An Architecture for Monitoring, Visualization and Control of Gigabit Networks

Guru Parulkar, Douglas Schmidt, Eileen Kraemer, Jonathan Turner, Anshul Kantawala
{guru,schmidt,eileen,jst}@cs.wustl.edu, anshul@arl.wustl.edu
Department of Computer Science
Washington University
St. Louis Mo. 63130, USA
Ph: 314-935-7534, Fax: 314-935-7302

August 8, 1997

Abstract

In this paper, we outline a design of a highly scalable network monitoring, visualization and control system (NMVC) system with advanced algorithmic and human-in-the-loop capability. This capability allows network administrators to calibrate and fine-tune network and application parameters in real-time according to observed traffic patterns. The goal of the NMVC system is to ensure adequate quality of service to network users, while maintaining high network resource utilization. The main components of our system are: a **network probe** and an **endsystem probe** which can probe gigabit/s links, **software network management agents** that provide extensible multi-attribute event filtering for highly scalable data/event collection, efficient online event ordering algorithms that can help synthesize and display a consistent view of network health, status and performance and a **View Choreographer** that allows management applications and administrators to specify the mapping of network events to higher-level events and to visualization objects and updates.

1 Introduction

Computer networks (such as the Internet and the Global Information Infrastructure) have become critical for education, research, business, and most importantly military operations. Recent advances in network infrastructure technology (such as ATM and IPv6) have enabled the development of high performance local area and wide area networks. Efficient management of these networks is essential. However, existing algorithmic methods for managing networks have not matured to the point where performance bottleneck and fault detection, isolation, correlation, and correction can be automated scalably [1]. Thus, it is crucial to build efficient and user-friendly network monitoring, visualization and control (NMVC) systems.

After studying several commercially available SNMP and CMIP based network monitoring and management systems/tools, we find that most are not suitable for the monitoring, control and visualization of large high-speed ATM and other networks that are the target of this project. While these tools provide basic features like network discovery and MIB browsing, they have many weaknesses. Most of these tools allow only *status polling*, which helps detect network failures. However, failure detection is a small component of today's network management. There are some tools that do both status and performance polling. However, they keep these two separate, and performance polling is done more often than status polling. This means that its takes longer to detect failures. Also, scalability is a major weakness of these tools. That is, they will not work or work well with large networks and with high speed networks [2, 3]. For example, most tools redundantly poll devices that are part of more than one network map. This generates excessive management traffic when administering large-scale MAN and WAN networks. For some of the tools, notably HP OpenView [4] and IBM NetView/6000 [5], the architecture is geared towards managing discrete entities with no knowledge of the overall network environment.

We propose a NMVC system that ensures adequate quality of service to network users while maintaining high network resource utilization. The main components of our system are: a **network probe**, an **endsystem probe**, **software network management agents**, **network operation centers** (NOCs), efficient online event ordering algorithms and a **View Choreographer**.

The network probe will be built using the ATM Port Interconnect Controller (APIC) chip and a CPU-memory module. The APIC-based probe meets all the requirements for a gigabit link probe. With two full duplex 1.2 Gb/s ATM ports, the APIC can be easily inserted in a link as a probe for packet/cell "snooping" to log traffic measurements without interfering with network traffic. Software network management agents in our proposed system are built atop network probes and used to track event flows, as well as classify and report events of interest to NOCs. We propose using a highly flexible, scalable and high performance event-filtering mechanism for the software network management agents that will effectively eliminate redundant management traffic using a dynamic trie-based filter fusion technique. The **View Choreographer** uses semantically-guided filtering algorithms to enable low latency updates of displays that present an accurate real-time snapshot of current network state.

Furthermore, the system's feedback control mechanisms will provide support for **network configuration management**, **ATM virtual circuit management**, **router-to-router link management** and **application-level congestion management**. The system will be experimentally evaluated by building a 3-5 node testbed and using a suite of multimedia traffic generator tools.

The rest of the paper is organised as follows. In Section 2 we enumerate a few motivating examples for the design of a gigabit NMVC system and describe the feedback capabilities needed for such a system. Section 3 gives a brief description of the overall architecture of the NMVC system. In Section 4 we describe the functions that a gigabit monitoring system must support and the design of our network probe and endsystem probe which meets these requirements. In Section 5 we present our software management agents which are responsible for mediating NOC access to managed objects within management information bases (MIBS). Section 6 describes the design of the visualization tool which uses the approach of distributed visualization using graphical animation. Section 7 describes our experimental testbed and Section 8 concludes the paper.

2 Feedback Control Capabilities

In high performance networks, it is often necessary to make control decisions on the basis of incomplete or imprecise information, resulting in sub-optimal network performance. To improve network performance, it is necessary to adapt various network control parameters in response to observed traffic behavior. To be most effective, such adaptation must be done at all levels, including the global network level, where network management personnel can observe patterns and react to them in real-time, and then establish policies that can be implemented automatically when specified circumstances arise.

The data gathering and visualization mechanisms in the proposed NMVC system allow administrators to identify such situations and either make immediate adjustments to the appropriate control parameters or specify high level policies that allow the system to adjust its control parameters as local circumstances or dynamically changing conditions require. Similar situations arise at various levels of control; for example, ATM admission control algorithms include parameters that affect virtual circuit admission and routing decisions. Similarly, fair queueing algorithms and ABR flow control mechanisms include a variety of control parameters that can materially affect the efficiency with which network resources are used and the performance experienced by end users and should be made available for adjustment by administrators. At higher levels, flow and packet routing algorithms provide opportunities for administrators to tune network performance to suit specific circumstances.

Some of the key capabilities that the feedback control mechanisms support are described below.

Network Configuration Management. Consider a network being deployed to provide support for a large-scale military operation. Such networks can be very complex, supporting thousands of computer systems, distributed over substantial geographic areas and requiring tens or even hundreds of switches, routers and interconnecting links (which may employ fiber optics, radio or satellites). Because of the time pressures associated with military operations, the ability to rapidly deploy such networks and quickly bring them on-line is crucial. Making rapid deployment work requires advance planning, trained personnel that can implement the network deployment plan and the ability to adjust the plan to meet local circumstances. Network configuration management capabilities are crucial to facilitating rapid deployment, detecting inconsistencies in actual and planned configurations, and for detecting failures. To facilitate rapid deployment of networks, we propose to provide mechanisms to automatically build a network configuration database from the state of network hardware and software, compare this to a stored *network deployment plan* and report variances to network managers, allowing them to determine what action should be taken.

ATM Virtual Circuit Management. In most bursty data applications, the traffic characteristics are highly unpredictable, forcing admission control and virtual circuit routing mechanisms to rely on approximate information since it is not possible to predict the traffic flow or accurately model the link capacity available at any instant. Through the use of appropriate measurements and control parameters, network managers can tune the performance of these mechanisms to adapt them to a specific network configuration or application mix. One key parameter for ATM admission control and routing of bursty data applications is the *overbooking ratio*. This is the sum of the peak rates of the virtual circuits using a particular link divided by the link capacity. In the absence of information about the average rate of virtual circuits, the overbooking ratio can be used for admission control and routing. The simplest way to use the overbooking ratio is through a hard limit; for example a network manager might specify a default overbooking ratio of at most five. By keeping the overbooking ratio low, the network manager can limit the likelihood that a link will become congested. New virtual circuit requests that would exceed the overbooking ratio on a link will be diverted to alternate paths or, possibly blocked. Allowing a higher overbooking ratio reduces the chance of blocking but increases the potential for congestion.

