Cover Page

U.S. Department of Energy Office of Science
Scientific Discovery through Advanced Computation Solicitation LAB01-06
National Collaboratories and High Performance Networks

Self-Configuring Network Monitor

A High Performance Network Engineering Proposal: Network Measurement and Analysis
For the period June 1, 2001 – May 31, 2004

Principal Investigator

Brian L. Tierney

Lawrence Berkeley National Laboratory
One Cyclotron Road, MS: 50B-2239
Berkeley, CA 94720
(510) 486-7381 (Voice)
(510) 495-2998 (Fax)
BLTierney@lbl.gov

Co-Investigators

Deborah Agarwal	Lawrence Berkeley National Laboratory	DAAgarwal@lbl.gov
Vern Paxson	Lawrence Berkeley National Laboratory	VEPaxson@lbl.gov
Michael Collins	Lawrence Berkeley National Laboratory/ESNet	measurement@es.net
Ted Sopher	Lawrence Berkeley National Laboratory	TGSopher@lbl.gov
Eli Dart	Lawrence Berkeley National Laboratory	EDDart@lbl.gov

Table of Contents

Abstract

Achieving the goals of high performance distributed data access and computing will require wringing the best possible performance from networks. But finding the speed bumps in networks is a longstanding problem. Most existing tools for end-to-end tests of network performance provide the end user little or no information regarding the entire network path (local and wide area networks). Without information about a stream from intermediate hops within the network, the end-to-end system is often unable to identify and diagnose problems within the network. We propose to design and implement a self-configuring monitoring system that uses special request packets to automatically activate monitoring along the network path between communicating endpoints. These request packets will pass through sensors deployed at the Layer three ingress and egress routers of the ESnet network and within the end site networks. A principal design goal of the system is to provide components that are secure, easy to install, and easy to maintain so that the system does not add a burden to the network's administration. This architecture will not require modifications to the application, network routing, or forwarding infrastructure, nor is human intervention required once monitoring has been triggered. Archived monitoring data will help point the way beyond the handcrafted systems of network testbeds to a production environment that can routinely support high performance distributed applications. This passive monitoring system will integrate with active monitoring efforts and provide an essential component in a complete end-to-end network test and monitoring capability. It will complement the existing network operation efforts.

1 Narrative

1.1 Background and Significance

1.1.1 The Opportunity: Creating a Network Monitoring Infrastructure

Application developers currently have very few tools to aid in developing distributed applications that effectively utilize the network; the tools which do exist are generally accessible only to the network engineer and do not provide information regarding the entire network path (local and wide area networks). Without information about a stream from intermediate hops within the network, the end-to-end system is often unable to identify and diagnose problems within the network.. For a distributed application to fully utilize the network, it must first know the current network properties. This knowledge is particularly critical in the case of high-speed wide-area networks. Knowledge of current and maximum bandwidth, and of current and minimum latency, make it possible for the application to adapt to the network conditions by, for example, using the optimal TCP buffer size and the optimal number of parallel streams.

Comprehensive end-to-end and top-to-bottom monitoring is critical for developing and debugging high performance, distributed applications. However, this service is largely unavailable to the application developer except in testbed environments. Increasingly the approach of these applications is to rely on "automatic" tuning of transport parameters such as TCP window size, parallel streams, etc. [1], [2], [6]. However, the results of the tuning still must be verified, and sometimes debugged, both of which rely on fine-grained network monitoring. In addition, end-to-end approaches are limited in their ability to diagnose problems in the intervening networks and to diagnose the impact of tuning on other traffic in the network.

Protocols like UDP and IP multicast do not inherently contain congestion control mechanisms and thus are prone to problems related to congestion collapse within the network. Monitoring IP Multicast traffic introduces the additional difficulty that the traffic is routed to multiple destinations using a dynamic tree. The tree is transient and is only kept as state in the routers while there is traffic, so timing of the monitoring is very important. Furthermore, the tree is independently determined at the routers and the dynamic nature of this tree makes it difficult to identify and diagnose problems with the traffic. Finally, for all types of traffic, it can be invaluable to determine where along a network path problems such as loss arise, but this can be very difficult to do using purely endpoint techniques such as "traceroute", which rely on what are in practice very noisy measurements of the network's internal state.

Because the network is dynamic, its properties need to be continuously tracked so that the application can be informed of the changes in network characteristics during execution. Currently, information about the network is very difficult to obtain and must generally be manually collected on a per-usage basis. A reliable, ubiquitous service, similar to DNS, needs to be available to allow applications to query for network properties. To support this service, the infrastructure of the network needs to contain both active and passive monitoring capabilities. The passive monitoring capability provides a means of observing the traffic as it flows through the network and the active monitoring capability provides the means of generating that traffic and measuring the resulting end-to-end performance.

Coordinating these capabilities requires a monitoring framework that can activate and control the individual sensors while maintaining some notion of the current sensor activities system-wide. But, comprehensive network monitoring also introduces an opportunity for malicious network attacks, such as theft by gaining access to private traffic, denial-of-service by running massive amounts of network test traffic, or spoofing of sensor results to misdirect or redirect traffic. Therefore, the monitoring framework must be designed with security from the ground up.

All of these issues must be addressed in order to provide routine use of production networks for high performance distributed applications, and this is the motivation for the proposed work. This proposal addresses the need for a network monitoring infrastructure to support both active and passive network monitoring. The infrastructure aims to provide accurate, comprehensive, and on-demand, application-to-application monitoring capabilities throughout the interior of the interconnecting network domains.

1.1.2 The State of the Art: Network Monitoring Software

Network measurement sensors that can be deployed to aid in identifying network performance characteristics and diagnose problems can be classified as active, passive, or application sensors. *Active* sensors inject traffic into the network to measure the network response to the traffic. *Passive* sensors measure characteristics of the traffic passing through the sensor. *Application* sensors provide the mechanisms to obtain detail about application timings.

