Security of High-Performance Messaging Layers on Programmable Network Interface Architectures

Srigurunath Chakravarthi
ecap@cs.msstate.edu

Department of Computer Science
Mississippi State University
Box 9637, MS 39762

Sponsoring Professors: Dr.Rayford Vaughn and Dr.Anthony Skjellum
Faculty of the Dept. of Computer Science, Mississippi State University

Abstract

This paper discusses security concerns introduced by high-performance messaging layers designed for distributed High Performance Computing (HPC) environments. In the recent past, computationally intensive scientific applications have begun to find high-speed Networks of Workstations (NOWs) as an economically attractive alternative to parallel computers. Conventional distributed systems have been studied and modeled for security. However, these models are not directly applicable to HPC systems primarily due to the design goals of the underlying system software, such as high-bandwidth and low-overhead communication and scalability. New security concerns also arise from the programmability of the network interfaces of the underlying high-speed network technologies. This paper presents guidelines to improve the assurance of typical high-performance messaging protocol layers operating in a System Area Network (SAN).

This paper identifies that an integral part of the security of any performance critical system is providing immunity from performance-degrading attacks. A secure messaging system should minimize threats to the availability of limited and shared resources in order to preserve the system's performance. The performance goals of HPC systems severely restrict the choice of usable security mechanisms, because of the inevitable processing overhead introduced. In view of this restriction, a trade-off model has been suggested to achieve the desired balance between performance and assurance.

Keywords: Secure Message Passing, Distributed System Security, High Performance Computing, Network Interface Architecture, Gigabit Network, Network Of Workstations, System Area Network, Myrinet.

1. Introduction

Traditionally, parallel super-computers have been the only viable choice for computationally intensive scientific applications [12]. While much of the scientific community still operates in the domain of super-computers and multi-processor computers, high-speed networks of workstations are fast becoming a popular alternative [12]. Emerging network technologies like Myrinet [5], ATM, and SCI [21] enable data transfer at the rates of Gigabits per second, thus achieving speeds comparable to that between processors in a single parallel computer. Low level messaging layers like BDM [11], Fast Messages [17], BIP [4], and GM [10] are designed to provide highly efficient protocols over these networks, to create a base for high performance distributed computing. The communication protocols implemented by these layers differ significantly from conventional protocols such as TCP/IP. The requirement of providing low-overhead communication potentially introduces exploitable vulnerabilities in such messaging software. High-speed network interface architectures often consist of a network co-processor, memory (RAM) and other hardware components that open up new paths for user applications to access the system's hardware resources. The vulnerabilities discussed in this paper are specific to high performance systems, and have not been dealt with in security models for several traditional distributed computing environments.

The inability to compromise performance for better assurance restricts the choices of security mechanisms that can be implemented in a high performance system. This paper examines some of the security issues in the design of new high-speed network technologies, and low level messaging layers. General solutions have been offered to many of these issues, keeping in view the primary requirement of a tolerable overhead. Trade-offs are inevitable in a situation where the cost of introducing security mechanisms is unacceptable. Some guidelines for the design of such systems have been suggested, so as to minimize the requirement of additional high-overhead security mechanisms.

Throughout the paper, a popular System Area Network technology, Myrinet [15], is cited while examining the low level software architecture of high performance systems. Myrinet is a widely used SAN technology in high-performance domains (refer to Figure 2). Myrinet supports full-duplex connections operating at 1.2 Gigabits/sec, and Myrinet cut-through switches route based on message header bits, with latencies as low as 0.5 microseconds. No particular implementation of the messaging layer has been specifically dealt with, to keep the discussion general. However, BDM [11], a multi-protocol messaging library on Myrinet, has been used as an example.

The paper is organized as follows: Section 2 defines the distributed HPC environment discussed in this paper, and compares it to a traditional parallel computer from a security viewpoint. Section 3 presents the software architecture of a message-passing HPC system, and enumerates the role of high-performance messaging layers. Section 4 describes the security requirements of Confidentiality, Integrity and Availability, and identifies the chief vulnerabilities introduced by the performance-goals of communication software and hardware. Section 5 presents guidelines for the design of a secure high-performance messaging system. Section 6 is the conclusion.

