

# An Architecture for Network Resource Monitoring in a Distributed Environment

Philip M. Irely IV, Robert W. Hott, David T. Marlow  
NSWC-DD Code B35  
17320 Dahlgren Road  
Dahlgren, VA 22448-5100  
pirey, rhott, dmarlow@nswc.navy.mil

## Abstract

*As part of its HiPer-D Program, the United States Navy is developing an experimental distributed system which achieves survivability by dynamically reconfiguring the system using replicated system components and resources. To enable the reconfiguration, resource monitors observe the behavior of the system and report this information to a resource manager. The resource manager makes reconfiguration decisions based on this information. Because all reconfiguration decisions are based on data obtained from resource monitors and the network is the common resource linking all components in the distributed system, this paper focuses specifically on network resource monitoring. A generalized network resource monitor architecture is proposed. Two instantiations of this architecture are then presented. The first is based on custom developed tools tailored to a specific application while the second is based on commercially available products (e.g. SNMP, RMON, etc.). Scalability, intrusiveness, and fidelity are identified as evaluation criteria against which implementation trade-offs are made. This paper presents the results of initial experiments as well as future research directions.*

## 1 Introduction

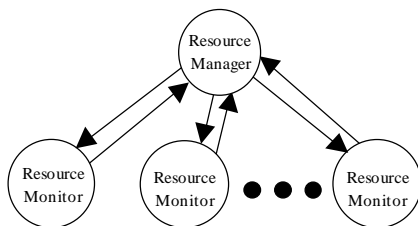
In the past, the United States Navy has used specialized system components such as custom developed computers and high-speed point-to-point communications channels to build combat systems. Specialized components were needed to meet the demanding real-time and survivability requirements of these systems. The requirements of real-time control loops, such as required to successfully engage a target moving at high speed, were generally beyond what could be provided by standard commercial technologies. In addition, the survivability requirements of a combat system dictated that critical system components be available when they are needed.

Today, new technology is rapidly being developed in the commercial sector that shows potential to meet the stringent requirements of the combat system. Consequently, the United States Navy is evaluating the use of Commercial-off-the-Shelf (COTS) technology to obviate the need for the specialty components, to keep pace with the state of technology, and to reduce system cost. HiPer-D is a program within the United States Navy investigating the use of COTS components to build distributed combat systems. Currently, the HiPer-D test-bed is composed of 30 workstations and servers interconnected using ATM, FDDI, and Ethernet networks. The test-bed is used to simulate an anti-air warfare system. Targets are tracked, classified, targeted, and fired upon within the simulation test-bed. Hiper-D is investigating a number of issues associated with the ability of COTS components to meet the demanding real-time and survivability requirement of the combat system.

HiPer-D is investigating how to achieve system survivability using replicated system components and resources. Resource monitors observe the behavior of the system and report this information to a resource manager as shown in Figure 1. If the resource manager determines that critical components have failed or if the available

resources have fallen below the requirements of the system, it can reconfigure the system using the replicated system components and resources at its disposal.

A resource manager must understand the requirements of the system to constructively reconfigure it. Key to the successful operation of a resource manager is



**Figure 1 Basic Resource Management and Monitoring Architecture**

the quality of the information provided to it by its resource monitors. Network resource monitors are particularly important in the reconfiguration process of distributed systems and are the focus of this paper. Issues associated with network resource monitoring and an approach for constructing network resource monitors are presented.

## 2 Related Work

There are numerous research efforts underway that are investigating the topics of resource management and resource allocation. [7] presents a complementary architecture in which a "distributed application management system" communicates with instrumented processes. Within the instrumented process, a management coordinator communicates with "sensors" or "actuators" to obtain information about the process or to change the characteristics of the process. The architecture in [7] is consistent with the monitoring architecture that we propose. The primary difference is that our architecture specifically deals with network monitoring, which could be considered an extension of the "process" paradigm. [8] observes that data obtained during the monitoring of resources could be correlated using relational database technologies. Our paper does not deal with the correlation of the data collected but discusses a natural evolution from the network monitoring tool used in [8] to the use of RMON probes. We also describe an approach that provides greater "fidelity" than an RMON based approach. Efforts within the IETF's Real-time Traffic Flow Measurement Working Group are also beginning to address the need to measure end-to-end traffic flows.

