Distributed Laboratories: A Research Proposal

College of Computing
Georgia Institute of Technology
GIT-CC-96-13

April 1996

Principal Investigators: Richard Fujimoto
Karsten Schwan
Mustaque Ahamad
Scott Hudson
John Limb

Participating Investigators: Mostafa Ammar
Norberto Ezquerra
Amarnath Mukherjee
Colin Potts
Kishore Ramachandran
Ellen Zegura

1 Executive Summary

The continuing merger of computer and communication technologies is leading to a new computing/communications infrastructure of unprecedented magnitude, enabling new applications with broad economic and social impact. Yet, such applications pose major challenges to researchers in Computer Science and in application domains.

Distributed Laboratories. The topic of the proposed research program is the realization of *distributed laboratories*, where individuals can interact with each other, and more importantly, with powerful, distributed computational tools as readily as if all were located in a single site. Our intent is to permit scientists, engineers, and managers at geographically distinct locations (including individuals "tele-commuting" from home) to combine their expertise in solving shared problems, by allowing them to simultaneously view, interact with, and steer sophisticated computations executing on high performance distributed computing platforms. The research results and tools resulting from the requested infrastructure will have broad application in many domains. However, we will use two specific applications to focus our efforts, and to help ensure that our results and software tools are properly integrated:

1. A distributed laboratory for experimentation with high performance numeric computations for applications in molecular physics, atmospheric sciences, and manufacturing systems.
2. A distributed laboratory for studying the behavior of future-generation, large-scale telecommunication networks through high performance parallel and distributed simulation models of wired and wireless networks, called the *virtual telecommunication networks* application.

Proposed research. Five research projects will develop underlying, enabling technologies to support these applications. Three are primarily concerned with "middle-ware" software and technologies that will be directly utilized by the driver applications. The first project will study the dynamic monitoring, adaptation, and interactive steering of high performance computations for online control of "virtual laboratory instruments" and for "what-if?" experimentation with complex simulation models by distributed laboratory users. The second project will explore the efficient execution of simulation programs, especially discrete-event simulations of telecommunication network models, on multi-granular compute servers (servers containing both tightly-coupled multiprocessors and loosely-coupled workstations). The third project will explore the collaboration technologies required in distributed laboratories, including the architecture of CSCW systems, user interfaces, and the indexing of video and audio records about experiments.

The fourth and fifth project are concerned, respectively, with the distributed systems and telecommunication networking technologies underlying distributed laboratories. They will provide technical support and software for each of the three middle-ware projects. The emphasis in distributed systems research is on support for shared-state in multi-granular and distributed computing environments. Networking research is concerned with providing the necessary communication facilities for distributed laboratory applications, both at the workplace and when tele-commuting from the home. Networking research also serves a dual role as a user of the proposed virtual telecommunication network environment.

Specific research issues. Research issues explored in each of the middle-ware projects include:

- **Interactive Steering**
  - Distributed laboratory architectures: We will develop a framework for the establishment, management, and growth, of integrated distributed laboratory environments.

- Distributed laboratory managers: We will develop mechanisms that support the creation and management of distributed laboratories, so that laboratory components can be "plugged together" and participants can be added and removed dynamically.
- Foundations: We will define formalisms for describing configurable and monitorable attributes of distributed/parallel programs, and develop portable tools using these specifications for on-line program monitoring, configuration, and display of high performance computations on multiprocessor and multi-granular compute servers.

- **Distributed Simulations**
  - Efficient, adaptive synchronization protocols: We will develop methods for efficient synchronization of simulations executing on multi-granular compute servers, and apply these techniques to simulations of large telecommunication networks.
  - Mapping and load balancing: We will study the effects of load management policies on the efficiency of synchronization mechanisms, and develop techniques to map and load balance distributed simulations on multi-granular compute servers.
  - Real-time and interactive simulations: We will develop methods to enable distributed discrete-event simulations to support interactive and hardware-in-the-loop simulations by guaranteeing real-time constraints.
  - Shared state abstractions: We will develop multi-versioned memory structures to support shared-state abstractions for distributed simulation applications.

- **Collaborative Systems**
  - System architectures: We will develop structures for open and extensible collaborative systems utilizing a shared-state approach.
  - User interface composition: We will develop automatic and semi-automatic techniques to construct user interfaces for distributed laboratory applications.
  - Indexing video and audio recordings: We will develop new techniques for indexing video and audio recordings, e.g., so late-joiners can quickly review missed discussions prior to joining a meeting.

Research issues explored in the state-sharing and telecommunication network projects include:

- **State Sharing**
  - Caching mechanisms: We will develop scalable techniques for efficient caching of shared state in distributed laboratories, e.g., by exploiting weakened consistency requirements.
  - Compile-time and runtime support: We will explore the use of compile-time and runtime information to reduce overheads for managing consistent copies of shared state.

- **Networking Research**
  - The ubiquitous network: We will develop new protocols to enable cost-effective, high-speed, networking both within the office and lab using ATM technology and to the home and community utilizing fiber/cable plant, enabling distributed laboratory applications to be utilized wherever the experimenter may be located.
  - Multicast communications: We will develop techniques for scalable multicast communications for collaborative systems, maintaining coherence in state-sharing coherence protocols, and in distributed simulation computations.

– Quality of service guarantees and resource management: We will develop models to characterize workloads for distributed laboratory applications, develop resource allocation algorithms to maintain quality of service guarantees of connections, and evaluate these algorithms on the virtual telecommunication network facility and with actual distributed laboratory applications.

These projects build on each other in a synergistic fashion. For example, results from the state sharing and multicast research will be exploited by the interactive steering and distributed simulation projects. Work in these latter two projects will be brought together to enable distributed simulation computations to be rolled back, and re-executed forward using different parameter settings, providing the ability to, in effect, execute the simulation program backwards. The collaborative work project will enable multiple users at different locations to interact with these simulations, and adds capabilities for human-to-human interaction both during and after the experiments. The distributed simulation and networking projects are closely coupled through the virtual telecommunication network application.

2 Research

Our research proposes to create *distributed laboratories* in which scientists and engineers working in geographically distributed locations share access to observational instruments and large-scale simulation computations, share information generated by such instruments as well as data produced by computational models (i.e., virtual instruments), and collaborate across time and space to evaluate and discuss their results. The *distributed laboratory testbed* proposed by our group will contain the computational and networking equipment required for the creation of such laboratories distributed across the Atlanta metropolitan area, including multiple physical laboratory sites at Georgia Tech and researchers participating in laboratory experiments from their homes.

Enabled by the planned national information infrastructure, we expect several benefits from the use of geographically distributed laboratories. First, scientists, engineers, and managers can work from their office locations, laboratories, or homes with similar ease, and they can interact with each other and with sophisticated computational tools with minimal regard to physical location. This can increase their productivity, reduce transportation costs and pollution (tele-commuting), and result in the rapid distribution of scientific insights among collaborators. Second, by dynamic extension of laboratories to new sites and by rapid inclusion of new investigators or instruments, special skills and tools available at different physical locations can be brought to bear on the solution of complex problems. Third, the ability to run and control experiments on distributed computational resources permits the co-location of instruments with relevant I/O facilities or even the emulation of future distributed physical systems by distributed simulation of their physical processes.

Two "driver" applications will be utilized to focus and ensure proper integration of our research results and tools, thereby maximizing its impact. The first is a distributed laboratory for experimentation with high performance numeric computations for applications in molecular physics, atmospheric and pollution modelling, and manufacturing systems. The second concerns a laboratory for experimentation with next generation telecommunication networks via sophisticated distributed discrete event simulation software, called the *virtual telecommunication network* application.

The research enabled by the proposed distributed laboratory facility will be *the provision of tools and principles with which 'distributed laboratories' can be constructed and controlled.* In contrast to facilities like the World Wide Web which are mostly employed for information access and browsing, a *distributed laboratory* has several essential and novel characteristics:

1. The information generated in a distributed laboratory must not only be accessible to all users, but users must also be given interactive interfaces for the on-line control of virtual or physical laboratory instruments, for steering simulations and other computations using instrument data, and for controlling information generation, analysis, storage, and sharing between sites and physical and virtual instruments. Furthermore, interactive laboratory access should be provided with little regard to the physical location of experimenters and instruments, including home locations with limited computational and storage capabilities, access bandwidths, and increased access latencies.

2. Users of a shared distributed laboratory must be able to participate in meetings and discussions where laboratory resource usage, experimental results, and research goals and plans can be discussed even when the users are not co-located or available for a meeting at the same time. In addition, and in contrast with current collaboration environments (e.g., support-

ing direct remote audio and video), a shared laboratory must also support communications through and with computations such as shared manipulations of visualizations, interactive experimentation with running simulations, remote and joint operation of programs controlling instruments, etc.

The research that will be performed using the requested facility is structured around five on-going research projects. Topics to be addressed by the proposed work include user interface technologies for collaboration, high-performance simulation computations, software tools and operating system technologies to support distributed laboratories, instruments and experiments, and finally, communication protocols and platforms offering the required communication latencies and bandwidths into laboratories, offices, and homes. Specific research problems addressed by our group include a framework for construction of collaborative systems, the interactive monitoring and control of instruments and experiments, the efficient implementation and execution of large-scale distributed simulations, the provision of operating system support for efficient representation and manipulation of the state shared among simulations, laboratory instruments and experimenters, and high performance networking across the range of physical media and locations used by distributed laboratories. Networking serves a dual role as both a principal area for much of our research and as an application using the tools we develop.

Application Scenarios. To illustrate the distributed computational laboratories we envision and to expose some of the underlying research issues that must be addressed, we next describe a scenario depicting envisioned laboratory use. This scenario uses an atmospheric science modeling application (one user of the tools we will develop) as an example, but our efforts will be directed at a variety of target applications, including the aforementioned virtual telecommunication network application which is based, in part, on on-going collaborative research between our group and Bellcore.

In the following scenario, multiple laboratory instruments are interconnected and multiple scientists are able to interact with each other and with distributed laboratory instruments on-line and across local or wide area networks. Prototype systems realizing scenarios such as this, as well as similar scenarios concerning the virtual telecommunication network application will be realized during the period of this proposal.

Consider two atmospheric scientists from different parts of a campus working together on one of the atmospheric modeling simulations. Dr. Smith (fictional name) is working on a high-end SGI workstation in the Atmospheric Sciences building, Dr. Jones is in the College of Computing building running on another SGI machine, and the simulation code itself is running on a multi-granular compute server. Collaborative work includes (a) debugging or simply experimenting with parameters used in the simulation code (e.g., wind fields, initial values of chemical concentrations, etc.) or (b) understanding observational data received from NASA sites or from the U.K. (the UARS satellite data sets) in reference to output data being produced by their simulation models. One individual in our scenario is a computer scientist knowledgeable about the simulation software, the multi-granular compute server, and data visualization tools. The atmospheric sciences researcher is primarily concerned with interpreting simulation inputs and outputs.

The scientists begin work by establishing a communications link between them that transports voice and a small video image of each other to their respective workstations. This connection is established by dragging an icon representing the other party into a shared workspace. Initially, the video connection has a fairly high scan rate for improved person-to-person communications (as they say hello, etc.). After a predetermined time (about 5 minutes) or when experimentation has shifted from discussion to work employing other media, the scan rate drops off to less than one frame per second to serve more of an awareness and less of a communications role.

