPI cannot control it, the HNP passively waits for all clients to call back. Information about HNP's listening ports, called *connection string,* is passed to the clients on the command line.

To simplify wire-up, we assume a single cell setup. The single cell concept means that all participating nodes support at least one kind of uniform and unique network addressing. In contrast, multi cell concepts rely on the notion of multiple disjointed networks which need to be connected via gateways. It is easier—although not necessary—to realise such a single cell setup with IPv6, because IPv6 is not short of available global addresses like IPv4. Unfortunately, we cannot expect IPv6 to be always available, as this would break backward compatibility with legacy (IPv4-only) networks. The IPv6-extended library is required to run in environments with and without IPv6 support. As a further complication, most of today's operating systems are IPv6-enabled, but only few networks have a working IPv6 configuration. That is why simply replacing all occurrences of AF_INET with AF_INET6, as it has been done in early experiments, is not sufficient. To ensure backward compatibility, we must consider different address classes: First, there are IPv6 global unicast addresses (i. e., 2001:638:906:1::1). Although

IPv6 also defines other address classes like site-local or link-local [14], these are not useful for our purpose. Second, IPv4 public unicast addresses (i. e., 141.35.14.189) provide the same global connectivity, but may not be available in sufficient quantities. Third, IPv4 private addresses (i. e., 192.168.1.1) are widely used in cluster setups, but are neither routable nor unique.

As the HNP has no way to determine upon startup which addresses are likely to work, we have extended ORTE to search for all locally configured addresses falling into one of these three classes. Now the HNP uses the retrieved list to construct the connection string by concatenating either "tcp" or "tcp6" to denote the address family, the IP address, and the port number. An example connection string looks now like:

0.0.0; tcp6://2001:638:906:2::1:52032;
tcp6://2001:638:906:1::1:52032;
tcp://141.35.14.189:57990;
tcp://192.168.1.1:57990

The HNP sorts its public addresses to define priority. To maximise chances to establish a connection, clients first eliminate all non-locally-present address classes and then try to connect the remaining addresses in order until one succeeds. Thus, private addresses are only used between two nodes if they do not share any public address class. Otherwise, in multi-cluster setups with duplicated private IPv4 addresses the wrong node could be contacted.

The example above shows different port numbers for IPv4 and IPv6. This indicates that the HNP needs separate sockets for each address family, because the socket API requires to explicitly specify the address family on opening a listening socket. Unfortunately, the original OOB implementation was not designed to operate on multiple sockets, and required some code modifications to be able to process events coming from multiple channels. Although this complicates relevant code parts significantly, the only other option, IPv4-mapped IPv6 addresses [15], is worse. IPv4-mapped IPv6 addresses would allow to use only one AF_INET6 socket for both IPv4 and IPv6 connections. But this would limit the applicability of Open MPI to systems where both IPv6 support is enabled and IPv4-mapped IPv6 addresses have not been switched off, as it is commonly the case on BSD-derived systems.

With all of the above mentioned enhancements and modifications, we have extended Open MPI's OOB component to cope with as many different IP network setups as possible to perform a reliable wire-up.

3.2 Byte Transfer Layer

Various BTL components provide communication channels for MPI message transfer. Open MPI dynamically chooses a selection of components depending on detected network adapters. Each module is then responsible for connections to remote nodes reachable via one particular link. For TCP, the original implementation used to create one BTL/TCP instance per kernel interface, relying on the assumption that each NIC has exactly one IPv4 address configured and can thus be represented by it. If multiple addresses are found both locally and for a given peer, Open MPI assumes that both ends share multiple network connections and uses striping to distribute messages over all available transports concurrently.

In IPv6-enabled systems, it is common for single interfaces to have more than one address configured. Simply extending the existing concept to create one module per address might lead to superfluous message striping between IPv4 and IPv6 modules on the same link, thus causing NIC oversubscription, increasing overhead and finally reducing overall throughput. To circumvent this problem, our implementation still instantiates one module per interface, but manages 1-to-n associations between interfaces and addresses, hence preventing local oversubscription. Unfortunately, it is still possible to overload remote nodes' NICs by opening multiple connections to different peer addresses ending on the same remote interface. We therefore do not only exchange addresses but also their corresponding kernel interface numbers, thus enabling peers to identify which remote addresses reside on the same device. As a consequence, we establish at most $\min\{\#\textit{local interfaces}, \#\textit{remote interfaces}\}$ connections between peers.

