I. Cover Page

Proposal to ITO The Advanced Research Projects Agency BAA 96-06: High Performance Distributed Services Technology (HPDST)
Technical Topic Area: Network Engineering

Virtual InterNetwork Testbed (VINT): methods and system

By INFORMATION SCIENCES INSTITUTE
UNIVERSITY OF SOUTHERN CALIFORNIA
LOS ANGELES, CALIFORNIA 90089-1147
PROPOSAL
ISI Proposal 96-ISI-05
ISI Technical Contact: Deborah Estrin
(310) 822-1511, estrin@isi.edu
FAX: 310/823-6714
ISI Administrative Contact: Beverly Hartmeyer
(310)822-1511, BEVERLYH@isi.edu
FAX: 310/823-6714
Type of Business: Other Educational
Principal Investigators:
Herbert Schorr, Executive Director, USC/ISI
and
Deborah Estrin, Associate Professor, USC

Lloyd Armstrong, Jr.
Provost and Senior Vice President for Academic Affairs
University of Southern California

The proposed work will be done jointly with personnel at Xerox PARC and Lawrence Berkeley Labs. They are submitting identical technical proposals. Technical contacts on cross-referenced proposals:
Scott Shenker (Xerox PARC, 3333 Coyote Hill Road, Palo Alto CA, shenker@parc.xerox.com) and
Sally Floyd (Lawrence Berkeley National Laboratory, Berkeley CA, 94720, floyd@ee.lbl.gov)

II. Summary of Proposal

A: Innovative Claims

The rapid diffusion of internetworking technology brings two major sources of stress to the underlying protocol mechanisms and associated design methods: scale and heterogeneity. Scale must be considered in evaluating both the correctness and performance of wide area internet protocols at every level (from routing, to transport and session protocols). Heterogeneity of applications translates into a larger number of interacting protocols, service requirements, and traffic patterns. In light of these phenomena we claim that current protocol design and engineering practices are critically inadequate and must be changed in order to produce robust and evolvable network technology in the future. The proposed Virtual InterNetwork Testbed system hopes to transform network protocol design and engineering practices in the same way that simulation and VHDL-based methods transformed chip and board level design. The following innovations will allow us to meet this goal.

- **Composable simulation framework:** The simulator architecture will provide a framework for composing simulation modules. This will create synergy between disjoint simulation efforts and enable the simulation of complex interdependencies between protocols. This framework is intended to model the modularity of the Internet architecture itself, which allows rapid and independent development of new protocols and applications.

- **Abstraction techniques and tools:** It is hard to identify, and focus on, the relevant phenomena buried within the mountain of data generated by a realistic simulation of an operational network. A crucial aspect of making simulation more valuable is to provide the ability to vary the level of abstraction, in both the analysis of the data and in the simulation itself. These tools will allow users to identify relevant phenomena using high-level simulations and then use detailed simulations to study the phenomena more extensively. Key to this technique will be providing useful correlation of detailed and high-level simulations.

- **Visualization techniques:** Many relevant network phenomena remain invisible when only aggregate statistics are collected. Visualization techniques are crucial in enabling users to identify interesting aspects of the simulation. These visualization tools will assist users in framing design questions and guiding the debugging process with detailed and coarser grain evaluations.

- **Emulation interface:** The emulation interface will allow actual network nodes to interface with the simulator. This will enable more complete debugging of code before deployment, and will also allow simulations to more accurately represent the behavior of the actual deployed code.

- **Libraries of Network Topologies and Traffic Generators:** The load mix of the Internet is shifting rapidly (now to WWW, and perhaps soon to audio and video), and the topology has always been in flux; it is important to provide simulation researchers with the ability to explore the implications of these topology and load shifts. By providing an extensive, and extensible, library of network topologies and traffic generators, it will be possible to test protocol performance over a wide range of conditions. The common library of simulation environments will also allow more comparison between different simulations, easing the verification of the simulation results.

This work will be performed in collaboration with Xerox PARC and LBNL.

B: Deliverables

- VINT System Components
  - Detailed simulator with components representing the basic modules of the integrated services internet (e.g., routing, reservation, scheduling, transport, and session protocols)
  - Session level simulator for thousands of nodes.
  - Visualization front end for detailed and session level simulators.
  - Emulation interface to interface actual network node (e.g., router or host) to simulations.
  - Libraries of topologies and traffic generation modules to be used as inputs to simulations.
- Abstraction and visualization methods.
- Simulations of key integrated services components and their interactions.

See Schedule and Milestones and the Technical Rationale for a more complete description of the above items.

This work will be performed in collaboration with Xerox PARC and LBNL.

C: Cost, Schedule, and Milestones

Detailed Simulator

1. Months 1-6: Evaluate LBNL's *ns* (network simulator) and *nam* (network animator) as basis for detailed protocol simulator platform. Enhance and release *ns*; that is, move *ns* from a beta release to a full release. Begin restructuring for extensibility purposes.
2. Months 3-12: Develop protocol modules for unicast routing, multicast routing, reservation setup, packet scheduling, unicast and multicast transport.
3. Month 6-12: Specify and implement network protocol description interface to formalize the composable framework.
4. Months 7-18: Develop Emulation interface to simulator.
5. Months 7-18: Implement distributed version of simulator.
6. Months 13-24: Work with external researchers to create large, multi-protocol, detailed, simulations of integrated services internet.
7. Months 15-36: Create and distribute version 2 of detailed simulator based on experience of first two years.

Session level simulator

1. Months 1-6: Construct session level simulator framework designed to maximize commonalities with detailed simulator.
2. Months 3-12: Develop protocol modules for routing, reservation setup, transport and session protocols.
3. Months 6-12: Specify and implement network protocol description interface.
4. Months 13-24: Work with external researchers to create large, multi-protocol, session-level, simulations of integrated services internet.
5. Months 13-24: Develop abstraction methods for guiding use of session level simulation based on detailed simulation results, and vice versa.
6. Months 15-36: Create and distribute version 2 of the session level simulator.

Visualization front-end and Libraries

1. Months 1-3: Enhance existing network animator and Routing Arbiter visualization tools, and release *nam*; establish task list for phase 1 visualization front end.
2. Months 3-36: Develop suite of visualization tools and interfaces to simulations, including PC-based client software.
3. Months 6-12: Create initial network topologies and traffic libraries.
4. Months 30-36: Update and refine libraries, including tools generating, capturing and verifying network traffic.