After a short discussion, the scientists begin work by preparing data for use as initial conditions to the simulation. This involves using observational data stored locally on Dr. Smith's workstation (or perhaps retrieved from a site across the country) and an interpolation/conversion program running on one of the local workstations. The data is initially visualized on both Smith and Jones' workstations using simple 3D visualization tools, where one tool creates iso-surfaces depicting chemical concentrations and a second tool draws vectors indicating wind fields. The purpose of the discussion among the two scientists is twofold: (1) to compose these separate visualizations into a single display using a simple depth-buffer technique developed by Dr. Jones, and (2) to determine visualization parameters (e.g., correctly using and inserting polar projections) resulting in output data useful to atmospheric science researchers. In addition, while Dr. Smith is intimately familiar with the actual input data set and with its processing, he does not understand how to import actual data into the visualization package (e.g., dealing with conversion among different data formats, etc.), which is Dr. Jones' domain.

Once the observational data has been interpolated into a suitable grid structure, Smith and Jones, decide to modify parts of the data in order to answer a what-if question via the simulation. For example, they increase chemical concentrations in one area to see how its effects propagate over time (e.g., to study the damage to ozone from high altitude aircraft). This is performed using a shared data manipulation tool used with the visualization. This tool runs at whatever location is maintaining the data, but exports a simple user interface to all sites. The tool internally represents selected information about the distributed laboratory as shared data, modifiable and accessible to all participants, and it permits its data representation (e.g., shared files or shared abstract types) and its enforced data consistency to be made specific to the target application program's needs. Once the initial data has been determined, data is input into a complex simulation program executing on the compute server. In effect, this simulation represents a complex *virtual instrument.* The simulation is initiated interactively by connecting an icon representing the dataset to an icon representing the virtual instrument.

After several time steps, the scientists begin to look at simulation (experiment) results, again using the volume visualization described above. However, to inspect output data in detail, a shared 3D 'slice tool' cuts the volume visualization at some plane and renders more detailed information on the cutting surface. In this case, the cutting plane shows a color coding of the grid and step sizes being used by the simulation. The cutting plane is moved back and forth within the volume to identify areas that have interesting features but coarse sizes, as well as to identify uninteresting areas that have too fine a granularity in the simulation. Performance tuning done in this fashion benefits from the knowledge of the computer science user about multi-granular machine and implementation characteristics (e.g., suitable granularity of parallelism, or required degree of locality in data and code access) and the knowledge of the atmospheric scientist about the application itself (e.g., permissible variations in grid sizes). Based on their findings and a discussion between them, the scientists may use user interface objects (buttons, sliders, etc.) exported by the simulation to modify, for example, parameters in the mathematical approximations capturing vertical constituent movement in the atmosphere. Atmospheric scientists desire to experiment with alternative formulations and implementations of such movement (since the actual physical processes in the atmosphere remain ill-understood), while computer scientists try to understand the performance effects of increased data sharing along the vertical dimension. Both sets of scientists can perform such work by jointly controlling the shared 'instrument' (i.e., steering the program).

Other aspects of our research not encompassed by the above scenario include extending our environment to the home, and temporally distributed (different time, different place) interaction between experimenters. Work in the virtual telecommunication network project also introduces

issues concerning interactive distributed discrete event simulations.

It is readily apparent how the research projects included in this program contribute to the realization of the scenario. The interactive steering and distributed simulation projects provide the necessary interactive, high performance computation tools. The collaborative work project provides not only a system for sessions such as that described above to take place, but also studies effective means of indexing video and audio recordings of prior and on-going sessions. These three projects, in turn, build on system software for supporting shared-state among program components and/or human users. Finally, the networking research examines future-generation networking technology that is essential to making these scenarios practical.

Integrated systems using the testbed facility will be realized utilizing tools and results from each of the five research projects. In the following we elaborate on the driver applications and the specific research problems that will be addressed in each of the five projects, and discuss the relationship among these projects.

2.1 Driver Applications

2.1.1 Toward Distributed Laboratories for Scientific Experimentation

Motivation. The novel characteristics of a *distributed laboratory* compared to arbitrary distributed programs have been identified earlier as (1) the need to make all information generated in such a laboratory on-line accessible and to all users, and the need to give users on-line control of experiments and laboratory instruments, and (2) the need for additional collaboration tools enabling users to interact not only via their instruments and experiments but also via video and audio interfaces supporting both on-line and time-delayed human interactions. This section elaborates on the *spatial* and *temporal* distribution of laboratory instruments, experiments, and experimenters, where multiple laboratory instruments can be interconnected and multiple scientists are able to interact with each other and with distributed laboratory instruments on-line and across time via local or wide area networks. The specific example used to explain these concepts is the aforementioned, on-going interdisciplinary research effort in *the interactive steering of global atmospheric models*, which currently brings together four sets of investigators at Georgia Tech, residing in (1) the School of Earth and Atmospheric Sciences, (2) the College of Computing, (3) the Center for Graphics, User Interfaces, and Visualization, and (4) the High Performance Computing group. This effort already utilizes distributed machines ranging from SGI and SUN workstations to the low-end supercomputers existing at Georgia Tech (a Cray, a KSR, and an IBM SP2 machine), linked with FDDI and Ethernet local area networks. With the proposed funding, the physical facilities available to this research will be enhanced significantly, including the availability of higher performance computing in form of the proposed multi-granular compute server. Furthermore, the I/O needs of atmospheric modeling applications will be served with the proposed, high performance communication infrastructure. This will enable investigators to use both off-line (typically CD-ROM or taped-stored) and on-line (acquired from satellite links) observational data in model computations, along with the outputs of previous model runs.

Definition of terms. A *distributed laboratory* consists of a collection of physical or virtual instruments, which are interoperable due to their use of a common data format. Each individual *instrument* consists of a program implementing or representing it, its defined inputs and outputs in terms of our common data format definition, and of interactive controls and sensors/probes able to steer and monitor its use. An *experiment* consists of a time duration within which a number of

instruments are used to solve some problem of interest to the experimenters. Associated with this experiment are (1) the required and generated input and output data, (2) descriptions of monitoring specifications and output data, and of control information (i.e., steering commands and instrument settings not specified by input data), and (3) discussion information recorded from the video and audio stream and indexed by the telepointing methods explained below, based on all of which the temporally offset experimenter can repeat experiment runs when necessary. The research issues arising for (3) are discussed in Section 2.2.3 of this proposal. Issues concerning (1) and (2) are addressed in Section 2.2.1. Operating system support for the efficient sharing of data and control between coupled instruments, collaborating end users, and instruments and their user interfaces are explained in Section 2.3.1.

Purpose of research. The purpose of our work is to enable experimenters to perform effective research in the physical and engineering sciences with physical or virtual instruments that are distributed across different physical laboratories, computational sites, and even experimenters' offices. Toward this end, we will develop the appropriate tools, systems support, and user interfaces for on-line interaction with remote models, instruments, or data sets in distributed and parallel systems. Our research has three specific aims:

1. *On-line instruments.* It should facilitate and enable collaborative, on-line experiments among different researchers who are all working with a single target simulation (the atmospheric modeling code) running on a single compute server. In essence, when researchers are sharing a single, remote instrument (i.e., the model), they should be given the ability to converse via this shared instrument (simultaneous or handed off instrument control, sharing of visual displays of instrument outputs, etc.).

2. *Spatially distributed experiments.* A distributed laboratory should facilitate the simultaneous use of multiple, spatially distributed instruments. In this example, such instruments include the pre- and post-processing packages required for atmospheric data (e.g., satellite-received data sets), the shared visualizations of complex 3D data sets, and the actual atmospheric modeling code.

3. *Temporally distributed experiments.* A more complex set of experiments involves multiple distributed instruments, where no single investigator can be assumed familiar with all of the instruments involved. The specific cases we are studying concern (1) the debugging of specific models (i.e., instruments) being developed jointly with Computer Science experts on parallel and distributed programming with domain experts familiar with atmospheric science applications and (2) the connection of local pollution models with a global atmospheric model, where outputs from the local model determine input values of the global model (e.g., the concentrations of certain constituents at selected geographical locations). Due to the inherent physical distribution of experimenters (e.g., atmospheric researchers at Georgia Tech primarily study global phenomena, whereas local phenomena and models are being studied at other sites in the U.S. and Europe), such experiments may require that experimenters 'joining late' can 'catch up' on previous experiment results and discussions, by reviewing experiment data and/or by reviewing video-taped discussions among experimenters.

2.1.2 Virtual Telecommunication Networks

The second driver application is an environment to support the design and management of large, complex, telecommunication networks among a collection of geographically distributed engineers,

scientists, and managers. We envision an environment to support (for instance) "brainstorming sessions" among a group of researchers, geographically distributed in their respective homes and/or offices, each viewing, on-line simulation experiments of future-generation telecommunication networks. We envision scenarios where collections of individuals simulate a variety of approaches to recover from a major system failure in order to identify the most suitable approach.

At the heart of this environment are interactive, large-scale, distributed simulators containing sophisticated models of the network. Interactive steering tools provide the ability to experiment with different "what-if?" scenarios by repeatedly rolling back the simulator to an earlier state (e.g., after steady state is reached, but just prior to some stress situation), changing model parameters to study network reaction to a specific phenomenon, and then repeating this exercise for different parameter settings. Collaborative work tools enable multiple participants to simultaneous view and interact with the simulator, as well as each other, in these brainstorming sessions.

The VTN laboratory will support research in wired and wireless networks, as well as full-system simulations utilizing both. Users of the tools including researchers in the telecommunication group here at Georgia Tech (including some of the research in telecommunication network design that is described elsewhere in this proposal) as well as researchers at Bellcore, with whom we have been working for the past two years in developing high performance simulators for personal communication services (PCS) and asynchronous transfer mode (ATM) networks.

It is crucial that the simulation tools provide sufficient performance to simulate large-scale networks with sufficient accuracy. But, simulations of high–speed networks are extremely time consuming on sequential machines. For example, up to three hundred thousand cells[1] may arrive in one second of real time operation of a 155 Mbps ATM link, depending on how heavily the link is loaded. Assuming 10% average utilization, a cell-level simulation of a few minutes of the operation of a network containing a few hundred links will require a minimum of 10^9 simulator events to be processed. This will require hours, if not days, of CPU time on a high performance workstation. Experiments with larger networks or over longer periods of time would simply not be done. Further, stringent Quality of Service (QoS) objectives associated with ATM networks also result in much longer simulation runs (e.g., tens of minutes of real time) to capture enough occurrences of rare-events (e.g., cell losses) to obtain a sufficiently accurate confidence interval.

Similarly, large amounts of computation are required for simulations of PCS networks. Due to limited computation power, simulation studies typically only examine small-scale networks containing fewer than 50 cells [23, 27]. However, Lin and Mak [24] showed that this approach may lead to biased output statistics, and suggest using simulations containing hundreds of cells over tens of thousands of seconds to obtain accurate steady state results. Such simulations will require many hours on sequential machines.

Distributed simulation techniques will be utilized to attack this problem, and forms a major component of the research that will be performed. In addition, we will also examine modeling techniques that significantly reduce the computation that is required. For instance, in [25] we employ burst-level simulations of ATM multiplexors to provide one or more orders of magnitude improvement in speed, but without sacrificing accuracy.

