An Overview of MSHN: The Management System for Heterogeneous Networks

Debra A. Hensgen[†], Taylor Kidd[†], David St. John[§], Matthew C. Schnaidt[†], Howard Jay Siegel[‡], Tracy D. Braun[‡], Muthucumaru Maheswaran[¥], Shoukat Ali[‡], Jong-Kook Kim[‡], Cynthia Irvine[†], Tim Levin[§], Richard F. Freund[μ], Matt Kussow[μ], Michael Godfrey[μ], Alpay Duman[†], Paul Carff[†], Shirley Kidd[§], Viktor Prasanna[¶], Prashanth Bhat[¶], and Ammar Alhusaini[¶]

[†]Department of Computer Science
Naval Postgraduate School
Monterey, CA USA

[‡]School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN USA

[¶]Electrical and Computer Engineering
University of Southern California
Los Angeles, CA USA

[μ]NOEMIX
San Diego, CA USA

[¥]Department of Computer Science
University of Manitoba
Winnipeg, Canada

[§]Anteon Corporation
Monterey, CA USA

Abstract

The Management System for Heterogeneous Networks (MSHN) is a resource management system for use in heterogeneous environments. This paper describes the goals of MSHN, its architecture, and both completed and ongoing research experiments. MSHN's main goal is to determine the best way to support the execution of many different applications, each with its own quality of service (QoS) requirements, in a distributed, heterogeneous environment. MSHN's architecture consists of seven distributed, potentially replicated components that communicate with one another using CORBA (Common Object Request Broker Architecture). MSHN's experimental investigations include: (1) the accurate, transparent determination of the end-to-end status of resources; (2) the identification of optimization criteria and how non-determinism and the granularity of models affect the performance of various scheduling heuristics that optimize those criteria; (3) the determination of how security should be incorporated between components as well as how to account for security as a QoS attribute; and (4) the identification of problems inherent in application and system characterization.

1. Introduction

The Management System for Heterogeneous Networks (MSHN[1]) project seeks to determine an effective design for a resource management system (RMS) that can deliver, whenever possible, the required quality of service (QoS) to individual processes that are contending for the same set of distributed, heterogeneous resources. Factors influencing QoS requirements include security, user preferences for different versions of an application, and deadlines. A set of QoS requirements, considered together with resource availability, determine whether all processes' requirements can be met.

An RMS, also sometimes called a meta-computing system, is similar to a distributed operating system in that it views the set of machines that it manages as a single virtual machine [51]. Also, like any distributed operating system, it attempts to give the user a location-transparent view of the virtual machine. Hence, as in the case of a distributed operating system, an RMS provides users with improved performance while the location of resources is hidden. The set of users of a system, which consists of both local and remote resources, that is managed by an RMS should be able to attain a higher level of availability and more fault tolerance than would be available from their local system alone.

This research was supported, in part, by the DARPA/ITO Quorum Program.

[1] Pronounced, "mission"

An RMS differs from a distributed operating system in that it does not micro-manage the resources of each computer. Instead, each computer runs its native operating system. Similarly, each router executes its own protocol and each file server executes a native distributed file system. The RMS is responsible for identifying the large-grained resources, i.e., compute servers and data repositories that should be used by each process, if there is a choice. It may be responsible for issuing a command to begin execution of the processes that comprise an application. It may monitor the status of both the resources in the system and the progress of the applications for which it is responsible.

It is unclear whether every request to execute an application that is submitted to any operating system on any of the machines in the distributed system must be controlled by the RMS. If all requests are controlled by the RMS, then allocation policies that attempt to optimize throughput for a set of well-understood applications will perform better. However, sometimes users wish to maintain control over which resources their application will use.

There are many active, on-going research projects, in addition to MSHN, in the area of resource management, and there are many major research problems to be solved. A problem that MSHN is not addressing is the best way for such a system to interact with human users to obtain their QoS preferences and requirements in the most user-friendly way. Indeed, simply identifying the syntax and semantics required to express all of the QoS preferences and requirements is a difficult problem [13][17][37][55]. While MSHN does not address this problem, the designers of MSHN expect to leverage results from research in this area. They assume, for example, that a request to execute an application is accompanied by a list of deadlines, preferences for various versions of an application, security requirements, and any restrictions on the variance of the time at which a request should be completed.

Before leaving the general topic of RMSs, it is imperative that we address the topic of "packaging." MSHN researchers do not see the fruits of the RMS research as a large, monolithic piece of software that will require its own separate installation and maintenance. The best way to package the eventual outcomes of the RMS projects may be to incorporate them into an infrastructure- or middleware-level standard similar to the Common Object Request Broker Architecture (CORBA), Domain Name Services, or other such resource location services. In this way, an RMS would not need to be separately maintained and would be consolidated with the services that distributed applications will most often use. However, it is still worthwhile to separate research on RMSs from research in all other aspects of distributed object computation that will be needed in future versions of such standards in order to first isolate, then solve some of the difficult resource management problems.

1.1. Background

MSHN evolved in part from a scheduling framework called SmartNet [19][29]. SmartNet's goal was to be able to wisely schedule sets of compute-intensive jobs, some of which may require the execution of multiple processes, onto members of a suite of heterogeneous computers. SmartNet provides a sophisticated scheduling module that had been successfully integrated with many RMSs and distributed computing environments. Hence, users who need to execute compute-intensive jobs and have access to a shared, heterogeneous environment can achieve superior performance, while continuing to work in an environment to which they have grown accustomed [23]. Additionally, for those users who do not already have one installed, SmartNet provided a basic RMS that makes use of its sophisticated scheduling capabilities. SmartNet's major research contributions include:

- The ability to predict the expected run-time of a job on a machine using the concept of compute characteristics and information collected from previous executions of the job.
- The ability to leverage the heterogeneity inherent in both a collection of jobs as well as in a collection of computers.

SmartNet was used successfully by DoD and the National Institutes of Health in scheduling their compute-intensive jobs, and by NASA's EOSDIS system in determining whether their resources were adequate to process data in the ways desired by their scientists.

SmartNet's scheduling algorithms are tuned to attempt to minimize the time at which the last job completes, although the designers of SmartNet recognized that similar algorithms may be useful in optimizing other criteria. Of course, minimizing the time at which the last job, of a set of jobs, completes is, in general, an NP-complete problem, so SmartNet employs heuristics when it searches for a near-optimal mapping of jobs to machines and job execution schedule. Many of the heuristics that it uses are well known and previously documented, however, they had not previously been used in a practical heterogeneous computing system [25]. It is likely that they were not previously used in actual systems because system designers had not tried to estimate average process run-times and because it was not previously recognized that exact run-times, though helpful, were not necessary [2][3][28].

1.2. Overview of MSHN's goals

MSHN differs from SmartNet in three major ways. First, SmartNet was expected, from the beginning, to be a

system that would actually be used in production. For this reason, much of the SmartNet developers' time was spent ensuring that SmartNet was at SEI Level 3. Despite this, SmartNet was able to make significant research contributions. MSHN is intended to be a research system, facilitating experiments by the investigators to determine how RMSs, that have somewhat broader goals than SmartNet, can be built. MSHN's research goals expanded upon SmartNet's in the following areas.

(i) MSHN needs to consider that the overhead of jobs sharing resources, such as networks and file servers, can have significant impact on mapping and scheduling decisions.

(ii) MSHN must support adaptive applications (defined below).

(iii) MSHN must deliver good QoS to many different sets of simultaneous users, some of whom may be executing interactive jobs; others, compute-intensive jobs; and still others, real-time requirements.

In SmartNet's model, applications consist of three distinct phases. In the first phase, which is short compared to the second phase, they acquire data from a data repository. In the second phase, they compute results based upon the data that they obtained during the first phase. In the third phase, which is again very short compared to the second phase, they write the result back to a possibly different repository. Because the first and third phases are so short, SmartNet's heuristics assume that there is no contention for either the network or the data repositories. However, they do account for the time required to access the resources, assuming that each application is the sole user of those resources. The model of applications that MSHN is meant to manage is more complex, permitting applications to transition through many more phases of variable length, each requiring not only sharing of compute resources, but also sharing of network and data repository resources. We discuss briefly in this paper, and elaborate elsewhere, both the problem of modeling the application and that of accounting for lower level policies that govern the sharing of resources. That is, because MSHN does not assume that it has any control over network routing, file server memory allocation, etc., it models, when necessary, the lower level operating systems and protocols. By doing so, the assignment of processes to resources will account for the sharing of those resources in the correct way.