D: Technical Rationale

The emerging integrated services internet infrastructure is different from other complex computing systems in its scale, collection of interacting elements, and rapidly evolving nature of application-drivers and underlying technologies. The continual change in application types, traffic loads, network topology and underlying media, presents a challenge to the design and implementation of network protocols. Current network engineering methods and tools are severely inadequate, resulting in various "failures" during all three phases of network protocol development.

Protocol design usually begins with individual or small group simulations of isolated protocol elements using small-scale topologies and simplified/static assumptions about higher and lower level protocols. Subsequently, many designers create small-scale testbed prototypes with assumptions very similar to those used in their simulations. Both simulation and prototype experimentation typically have one or both of the following goals: (1) proof of concept in terms of protocol function and dynamics, and (2) protocol performance characterization. Unfortunately, it is generally difficult to compare alternative designs because of the lack of standardization and reproducibility of simulations of different design groups. In addition, because the individual simulations are not implemented in a compatible manner, it is virtually impossible to study the interaction between different coexisting protocols without actual implementation or construction of a special simulator.

Implementation starts with single system implementation followed by small-scale prototypes deployed on testbeds with moderate loads and a limited set of application drivers. Typically, the first rigorous debugging that explores scale and protocol interaction occurs only after deployment in an operational network.

Troubleshooting is always needed in the context of network growth, new applications, and new protocols. System complexities and "open/adaptive design" is such that the protocols are constantly being used in contexts in which they have not been tested before deployment. At the same time the proliferation of network usage is leading to an increasingly large and diverse population of network application designers and implementors with varying levels of understanding of the underlying technology. Often it is not possible to recreate the topologies in which problems arise on existing small development testbeds. This is a problem for fault isolation, and even more so for evaluation of alternative fixes to the problem. It can take several iterations of trial and error in the operational network to evaluate alternative solutions. We therefore require new engineering methodologies and tools for experimentation.

The result of these inadequacies is an increasingly fragile system whose behavior is difficult to predict and manage, and whose functionality is difficult to enhance. Because of this we believe that *the* issue in network/protocol engineering is development of methods to evaluate designs and implementations. This is critical to the continued operation and evolution of internet technology for both critical and generic applications. Current methods will not be able to track the explosion in both scale and service diversity.

As shown in Figure 1, protocol design, evaluation and implementation are complicated by 3 factors: scale (number of nodes/domains in the network), complexity (number of interacting protocols across all levels of a stack), and detailed system issues (capture of hardware and operating system issues). Simulation is commonly used in current protocol design efforts by individual Principal Investigators. However, most simulation efforts are narrow, primitive and isolated. They are narrow in that they tend to focus only on single protocols and simulate only those protocols in isolation. They span different scales, but usually consider only small scale. Furthermore, they abstract away detailed system issues and they do not address interactions with other components of the architecture.

In addition to their limited focus, network protocol simulations typically are primitive, if not completely lacking, in their use of visualization. Visualization techniques can facilitate the identification and understanding of complex network phenomena. The narrow scope and primitive nature of many simulation efforts arise because almost all are isolated. Most research and development groups have their own

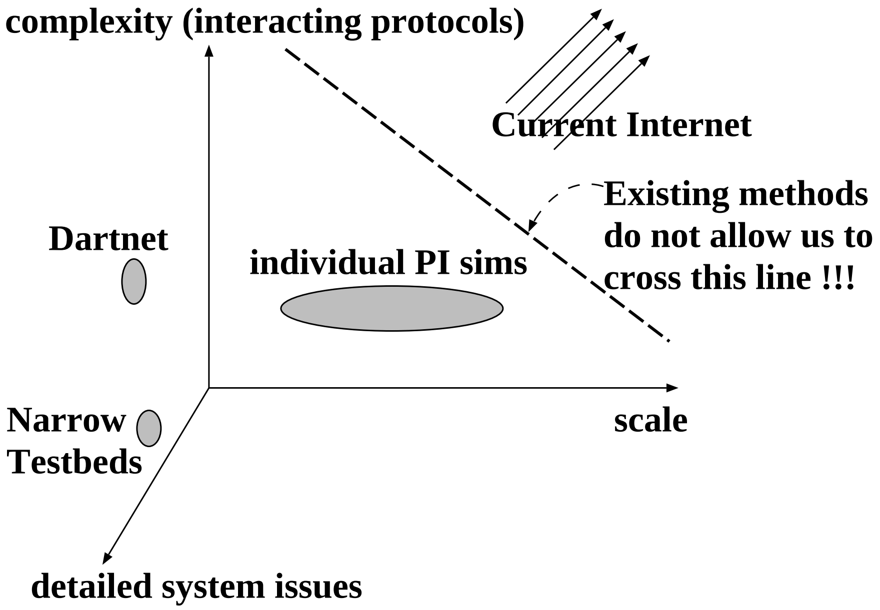

Figure 1: *Why we need VINT. Protocol design and evaluation are complicated by scale, complexity and detailed systems issues. Existing simulation methods and testbeds–which typically only address one, or at most two, of these dimensions–do not allow designers, implementors, or operators to study issues that arise in large, heterogeneous networks.*

independent simulation efforts and cannot import modules from other efforts. Because the startup costs are so high, no individual group has the resources to create a comprehensive and advanced networking simulation environment. One of the main goals of VINT is to increase the synergy among the protocol design community that uses simulation so that startup costs will be greatly lowered and researchers can turn their attention to advancing the state of the art rather than merely reproducing it.

Testbeds are another common method of evaluating protocols. Testbeds enable the study of detailed systems issues that are typically absent from simulations. However, testbeds have very high startup costs (both in terms of money and effort). Moreover, they are severely limited in scale due to the expense of deploying physical lines and boxes. Thus, simulation remains the best option for systematically evaluating protocols and guiding protocol design.

ARPA has a critical role to play in the initiation and support of this work. Given that the global internet infrastructure itself is not centrally designed or engineered, it is all the more critical that we develop methods and tools to evaluate our technologies before (and even after) deployment. However, unlike the chip industry which faces similar needs and complexities, it is not in the commercial interest of any one commercial outfit to develop the methods and tools that would allow evaluation at the scale and heterogeneity needed. In other words, this project is beyond the scope of any single commercial company, at the same time that it is critical to the long term success of both those companies and their clientele. Similarly, the level of effort required to create such tools is beyond the scope of individual university researchers, but the tools would significantly increase the quality of their research and the synergy between their independent efforts. VINT's goal is to qualitatively improve the robustness and evolvability of future internet technology, a technology on which ARPA and its larger community, as well as the world at large, are increasingly dependent.