## 3 System Model

Systems considered for monitoring in this paper are comprised of up to  $10^2$  interconnected networks,  $10^3$  computers, and  $10^4$  processes. These numbers correspond to the expected number of resources on future naval vessels. To monitor a system of this size, it must be decomposed into smaller components.

We use the dynamic path abstraction defined in [2] to decompose the system. Instead of monitoring the network resources of the communications infrastructure as a whole, specific application level communication paths through the system are monitored. Each path corresponds to a critical series of communications through the system. A path can consist of just two application processes (e.g. a point-to-point communication) or a series of them.

## 4 Network Resource Monitoring

In this section, an architecture is proposed for a network resource monitor. Given that architecture, several metrics appropriate for network resource monitoring along with instrumentation points for monitoring these metrics are presented. Finally, evaluation criteria for comparing network resource monitor implementations are presented.

### 4.1 Network Resource Monitor Architecture

The proposed architecture for a network resource monitor is shown in Figure 2. It has three components: (1) network sensors; (2) a sensor director; and (3) a database. Network sensors are responsible for collecting network performance data. The sensor director initiates the collection of data by the network sensors in response to requests from the resource manager. These requests come in the form of a list of application level paths to monitor and the metrics to monitor for each of those paths. Each path is described by the list of application processes in the order in which they occur on that path. The sensor director records data collected by the sensors in the database if requested. If desired, it can also report the results back to the resource manager either synchronously or asynchronously in the form of (path, metric)-tuples. The database enables both current value and last known value reporting to the resource manager. It should be noted that the resource manager understands the system topology at the Application and Support layer as shown in Figure 3 and need not have knowledge of the structure of the underlying layers.

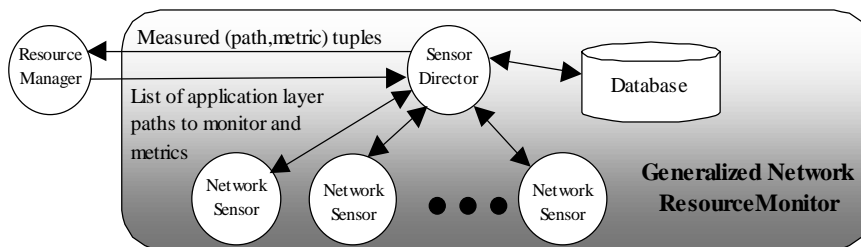


Figure 2 Network Resource Monitor Architecture

### 4.2 Metrics for Network Resource Monitoring

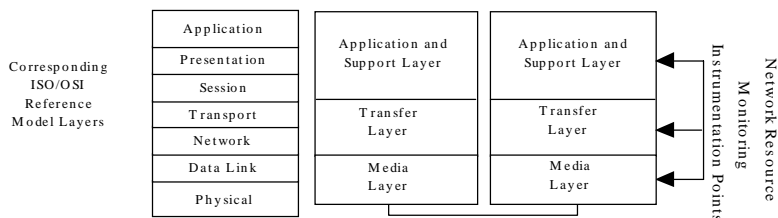
Three metrics shall be considered by network resource monitors in this paper: end-to-end throughput, one-way-latency, and reachability. The end-to-end throughput and one-way latency are defined in detail in [1] and [3]. Other metrics defined in those papers are important and can be used for network resource monitoring but will not be discussed in this paper. The reachability metric is used to determine if a host can communicate with another host through the communication infrastructure.

### 4.3 Instrumentation Points for Network Resource Monitoring

There are a number of points in a system where the aforementioned metrics can be collected as shown in Figure 3. Each monitoring point has advantages and disadvantages as will be shown below.