The VTN laboratory will utilize many of the tools and technologies developed in conjunction with the interactive virtual instruments laboratory described earlier. Both utilize tools for collaborative work and interactive steering, and are supported by the underlying networking and state sharing projects. The main distinguishing characteristic (other then the application domain itself)

[1] A cell is a 53 byte fixed size packet.

is the reliance on distributed discrete-event simulation tools on which the VTN laboratory is based.

2.2 Tools Research

We now discuss specifics of the projects in the distributed laboratory research program.

2.2.1 Interactive Program Steering

(Schwan)

Our research focusses on the on-line steering and monitoring of the distributed and parallel programs constituting the components of a distributed laboratory:

- A *distributed laboratory manager* will support the dynamic creation and management of distributed laboratories, so that laboratory components can be 'plugged' together, participants in laboratory experiments can be added and removed, and laboratory sessions can be recorded and replayed (also see Section 2.2.3).
- *Programming libraries* will support (1) the on-line monitoring of arbitrary aspects of distributed laboratories (i.e., of distributed and parallel programs), (2) the on-line steering of distributed instruments, models, and analyses, and (3) the interactive and distributed display and visualization of monitored information and of the results being produced by laboratory experiments.
- An *open environment* for laboratory and experiment monitoring will permit different instruments to be connected easily to each other via shared inputs and outputs, as well as to additional components performing on-line instrument monitoring, experiment (session) establishment and management support, and others. An important issue to be addressed by our work is how such diverse components may be integrated within a single coherent framework so that they can communicate and interoperate[10, 7].

Research approach. Concerning *program monitoring and steering*, we will construct mechanisms and tools for interacting with distributed and parallel programs executing on the multi-granular compute server, such that their on-line monitoring and control are possible. The interactive interfaces should permit end users to continuously assess experiment progress, to steer the experiment's execution toward important data regions, and perhaps, even improve performance by elimination of needless computations. The promise of such *interactive program steering* is that it can make large-scale scientific or engineering computations more accessible to end users by permitting them to rapidly understand computational results (or problems). The additional promise of distributed laboratories is that the domain expertise of multiple users can be brought to bear on solving complex problems. While it will be hard to prove promises of enhanced utility, effectiveness, or performance attained with distributed laboratories, it is also inevitable that fragments of distributed laboratories will be built and that program steering will be performed in the future, in part because scientists now have available to them elements of the proposed national information infrastructure, the means for interactive data visualization, for virtual reality interfaces to programs, and the computational and network power for interactive execution of interesting physical simulations.

The technical content of our research is the development of the conceptual foundation and the technologies required for on-line program steering and monitoring, including (1) formalisms for the description of configurable and monitorable attributes of parallel application programs, (2) portable

tools using these formalisms for on-line program monitoring, configuration, and display, and (3) mechanisms and means for integration of such tools in distributed environments. In addition, we will use known and develop new algorithms for automatic, on-line program configuration, for certain classes of applications and for specific target hardware and operating system mechanisms.

We will answer questions such as: (1) how can monitoring be performed on-line such that the latencies between event occurrence and event detection by the human or algorithm performing steering are minimized, (2) how can we describe and capture the application-specific information desired by end users (e.g., 'heat' variables rather than "processor utilization"), (3) what are the appropriate mechanisms with which such dynamic monitoring and steering may be performed so that performance improvements are attained or functionality is increased without undue performance penalties, and (4) how can monitoring and steering be made 'dynamic', enabling end users to easily change the program characteristics they wish to inspect and/or alter? An issue only partially addressed by our research will be effective user interfaces for program monitoring and steering.

There are no generally accepted mechanisms for on-line and application-specific program monitoring or configuration, so that the diverse systems in existence or being developed must offer their own formalisms and frameworks (a) for capture and analysis of the on-line program information required by individual configurations, (b) for specification of desired program manipulations, and (c) for enactment of specific program changes[1, 5, 3]. As a result, the main technical contribution of our work will be the *development of principles and of a common framework* for both program monitoring and program configuration. Our approach to developing this framework will have three components, each of which are described below.

Uniform model of monitorable and configurable software. The first component is the design and implementation of a uniform model of monitorable and steerable software. The technical approach chosen for specification of program configuration and monitoring is language and operating system independent. Specifically, we assume that implementors explicitly specify both *monitoring* and *configuration* program *attributes*. The framework, then, permits the on-line inspection of monitoring attributes, and the on-line alteration of configuration attributes.

Efficient access to monitoring information. The second component of the framework addresses the program state collected via on-line monitoring, which must be efficiently accessible to algorithms or displays using the information for purposes of program steering. In addition, the programs being steered must be represented such that their steering attributes are easily described and manipulated. Toward these ends, we will develop descriptions, storage, and access methods for the program attributes required for on-line steering.

Open system for inclusion of distributed instruments. The third component of the framework addresses the inclusion of instruments not specifically developed for their use within a distributed laboratory. We will use object-oriented methods for this purpose. In addition, in our current research on interactive atmospheric modeling, we have already developed tool integration methods relying on the exchange of binary files such that 'file' accesses can be directed to local vs. remote files as well as to Unix communication sockets across which such files are transferred in a manner transparent to the accessing programs.

Evaluation and relationship to other projects. The evaluation of the proposed work will proceed in three phases: (1) *internal testing and evaluation* involving implementations and performance evaluation on the multi-granular compute server. (2) *Internal cross-project evaluation*, where on-line steered applications make use of the state sharing libraries and protocols already available and/or being developed as part of the proposed research, which also results in the use of

distributed laboratory applications as benchmarks for state sharing library and protocol research. In addition, by using the tools, libraries, and mechanisms or algorithms offered by the collaborative systems project, on-line program steering will be enriched significantly, by addition of application-independent interfaces between different experimenters jointly performing distributed experiments. As a result, distributed laboratories will also provide a testbed for understanding future collaborative, interactive environments operating on distributed multi-granular computing platforms. (3) Last, we will perform *external evaluations*, where on-line steering and monitoring mechanisms are exported as generally useful libraries to research and user communities via the Internet. This will and has been involving collaborations with industrial partners.

Steps (1)-(3) have already been performed using existing parallel computers (the KSR-2) with initial releases of our software. A monitorable threads package has been exported to a large number of sites in the U.S. and internationally; a release of 'visual threads' is planned via the Internet and to our industry partners for early 1995. Our NASA funding requires similar software releases to NASA sites for future work. Research funding from ARPA (jointly with Honeywell) will result in additional pressure toward making our software available to a large number of university and potential industry partners.

Several comments concerning the steering project's use of supporting technologies appear next. The principal interface of this project with supporting technologies is with the proposed state sharing and communication protocol technologies essential for the efficient representation and transport of information shared across different processors of the single, multiprocessor platform on which steered applications are run, and across multiple, networked machines on which the parallel codes are run concurrently with visualizations and animations of the codes' output and performance data. The former work has already resulted in the construction of a library for sharing typed state first implemented on distributed memory machines [9] and now developed for shared memory machines[2] and for workstation platforms (led by M. Ahamad – see Section 2.3.1). In addition, this work has resulted in the development of communication protocols able to be configured for the efficient transport of typed, shared information described in more detail in [4] and performed jointly with M. Ammar. Additional on-going work concerns the development of a realistic networking benchmark program able to impose the computational and communication loads of future distributed instruments on the underlying platforms in a controlled fashion (joint with E. Zegura and A. Mukherjee).

2.2.2 Distributed Simulation

(Fujimoto and Schwan)

Discrete-event simulation has long played a central role in the design and evaluation of complex dynamical systems such as telecommunication networks, computer systems and manufacturing facilities. Distributed simulation technology is a key and essential facility for the Virtual Telecommunication Networks application. For applications such as this, simulation technology for the 21st century will have at least two key requirements: (1) high performance, necessitating use of distributed and/or parallel simulation techniques to model systems of interesting size and complexity with sufficient detail, and (2) user interfaces that allow human operators to easily interact with the simulator for hypothesis testing, rapid understanding of simulation results, and in some cases, interactions with other operators. Geographically distributed resources, e.g., databases or interactive users, may also necessitate use of distributed simulation techniques.

The distributed simulations required for the VTN application must produce accurate numeric results. This is in contrast to simulations used for training applications (e.g., the DIS environment)

where errors resulting from distribution can be tolerated. The distributed simulators that are discussed here produce identical results as a sequential execution, with only modest constraints (e.g., random numbers must be generated from multiple uncorrelated streams rather than a single stream).

A multi-granular compute server is an essential component for our work in distributed simulation. Compute servers such as this are becoming widespread in engineering environments, where discrete event simulation tools are widely used. However, the execution of distributed simulation mechanisms on these platforms has not been widely studied.

Proposed Research. We have over seven years of experience in building simulations on distributed and parallel computers, especially shared-memory multiprocessors. A key emphasis in our future work is in developing efficient simulations in multi-granular computing environments consisting of shared memory multiprocessors and clusters of uniprocessor workstations.

Adaptive, Optimistic Synchronization. The irregular, data dependent nature of discrete event simulations makes synchronization much more complex than other, more well structured computations. Ensuring that computations (events, in simulation terminology) that depend on each other are processed in the correct (according to the simulated time of the events) order is difficult because it is non-trivial to determine which events depend on each at run-time. Indeed, an event that affects one being considered for execution may not have even been created yet; dependence constraints between events are virtually impossible to determine at compile-time.

Optimistic synchronization mechanisms such as Time Warp [22] detect out of order execution of dependent events (synchronization errors) and recover using a rollback mechanism. They offer excellent potential for transparency and high performance. We have obtained good performance using Time Warp in telecommunication network simulations, including nearly a 38-fold speedup on 42 processors in a PCS network simulation containing 2048 cells (radio ports) and over 75,000 portables (mobile phones) [14]. This simulation model, developed in collaboration with Bellcore [13], is able to simulate up to 30 million phone calls per minute (wall clock time) on a KSR-2.

For the VTN application we are particularly concerned with large-scale (tens or hundreds of thousands of simulator objects) with relatively little computation per event (e.g., hundreds of machine instructions). The aforementioned PCS simulation is one example of such a computation. Simulations of ATM networks also have small granularities, though the number of objects may not be this large. The nature of these computations call for efficient data structures for handling large amounts of data, careful control of memory allocation to avoid excessive memory utilization, and simple execution mechanisms so that the overhead in processing each event is minimized.

Using a multiprocessor workstation, we will continue our work on shared-memory platforms by developing efficient data structures and optimizing execution to best utilize the multiprocessor's caching and memory management mechanisms. In addition, we will extend this work to distributed multi-granular computing platforms where portions of the model execute on the multiprocessor, while others execute on the rack-mounted workstation cluster. A key question is how to offset the higher communication latencies that arise in such configurations. To address this question, we will study the impact of communication latencies on optimistic synchronization mechanisms (e.g., do more rollbacks occur as communication latency increases?), and study countermeasures (e.g., batch processing of events) and evaluate their effectiveness. Our initial experience in utilizing Time Warp in distributed computing environments have been encouraging [13, 17].