The second major difference between SmartNet and MSHN's research goals is that MSHN attempts to provide support for **adaptive** and **adaptation-aware** applications. By adaptive applications, we mean idempotent applications that can exist in several different versions. Different versions may have different values to a user due to factors such as precision of computation or input data. Additionally, different versions may have different communication and computation needs. Or, one version may execute on Windows NT while another version is an executable for Linux. MSHN's goal is to support adaptive applications by being able to terminate one version of an application if MSHN perceives that the currently executing version will not meet the users' QoS expectations.[2] In that case, MSHN would terminate the executing version and start up another version from the beginning (if there were sufficient resources to execute that other version). The requirement that adaptive applications be idempotent permits the application to be safely restarted from the beginning without corrupting any resource such as a database. Similarly, there may be times when MSHN determines that delivery of a better QoS is possible to a user by changing to a version that better meets that user's preferences.

An adaptation-aware application differs from an adaptive application in two ways. First, when it is terminated, the new version need not be restarted from the beginning. Instead, a different version from the one that terminated may be started, using information about a previous state that was obtained from the execution of the previous version. Second, an adaptation-aware application may be able to adapt its resource usage during execution, without restarting.

Finally, MSHN's goals differ from SmartNet's in that MSHN seeks to determine how to meet multiple different QoS requirements to multiple different applications simultaneously. There are really two issues bound up in this difference. First, a way to incorporate, dynamically, the mixture of QoS requirements into a single measure must be determined. Second, an assignment of applications to resources must also be determined that optimizes the identified measure. In resolving this second issue, we can strongly leverage SmartNet's emphasis on the separation of optimization criteria and search algorithms and the recognition that similar algorithms can be used to search many different types of spaces for optimal values. We elaborate on this below.

1.3. Related work

There are other research groups examining the issues important to building an RMS, many within DARPA's Quorum project. Here, we look at some of the projects related to MSHN. Some of these groups are engaged in research complementary to MSHN's goals. For the sake of brevity, only a short synopsis of each project, as it relates to MSHN, is presented.

DeSiDeRaTa. The University of Texas at Arlington has a project called "DeSiDeRaTa: QoS Management Tools for Dynamic, Scalable, Dependable, Real-Time

[2] We note that a version of one application may be terminated because MSHN detects that another user's application will not meet its QoS expectations. This phenomenon can occur due to priorities.

Systems." DeSiDeRaTa is focusing on QoS specification, QoS metrics, dynamic QoS management, and benchmarking of specific computing environments, such as the distributed Anti-Air-Warfare system at the Naval Surface Warfare Center, Dahgren Division. A unique concept that has come out of the DeSiDeRaTa project is that of an application "path" [56].

Globus. Globus is a large, joint project from Argonne National Laboratory and the University of Southern California's Information Sciences Institute. Parts of the Globus project are devoted toward resource management issues. The Globus architecture depends on an advance or immediate resource reservation protocol layer, for which a standard does not yet exist [14][18].

RT-ARM. Honeywell is developing a "Real-Time Adaptive Resource Management" system aimed primarily at high-end, real-time military embedded systems such as the Navy Surface Combatant Ship SC-21. Some of the specific issues they are concentrating on include modeling embedded systems and finding practical techniques for predictable real-time performance [24].

EPIQ. The EPIQ project, from the University of Illinois at Urbana-Champaign, is building an infrastructure for providing guaranteed QoS features, upon which RMSs may be built. part of their infrastructure involves building their own runtime environment [35].

ERDoS. SRI International is running a project called ERDoS (End to End Resource Management for Distributed Systems) which is developing an architecture for adaptive QoS-driven resource management. The ERDoS project emphasizes a comprehensive definition of QoS and the development of models that capture information required for making resource management decisions [46].

QUASAR. The QUASAR (QUAlity Specification and Adaptive Resource management for distributed systems) project, at the Oregon Graduate Institute of Science and Technology, is investigating techniques for specifying and utilizing QoS in adaptive, distributed systems. QUASAR is concentrating on the translation of QoS specifications from the application-level to the resource-management-level, and its use in reservation-based resource management, primarily in the multimedia domain [53].

ASSERT. The ASSERT System at the University of Oregon, Eugene, is focusing on dynamic, distributed, real-time environments. The core of the project estimates and monitors the relevant QoS parameters of running applications. ASSERT is not an RMS, nor an RMS framework; rather, the ASSERT project is looking at a specific issue of RMSs: QoS monitoring and estimation [16].

QuO. The Quality Objects (QuO) project, from BBN Systems Technologies, is attempting to add QoS specification and delivery to CORBA. Rather than provide absolute QoS guarantees, QuO seeks to combine knowledge about resource and application conditions in order to reserve enough end-to-end resources for predictable execution of distributed applications [52].

MOL. The MOL (Metacomputing OnLine) project from the Paderborn Center for Parallel Computing has as a goal the utilization of multiple high performance systems for solving problems too large for a single supercomputer. The MOL approach does not assume absolute control of resources under its management. The MOL project is addressing several of the issues key to resource management, including QoS specification [42].

1.4. Organization of the paper

In the next section of the paper we motivate and discuss MSHN's architecture. Even though SmartNet was successful in achieving its functionality, rather than using SmartNet's architecture exactly, we based MSHN's architecture upon lessons learned from SmartNet, because MSHN's goals are substantially different. In particular, we clearly delineated certain of SmartNet's modules into separate components. This delineation makes it easier to experiment with different designs for each of the components. In section 3, we then discuss many of the research issues that the MSHN investigators are studying and highlight some of the results. Additionally, this section provides references to the numerous articles that describe this research in more detail. We conclude by summarizing the status of the MSHN project.

2. MSHN's architecture

In this section, we first describe some of the concepts that went into MSHN's architectural design. This description motivates the need for the various major components and explains why they must be replicated to varying degrees. The architectural design was driven by the need to support the RMS research that we will discuss in the next section and was aided by our previous experience with SmartNet. We then present MSHN's current architecture in detail.

2.1. Motivation

We first motivate the need for each of the major components of MSHN's architecture, then discuss how those components interact with one another.

We recall from the previous section that an RMS needs to transparently locate the resources that should be used when execution of an application is requested. Therefore, it must be made aware of any request, by either a user or an application, to start executing another application. Many early RMSs required the user to explicitly log in to the system to start a job. If an application was to be started from within another application, e.g., through

`fork` and `exec` system calls, then the application that makes the request would be required to be specially designed to embed these requests within a function call to an RMS library. This restriction required that applications be specifically written or modified for a particular RMS.

The MSHN designers do not want to force a user to explicitly log into an RMS, or to modify their existing programs. Instead, MSHN transparently intercepts calls to system libraries that would otherwise initiate execution of a new process and diverts those calls to a MSHN Client Library. After MSHN decides where the newly requested application should execute, the MSHN Client Library uses whatever mechanisms available at the resource site to initiate execution of the remote process.

The environments for which MSHN is designed contain many different types of computers, each possibly executing a different version of an operating system. Rather than requiring the Client Library, which is linked with every MSHN application, to contain a substantial amount of code that is specific to each of these computers, we chose to make use of a MSHN Daemon. Whenever a computer is added to a system, a MSHN Daemon is started on that computer. When a Client Library needs to start a process on a remote machine, it simply contacts the MSHN Daemon on that machine and requests that the Daemon start the process on the Client Library's behalf. Of course, the general mechanism that we use in the Daemon is not new, and is therefore not a research issue.

When a remote process needs to communicate with the initiating process, it contacts the Client Library, which passes the information on to the initiating process, just as though the remote process were started locally. Being able to transparently provide this service to applications, whether or not they are command interpreters, requires that the Client Library intercept, and at least pre-process if not divert, other system library calls in addition to the previously mentioned `exec` call. For example, all of the socket calls and all calls to open, close, read, and write files must be intercepted and replaced or at least pre- and post-processed.