Proposed Approach

This proposal suggests the development of innovative methods and tools to address these needs. We will focus on what we consider to be the key challenges: scaling and protocol interaction issues in integrated services internets. While the long-term objective of a scalable, multiprotocol, composable virtual internet testbed is very ambitious, we propose the following areas for the focus of our work: detailed simulator, session-level simulator, visualization front-end, and libraries.

Detailed multi-protocol simulations We will build a multi-protocol simulator that supports more extensive study of network protocols. We will achieve this goal by innovating in the area of composability and visualization.

Our composable component framework will allow component modules from multiple contributors. This mirrors network development and is feasible because of the modularity inherent in the network protocols themselves. For example, we wish to enable the designers of a reliable multicast transport protocol to evaluate their design in the presence of simulation modules that represent dynamic multicast routing and packet scheduling algorithms. Or similarly, we want to allow the designers of packet scheduling and reservation protocols to evaluate their design in the presence of simulation modules that represent various subnet technologies (e.g., wireless, ATM).

In addition to creating the composable framework, as part of the VINT project we will implement key protocol modules, including unicast routing, multicast routing, reservation, transport and session protocols. This will allow us to evaluate and improve the simulator framework, and more importantly, will give other researchers a relatively complete "internet context" in which to plug their components for evaluation.

As part of our more advanced development of the detailed simulator we will develop tools for creation of protocol modules based on existing implementations of the protocol functions. Extending this notion further, we will develop interfaces that allow the simulator to be driven by a) modules of implemented code (referred to by others as direct execution[9]), or b) operational network nodes . These techniques will enable improved validation methods and practices by facilitating the iteration between implementation and simulation. We will also explore the use of distributed simulation in conjunction with direct execution of protocol implementation components. In this context, the simulator will act as a topology emulator, interconnecting and coordinating among multiple instances of protocol code modules running on workstations in a cluster.

One of the most significant impediments to more extensive simulation use is the amount of effort typically involved in setting up and interpreting simulations. Graphical manipulation of inputs and animation of simulation processes and outputs should provide the same benefits to network simulation as they have to so many other areas of computer aided design and visualization.

In summary, our goal for the detailed simulator component of VINT is to enable extensive simulations by innovating in the area of simulator composability and visualization. However, such a system alone is not adequate because the detail necessarily limits the tractability of studying very large networks. Therefore, we include the second component of our system described in the next section.

Session level multi-protocol simulation In order to study protocol interaction and behavior at significantly larger scale, we will create a session level simulator. One goal is to achieve as much commonality as possible between the two simulators in terms of user interface, data structures and protocol modules. As with the detailed simulator, it will also be composable in framework, and we will seed it with initial internet protocol modules.

One of the greatest challenges, with respect to this component of VINT, is to develop methods for moving between detailed and high level abstraction: for example, creation of static scenarios at one level based on results at the other, and dynamic interfacing of detailed and abstract modules. In addition, we will investigate tools for creation of session level protocol modules based on detailed simulation or implementation modules.

Special visualization techniques will be needed for the session level simulator to allow graphical definition and manipulation of the very large networks, and visualization of their processes (intermediate states) and results.

In summary, session level simulations will emphasize scaling issues in addition to multiprotocol composability. We will develop and apply validation techniques by correlating session-level simulation results with both detailed simulations, smaller scale testbed experiments, and operational network measurement.

VINT Application target In summary, the Virtual InterNetwork Testbed will consist of a suite of software, the methods for their use in design and deployment of new protocols, and application of the techniques to those protocols that are critical to realization of integrated services on a global scale.

To achieve this last goal, we will use early prototypes of VINT in an active research effort. This will provide initial experience with VINT and provide feedback to its design. These research efforts will explore interactions between component protocols that constitute any integrated services internet. In particular we will study: the interaction of routing and reservation setup protocols and mobility; IP integrated services support over diverse subnet technologies such as ATM and wireless; security (e.g., access control and authentication) mechanisms in IP, routing, and RSVP; reliable multicast transport and underlying multicast routing; Web caching mechanisms and multicast routing; switch packet scheduling algorithms and end node application performance.

Some of these issues require a detailed study of protocol interaction to evaluate their functional correctness; others require coarser grained evaluation to characterize performance issues. For example, we require a detailed protocol simulator and associated visualization tools to evaluate the functional interactions between the stack of protocols needed to support integrated services. Reservation setup protocol messages travel along the multicast distribution trees established by the multicast routing protocols, which in turn depend on the unicast routing protocol; moreover, the protocols' behavior is affected by the presence of multiaccess subnets and must be studied in topologies with both these and point-to-point links.

A second set of examples involves the performance of web protocols. For instance, it has been proposed that the web-cache placement could make use of either network topology information provided by monitoring of network paths, by the routing protocol, or by a network management function such as the Internet Routing Registry. We could use a session level simulator to evaluate the tradeoffs associated with the granularity at which such topology information is provided. On the other hand, a detailed simulator would be needed to study another frequently discussed performance issue associated with the World Wide Web, namely, the interaction between HTTP and the congestion control mechanisms being used by other types of traffic.

Impact

Of course it is not possible to create design tools that eliminate *all* errors from complex protocol implementations. However, we can qualitatively improve the robustness and evolvability of internet technology by developing systematic and realistic methods and tools for network study. The more we expose protocol designs to interactions with other protocols, and a wide range of topologies, traffic models, and network events, the more robust they will be when they are deployed in operational networks.

E: Related Work

- Protocol design, simulation, and implementation of integrated services and routing protocols:
  As a group, the researchers involved in this proposal have extensive protocol design experience. ISI, LBNL, and Xerox PARC were the primary contributors to the design of the three most critical components of any integrated service infrastructure: reservation setup (RSVP), multicast routing (PIM), and packet scheduling and admission control mechanisms (CBQ, CSZ, etc.)

- Network Simulation:
  As a group, the researchers involved in this proposal have extensive experience with network simulation in the process of network protocol design and performance evaluation (e.g., TCP, RSVP, PIM, CSZ and CBQ scheduling, admission control, RED queue management, inter-domain routing design). Based on our experience with MIT's *netsim*, University of Maryland's *MaRS*, UC Berkeley's REAL, Columbia's NEST, and LBNL's *ns*, we propose to build on the *ns* (network simulator) and *nam* (network animator) code base.

- Traffic and topology models:
  USC and LBNL have done extensive work in the area of traffic modeling (Danzig, Estrin at USC [12] and Floyd and Paxson at LBNL [61, 62]), and have both produced TCP libraries [19, 60] that are in use by a wide range of researchers. In addition, USC has developed network topology generators for previous work simulating routing, multicast, and reservation setup protocols. We hope to collaborate with others working in this area as well such as Zegura at Georgia Tech.