There are many active network sensors that have been developed for use in measuring network capabilities. `Ping` and `traceroute` are two of the earliest examples of active sensors which use ICMP packets. Active sensors like `pathchar`[37], `pchar`[38], and `pipechar`[39] use varying-size ICMP packets to probe the network in the direction of a destination end-point. They send ICMP packets probing each hop of the network path, attempting to identify the bottleneck link and its bandwidth. The `treno` [43] sensor uses ICMP packets sent to the destination site to simulate a TCP flow. It treats the ICMP replies as if they were TCP acknowledgments and uses the TCP flow control algorithms to estimate the bandwidth that would be achieved by a TCP connection. The `pathchar`, `pchar`, `pipechar`, and `treno` sensors are all run using a single end-point of a network path and do not require there to be a matching process at the destination end-point. The `iperf`[33] sensor uses a TCP connection to send data between two end-points and measures the actual TCP throughput achieved. The `iperf` interface provides access to parameters such as buffer size, duration of test, and number of parallel streams to use. The secure network performance measurement framework we have been developing at LBNL called `nettest` [22] provides a framework for launching active sensors securely.

Simple Network Management Protocol (SNMP), Cisco's NetFlow[36], and the Cooperative Association for Internet Data Analysis's (CAIDA) CoralReef [24] are tools generally available only to network administrators. SNMP provides an interface to traffic statistics, usually gathered by passive sensors from routers and end hosts but the requestor must have a community string or router access to get the SNMP data. ESnet engineers use the Multi Router Traffic Grapher (MRTG) to collect and display the SNMP results from the routers. NetFlow is a tool developed for Cisco routers to track flows and is generally used with `cflowd` [23] to analyze network flows. CoralReef provides current network traffic statistics as observed at a network location. Generally, providing access to tools like these which are used by the network administrators is

considered a security risk since they allow users to determine too much information about all users' traffic on the network. Also, detailed information from SNMP regarding the state at each router is likely to be misinterpreted by people not familiar with the current state of the network. This can lead to a significant amount of incorrect incident reporting and unnecessary work on the part of the network administrators.

Another passive network sensor is tcpdump, which can be used to record network traffic. Tcpanaly [7] can be programmed to interpret the network stream and diagnose various problems with the communication. The Bro [10] system is a passive network sensor that is primarily used to detect network-based intrusion attempts. It and CoralReef were specifically designed to operate on high-bandwidth streams (OC-48 in the case of CoralReef). Over the past several years CAIDA has engineered several other very useful and robust tools for probing and plotting the Internet. Tools such as `skitter` [41] and `geoplot` have been used to plot traffic and congestion by time and geographical location and have proven very useful in predicting trends of various protocols (FTP, HTTP, Napster).

Application sensors, such as the NetLogger Toolkit[14], provide the mechanisms to obtain detail about actual distributed application timings. NetLogger provides an instrumentation library designed for wide-area distributed applications that can record precision-timestamps associated with application-specific information at critical points in the application.

There are several current efforts to design and deploy a network monitoring system which allows high-bandwidth and critical areas of the network to be continuously monitored and tested, enabling hop-by-hop analysis of the network paths. Existing monitoring tools such as the Network Weather Service (NWS) [21] are typically deployed and configured in a very static manner, and provide no security mechanisms. The NIMI[7],[11] project is developing a software system for development of network measurement infrastructure for dedicated sensors. Active monitoring tools can be employed in conjunction with the passive monitoring in these systems to measure reference data streams. This methodology will also provide monitoring capabilities for UDP traffic.

The Web100 project [44] is focused on solving one specific aspect of the larger problem: end-to-end TCP performance. They are developing a kernel-level TCP implementation that 'tunes' the TCP buffer size parameter incrementally based on the current network behavior. This TCP implementation will also provide information on current status to the application. The Linux 2.4 development team is also developing an autotuning TCP implementation. There are also some excellent application tuning advice guides that are starting to appear[20].

1.1.3 Our Proposal: Creating a Self-Configuring Passive Network Monitoring System

The goals of this project are:
An infrastructure that is available to applications and securely enables routine monitoring through a network domain (ESnet and end sites in this case) in order to address:

- Debugging of distributed application communication
- Evaluation of application transport tuning strategies
- Evaluation of the impact on overall network performance of transport tuning strategies
- Support for both active and passive monitoring

- Support for both unicast and multicast traffic monitoring

A monitoring infrastructure that

- Is self-configuring with respect to monitor requests
- Introduces traffic only in the form of monitoring data and requests for monitoring
- Provides for access control on both monitoring requests and the data archives

Investigation of the Grid Forum monitoring architecture, both for data generated by the approach described here and data that is collected by other approaches

An authorization framework that is sufficiently flexible that various approaches can be implemented, used, and evaluated

A monitoring infrastructure that is general enough to be applicable to any network environment

1.2 Preliminary Studies

LBNL has extensive experience with developing and deploying passive monitoring, first with the development of the widely used tcpdump packet capture tool (and its associated library, libpcap, and kernel packet filter, BPF [7]), and more recently with the development of the Bro intrusion detection system[10]. Currently, 12 Bro monitors are operational on the LBNL, NERSC, ESnet, JGI, and ICSI networks. The monitors run 24x7 and monitor high-volume Gigabit Ethernet and Fast Ethernet links, generating attack alerts in real-time.

Application monitoring has also been an important aspect of LBNL work. The NetLogger Toolkit[14], which was developed in order to debug distributed applications, has been used extensively in several high-speed networking applications, including the DPSS [17], Radiance [4], BaBAR [16], Visapult [2], and GridFTP [1].

LBNL has been heavily involved in the Grid Forum (GF) and, of particular interest here, the GF Performance Working Group. LBNL co-authored the white paper for a monitoring data architecture called the Grid Monitoring Architecture (GMA), which will be discussed in detail later. The input into the GMA was drawn from experiences developing a system at LBNL called Java Agents for Monitoring and Management (JAMM) [15]. The JAMM agents, whose implementation is based on Java and Java Remote Method Invocation (RMI), can be used to launch a wide range of both active and passive system and network endpoint monitoring tools. The agents then extract, summarize, and publish the results.