An effective optimistic synchronization mechanism must avoid overly optimistic execution (allowing some simulator objects to advance too far ahead of others). We propose an approach to

control memory allocation to prevent such behavior; limiting the amount of memory provided to an object effectively limits how far it can advance ahead of others. We have studied performance memory tradeoffs extensively both analytically [12] and experimentally [15], and demonstrated good performance using only a modest amount of memory beyond that required for sequential execution. Based on this work, we have developed an adaptive flow-control protocol that monitors the simulator's execution, and *automatically* adjusts the amount of memory used by the simulator to maximize performance [16]. We will examine the extension of this memory management to multi-granular distributed computing environments. Issues that must be addressed include the lack of a global memory pool and high communication latencies. New, efficient memory management protocols for distributed memory systems will be developed in this project.

Complex simulators may consist of separately developed continuous and discrete simulation programs that are combined into a single, integrated system. For example, one could envision adding continuous models for signal propagation to our discrete PCS network simulation. A key issue in these systems is to develop techniques to effectively integrate different synchronization mechanisms, particularly rollback-based mechanisms such as Time Warp with time-stepped simulation mechanisms. Open research questions regarding these *federated simulators* include: What is the best approach for integrating these protocols to maximize performance? Should one or more protocols be modified to accommodate interaction with other protocols? If so, how? What is the nature of the interaction of disparate protocols and how does one impact the performance of another?

Mapping and Load Balancing. The mapping of large-scale simulations to multi-granular platforms is an open question. Load-balancing algorithms will be needed for networks with dynamically changing traffic flows as well as for simulators where good static mappings cannot be easily obtained because traffic patterns are unknown. The interplay between the mapping/load balancing algorithm and synchronization mechanisms is not well understood. It is widely recognized that a poor mapping algorithm can lead to catastrophically poor performance, but it is not known how "carefully" one must balance the workload to achieve acceptable performance; for instance, it is not known how frequently the workload must be re-balanced in large-scale, small granularity simulations.

Load management in optimistic distributed simulators must use different metrics for balancing decisions than conventional programs. Processor utilization is a poor metric because it does not take into account processors that are experiencing an excessive amount of rollback, or are advancing too far ahead of other processors. In both cases, the processor should receive more of the simulation workload, even though its utilization may be very high. Metrics such as effective utilization [26] account for rolled back computation, but not overly optimistic execution. Further, it is an open question as to whether standard load balancing algorithms (modified to utilize different metrics) are effective for optimistic simulators, or whether entirely new algorithms are required.

We will initially study these questions in the context of shared-memory multiprocessors, and then expand our work to the more complex question of multi-granular computing platforms. In the latter case, it is clear that communication latencies play an important role in load management decisions.

Interactive simulations. Simulation models that include embedded physical devices (e.g., real-time simulators may include portions of an actual network, and simulation models for other components) or interactive components necessitate that the simulator be able to interact with an external environment in a timely fashion to satisfy certain hard and/or soft real-time deadlines. This is a particularly challenging problem for optimistic simulations because rollbacks (which are inherently unpredictable) must be factored into schedulability computations. A key observation is that one must be able to characterize or limit the effect of incorrect computations (incorrect in the sense

that they will later be rolled back) in optimistic simulations. In recent work [21], we have shown that parallel simulation protocols with limited optimistic execution show good promise in providing both high performance and predictability. We will use the requested equipment to continue to explore this issue in multi-granular computing platforms.

A related problem concerns storage reclamation in optimistic distributed simulators. This function, commonly referred to as *fossil collection* is analogous to garbage collection in symbolic computing systems. Fossil collection in large-scale simulations containing hundreds of thousands of simulator objects is time consuming, and represents a significant roadblock to interactive use. We propose to attack this problem by using "on-the-fly" fossil collection where memory is reclaimed incrementally as it is needed, rather than all at once during a periodic fossil collection procedure. This is possible because each block of memory contains a timestamp, and memory with timestamps less than global virtual time (a computed value indicating a lower bound on the timestamp of any future rollback) can be reclaimed. The use of timestamps to determine which memory can be reclaimed distinguishes fossil collection from garbage collection in symbolic computing systems.

Another important application of rollback is in interactive steering of computations, whereby one might run a simulation until a steady state or some interesting event occurs, and then modify certain system parameters to experiment with different scenarios. This is precisely the functionality we need for implementation of simulations that may be used for telecommunication network control, where a discrete event simulation may be used for experimentation with alternative scenarios in response to exceptional network or traffic conditions. In such simulations, the built-in rollback mechanism can be used to reset the simulator to some previous point in time to begin a new experiment.

Shared state and space-time memory. Some simulation models are most naturally represented as collections of simulator objects migrating over a shared state space. For instance, in a PCS network simulation, the objects are mobile units, and the shared state space is the physical terrain over which the objects move. Shared state is seldom allowed in distributed simulations because at any instant, the simulator objects are usually at different instances of simulated time, e.g., one object might have been simulated up to 10:00 AM, while another may have been simulated up to 10:15 AM. If both objects are simultaneously observing a state variable, e.g., the number of channels currently in use in the radio port, some method is necessary to ensure that each object sees the version that existed at its current simulated time.

Space-time memory is a memory system with a two-dimensional addressing structure. Memory accesses specify both the *spatial address* (similar to conventional memory addresses) to indicate the variable being accessed, and a simulated time for the access [18]. An optimistic (rollback-based) synchronization mechanism is used to ensure that each read access returns the most recently written value in simulated time. For instance, in the above example, if the object at time 10:15 reads the channel value prior to the object at time 10:00 modifying it, the former object will be rolled back and re-executed using the correct value. Previously, we have studied space-time memory in the context of shared-memory multiprocessor systems [19]. We will extend this work to distributed and multi-granular computing platforms.

Relation to other Projects and Equipment Needs. The space-time memory work will utilize results (and software) from the shared-state project. Global computations such as global virtual time (used for fossil collection) can exploit results from the multicast research described later (Section 2.3.2). As noted above, there is also a natural synergy between the interactive steering project and the rollback-based synchronization mechanism used here. The interactive and real-time simulation work described above is already a collaborative effort between Fujimoto and Schwan

[20, 21]. Last, it is clear that once program steering can be effectively provided to a single user, the next logical step is to make this capability available to multiple users at different locations using software developed with the proposed collaborative work project.

A multi-granular compute server (including the multiprocessor and rack-mounted machines) is an ideal platform for the research in the distributed simulation project. The workstations for the collaborative work project and for home use provide excellent vehicles for interactive use of the simulators.

Evaluation. Evaluation of the work will be performed both by our group as well as by end users. At the system level, our goal is to obtain performance within each processor equal to at least 90% of an efficient sequential simulator, and rollback efficiency exceeding 90% (i.e., fewer than one in ten computations rolled back) for benchmark applications that simulate telecommunication networks (both wireless and ATM networks), enabling our system to yield speedups of $0.8N$ or more when using N processors to simulate large-scale applications. Simulation validations will be performed, at least in part, by comparing performance predictions with operational networks and existing commercial simulators.

Related Research. The research on this project extends prior research on optimistic synchronization of parallel simulations in several ways, including: combining memory management protocols with mechanisms to prevent excessive rollback in adaptive simulation protocols [16] and integration of different synchronization mechanisms, mapping and load management issues for networked multiprocessor platforms, developing mechanisms to exploit optimistic execution in the context of real-time simulations [21], and combining interactive steering with parallel simulation techniques.

2.2.3 Collaborative Systems Infrastructure

(Hudson, Ezquerra, and Potts)

The collaborative system testbed proposed here forms a central bridge between several other aspects of this project. On one end, it provides an interactive base upon which the proposed driver applications can be built and tested. On the other, it provides a customer for use and testing of the underlying system components of network infrastructure and shared state maintenance in a practical and demanding setting.

To support the prototype instrumentation laboratory and virtual telecommunications network simulations (as well as similar general class of distributed interactive applications), we propose to construct an open and extensible system structure for composing and controlling collaborative tools made up of small semi-independent programs. This structure, in addition to providing a base for our testbed applications, will make use of several of the proposed system capabilities described later, and will serve as a solid experimental testbed to explore the use of these technologies in realistic, practical, and demanding circumstances.

The key to this testbed structure will be its open nature – allowing a number of relatively small tools to be used together – along with some key system technologies needed to facilitate smooth sharing, communications, and user interaction. On top of this infrastructure base, we propose to develop and/or adapt several general application level tools, as well as our specific proposed applications, in order to both test the infrastructure and enable the types of distributed collaboration illustrated above.

In the setting of collaborative applications, it is convenient to structure programs around shared data – viewing user actions as manipulations of pools of shared data and system presentations as

views of that data. Because of its advantages in supporting direct manipulation interfaces [29, 30] this approach has been advocated in the user interface software community for single user interfaces for some time. We believe this approach will be particular advantageous in collaborative interfaces, and that we will be able to exploit new system capabilities for providing shared state to make such systems practical in a distributed setting.

Proposed Research. Central to the collaboration infrastructure we are proposing will be a *user interface server*. A copy of this server will run at each participant's local workstation and accept small scripts, which when executed by the interpreter which makes up the heart of the server, will create part or all of a user interface. In this regard, what we are proposing can be handled by existing tools, most notably TCL/Tk [28]. However, our work will extend these capabilities in several significant ways. First, as indicated earlier, interfaces will be strongly based on the notion of shared data. Interfaces will be seen primarily as interactive mechanisms to allow an end user to manipulate shared values (rather than the more typical approach of treating user interfaces as a way to let the user execute pieces of code).

To accomplish this, the UI server, in addition to making heavy use of shared state maintenance techniques proposed here, will also use techniques for user interface *composition*. These techniques treat user interfaces as hierarchical structures where parent objects are responsible for composing the smaller user interface objects defined by their children into larger interface objects. The simplest types of these compositions include things such as rows and columns which simply place their child objects in simple arrangements. However, this general technique can be used to achieve a wide range of more sophisticated effects (see for example [31, 32, 33]). For this work, we will be most interested in composition techniques which can be performed in an automatic or semi-automatic fashion.

In addition, this research will pursue new techniques for supporting transitions between synchronous and asynchronous interactions. For example, support to allow late-joiners to quickly review the discussions of a meeting before actually connecting to it, and techniques to allow better use of recorded video and audio information to review results after a collaborative session is complete.

Video and audio recordings often contain the richest sources of information about what has happened during an interactive setting. Unfortunately, the information contained in audio and video is sequential in nature, and very difficult to index. One of our central goals in supporting transitions between synchronous and asynchronous interactions will be to improve the usefulness of video and audio recordings by allowing them to be indexed in some from.

Indexing of actual video images or audio signals is very difficult. To overcome this difficulty, we are proposing to use ancillary information to form an index. In particular, we propose to record other interactive actions – particularly the use of telepointers and other interactive devices along with information about the context in which they operate (e.g., which document or shared object a telepointer refers to) – and use this information as an index back into the more informal information contained in audio and video recordings. For example, after a collaborative session we might wish to review the discussion of atmospheric data in a particular geographic region. We could retrieve much of this discussion using a query that selects recorded telepointing actions (recorded as interactive *events* in an *event stream* associated with the interaction) which occurred over a visualization of that region. The recorded timestamps from those events can then be used to retrieve and replay selected portions of the video and audio recording.

In general, asynchronous participants will be able to search to find periods of time that are likely to contain discussions of interest to them by formulating queries against the event stream. These queries can be formulated in terms of any sort of recorded facts such as: what data sets were being displayed, what areas of various displays were being telepointed to, and who was actively participating (manipulating interactive components) in the discussions. In addition to automatically recorded interactive events, it may also be possible for the original participants to provide additional markers or annotations within the event stream, for example by pressing simple buttons labeled "remember this," or "that's interesting."