The MSHN project required a mechanism for intercepting these calls without requiring source modification. We initially turned to the Condor project for help with this problem [36]. Condor is a project at the University of Wisconsin that performs transparent migration of processes in a Unix environment. To perform this migration, Condor also had to intercept these calls to system libraries. Using techniques similar to those used by Condor, we were able to intercept these calls without requiring source code modification.[3] The mechanism is described in detail elsewhere [44].

In addition to providing a mechanism for transparently executing remote processes, the Client Library is in a unique position to passively determine the status of resources, because it is assumed to be linked with any application executing in an environment managed by MSHN. That is, the MSHN Client Library can pre- and post-process system calls, because it is intercepting all such calls made to the operating system, which are executed when a process needs to use a hardware resource. In so doing, it can determine the low level, end-to-end QoS that an application is receiving from a particular resource. We will discuss this functionality of the Client Library further in the next section.

When the MSHN Client Library intercepts a call to execute a new process, it must have some way of determining which resources that new process should use, i.e., which computer should primarily be responsible for executing the new process.[4] Rather than requiring that decision to be made independently by each Client Library that is linked with each application, we chose to have the Client Library first check the request against a list of applications managed by MSHN. If the requested application is not on that list, the MSHN Client Library simply passes the requested application directly to the local operating system. If the requested application is on that list, it instead passes the request to the MSHN Scheduling Advisor. It is the Scheduling Advisor's job to determine which set of resources the newly requested process should use.

The MSHN Scheduling Advisor is itself a complex package, associated with many different research issues which we discuss more fully in the next section. Among the primary research issues are: (i) what criteria should be optimized in the choice of resources? (ii) Because optimizing the criteria is likely to be an NP-complete problem, if *n* is too large, which heuristic should be used to search for an optimum resource assignment? (iii) With what granularity must the Scheduling Advisor model both the policies and protocols associated with allocation of the lower level resources and what granularity of model should it use to define the resource requirements of a process?

For the Scheduling Advisor to determine a good assignment of resources for a process, it must know both which resources and how much of each resource would be required for a process to execute and meet its QoS requirements and preferences. Therefore, to assist the Scheduling Advisor in making its decision as to the

[3] These techniques, however, require that the object code files be linked with the MSHN Client Library, therefore they require object code files. However, another tool, the Executable Editing Library (EEL) which evolved from the University of Wisconsin's Paradyn project could be used to link an executable with the MSHN Client Library, instead [32].

[4] In modern systems, the choice of computer that is responsible for executing a process often carries with it, implicitly, a choice of file servers and other distributed resources such as networks. Therefore, when we say that MSHN chooses a computer to be responsible for executing a process, the choice of other resources external to that computer may be implicit in that assignment.

assignment of resources, we designed both the MSHN Resource Requirements Database and the MSHN Resource Status Server.

The Resource Status Server is a quickly changing repository that maintains information concerning the current availability of resources. Information is stored in the Resource Status Server as a result of updates from both the MSHN Client Library and the MSHN Scheduling Advisor. The Client Library can update the Resource Status Server as to the currently perceived status of resources, which takes into account resource loads due to processes other than those managed by MSHN. The Scheduling Advisor can provide expected future resource status based upon the resources that it expects will be used by the applications that it assigns. Additionally, the Resource Status Server can statistically process its historic knowledge to make predictions of resource status even further in the future.

As compared to the Resource Status Server, the information maintained by the MSHN Resource Requirements Database changes much more slowly. The Resource Requirements Database is responsible for maintaining information about the resources that are required to execute a particular application. Although the initial MSHN prototype only implements a single source for the information stored in this database (statistically analyzed historical information), we envision that many other on-going research projects will also serve as sources for this information.

MSHN's current source for the information that is maintained by the Resource Requirements Database comes from data collected by the MSHN Client Library when the application was previously executed. Although patterned after SmartNet in this way, and leveraging the concept of compute characteristics that SmartNet pioneered, MSHN does not collect the same information as SmartNet collects. SmartNet's information is coarse-grained; that is, it maintains only the total amount of wall-clock time that is required to execute a program from beginning to end for each particular machine. This measure is sufficient for SmartNet's needs due to the requirements of its intended applications (three phases) and the expected environment (each job has exclusive access to the resources that it is using). However, in MSHN, resources are shared and applications have more phases, so maintaining only this coarse grain information is insufficient. Therefore, the Resource Requirements Database has the ability to maintain very fine grain information collected by the MSHN Client Library. Eventually it is hoped that the Resource Requirements Database can also be populated with information from smart compilers and possibly advice from application writers.

Applications, of course, are needed to test any system. Unfortunately, executables for many different platforms would be needed to test MSHN's ability to manage them in a distributed, heterogeneous environment. Producing such actual applications would require tremendous effort to obtain the source code for numerous applications, some of which may be classified or proprietary, port the source code to the different platforms, and compile and link them. We decided that this effort was better spent on our research system itself, so we looked for another viable solution. One solution that we considered was to use benchmarks, because many of them have already been ported to many different platforms. However, we wanted to make sure that our system could manage a wide variety of applications. We finally settled on writing a general-purpose application emulator whose parameters could be specified to cause it to imitate a wide variety of applications. We discuss the problem of deciding how best to construct such an emulator under the research topics in the next section.

The Client Library, which is linked with each executing MSHN application, informs the Resource Status Server about the current perceived status of the resources that the applications are using. The Scheduling Advisor informs the Resource Status Server only about the load that it expects the processes, which it has scheduled, to place on certain resources. However, neither class of information indicates the condition of resources that no MSHN application is currently using or is planning on using. Therefore, we use a MSHN Application Emulator linked with the Client Library to obtain information about the condition of such resources.

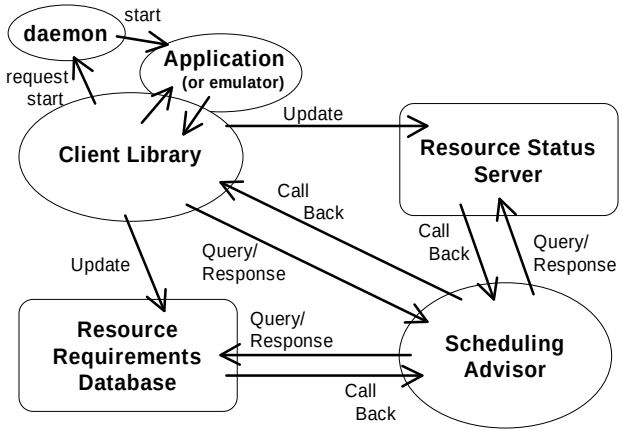


Figure 1 MSHN's conceptual architecture.

MSHN's conceptual architecture is shown in Figure 1. As can be seen in the figure, every application running with MSHN makes use of the MSHN Client Library that intercepts the application's operating system calls. When the Client Library intercepts a request to execute a new application, and that application requires that the MSHN Scheduling Advisor be consulted to determine the resources that the application should use, the Client

Library invokes a scheduling request on the Scheduling Advisor. The Scheduling Advisor queries both the Resource Requirements Database and the Resource Status Server. It uses information that it receives from them, along with an appropriate search heuristic, to determine where the newly requested process should execute. After determining which resources should host the new process, the Scheduling Advisor returns the decision to the Client Library, which, in turn, requests execution of that process through the appropriate MSHN Daemon. The MSHN Daemon invokes the application on its machine. As a process executes, the Client Library updates both the Resource Status Server and the Resource Requirements Database with the current status of the resources and the requirements of the process. Meanwhile, the Scheduling Advisor establishes callbacks with both the Resource Requirements Database and the Resource Status Server. Using callbacks, the Scheduling Advisor is notified in the event that either the status of the resources has significantly changed, or the actual resource requirements are substantially different from what was initially returned from the Resource Requirements Database. In either case, if it no longer appears that the assigned resources can deliver the required QoS, the application must be adapted or terminated. Upon receipt of a callback, the Scheduling Advisor might require that several of the applications adapt so that more of them can receive their requested or desired QoS.