Since all messages exchanged in the system pass over the communications media, the Media layer would appear to be a good place to monitor. In reality, however, there are difficulties in monitoring the metrics of interest at this layer. The

end-to-end throughput and one-way latency metrics, as defined, measure performance realized at the Application and Support layer. Monitoring at the Media layer for these metrics only approximate the performance which would be measured at the Application and Support layer. Metrics collected at the Media layer can be combined with metrics collected at higher layers to improve the accuracy of the approximation. The reachability metric may be partially realized on a broadcast media by monitoring the Media layer and noting occurrences of packets whose source address is that of the source host being tested for reachability. This implies periodic messages sent from the source host of interest that may be part of the normal operation of the application or may be strictly to enable this type of instrumentation. This solution does not address all the issues for monitoring this metric at this layer, however. In an environment where asymmetric routes exist between two hosts, information may flow in one direction but not the in the other. Thus, receiving packets from a host does not mean that you can transmit packets to that host. In a switched environment, “sniffing” may not be possible since a non-broadcast media is used.



**Figure 3 Network Resource Monitoring Instrumentation Points**

Monitoring within the Transfer layer provides a closer approximation to the performance measured in the actual application than monitoring at the Media layer. Again, these measurements are only approximations.

All metrics of interest can be monitored accurately within the Application and Support layer. Network resource monitors, such as those defined in the “high-fidelity” implementation in Section 5.1, can be set up to accurately model the behavior of applications and to collect the metrics of interest.

#### 4.4 Evaluation Criteria

Three criteria will be used to evaluate network resource monitoring implementations: fidelity, intrusiveness, and scalability. Implementations presented here will be evaluated subjectively against the criteria rather than quantitatively. An ideal implementation would have high fidelity, be minimally intrusive, and be highly scalable.

Fidelity is the ability of the implementation to accurately capture the metrics presented in Section 4.2. There are two important components of the fidelity metric: senescence and accuracy. Senescence is a measure of the age of the data collected for a given metric. The older the data, the less value it has to the system. Accuracy relates to the measurement quality for a given metric. Factors that contribute to the measurement quality include how well a metric captures the requirements of the application and the precision of the measurement tool in gathering that metric.

Intrusiveness is a measure of the network overhead introduced by the implementation to do network resource monitoring. Although other measures of

intrusiveness relate to network resource monitoring (e.g., CPU utilization) they will not be considered here.

Scalability deals with the ability of the implementation to operate correctly and perform satisfactorily over a range of system sizes. Although an implementation may perform well for a system with a few hosts, it may not perform well in a system consisting of hundreds of systems such as those considered here.

## **5 Network Resource Monitor Implementations**

Two network resource monitoring implementations are presented below. The High Fidelity implementation was defined to meet the requirements of an application which needed high fidelity network resource monitoring. The Scalable implementation was defined to address some of the scalability problems introduced by the High Fidelity implementation.

### **5.1 High Fidelity Network Resource Monitor Implementation**

The need for high fidelity network resource monitoring was identified for a client/server based application used in the HiPer-D [5] test-bed. The Radar Track Data Server (RTDS) distributes radar tracks to clients for processing. The correct operation of the combat system depends on the availability of the client and server components and the resources they need to operate. Consequently, the client and server processes were put under the control of the resource manager. When the resource manager determines that a process fails or becomes unreachable from reports received by its resource monitors, it selects a new host on which to resume the operation of the failed process. A customized network resource monitor was developed since there were no known commercial products that could easily collect the required network performance data.

#### **5.1.1 Application Monitoring Requirements**

The test-bed host hardware and software configurations require that client and server applications be executed on processors from a specific pool of hosts. In the test-bed, the number of clients,  $C$ , in the client pool was 9 hosts and the number of servers,  $S$ , in the server pool was 3 hosts as shown in Figure 4(a).

#### **5.1.2 Path List**

A network resource monitoring path list is constructed by the resource manager and sent to the resource monitor. This path list includes a path from each server in the server pool to all clients in the client pool. There were a total of 27 different paths,  $C*S$ , to be monitored in the HiPer-D configuration, as shown in Figure 4(b).