The results of a query against the event stream will be one or more time intervals. These time intervals can then be used to control the playback of the full recording with a VCR-like control interface. In addition to simple playback (e.g., of an audio stream) in some cases this will involve gathering recorded information (such as changes to shared values) in order to recreate displays that were not recorded as a whole.

To realize recording, indexing, and playback of collaborative sessions, we will assume that every media type and interactive program involved in collaboration (or at least the part of a collaboration that we will record and playback) will be able to make a recording of its actions. These will either be a direct recording of results as presented to the user (for example, we would initially record video and audio with a relatively simple computer controlled VCR), or a series of critical values, which, when combined with the original datasets involved, are sufficient to recreate the original presentations. Each of these media recordings will be accessed only on the basis of time. In addition to these heterogeneous recordings, an additional recording of all user interface actions – the event stream – will be made by the user's various user interface servers. As described above, this event stream will serve as the primary content-based index. Although this plan presents a potential synchronization problem at playback, our experience shows that as long as the audio and video streams are tightly synchronized (typically by being recorded and played back via the same device), considerable timing skew can be tolerated between other media – perhaps as high as 750ms.

Evaluation Plan. To evaluate the effectiveness of the techniques proposed here we will follow an overall two stage plan. Our first objective will be to test whether the system can in fact be constructed, and more specifically to discover where the difficulties and bottlenecks lie, particularly with respect to performance and function of the underlying system components. This will be done in the context of building prototypes for the proposed collaborative applications. Our evaluation at this stage will be aimed both at validating and refining the architectural concepts of the collaborative testbed, and at providing feedback to the underlying system layers about performance and functionality needs. Our goal in this first phase will be to simply to build functional prototypes of collaborative applications.

In the second phase we will begin to evaluate the usability aspects of these applications. To do this, we will draw on the considerable expertise in usability testing found in the Graphics, Visualization, and Usability Center (for example, we have an existing usability testing facility with one-way glass, remotely controlled video cameras, recording and mixing equipment, etc. that is regularly used for testing of this type). Here, our goal will be to discover aspects of the collaborative testbed that directly impact usability for the end-user, and translate these into implications for the underlying system components. For example, we would expect to discover which performance aspects actually impact usability (and hence should perhaps be given high priority for additional work) and which do not.

2.3 System and Networking Technologies

2.3.1 State Sharing

(Ahamad and Ramachandran)

Information sharing via shared state is pervasive throughout the envisioned distributed laboratory environment. As discussed in Section 2.2.3, interactions between humans in the collaborative work project are based on a paradigm utilizing a hierarchy of shared objects. Similarly, shared state is utilized to realize interactions between users and programs, and between program components in the interactive steering work (Section 2.2.1). As mentioned in the distributed simulation project (section 2.2.2), abstractions for shared state for discrete-event simulations facilitate program development.

Many different types of information are shared with differing requirements in terms of latency and consistency of shared state. For example, when a scientist is "watching" the visualization of a simulation that is running on a remote machine, strong consistency of the shared state between the visualization and simulation is not necessary. This is because it is acceptable if the visualization is somewhat delayed from the actual state of the simulation since the simulation state constantly evolves. On the other hand, strong consistency may be necessary between the components of the simulation executing on different processors. The "virtual meeting" applications supported by the collaborative work system also require support for state sharing. The shared state in such a system includes video and audio information, visualization output, and meta-data that is used to index into the recordings of the meetings. Again, consistency requirements differ for these different types of shared information. The requested infrastructure provides an essential testbed for implementing and experimenting with state sharing under realistic workloads.

Existing state sharing technologies range from shared file systems to software supported distributed shared memory (DSM) abstractions. Although these technologies can support initial prototypes of distributed laboratories, we claim that they cannot meet either the consistency or the performance requirements of such applications. This is mainly due to the fact that these state sharing systems do not exploit application semantics in maintaining consistency of shared state. For example, a distributed file system can be used to share data sets but it does not provide the level of consistency that is needed when the data sets are read and written concurrently at different systems. The Sprite file system provides strong consistency by disabling caching when there is a writer but that could limit performance. The sharing of state between program components differs significantly from the coarse-grain sharing that occurs between participants in a virtual meeting.

Our goal is to explore state sharing techniques that meet both the consistency and performance requirements of distributed laboratories. We will explore how application semantics captured by attributes such as the level of consistency and granularity of sharing can be used in developing efficient state sharing systems.

Proposed Research. To reduce access latency and communication overhead, shared information must be cached. The caching of shared information introduces the problem of consistency among its copies that are stored at different nodes. We will use several techniques to investigate how an appropriate level of consistency can be provided efficiently. First, we will use alternate notions of consistency which are appropriate for the applications being considered.

Weak consistency. The performance of protocols that allow data to be shared across nodes critically depends on the degree of consistency that needs to be provided for the data. In the distributed laboratory applications, it is possible to weaken consistency requirements to provide better per-

formance. *Causal consistency*, which captures causal relationships between data accesses in a distributed system, provides the level of consistency that is appropriate in many of these applications. For example, if Dr. Jones (in the scenario described earlier) makes a change which results in changes in the simulation execution and also in the visualization at Dr. Smith's workstation, causal consistency ensures that changes in the visualization become visible before the changed output is received from the simulation program. Causal consistency not only meets the consistency requirements of many applications but can also be implemented efficiently because it allows both read and write operations on cached data to complete without requiring communication with other nodes. This can be done even when other nodes are accessing the data concurrently. The changes made to shared state at a node can be propagated to other nodes asynchronously.

We will explore scalable solutions for providing causal consistency. We assume a logically shared object space where nodes cache copies of only the objects that have been accessed in the recent past. One problem is how to detect if copies of data objects cached at a node are mutually consistent with respect to causal consistency. Also, to ensure that changes to shared objects propagate to other nodes, we want to develop schemes that do not require any global synchronization operations, but rather, require a more limited multicast operation. For example, a node should be able to propagate a change to its neighbors which can then propagate it further to other nodes (such schemes are used for propagating routing information in distributed routing protocols). The key problem is how to maintain ordering information between updates and apply them in a consistent order at the nodes that cache shared data. We will also explore techniques that do not depend on broadcast or ordered communication.

Multi-granular compute servers introduce new issues with respect to maintaining consistency. Strict, hardware enforced consistency may be automatically provided within the multiprocessor, but this property is lost as soon as the shared state migrates beyond the multiprocessor's shared memory. It is not clear how one can seamlessly support multiple consistency mechanisms among concurrent programs in a way that is efficient and easy to use.

We will implement the new algorithms as user level libraries on the requested infrastructure. Currently we are exploring a CORBA compliant framework that will allow applications to benefit from caching without any change in programming. Our system will also provide support for shared objects that offer different degrees of consistency. For example, in the shared visualizations application, a visualization is driven by the state of objects, and these objects should be kept causally consistent. Our algorithms will detect when accessed objects are not locally cached, fetch their current copies and ensure that copies of locally cached objects remain causally consistent when a new object is brought to a node.

Program-driven consistency. To support program state sharing we are exploring the use of compile-time and runtime information in conjunction with the inherent synchronization in the application to reduce the overhead of consistency maintenance for shared data structures. The research issues here are performing compile-time data flow analysis of the synchronization regions in a parallel program, and development of efficient primitives to be implemented in the operating system for enforcing consistency at runtime when so-directed by the flow analysis. To enable the sharing of resources between the components of a distributed application, we are exploring the development of appropriate system abstractions. In particular, a combination of message-passing and shared memory semantics would be appropriate depending on the locality properties of shared resources.

The notion of shared state takes on a new, additional meaning in certain distributed laboratory applications. Consider individual processors in the multi-granular compute server, where each is generating a specific portion of the global system state, as is the case in many distributed

simulations. Different simulation models may require different views of the same portion of the global state. These views must remain consistent when changes to the state occur. The traditional view in distributed shared memory systems is clearly not appropriate for this problem.

We define a new abstraction that seems more meaningful in this setting, namely, *versioned shared memory* (VSM). A request for a copy of an object may create a new version. Consistency across versions of the same object will be enforced as needed using the attributes of the object. Techniques used in database systems to enhance concurrency may be brought to bear here. We have been conducting some work along these lines where we use DSM as an abstraction for supporting the organization of distributed databases. The issues we have explored so far include enhancing client-server database architectures using DSM, increasing transaction throughput and scalability through DSM techniques [51], enhancing concurrency and reducing crash recovery times through versioning techniques [52]. The research challenge in defining the semantics of VSM in the context of distributed laboratory applications is in exploiting some of the well-known database technologies in this domain without any of the usual rigidity of database systems such as serializability.

Previous Research. We have undertaken several research projects in areas related to state sharing [53, 54, 55, 38]. One project addressed the implementation of shared memory abstractions on workstation networks. It addressed the formal characterization of shared memory systems [34, 48], and also implemented consistency protocols derived from a number of approaches that have been proposed for building high performance distributed memory systems [38]. For a comprehensive evaluation of the coherence approaches represented by the different protocols, we used applications that capture a range of data sharing patterns. The results of our studies led to important insights into the operation of DSM protocols for scientific applications. We have also further explored the benefits of weaker consistency provided by causal memory [37, 49, 36], and explored the programming of causal memory for various types of programs [35]. Our detailed performance studies have shown that for many parallel applications, causal memory significantly reduces the time the applications spend in state sharing related activities.

We have extended the DSM work to address cache consistency issues in a tightly coupled shared memory multiprocessor systems as well [57, 58, 59]. We have also researched the kinds of architectural support that are pertinent to support efficient message passing in a distributed system [60, 61, 62]. This combined system experience of addressing the system issues for both "message passing" and "shared memory" computation in a distributed setting gives us a unique edge in being able to answer some of the challenges involved in devising the system abstractions in a heterogeneous environment. In a related NSF-funded project, we have developed a framework and a comprehensive set of tools for scalability studies of parallel systems [63, 64]. At the heart of the framework is an execution-driven simulator called SPASM which identifies, isolates, and quantifies the overheads that occur in a parallel system. Further, we have also implemented a suite of parallel applications drawn from several different domains including scientific, image understanding, and combinatorial optimization, on a variety of parallel architectures including KSR-2, MasPar MP-2, Intel iPSC/2, and PVM-based workstation clusters, to enable this scalability study [65, 66]. This framework will be useful in this evaluation process for the system abstractions for state sharing.

We have also investigated protocols that can be used to maintain the consistency of replicate data such as files. Our work in replicated data management has ranged from development of new protocols that provide a higher degree of load sharing to techniques that can be used to evaluate the performance of various protocols [50, 43].

Other Related Research. Data sharing techniques have been investigated widely in distributed systems. For fine-grain sharing, many implementations of distributed shared memories (DSM)

exist. These include Ivy [45], Munin [39], Treadmarks [41], Midway [46] and several others. Weakly ordered memory systems have also been proposed [40, 44, 47] which have led to multiprocessor cache coherence protocols that guarantee memory consistency only at well defined points in a program. We feel that exploitation of weaker consistency that meets application needs and the structure of shared state are the two key aspects of our proposed work which have not been explored by past work.