2. Parallel and Distributed Environments

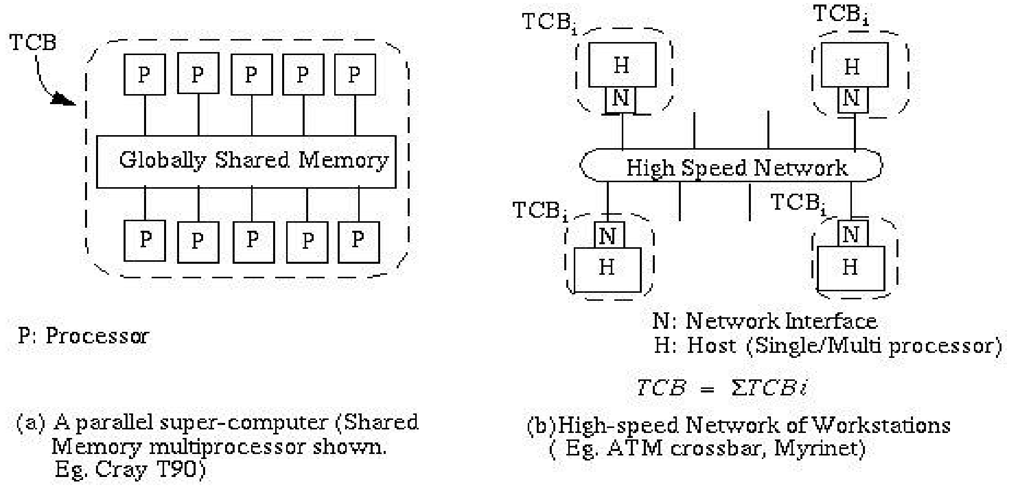
TCB
P
Globally Shared Memory
P: Processor
(a) A parallel super-computer (Shared Memory multiprocessor shown. Eg. Cray T90)
TCB$_i$
H
N
High Speed Network
N: Network Interface
H: Host (Single/Multi processor)
$TCB = \Sigma TCBi$
(b)High-speed Network of Workstations (Eg. ATM crossbar, Myrinet)

Figure 1 Security perimeters in Parallel and Distributed High Performance environments.

The HPC environment discussed in this paper is a distributed system comprising of a high-speed System Area Network (SAN) of single- or multi-processor computers (see Figure 1(b)). Such a Network Of Workstations (NOW) can function as a virtual super-computer, using high-speed communication among processors on different hosts. Parallel super-computers such as CRAY, SP2, and CM-5 achieve high computational power by using the raw CPU power of multiple processors, to execute instructions parallelly in a single machine [12]. Figure 1(a) shows a shared-memory parallel architecture using uniform memory access.

Two of the main differences between a parallel and a distributed HPC system, with regard to their security, are (refer to figure 1):

1. The data transfer mechanism between processors in a parallel computer is internal to the system hardware, unlike in a networked environment. In other words, the data path traversed between processors in a parallel computer is encapsulated totally within the security perimeter of the Network Trusted Computing Base[7,16], whereas this is not the case with a network of computers.
2. A messaging layer that is external to the native host Operating System implements a high-speed communication interface between nodes on the network.

3. Myrinet-based Hardware and Software Architectures

The components of a Myrinet network are shown in Figure 2.

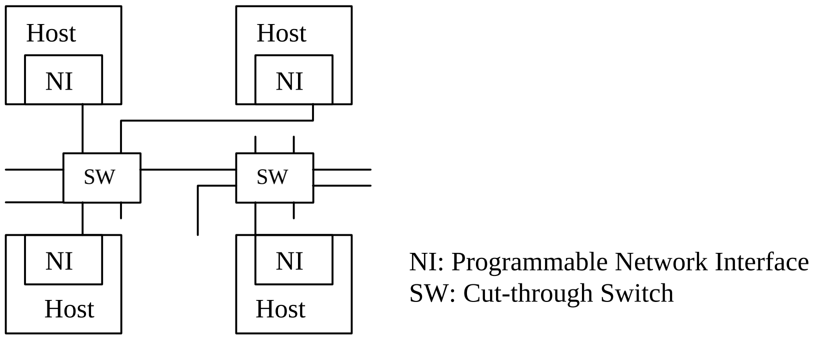

Figure 2: Myrinet - a high-speed System Area Network

Compared to a conventional Ethernet Local Area Network (LAN), the following differences hold for Myrinet SANs (see Figure 2):

- Topologically, they are point-to-point networks with nodes interconnected by switches, whereas an Ethernet LAN is a bus-based broadcast network.
- Network data is routed by low-latency switches, based on the header bits (i.e. source routed). Intermediate hosts are absent. Thus, there is no *Storing-And-Forwarding* of data that is present in conventional IP routing.
- The Network Interface (NI), consists of an on-board processor, memory and Direct Memory Access (DMA) engines. This *intelligent* NI, is absent in conventional LANs.
- They are capable of providing significantly higher bandwidth (Gigabits/sec) *between network interfaces* of interconnected nodes.
- They provide a degree of reliability in the Link Layer [24] itself (1 bit error in every 2^{15} bits). The upper protocol layers of software can take special advantage of this reliability to achieve high bandwidths.