Also in coordination with the Grid Forum Performance Working Group, LBNL has implemented a prototype of the GMA's proposed XML producer-consumer protocol [30]. Interoperability with a similar prototype developed by Warren Smith at NASA Ames has been established, and the two implementations are being used to explore and improve the protocol.

In partnership with the Pittsburgh Supercomputing Center, LBNL has designed and deployed NIMI, a large-scale (50 nodes) Internet measurement infrastructure [11]. NIMI is primarily used for active end-to-end measurements, but its architecture can also accommodate passive measurement. For the work proposed here, NIMI itself is too heavyweight a solution; we do not require the scheduling flexibility it provides because the deployed monitors will be performing a simple task (recording packets) on demand, not a set of complex measurement operations

coordinated in advance. But NIMI has provided valuable experiences in the management of a secure, large-scale distributed measurement infrastructure.

The members of the Distributed Systems Department at LBNL have a great deal of experience in building and tuning data-intensive, wide-area distributed applications. This experience has made us highly aware of functionality gaps in the currently available monitoring facilities. At the Supercomputing 2000 conference LBNL received the "Fastest and Fattest" Network Challenge award for the application that best utilized the wide-area network. The application was a remote data visualization application called Visapult [2], which used a peak rate of 1.5 Gbits/second, and a sustained data rate of 680Mbps. This performance was the result of a great deal of hand tuning of the SC2000 network. Visapult transforms data from a simulation using parallel compute nodes and transmits the transformed data in parallel over the network for rendering. The dataset, 80GB in size, was stored at LBNL and the compute cluster, an 8-processor SGI Origin with 4 Gigabit Ethernet interfaces, was in Dallas. Parallel reading of large datasets stored at a distant location is common in high energy physics (HEP) applications.

Debugging of network problems encountered during the SC2000 network challenge would have benefited significantly from information about the behavior of the data stream along the network path: we were only able to instrument the endpoints using NetLogger. In this challenge, using our access to the routers, we polled the SNMP counters on the three main routers every 5 seconds. This provided a rough estimate of the traffic characteristics, but neither the NetLogger instrumentation nor the SNMP polls provided a clear indication of the path being taken by the application traffic. This information is critical, because in a testbed network, much of the routing is hand-configured, and in the case of equipment failure, re-configured. We would first identify a serious problem as a drop in application performance (which could have many causes). Our only available mechanisms to identify and diagnose network problems was to frequently run traceroute, ping, and iperf to verify that the routes were correctly configured and that the hardware was working properly.

Monitoring of the actual application data path from LBNL to the Supercomputing show floor, if available, would have enabled us to immediately identify misconfigured routing tables. The symptom would have been packets veering off the high-speed testbed path, only to reappear later and limp in at some fraction of the testbed network's bandwidth to the show floor. Moreover, these results would have been more useful than the SNMP counters, and secure enough to be shared by all the participants in the challenge (we had to have root access to certain machines to get the SNMP statistics). They would not have interfered with the application's performance, so they could have been left running during the debugging and actual competitive runs of the application.

1.3 Research Design and Methods

1.3.1 Overview of Approach

The "self-configuring" monitoring system will use special "request" packets to automatically activate monitoring along the network path between communicating endpoints. The request packets pass through passive sensors that are deployed at the layer three ingress and egress routers

of the wide-area networks and at critical points in the end site networks. To activate monitoring, an endpoint of a data stream runs a program that sends (authenticated) request packets to the other endpoint. The goal of these packets is to alert each monitor in the interior of the network that the corresponding application flow is requesting monitoring from the network. The request packet contains authentication information, a description of the data to be monitored, and information about where to archive the data. When the monitors detect a request packet on the links to which they are attached, they authenticate the request, check authorization, and they activate recording of the headers of the corresponding application traffic.

Once activated, the monitors open a connection to a remote agent. The sensors will send to the agent a stream of monitoring data extracted from the packet flow. The agent will be an implementation of the Grid Forum [29] network monitoring management and archiving architecture [19], which will store the data or provide it to a third party in real time. Having been dynamically activated, the sensors then track the traffic without further intervention on the part of the requester and with no intervention on the part of the network operators. Thus, the architecture provides for a fluid low-maintenance internal monitoring capability: applications do not require modifications, nor does the network's routing or forwarding infrastructure, nor is human intervention required once monitoring has been triggered. The monitoring data is archived or returned to the endpoints by the agent, depending on parameters set in the request packet and the sensor's policy configuration.

We propose to deploy this system within the ESnet production network and at a few prototype end sites. This passive monitoring system will provide an essential component in a complete end-to-end network test and monitoring capability and will complement the existing network operation efforts.

The Self-Configuring Network Monitor system is designed to provide a passive monitoring capability in the network that is accessible to both network engineers and application developers. The monitoring will be performed by a number of passive sensors. Each sensor will be independently configured and installed in the network. Receipt of a request packet describing the characteristics of the traffic to be monitored will activate the sensor. Request packets will also contain authorization credentials and instructions for the disposition of the monitor data. A principal design goal of the system is to provide this capability with components that are secure, easy to install, and easy to maintain so that the system does not add a burden to the network's administration.

Active monitoring tools can be employed in conjunction with the passive monitoring to measure reference data streams. Examples of such tools are NIMI, NWS [21] and the secure network performance measurement tools we have been developing at LBNL [22].

The proposed methodology will also provide monitoring capabilities for UDP traffic. Ideally the monitoring network will be configured to monitor multicast traffic. The monitors on the multicast routing tree can be activated by periodically multicasting request packets. A representation of the multicast routing tree can be generated by running *mtrace* from the monitors to trace back to the source of the request packet.

One challenge is that high bandwidth links may prove difficult to effectively monitor without improvements to the current packet filtering approaches. One possible solution is to modify the Berkeley Packet Filter (BPF) to allow highly efficient hash-driven matching of multiple flows in the kernel of the monitoring system, rather than its current if-then-else matching.