- Visualization Tools:
  As part of the Routing Arbiter project we are developing visualization tools for inter-domain routing. In this context we face similar scaling issues. We propose to combine this experience with related work in the area of protocol and algorithm animation to produce visualization methods and tools that can be integrated into the protocol design and debugging process.

- Parallel network simulation:
  The focus of our work is *not* on methods for parallelizing our simulations. However, because we will certainly be stressing the limits of single-processor computational power, we hope to collaborate with and apply the techniques developed by a number of other research efforts in this area, e.g., Univ. Mass., Amherst NSF-Industry funded center on Parallel Network Simulation techniques, and the work of Fujimoto at Georgia Tech.

F: Personnel

Xerox PARC and LBNL are submitting identical technical proposals and we plan to collaborate as follows: Breslau, Estrin, Floyd, McCanne and Shenker will design the initial system architecture and components. The ISI programmers will implement the core simulation and visualization components according to this design. In addition, Breslau and Fall will construct some of the initial integrated services protocol modules for the internet simulations, and Breslau, Fall, and Floyd will conduct sample studies.

- ISI:
  - Estrin (33%)
  - Project manager (67%)
  - 2 programmers (100% each)
  - 2 Graduate Research Assistants
- Xerox:
  - Breslau (75%)
  - Shenker (10%)
  - summer intern
- LBNL:
  - Fall (38%)
  - Floyd (38%)
  - Other members of LBNL's Network Research Group (Group Leader Van Jacobson, Vern Paxson, and most particularly Steven McCanne) will also be involved as appropriate.

Longer term we hope to engage additional colleagues in the project.

III. Detailed Proposal Information

A: Statement of Work (SOW)

Detailed Simulator

1. Evaluate LBNL's *ns* (network simulator) and *nam* (network animator) as basis for detailed protocol simulator platform. Begin restructuring for extensibility purposes.
2. Develop protocol modules for unicast routing, multicast routing, reservation setup, packet scheduling, unicast and multicast transport.
3. Specify and implement network protocol description interface to formalize the composable framework.
4. Develop emulation and code-driven interfaces to the simulator to allow driving of simulations with operational network nodes and with direct execution of network protocol code, respectively.
5. Work with external researchers to create large, multi-protocol, detailed, simulations of integrated services internet.
6. Create and distribute detailed simulator based on experience of development and use.

Session level simulator

1. Construct session level simulator framework designed to maximize commonalities with detailed simulator.
2. Develop protocol modules for routing, reservation setup, transport and session protocols.
3. Specify and implement network protocol description interface to formalize the composable framework for session level simulator.
4. Work with external researchers to create large, multi-protocol, session-level, simulations of integrated services internet.
5. Develop abstraction methods for guiding use of session level simulation based on detailed simulation results, and vice versa.
6. Create and distribute the session level simulator based on experience of first two years of development and use.

Visualization front-end and Libraries

1. Enhance existing network animator and Routing Arbiter visualization tools.
2. Develop suite of visualization tools and interfaces to simulations, including PC-based client software.
3. Create initial network topologies and traffic libraries.

Simulation studies of integrated services and WWW protocols

1. Conduct simulations of existing and advanced internet capabilities.
2. Demonstrate VINT capabilities and contribute to understanding of the specific internet protocols.

All of the above work will be performed in collaboration with Xerox PARC and LBNL.

B: Description of the results, products, transferable technology, and expected technology transfer path

Results

The results/products of this work will include:

- Software: VINT simulator and visualization software, along with a suite of verification tests.
- Techniques/Methods: Abstraction methods, Integration of session and detailed level simulations, Visualization techniques.
- Protocol development: simulation studies of key integrated services internet protocols.

Transferable technology and transfer path

We will promote the use of VINT through the following activities:

- Research community: demonstrate VINT to Sigcomm and Infocom communities through publications, advertisement of tool availability, and conducting of network-based teleconferences to demonstrate capabilities and obtain early feedback from users and potential users. Investigate initiation of VINT related IRTF research group. Promote VINT simulations as community standard for comparisons in published work.
- Vendors and large users (e.g. private networks and service providers): illustrate utility of techniques through application of VINT to problem areas in the community. Build modules corresponding to active design and standardization discussions (IETF, ATM forum, Nanog).

C: Detailed Rationale

Current protocol design and engineering practices are inadequate for the size and complexity of current and future wide-area networked systems. In particular, the individual simulators used in protocol research and design typically focus on a single protocol or phenomenon. This narrow focus limits the operating region explored by each simulator. Moreover, individual researchers often build their own simulators, even when they are exploring very similar problems. While some independence between simulation efforts can be good for validation, the almost complete lack of a shared simulation infrastructure leads to massive duplication of effort and lack of comparability across simulations. Because most protocol simulators were developed for a relatively narrow purpose and for use by one research group, these simulation systems remain relatively primitive, with little advance in the state of the art in the past decade. For example, there has been only minimal investment in visualization software, abstraction tools, and load and topology libraries.

Partially as a result of the primitive design methods and inadequate tools, but also because of their inherent scale and heterogeneity, wide-area internetworks are fragile and difficult to both manage and enhance. For the Internet to become a scalable, dependable, and evolvable component of our telecommunications infrastructure, the community must have the tools to study system behavior in a context closer to the intended operating environment: e.g., with multiple interacting protocols and applications, and at large scale. While it is impossible to recreate all possible operating scenarios, our system and methods will enable far greater coverage of the design space and operating regions than present approaches provide.

Of course it is not possible to create a simulation environment that can capture simultaneously all the interacting protocols, on a very large scale, and at a detailed packet level. Our rationale is to push the envelope in both the complexity and scaling dimensions, somewhat independently, and then to bridge and integrate the two types of analyses with visualization and abstraction methods. VINT is intended to study the correctness and gross-performance characteristics of wide area communication systems. There are two fundamental reasons why we believe this approach will succeed:

- First, network protocols are inherently modular and composable. Therefore, the proposed composable multi-protocol framework of VINT matches the modularity of the artifact itself–both in design and implementation. We are therefore at a significant advantage over some systems because we do not need to introduce artificial divisions and abstractions just to obtain modularity and composability in simulation design.