If there are now several matching address pairs available for a given interface pair between two nodes, each BTL/TCP module must select the peer address which is most likely to work. These addresses are determined in the same way it is done in the OOB component: prefer public address classes and only use private IPv4 addresses as last resort. Additionally, each module is constrained to remote interfaces not already in use by other local BTL/TCP instances.

The resulting enhanced BTL component provides TCP message transport for a wide range of IP network configurations and uses the available bandwith via message striping while preventing link oversubscription.

3.3 Infrastructure Issues

Beyond the direct implementation consequences, the transition towards IPv6 to perform multi-clustering on a campus network has several side effects which should also be considered. In the following we will discuss two important ones, IPsec support and LAN routing.

By using IPv6, we can take advantage of the widespread availability of the IP security protocol suite (IPsec) as a standard encryption option [16]. Since the IPv6 specifications recommend IPsec, it can be expected to be widely available. Thus, IPsec

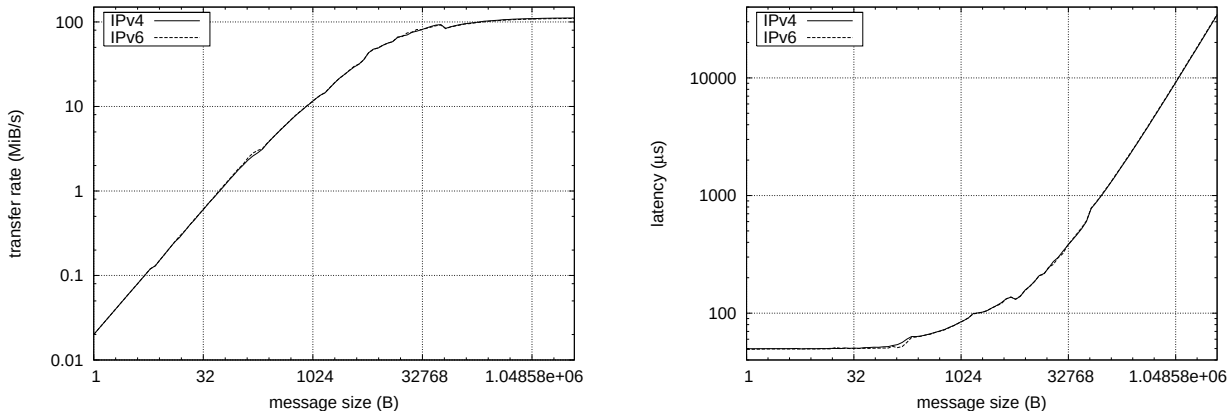

Fig. 2
Intra-cluster ping-pong benchmarks for IPv4 and IPv6 show hardly any difference for (a) throughput and (b) latency.

is a good candidate for a default data authentication and encryption protocol to protect inter-cluster traffic. Besides its good availability, IPsec supports automatic key exchange via IKE, and therefore requires little administration. It is still left to further research to evaluate the general feasibility, best mode of operation, and performance of the use of IPsec in such a context.

Obviously, native IPv6 connectivity on a campus network requires IPv6 support on local routers. Although IPv6 is well known for years, some router manufacturers still do not support it by default. For example, to use IPv6 on popular Cisco routers, one has to acquire the costly "Advanced IP Services" feature set. However, we expect especially U. S. based vendors to change their product strategy in near future as IPv6 is gaining acceptance. For example, the Pentagon is planning a transition to IPv6 for military networks beginning in 2008 [17].

Although the deployment of IPv6 and the adaptation of applications requires some effort, we think that the gained advantages outweigh the costs. This section has shown that a MPI implementation can be extended in such a way that it allows to connect multiple clusters easily. In the next section, we investigate the performance of our IPv6-extended Open MPI in typical situations.