We have claimed that causal consistency is natural for many of the interactions in driver applications. Causality has also been investigated widely in distributed systems. Although the ISIS causal broadcast [42] can be used to update replicated data, this represents a particular policy for handling replication. Since the choice of a policy depends on application requirements, we plan to explore techniques which allow objects to be cached and uncached at a node dynamically without executing multicast group membership change protocols.

Coarse grain sharing can be done via file access. Many distributed file systems exist but they either do not provide the desired level of consistency (e.g., the NFS file system) or restrict caching that could impact performance.

2.3.2 Network Support for Distributed Laboratories

(Limb, Ammar, Mukherjee, and Zegura)

It is clear that a versatile, high-performance, communication infrastructure is essential to the future success of distributed laboratories. Transport facilities must carry a complex mix of traffic types in an integrated manner. When sharing visualizations, multicast communication must be efficiently performed, while also guaranteeing some latency requirements imposed by human end-users. Further, when supporting the real-time transfer of video and audio, protocols must be able to meet varying quality of service demands as users' requirements change (e.g., when actually looking at a remote collaborator vs. when simply having a collaborator's picture in the background). Finally, the service should be ubiquitous; collaboration should be possible from the office, laboratory, remote field station, on-the-road, and importantly, from home.

The specific issues we propose to address in the provision of a networking infrastructure for distributed laboratories are: (1) high-speed ubiquitous networking connectivity between industry, lab, home and community, (2) efficient multicast transport for data representing multiple media types with varying quality of service requirements (e.g., latency and reliability), for maintaining consistency of shared state (see section 2.3.1) and global operations for distributed simulations (see section 2.2.2), and (3) the accurate modelling of the traffic patterns that are generated by the distributed laboratory applications so that we can study how to better manage and utilize the network resources.

The Ubiquitous Network. The ubiquitous network provides the physical network on which the distributed laboratories (and the proposed virtual telecommunication network laboratory) will operate. Within industry and academe high-speed network connectivity is usually available. This may take the form of DS1 (1.5 Mb/s) or DS3 (45Mb/s) connectivity to either a private network or to a public packet network such as the Internet. While these speeds are adequate for many purposes, higher transmission rates will be required to fully extend distributed laboratory environments to the home.

When we move into the home or the community at large the picture changes and the highest communication channel possible may be a 9600 baud MODEM, or if one lives in certain locations, 128 Mb/s may be possible via ISDN. While higher transmission speeds can be obtained over tele-

phone lines, it is relatively expensive to do so and most telephone companies together with the cable companies are looking toward some mixture of fiber and coaxial cable as the most attractive means to achieve larger bandwidths to the home.

This research will focus on providing higher bandwidths, inexpensively, wherever the virtual laboratory may reside. While much of the networking research described below focuses on providing faster, cheaper ATM connectivity in the workplace, cable technology appears to have the greatest potential for extending this capability to the home.

High-speed Data Over Cable. Today fiber/cable plant covers about half all residential areas in the U.S. and the coverage is continuing to increase. Techniques are being developed to enable this channel to support high-speed digital communications. On both the downstream, broadcast channel and the upstream channel shared access is naturally provided by the physical nature of the medium. However, while in some ways it is similar to a satellite channel and in other ways similar to a dual bus, it is unique. Efficient exploitation requires that the channel access protocols be tuned to this medium. Our investigations so far suggests that a centrally controlled reservation protocol, with random access to a separate reservation channel appears most appropriate. Such a mechanism can efficiently support stream traffic such as voice and video and bursty traffic such as file transfers and scientific data transfer. Our plan is to emulate these protocols in software and then implement them within the laboratory on fiber/cable plant. If successful at this level we would collaborate with the cable/telephone industry to evaluate the system in a field trial.

There is very little other research ongoing in this area that is public. The one piece of research that speaks directly to this area concerns a protocol called DQRAP and XDQRAP [67, 68]. Our approach uses similar protocols but it directly accounts for the limitations in the physical medium that we do not see addressed in the above work. For example, a station cannot reliably read any signal originated by another station on the network apart from the head-end. There are a limited number of products on the market. They apply the Ethernet protocol to the cable topology, a rather inefficient way to use Ethernet.

Multicast Communication. A multicast communication service is one that is called upon to deliver copies of the same information to multiple recipients. It is a primary form of communication in distributed laboratories. There are many approaches to providing a multicast service in a network. Current thinking favors explicit support for multi-casting at the network layer. In particular, explicit support for multi-casting has been defined in the context of the Internet Protocol (IP) [69]. The presence of explicit support for multicast communication has allowed multi-casting to become an efficient way to simultaneously communicate with multiple destinations.

Adequate support for multicast communication, while providing economy of scale, has the effect of detaching a multicast source from the actual cost of the traffic it injects into the network [2]. In typical scenarios the effort a source exerts in generating a multicast packet is much smaller than the cost incurred in transmitting and/or receiving this packet. It is this "lack of scale awareness" effect that makes multicast communication such a desirable paradigm. It is also precisely this same effect that has the potential of making large scale multicast communication hazardous to the network, to the multicast source, to the destinations, or to all three. A good example of this is the *implosion problem* where a multicast message to a large size group can result in a large number of returned responses that can congest a network and flood the source's receive buffer.

Another negative effect of multicast communication is its inability to respond to the individual needs of specific members of a destination group. We would like to group various communication

[2] A good example of this in a different context is in the use of electronic mailing lists.

needs to make use of the economies of scale [3] provided by multicast communication. We are obliged, however, to insist that all destinations share the same requirements and properties. An example of this is the multicast of a video stream to receivers with heterogeneous capabilities and requirements.

Based on the discussion above, the multicast paradigm can be said to possess a "split personality." On the one hand, it is a paradigm designed to offer immunity from scalability to its user, but on the other hand, this immunity makes the network much more susceptible to being overloaded. On the one hand, it allows for potential savings by providing economies of scale, but on the other hand, this saving disallows any special treatment of a member of a destination group.

Our work in this area aims at understanding the two sides of this personality and the tradeoffs that exist between the benefit of providing the economy of scale inherent in multicast communication and the negative effects this can have. We propose to study and apply a suite of techniques designed to help understand and deal with the tradeoff discussed above.

Individualizing a multicast service. This technique encompasses procedures that can be used to provide a certain level of individual treatment of a multicast destination while maintaining economy of scale. We consider two approaches. Our first technique is dubbed *destination set splitting* where it is allowable to split a single multicast group into several smaller subgroups. The source, then, carries on multiple multicast conversations, one with each subgroup.

With this technique some economy of scale is sacrificed in order to ensure fairness of operation and in order to deal with specific user needs. This technique introduces some added complexity to a multicast source's operation. Our second technique adds complexity and buffering to a multicast receiver to allow for individual receiver functionality. We apply these techniques to video distribution and video-on-demand applications, an activity that will be prevalent in the distributed laboratory environment. Examples of our work in this area can be found in [70, 71, 72, 73].

Multicast addressing. Multicast IP uses *group addressing* where nodes that belong to the same multicast group are made to recognize a single multicast address. A source sending multicast packets to the group uses the multicast group address as the destination address. Whereas the multicast group concept is fundamental to multi-casting, group addressing is one alternative among many for identifying and routing to multicast groups.

Many of the scenarios we envision in the distributed laboratory application require transaction-oriented, short-lived multicast connections. In such an environment, the overhead of dynamic multicast address assignment and routing cannot be tolerated. A preassigned addressing scheme is required where multicast addresses are assigned to groups for prolonged periods of time. Clearly it is not feasible to assign one address per subset of hosts, as the number of addresses required is 2^n, where n is the number of hosts. We are studying alternative techniques for multicast addressing that support short-lived connections with reasonable bounds on the size of the multicast address space, and minimal additional bandwidth and processing overhead.

Multi-casting with application semantic knowledge. In some instances, knowledge of application semantics (i.e., knowledge of what the application is trying to accomplish with a particular multicast) can provide guidance in dealing with large scale multicast. Specifically this knowledge can be used to avoid the aforementioned implosion problem. For example, multicast communication can be used to provide an "Any-casting" service [74] where only one response from a potentially large set of destinations is desired. Clearly this knowledge can help in limiting any negative effects

[3] That is, the cost per multicast receiver decreases from the source's or the network's viewpoint as the size of the multicast group increases.

associated with the size of the destination group. *Probabilistic Multi-casting* [75], is one technique that we have developed in this context.

An Architecture for Providing Reliable Multi-casting. Reliable transport layer multicast communication is particularly troublesome in the context of existing Internet protocols. Although IP supports multicast communication, TCP is an inherently single destination protocol. Most work to date in this area has started from the premise that in order to perform reliable transport layer multi casts, a new and somewhat different multicast transport protocol will need to be developed.

We propose an approach that would keep TCP as the heart of a multicast transport protocol. A *single connection emulation* (SCE) protocol is introduced between TCP and (multicast) IP. It mimics a single destination IP interface to TCP and use the multicast IP interface. The SCE will also provide services to the application layer for the application to be able to control several aspects of the multicast connection including security and reliability requirements.

There are several advantages to this approach. First, TCP is a well understood and mature transport protocol and IP multicast is gradually gaining a level of maturity. This makes it highly attractive to maintain them as components of a reliable multicast transport protocol. Second, the modularity aspect of this architecture will allow any enhancements of TCP and multicast IP to be automatically incorporated into the workings of the reliable multicast transport protocol.

The SCE layer is the place where we propose to include support for individualizing a multicast conversation and incorporating application semantic knowledge. A preliminary discussion of this concept can be found in [76].

Quality of Service Guarantees and Resource Management. Traffic measurements from today's medium speed networks and emerging high speed networks form the basis of our research on resource management studies and quality of service guarantees for multi-media applications. Recent studies have demonstrated that (i) actual network traffic over LANs and WANs exhibit long-range dependence characteristics that cannot presently be accounted for by traditional modeling approaches[78, 80, 82, 83, 85], (ii) the long-range dependence nature of network traffic can have a dominant impact on network performance[77, 79, 81], and (iii) long-range dependence impacts practically all aspects of network engineering and design. As an example, our results show that the ratio of mean queue lengths from measured traces to that of an M/M/1 queue can be as large as 7500 for an utilization as low as 0.2, the actual numbers depending on the long-range dependence parameters of the data set.

Computer and communication networks have traditionally been designed using Markovian models for buffer-sizing and capacity planning decisions. This is because of the well known theorem that states that multiplexing a large number of independent "customers" leads to a Poisson arrival process. In packet-networks, this is emphatically *not* true, even for large wide area multiplexors such as NSFNET core switches: there do exist independent customers at the user level (or TCP application level); however, each such "customer" produces a series of packets at a widely varying rate before it is done.

This has serious implications to the overall arrival process, and the resulting performance. Our measurements and analyses show that the primary cause of the extreme burstiness (and a possible "self- similar" nature of traffic at different time scales) is the heavy tail in the probability distribution of the number of packets transmitted, and the rate at which they are transmitted. The first appears to match a stable Pareto distribution which has an infinite variance ("heavy tailed"). The second variable is highly correlated with the first. Together, they create a burstiness phenomenon that does not die out even in large multiplexors such as a router in the NSFNET

backbone, where 1000+ simultaneous active flows are not uncommon[82].