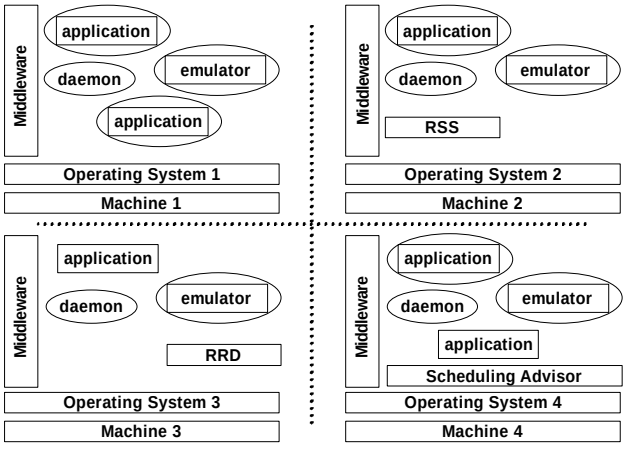


Figure 2 Physical instantiation of the MSHN architecture.

Although all MSHN components could run on the same machine, they can also be distributed and replicated across many different computers using tools such as ISIS, Horus and Ensemble [7][50][49]. Results from control theory will also be useful here in ensuring that the process of replicating and merging components is stable and does not result in oscillation. Additionally, results from control theory must be incorporated into the replicated Scheduling Advisor itself to ensure that modifications requested of adaptive and adaptation-aware applications do not become unstable. MSHN components might even replicate as needed [20][21]. **Figure 2** illustrates a simple instantiation of the MSHN system.

In addition to the components discussed above, we found it convenient to add a MSHN Visualizer that enabled us to examine, for both functional and performance debugging purposes, the current states of the various MSHN components. The MSHN Visualizer captures all significant events within and between the core MSHN components for real-time and post-mortem analysis.

Security within the MSHN architecture has been considered. Policies of interest are:

- Component authentication. This includes authentication of MSHN core components to each other; authentication of resource-based clients to the MSHN core; and authentication of applications to selected MSHN components.
- Hierarchical least privilege. Within the MSHN context, the core components are the most privileged, while user applications are the least privileged.
- Communications integrity and confidentiality. Communications are protected from unauthorized modification and disclosure.
- Access control. Access to MSHN core databases and to job histories may be mediated.

The security architecture creates keyed domains, supporting least privilege, authentication, confidentiality and integrity by using the Common Data Security Architecture facilities for security services and key management[5] [57][58].

2.2.1. The current MSHN architecture. A high level description of the current MSHN architecture is presented. For a more detailed description, we refer the reader to other publications [43]. High-level diagrams are presented for each MSHN component, with arrows indicating the direction of communication or action. In addition to these diagrams, a short description of each component's functions is given. In the description of the MSHN architecture, we represent MSHN components and external components as Unified Modeling Language (UML) actors [8]. The symbols used for this representation are shown in Figure 3. The core MSHN components include the Scheduling Advisor (SA), the Client Library (CL), the Resource Status Server (RSS), the Resource Requirements Database (RRD), the Daemon (D) and the Application Emulator (AE).

[5]As in any RMS, assurance of MSHN's security properties is built on and limited by the effectiveness of the security environment provided by the underlying operating system(s) and hardware base(s).

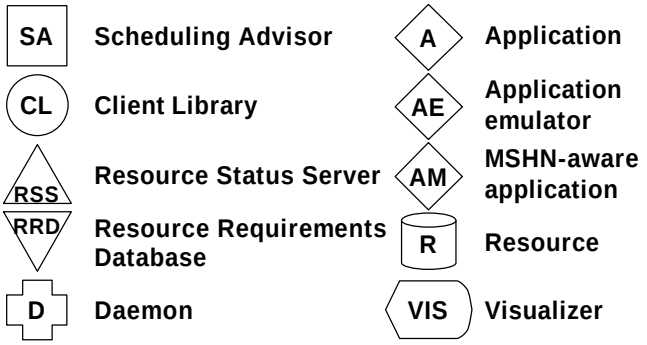
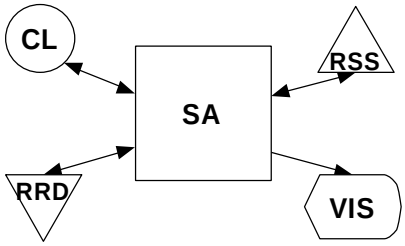
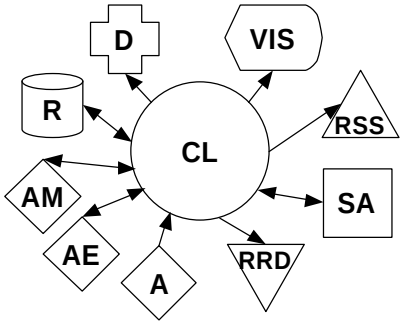


Figure 3 Symbols representing actors in the MSHN architecture.

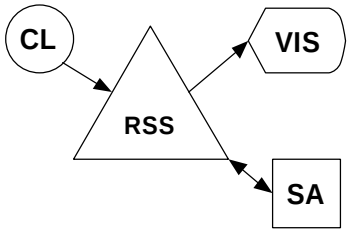
Scheduling Advisor (SA) functionality. The primary responsibility of the SA is to determine the best assignment of resources to a set of applications, based on the optimization of a global measure, which we describe in the next section. The SA depends on the RRD and the RSS in order to identify an operating point that optimizes the global measure. It responds to resource assignment requests from the CL. When appropriate, the SA requests application adaptations via the CL. The SA is also responsible for establishing callback criteria (thresholds) with the RSS and RRD. All MSHN components update the MSHN Visualizer with all significant display and post-mortem analysis events.



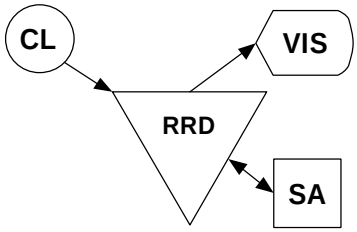
Client Library (CL) functionality. The CL is linked with both adaptive and adaptation-aware applications. It provides a transparent interface to all of the other MSHN components. The CL intercepts system calls to collect resource usage and status information, which is forwarded to the RRD and the RSS. The CL also intercepts calls that initiate new processes (such as `exec()`) and consults the SA for the best place to start that process. It requests (possibly remote) daemons to execute applications based on the SA's advice. The CL invokes adaptation on adaptation-aware applications when notified by the SA via callbacks. One such invocation is the special case of setting emulator parameters.



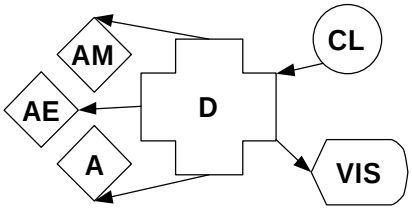
Resource Status Server (RSS) functionality. The role of the RSS is to maintain a repository of the three types of information about the resources available to MSHN: relatively static (long-term), moderately dynamic (medium-term), and highly dynamic (long-term) information. The RSS is updated with current data via the CL or through a system administrator. The RSS responds to SA requests with estimates of currently available resources. The SA sets up callbacks with the RSS based on resource availability thresholds and CL update frequency requirements.



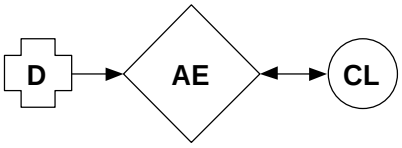
Resource Requirements Database (RRD) functionality. The RRD is a repository of information pertaining to the resource usage of applications. The RRD provides this information to the SA. Callbacks to the SA are based on either the occurrence of a threshold violation or update frequency requirements. It is updated by the CL.



Daemon (D) functionality. The MSHN Daemon executes on all compute resources available for use by the SA. Its sole purpose is to start applications as requested by the CL. It therefore has the capability and responsibility of initiating the default application emulator at start-up to determine resource status information.



Application Emulator (AE) functionality. The AE emulates a running application by stressing particular resources in the same way as the real application does. The AE serves two purposes: The first is to run simulated applications (that statistically leave the same resource usage footprint of the real applications) without the overhead and uncertainty of actually installing, maintaining, and running that particular application. The second is to be a monitor, in the absence of any other MSHN-scheduled applications. That is, it can determine the status of resources that are not being otherwise used by MSHN-scheduled applications, and therefore not being monitored by an existing CL. The Daemon starts one instance of the AE, by default, at startup. Other instances may be started at any other time through a command interpreter or other application.