The appropriate choice of overbooking ratio depends on a variety of factors. On a gigabit link, a higher overbooking ratio may be permitted, since the inherent statistical multiplexing advantage of higher bandwidth links allows higher average utilization for a given level of performance. On the other hand, a network manager may also choose to specify a higher than normal overbooking ratio on a low bandwidth bottleneck link because there are no alternate routes available and it is deemed better to allow the connections needing that link to adjust their data rates in response to congestion than to block connections. Dynamic adjustment of the overbooking ratio can also be appropriate. In particular, by observing the actual cell traffic on a link and the frequency with which connection requests for that link are refused, the system could determine that the current admission control policy for the link is too conservative or too liberal and adjust the allowed overbooking ratio accordingly. Such an adjustment could be done automatically under control of a specification provided by the network manager, or could be done interactively, with the system simply detecting the potential need for an adjustment and allowing the network manager to make the actual decision.

Virtual circuit routing algorithms also have parameters that can affect their performance in significant ways and that can be useful to network managers. In particular, it is common in routing algorithms to limit the length of a path used to connect any two points in the network, relative to a shortest path. That is, if the network topology includes a k-hop path between two endpoints, the routing algorithm may consider only paths of length $\leq k+2$ when routing between those two endpoints. Such constraints help prevent overloads at the virtual circuit level. By blocking virtual circuits that consume an excessive amount of resources, one preserves those resources for use by other virtual circuits that can use them more efficiently. At the same time, such constraints may limit traffic excessively in certain situations, making it appropriate to give network managers the opportunity to adjust them in response to local conditions.

Router-to-Router Link Management. IP and ATM are often used together, with ATM switches and links providing a substrate on which an IP network is overlaid. Permanent virtual circuits with fixed bandwidth are established between IP routers and from routers to end systems. To enable IP level resource management to work effectively, peak rate allocation can be used for these PVCs and priority mechanisms used in the ATM switches to isolate their traffic. The existence of an ATM substrate allows the bandwidth of these PVCs to be adjusted dynamically. Implementing dynamic bandwidth adjustment requires observations about traffic and congestion conditions on these PVCs and the knowledge of how PVCs are mapped onto ATM links so that if bandwidth on some PVCs must be decreased to accommodate an increase on other PVCs, this can be done with knowledge of the overall traffic distribution.

Gathering the information needed for dynamic bandwidth distribution can be done at either the ATM level or the IP level. IP routers can report average PVC utilization and congestion and queue length information. Per virtual circuit cell counters within the ATM switches can be monitored to measure traffic flows on the given PVCs. Given this information, a network manager can specify that the bandwidth allocation on some PVCs be adjusted to accommodate the current traffic flows as shown in Figure 6. This can be done interactively, with the support of a display that provides a high level view of the inter-router traffic, or can be done algorithmically, using specifications provided by the network manager. The problem of assigning bandwidths to PVCs with known traffic requirements so as to minimize the unsatisfied demand can be solved using linear programming. However, the dynamic adjustment problem is complicated by the fact that demands keep changing and are not completely known (a congested PVC doesn't really tell you what the traffic demand is for that PVC, only that the demand is not being satisfied).

Once it has been determined that adjustments to PVC bandwidths are called for, the NMVC system must implement those adjustments by interacting with the ATM network control software and the affected routers, since they must adjust the rates at which they send on those virtual circuits. This will typically involve modification to a hardware pacing parameter in one of the router's network interface cards.

Of course, one can also envision the establishment of new PVCs between previously unconnected routers in response to more traffic between those routers than originally anticipated. This can reduce load on possibly congested transit routers, but may require allocation of bandwidth to the new PVC, possibly requiring adjustment to the bandwidth of other PVCs. Effecting this kind of a change requires interaction with both the ATM network control and the router configuration control and routing tables.

Application Level Congestion Management. Multimedia applications can provide novel ways of coping with network congestion. In particular, video traffic can be coded in multiple *layers* with the lowest layer providing a usable signal and upper layers providing increased resolution and image quality. Layered video can be handled in an ATM network by using different virtual circuits for the different layers. In a multicast environment, a different subset of layers may be delivered to different users, based on the presence of congestion in the network or variations in the different end systems' video capabilities.

The inherently high data rate associated with video applications and the ability to view video usefully at different resolutions makes control of video layers a potentially powerful instrument for coping with peak traffic demands. To implement this approach, we need mechanisms to detect congestion and to exert control. Congestion may be signalled by an unacceptably high cell loss rate or AAL 5 frame loss rate on a given ATM link. Based on such indications, virtual circuits carrying higher level coding information can be turned off, either at the local switch or at the source. To enable such control, the ATM signaling software must provide a way for the user to designate selected virtual circuits as non-essential, or the network management system must have side information available to it that it can use to determine which virtual circuits can be turned off.

3 Overall Monitoring, Visualization and Control System

The NMVC system as shown in Figure 1 is composed of:

- **Network Probe**
  The network probe will be built using an ATM Port Interconnect Controller (APIC) Chip and a CPU-memory module. With two full duplex 1.2 Gbps ATM ports, the APIC can be

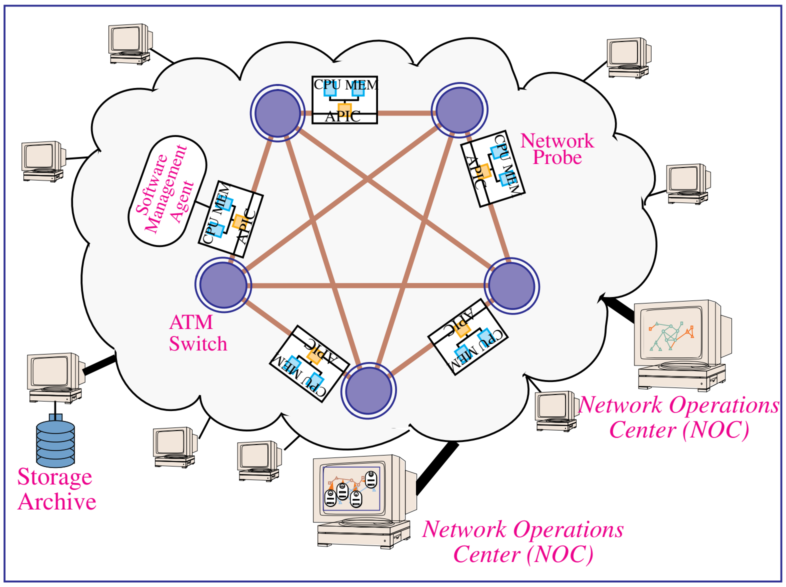

Figure 1: Topology of the Proposed Gigabit Network Monitoring, Visualization and Control System

easily inserted in a link and can thus, efficiently "snoop" traffic as the ATM cells move from the input to the output port. The APIC can sustain an aggregate bi-directional data rate of 2.4 Gbps.