Our attention to security will be focussed on the messaging library software built upon a Myrinet SAN. However, at this point we draw some straightforward conclusions about the security implications of SANs. A small-sized and closed network (in a laboratory room, ship or aircraft) implies that physical security can be reliably achieved. Hence we will assume throughout our discussion that the network is physically secure, and do not discuss threats arising from an intruder gaining physical control of one or more nodes in the network. Confidentiality threats emerging from electromagnetic emanations are also assumed to be absent.

Point-to-point connections reduce the confidentiality threat of network sniffing. The absence of intermediate hosts reduces threats arising from interception and re-transmission of messages and traffic analysis at message hops.

The above-mentioned properties tempt us to infer that SANs are more secure than traditional LAN based distributed environments [3]. This is, however, not the case because their host interface architecture introduces several vulnerabilities which are absent in a LAN interface. The Network Interface Controller (NIC) of Myrinet is programmable. It has an on-board processor (called LANai), and up to a megabyte of SRAM. The LANai processor can access the entire host memory using an on-board DMA engine. DMA transfers can also be initiated by host programs via memory-mapped DMA control registers residing on the Myrinet board. The LANai processor runs a custom-built program called the Myrinet Control Program (MCP) [22]. The presence of a network processor running untrusted custom code and a DMA engine on hardware with minimal built-in protection mechanisms, is essentially insecure. Myrinet based high-performance communication layers designed without special considerations to security can potentially expose these architectural deficiencies to user programs. These potential vulnerabilities are discussed after presenting the software architecture.

The software architecture of a typical message-passing based distributed system is shown in figure 3. The vulnerabilities introduced by the shaded portions are of interest to this paper.

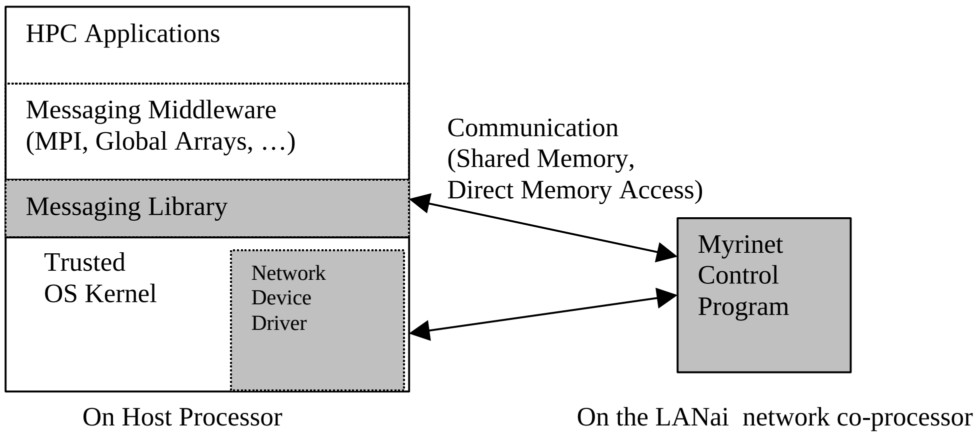
HPC Applications
Messaging Middleware (MPI, Global Arrays, …)
Messaging Library
Trusted OS Kernel
Network Device Driver
Communication (Shared Memory, Direct Memory Access)
Myrinet Control Program
On Host Processor
On the LANai network co-processor

Figure 3: HPC software architecture of a typical message passing system on Myrinet. Vulnerabilities introduced by the shaded regions are discussed in this paper.

The custom-built MCP runs on the network co-processor. This program handles the sending and receipt of network data, thus off-loading these duties from the host processor. The MCP works in co-operation with the messaging library to handle data transfers between host memory and on-board buffers. Incoming and outgoing data is buffered at the network interface.