- Second, to a large degree, wide area networks are inherently loosely coupled because of the propagation and processing delays encountered by packets. Therefore, many aspects of wide area protocol behavior are dominated by time constants that are long relative to the timing details of individual events, such as packet transmission times. This is not to say that detailed timing interactions are always irrelevant; indeed, they are often significant in the context of small(er) subsets of directly connected/coupled systems and can lead to the special types of network phenomenon discussed later in this section. However, detailed timing issues are often not relevant to the general correctness or performance of wide area network protocols; consequently, large-system studies can often be carried out at a higher level of abstraction than that of individual data packets.

VINT will support both large scale, coarse-grained, multiprotocol simulations and small scale, detailed, multiprotocol simulations. In addition, VINT will provide the visualization and abstraction methods used to integrate the two. Our objective is to make the design process more efficient, systematic, and rigorous, and to thereby increase the performance, maintainability, and evolvability of future networks.

In this section we illustrate our objectives and rationale by describing three types of studies that are not achievable without the development of a system like VINT. While there is overlap between these three types of studies, each grouping has a different emphasis. We give specific examples of studies within each subsection.

Systematic comparisons of mechanistic tradeoffs in protocol design

As in most design activities, much of the time spent in protocol design, redesign, and debugging, concerns evaluation of the various mechanistic alternatives for achieving a particular function. There are two ways in which current practice falls short.

- The first is that individual designers tend to evaluate a proposed mechanism using a simulator developed specifically to study that mechanism. Their evaluations are therefore difficult to compare with those of their colleagues who are studying alternative mechanisms. As a result, in many areas there is a dearth of systematic comparisons of detailed mechanistic alternatives. Consequently, the design discourse takes much longer to converge because discussions are often based more on speculation than on directly comparable data. In the current paradigm, directly comparable data would be available only if each individual designer implemented, within their own simulator, all of the competing mechanisms. Very few research groups have the resources to do this, and it is often most effective to have a simulation component constructed by those who know most about the particular protocol represented by the component. With VINT, the composability of the modules would allow each researcher to import the simulation modules embodying the design alternatives from their colleagues, and so direct comparisons would be facilitated.

- The second failing concerns the inadequate scale and scope of the individual simulations so that the evaluations of protocol mechanisms are not conducted across a wide enough operating region. In many cases there is no design choice that is superior in all settings, and so it is important to understand the relative performance of design alternatives across a broad range of operating conditions. Testing across a wide range of conditions requires extensive topology and load libraries, and the development of these libraries is an important component of the VINT approach.

Examples of protocol design issues that could (or would have) benefit(ted) from the use of VINT to enable systematic discourse and evaluation include:

- Soft State: A number of the proposed enhancements to the internet architecture require addition of new state/information in network nodes. A central design choice is the manner in which this information is established and maintained, in the context of network element failures of various kinds. Soft state protocols maintain state in intermediate nodes using refreshes that are periodically initiated by endpoints. When the endpoints stop initiating refreshes the state automatically times out. Similarly, if the intermediate state disappears, it is reestablished by the endpoint-initiated refreshes. The use of soft-state mechanisms is one of the important differences between RSVP and ST2, and between PIM and CBT, for example[4, 20, 55, 71, 76]. Systematic comparisons of these protocols (e.g., RSVP vs. ST2, or PIM vs. CBT) would provide a better understanding of the impact of this major design decision. For example, what effect do the soft-state periodic refreshes have on protocol overhead? How quickly do alternative, hard-state, protocols react to a wide range of network failures?

  In addition to the basic question of soft vs. hard state, open design issues exist within the soft state paradigm itself. Many initial soft state designs, such as in RSVP [76], used best-effort messages to carry the refreshes. More recently, some designers have suggested using explicitly reliable messages for these periodic refreshes. Using reliable message transport entails an increase in the complexity of the mechanism. However, the reliable nature more completely decouples timer management of soft state from the issues of packet losses of these refreshes. Scalable timers is another approach under investigation to improve the performance of soft state mechanisms[1]. Scalable timers replace the fixed timer settings used by existing soft state protocols with timers that adapt to the volume of control traffic and available bandwidth on the link. Extensive simulations, under a range of operating conditions, will be needed to evaluate the impact of such changes in the approach to soft-state management.

- ATM Congestion Control: There have been a myriad of congestion control proposals for ATM; see, for example [8, 49]. The studies supporting these proposals have typically simulated each individual proposal in isolation, and few detailed and extensive comparative tests have been made. This has made the evaluation of their relative performance rather difficult. These congestion control proposals involve several different design decisions, such as credit vs. rate, end-to-end vs. hop-by-hop, and forward congestion notification vs. backward congestion notification.

  The end-to-end vs. hop-by-hop decision is perhaps the most fundamental. The Internet community has adopted the end-to-end congestion control approach, and there is a vast literature of simulation results describing the performance of various instantiations of this approach (e.g., TCP Reno, TCP Vegas [10], etc.). Few of these simulation results have been directly applied to the current ATM congestion control debates because it has been difficult to transfer the simulation scenarios to the ATM setting. With our composable simulation framework, replication of these experiments in the ATM setting would be much more straightforward.

  The hop-by-hop approach embodies a rather different architectural perspective, and occupies a rather different niche in the design space. The ability to sustain high throughput without losses has been verified in small-scale simulations with rather stable loads [49]. However, there is concern in the research community that in larger and more complex topologies, with more dynamic loads, throughput will suffer in these hop-by-hop approaches. The lack of a library of such topologies, along with sufficiently diverse load models, has seriously hindered the investigation of such phenomena.

- Real-Time Service and Scheduling: There is widespread belief that the Internet infrastructure should be extended to support real-time services. Many different algorithms[25, 38, 46, 59, 73, 75], protocols[65, 71, 76] and services[66, 67, 68] have been suggested to support real-time service. While some of these mechanisms have been simulated, they have usually been compared to at most one alternative. Because of the nature of simulation technology and infrastructure, no common framework exists for readily comparing alternative mechanisms. Thus, debates about the efficacy of various services are being carried on without a fundamental understanding of the impact of various choices on network performance.

  For example, tradeoffs exist between the level of assurance provided by a real-time service and network utilization, with higher levels of assurance achieving lower network utilization. Individual researchers have used simulation to study the magnitude of the tradeoff. However, because the relative advantages of different scheduling and admission control algorithms depend to a very large degree on the traffic models used, results are often not directly comparable. A library of source models, like we propose for VINT, as well as a composable simulator framework, would greatly facilitate more meaningful and thorough comparisons.

[1] As demonstrated in LBNL applications and work in progress at LBNL (Floyd, Jacobson) and USC (Sharma, Estrin) on application to soft state protocols.