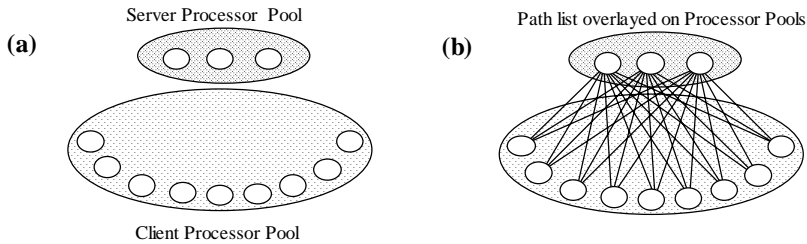
#### **5.1.3 Metric Collection**

NTTCP [1], a communications analysis and simulation tool developed by NSWC-DD was modified to serve as a network resource monitor by collecting the metrics of interest for the 27 paths required by the RTDS application. NTTCP was already capable of measuring all the metrics of interest (and many more) but tradeoffs led to specific configuration option settings and a structural redesign.

##### **5.1.3.1 High Fidelity Data Collection**

To achieve high fidelity measurements on a path, the NTTCP network analysis tool was configured to mimic the behavior of the RTDS server. Two NTTCP parameters were set to do this: the inter-send time [4] (i.e., the period between the

transmission of successive messages),  $P$ , and the length of the messages to be sent,  $L$ . Each was set to a value found experimentally to correspond with the behavior of the RTDS application.



**Figure 4 Processor Pool and Path List for Application of Interest**

To reduce the intrusiveness of the network resource monitoring function, the number of messages to send, was set to a fixed value. Instead of sending a constant stream of data and periodically sampling that stream, the data was sent in bursts. The longer the burst, the more intrusive the measurement operation. Experiments have shown that bursts, which are too short, yield inaccurate results because they are too susceptible to transient conditions. For each application, an optimal burst size should be found through experimentation.

### 5.1.3.2 Fidelity Versus Scalability Tradeoff

NTTCP is typically used to analyze the communications performance between a pair of hosts. In the RTDS application, 27 pairs of hosts need to be monitored. It can be a very intrusive operation to monitor all host pairs simultaneously. The peak overhead introduced for this configuration is  $C*S*(L/P)$ . Values typical of the HiPer-D testbed would result in an overhead of 59 megabits/second ( $9*3*(8192 \text{ bytes}/.03 \text{ seconds})*(8 \text{ bits/byte})$ ). In this case, a single application is consuming a significant percentage of the capacity of both the FDDI and ATM networks used in the testbed. Scaling to more than one application would not be possible for this implementation.

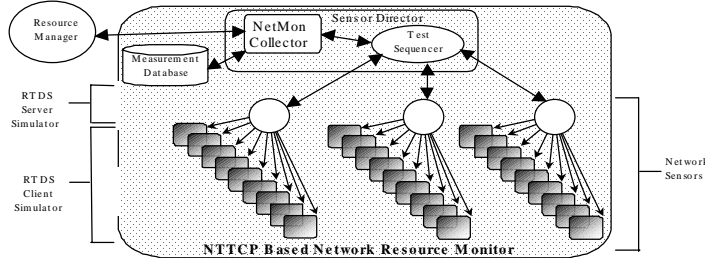
To reduce the overhead of testing all paths simultaneously, a sequencer was implemented within the network resource monitor. The sequencer was used to test each path in series rather than all of them in parallel. With this implementation, the peak overhead was only 2.18 megabits/second ( $(8192 \text{ bytes}/.03 \text{ seconds})*(8 \text{ bits/byte})$ ). This implementation increased the senescence of the data, however. The minimum time between samples for a given path was now  $C*S*T$ , where  $T$  is the time it takes to do a single sample for a single path.

### 5.1.3.3 Intrusiveness Versus Fidelity Tradeoff

To calculate one-way latency, NTTCP normally computes the clock offset between the hosts under test. This computation requires a number of packets to be exchanged between the hosts. It was determined that the overhead of the clock offset calculation was significantly intrusive compared to the overhead of running a clock synchronization protocol (e.g. NTP) for network resource monitoring of the RTDS application.