3. MSHN Research Issues

In this section of the paper, we describe some of the major issues being investigated by the MSHN team members. We also briefly summarize some of the results to date. Of course, there is not sufficient space to completely describe all of the issues and results in detail, so the reader is also referred to relevant papers on each topic. We have attempted to associate the issues with the component of the MSHN architecture that they most strongly affect. However, certainly many issues that affect the Scheduling Advisor also affect the Resource Status Server and Resource Requirements Database. Additionally, this work is non-orthogonal to research being done by many investigators outside of the MSHN team who are examining such issues as how QoS requirements are derived from smart compilers and how they can be best expressed.

3.1. Scheduling Advisor research issues

In this section we discuss some issues that most strongly affect the Scheduling Advisor. First, we examine how to quantify the needs of all of the processes that require resource allocation by the Scheduling Advisor. Then, we consider the ramifications of not precisely knowing the resource requirements, and consequently, the exact future status of all of the resources. Finally, we discuss the class of heuristics that have thus far been implemented in MSHN and why there is a need for a variety of heuristics.

3.1.1. Optimization criteria. Optimal resource allocation always involves attempting to solve an optimization problem, which is usually NP-complete. SmartNet's primary optimization criterion was to minimize the time at which an application completes, assuming that all of the applications were of a particular form. Later versions of SmartNet also accounted for priorities. MSHN maximizes a weighted sum of values that represents the benefits and costs of delivering the required and desired QoS (including security, priorities, and preferences for versions), within the specified deadlines, if any. We now discuss the effect of each of these attributes on the optimization criteria.

- MSHN's consideration of security as an optimization criterion allows the trade-off of security with other QoS constraints when there are insufficient resources to complete all requests. This is done in a fashion similar to other recent projects [45]. MSHN associates a cost to security levels that varies, depending upon which resources are being used to obtain a given level of security (for more details on security viewed as a QoS parameter, see section 3.2).
- MSHN attempts to account for both preferences for various versions and priorities. That is, when it is impossible to deliver all of the most preferred information within the specified deadlines due to insufficient resources, MSHN's optimization criteria are designed to favor delivering the most preferred version to the highest priority applications.
- In MSHN's optimization criteria, deadlines can be simple or complex. That is, sometimes a user will be satisfied if a result is received before a specific time. Other times, a user would like to associate a more general benefit function, which would indicate that information might have different values based upon when it is received.

Further information about MSHN's optimization criteria can be found elsewhere [22][30].

In addition to a cost function that is optimized, optimization problems usually have a set of constraints that must be met in order for a solution to be viable. The constraints of a resource allocation optimization problem are that the resources allocated to meet the needs of the processes must be less than or equal to the available resources at any point in time. The actual inequalities required not only depend upon the QoS constraints, but

also upon the sharing policies used by the local operating systems and network protocols, and upon the granularity with which both those policies and resource usage should be known (see Granularity Issues in Section 3.2).

3.1.2. Inexact knowledge of job resource usage. Even if it is possible to find a perfect solution to the optimization problem that is posed by instantiating the constraints and optimization criteria to the current situation, the expected resource usage of any given application is often only an estimate. In real-time systems, the worst case estimate is often used to assign resources to processes; however, many other systems use the mean expected resource usage. Our recent analysis has revealed that using the mean will cause the actual run-time to be generally underestimated and that a better assignment can be made if both the mean and distribution of the expected resource usage is accounted for, when appropriate [28].

This leads to another question concerning whether the extra complexity involved in using a sophisticated heuristic will yield a better schedule than using a simple heuristic if the actual variance of run-times is large, and scheduling is done using the mean, or both the mean and the distribution. Our recent results in this area have shown that, in many cases, complex heuristics can determine schedules that, when executed, sometimes perform much better than the schedules derived from very simple heuristics, even when the variance is large. However, sometimes very simple heuristics perform just as well as the more complex ones. The difference in quality of the schedules produced by the various heuristics was found to be closely correlated with the type of heterogeneity in a system. For example, when both the machine and application heterogeneity is very low, a simple heuristic performs just as well as more complex ones. Several papers have described our results concerning this research [2][3][10][40].

3.1.3. Performance of search algorithms. SmartNet's organization leveraged the idea of independence of search algorithms and optimization criteria. That is, most heuristics for searching the space of mappings can be modified to search for solutions to different optimizations within the same space. For example, Dantzig's Simplex Method is useful with all problems whose optimization criteria and constraint inequalities can be stated using only linear combinations of the variables. Sometimes, many different heuristics will work, but, depending upon the characteristics of a given problem, certain heuristics may be preferable to others. For example, the MSHN team has obtained extensive results identifying the regions of heterogeneity where certain heuristics perform better than others for maximizing throughput by minimizing the time at which the last application, of a set of applications, should complete [2][3][10][40]. Re-targeting of these heuristics to other optimization criteria is currently underway.

Additionally, MSHN team members have performed extensive research into accounting for dependencies between applications or processes that make up a single application [40][47][48][54]. This includes promising results from investigating data dependencies and mapping of iterative applications [1][4][5][6][11].

3.2. Resource Status Server and Resource Requirements Database research issues

Part of the MSHN team's investigation has been aimed at determining what information should be stored in the Resource Requirements Database and maintained by the Resource Status Server. First, a taxonomy for the types of information that could be stored there was required. We discuss this taxonomy below. We also discuss the impact that viewing security as a QoS has on these two MSHN components. Finally, one of the most important issues in designing effective RMSs is determining the level of granularity of information that must be maintained concerning the status of resources and the requirements of applications. We now discuss each of these issues in somewhat more detail and refer the interested reader to relevant publications.

3.2.1. A taxonomy. The MSHN team has formulated a three-part taxonomy for classifying systems. The three different components include methods for describing the applications, the computing environment, and the mapping strategy that is used. Some of the relevant characteristics that need to be instantiated concerning each application include

(i) Its size, that is the number of tasks or sub-tasks associated with it.

(ii) Whether the sub-tasks are independent of one another or, if they are dependent, the types of dependencies.

(iii) The I/O distributions of the application and the sources of the I/O, i.e., whether it performs all input in the beginning and all output at the end or whether one or the other is performed continually throughout the lifetime of the processes and whether the input data is obtained through interacting with a person or some other source that has highly variable response times.

(iv) The deadlines and other QoS requirements, including security, if any, associated with the applications and/or the subtasks that comprise the application.

Similarly, the computing environments and mapping strategies have numerous, hierarchically characterizable, attributes that are more fully documented in other publications [9].

3.2.2. Security as a quality of service. Security in the context of QoS is a current research area [34][45]. The security capabilities of resources and security requirements of applications must influence the assignment of applications to resources. We can obtain information concerning the user security requirements from the Resource Requirements Database and information concerning the security capabilities of the resource from the Resource Status Server. For example, if the output of an application must be encrypted using a particular algorithm, with a key size chosen within a particular range, then that requirement must be stored in the Resource Requirements Database along with the amount of data that must be encrypted. Also, the Resource Status Server must know whether each particular computing resource is capable of performing the required cryptographic algorithm and the cost, in terms of run-time per byte, for example, of encrypting the data. Members of the MSHN team have developed an initial framework, which they are currently refining, for characterizing the overall security attributes of a network and for determining a cost and benefit value for providing required and preferred security to an application [26][27][33][34].

3.2.3. Granularity issues. Another very important question that concerns both the Resource Requirements Database and the Resource Status Server has to do with how much detail should be maintained concerning the status of resources and the requirements of applications. Obviously, while a very accurate, detailed set of information might prove quite useful to the scheduling algorithms, it would be at the least very expensive and difficult to collect if not expensive to process within the algorithm itself.

The MSHN team has obtained initial estimates for the overhead of capturing system calls to determine the cost of collecting various granularities of such information [44]. Members of the team are currently using this technique to record fine-grained information for a program that analyzes air tasking orders and will report both the information concerning the resources that were used, as well as the overhead involved in collecting the resource usage information [41].

In addition to the cost associated with collecting fine-grained information concerning applications' use of resources, there is the question of how much information is sufficient. Current experiments of the MSHN team focus on determining whether fairly simple models can be used to predict the relative performance of application/resource assignments. To perform realistic experiments, the team has built an initial application emulator (see below) and is actually executing it with different parameters on different systems, using all possible configurations to compare the actual received QoS to the predicted QoS. Thus far we have determined that the Resource Status Server must, directly or indirectly, contain information concerning whether native threads are supported by the operating system. If this information is not maintained, the scheduling algorithm, which must choose between two platforms that are identical except for the operating system version that they execute, may assign a process which could be handled better by one platform to the other. Similarly, the Resource Requirements Database must indicate whether or not the application is multi-threaded and the number and nature of threads that it uses. Information concerning these results can be found in other publications [12].