4 Performance Evaluation

IPv6 message passing delivers decent performance and is therefore worth considering in multi-cluster setups. It is justified to expect good results since IPv6 routing is done at kernel level. Proxy- or gateway-based solutions are facing an additional overhead of handling packets in user space, which not only includes traversing the TCP/IP stack twice, but also lacks interrupt-driven access to network adapters.

The performance evaluation is done in two steps: first we compare intra-cluster performance of IPv6 with IPv4, followed by determining the efficiency in a typical multi-cluster environment using Open MPI with IPv6.

4.1 IPv4 Versus IPv6

Due to the increased size of IPv6 headers compared to IPv4 headers, IPv6 performance will naturally be below that of IPv4. To evaluate Ethernet performance of IPv6 in comparison to IPv4, we measure throughput and latency between two compute nodes on the same cluster using the Intel MPI Benchmark Suite (IMB) 2.3. The benchmarks have been executed on two identical machines with dual AMD Opteron 250 processors running Linux 2.6.18.6 connected via Broadcom BCM5704 Gigabit Ethernet adapters to a Netgear ProSafe GS724T switch.

Figure 2 shows the throughput and latency for the IMB ping-pong micro-benchmark. The results indicate that IPv6 throughput is almost on par with IPv4, maxing out at 110.0 MiB/s whereas IPv4 reaches its maximum at 111.5 MiB/s. The measured drop of 1.4% is very close to the expected loss of 1.37% caused by the decreased maximum segment size (1460 B for IPv4 vs. 1440 B for IPv6) resulting from larger IP headers. The latency figures do not show any noticeable difference beyond the precision of measurement, making IPv6 a competitive solution even for intra-cluster communication.

4.2 End-to-End Communication

To evaluate the end-to-end communication performance of our IPv6 implementation in a realistic multi-cluster scenario, we employ two separate clusters A and B, as shown in Figure 4. Both head nodes are connected to the faculty network through a layer 3 switch, a Cisco Catalyst 6509. Cluster A con-

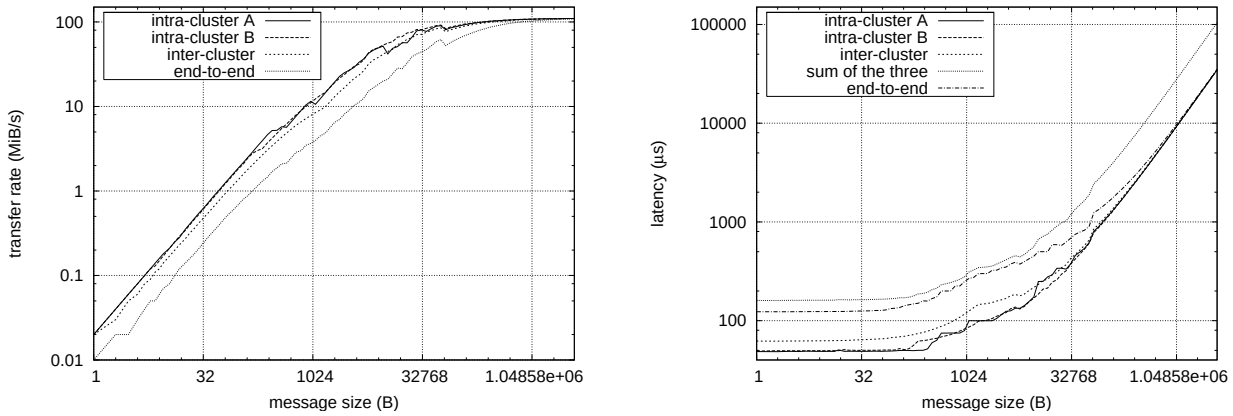

Fig. 3
End-to-end ping-pong performance is compared to the performance of the individual links with regard to (a) throughput and (b) latency.