Analyzing complex behavior–spotting patterns in the data

A second type of study that will be enabled by VINT is the analysis of complex network behavior. While this can well overlap with protocol issues discussed in the previous and subsequent subsections, the critical innovations needed here are twofold.

- First, visualization and animation tools are needed to allow researchers to see more clearly what is happening in the network. Given the complex range of behaviors, and the large scale of the networks involved, merely providing tables of summary performance numbers will not adequately describe the behavior of the network. Moreover, such static representation will not allow researchers to develop intuition about, and understanding of, the underlying network dynamics; and this understanding is crucial in guiding the design of future protocols. Little work has been done to date on the required network visualization and animation techniques.

- The second critical innovation is to allow graceful changes in the level of granularity in the data analysis (and the underlying simulation). While visualization techniques at a relatively high level of abstraction can aid in the identification of interesting phenomena, more detailed methods must be used to analyze the phenomena in detail. This shift from higher levels of granularity to more detailed analysis should be done in an integrated manner so that interesting leads can be more easily followed and time is not wasted attempting to reproduce the phenomena with new initial conditions. Similarly, if a simulation user determines that a certain detail is not relevant to the question being studied, the user should be able to eliminate that detail to move to a higher level of abstraction without having to switch environments and configurations, i.e., without losing the context of the detailed simulation.

There are several examples where details of the underlying dynamics were, or are, crucial in understanding the more global behavior. Such cases, like those listed below, could have been more easily identified with the visualization and granularity transition innovations proposed above.

- Segregation Phenomena: The goal of TCP congestion control mechanisms is to achieve a relatively fair and efficient allocation of bandwidth. However, under some circumstances, the TCP connections competing for a highly congested link will not split the bandwidth evenly, but will segregate into “winners” and “losers” with the losers receiving essentially no bandwidth at all [21, 28]. The segregation phenomena depends on the details of timer settings, retransmission algorithms, and topology, and therefore its occurrence is difficult to predict without simulation. While this segregation phenomenon derives from very fine grained details, its effect is visible in very coarse macroscopic measures of bandwidth. The ability to shift the granularity of the data analysis from coarse-level throughput summaries to detailed packet dynamics would have been of great benefit in identifying this phenomenon.

- Ack-Compression: When TCP data traffic is flowing in both directions on a link, the acknowlegements and data packets become mixed together. The acknowledgements are “compressed” together in the queue. This ACK-compression destroys the proper spacing between the acknowledgements, and the resulting clump of acknowledgements triggers a burst of data packets from the source [77]. This is another example of a phenomena whose roots are in the detailed packet dynamics, but whose effect is visible in more macroscopic measures. In fact, much of the rapid variation in router queue lengths seen in some simulations can be attributed to “ack-compression”. While these queue variations had long been visible in simulations, the underlying cause was not identified for years because the simulation tools used did not facilitate correlating queue variation with underlying packet dynamics.

- Thrashing: The introduction of resource reservation into the Internet will usher in a new form of resource contention. In addition to the packet-level contention for resources (bandwidth and buffers), there will also be contention at the reservation level. The admission of one flow may cause some later arriving flows to be rejected by admission control. Initial studies of this form of resource contention revealed that under some conditions "thrashing" can arise, in which the overall call acceptance level goes down as the offered load increases[54].

  This effect is accentuated by the presence of multicast flows, and depends on the topology of the network. There is little understanding of this phenomenon, and in particular there is little insight as to whether it would arise under conditions foreseen for the future Internet, or whether it only occurs under very special and pathological conditions. Visualization techniques would be very helpful in determining, in a rather large simulation, whether portions of the network had entered a "thrashing" state. In addition, since this effect is driven by the call-level semantics, the use of a session-level simulator would be appropriate here.

As new protocols and applications are introduced into the Internet, we expect that new dynamic phenomena will arise whose identification will be greatly aided by the presence of VINT. While the Internet was in its research and prototype phase, it was perhaps acceptable to discover such phenomena long after the protocols were developed and deployed. However, as the Internet becomes one of the crucial foundations of the nation's telecommunications infrastructure, we must identify and understand such phenomena as much as possible well before they can put the network at risk.

Multi-Protocol Interaction

Our final but equally critical example is enabling studies of multiprotocol interaction. This category is somewhat orthogonal to the previous two in that we may conduct systematic studies of tradeoffs, as well as studies of dynamics, in the context of multi-protocol interactions. As with previous studies described, it is obviously not possible to explore all possible interactions between protocols. While protocols have well defined interfaces that constrain their direct interaction, often there are unintended interactions between the dynamics of protocols that leads to unexpected results.

The advances in networking to support multimedia communications and other new services are realized through the introduction of new protocols; these protocols interact with each other and with previously existing protocols. The range of possible interactions is far greater than what can be predicted beforehand and what can be studied by looking at narrow single-protocol simulations. The current state of the art in protocol design typically does not include exploration of such interactions at all until deployment in a testbed or operational network; and the testbeds are typically so small that they do not significantly extend the richness of experience with the protocol. Designers require a framework in which correct protocol behavior and basic design tradeoffs can be explored in the context of such interactions. The following examples illustrate the need for studying multiprotocol interaction in more detail.

- TCP Congestion Control: There are a number of changes being proposed to TCP: the addition of Selective Acknowledgements, changes in the window increase and decrease algorithms (e.g., TCP Vegas), "snooping agents" in the network to improve the performance of TCP over lossy wireless links, large window modifications, modification to TCP to improve performance over paths with severely asymmetric bandwidth, and modifications to ATM networks to improve the performance of TCP over ATM. While there is a good deal of potential interaction between these different proposals, these areas are generally each being investigated in isolation. How will "snooping agents", designed to intercept acknowledgement packets and to work with current TCP implementations, interact with Selective Acknowledgements? How would the addition of Selective Acknowledgements affect the performance of TCP Vegas, or the performance of TCP over ATM? Questions such as these can only be answered with a simulator that implements all of the supposedly independent changes to TCP that researchers are investigating in their individual simulators or in their experimental apparatus.

Note that this is not a case of competing protocol modules such as PIM vs. CBT or RSVP vs. ST2. This is a case of exploring the interaction of somewhat-orthogonal changes to the network designed to affect the performance of TCP: the addition of selective acknowledgements to both the sending and the receiving TCP; a change to the TCP sender's window increase algorithm; the addition of a "snooping agent" inside the network; a change to the cell-drop policies or to the ATM-level congestion control mechanisms within an ATM cloud.