- **Endsystem Probe**
  The endsystem probe comprises protocol probe functions that are embedded within the main data processing path of the protocol. Once activated, a probe function is called for each packet, and if the packet belongs to a connection or flow of interest, selected parameters of the packet and of the connection state are retrieved and written to a kernel trace log, which is a circular buffer maintained in wired down memory.

- **Software Management Agents**
  Software network management agents in our proposed system are built atop network probes and used to track event flows, as well as classify and report events of interest to NOCs. Common events reported to NOCs include *alarms* (such as delay or packet loss thresholds being exceeded for a service class), *quality of service statistics* such as the performance of particular IPv6 packet flows, and ATM connection blocking rates. When these types of events are detected by the agent, it notifies the NOC via a trap and/or performs a corresponding local action.

- **Network Operation Centers (NOCs)**
  NOCs (Figure 2) can remotely install and (re)configure the software management agents. In addition, NOCs are responsible for collecting events, ordering them correctly, synthesizing multiple coherent views of network operation, and providing the human-in-the-loop capability.

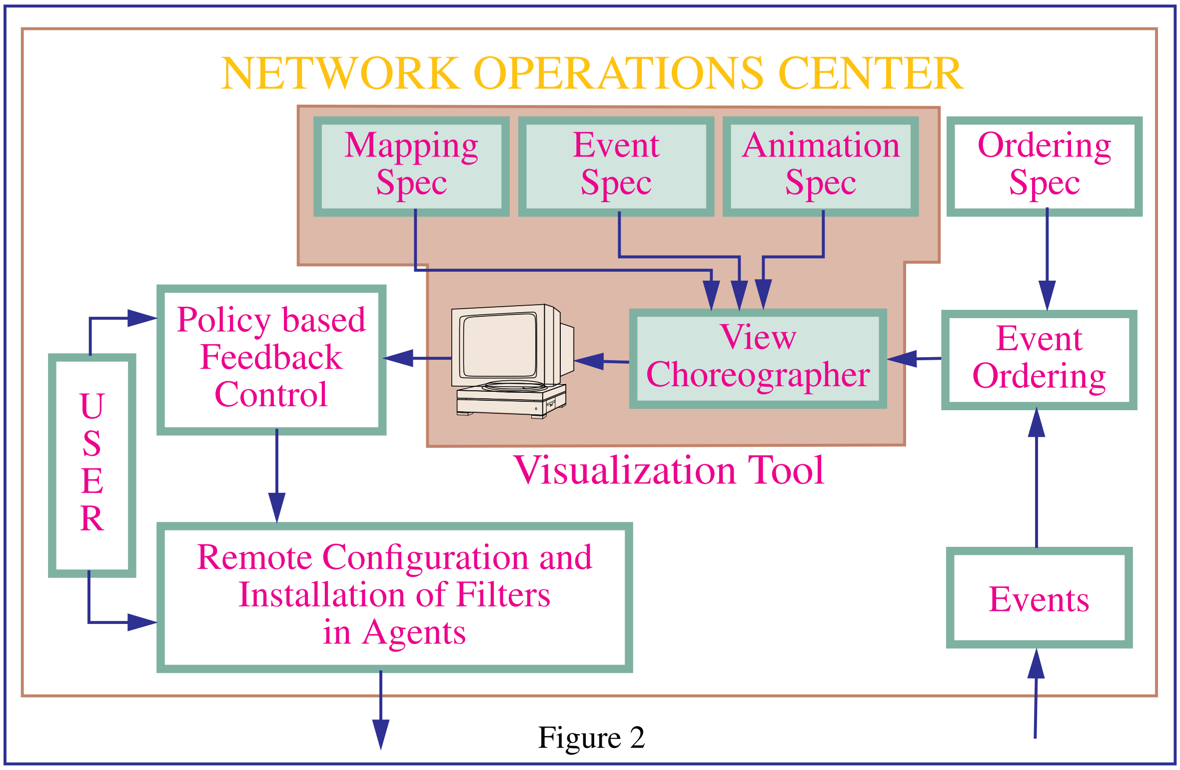

Figure 2: Network Operations Center

- **Network Visualization Tool**
  The visualization tool, represented by the shaded area in Figure 2, consists of a display component, and a mapping component. The display component in our NMVC system will be prototyped using a version of POLKA [6], which is an animation toolkit designed to support animations of concurrent systems. However, since POLKA is too slow to handle the frequency and volume of events generated by a gigabit network, we are developing a more highly optimized tool for network visualization which retains POLKA's features for animation of concurrent systems. The mapping component, called the **View Choreographer** in the NMVC system, performs user-specified mappings from network events to both higher-level events and to changes in graphical views. These higher-level events may also be mapped to display updates.

4 Network and Endsystem Probes

One of the key challenges in realizing a gigabit NMVC system is the ability to "probe" various components of high-speed networks and endsystems with minimum interference. Our goal is to provide a comprehensive infrastructure with tools to monitor network behavior at all levels (i.e., ATM-to-application) and allow users and network operators to see protocol interactions as they happen. We believe that this will yield many interesting insights and opportunities for system-level optimizations. In the following section we briefly discuss the requirements of a

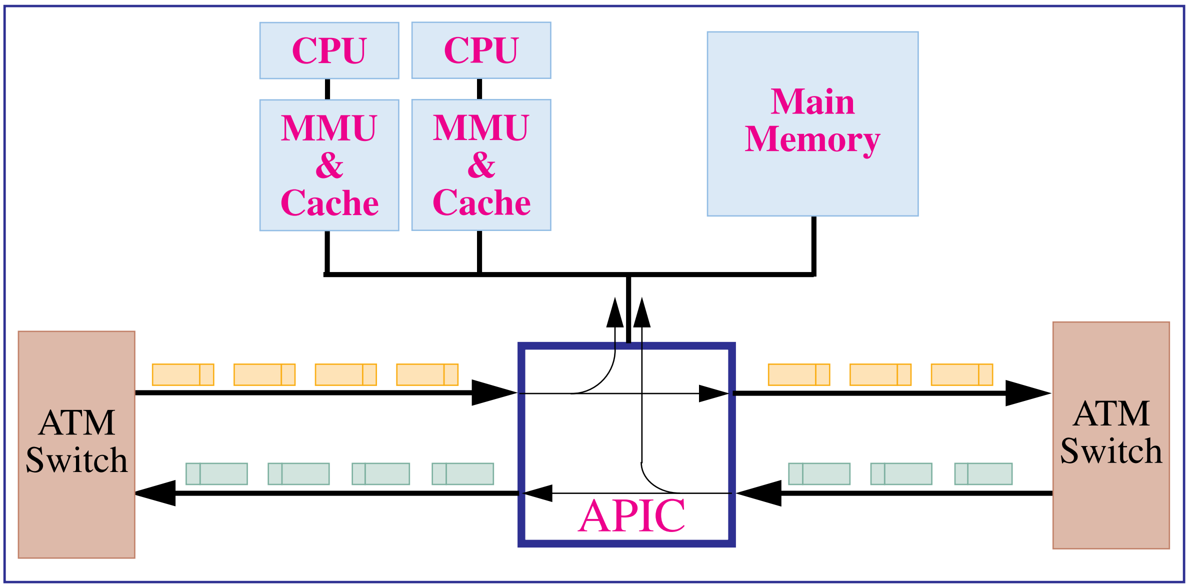

Figure 3: A High Performance Network Probe using APIC

network probe for gigabit links and our design which meets these requirements.