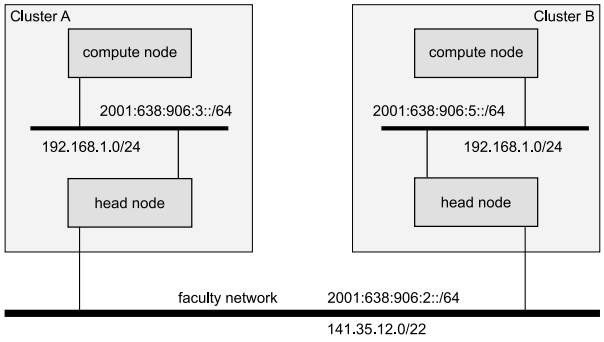

Fig. 4
Clusters A and B with private IPv4 addresses inside are connected to a public network.

tains AMD Opteron 250 compute nodes connected by Broadcom BCM5704 Gigabit Ethernet adapters, and an identical head node connecting to both the intra-cluster-network and the faculty network using the same Broadcom Ethernet adapters. Both nodes are running Linux 2.6.18.6. Cluster B contains a head node identical to the head node of cluster A, and compute nodes with 3 GHz Intel Pentium 4 processors running Linux 2.6.18.1, connected to the network using Intel 82547EI Gigabit Ethernet adapters. A message sent from a compute node of cluster A to a compute node of cluster B is passed via the intra-cluster network to the head node of cluster A, then traverses the faculty network to the head node of cluster B and finally reaches its destination via the intra-cluster network of cluster B. The IMB ping-pong benchmark has been used to provide latency and throughput figures. In addition to the performance measurements for the connection between two compute nodes (end-to-end communication), we have also measured each link independently.

The maximum throughput achieved by the IPv6 end-to-end link, as shown in Figure 3(a), is very close to the maximum throughput of the weakest link, which is the connection between the head nodes over the faculty network.

Figure 3(b) shows the latency for end-to-end communication and for each network connection separately. Additionally, the latencies of the three individual links are summed up. The end-to-end latency is lower than the sum of the individual latencies, because measurements for each link include TCP and MPI processing overheads. These overheads are included three times when adding the values, whereas they account only once for the end-to-end latency. However, any solution employing user space gateways does not have this advantage. We expect the latency in such a scenario to be higher than the sum of the individual links' latencies. Because of the decreased overhead, in-kernel routing performs inherently better than even well tuned gatewaying. Furthermore, connecting clusters using Virtual Private Networks generally yields higher latency and lower throughput on comparable hardware [10].

The measurements show that message passing based on IPv6 is worth further consideration. The performance penalty compared to IPv4 is barely noticeable, and due to efficient IPv6 routing, inter-cluster communication via IPv6 is expected to outperform solutions run in user space.

5 Conclusions

We have outlined that message passing over IPv6 is a promising approach to distributed computing on multi-clusters in a campus network. One big advantage compared to other solutions, which often involve user space gateways, is the simplified setup. Once IPv6 routing has been established, it is achievable to get multi-cluster message passing that works "out of the box." As another advantage, we are able to obtain low end-to-end latencies and throughput of

more than 100 MB/s by relying on built-in kernel routing. Our IPv6-extended version of Open MPI is fast enough to saturate connections crossing cluster boundaries at Gigabit Ethernet speed. Thus, we would like IPv6 to get more attention in the high performance community, as it reveals a great potential for fast and easy multi-clustering. This could in turn have positive effects on Grid Computing.

Currently, we are gathering more experiences how IPv6-enabled Open MPI behaves in various network setups. This concerns also our work on integrating our changes into the official Open MPI code base. Furthermore, still unresolved issues include costs with IPv6 deployment on popular router hardware and a lack of experience with IPsec regarding the best mode of operation. In the future, MPI development should focus more on topology aware implementations of MPI's collectives (i. e., Open MPI's coll_hierarch component) to fully take advantage of hierarchical network structures like local Infiniband and campus-wide Ethernet.