The messaging library interface is very similar to a TCP/IP socket communication interface. A primary requirement of this library is to provide high-bandwidth and low processing overhead communication. BDM [11], Fast Messages [17], and GM [10] are examples of popular high-performance messaging layers. Normally, a middle-ware layer provides an Application Programmer's Interface (API) for HPC applications. While discussing security concerns in this paper, we refer to user programs, which are programs written using the messaging library's API, i.e. the middle-ware is absent.

As a background to the following sections, we need to specify the meaning of three terms of significance in high-performance message-passing computing environments, as used in discussions ahead. *Bandwidth* is the maximum rate at which user processes on the network can communicate their data. This bandwidth is typically much lower than the raw bandwidth offered by the SAN. *Latency*, or end-to-end latency, is the total time taken by a message from the beginning of its construction at the sending user program to its receipt by the receiving user program. Low latency is desired for better performance. *Direct Memory Access* (DMA) is a mechanism of data transfer that allows peripherals to read and write to memory without involving the CPU. DMA transfers are important to achieve high bandwidths, because the CPU can do useful computation during a DMA transfer. DMA transfers discussed in this paper refer to those between host memory and network board memory.

4. Security Implications for HPC networks

The TCSEC Trusted Network Interpretation [16] provides U.S. national security evaluation criteria for a networked environment. The concept of a Network Trusted Computing Base [6] has been adopted by traditional secure distributed systems[19,20]. These criteria and models cannot easily be applied to a distributed environment dedicated to high performance computing because of differences in network interface architectures and performance goals of messaging-layers that impose constraints on their design.

The most important difference is the design philosophy of a high performance system. Efficient systems are designed to endow user programs with as much of the underlying raw hardware power as possible. This is normally achieved by mechanisms such as reducing the layers between hardware resources and user level software, minimizing context switches from user to kernel mode, and eliminating extra copying of data within the host memory itself, sometimes even by bypassing the OS kernel. Consequently, one of the basic requirements of a security kernel, that of *Completeness* [6] stands violated, as shown in Figure 4.

Figure 4(a) shows the software architecture and access paths for hardware resources with a typical performance-driven messaging layer over a programmable network interface. Figure 4(b) shows the conventional

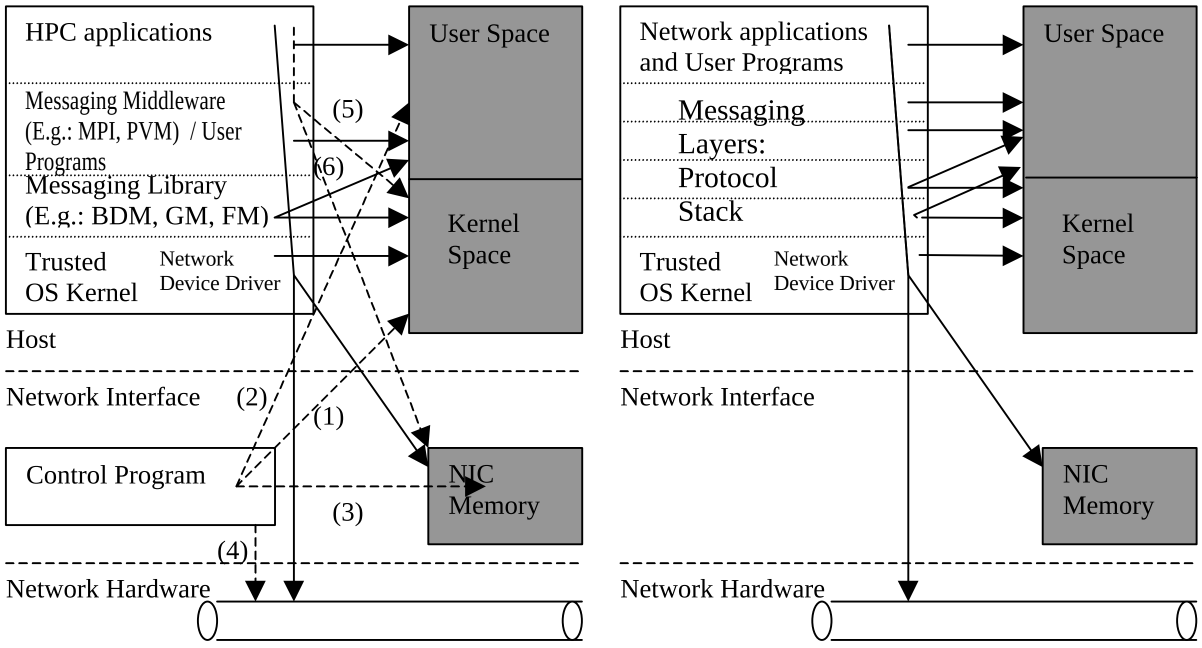

Figure 4: Data access paths of (a) high-performance messaging layer with Myrinet network interface, (b) conventional network protocol layers over Ethernet