3.3. Application Emulator research issues

The MSHN team is designing and implementing an application emulator for two different reasons. One reason is that it is needed within the MSHN architecture to monitor the end-to-end status of the resources. The other reason is to be able to easily construct a very large suite of application emulators that place loads on resources in the same way that the actual applications would. When used in conjunction with resource usage measurements from linking actual applications to MSHN's Client Library, the MSHN Application Emulator can be used to emulate the execution of the actual applications without requiring the applications to actually be ported to many different platforms. The obvious advantage of using such an application emulator, rather than porting the applications themselves, is to enable the MSHN researchers to test their architecture more quickly under many different situations.

To meet the first purpose of the MSHN Application Emulator, we first had to define the meaning of loading resources for various resources. Percentages cannot be used, as they are not transferable between either computing platforms or network media. Rather, each category of resource was identified and units that can be most easily translated between different platforms, such as FLOPS and bytes/sec, were chosen to quantify resource use. Also recognized at this stage was the need to have both multi-threaded and non-multi-threaded application emulator capability. Finally, not only can a single application be comprised of multiple threads, but it can also be comprised of multiple heavy-weight processes.

When designing the Application Emulator to meet both of its requirements, we recognized that distributions reflecting communication and computation alone were insufficient; conditional probabilities were required. That is, many times the purpose of one process sending a message to another process is so that the receiving process will perform work on behalf of the sending process. Therefore, we designed our most general emulator to also have the capability of sending work-bearing messages.

To this end, we have completed an initial implementation of an application emulator that we have used for our granularity research and are testing the more general application emulator. Documentation concerning both of these application emulators can be found elsewhere [12] [15].

3.4. Client Library research issues

The research issues having to do with the Client Library component involve both mechanism and policy. The mechanism issues have to do with how to transparently link the Client Library with applications. Previous research in the areas of process migration and tools for debugging parallel and distributed programs provide us with easy solutions, as mentioned earlier. Therefore, the only issue that remains is how best to transparently determine the end-to-end availability of resources. First, simply determining that the Client Library could perform this functionality better than providing the functionality external to the applications themselves is an important contribution. However, determining the average end-to-end availability of a network resource is not a trivial problem. The MSHN team's initial progress in this area has already been detailed elsewhere [30][31][44].

4. Summary and future work

In this paper we summarized the purpose of a resource management system (RMS) in general and the research goals of one particular experimental RMS, the Management System for Heterogeneous Networks (MSHN). Motivation was provided for all of the major components of MSHN, and the architecture that contains those components was explained. Some of the research questions that the MSHN researchers are seeking answers to were described. References were provided that enable the reader to better understand MSHN, and to learn more about the MSHN experiments. There are many other interesting RMS research projects in progress today, but space permitted us to survey only a few of them. In addition to continuing the on-going experiments described in the paper, future MSHN investigation will focus on (i) reaching a better understanding of the level of granularity obtainable from applications and the level required to perform sufficiently good resource assignment; (ii) more detailed characterization of security costing and metrics; and (iii) determining the best search algorithms to use for the MSHN optimization criteria under various conditions.

Acknowledgments – The authors thank Shushanna St. John and LCDR Wayne Porter for their comments.

References

[1] A. H. Alhusaini, V. K. Prasanna, and C. S. Raghavendra, "A unified resource scheduling framework for heterogeneous computing environments," *Proc. 8th IEEE Heterogeneous Computing Workshop*, April 1999.

[2] R. Armstrong, *Investigation of Effect of Different Run-Time Distributions on SmartNet Performance*, Thesis, Department of Computer Science, Naval Postgraduate School, Monterey, CA, Sept. 1997.

[3] R. Armstrong, D. Hensgen, and T. Kidd, "The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions," *Proc. 7th IEEE Heterogeneous Computing Workshop*, March 1998, pp. 79-87.

[4] P. B. Bhat, V. K. Prasanna, and C.S. Raghavendra, "Adaptive communication algorithms for distributed heterogeneous systems," *Proc. IEEE Intl. Symp. High Performance Distributed Computing*, July 1998, pp. 310-321.

[5] P. B. Bhat, V. K. Prasanna, and C.S. Raghavendra, "Block-cyclic redistribution over heterogeneous networks," *Proc. ISCA Intl. Conf. Parallel and Distributed Computing Systems*, Sept. 1998, pp. 242-249.

[6] P. B. Bhat, V. K. Prasanna, and C.S. Raghavendra, "Efficient collective communication in distributed heterogeneous systems," *Proc. IEEE Intl. Conf. Distributed Computing Systems*, 1999, to appear.

[7] K. Birman, "Replication and fault-tolerance in the ISIS system," *10th ACM Symposium on Operating Systems Principles*, Dec. 1985, pp. 79-86.

[8] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*, Addison Wesley, Reading, MA, 1999.

[9] T. D. Braun, H. J. Siegel, N. Beck, L. L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, and B. Yao, "A taxonomy for describing matching and scheduling heuristics for mixed-machine heterogeneous computing systems," *Proc. IEEE Workshop on Advances in Parallel and Distributed Systems*, October 1998, pp. 330-335 (included in the proceedings of the *7th IEEE Symposium on Reliable Distributed Systems*, 1998).

[10] T. D. Braun, H. J. Siegel, N. Beck, L. L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, and R. F. Freund, "A comparison study of static mapping heuristics for a class of meta-tasks on heterogeneous computing systems," *Proc. 8th IEEE Heterogeneous Computing Workshop*, April 1999, to appear.

[11] J. R. Budenske, R. S. Ramanujan, and H. J. Siegel, "A method for the on-line use of off-line derived remappings of iterative automatic target recognition tasks onto a particular class of heterogeneous parallel platforms," *The Journal of Supercomputing*, Vol. 12, No. 4, Oct. 1998, pp. 387-406.

[12] P. Carff, *Granularity*, Thesis, Department of Computer Science, Naval Postgraduate School, Monterey, CA, March 1999.

[13] P. Chandra, A. Fisher, C. Kosak, T. S. E. Ng, P. Steenkiste, E. Takahashi, and H. Zhang, "Darwin: Resource Management for Value-Added Customizable Network

Service," *Proc. 6th IEEE International Conference on Network Protocols*, October 1998, pp. 177-188.

[14] K. Czajkowski, I. Foster, C. Kesselman, N. Karonis, S. Martin, W. Smith, and S. Tuecke, "A resource management architecture for metacomputing systems," *Proc. Workshop on Job Scheduling Strategies for Parallel Processing*, 1998, pp. 62-82.

[15] T. Drake, *A Load Emulator Toolkit and Analysis of HiPer-D Resource Requirements*, Thesis, Department of Computer Science, Naval Postgraduate School, Monterey, CA, June 1999.

[16] S. Fickas, and M. S. Feather, "Requirements Monitoring in Dynamic Environments," *Proc. 2nd IEEE Intl. Symposium on Requirements Engineerings*, March 1995, pp. 140-147.

[17] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke, "A Directory Service for Configuring High-Performance Distributed Computations," *Proc. 6th IEEE Symp. on High-Performance Distributed Computing*, 1997, pp. 365-375.

[18] I. Foster, and C. Kesselman, "The Globus project: a status report, " *Proc. 7th IEEE Heterogreneous Computing Workshop*, 1998, pp. 4-18.

[19] R. F. Freund, M. Gherrity, S. Ambrosius, M. Campbell, M. Halderman, D. Hensgen, E. Kieth, T. Kidd, M. Kussow, J. D. Lima, F. Mirabile, L. Moore, B. Rust, and H. J. Siegel, "Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet," *Proc. 7th IEEE Heterogeneous Computing Workshop*, March 1998, pp. 184-199.

[20] D. Hensgen, *Squads: Server Groups that Dynamically Adapt to Improve Performance*, Ph. D. Dissertation, Department of Computer Science, University of Kentucky, 1989.