4.1 Network Probe

A gigabit per second (Gb/s) data rate has several implications on the design of a network probe, which is responsible for traffic measurements at the lowest level. For example, the network probe must be able to extract and record necessary packet header information (or ATM cell counts) for packets being received at a Gb/s rate, and be able to deal with thousands of packet flows or ATM connections on a given physical link. Interactive steering or feedback control requires that the network probe be very responsive and be able to change what it logs for particular packet flows in response to requests from the controlling network management agent. Finally, it is essential that the network probe not interfere with actual traffic while logging traffic data.

The APIC-based probe meets all the requirements mentioned above. In our prototype implementation (Figure 3), each APIC supports two full duplex 1.2 Gb/s links. Thus, the APIC can be easily inserted in a link as a probe for packet/cell snooping to log traffic measurements [7, 8, 9]. Moreover, the APIC's external memory/bus interface has been designed to deliver very high throughput and very low latency to applications [7]. We have performed extensive VHDL simulations of our design and the results verify that APIC can deliver the expected performance. The simulation tests show that APIC has a full-duplex throughput of 816Mb/s (1.18 Gb/s half-duplex) on PCI-64 and 456Mb/s (900 Mb/s half-duplex) on PCI-32 machines. Note that when the APIC is snooping on native ATM connections, it needs to only log the frame/packet header information, cell counts, and corrupted or lost frames in memory and does not need to bring the entire frame into memory. This leads to significant savings in the memory bandwidth requirements.

Key mechanisms of APIC that provide the necessary functionality and performance for gi-

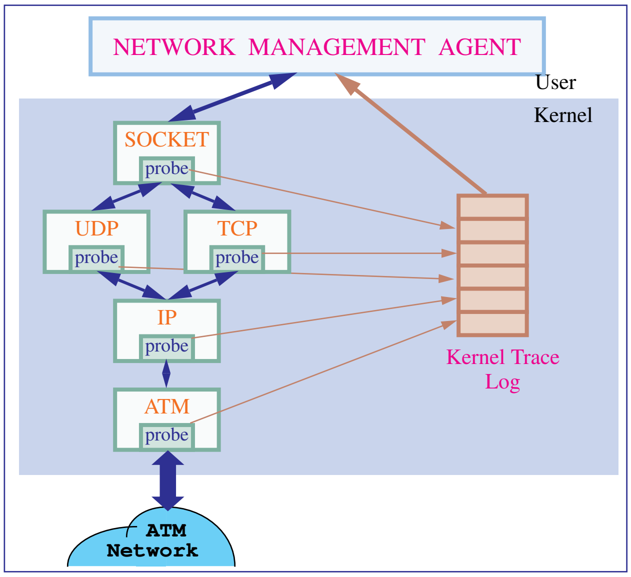

Figure 4: Probes for Kernel Resident Protocols

gabit data rates are described below:

- The *Protected I/O* mechanism allows applications residing in user space programmed I/O access to memory-mapped registers on the APIC, with specific access permissions to individual registers specifiable by the OS kernel. The *Protected* DMA mechanism allows user applications to directly queue buffers to the APIC for sending or receiving without kernel involvement. Used together, these two mechanisms allow data transfers to occur directly from user space to the APIC without system calls or any other form of kernel intervention, and without compromising the security and protection mechanisms of the operating system. This translates to greatly improved throughput and latency characteristics for end applications [10].

- The *Pool* DMA *with Packet Splitting* mechanism enables construction of zero-copy protocol stacks, using page remapping to avoid data copying.

- *Interrupt Demultiplexing* is a technique used by the APIC to significantly reduce the frequency of interrupts to the host processor. This technique exploits the connection-oriented nature of ATM to differentiate interrupts associated with distinct flows, and to mask out interrupts corresponding to flows that have already been queued for service.

Related Work

The ATTILA [11] project tries to measure and monitor traffic on an ATM/SONET link using a network probe. However, a SUN SparcStation10 is required for each network probe, and the data collected is at the cell/link level only. Further, no attempt is made to get higher level statistics such as end-to-end throughput or Quality of Service for different TCP connections.

OC3MON [12] can be inserted into a OC-3 network as a probe without any disruption to network traffic through the use of a splitter. It also has the added benefit of logging only header cells thus leading to significant memory bandwidth savings. However, it can currently operate only at OC-3 speeds (155 Mb/s) which is significantly slower than our proposed APIC probe which can probe gigabit speed links. An additional disadvantage of the design is that it needs a splitter to be able to snoop without disrupting network traffic. As link speeds increase, we expect the cost of such a splitter to increase thus driving up the cost of the probe.

4.2 Endsystem Probe

Another objective of the NMVC system is to allow a network provider or a network user to ensure that various classes of applications receive their promised service and to respond to changing traffic patterns by tuning protocol control knobs at various layers of protocols. Such end-to-end network management capabilities require that not only network nodes but also endsystems participate in the measurement and control of the network. We propose using software probes in protocols to make timing measurements and to retrieve the state of execution of a protocol stack. The reference configuration of an endsystem probe is as shown in Figure 4.

The endsystem probe will be built using a set of protocol probe functions embedded in kernel resident protocols such as TCP, UDP, IP, and ATM, and will be controlled by network management agents (Figure 4). Each protocol has a probe function embedded within the main data processing path. The probe function is called for each packet, and if the packet belongs to a connection or flow of interest, selected parameters of the packet and of the connection state are retrieved. For example, the probe in the TCP code can be programmed to retrieve the congestion window size of a given TCP connection, in order to see how the window size changes with time. The probe functions write the selected values to a kernel trace log, a circular buffer maintained in memory.

The probe functions within the protocol stack are all independent and are programmed via the Network Management agent. Once the probe functions are activated and start writing to the kernel trace log, it is the responsibility of the network management agent to retrieve the parameters written to the log and to do preliminary data analysis and compaction, before returning the desired network events to the NOC.

In addition to monitoring the state of any flow, it is also possible to program the probe function to modify the state of a flow by changing various parameters associated with the flow state and the protocol state. So it is possible to implement feedback control from a NOC by setting protocol control knobs on remote end stations, for example, to limit traffic during periods of congestion. It is important to note that this capability of reading and writing flow state parameters is different from, and complements, the functionality offered by SNMP. SNMP does not provide the fine grained per flow state monitoring and control described in our design.

5 Software Network Management Agents

Network management agents are responsible for mediating NOC access to managed objects within management information bases (MIBS). Managed objects contain attributes describing the health and status of network elements. Standard MIBS contain pre-defined objects corresponding to protocol and hardware resources in an Internet environment (such as IP, TCP, UDP, and network adapters).