- Reliable Multicast Transport and Routing: Consider the interaction between Scalable Reliable Multicast (SRM)[29] and Protocol Independent Multicast (PIM)[20]. SRM takes advantage of identifying error-dependent neighbors (that is, neighbors who are behind the same lossy link) to improve protocol performance. Continual monitoring of the loss-behaviors guides a member to achieve better performance if the network environment migrates because of gradual membership changes. We would like to test whether the learning procedure is still valid and convergence is acceptable if the error-dependency between members changes dramatically due to regular network events or actions of another protocol.

  One such scenario to test is the effects of multicast routing changes; for example, the effects of a multicast router switching from the shared rendezvous tree to the shortest path tree (i.e., in PIM). Two neighbors that were not error-dependent could become error-dependent, and members that were error-dependent may become error-independent. In fact, a large portion of the group members may face a new environment in which they have to adapt by learning from scratch.

  By analyzing protocol behavior in the context of the multicast stack, we can better study, simulate and evaluate the correctness, functionality and performance of the learning mechanism.

- Reservation Setup and Routing: Interaction issues also arise between the reservation setup protocol (such as RSVP) and an underlying multicast routing protocol (such as PIM). There is a strong potential for interaction (both beneficial and detrimental) between RSVP reservation styles and the type of multicast tree used. For example, if the multicast routing protocol uses a source's data rate as the trigger to establish a source-specific tree, what is the best way for RSVP to handle reservations? Should RSVP set up reservations on the shared tree, and then again over the shortest path tree? Should RSVP be able to request switching to the shortest path tree independent of current source data rate to expedite stabilizing the route and reservation? RSVP allows a single reservation to be shared among multiple senders to the same session. What are the implications of the type of multicast tree used on the efficiency of such shared reservations?

  Other open design issues exist regarding the interaction of RSVP and routing. For example, if the shortest path tree reservation is rejected, should the receiver revert back to the shared tree, prefer unreserved packet delivery over the shortest path tree, or possibly join on an alternate path to select an alternate source-specific-tree branch. The design of such advanced multicast services requires far better tools than are currently available to systematically study correctness and performance questions regarding protocol interaction.

- QOS Reservations and Wireless Networks: RSVP and the use of reservations also potentially interact with the existence of wireless overlay networks. One use of wireless networks is to provide last hop connectivity to mobile end systems; and they do so at varying bandwidth. One of the proposals for dealing with such variable bandwidth is the use of proxies that carry out rate-adaptation, for example, to match the current bandwidth available on the last hop link[2]. However, it has also been proposed that hierarchically encoded signals, such as video, might be sent in separate sublayers on different multicast addresses and receivers would adjust their reservations on the different groups according to available bandwidth near the end system [52]. How would these two somewhat redundant mechanisms interact? Should the proxy eventually relinquish this task, or is it a more fundamental one?

Such questions can only be explored through study of these protocols all in operation at the same time, and adjusting to the common underlying network dynamics with respective time constants, etc. Note however, that the interactions have little to do with the actual data packet flow and might be well suited to higher level simulation.

- Security and Resource Reservations: Two of the important impending changes for the Internet are the introduction of a security architecture (IPSEC) and the introduction of resource reservation (the work of the IETF intserv and RSVP working groups). The use of resource reservation will likely be accompanied by some usage restrictions or disincentives, either in the form of access controls, usage quotas, or usage-based fees. Implementation of these usage controls, in whatever form, will require authentication and other support from the security infrastructure. Thus, resource reservation and security protocols will interact quite strongly. Despite these important interactions, most of the design of these two architectural modifications were done without significant consideration of each other. A simulation facility like VINT would enable rapid exploration and debugging of these interactions. In particular, the scaling behavior of the interaction between these protocols is not well understood. For example, popular pay-for-view Internet multicast distributions, like a concert, would have large numbers of receivers attempting to simultaneously authenticate themselves to gain access to the multicast distribution. While the scaling behavior of multicast routing and resource reservation are reasonably well understood in isolation, it is not clear that the interactions with the authentication required for commercial usage will scale sufficiently well.

- WWW–Caching, Transport, Multicast, and Routing: A number of potentially major changes lie on the horizon for HTTP and the Web, and there are significant interactions between these planned changes. T/TCP, if widely deployed, will considerably shorten the number of packets in a typical HTTP exchange, as well as reducing latency. In a separate approach, "persistent HTTP" [56] leaves HTTP connections open in hopes that they will be reused shortly. Persistent HTTP is entirely implemented above the transport layer, but the effectiveness of persistent HTTP depends on the underlying TCP dynamics. For example, will there be problems because TCP connections left idle after a lengthy lull use stale window information?

  Widespread Web caching would reduce latency in fetching Web items, diminishing one of the main reasons for switching to T/TCP. Web caching may prove most effective if implemented using multicast, in which case the caching dynamics may stress the current multicast design and point to new multicast mechanisms or revised multicast routing algorithms. The future effectiveness of web caching will also depend heavily on the future evolution of web traffic patterns. Assessing all of these tradeoffs requires rich multi-protocol simulation with sophisticated traffic generators.

This section raised a wide range of questions regarding protocol design and dynamics that VINT would allow us to pursue in a systematic, scientific, manner. The availability of VINT capabilities has the potential to radically change network protocol design and engineering practices. The next section describes how we envision realizing this potential.

[2] Personal communication, Randy Katz, UC Berkeley.

D: Detailed Approach

We propose to bring about a radical change in protocol engineering practices by enabling the study of protocol interactions and scaling. The availability of our proposed system will reduce the massive duplication of effort currently expended in the networking community. At the same time, it will facilitate the comparison of alternative designs and approaches. VINT will provide a composable simulation framework that operates at two complementary levels of abstraction–detailed and session level. Another major emphasis of our work will be the development of a sophisticated visualization front end to enable sorting through the massive amounts of data that will be generated, and to guide further studies. To complement the development of these tools, we will explore methodologies to assist in the systematic exploration of the very large design spaces, and the validation of our results using experimental testbeds and operational networks.

Detailed simulator

To create a composable component, detailed simulator we must develop a composable framework, an emulation interface, a simulation engine, validation techniques, and visualization methods.

We will develop a protocol interface definition for construction and interconnection of independently developed modules. We will begin with a prototype that will be developed based on general understanding of protocol interaction and then refine the model of both the different components and their interaction by applying our framework to the multiprotocol simulations described earlier.

We will use this same interface definition as the basis for our emulation interface. This development will allow us to incorporate a real network node as a component in a simulation.[3] The emulation interface will exchange protocol control packets between the simulator and a real implementation running in an end system or router. It will have very limited support for exchange of data packets because of the difficulty of overcoming timing mismatches. On the other hand, exchange of protocol messages operates on a relatively coarse time granularity compared to packet forwarding, and therefore the speed mismatches should be less perturbing. The main purpose of the emulation interface will be to allow protocol interaction with a fully implemented node as a means of exposing the implementation to a larger variety of protocol events than is possible to create in small testbeds.