[21] D. Hensgen, and R. Finkel, "Dynamic server squads in Yackos," *Proc. Workshop on Experiences with Building Distributed and Multiprocessor Systems*, Oct. 1989, pp. 73-90.

[22] D. Hensgen, T. Kidd, H. J. Siegel, J. K. Kim, D. St. John, C. Irvine, T. Levin, V. Prasanna, and R. Freund, *A performance measure for distributed heterogeneous networks based on priorities, deadlines, versions, and security*, Technical Report, School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, Feb. 1999.

[23] D. Hensgen, L. Moore, T. Kidd, R. F. Freund, E. Keith, M. Kussow, J. Lima, and M. Campbell, "Adding rescheduling to and integrating Condor with SmartNet," *Proc. 4th IEEE Heterogeneous Computing Workshop*, April 1995, pp. 4-11.

[24] J. Huang, R. Jha, W. Heimerdinger, M. Muhammad, S. Lauzac, B. Kannikeswaran, K. Schwan, W. Zhao and R. Bettati, "RT-ARM: a real-time adaptive resource management system for distributed mission-critical applications", *Workshop on Middleware for Distributed Real-Time Systems*, 1997, pp. 179-186.

[25] O. Ibarra and Kim, "Heuristic algorithms for scheduling independent tasks on non-identical processors," *Journal of the ACM*, Vol. 24, No. 2, 1977, pp. 280-289.

[26] C. Irvine and T. Levin, *A note on mapping user-oriented security policies to complex mechanisms and services*, Technical Report, Department of Computer Science, Naval Postgraduate School, Monterey, CA, in progress.

[27] C. Irvine and T. Levin, *Toward a taxonomy and costing method for security metrics*, Technical Report, Department of Computer Science, Naval Postgraduate School, Monterey, CA, in progress.

[28] T. Kidd and D. Hensgen, *Why the Mean is Inadequate for Making Scheduling Decisions*, Technical Report, Department of Computer Science, Naval Postgraduate School, Monterey, CA, Jan. 1999.

[29] T. Kidd, D. Hensgen, R. Freund, and L. Moore, "SmartNet: a scheduling framework for heterogeneous computing," *Proc. 2nd Intl. Symposium on Parallel Architectures, Algorithms, and Networks*, June 1996, pp. 514-521.

[30] J. P. Kresho, *Quality Network Load Information Improves Performance of Adaptive Applications*, Thesis, Department of Computer Science, Naval Postgraduate School, Monterey, CA, Sept. 1997.

[31] J. P. Kresho, D. Hensgen, T. Kidd, and G. Xie, "Determining the accuracy required in resource load prediction to successfully support application agility," *Proc. 2nd IASTED Intl. Conf. European Parallel and Distributed Systems*, July 1998, pp. 244-254.

[32] J. Larus and E. Schnarr, "EEL: machine-independent executable editing," *SIGPLAN PLDI 95*, 1995, pp. 291-300.

[33] T. Levin and C. Irvine, *An approach to characterizing resource usage and user preferences in benefit functions*, Technical Report, Department of Computer Science, Naval Postgraduate School, Monterey, CA, in progress.

[34] T. Levin and C. Irvine, *Quality of security service in a resource management system benefit function*, Technical Report, Department of Computer Science, Naval Postgraduate School, Monterey, CA, in progress.

[35] J. W. S. Liu, K. Nahrstedt, D. Hull, S. Chen, and B. Li, *EPIQ QoS Characterization, Draft Version*, July 1997, http://epiq.cs.uiuc.edu/files/qos-970722.pdf .

[36] M. Livny, M. Litzkow, T. Tannenbaum, and J. Basney, "Checkpoint and migration of UNIX processes in the Condor distributed processing system," *Dr Dobbs Journal*, Feb. 1995, pp. 40-51.

[37] J. P. Loyall, R. E. Schantz, J. A. Zinky, and D. E. Bakken, "Specifying and measuring quality of service in distributed object systems," *Proc. 1st Intl. Symposium on Object-Oriented Real-Time Distributed Computing*, April 1998, pp. 20-22.

[38] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems," *Proc. 8th IEEE Heterogeneous Computing Workshop*, April 1999, to appear.

[39] M. Maheswaran, T. D. Braun, and H. J. Siegel, "Heterogeneous distributed computing," *Encyclopedia of Electrical and Electronics Engineering*, J. Webster, ed., John Wiley & Sons, New York, NY, to appear 1999.

[40] M. Maheswaran and H. J. Siegel, "A dynamic matching and scheduling algorithm for heterogeneous computing systems," *Proc. 7th IEEE Heterogeneous Computing Workshop*, Mar. 1998, pp. 57-69.

[41] N. W. Porter, *Resource Requirement Analysis of GCCS Modules and EADSim and Determination of Future Adaptivity Requirements*, Thesis, Department of Computer Science, Naval Postgraduate School, Monterey, CA, June 1999.

[42] A. Reinefeld, R. Baraglia, T. Decker, J. Gehring, D. Laforenza, J. Simon, T. Römke, and F. Ramme, "The MOL project: an open extensible metacomputer," *Proc. 6th IEEE Heterogenous Computing Workshop*, April 1997, pp. 17-31.

[43] D. St. John, S. Kidd, D. Hensgen, T. Kidd, and M. Shing, *Experiences using semi-formal methods in MSHN*, Technical Report, Department of Computer Science, Naval Postgraduate School, Monterey, CA, Feb. 1999.

[44] M. C. L. Schnaidt, *Design, Implementation, and Testing of MSHN's Application resource Monitoring Library*, Thesis, Department of Computer Science, Naval Postgraduate School, Monterey, CA, Dec.1998.

[45] P. Schneck and K. Schwan. "Dynamic authentication for high-performance networked applications," *Proc. 6th IEEE/IFIP Intl. Workshop on Quality of Service*, May 1998, pp. 127-136.

[46] J. Sydir, B. Sabata, and S. Chatterjee, " QoS middleware for the next-generation Internet," position paper, *Proc. NASA/NREN Quality of Service Workshop*, Aug. 1998, pp. 25-27.

[47] M. Tan and H. J. Siegel, "A stochastic model for heterogeneous computing and its application in data relocation scheme development," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9, No. 11, Nov. 1998, pp. 1088-1101.

[48] M. Tan, H. J. Siegel, J. K. Antonio, and Y. A. Li, "Minimizing the application execution time through scheduling of subtasks and communication traffic in a heterogeneous computing system," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 8, No. 8, Aug. 1999, pp. 857-871.

[49] R. van Renesse, K. Birman, M. Hayden, A. Vaysburd, and D. Karr, *Building adaptive systems using Ensemble*, Cornell University Technical Report, TR97-1638, July 1997.

[50] R. van Renesse, K. Birman, and S. Maffeis, "Horus, a flexible group communication system," *Communications of the ACM*, April 1996, pp. 76-83.

[51] R. van Renesse and A. S. Tanenbaum, "Distributed operating systems," *ACM Computing Surveys*, Vol. 17, No. 4, Dec. 1985, pp. 419-470.

[52] R. Vanegas, J. A. Zinky, J. P. Loyall, D. A. Karr, R. E. Schantz, and D. E. Bakken. "QuO's runtime support for quality of service in distributed objects," *Proc. IFIP Intl. Conf. on Distributed Systems Platforms and Open Distributed Processing* , Sept. 1998, pp. 207-222.

[53] J. Walpole, C. Krasic, L. Liu, D. Maier, C. Pu, D. McNamee, and D. Steere, "Quality of service semantics for multimedia database systems," *Proc. Data Semantics 8: Semantic Issues in Multimedia Systems IFIP TC-2 Working Conference*, Jan. 1999, pp. 393-412.

[54] L. Wang, H. J. Siegel, V. P. Roychowdhury, and A. A. Maciejewski, "Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach," *Journal of Parallel and Distributed Computing*, Vol. 47, No. 1, Nov. 199, pp. 8-22.

[55] L. R. Welch, B. Ravindran, B. A. Shirazi, and C. Bruggeman, *Specification and modeling of dynamic, distributed real-time systems*, Technical Report Number TR-CSE-98-003, Department of Computer Science and Engineering, The University of Texas at Arlington, Arlington, TX, Sept. 1998.