1.3.2 Monitoring Activation Mechanism

The design will use detailed packet capture and filtering capabilities that have already been developed in the context of the Bro intrusion detection system [10]. These modules will detect request packets that are the activation mechanism to monitor a stream. A probe packet will contain authorization credentials, flow specification, destination address for results, and the nature and/or identity of the monitor data management agent. Having detected and decoded such a request packet, the monitor will activate the appropriate tracing program that will provide packet header data to the agent, which will in turn archive the data or provide it to a third party in real time. The request packet dynamically instantiates monitors along the network path. These monitors provide monitoring and delivery of monitor data for some (generally short) period of time. Unless another request packet is received within a timeout, the timer expires and monitoring ceases. This soft state process should require no other intervention on the part of the requester and no intervention on the part of the network operators.

There are a few candidate request mechanisms. One possible instantiation of the request mechanism is a modified ICMP ping type program that configures and activates the monitors along the network path between application endpoints. The advantage of using ICMP is that most systems will echo ICMP packets, and therefore automatically provide monitor triggering along both the forward and the return path. If the request packets have the same QoS characteristics as the flow, then they should be routed the same way that data packets are. Another potential mechanism is UDP packets, which may prove easier to align with a given flow's QoS characteristics, and less likely to suffer perturbation by the network due to effects such as ICMP rate-limiting.

Ideally the monitoring network will also be configured to monitor multicast traffic. One of the primary difficulties in monitoring multicast traffic will be activating the monitors on the multicast routing tree. The multicast routing tree is dynamically changing so a static activation of monitors would be ineffective. One solution is to configure the monitoring activation program to multicast request packets. Since the request packets are sent periodically, they would incrementally activate monitors on the tree. New branches of the routing tree will begin to be monitored with the next request packet. Pruned branches will discontinue monitoring once the request packet refresh timeout occurs. A representation of the multicast routing tree can be generated by running `mtrace` from the monitors to trace back to the source of the monitoring request. Combining the `mtrace` results can provide a picture of the tree and allow the monitoring results to be directly correlated with the routing tree.

We will create an executable that can activate the passive monitors so that applications do not need any modifications to use the monitor. The NetLogger library is already used by many applications to measure performance information. We also intend to incorporate our request mechanism into the NetLogger library as one of the function calls so that the Self-Configuring Network Monitor can be activated by an application using NetLogger.

1.3.3 Passive Sensors

There are several possible sensors that can be deployed to provide passive monitoring at the network sites. We intend to use a tcpdump-like packet capture program written using libpcap in order to both sense the measurement activation requests and to then record the associated traffic. The general design philosophy of the monitors is that their role is to record and deliver packet

headers, and not to perform analysis themselves, as often the analysis will require synthesizing a view of a transport flow based on the traffic recorded by multiple monitors. However, we anticipate that we may need to develop a data reduction agent to reduce the volume of the recorded traffic. For the off-monitor analysis, we will develop a tool based on tcpanaly [12] to provide real-time flow bandwidth and retransmission information. The flow analysis of the trace can then be used to detect the characteristics of and problems with the TCP streams.

1.3.4 Interaction with Active Sensors

Monitoring of effects of a particular stream on other network traffic will be accomplished through coordination with a network performance measurement tool that actively injects traffic into the network. Examples of such tools are NIMI and the secure network performance measurement tools we have been developing at LBNL [22]. This methodology will also provide monitoring capabilities for UDP traffic.

In addition to this proposal, there are several other SciDAC proposals with active monitoring components. We will integrate the work from any SciDAC funded active monitoring project, sharing monitoring infrastructure and data wherever possible. The use of the GMA monitoring architecture makes it easy to collect monitoring data from several tools in a common way. The combination of active monitoring tools such as NIMI, NWS and pingER, along with the passive monitoring tools proposed here will create an extremely powerful and flexible monitoring environment, capable of tracking down almost any sort of network problem.

Having active sensors co-located with passive sensors greatly enhances our ability to diagnose problems in the network. When a problem is detected by the passive monitoring tools, traffic can be generated using the active tools, allowing one to collect additional data to further study the problem. By having these monitors deployed at every router along the path, we can study only the section of network that seems to be having the problem.

1.3.5 Archiving and Monitoring Infrastructure

One purpose of this proposal is to move from the one-off, handcrafted systems of network testbeds to a production environment that can routinely support high performance distributed applications. One of the issues in this regard is that the network itself can be studied for its response to various transport optimization strategies. In order to do this in a systematic fashion, a historical perspective must be built up so that, for example, the impact of one strategy may be compared to the impact of some other strategy used in the past. To support this, results of both experiments and routine monitoring must be organized and archived for network performance / response analysis over a period of time. The Grid Monitoring Architecture provides an architecture for accomplishing this archiving.

Grid Monitoring Architecture

With the potential for thousands of resources at geographically different sites and tens of thousands of simultaneous Grid users, it is important for the data management and collection facilities to scale while, at the same time, protecting the data from spoiling. The Grid Monitoring Architecture (GMA) was designed to provide a framework for solving this problem.

In some models, such as the CORBA Event Service, all communication flows through a central component, which represents a potential bottleneck. In contrast, performance monitoring event data, which makes up the majority of the communication traffic, should travel directly from the

producers of the data to the consumers of the data. In this way, individual producer/consumer pairs can do "impedance matching" based on negotiated requirements, and the amount of data flowing through the system can be controlled in a precise and localized fashion based on current load considerations. The GMA design also allows for replication and reduction of event data at intermediate components acting as consumer/producer caches or filters. Use of these intermediate components lessens the load on producers of event data that is of interest to many consumers, with subsequent reductions in the network traffic, as the intermediaries can be placed "near" the data consumers. The directory service contains only metadata about the performance events and system components and is accessed relatively infrequently, reducing the chance that it would be a bottleneck.

Terminology

The monitoring data that the GMA is designed to handle are timestamped *events*. An event is a named structure that may contain one or more items of data that relate to one or more resources, e.g., memory usage, network usage, or "error" conditions such as a server process crashing. The *producer* is the component that makes the event data available. A *consumer* is any process that requests or accepts event data. A *directory service* is used to publish what event data is available and which producer to contact to get it.