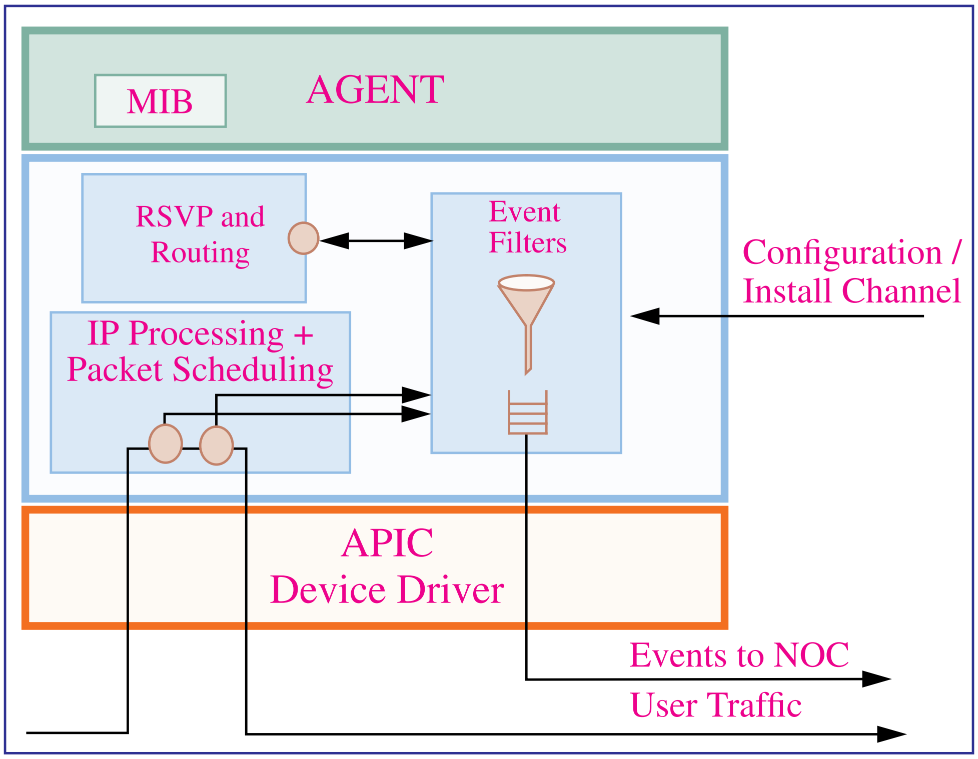

Figure 5: Software Organization in the Endsystem

As shown in Figure 5, agents in our proposed NMVC are built atop network probes and used to track event flows, as well as classify and report events of interest to NOCs. Common events reported to NOCs include *alarms* (such as delay or packet loss thresholds being exceeded for a service class), and *quality of service statistics* such as the performance of particular IPv6 packet flows, and ATM connection blocking rates. When these types of events are detected by the agent, it notifies the NOC via a trap and/or performs a corresponding local action.

Agents help NOCS maintain the reliability and quality of service for the overall system. In conventional low-speed networks this is typically accomplished by having NOCS periodically poll certain attributes of objects managed by agents. Polling is used to monitor the current state of the network elements and to take corrective action when problems are detected. However, selecting an appropriate polling interval is difficult. Polling too often creates excessive network and system load, whereas polling too infrequently prevents NOCS from detecting and reacting to serious system problems.

In gigabit networks, polling is even more problematic because the tremendous volume of events can trigger an enormous number of state changes between polling intervals. However, administrators and management applications may only be interested in a subset of these events (*e.g.,* alarms) and state changes (*e.g.,* failure of a network adapter or an image server). This problem motivates the use of trap-directed polling and high-performance event filtering in our proposed NMC system.

5.1 Event Filtering

We plan to enhance the flexibility, scalability, and performance of event filtering in NMVC network management agents by developing an event filtering framework based on JAVA. We will use JAVA to create a highly portable, configurable, and efficient filter installation and execution environment. The significance of using JAVA is that it is platform independent and thus can be used to download and reconfigure remote filters. The JAVA filter code can be compiled on the local machine making it more efficient than using interpreted code.

• **Filter specification:** Filters will be specified in a high-level specification language, transmitted across the network to the node agents, and compiled into machine code that is optimized for each network node platform. Our high-level specification language will be based on JAVA. Filters will be defined as a set of predicates using JAVA features like objects, methods, and expressions. These filter predicates can be used to specify protocol header fields (such as TCP port numbers, IP source and destination addresses, ATM VCs), application-level information (such as timestamps, telemetry measurand values, and image type), as well as node parameters (such as buffer occupancy, cumulative packet loss, and average delay).

• **Filter installation:** The use of JAVA greatly simplifies remote installation and configuration. JAVA programs can be downloaded from NOCS to agents using the widely available HTTP protocol. Moreover, they can executed in a portable manner on a broad range of hosts and agents.

• **Filter fusion:** Our proposed NMVC will compose multiple filters to reduce the number of operations required to process E messages through N event filters with P predicates from $O(E \times N \times P)$ to $O(E \times P)$, which is the minimum possible. This enables scalable performance improvements when N becomes large (as is the case for agents in large-scale networks). Efficient filter fusion will be implemented using a dynamic trie-based filter engine. A trie is an indexed-based search tree which eliminates expensive comparisons. Multiple filter expressions can be composed using dynamic trie-based data structures. To reduce search overhead, the trie coalesces common predicates according to criteria such as frequency of duplication or predicate priority. This trie-based approach is a generalization of the DAG-based techniques presented in [13]. Every node in the trie implements a particular type of branching mechanism. The branching mechanism selected at a node employs a lookup function (such as a hashing function, a binary search, or a series of inlined relational expression comparisons). During trie construction, the appropriate type of lookup function is selected based upon the type and the range of data values in a node.

• **Filter Compilation:** To eliminate the cost of interpretation, we will apply JAVA "just-in-time" compilation techniques to flexibly configure (and reconfigure) high-performance event filters. This supports flexibility *and* high performance. For instance, filters can be reprogrammed by NOCs to adjust their sampling rate for different classes of events. These new filters can be downloaded from the NOCS to the agents as portable bytecode. The agent will then fuse this bytecode with existing filters and recompile it into the native instruction set for the agent

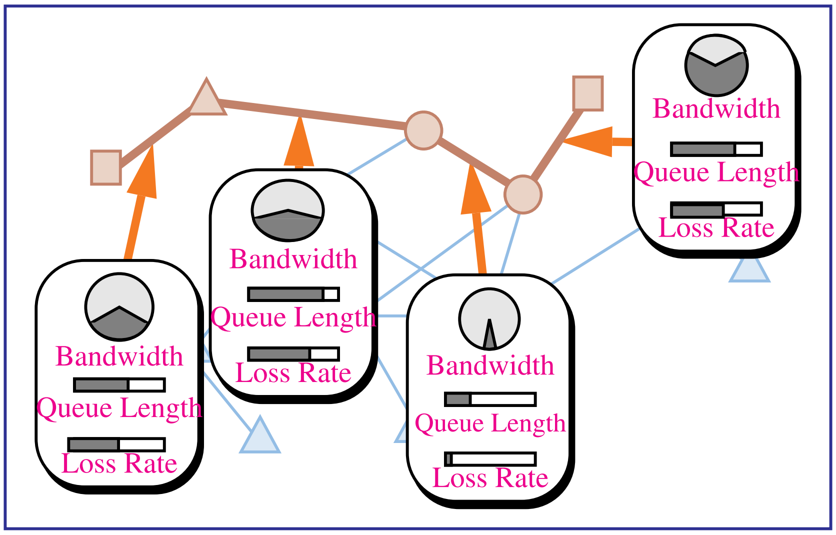

Figure 6: Application Connection Tracing

host. Thus, filters will be able to execute directly on the native agent hardware, which is an order of magnitude faster than interpreted code.