network software architecture designed on lines of the OSI software model [24]. The bold arrows, which are common to both figures, show secure resource access paths. In figure 4(b), all accesses to kernel space memory and network hardware arise from the trusted OS or only from the lowest layer(s) of the network protocol stack. User programs can never bypass these trusted software layers. These rules do not apply in high-speed messaging layers.

Figure 4(a) shows the newly introduced access paths are shown as dashed arrows. The programmability of the NIC is a fundamental difference between the two environments. The presence of an intelligent (programmable) network processor directly introduces four new access paths numbered 1,2,3, and 4 in Figure 4(a). In a Myrinet interface, the LANai network processor can access user and kernel memory via on-board

DMA engines. Access to kernel space (via path 1) by the control program – an untrusted piece of code - can potentially cause vulnerabilities unless special considerations are taken in the design of the control program. Path 2 can potentially cause confidentiality violations between multiple user programs on one host that are serviced by the control program. Paths 3 and 4 are a consequence of the host processor offloading the buffering of incoming and outgoing network messages to the network processor. These paths can introduce threats in the presence of vulnerabilities in the control program.

Access path 5 allows accesses to kernel pages that are mapped to user space by the network device driver. Mapping of kernel memory to user space is one popular technique to implement zero-copy data transfers. Although a trusted device driver does the mapping, this driver typically does not validate accesses to and from this kernel region in order to avoid user-to-kernel mode context-switch time. Path 6 provides direct access to network buffers to user applications written directly over the low-level messaging layers (or to the messaging middle-ware, if present). These access paths are the result of a technique called user-DMA. Its implementation in a certain flavor of BDM [13] is described below.

Figure 5(b) shows a potentially insecure data transfer path via a mechanism known as *user DMA*, usually provided by an efficient messaging system to eliminate making extra copies of data before it is transferred to the network buffers. User DMA also eliminates context-switching time to kernel. User DMA by itself is not necessarily insecure, but while providing such a mechanism, messaging libraries should be extremely carefully designed not to allow the user to misuse it. This zero-copy data transfer mechanism is shown in Figure 5(b), and compared to a single-copy DMA transfer shown in 5(a). User DMA allows user programs to initiate a DMA transfer to or from user memory via the messaging library, but without mediation from the trusted OS. Thus, it is the responsibility of the messaging layer to validate the source and target addresses and the data length associated with every DMA transfer.

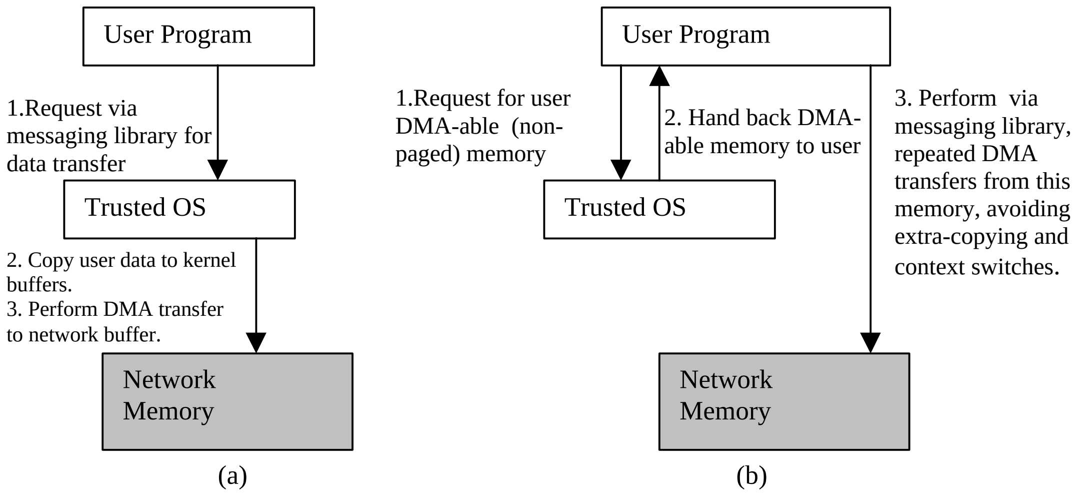
User Program
1.Request via messaging library for data transfer
Trusted OS
2. Copy user data to kernel buffers.
3. Perform DMA transfer to network buffer.
Network Memory
(a)
User Program
1.Request for user DMA-able (non-paged) memory
2. Hand back DMA-able memory to user
Trusted OS
3. Perform via messaging library, repeated DMA transfers from this memory, avoiding extra-copying and context switches.
Network Memory
(b)