Architecture

The GMA architecture supports both a producer/consumer model, similar to several existing Event Service systems such as the CORBA Event Service[25], and a query/response model. For either model, producers or consumers that accept connections publish their existence in a directory service. Consumers use the directory service to locate one or more producers generating the type of event data they are interested in. Each consumer then subscribes to or queries the matching producer(s) directly. Likewise, a producer may query the directory service to locate consumer(s) that accept and process event data in a given manner – for example, a consumer that archives event data for later analysis. Once the appropriate consumer is identified, the producer would connect to it directly and stream the event data – similar in behavior to when a consumer subscribes to a producer, but initiated by the producer. The connections between these three components (consumers, producers, and directory service) are shown in Figure 1.

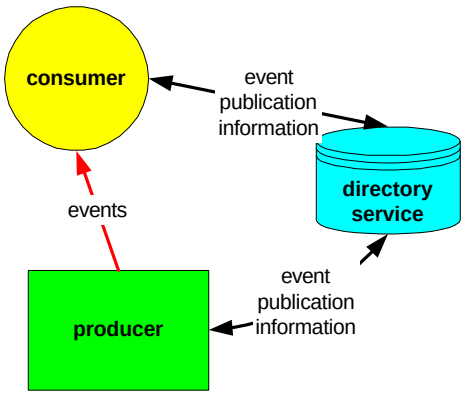

Figure 1. Grid Monitoring Architecture

By defining three interfaces: the consumer to producer interface, the consumer to directory service interface, and the producer to directory service interface; "standard" Grid monitoring services can be built which will all inter-operate. Examples of two services that could be implemented within this architecture, a consumer/producer pipe and an archiver, are described below.

Consumer/Producer Pipe

Producers and consumers may be arranged in any number of ways. One useful arrangement colocates the consumer and the producer in a single component, in effect creating a "pipe" (similar to the Unix pipe for process input and output) for the monitoring data that can aggregate data from several producers, disperse data to several consumers, or do both. Inside the channel, data can be modified, added, or removed. For example, a consumer/producer pipe might receive event data from several producers, use that data to generate a new derived event data type, and then make the derived events available to other consumers, as shown in Figure 2.

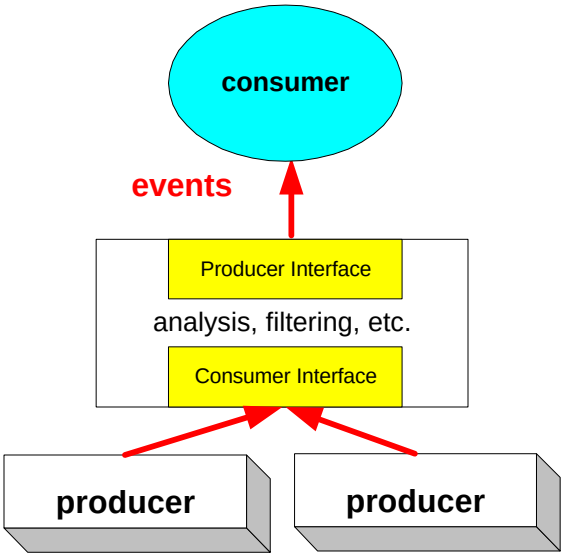

Figure 2. Combination Consumer/Producer Component

There are many other services that such a compound consumer/producer pipe might provide, such as event filtering and caching. For example, it could optionally perform any intermediate processing of the data the consumer might require. This component will be used to analyze the results of the passive network monitors, and to generate new events based on that analysis. This is a general technique, so multiple consumer/producer pipes may be connected to provide a modular pipeline for analysis of event data.

Archive Service

Monitoring data, as well as the network topology and configuration information, will need to be archived. An archival system can either be a consumer of performance event data, providing a repository for performance data, or a producer, providing the consumer with the requested event data, or both. The archive will provide the ability to correlate data over a large time period and across levels of the network infrastructure. Standard protocols for querying the archive will be developed in cooperation with the Grid Forum Performance working group [30].

In order to handle the volume of incoming data, archives may be distributed. This dovetails nicely with the GMA, which assumes that all producers and consumers of monitoring data are distributed and are centrally coordinated only through the information they place in the directory service. If the archives are distributed, queries will have to address multiple archives. Consumers can query multiple archives directly, but in order to simplify the task of making complex distributed queries, a consumer/producer pipe will be created to find all the relevant archives for a given query, send the query to each, and then sort and aggregate the results before returning them to the consumer.

1.3.6 Efficient Packet Filtering

We want to support concurrent measurement of a significant number (target: 100's) of flows because we do not want to limit the applicability of the infrastructure to the monitoring of large-scale distributed applications, nor to monitoring only one application at a time. Capturing packets for a large number of different flows at very high bandwidths is difficult with current packet filter technology. The main problem is that the packet filter cannot afford to test the packets against each possible flow in turn, which is how it is currently designed; it must instead match against all of the possible flows concurrently. We propose modifying BPF [7] to allow highly efficient matching of multiple flows in the kernel by adding a hash-based associative lookup operation to its virtual machine, along with corresponding libpcap mechanisms for accessing the mechanism and optimizing its use.

1.3.7 Security and Policy Control

Placing a system containing passive monitoring capabilities on a site DMZ will require establishing polices which satisfy several stakeholders. Identifiable stakeholders include ESnet, the Site and the Research teams. The policy issues that need to be addressed include:

?? Who will have access to the system and at what levels they will have access. The site staff may want access to monitor the site's traffic. ESnet staff may want access to resolve operational issues. The research teams obviously want access to use and tune their monitoring software.

?? Who will administer the system and install upgrades and patches.

?? Security and the basic host security configurations. The security policy for the system will need to satisfy all stakeholders. The overall security policy for the monitoring system will need to be specified. The location of the system in respect to a firewall will need to be determined. In some cases host security configuration may have to be very strong (when the box is on the network side of a security firewall).

?? What traffic the can the passive monitors collect. There may be prohibitions on monitoring traffic sourced by or destined to certain hosts, collections of hosts or subnets.

?? Who will have access to data collected by the components of the system and stored locally.