5.2 Online Event Ordering

Many of the displays and analysis tools in our NMVC system require the event stream to maintain causal ordering (*i.e.,* events must obey the Lamport "happened-before" relationship [14]). A common way to maintain event ordering in centralized systems is to timestamp the events and sort them before analysis. However, the distributed, real-time nature of large-scale high-speed networks prevents the use of globally consistent event timestamps. Therefore, as event streams from individual nodes in the network are merged at a NOC, variation in network congestion, processing delays, and buffering of events by software agents can "misorder" events in the combined event stream.

Out-of-order events can confuse viewers as well as produce misleading or incorrect displays [15, 6]. For example, Figure 6 displays an end-to-end path trace that illustrates the bandwidth, queue length, and loss rate for each link. As a burst of traffic passes along this path, we would expect to see the bandwidth and queue length indicators rise and fall as the traffic burst passes through each link. However, the misordering of events arriving at the NOC could cause the indicators to rise and fall in reverse order, or even in unison. To alleviate this problem of misleading displays, we are developing advanced ordering filters that ensure that the event stream reaching NOC display and analysis tools adheres to causal ordering.

We have developed a similar filter for use in the Falcon [16] project. However, the filtering code used in Falcon is specific to the particular instrumented threads library in use in that project. We are proposing a more general technique for the generation of ordering filters, based on a specification of ordering relationships and events. Thus, the ordering components that we develop will be easily transferable to other projects and uses.

These event ordering or *causality* filters enforce a set of ordering rules [16]. They examine each incoming event record and check the applicable rules (*i.e.,* specified causal relations). If no rules are violated, internal data structures are updated to reflect the occurrence of this event, and the event is passed on. Conversely, if a rule violation is detected, the event record is queued until the applicable rule is satisfied.

5.3 Related Work

While the commercially available SNMP and CMIP based network monitoring and management tools perform fairly well for conventional IP networks in a LAN environment, most have limitations that make them inadequate for the challenges of high-speed network management, visualization, and control.

5.3.1 Software Management Agents

With few exceptions (such as Cabletron's SPECTRUM and OSI NetExpert), the tools we surveyed did not handle correlations between network entities. Thus, many alarms and traps can potentially be generated due to a single point of failure. SPECTRUM is built around an artificial intelligence engine, called the Inductive Modeling Technology (IMT), which together with its object-oriented design, permits SPECTRUM to understand dependencies. The SPECTRUM server allows both server-initiated and operator-initiated device polling. Further, no redundant monitoring of devices occurs when multiple clients are monitoring the same device on different maps. However, the package has quite a few reported performance inefficiencies and thus, it is slower than other commercially available tools and cannot scale to networks operating at Gb/s.

The Center for Telecommunication Research at Columbia University has proposed an Integrated Reference Model (IRM) [17] for the management and control architecture of gigabit networks based on the OSI CMISE. The core of CTR IRM is a centralized network telebase that contains extensive information about network operations. However, the structure of the database and issues involved in data collection, event-ordering, and feedback control are not adequately addressed. The system runs on a WAN emulator with a software emulation of a subset of IRM on a supercomputer. Empirical results based on actual networks have not been obtained. Also, the thrust seems to be on developing a Virtual Reality GUI rather than on real-time measurement, visualization, and control [2, 3].

The group at LBNL [18] recognized the problem that the applications on high speed networks typically get only a small fraction of the network bandwidth. To analyze the problem and pinpoint the bottle-necks, they have developed a logging facility and have used it for a distributed application running on their testbed ATM network. This has been very useful in identifying performance bottlenecks and understanding interactions of protocols at various layers. However, while events are logged in real-time, all analysis is done offline, that is, after all the logging has terminated. This does not give much scope for online feedback control. There is no auto-detection of faults or human-in-the-loop, issues which our proposed scheme addresses fully. The design is tightly coupled to and oriented towards their chief application - the Distributed Parallel Storage Server [18]. Also, there is no support for online event-ordering and, except for the logging of events, only commercially available monitoring systems are used which can be inadequate.

OC3MON [12] does provide a wide variety of filters for collecting network traffic statistics such as number of active flows, traffic in bytes, packets and so on but the set of filters is fixed and new filters cannot be added at runtime. Our system provides a mechanism using Java and an efficient filter fusion technique to allow network managers to configure and install filters matching their specification to remote agents as discussed in Section 5.1. Also, OC3MON does not address the issue of event ordering.

5.3.2 Event Filtering

An extensive survey of existing event filtering mechanisms spanning domains such as active databases, distributed systems, network monitoring and management, on-line network services, and user-level communication protocols can be found in [19]. From this survey, we see that in general, conventional mechanisms provide limited support for filter fusion, focusing primarily on packet filters for the TCP/IP communication protocol suite [20, 21, 22, 13]. Our work extends existing research by producing a flexible object-oriented framework for filter fusion based on a dynamic trie-based filter engine.

HP OpenView provides an implementation of the ISO OSI event report management services [4]. HP OpenView filtering supports the registration of Event Forwarding Discriminators (EFDs) on remote agents in a network. However, OpenView's implementation of OSI EFDs is inadequate for high-performance event filtering. It suffers from a highly inefficient process architecture that requires 4 context switches and 3 interprocess communication (IPC) exchanges to receive, filter, and deliver each event end-to-end [19].

Bates has implemented dynamic analysis using primitive and high-level events, where some filtering is performed for high-level events in order to reduce the amounts of information presented to users. However, in contrast to our approach, Bates performs filtering after data collection has been completed [23].

Isis [24] supports event filtering in its *Reliable Distributed Objects* (RDO) News service [25]. In Isis RDO News, filtering is performed at the destinations since all consumers in a process group receive all events sent by suppliers. Therefore, the RDO News filtering architecture will not scale to accommodate a large numbers of consumers that possess complex filtering requirements. In contrast, the event filtering mechanism proposed in our effort allows filtering at the agents, which supports much greater scalability in gigabit network environments.

CSPF is a stack-based packet filter that operates on binary instructions (*e.g.,* `pop` and `push`). CSPF uses boolean expressions, along with a tree model, to configure its filtering engine. BPF extends and enhances CSPF to overcome certain performance limitations with CSPF. Both CSPF and BPF maintain a list of configured packet filters (the list maybe reorganized dynamically to move frequently accessed filters to the front of the list). This approach works well if there are relatively few packet filters, or if only a small number of consumers are active simultaneously. When there are hundreds of filters and/or hundreds of consumers, however, this approach does not scale up since the time required to filter packets grows linearly with the number of filters.

The Mach Packet Filter (MPF) is designed to support user-level implementations [26] of layered protocol stacks (such as TCP/IP). The MPF composition technique assumes the existence of a common prefix across protocol headers. This assumption enables low latency setup and removal of composite packet filters. However, this particular optimization strategy does not generalize to compose more complex filters.