References

[1] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear, "Address Allocation for Private Internets," RFC 1918 (Best Current Practice), Internet Engineering Task Force, Feb. 1996. [Online]. Available: http://www.ietf.org/rfc/rfc1918.txt

[2] R. Keller, E. Gabriel, B. Krammer, M. S. Müller, and M. M. Resch, "Towards efficient execution of MPI applications on the Grid: Porting and optimization issues." *J. Grid Comput.*, vol. 1, no. 2, pp. 133–149, 2003.

[3] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460 (Draft Standard), Internet Engineering Task Force, Dec. 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2460.txt

[4] R. L. Graham, G. M. Shipman, B. W. Barrett, R. H. Castain, G. Bosilca, and A. Lumsdaine, "Open MPI: A high-performance, heterogeneous MPI," in *Proceedings, Fifth International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks*, Barcelona, Spain, September 2006.

[5] N. T. Karonis, B. Toonen, and I. Foster, "MPICH-G2: A Grid-enabled implementation of the message passing interface," *J. Parallel Distrib. Comput.*, vol. 63, no. 5, pp. 551–563, 2003.

[6] E. Gabriel, M. M. Resch, T. Beisel, and R. Keller, "Distributed computing in a heterogeneous computing environment." in *EuroPVM/MPI*, ser. Lecture Notes in Computer Science, V. N. Alexandrov and J. Dongarra, Eds., vol. 1497. Springer, 1998, pp. 180–187.

[7] T. Imamura, Y. Tsujita, H. Koide, and H. Takemiya, "An architecture of Stampi: MPI library on a cluster of parallel computers." in *EuroPVM/MPI*, ser. Lecture Notes in Computer Science, J. Dongarra, P. Kacsuk, and N. Podhorszki, Eds., vol. 1908. Springer, 2000, pp. 200–207.

[8] O. Aumage, "Heterogeneous multi-cluster networking with the Madeleine III communication library," *IPDPS*, p. 85b, 2002.

[9] M. Müller, M. Hess, and E. Gabriel, "Grid enabled MPI solutions for clusters," in *Proc. 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2003)*. Los Alamitos: IEEE Computer Society, 2003.

[10] C. Kauhaus and D. Fey, "Building Mini-Grid environments with Virtual Private Networks: A pragmatic approach," in *Proc. PARELEC, Bialystok, Poland, September 13–17*. Los Alamitos, CA, USA: IEEE Computer Society, 2006, pp. 111–115.

[11] R. Martínez-Torres, "PVM-3.4.4 + IPv6: Full grid connectivity." in *EuroPVM/MPI*, ser. Lecture Notes in Computer Science, B. D. Martino, D. Kranzlmüller, and J. Dongarra, Eds., vol. 3666. Springer, 2005, pp. 233–240.

[12] L. Schneidenbach and B. Schnor, "Migration of MPI applications to IPv6 networks," in *Proc. Parallel and Distributed Computing and Networks (PDCN 2005)*, T. Fahringer and M. H. Hamza, Eds. Calgary: ACTA Press, 2005.

[13] E. Gabriel, G. E. Fagg, and G. Bosilca, "Open MPI: Goals, concept, and design of a next generation MPI implementation," in *Proceedings, 11th European PVM/MPI Users' Group Meeting*, Budapest, Hungary, September 2004, pp. 97–104.

[14] R. Hinden and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture," RFC 3513 (Proposed Standard), Internet Engineering Task Force, Apr. 2003, obsoleted by RFC 4291. [Online]. Available: http://www.ietf.org/rfc/rfc3513.txt

[15] R. Gilligan, S. Thomson, J. Bound, J. McCann, and W. Stevens, "Basic Socket Interface Extensions for IPv6," RFC 3493 (Informational), Internet Engineering Task Force, Feb. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3493.txt

[16] S. Kent and K. Seo, "Security Architecture for the Internet Protocol," RFC 4301 (Proposed Standard), Internet Engineering Task Force, Dec. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4301.txt

[17] J. P. Stenbit, "Internet protocol version 6 (IPv6)," Department of Defense, Washington, DC, Memorandum for Secretaries of the Military Departments [...], Jun. 2003. [Online]. Available: http://www.defenselink.mil/news/Jun2003/d20030609nii.pdf