Figure 5: Steps involved in transferring data to NIC buffers using (a) traditional DMA (b) user DMA

Data can be in three states: in the host memory, in the NIC buffers, and in transit on the network. Confidentiality, integrity and availability of this data is required in all three states.

Security of Data in Host Memory

Typically, confidentiality and integrity of data in host memory is provided by the Trusted Computing Base (TCB) [7]. In the presence of a programmable NI and on-board DMA engines, however, additional data paths are introduced as discussed above. The messaging software often entails short-cuts to reduce hardware access time for user programs.

Security of Data in NIC buffers

Threats to data in NIC memory arise from the lack of hardware protection mechanisms such as write-protection and virtual memory for the network processor and memory. Myrinet provides a primitive write-protection mechanism for a programmable range of the on-board memory, but this protection does not apply to writes initiated from the host. Thus, it is possible to overwrite a currently executing control program by another one. A maliciously overlaid MCP can provide access to the entire host memory via the on-board DMA engines.

Security of Data in transit

The topology and hardware routing characteristics of Myrinet, as specified earlier, reduces threats to confidentiality and integrity of data in transit. Data encryption, a fundamental tool for the protection of data in transit [2], is not being used by most HPC messaging layers. Software data encryption is CPU-intensive, and thus infeasible to implement in software. If encryption is desired, it is clear that the network interface hardware should provide it.

Availability Requirements

Availability refers to unhampered access to legitimate information, and to a fair share of system resources [1]. Availability threats have been addressed separately in this sub-section because the author believes that such threats are most detrimental to the performance of the system. The performance of the entire system is adversely affected if access to shared resources is influenced by even one process. Additionally, availability attacks are hard to detect or avoid because of extensive resource-sharing among various processes.

The critical system resources are network bandwidth, CPU time, host memory and network buffer space. Specific threats to availability can arise by exploiting memory access priorities associated with a shared memory resource accesses by the various processing units of the system. For instance, repeated polling of LANai memory by the host processor, can starve LANai access to its own memory [14].

5. Design Guidelines for Secure High Performance Messaging Layers

A fundamental element of building a secure system [9,18,23] is to engineer a security architecture from the beginning stage - a principle not being followed by the emerging HPC domains. Many of today's high-performance environments are not designed with a special consideration given to security. The pace of SAN technology, and lack of a uniform standard for high performance architectures have been identified by [8], as a major source of security issues. We now proceed to list some general guidelines for the design of a secure high performance distributed system, particularly a secure messaging layer, with a goal of minimizing security overhead.

Secure Library Interface for DMA

A library that provides zero-copy user DMA should be designed to validate all DMA parameters such as addresses and data-length, for every transfer. If kernel pages are mapped to user-space to enable user DMA, it is essential that information about these DMA-able buffers (such as the physical address and size) be opaque to the user. Validation steps will add to processing overhead, but this is inevitable if security is desired.

Often, a messaging library may use a single buffer pool for allocation requests from any user program. If buffer re-use among different processes is present, it is essential to zero-out the contents of every buffer before assigning it to a new process.

Maximizing Set-up Time Operations

The design of a secure HPC system should aim at incorporating much of the security related operations at the set-up or initialization stage rather than during process execution.

As a rule, performance overhead incurred while setting up processes are more tolerable than those that need to be done repeatedly during process execution of an HPC application. Computationally intensive applications typically run for long periods of time, and therefore the effect of additional overhead during process initialization on the run-time is nearly insignificant. It is, however, acknowledged by the author that it is not possible to accommodate all security checks in the process initialization stage.