Monitoring of this type must address both privacy and denial-of-service vulnerabilities. It should not be possible for a data stream to be monitored except by someone who is associated with the applications, systems, networks, etc., of that stream. The monitoring system should not introduce vulnerabilities into the network, and should not itself be overly vulnerable. The need for

authorization does not necessarily stop with the activation of the monitors, the mechanisms used to disseminate the results must also address appropriate privacy vulnerabilities.

The monitor infrastructure will require a policy framework to express what sort of trigger requests the monitors will honor in terms of the type of traffic to monitor and where the results should be sent, and how these requests must be authenticated. It is important that from the initial design the framework allows (1) differing policies for different monitors, (2) secure, dynamic updates to policies, and (3) authorization and authentication that will work for a single trigger request transiting multiple policy domains.

1.3.8 Network Deployment

For such a monitor infrastructure to be feasible, it must be able to be deployed and operated in the production environment. This means that the installation, configuration and operation must be as easy and non-intrusive as possible, and that mis-configuration and mis-operation of the monitors should not be able to impact the network itself. Ideally, installing the monitoring system will not require any special configuration of the production data path, as it is desirable that the monitoring equipment itself not introduce any network artifacts.

The monitoring system described in this proposal can be deployed within any network domain to allow measurement of traffic traversing the network. In fact, if there are multiple networks in the path between the endpoints, ideally all of these networks will have monitors deployed within them. Since ESnet is the core network connecting the national laboratories, it has been identified as the initial deployment network for this monitoring system. We will deploy monitor boxes at critical nodes of ESnet with all software pre-installed. We will also deploy monitors at strategic locations within a few end-site networks as prototype deployments. This will allow for a complete end-to-end view of the path being tested.

Optical splitters will be used to connect the monitor boxes to links at ESnet egress points (Gigabit Ethernet and FDDI) and at end site installations. We will also investigate ATM tapping options as needed, realizing that the layer two to layer three conversion required to see the individual IP flows at ESnet hub aggregation points may be a substantial undertaking. A permanent monitoring archive(s) will be maintained by ESnet, which will store the monitoring data and make it accessible to authorized users in real time. Endpoints of individual tests can also directly receive monitoring data from those particular tests. Prototype end-site deployment will be in the LBNL and NERSC networks.

We believe that there are several projects that would like to deploy network measurement boxes at critical points in the network. Our Self-Configuring Network Monitor proposal has the same objective and would like to propose that we will deploy boxes that can also be used by the other projects. The boxes we intend to deploy will use the FreeBSD operating system and we intend to deploy these boxes at the ESnet ingress/egress points of the participating sites. We also intend to deploy boxes at critical points in the LBNL and NERSC networks. Our measurements will require optical splitters on the ESnet ingress/egress link and we also need a network interface for communicating with the box. The other active measurement proposals would be able to make use of this network interface to perform their measurements. An issue that will have to be addressed in aggregating these capabilities is responsibility for administering the system or components of the system. There will be several different sources of software including the OS, utilities and the active and passive monitoring systems. An additional issue that will have to be addressed for the

active monitors is, how much traffic may be injected as part of the active probes or tests. A site may have concerns that the traffic injected by an active monitor may have a negative impact on production traffic.

In order to perform meaningful analysis of the monitoring data, the clocks of all relevant hosts must be synchronized. This can be achieved using a tool which supports the Network Time Protocol (NTP) [8], such as the *xntpd* daemon. By installing a GPS-based NTP server on each subnet and running xntpd on each host, all the hosts' clocks can be synchronized to within about 0.25ms. We will ensure that all hosts that are part of the monitoring testbed will be running NTP.

1.3.9 Applications

We plan to leverage our ties to several large “Data Grid” [3] projects to provide a realistic test environment for the tools and techniques developed in this proposal. These projects include he Particle Physics Data Grid [40], GriPhyN [28], the EU DataGrid [5], and the Earth Systems Grid [26]. These projects all require the efficient transfer of very large scientific data files across the network, and would all benefit from the work described here.

The first three of these projects are all HEP applications. Experimental physics applications operate on and generate large amounts of data. For example, beginning in 2005, the Large Hadron Collider (LHC) at the European physics center CERN will produce several petabytes of raw and derived data per year for approximately 15 years. The consumers of experimental physics data will number in the hundreds or thousands. These users are distributed at many sites worldwide. Hence, it is often desirable to make copies or *replicas* of the data being analyzed to minimize access time and network load. Files will be replicated in a hierarchical manner, with all files stored at a central location (e.g., CERN) and decreasing subsets of the data stored at national and regional data centers, as described in [5]. The intensive networking demands of each of these projects will provide an idea environment for testing the advanced network services proposed here.

Another class of applications is climate modeling, as represented by the Earth Systems Grid project. Climate modeling research groups generate large (multi-terabyte) reference simulations at supercomputer centers. These data are typically released in stages to progressively larger communities: first the research collaboration that generated the data, then perhaps selected colleagues, and eventually the entire community. As in the physics application, climate modeling researchers find it convenient to create local copies of portions of the data. Therefore, the application has similar needs for managing copies of datasets at multiple locations, as well as for higher-level services such as replica selection or automatic replica creation. This project also has the networking requirements which provide an ideal environment for testing the advanced network services proposed here.

An important consideration in the design of the network monitoring infrastructure is the mechanisms for presentation of the capability and results to the application and end user. The LBNL JAMM project [16] has been building tools to provide the basic infrastructure for collecting monitoring data and making it available to applications. The SciDAC Net100 proposal, if funded, will build upon this work, and provide the ability to combine several sources of monitoring data in an "application-friendly" manner. This will allow applications to become network-aware, optimize their throughput, and adapt to changes in network conditions.

We will also work closely with the SciDAC proposed “DOE Science Grid” project, led by William Johnston, LBNL. This project proposes creating a multi-laboratory Collaboratory Pilot aimed at integrating, deploying, and supporting the persistent services needed for a scalable, robust, high-performance DOE Science Grid, thus creating the underpinnings for a DOE Science Grid Collaboratory Software Environment (DSG-CSE).