The PathFinder tool described in [13] presents a more general technique for coalescing filters with common prefixes. PathFinder is a packet classifier that combines software and hardware techniques to optimize the composition of complex filtering patterns. The software portion of PathFinder builds a directed-acyclic graph (DAG) interpreter based upon patterns specified by a user-defined declarative syntax. One limitation with PathFinder is the relatively high latency required to setup/remove patterns from the DAG (since the DAG must be searched from the root to the leaves when inserting/deleting a new pattern). In addition, the software implementation of the DAG uses an interpreter, rather than a compiler.

6 Network Visualization Tool

Visualization tools for parallel and distributed systems assist developers in debugging, evaluating, and tuning parallel and distributed programs by presenting displays of communication behavior, utilization, and computation. Existing algorithmic methods for managing networks have not matured to the point where fault detection, isolation, correlation, and correction can be automated scalably [1].

Paragraph [27] is a portable trace analysis and visualization package for message-passing programs, producing displays of processor utilization, communication, etc. However, it relies on the production of PICL [28] traces, does not support the creation of new views, and is designed for the visualization of application programs rather than networks. Similarly, the Pablo project [29] addresses source code instrumentation, data capture, and data analysis for massively parallel systems, but does not adequately address the needs of high-speed networks. The PARADE project [30] is composed of the POLKA animation toolkit [6], designed to support the visualization of parallel and distributed systems, an Animation Choreographer [15], and various libraries of predefined animation views for particular architectures and programming paradigms such as PVM, Conch, parallel C threads, and High-Performance FORTRAN. We will extend this approach to the online visualizations of network status and behavior. Falcon is a system for on-line monitoring and steering of large-scale parallel programs. The purpose of such interactive steering is to improve the program's performance or to affect its execution behavior. The Falcon system is composed of an application-specific on-line monitoring system, an interactive steering mechanism, and a graphical display system. However, Falcon is designed for the monitoring and steering of threads-based application programs [16].

Our proposal is based on an alternative approach: *distributed visualization using graphical animation.* Graphical animation taps into well-developed human abilities to detect patterns and recognize anomalies in those patterns. Thus, visualization can help administrators detect recurring traffic patterns, determine appropriate quality of service parameters, and fine tune network performance. To be an effective tool for managing large-scale, high-speed networks, visualizations must exhibit scalability and celerity. Scalability is achieved through the hierarchical presentation of views at various levels of abstraction and through support for interactive navigation through these views. Celerity or speed is achieved through careful design of displays and by semantic filtering of animation events resulting in timely display that approximates the current global state of the network.

The visualization tool, represented by the shaded area in Figure 2, consists of a display component, and a mapping component. The display component in our proposed NMVC system will initially be prototyped using a version of POLKA [6], which is an animation toolkit designed to support concurrent animations. POLKA supports color, real-time, $2\frac{1}{2}$ dimensional, smooth animations, and focuses on a balance between power and ease-of-use. POLKA provides its own high-level abstractions to make the creation of animations easier and faster than with many other systems. Thus, the user can easily modify the existing display, or specify new ones. POLKA has been successfully used in the visualization of the execution of PVM, HPF, and parallel C threads programs, and as the basis for program steering decisions in the Falcon system [16]. However, POLKA will not scale up to be able to handle the frequency and volume of events generated by a gigabit network. Hence, we are creating a more highly optimized display package for the NMVC system that retains POLKA's capability of animation of concurrent systems. One idea is to use Java to create the display package which can then be transferred and compiled at a NOC.

An essential element of the visualization tool is a mapping component, called the **View Choreographer** in the proposed NMVC system. The View Choreographer relies on specifications of the following: the format of events produced by the monitoring system; the set of views available for display; high-level (composite) events and their component low-level (primitive) events; and the mapping from events (either high-level or low-level) to updates of the graphical views. Based on these specifications, the View Choreographer transforms the event stream emanating from the ordering filters into the appropriate calls to update the display. Another function of the Choreographer is filtering based on the graphical semantics of the events passed on to the Choreographer. The goal of this filtering is to avoid redundancy in the display by selectively combining, ignoring, or postponing the presentation of selected execution events.

In the development of this network visualization and control system, we focus on creating a *parameterized, modular* system that provides displays that are both *scalable* and *informative.* In the paragraphs below we discuss the importance of these characteristics in such a system.

Research in communication networks continues at a rapid pace. In recent years, various representations for network management information have been proposed, including SNMP, CMIP, and SNMPv2, and assorted proprietary formats. To promote the continued usefulness of our network visualization system in the face of changing standards, we propose a modular framework that will permit the specification of new and different event types and displays. This flexibility is accomplished through the use of event, display, and mapping specifications and the automated generation of a mapping component, the *View Choreographer.* This clean, well-defined interface between data collection and display will allow new standards for network performance information to be easily incorporated as they are developed. Similarly, new displays that are found to be useful to network administrators can be easily added to the system.

A suite of displays will be developed, including displays to assist with the rapid depolyment of networks, to check the configuration of network topology and connectivity, and for performance management, fault handling, and accounting management. To be effective, these displays for managing large-scale, high-speed networks must also be *scalable.* Some of the views in the suite will present a high-level view of the network. Others will be lower-level, such as views of individual nodes and links, representations of bandwidth utilization, loss rates, and queue lengths, and other pertinent information. In general, high level views will be provided and statistics for lower-level visualization is collected only upon a specific request by a network manager. On receiving such a request, the remote software monitoring agent would be reprogrammed (using trie-based filter fusion for efficiency) to probe for example a specific virtual circuit. Views will be connected in a hierarchical manner, and the system will support the navigation of views at multiple levels of abstraction. Lower-level views may be activated by "warning" or "alarm" events or through user interaction with higher-level views.

An essential characteristic of these displays is that they be *informative.* This implies that not only must they contain the data essential to an administrator charged with monitoring and controlling the network, but that the information be presented in a way that is easily perceived by the viewer, and in a timely manner. This will require the careful application of principles of design for human-computer interaction to the creation of displays. To ensure that visualizations represent recent states of the network that approximate the overall global state, displays will be designed to allow rapid update. Further, the filtering functions of the View Choreographer will assist in minimizing the delay between the arrival of information regarding the network state and the visual presentation of this data.

7 Proposed Experimental Testbed

To experimentally evaluate the capabilities of our NMVC system, we propose building a 3-5 node testbed composed of ATM switches, network probes, IP packet processing elements, network management agents, network operation centers (NOCs), a storage archive for off-line analysis (shown in Figure 1), and many endsystems that will host distributed applications.

The ATM switches are replicas of Washington University's gigabit ATM switching systems. These switches are highly scalable in terms of the number of ports and provide extremely efficient support for multicasting. The network probes will be built using an APIC chip and a CPU-module. To test the features of NMVC components (such as ATM switches, network probes, IP packet processing elements, filtering agents, and visualization tools) we will develop a suite of multimedia traffic generator tools. These tools will enable us to generate network traffic according to different traffic models in a controlled manner. We will use these tools to evaluate the NMVC components and to explore the opportunities for interactive and algorithmic feedback control.

The main objectives for building the testbed is to be able to experimentally prove the functionality of our NMVC system, gain insight from the event traces to develop automatic control algorithms and use the event traces to drive a simulation model of a large managed network. We believe that we can use such a simulation study to prove that our system is massively scalable.

8 Conclusion

In this paper, we have described a design of a highly scalable NMVC system with advanced algorithmic and human-in-loop capability. To meet its objective, the NMVC system is composed of the following innovative components.