Consider the same example of DMA-able buffers shared by multiple processes on a host. If a common buffer queue is implemented for all processes, a memory zeroing operation will be required every time a process allocates a buffer for sending network data. On the other hand, if separate regions of memory are dedicated for each process, the zeroing of memory contents needs to be done only once during process initialization, because processes will see their own residual data upon re-allocation of a buffer.

Characterizing Availability Threats

It is well known that availability cannot be easily enforced with a high level of assurance [6]. Availability threats in a HPC environment are especially difficult to detect or avoid because a legitimate request for resources cannot be easily distinguished from malicious ones.

The messaging software should impose usage limits on resources such as DMA-able memory and network buffer space, particularly on systems where the RAM is scarce. The design should minimize the user from potentially exploiting memory access priorities that govern the access to memory by the processors in the system. For example, in BDM, the user can starve the LANai processor of access to its own memory, by legitimately making repeated call to a function that polls LANai memory to check for received messages. If such a threat is likely from user processes, it will be advantageous to implement an interrupt-driven receiving mechanism, rather than having the host poll LANai memory.

Trade-off Model Requirement

Two pairs of conflicting goals for the design of a secure HPC system are clear. One is the goal to maximize performance and yet incorporate security features that inevitably downgrade performance. The second conflicting pair is specific to a design philosophy rather than the goal of HPC systems. High performance environments are designed to endow the user with as much capability as possible, in order to maximize the utilization of the raw power of the underlying hardware by user programs. There is an implicit trust in the user program to not misuse these capabilities. From a security viewpoint, user programs are not trusted and hence should be denied unmediated access to system resources.

The conflicting goals of performance and security can only be met by trading off one for another [13]. A trade-off model is required to identify the desired balance of assurance and performance. Security policies should first be identified for inclusion, based on their significance to the desired level of security. The available security mechanisms to implement all chosen policies should then be quantified based on the processing overhead they introduce. Before a design choice is made, systems should be characterized on a performance versus assurance graph for various design alternatives. The final choice of a mechanism is made based on its influence on the performance parameter(s) that is/are most significant to the system.

As an example, let us assume for the sake of argument that the confidentiality of network data is important to our system, and among the performance parameters, bandwidth is most critical. Thus we decide to implement encryption of network-bound data. Having chosen the policy of encryption, we examine the various mechanisms available to implement it and characterize them based on their associated performance overhead and their level of assurance. In this example, the choices are based on the set of available encryption algorithms, and which software layer(s) we choose to implement the encryption in. We then quantify the average processing time for encryption and decryption of data for each choice. An estimation of how often encryption or decryption is invoked by the system, when combined with its processing overhead, will help us estimate the degradation of performance parameters like bandwidth and latency. We now pick a mechanism that degrades the bandwidth approximately the least (even though it may degrade some other performance parameters more than other choices do), and is known to be most secure (among all mechanisms with comparable bandwidth degradation).

The above model assumes that performance parameters can be categorized on the basis of their significance to the system. If all performance parameters are equally important, the sum total of the quantified degradation of all these parameters is used for choosing the appropriate mechanism.

6. Conclusions

High-performance communication layers operating on System Area Networks are becoming popular in distributed computing. Security issues in distributed HPC domains are markedly different from those in conventional distributed systems. This is owed to factors such as differences in the set of services offered, characteristics of the underlying network technology, and performance requirements of a HPC system.

Most issues have been identified as owing their origins to a small number of factors: 1. The presence of a network processor at the network interface with limited built-in protection mechanisms, and 2. The design goal of high-performance distributed systems to maximize usable hardware capability that introduces unmediated resource access paths to untrusted programs. This paper identified specific security threats existing in a typical high-performance messaging layer, BDM, on a Myrinet network. Availability threats have been identified as most detrimental to any performance critical system hosting multiple user programs with shared resources.

General guidelines for the design of a secure high-performance communication layer were presented. Low level messaging layers should accommodate maximal amount of security checks at process start-up time. An understanding and characterization of availability threats is crucial.

The introduction of security checks in the communication-layer software often introduces unacceptable processing overheads, thus offsetting performance. Due to this pair of conflicting needs, this paper proposed the construction of a trade-off model to strike a balance between assurance and performance of a system.

Future directions

A host of security issues exist in the realm of high performance computing based on system area networks, as it stands today. This paper has not discussed the issues of identification, authentication and audit mechanisms. Suggestions have been made to integrate security into the messaging system. An alternate approach of dedicating one of the processors for enforcing security checks has not been discussed, but promises to be worthy of research. The study of security issues in a HPC environment with multi-level security is also a promising direction for future research.