We will use our experience and expertise in tuning high-performance distributed applications to validate the techniques from this proposal. We will work closely with ESnet, LBL networking, and NERSC networking to deploy these new monitors. We also have close ties to NTON [35], and SuperNet [42] networks, and will work with each of them to deploy these new network tools and services in each of these environments. This will allow us to validate the utility of these tools and services in a real network environment.

1.3.10 NERSC Involvement

NERSC (National Energy Research Scientific Computing Center) is, first and foremost, a production facility. In working with both local and remote customers, NERSC staff experience the issues that the Self-Configuring Network Monitor would help address on a regular basis. These include performance issues for bulk data transfers, improving access to computational resources, and the continuing evolution of the NERSC network infrastructure to best anticipate our customers' future requirements.

There are many ways in which the NERSC facility would benefit from the Self-Configuring Network Monitor. Perhaps the most important of these is reducing the time required to tune our infrastructure to provide the best support for a customer's application. In many cases, this can be a long and involved process, requiring the co-ordination of busy staff members from multiple sites, as well as the use of production infrastructure for probing and experimentation as staff attempt to diagnose problems and propose solutions. If a tool such as the Self-Configuring Network Monitor were available, the information required to tune the NERSC infrastructure could be much more readily available, resulting not only in faster solutions to performance problems, but improved productivity for NERSC staff as well.

In addition to improving the solution process for network performance issues, the Monitors would benefit the NERSC staff by providing them with a tool for end-to-end network characterization that is superior to those currently available. Improvements in network monitoring services would have real-world benefits for the staff tasked with maintaining and growing a high performance production infrastructure such as NERSC. Note that hand-tuning of the NERSC infrastructure has resulted in performance increases for remote sites of a factor of 4 to 10 over the performance before the tuning was done. This underscores both the benefits of tuning the NERSC networks, and the potential benefits of the Monitor once it reaches production capability.

We envision a 3-stage process for implementing this project at NERSC. First, we will provide a framework and facilities at NERSC for initial development and testing of the Monitor in a way that makes sense for NERSC, and avoids adverse impacts on production systems. Second, after the Monitor has progressed to the point that it is stable and will not impact production operations, we will put it into use by NERSC staff for troubleshooting in real-world situations. Third, provided the Monitor develops to the point that it is safe, secure, robust and does not cause disruption of production services, we will make it available for use by NERSC customers.

1.3.11 Tasks and Milestones

Year 1:

- ?? Design and implement a secure sensor activation protocol
- ?? Design and implement passive sensor improvements
- ?? Research optimal strategies for archiving the monitoring data in a manner that is efficient for both storage and retrieval.
- ?? Deploy prototype passive monitoring system in ESnet using tcpdump, NetLogger for archiving and manual configuration mechanisms
- ?? Deploy active monitoring tools (nettest, NWS) at a set of sites
- ?? Research issues involved in building multicast monitoring tools
- ?? Implement timing synchronization mechanisms
- ?? Review security issues for systems in the NERSC domain

Year 2:

- ?? Deploy implementation of the Grid Monitoring Architecture for the collection and archiving of monitoring data
- ?? Deploy implementation of the secure sensor activation protocol
- ?? Deploy initial implementation of passive sensor improvements
- ?? Begin working with application developers running applications across ESnet
- ?? Extend tcpanaly tool to provide real-time flow monitoring
- ?? Expand number of passive monitors and active monitoring sites
- ?? Complete the monitoring data archive
- ?? Design and implement Berkeley Packet Filter (BPF) enhancement to improve handling of large numbers of flows
- ?? Collection of data from the NIMI, NWS, and pingER tools from the "Active Internet measurement for ESnet" (AIME) proposal in our monitoring data archive

Year 3:

- ?? Continue to work with Applications, and determine what type of monitoring is most beneficial to applications
- ?? Explore scalability issues, and enhance tools where necessary
- ?? Deploy multicast monitoring capabilities

1.3.12 Connections

In addition to this proposal, there are several other SciDAC proposals with active monitoring components. We will integrate the work from any SciDAC funded active monitoring project, sharing monitoring infrastructure and data wherever possible. The use of the GMA monitoring architecture makes it easy to collect monitoring data from several tools in a common way. The combination of active monitoring tools such as NIMI, NWS and pingER, as would be deployed by Les Cottrell's "Active Internet measurement for ESnet" (AIME) proposal, along with the passive monitoring tools proposed here will create an extremely powerful and flexible monitoring environment, capable of tracking down almost any sort of network problem.

2 Literature Cited

[1] W. Allcock, J. Bester, J. Bresnahan, et. al., “Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing”, http://www.globus.org/

[2] Bethel, W., B. Tierney, J. Lee, D. Gunter, S. Lau, “Using High-Speed WANs and Network Data Caches to Enable Remote and Distributed Visualization”, Proceeding of the IEEE Supercomputing 2000 Conference, Nov. 2000.

[3] Chervenak, A., Foster, I., Kesselman, C., Salisbury, C. and Tuecke, S. “The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets”. Journal of Network and Computer Applications, 2000.

[4] D. Gunter, B. Tierney, B. Crowley, M. Holding, J. Lee NetLogger: A Toolkit for Distributed System Performance Analysis Proceedings of the IEEE Mascots 2000 Conference (Mascots 2000), August 2000, LBNL-46269.

[5] W. Hoschek, Jaen-Martinez, J., Samar, A., Stockinger, H. and Stockinger, K., Data Management in an International Data Grid Project. In Proc. 1st IEEE/ACM International Workshop on Grid Computing, 2000, Springer Verlag Press. Bangalore, India, December 2000.

[6] Lee, J., D. Gunter, B. Tierney, W. Allock, J. Bester, J. Bresnahan, S. Tuecke, “Applied Techniques for High Bandwidth Data Transfers across Wide Area Networks”, Dec. 2000, LBNL-46269. http://www-didc.lbl.gov/publications.html

[7] S. McCanne and V. Jacobson, "The BSD Packet Filter: A New Architecture for User-level Packet Capture," Proc. Winter USENIX Technical Conference, Jan. 1993.