### 5.1.4 Test Sequencer

The basic operation of the sequencer is shown in Figure 5. The system resource manager passes its monitoring request (as described in Section 4.1) to the network resource monitor. The NetMon collector within the network resource monitor receives this list and formats it for delivery to the test sequencer. The NetMon Collector and Test Sequencer correspond to the sensor director in the network resource monitor architecture.



**Figure 5 Network Resource Monitor for Radar Track Data Server**

The test sequencer and RTDS server simulators perform the sequential (rather than parallel) data collection process. The test sequencer passes a list of RTDS clients to monitor to the first RTDS server simulator. The server monitors the performance to each client in sequence saving the metrics collected for each client. When the metrics are collected for the last client, the RTDS server simulator passes these results back to the test sequencer where they are recorded. The test sequencer then passes the next list of RTDS clients to be monitored to the next RTDS server. When the results are reported back from the last RTDS server, the test sequencer passes the results of measurements for all clients back to the NetMon collector where they are available to the system resource manager.

## 5.2 Scalable Network Resource Monitoring Implementation

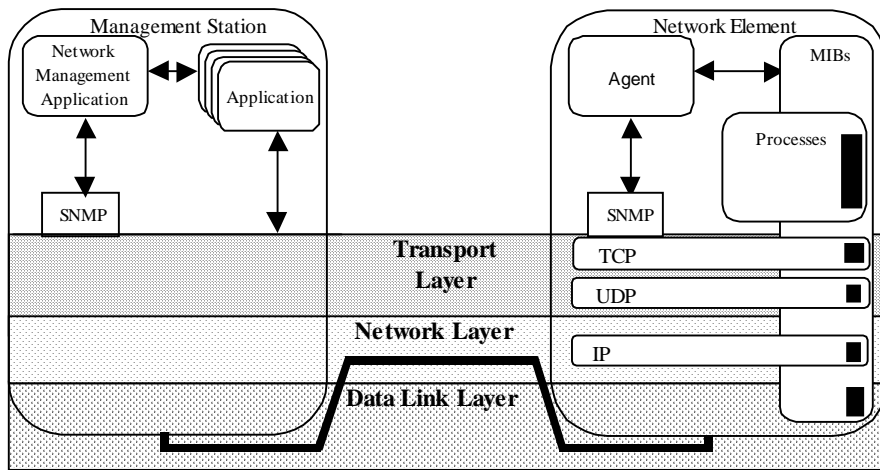
Two limitations of the high fidelity implementation led to the investigation of other designs. First, the lack of scalability in the high fidelity implementation motivated a more scalable design. Second, there was a desire to move away from custom designed solutions and to use COTS solutions if possible.

### 5.2.1 COTS Network Management Overview

One potential alternative for network resource monitoring is the use of COTS network management components and technologies. Network management provides a means for network operators to detect, locate and recover from faults and to monitor and control security, configuration, and performance. Network management capabilities are typically embedded within commercial networking products to enable network operators to effectively run their networks. To build a network resource monitor using COTS products, we need to determine which network management capabilities can provide the needed functionality.

The most common network management framework in use is comprised of management stations and network elements as shown in Figure 6. The management station generates requests for information from management agents and performs actions (or generates reports) based upon information it receives. The agent collects information from within network elements. The information collected is logically

grouped into Management Information Bases (MIBs). Management information is generally available at all layers of a system, from the process, or application, down to the physical network interface. The standard protocol used to communicate this management information is the Simple Network Management Protocol (SNMP).



**Figure 6 SNMP Network Management Architecture**

A management station periodically polls agents, embedded in the network elements, to obtain dynamic and statistical MIB parameters. An additional event driven capability, called a trap, exists where agents can asynchronously pass information to management stations upon the occurrence of significant events. SNMP standard MIBs provide basic information about network elements including information about interface(s) and standard protocols (e.g. IP, TCP, and UDP). Most vendors also provide additional information, in the form of private MIBs, specific to their products.

Additional network management functionality is brought by the RMON (remote network monitoring) MIB specifications which provide two basic capabilities: collecting data for an entire subnetwork (e.g. an Ethernet LAN) and performing the function of a programmable network monitor (or probe).