Several first-generation ATM switches have been designed with Markovian input models, with buffer sizes in the range 10-100, exactly as Markov-based models would recommend. However, recent articles in the trade press (e.g., [79]) have made it clear that something must have gone terribly wrong; when deploying these switches in the field and exposing them to "real" traffic, cell losses far beyond normal were experienced and resulted in costly redesigns of the switches and loss of market shares for the vendors. Knowing about the self-similar behavior of "real" traffic would have largely prevented these vendors from going through this recent experience.

We have created a laboratory environment for collecting and studying traffic on campus LANs and the NSFNET backbone (the latter in collaboration with Merit and IBM). Based on the statistical properties of the NSFNET data, a packet-level simulation model for the NSFNET has been created on top of a commercial simulation package (BONeS). Currently, this model can generate temporal correlations in packet-level traffic on the NSFNET. A model with spatial correlations of the NSFNET (suitable for studying routing issues) is currently in progress. Portions of this model have now been realized on our distributed simulation system.

As an example of the use of real traffic data for performance evaluation, an optimal, but NP-hard, graph algorithm with application to networking had several heuristic alternatives. Spatial data on a complete set of {(source, destination)} pairs was available to us from the NSFNET which happened to be a potential application environment for the algorithm. We, therefore, used the data to evaluate and compare the performance of alternative heuristics with the known optimal (NP-hard) algorithm[84] and determined that one simple heuristic based on simulated annealing could result in a performance close to the optimal for all traffic data sets that were available to us.

The proposed research will be concerned with investigation of (i) statistically accurate workload models and (ii) algorithms for congestion-control and resource-management for providing guarantees on quality of service to applications. The distributed laboratory applications that will run on top of the proposed infra-structure will provide data on interactive, real-time, multi-media, multicast applications. These will be valuable additions to the simulation model. The resulting platform will enhance our studies on protocol performance in broadband networks.

Faster, Cheaper ATM. As mentioned above there are a number of open problems involved in applying ATM technology, particularly in distributed laboratory applications. Here, we focus on the cost of implementing the technology. Today the cost of a switch is approximately $1000 per 155Mb/s port for the switch alone for the latest generation of switch. On top of this there are the processor interface costs and the optical communication costs. In conjunction with the new NSF Engineering Research Center in Low-cost Packaging at Georgia Tech, we are exploring the opportunities provided by multichip module (MCM) technology to implement switches with better performance at lower cost. MCMs permit larger I/O pin densities, higher speed and denser interconnect and the potential for lower cost than VLSI circuits mounted on printed circuit boards. We are exploiting this work by tuning the architecture of an ATM switch to the inherent advantages of MCMs. Our initial work indicates that a combination of bus based and memory based architectures appears optimal for medium sized switches. We plan to extend the work to other elements of ATM systems. The proposed virtual telecommunication network facility provides an excellent platform for experimenting with such architectures because long simulations are needed to model even modest sized switches.

Bringing It All Together. As the different networking research areas produce results they will be brought together to provide an efficient, cost-effective network that provides a seamless framework

that permits the distributed laboratory to be created wherever the researcher may be located. By addressing the problems associated with multicast and multi-media service, by understanding, and being able to accurately model the traffic, and by creating a ubiquitous network we are attacking three fundamental problem areas that stand in the way of fully exploiting telecommunications. In the longer run, however, the larger impact will likely be in the business environment where executives, knowledgeable workers or sales representatives want their office to move with them, whether it be to another company location, to a hotel room, or to their home.

3 Conclusion and Impact of Research

The proposed research program develops and integrates new technologies that span the range from applications to middle-ware software tools to the required underlying system and communication technologies. At the same time, we will experiment with these technologies in the context of a computing/communication infrastructure spanning a wide range of computing capabilities (from multiprocessors to individual CPUs) and communication speeds (high performance ATM LANs to home computing environments). The resulting multi-granular nature of this infrastructure presents unique opportunities concerning the development of effective techniques that can cover this broad spectrum, particularly as distributed laboratory environments extend into the home, and must necessarily contend with lower bandwidths and higher latencies. Equally important is the fact that the proposed work is driven by large-scale applications that require the technologies we will be developing. This enables us to test our ideas and promote our results in realistic settings.

There are numerous ties between the individual research projects, many of which have been discussed throughout the proposal. A few of the these linkages include:

1. Mechanisms used for synchronization in the distributed simulation project are leveraged with interactive steering research to provide sophisticated simulation environments for experimenting with virtual instruments and yet-to-be-realized systems.

2. Research to develop effective implementations of shared state is leveraged in the collaborative work project to realize flexible, extensible, system architectures for human-to-human communication. Other linkages between the shared state and interactive steering and distributed simulation projects explore human-to-computer and computer-to-computer interaction.

3. There is a natural synergy between the distributed simulation and networking research projects. On the one hand, the distributed simulation project provides a suite of tools for research in telecommunications through the virtual telecommunication network application. On the other hand, it provides an application for evaluating the effectiveness of communication mechanisms such as multicast and resource allocation algorithms for quality of service guarantees. Moreover, the entire distributed laboratory facility provides a challenging, concrete test case to drive the networking research.

4. The ubiquitous network vision provides new challenges for systems and tool building projects in exploring the feasibility of realizing distributed laboratory environments.

References

[1] T. Bihari and K. Schwan. Dynamic adaptation of real-time software. *ACM Transactions on Computer Systems*, 9(2):143–174, May 1991.

[2] Christian Clemencon, Bodhisattwa Mukherjee, and Karsten Schwan. Distributed shared abstractions (dsa) on large-scale multiprocessors. In *Proc. of the Fourth USENIX Symposium on Experiences with Distributed and Multiprocessor Systems*, pages 227–246. USENIX, September 1993.

[3] Jeff Kramer and Jeff MaGee. Dynamic configuration for distributed systems. *IEEE Transactions on Software Engineering*, SE-11(4):424–436, April 1985.

[4] Bert Lindgren, Bobby Krupczak, Mostafa Ammar, and Karsten Schwan. An architecture and toolkit for parallel and configurable protocols. In *Proceedings of the International Conference on Network Protocols (ICNP-93)*, September 1993.

[5] Keith Marzullo and Mark Wood. Making real-time systems reactive. *ACM Operating Systems Review*, 25(1), Jan. 1991.

[6] Barton P. Miller, Morgan Clark, Jeff Hollingsworth, Steven Kierstead, Sek-See Lim, and Timothy Torzewski. IPS-2: The second generation of a parallel program measurement system. *IEEE Transactions on Parallel and Distributed Systems*, 1(2):206–217, April 1990.

[7] David M. Ogle, Prabha Gopinath, and Karsten Schwan. Tool Integraton in Distributed Programming and Execution Environments – Representing and Using Monitored Information. In *Proceedings of the IEEE Workshop on Experimental Distributed Systems, Huntsville, AL*, pages 83–90. IEEE, 1990.

[8] Daniel A. Reed, Ruth A. Aydt, Roger J. Noe, Keith A. Shields, and Bradley W. Schwartz. *An Overview of the Pablo Performance Analysis Environment.* Department of Computer Science, University of Illinois, Urbana, Illinois, November 1992.

[9] Karsten Schwan and Win Bo. Topologies – distributed objects on multicomputers. *ACM Transactions on Computer Systems*, 8(2):111–157, May 1990.

[10] Karsten Schwan, Rajiv Ramnath, Sridhar Vasudevan, and Dave Ogle. A language and system for parallel programming. *IEEE Transactions on Software Engineering*, 14(4):455–471, April 1988.

[11] Allan Tuchman, David Jablonowski, and George Cybenko. A System for Remote Data Visualization. CSRD Report No. 1067. June 1991. University of Illinois Urbana-Champagne.

[12] I. F. Akyildiz, L. Chen, S. Das, R. M. Fujimoto, and R. F. Serfozo. The effect of memory capacity on time warp performance. *Journal of Parallel and Distributed Computing*, 18(4):411–422, August 1993.

[13] C. D. Carothers, R. M. Fujimoto, Y-B. Lin, and P. England. Distributed simulation of large-scale pcs networks. In *Proceedings of the 1994 MASCOTS Conference*, January 1994.

[14] S. Das, R. Fujimoto, K. Panesar, D. Allison, and M. Hybinette. GTW: A Time Warp system for shared memory multiprocessors. In *1994 Winter Simulation Conference Proceedings*, December 1994.

[15] S. R. Das and R. M. Fujimoto. A performance study of the cancelback protocol for time warp. In 7^{th} *Workshop on Parallel and Distributed Simulation*, volume 23, pages 135–142. SCS Simulation Series, May 1993.

[16] S. R. Das and R. M. Fujimoto. An adaptive memory management protocol for Time Warp parallel simulation. In *Proceedings of the 1994 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 201–210, May 1994.

[17] C. Carothers R. M. Fujimoto and P. England. Effect of communication overheads on time warp performance. In 8^{th} *Workshop on Parallel and Distributed Simulation*, July 1994.

[18] R. M. Fujimoto. The virtual time machine. In *International Symposium on Parallel Algorithms and Architectures*, pages 199–208, June 1989.

[19] K. Ghosh and R. M. Fujimoto. Parallel discrete event simulation using space-time memory. In *Proceedings of the 1991 International Conference on Parallel Processing*, volume 3, pages 201–208, August 1991.

[20] K. Ghosh, R. M. Fujimoto, and K. Schwan. Time warp simulation in time constrained systems. *Proceedings of the 7th Workshop on Parallel and Distributed Sim ulation (PADS)*, May 1993.

[21] K. Ghosh, K. Panesar, R. M. Fujimoto, and K. Schwan. Ports: A parallel, optimistic, real-time simulator. In 8^{th} *Workshop on Parallel and Distributed Simulation*, July 1994.

[22] D. R. Jefferson. Virtual time. *ACM Transactions on Programming Languages and Systems*, 7(3):404–425, July 1985.

[23] Kuek, S.S., and Wong, W.C. Ordered Dynamic Channel Assignment Scheme with Reassignment in Highway Microcells. *IEEE Trans. Veh. Technol.*, 41(3):271–277, 1992.

[24] Lin, Y.-B., and Mak, V.K. On Simulating a Large-Scale Personal Communications Services Network. To appear in ACM TOMACS, 1993.

[25] I. Nikolaidis, R. M. Fujimoto, and A. Cooper. Time parallel simulation of cascaded statistical multiplexers. In *Proceedings of the 1994 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 231–240, May 1994.

[26] P. L. Reiher and D. Jefferson. Dynamic load management in the Time Warp Operating System. *Transactions of the Society for Computer Simulation*, 7(2):91–120, June 1990.

[27] Zhang, M., and Yum, T.-S. Comparisons of Channel-Assignment Strategies in Cellular Mobile Telephone Systems. *IEEE Trans. Veh. Technol.*, 38(4):211–215, 1989.

[28] Ousterhout, J. , Tcl And The Tk Toolkit. Addison Wesley, Reading Massachusetts, 1994.

[29] Shneiderman, Ben, Direct Manipulation: A Step Beyond Programming Languages, *IEEE Computer*, v. 16, n. 8, August 1983, pp. 57-69.

[30] Hutchins, Edwin L., Hollan, James D., Norman, Don A., Direct Manipulation Interfaces in User Centered System Design: New Perspectives on Human-Computer Interaction, Norman, Don A., and Draper, Stephen W. (Editors) Lawrence Erlbaum Associates, Hillsdale, NJ, 1986, pp. 87-124.