[8] D. Mills. Simple Network Time Protocol (SNTP). RFC 1769, March 1995

[9] V. Paxson, J. Mahdavi, A. Adams, M. Mathis, “An Architecture for Large-Scale Internet Measurement,” IEEE Communications, Vol. 36, No.8, pp 48-54, August 1998. (ftp://ftp.ee.lbl.gov/papers/nimi-ieee-comm98.ps.gz)

[10] V. Paxson, “Bro: A System for Detecting Network Intruders in Real-Time,” Proceedings of the 7th USENIX Security Symposium, San Antonio, TX, January 1998.

[11] V. Paxson, A. Adams, M. Mathis, “Experiences with NIMI,” to appear in Proceedings of Passive & Active Measurement: PAM-2000.

[12] V. Paxson, “Automated Packet Trace Analysis of TCP Implementations,” ACM SIGCOMM '97, September 1997, Cannes, France.

[13] J. Semke, J. Mahdavi and M. Mathis, "Automatic TCP Buffer Tuning," Proc. ACM SIGCOMM, Oct. 1998.

[14] B. Tierney, W. Johnston, B. Crowley, G. Hoo, C. Brooks, D. Gunter, “The NetLogger Methodology for High Performance Distributed Systems Performance Analysis”, Proceeding of IEEE High Performance Distributed Computing conference, July 1998, LBNL-42611. http://www-didc.lbl.gov/NetLogger/

[15] B. Tierney, W. Johnston, J. Lee, M. Thompson, "A Data Intensive Distributed Computing Architecture for Grid Applications", Future Generation Computer Systems (an Elsevier Journal), vol 16, #5, April 2000, pp 473-481.

[16] B. Tierney, B. Crowley, D. Gunter, J. Lee, M. Thompson, "A Monitoring Sensor Management System for Grid Environments", Cluster Computing Journal, vol 4-1, 2001, Baltzer Science Publications

[17] Tierney, B., J. Lee, B. Crowley, M. Holding, J. Hylton, and F. Drake, "A Network-Aware Distributed Storage Cache for Data Intensive Environments", *Proceedings of IEEE High Performance Distributed Computing conference* (HPDC-8), August 1999, LBNL-42896.

[18] B. Tierney, D. Gunter, J. Becla,, B. Jacobsen, and D. Quarrie, "Using NetLogger for Distributed Systems Performance Analysis of the BaBar Data Analysis System", Proceedings of Computers in High Energy Physics 2000 (CHEP 2000), Feb. 2000, LBNL-44828.

[19] B. Tierney, R. Wolski, R. Aydt, V. Taylor, et. al.,“A Grid Monitoring Service Architecture”, Global Grid Forum White Paper, http://www-didc.lbl.gov/GridPerf/papers/GMA.pdf.

[20] B. Tierney, “TCP Tuning Guide for Distributed Application on Wide Area Networks”, Usenix ;login, Feb. 2001. (http://www-didc.lbl.gov/tcp-wan.html).

[21] R. Wolski, N. Spring, J. Hayes, “The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing,” Future Generation Computing Systems, 1999. http://nws.npaci.edu/NWS/

[22] Advanced Visualization Communication Toolkit (http://www-itg.lbl.gov/~deba/NGI/AdvVizComm.html) and Methods for Network Analysis and Troubleshooting (http://www-didc.lbl.gov/~jin/network/net-tools.html), Lawrence Berkeley National Laboratory.

[23] Cflowd: http://www.caida.org/tools/measurement/cflowd/

[24] CoralReef: http://www.caida.org/tools/measurement/coralreef/

[25] CORBA, “Systems Management: Event Management Service”, X/Open Document Number: P437, http://www.opengroup.org/onlinepubs/008356299/

[26] Earth Systems Grid Project: http://www.scd.ucar.edu/css/esg/

[27] EU DataGrid: http://www.eu-datagrid.org/

[28] GriPhyN Project: http://www.griphyn.org/

[29] The Grid Forum (http://www.gridforum.org/) is a consortium of institutions and individuals working on wide area computing and computational Grids

[30] Grid Forum “Grid Performance” Working Group: http://www-didc.lbl.gov/GridPerf

[31] Grid Monitoring Architecture prototype in Java™: http://www-didc.lbl.gov/prod-cons/

[32] Enable Project: http://www-didc.lbl.gov/ENABLE/

[33] Iperf: http://dast.nlanr.net/Projects/Iperf/index.html

[34] Multi Router Traffic Grapher: http://ee-staff.ethz.ch/~oetiker/webtools/mrtg/

[35] National Transparent Optical Network (NTON); http://www.ntonc.org/

[36] NetFlow whitepaper: http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm

[37] Pathchar – Van Jacobson, ftp://ftp.ee.lbl.gov/

[38] Pchar: http://www.employees.org/~bmah/Software/pchar/

[39] Pipechar – Guojun Jin: http://www-didc.lbl.gov/~jin

[40] Particle Physics Data Grid: http://www.ppdg.net/

[41] Skitter: http://www.caida.org/tools/measurement/skitter/

[42] SuperNet Network Testbed Projects: http://www.ngi-supernet.org/

[43] Treno: http://www.psc.edu/networking/treno_info.html

[44] The WEB100 Project, Facilitating Effective and Transparent Network Use, http://www.web100.org/

3 Description of Facilities and Resources

The Distributed Systems Department (DSD) is a computer science research department of the NERSC Division of Lawrence Berkeley National Laboratory. NERSC (National Energy Research Scientific Computing Center) is the foremost resource for non-classified large-scale computation within DOE. Within the NERSC division, the Distributed Systems Department focuses on research to allow scientists to address complex and large-scale computing and data analysis problems beyond what is possible today. A current focus of DSD is to expand the capabilities of the DOE Science Grid. The DOE Science Grid's major objective is to provide the advanced distributed computing infrastructure based on Grid middleware and tools to enable the degree of scalability in scientific computing necessary for DOE to accomplish its missions in science.

DSD does its development work on Solaris, FreeBSD, and Linux. It runs a Netscape Certificate Authority and LDAP Directory Server to support the authentication needs of Grid users, and to support it’s security and directory services R&D.

ESNet has equipment space at all of the DOE Labs for the ESNet egress routers, and this is where the monitor systems will be deployed.