An RMON probe can passively monitor a subnetwork, actively filter data packets, identify a triggering condition, capture packets, asynchronously inform a management station with a trap, and support the download of captured packets to a management station. The RMON specifications support extended analysis by supplying MIB variables with extended statistical information. The RMON specifications support remote monitoring at a number of protocol levels (i.e. subnetwork, IP, TCP and Application levels).

### **5.2.2 Using COTS Networking Management for Network Resource Monitoring**

Various components of COTS networking management products look useful for building a scalable network resource monitor. No COTS networking components were identified to directly replace the sensor director. It appears that this may have to

be a custom developed application. Various network management components could be used as network sensors. For example, the sensor director could translate (path, metric)-tuples sent by the resource manager to SNMP MIB queries. Depending on how the resource monitor requested the metric to be reported back (e.g. request/response or polled), the results of the query may or may not be written to the database. If the request was for the current value (e.g. request/response), the value of the MIB variable could be directly reported back to the resource manager without writing it to the database. Traps could be used for asynchronous notification. For example, if the resource monitor requests to be notified when the communications capacity falls below a certain value, a trap could be set up in an RMON probe or probes to monitor network capacity on the specified path. When a probe generates a trap, it is fielded by the sensor director and reported back to the resource manager.

The COTS network management framework is very similar to the architecture identified for the generalized network resource monitor. One comparison of the two models aligns the management station with the resource manager. In this view, the network sensors are aligned with the network elements, the sensor director with the agent, and the database(s) with the MIB(s). A second comparison of the two models aligns the management station with the sensor director and the sensors with agents. The database capability, a common feature in management stations, would be required in order to deal with the resource manager. This view presents the more traditional relationship between the network element and the agent (e.g., a single agent for a workstation).

### **5.2.3 Initial COTS Based Network Resource Monitor Experiment**

Initial experiments are underway to evaluate the feasibility of COTS based network management technology for network resource monitoring. The ongoing evaluations are utilizing the NTTCP tool to simulate the application load and COTS network management components to monitor that load. The benefit of using NTTCP in this environment is that it allows for direct comparison between the scalable network resource monitoring implementation and the high fidelity monitoring implementation. Tests were performed by injecting simulated application loads onto the LAN and monitored using RMON and standard MIBs. In addition, the asynchronous trap processing capabilities of SunNet Manager have been tested.

Network hardware used in the testbed is interconnected with a shared Ethernet LAN and includes: a single-port Ethernet RMON probe from SolCom Systems Ltd., Cisco 7513 routers with SNMP-based agents, Pentium II PCs, and Sun Ultra 2 workstations. Software used in the testbed includes: the NTTCP network analysis tool, SolCom's LANmaster RMON Utilities, and Sun's SunNet Manager.

### **5.2.4 Issues with using COTS Network Management**

Initial testing has shown that the SolCom RMON probe was capable of collecting RMON metrics during heavy load conditions on a shared Ethernet LAN. Under the same load conditions, the Cisco router was also capable of collecting standard interface metrics. The RMON probe was capable of generating traps when thresholds were exceeded. During very high load test situations, SNMP requests and responses, including traps, were lost. This was likely due to the SNMP being transported over the unreliable User Datagram Protocol (UDP). Under lighter network loads, the threshold trap capabilities were tested on the RMON probe. Neither the RMON probe nor the Cisco router was capable of matching the fidelity of the NTTCP

network analysis tool. Both systems provide a number metrics that may be used to approximate end-to-end throughput (e.g., utilization, octets transferred, packets transferred, and protocol, and interface specific counters). Clock granularity appears to be limited in both the probe and the router. This also appears to be an issue with the management system applications. Experiments were also performed to test the ability of SunNet Manager to accept large numbers of traps within a short period of time. Fixed numbers of traps were launched to the management station. The trap counts and timing were examined at the trap reporting application level. Results were dependent upon the platform configuration (e.g., memory, CPU). Experiments showed that the management station could be overrun by asynchronous traps.