[31] Linton, M., Vlissides, J., and Calder, R., Composing User Interfaces with InterViews *IEEE Computer*, v. 22, n. 2, February 1989, pp. 8-22.

[32] Henry, Tyson R., Hudson, Scott E., and Newell, Gary L. Integrating Snapping and Gesture in a User Interface Toolkit *Proceedings of the ACM Symposium on User Interface Software and Technology*, October 1990, pp. 112-122.

[33] Hudson, Scott E., and Stasko, John T., Animation Support in a User Interface Toolkit: Flexible, Robust, and Reusable Support, *Proceedings of the ACM Symposium on User Interface Software and Technology,* November 1993, pp. 57-67.

[34] Mustaque Ahamad, Rida Bazzi, Ranjit John, Prince Kohli and Gil Neiger. In Proc. of ACM Symposium on Parallel Algorithms and Architectures. June 1993.

[35] Mustaque Ahamad, Gil Neiger, Prince Kohli, Jim Burns and Phillip Hutto. Causal Memory: Definitions, Programming and Implementations. Submitted for publication.

[36] Mustaque Ahamad, Ranjit John, Prince Kohli and Gil Neiger. Causal Memory Meets the Consistency and Performance Needs of Distributed Applications. To appear in Proc. of ACM SIGOPS Workshop, September 1994.

[37] Mustaque Ahamad, Phillip W. Hutto, and Ranjit John. Implementing and programming causal distributed shared memory. In *Proceedings of the 11th International Conference on Distributed Computing Systems*, May 1991.

[38] Ranjit John, Mustaque Ahamad, Kishore Ramachandran, R. Ananthanarayan and Ajay Mohindra. An evaluation of state sharing techniques in distributed operating systems. Submitted for publication.

[39] J. K. Bennett, J. B. Carter, and W. Zwaenepoel. Adaptive software cache management for distributed shared memory architectures. In *Proceedings of the 17th Annual Symposium on Computer Architecture*, pages 125–135, May 1990.

[40] M. Dubois, C. Scheurich, and F. A. Briggs. Memory access buffering in multiprocessors. In *Proceedings of the 13th Annual International Symposium on Computer Architecture*, June 1986.

[41] Pete Keleher, Sandhya Dwarkadas, Alan Cox, and Willy Zwaenepoel. TreadMarks: Distributed shared memory on standard workstations and operating systems. In *Proceedings of the 1994 Winter Usenix Conference*, January 1994.

[42] Kenneth Birman, Andre Schiper, and Pat Stephenson. Lightweight causal and atomic group multicast. *ACM Transactions on Computer Systems*, 9(3):272–314, August 1991.

[43] M. Ahamad, M. H. Ammar, and S. Y. Cheung. Multi-dimensional voting. *ACM Transactions on Computer Systems*, 9(4):399–341, November 1991.

[44] Kourosh Gharchorloo, Daniel Lenoski, James Laudon, Phillip Gibbons, Anoop Gupta, and John Hennessy. Memory consistency and event ordering in scalable shared memory multiprocessors. In *Proceedings of the 17th Annual International Symposium on Computer Architecture*, May 1990.

[45] Kai Li and Paul Hudak. Memory coherence in shared virtual memory systems. *ACM TOCS*, 7(4):321–359, November 1989.

[46] Brian N. Bershad and Matthew J. Zekauskas. Midway: Shared memory parallel programming with entry consistency for distributed memory multiprocessors. Technical Report CMU-CS-91-170, Carnegie Mellon University, September 1991.

[47] Sarita V. Adve and Mark D. Hill. Weak ordering - a new definition. In *Proceedings of the 17th Annual International Symposium on Computer Architecture*, pages 2–14, May 1990.

[48] Prince Kohli, Mustaque Ahamad, and Gil Neiger. Characterization of scalable memories. International Conference on Parallel Processing, 1993.

[49] Ranjit John and Mustaque Ahamad. Implementing and Evaluating Causal Memory for Data-race-free Programs. Submitted for publication.

[50] S. Y. Cheung, M. H. Ammar, and M. Ahamad. The grid protocol: A high performance scheme for maintaining replicated data. *IEEE Transactions on Knowledge and Data Engineering*, 4(6):582–592, December 1992.

[51] Vibby Gottemukkala, E. Omiecinski, and U. Ramachandran. A scalable sharing architecture for a parallel database system. In *Proceedings of the 6th IEEE Symposium on Parallel and Distributed Processing*, October 1994. To appear.

[52] S. Gukal, E. Omiecinski, and U. Ramachandran. Avoiding contention between reads and writes using dynamic versioning. Technical Report GIT-CC-94-13, College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, February 1994.

[53] U. Ramachandran, M. Y. A. Khalidi, "An Implementation of Distributed Shared Memory," *SOFTWARE — Practice & Experience*, 21(5):443–464, May 1991.

[54] Partha Dasgupta, Richard LeBlanc, Mustaque Ahamad, U. Ramachandran, "The Clouds Distributed Operating System," *IEEE Computer*, Vol. 24(11):34-44, November 1991.

[55] U. Ramachandran, M. Ahamad, M. Y. A. Khalidi, "Coherence of distributed shared memory: Unifying synchronization and data transfer," In the *Proceedings of the 18th International Conference on Parallel Processing*, August 1989, Vol II:pp.160-169.

[56] R. Ananthanarayanan, Sathis Menon, Ajay Mohindra, U. Ramachandran, "Experiences in Integrating Distributed Shared Memory with Virtual Memory Management," *ACM Operating Systems Review*, Vol. 26(3):4-26, July 1992.

[57] Joonwon Lee and Umakishore Ramachandran. Synchronization with Multiprocessor Caches. In *Proceedings of the 17th Annual International Symposium on Computer Architecture*, pages 27–37, 1990.

[58] Joonwon Lee and Umakishore Ramachandran. Architectural Primitives for a Scalable Shared Memory Multiprocessor. In *Proceedings of the Third Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 103–114, Hilton Head, South Carolina, July 1991.

[59] Gautam Shah and Umakishore Ramachandran. Towards exploiting the architectural features of beehive. Technical Report GIT-CC-91/51, College of Computing, Georgia Institute of Technology, November 1991. Appeared in *1993 ISCA workshop on Scalable Shared Memory Multiprocessors*, San Diego, 1993.

[60] U. Ramachandran. *Hardware Support for Interprocess Communication*. PhD thesis, Computer Sciences Department, University of Wisconsin — Madison, September 1986. (Technical Report #667).

[61] U. Ramachandran, M. Solomon, and M. Vernon. Hardware support for interprocess communication. In *Proceedings of the 14th Annual International Symposium on Computer Architecture*, pages 178–188, June 1987.

[62] U. Ramachandran, M. Solomon, and M. Vernon. Hardware support for interprocess communication. *IEEE Transactions on Parallel and Distributed Systems*, 1(3):318–329, 1990.

[63] A. Sivasubramaniam, A. Singla, U. Ramachandran, and H. Venkateswaran. An Approach to Scalability Study of Shared Memory Parallel Systems. In *Proceedings of the ACM SIGMETRICS 1994 Conference on Measurement and Modeling of Computer Systems*, pages 171–180, May 1994.

[64] A. Sivasubramaniam, A. Singla, U. Ramachandran, and H. Venkateswaran. A Simulation-based Scalability Study of Parallel Systems. *Journal of Parallel and Distributed Computing*, 22: 411-426, 1994.

[65] W. B. Ligon III and U. Ramachandran. Evaluating multigauge architectures for computer vision. *Journal of Parallel and Distributed Computing*, 21:323–333, June 1994.

[66] U. Ramachandran, G. Shah, S. Ravikumar, and J. Muthukumarasamy. Scalability study of the KSR-1. In *Proceedings of the 1993 International Conference on Parallel Processing*, pages I–237–240, August 1993.

[67] Chien-Ting Wu and Graham Campbell, "Extended DQRAP [XDQRAP] A Cable TV Protocol Functioning as a Distributed Switch." *Proc. of the 1994 1st International Workshop on Community Networking*, July 13 -14, 1994, pp. 191-198.

[68] Wenxin Xu and Graham Campbell, "A Distributed Quening Random Access Protocol for a Broadcast Channel," ACM SIGCOMM'93, pp 270-278.

[69] S. E. Deering, "Host extensions for IP multicasting," *RFC 1112*, August 1989.

[70] M. H. Ammar and L.-R. Wu, "Improving the performance of point-to-multipoint arq protocols through destination set splitting," in *Proceedings of INFOCOM '92*, pp. 262–271, IEEE, May 1992.

[71] Cheung, S. Y., Ammar, M. H., "Using Destination Set Grouping to Improve the Performance of Window-controlled Multipoint Connections," GIT, College of Computing, Tech Report GIT-CC-94-32, August 1994.

[72] Almeroth, Kevin and Ammar, Mostafa. "A Scalable, Interactive Video-On-Demand Service Using Multicast Communication", In Proceedings of International Conference on Computer Communication and Networks , San Francisco, California, September, 1994, pp292-301.

[73] Almeroth, Kevin and Ammar, Mostafa. "On the Performance of a Multicast Delivery Video-on-Demand Service with Discontinuous VCR Actions", Georgia Tech, College of Computing, technical Report GIT-CC-94-49, October 1994.

[74] C. Partridge, T. Mendez, and W. Milliken, "Host anycasting service," *RFC 1546*, November 1993.

[75] M. H. Ammar, "Probabilistic multicast: Generalizing the multicast paradigm to improve scalability," in Proceedings of IEEE INFOCOM 94, June 1994, pp848-855.

[76] Talpade, R., Ammar, M. H., "Single Connection Emulation (SCE): An Architecture for Providing a Reliable Multicast Transport Service," GIT, College of Computing, Tech Report GIT-CC-94-47, October 1994.

[77] Adas, A. and A. Mukherjee, "On resource management and QoS guarantees for long range dependent traffic," submitted for publication, 1994.

[78] Beran, J., R. Sherman, M.S. Taqqu and W. Willinger, "Variable-bit-rate video traffic and long-range dependence," accepted for publication, subject to revision *IEEE Trans. Communications,* 1993.

[79] *Communications Week*, May 16, 1994.

[80] Floyd, S. and V. Paxon, "Analysis of TCP traffic: failure of the Poisson model,", *Proc. ACM Sigcomm,* London, 1994.

[81] Garrett, M., and W. Willinger, "Analysis, modeling and generation of self-similar VBR video traffic," *Proc. ACM Sigcomm,* London, 1994.

[82] Klivansky S., A. Mukherjee and C. Song, "Factors contributing to self-similarity over the NSFNET," submitted for publication, 1994.

[83] Leland, W.E., M.S. Taqqu, W. Willinger and D.V. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE Trans. Networking,* 2, (1), pp 1-15, February, 1994.

[84] B. Mukherjee, S. Ramamurthy, D. Banerjee and A. Mukherjee, "Some principles for designing a wide-area optical network," *Proc of IEEE Infocom,* Toronto, 1994. Submitted to *IEEE Trans. Networking,* July 1994.

[85] Pancha P., and M. El Zarki, "Variable bit rate video transmission" *IEEE Commun. Mag.*, Vol.32, No.5, pp. 54-66, May 1994.