- An inexpensive, yet high performance network monitoring and observation probe that does not interfere with high speed network traffic.

- An efficient probing mechanism for endsystems (i.e., workstations and servers) that can collect relevant performance statistics from both applications and kernel resident protocols (such as TCP/IP in Unix).

- A highly flexible and highly efficient software network management agent design that extends the functionality and scalability of the network probe by filtering data based on programmable criteria (such as protocol header attributes or application-level frames).

- Efficient online event ordering algorithms that can help synthesize and display a consistent view of network health, status, and performance using events collected throughout a large-scale high speed network.

- A *View Choreographer* that allows management applications and administrators to specify the mapping of network events to higher-level events and to visualization objects and updates.

- A visualization system that is highly configurable and can support multiple simultaneous real-time display views of a large-scale network.

An in-depth survey of current network management systems has shown the existing algorithmic methods for managing networks have not matured to the point where performance bottleneck and fault detection, isolation, correlation, and correction can be automated scalably. We believe that our NMVC system design will lead to a complete and highly scalable solution for network management and ensures adequate quality of service to network users, while maintaining high network resource utilization.

References

[1] O. Wolfson, S. Sengupta, and Y. Yemini. Managing Communication Networks by Monitoring Databases. *IEEE Transactions on Software Engineering*, 17:944–952, September 1991.

[2] Laurence Crutcher and Aurel A. Lazar. Management and Control for Giant Gigabit Networks - are we ready for B-ISDN. Technical Report 341-93-21, Center for Telecommunications Research, Columbia University, 1993.

[3] Laurence A. Crutcher, Aurel A. Lazar, Steven K. Feiner, and Michelle Zhou. Management of Broadband Networks Using a 3D Virtual World. In *Proceedings of the 2nd International Symposium on High-Performance Distributed Computing*, July 1993.

[4] Hewlett-Packard OpenView.

[5] IBM NetView/6000.

[6] J. T. Stasko and E. Kraemer. A Methodology for Building Application-specific Visualizations of Parallel Programs. *Journal of Parallel and Distributed Computing*, 18:258–264, June 1993.

[7] Zubin D. Dittia, J.R. Cox Jr., and Guru M. Parulkar. Design of the APIC: A High Performance ATM Host-Network Interface Chip. In *IEEE INFOCOM '95*, pages 179–187, Boston, USA, April 1995. IEEE Computer Society Press.

[8] Zubin Dittia, J.R. Cox Jr., and Guru Parulkar. Catching Up With the Networks: Host I/O at Gigabit Rates. Technical Report 94-11, Dept. of Computer Science, Washington University in St. Louis, 1994.

[9] Zubin Dittia, J.R. Cox Jr., and Guru Parulkar. Using an ATM Interconnect as a High Performance I/O Backplane. In *Proceedings of the Hot Interconnects Symposium*, August 1994.

[10] Robert J. Walsh. DART: Fast Application-Level Networking via Data-Copy Avoidance. *IEEE Network*, (4), July 1997.

[11] Dan Wilkelstein. ATTILA: Performance Analysis System for ATM Networks. Technical report, MCNC, NC, 1994.

[12] Joel Apisdorf, K. Claffy, Kevin Thompson, and Rick Wilder. OC3MON: flexible, affordable, high performance statistics collection. In *Proceedings of USENIX*, September 1996.

[13] M.L. Bailey, B. Gopal, P. Sarkar, M.A. Pagels, and L.L. Peterson. Pathfinder: A pattern-based packet classifier. In *Proceedings of the* 1^{st} *Symposium on Operating System Design and Implementation*, November 1994.

[14] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21:558–565, July 1978.

[15] Eileen Kraemer and John T. Stasko. Toward flexible control of the temporal mapping from concurrent program events to animations. *Proceedings Eighth International Parallel Processing Symposium*, pages 902–908, 1994.

[16] W. Gu, G. Eisenhauer, K. Schwan E. Kraemer, J. Stasko, J. Vetter, and N. Mallavarupu. Falcon: on-line monitoring and steering of large-scale parallel programs. In *Proceedings of the Fifth Symposium on the Frontiers of Massively Parallel Computation*, pages 422–429, February 1995.

[17] Aurel A. Lazar. A Real Time Control, Management and Information Transport Architecture for Broadband Networks. In *Proceedings of the International Zurich Seminar on Digital Communications*, March 1992.

[18] Brian Tierney, William Johnston, Jason Lee, and Gary Hoo. Performance Analysis in High-Speed Wide Area IP over ATM Networks: Top-to-Bottom End-to-End Monitoring. *Submitted to IEEE Networking*, 1996.

[19] D. C. Schmidt. High-Performance Event Filtering for Dynamic Multi-point Applications. In 1^{st} *Workshop on High Performance Protocol Architectures (HIPPARCH)*, December 1994.

[20] J. C. Mogul, R. F. Rashid, and M. J. Accetta. The Packet Filter: an Efficient Mechanism for User-level Network Code. In *Proceedings of the* 11^{th} *Symposium on Operating System Principles (SOSP)*, November 1987.

[21] V. Jacobson and S. McCanne. The BSD Packet Filter: A New Architecture for User-level Packet Capture. In *Proceedings of the Winter USENIX Conference*, pages 259–270, January 1993.

[22] M. Yuhara, B. Bershad, C. Maeda, and E. Moss. Efficient Packet Demultiplexing for Multiple Endpoints and Large Messages. In *Proceedings of the Winter Usenix Conference*, January 1994.

[23] P.Bates. Debugging Heterogeneous Distributed Systems Using Event-based Models of Behavior. In *SIGPLAN Notices*, volume 24, pages 11–22, January 1989. In *Proceedings of the Workshop on Parallel and Distributed Debugging, May 1988.*

[24] Kenneth Birman and Robbert van Renesse. *Reliable Distributed Computing with the Isis Toolkit.* IEEE Computer Society Press, Los Alamitos, 1994.

[25] Isis Distributed Systems, Inc., Marlboro, MA. *Isis Users's Guide: Reliable Distributed Objects for C++*, April 1994.

[26] C. Maeda and B. Bershad. Protocol Service Decomposition for High-Performance Networking. In *Proceedings of the* 14^{th} *Symposium on Operating System Principles*, Asheville, North Carolina, December 1993. ACM.

[27] Michael T. Heath and Jennifer A. Etheridge. Visualizing the performance of parallel programs. *IEEE Software*, 8(5):29 –39, September 1991.

[28] G.A. Geist, M.T. Heath, B.W. Peyton, and P.H. Worley. A user's guide to PICL: A portable instrumented communication library. Technical Report TM-11616, Oak Ridge National Laboratory, September 1990.

[29] Daniel A. Reed, Ruth A. Aydt, Roger J. Noe, Phillip C. Roth, Keith A. Shields, Bradley Schwarta, and Luis F. Tavera. An overview of the pablo performance analysis environment. In *Proceedings of the Scalable Parallel Libraries Conference*, pages 104–113, Mississippi State, MS, USA, October 1994.

[30] John T. Stasko. The parade environment for visualizing parallel program executions: A progress report. Technical Report GIT-GVU-95-03, Graphics, Visualization, and Usability Center, Georgia Institute of Technology, Atlanta, GA, January 1995.