References

[1] Abrahams, M. D., and H. J. Podell. *Information Security: An Integrated Collection of Essays, Essay 2 - Concepts and Terminology for Computer Security*, IEEE Computer Society Press, Los Alamos, CA, 1995.

[2] Abrahams, M. D., and H. J. Podell. *Information Security: An Integrated Collection of Essays, Essay 15 - Cryptography*, IEEE Computer Society Press, Los Alamos, CA, 1995.

[3] Abrahams, M. D., and H. J. Podell. *Information Security: An Integrated Collection of Essays, Essay 16 - Local Area Networks*, IEEE Computer Society Press, Los Alamos, CA, 1995.

[4] *Basic Interface for Parallelism*. http://lhpca.univ-lyon1.fr/bip.html (Accessed September, 1998).

[5] Boden, J. Nanette, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Scizovic, and W. K. Su. *A Gigabit-per-Second Local Area Network.* IEEE-Micro, 15 (1): 29-36. February 1995.

[6] Brinkly, D. L., and R. R. Schell. *Information Security: An Integrated Collection of Essays, Essay 2 - Concepts and Terminology for Computer Security*, IEEE Computer Society Press, Los Alamos, CA, 1995.

[7] Department of Defense *Trusted Computer System Evaluation Criteria.* No. CSC-STD-001-83, Department of Defense Computer Security Center, Ford Meade, MD, August 1983.

[8] Dimitrov, R. and M. Gleeson. *Challenges and New Technologies for Addressing Security in High-Performance Distributed Environments.* Proceedings of the 21th National Information Systems Security Conference, Arlington, Virginia, USA, October 1998, Vol. 2, pp:457-468.

[9] Gasser, Morrie. *Building a Secure Computer System*. New York, NY: Van Nostrand Reinhold, 1988.

[10] GM. http://www.myri.com/GM/ (Accessed February, 1999).

[11] Henley, Gregory, N. Doss, T. McMahon and A. Skjellum. 1997. *BDM: A Multiprotocol Myrinet Control Program and Host Application Programmer Interface*. Mississippi State University: Department of Computer Science. Technical Report MSU-EIRS-ERC-97-3.

[12] Kumar, Vipin, A. Grama, A. Gupta, and G. Karypis. 1994. *Introduction to Parallel Computing: Design and Analysis of Algorithms.* Benjamin/Cummins.

[13] Meadows, A. Catherine. *Tradeoffs in Secure System Development: An Outline*. Proceedings of CSESAW '94, Naval Surface Warfare Center, July 1994.

[14] Myricom. LANai 3/4 Specification. Available at http://www.myri.com:80/scs/L3/documentation.html (Accessed February, 1999).

[15] Myricom. Myrinet Users Guide. http://www.myri.com:80/scs/documentation/mug/ (Accessed February, 1999).

[16] National Computer Security Center, *Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria,* NCSC-TG-005. July 1987.

[17] Pakin, Scott, M. Lauria, A. Chien. High performance messaging on workstations: Illinois Fast Messages (FM) for Myrinet. http://www.chg.ru/SC95PROC/567_SPAK/SC95.PS (Accessed January, 1999).

[18] Pfleeger, P. Charles. *Security in Computing*. Prentice Hall, 1989.

[19] Rushby, J., and B. Randell. *A Distributed Secure System,* Computer, July 1983, Volume 16, No. 7.

[20] Sansom, R. D. May 1988. *Building a Secure Distributed Computer System.* Ph.D. diss., Carnegie Mellon University.

[21] *Scalable Coherent Interface*. ANSI/IEEE Std. 1596-1992. http://www1.cern.ch/RD24/TwoPages/TwoPages_2.html (Accessed April, 1998).

[22] Skjellum, Anthony, G. Henley, N. Doss, and T. McMahon. 1996. A guide to writing Myrinet control programs for LANai 3.x. *Tutorial Myrinet control programs.* http://www.erc.msstate.edu/labs/icdcrl/learn_mcp/smp.ps (Accessed January 26, 1997).

[23] Stallings, William. *Cryptography and Network Security: Principles and Practice.* Prentice Hall Engineering, Science & Math, 1998.

[24] Stevens, W. Richard. 1994. *TCP/IP Illustrated, Volume 1: The Protocols.* Menlo Park, CA: Addison-Wesley Publishing Company.