Network management offers a great deal of potential for supporting the real-time network resource monitor function addressed by this paper. Use of standard MIB information is supported in most COTS network components providing a cost effective and standards based means of implementing portions of the network resource monitor function.

The two primary issues regarding the use of SNMP management to support the real-time network resource monitor function are the availability of needed information within standard MIBs and the fidelity of the information obtained. The standard MIBs provide only basic functionality; for example, each TCP connection has twenty two separate state variables, SNMP's standard MIBs support the exchange of only five of these items (see page 111 of [6]). Further experiments are planned to address the fidelity issue. Real-time network monitoring requires the knowledge of the networking components current state. The update rate of critical dynamic MIB parameters is a vendor specific issue that may even change between product updates.

SNMP based network management supports approaches for obtaining network resource monitoring information which are not intrusive; however, if not properly architected, they too can be intrusive. For example, short interval, periodic polling of a large network, or the heavy use of downloading captured information from RMON probes can introduce a significant overhead on the network. Since SNMP based management uses connectionless data exchange, a network monitor may need to perform background polling to detect network failure between it and the network element which would prevent the reception of traps.

## **6 Conclusions**

The network resource monitor architecture defined looks promising for implementing network resource monitors. Two instantiations of the architecture are shown. The high fidelity implementation presented collects the data of sufficient quality but lacks scalability and is intrusive. The scalable network management based implementation has the potential for providing the tools at little additional cost since they are built into COTS products, but some components require considerable customization. There are concerns as to whether monitoring can be done at the fidelity required and whether consistent results will be obtained from various COTS vendors. It should be noted that COTS based solutions were not intended to address the complex requirements associated with selecting application resources to use in the real-time distributed environment.

## **7 Future Work**

A series of future experiments are planned to compare the fidelity, scalability, and intrusiveness of the implementation approaches presented. These experiments

will employ quantitative analysis of the experimental data collected. A promising approach appears to be a hybrid implementation that combines components of the high fidelity and scalable COTS implementations.

## 8 References

- [1] Philip M. Irey IV, Robert D. Harrison, David T. Marlow, "Evaluating LAN Communications Performance for a Real-Time Environment", Proceedings of the 4<sup>th</sup> International Workshop on Parallel and Distributed Real-Time Systems, 1996.<sup>†</sup>
- [2] Lonnie R. Welch, "Large-Grain, Dynamic Control System Architectures", Proceedings of The Joint Workshop on Parallel and Distributed Real-Time Systems.
- [3] Philip M. Irey IV, Robert D. Harrison, David T. Marlow, "Techniques for LAN Performance Analysis in a Real-Time Environment", Real-Time Systems – International Journal of Time Critical Computing Systems, Volume 14, Number 1, January 1998.<sup>†</sup>
- [4] Philip M. Irey IV, David T. Marlow, Robert D. Harrison, "Distributing Time Sensitive Data in a COTS Shared Media Environment", Proceedings of the Joint Workshop on Parallel and Distributed Real-Time Systems, 1997.<sup>†</sup>
- [5] Geary and Masters, "Investigating New Computing Technologies for Shipboard Combat Systems" Naval Engineers Journal, Volume 107, Number 3, May 1995.
- [6] William Stallings, "SNMP, SNMPv2, and RMON: Practical Network Management, Second Edition", Reading, Massachusetts, Addison Wesley Longman, Inc., 1996.
- [7] Michael J. Katchabaw, Stephen L. Howard, Hanan L. Lutfiyya, Andrew D. Marshall, Michael A. Bauer, "Making Distributed Applications Manageable Through Instrumentation", Proceedings of 2nd International Workshop on Software Engineering for Parallel and Distributed Systems, 1997.
- [8] Leander Conradic, Maria-Athina Mountzia, "A Relational Model for Distributed Systems Monitoring using Flexible Agents", Proceedings of 3rd International Workshop on Services in Distributed and Networked Environments, 1996.

<sup>†</sup>See <http://sholeh.nswc.navy.mil/documents> for electronic copies of these documents.