[56] L. R. Welch, B. Ravindran, B. A. Shirazi, and C. Bruggeman, "DeSiDeRaTa: QoS Management Technology for Dynamic, Scalable, Dependable, Real-time Systems," *Proc. 15TH IFAC Workshop on Distribute Computer Control Systems*, Sept. 1998, pp. 7-12.

[57] R. E. Wright, *Management System for Heterogeneous Networks Security Services*, Thesis, C4I Academic Group, Naval Postgraduate School, Monterey, CA, June 1998.

[58] R. E. Wright, D. J. Shifflett, and C. E. Irvine, "Security for a virtual heterogeneous machine," *Proc. 14th Computer Security Applications Conference*, Dec. 1998, pp. 167-177.

Biographies

Debra Hensgen received her PhD in the area of Distributed Operating Systems from the University of Kentucky. She is an Associate Professor in the CS Department at The Naval Postgraduate School. She has co-authored numerous papers about the Concurra toolkit for automatically generating safe, efficient concurrent code, the Graze parallel processing performance debugger, the SAAM path information base, and the SmartNet and MSHN Resource Management Systems.

Taylor Kidd obtained his PhD from the University of California at San Diego in 1991. His interests include both theoretical and applied distributed computing. He led the research component of the SmartNet team at NRaD and instigated a number of important advances to the SmartNet Scheduling Framework. He is a co-PI for the DARPA-sponsored MSHN project and a co-investigator on the DARPA-sponsored SAAM Project.

David St. John is the head of staff at the Heterogeneous Network & Computing Laboratory, Naval Postgraduate School. He has over six years experience in object-oriented software development in the areas of distributed computing, process control, sensor collection, and Internet transaction processing systems. He is a member of IEEE and IEEE Computer Society. He received a BSME with High Honors from the University of Florida and an MSE from the University of California, Irvine.

Matt Schnaidt earned his professional engineering license while serving as the Battalion Adjutant. He received an MS from the Computer Science Department of the Naval Postgraduate School in 1998. Currently, Major Schnaidt is working on the Battle Management Command, Control and Communication component of the National Missile Defense Program.

H. J. Siegel is a Professor in the School of Electrical and Computer Engineering at Purdue University. He is an IEEE Fellow and an ACM Fellow. He received two BS degrees from MIT, and the MA, MSE, and PhD degrees from Princeton University. He has coauthored over 250 technical papers, was a Coeditor-in-Chief of the Journal of Parallel and Distributed Computing, and was an editor of the IEEE Transactions on Parallel and Distributed Systems.

Tracy D. Braun is a PhD student and Research Assistant at Purdue University. He received his BSEE with Honors and High Distinction from the University of Iowa in 1995. In 1997, he received his MSEE from the School of Electrical and Computer Engineering at Purdue. He is a member of IEEE, IEEE Computer Society, and Eta Kappa Nu honorary society. His research interests include parallel algorithms, heterogeneous computing, computer security, and software design.

Muthucumaru Maheswaran is an Assistant Professor in the Department of Computer Science at the University of Manitoba, Canada. He received a BSc degree from the University of Peradeniya, Sri Lanka and the MSEE and PhD degrees from Purdue University. He received a Fulbright scholarship to pursue his MSEE degree at Purdue University. His research interests include computer architecture, distributed computing, heterogeneous computing, and resource management systems for metacomputing.

Shoukat Ali is an MSEE student at the School of Electrical and Computer Engineering at Purdue University. His main research topic is dynamic mapping of meta-tasks in heterogeneous computing systems. He has held teaching positions at Aitchison College and Keynesian Institute of Management and Sciences, both in Lahore, Pakistan. Shoukat received his BS degree from the University of Engineering and Technology, Lahore, Pakistan in 1996. His research interests include computer architecture, parallel computing, and heterogeneous computing.

Jong-Kook Kim is an MSEE student and Research Assistant in the school of Electrical and Computer Engineering at Purdue University, currently working on the DARPA/ISO sponsored BADD program. He received his BSEE from Korea University, Korea. He served in the ROK Army working with the US Army on the Theater Automated Command and Control Information Management System and received the US Army Commendation Medal. His research interests include heterogeneous computing and performance measures for distributed systems.

Cynthia E. Irvine is Director, Naval Postgraduate School Center for INFOSEC Studies and Research and an Assistant Professor of Computer Science at the Naval Postgraduate School. Dr. Irvine holds a PhD from Case Western Reserve University. She has over twelve years experience in computer security research and development. Her current research centers on architectural issues associated with applications for high assurance trusted systems, security architectures combining popular commercial and specialized multilevel components, and the design of multilevel secure operating systems.

Timothy Levin is currently doing research at the Naval Postgraduate School. He received a BS in Computer and Information Science from the University of California at Santa Cruz, 1981. His secure system work includes design of security features, and formal verification and formal covert channel analysis of an A1 operating system, and enterprise security features for a commercial relational database system. He has been certified by the NSA as a Vendor Security Analyst, for participation in their Trusted Product Evaluation Program.

Richard Freund is a founder and CEO of NOEMIX, a San Diego based startup to commercialize distributed computing technology. Freund is also one of the early pioneers in the field of distributed computing, in which he has written or co-authored a number of papers. In addition he is a founder of the Heterogeneous Computing Workshop, held each year in conjunction with IPPS/SPDP. Freund won a Meritorious Civilian Service Award during his career as a government scientist.

Matt Kussow, B.S.C.S., has 12 years of experience in software development, research, design, process analysis, and project management. Currently he holds the position of Vice President of Product Development at Noemix. He has extensive experience in designing and developing software for high performance computing, parallel algorithms, network computing, and database systems.

Michael Godfrey, B.S. C.I.S, UCSD Java Certified, has over 6 years of software research, design, and development experience with high performance computing, secure world wide web, health care, and data base systems for Science Applications International Corporation (SAIC) and Noemix, Inc. He has developed systems level applications on a variety of UNIX and Win32 platforms.

Alpay Duman is a LTJG in the Turkish Navy. He graduated from the Turkish Naval Academy with a BS in Operations Research with honors. He received his MSCS degree in the area of Systems Design and Architecture from the Naval Postgraduate School. He is currently a systems engineer at Turkish Navy Software Development Center working on a CORBA based communication infrastructure for Command Control Systems.

Paul F. Carff, LT US Navy, is a Masters student in the Computer Science Department at the Naval Postgraduate School, working on the Management System for Heterogenous Systems (MSHN). He received a BS in Engineering Physics from Santa Clara University in 1991. Prior to coming to Naval Postgraduate School, LT Carff served 3 years as a Nuclear Power Officer aboard the USS Salt Lake City (SSN 716).

Shirley Kidd is one of the supporting staff at the Heterogeneous Network & Computing Laboratory. She has 4 years of experience in the aerospace industry and another 6 years at a commercial marketing company during which she worked in both industries as a programmer analyst. She has a BS in Applied Mathematics from the University of California, San Diego.

Viktor K. Prasanna (V.K. Prasanna Kumar) is a Professor in the Department of Electrical Engineering Systems, University of Southern California, Los Angeles. He obtained his PhD in Computer Science from Pennsylvania State University in 1983. He has published and consulted for industries in parallel computation, computer architecture, VLSI computations, and high performance computing for signal and image processing, and vision. He serves on the editorial boards of the Journal of Parallel and Distributed Computing and IEEE Transactions on Computers. He is the founding Chair of the IEEE Computer Society Technical Committee on Parallel Processing, and a Fellow of the IEEE.

Prashanth B. Bhat is a PhD candidate in Computer Engineering at the University of Southern California, Los Angeles. He received his B.Tech. degree in Computer Engineering from the Karnataka Regional Engineering College, India, in 1992. He received his ME degree in Computer Science and Engineering from the Indian Institute of Science, Bangalore, in 1994. His research interests include scheduling techniques for parallel and distributed systems, High Performance Computing and parallel computer architecture. During the summer of 1998, he was a research intern at Hewlett-Packard laboratories, Palo Alto.

Ammar Alhusaini is a PhD student in the Electrical Engineering Department at the University of Southern California. His main research interest is task scheduling in heterogeneous environments. He received an MS degree in computer engineering from the University of Southern California in 1996. He is a member of IEEE, IEEE Computer Society, and ACM.