We will initially build upon the object oriented protocol simulator developed at LBNL, *ns*. It has the appropriate object oriented structure, with a complementary visualization interface system called *nam*, described later. *ns* is an event-driven network simulator embedded in the Tool Command Language (Tcl). The simulation engine is implemented in C++ and is configured and invoked via a Tcl interface. Tcl procedures are used to provide flexible and powerful control over the simulations (e.g., start and stop events, network failures, statistics gathering, and network reconfiguration). *ns* extends the Tcl interpreter with commands to create the network topology of links and nodes and the agents (processes) associated with nodes. *ns* has been used mostly for small scale simulations of queuing algorithms, transport protocol congestion control, and some multicast related work. As part of VINT, we would develop extensive libraries of protocol agents and push the scaling capabilities as far as possible.

One scaling constraint in *ns* that must be addressed is the storage requirements of the routing tables. Currently, each node maintains a route to all other nodes in the network, resulting in aggregate memory requirements that grow quadratically with the number of nodes. We can solve this problem by replacing *ns*'s flat addressing conventions with a hierarchical approach and by implementing efficient hierarchical routing table look-ups in the node objects. Without this change, we could not accurately model the

[3] Note we are not proposing to incorporate fully representational modules of a commercial router. Rather we will create an interface by which we can hook a commercial router into a simulation and test its conformance.

scaling behavior of real networks since large-scale operation is critically dependent upon hierarchical routing protocols.

Currently, *ns* represents an underlying transmission link and its corresponding scheduling and queueing algorithms in a single object. The VINT simulation framework, however, will require that these two entities be separate. By making this separation, we will be able to flexibly combine different scheduling algorithms with different underlying link technologies. In the current application architecture, adding modules to support new link types like multiaccess subnets or wireless links requires modification to each module that implements a scheduling discipline.

Another urgently needed *ns* modification is a performance improvement to its event scheduler, which is currently based on a linear search insertion algorithm. Performance is particularly poor when the number of outstanding events grows large, which can easily happen when simulations are run with large queue sizes or many active traffic sources. The insertion-sort queue should be replaced with either a heap or a calendar queue. *ns*'s C++ implementation makes it easy to experiment with multiple scheduling algorithms since coexistent schedulers can be derived from a single base class abstraction.

A key aspect of the detailed simulation effort will be validation of the individual protocol modules. For some protocol modules, validation can be carried out by conducting batteries of experiments on small testbeds and comparing their results to those of the simulation modules. Successful interoperation of the simulator with the emulation interface can provide additional validation of simulator modules.

Other protocol modules can be validated with a suite of validation tests, each one of which verifies one aspect of the protocol's behavior. These validation tests will be distributed with the simulator, as a concrete illustration of the behavior of the protocol modules, and as a tool that experimenters can use to check the behavior of protocol modules after they have made changes to the simulator. This approach to validation is the one currently used by *ns*, which is distributed with a suite of validation tests [24, 26, 27] illustrating the various TCP (Tahoe, Reno, SACK, etc.), RED (Random Early Detection) gateway, and CBQ (Class Based Queueing) modules implemented in the simulator.

Visualization will play a key role in using the detailed simulator; it would be difficult, if not impossible, to deal with the large mass of data without it. We discuss the needed visualization capabilities later in this section.

Session simulator

The session level simulator will have similar protocol components but will operate at the session level. In order to retain as much capability as possible for studying the details of the network protocol dynamics, the session level simulator will continue to exchange control messages in the form of packets. However, the data packets themselves will be represented by sessions (flows) instead of individual packets. By abstracting away the details of individual packets and forwarding, we hope to achieve significant increase in the scale of simulatable systems.

As shown in figure 2, the session level simulator has protocol modules that send and receive protocol messages and in many cases these can be the same as in the detailed simulator. In place of the data packet generator and scheduler, the session simulator will have session generator and scheduler functions. The node and link modules will operate in terms of sessions as well. Consequently the session simulator will reduce the number of events and state required by the simulator, in proportion to the ratio of data to control packets; e.g., for a 10Mbps flow, RSVP has a steady state refresh rate of one message every 30 seconds per link, for an approximate ratio of 10^5 data to control. The emulation interface will drop all data packets and pass protocol control messages to the corresponding module. For best effort traffic, the monitored data packet pattern may be fed into the session description.

Of course, this abstraction comes at the cost of lost detail and therefore departure from "reality". Consequently the most important aspect of this part of our work will be the development of methodologies

Figure 2: Block diagram of protocol simulator. The protocol processing modules will implement the protocol actions. Node modules will handle packet or session queues and available resources. Link modules will model link bandwidth and loss attributes, as well as multi-access characteristics. Traffic input characterizes the arrival patterns of packets or sessions. Topology and call patterns input indicate the configuration of nodes and links and the location of sources and receivers. The emulation interface allows a real operational node to interface with the simulator. The simulation engine controls and coordinates the functions of the other modules, and the visualization/animation interface delivers the simulation "results" to the visualization tools (e.g. network state, events, topology, etc.)

for using high level simulations to guide detailed simulations and detailed simulations to determine the salient properties to be included in higher level abstractions.

In most cases the agents that process and generate protocol control messages will be realized using code modules from the detailed simulation. Similarly, the network conditions defined and captured in one context can be pursued in the other. This commonality of structure will greatly facilitate moving back and forth from fine to coarse grain simulation.

The higher level simulator will use a range of techniques for abstracting away the details of individual packets; dealing with entire sessions rather than individual flows is only one of these techniques. For example, for the numerical examinations of the request/repair algorithms in Scalable Reliable Multicast (SRM), the goal was to investigate the probabilistic behavior of the algorithms in multicast groups of up to a thousand members embedded in networks of up to five thousand nodes. Not only was an individual simulation large-scale, but each simulation was repeated many times to explore the probabilistic behavior. The central issue of the simulations concerned not the behavior of the algorithms in the presence of a particular environment of competing traffic, but the optimal behavior of the algorithms in an unloaded network.

For these simulations, we could not afford to use a packet-level simulator that traced each packet through each router and link along the path. For each packet that was sent by a member of the multicast group, our simulator for the SRM experiments simply calculated the time that that packet would be received by each of the other members of the multicast group, given that the packet followed a shortest-path tree and encountered no congestion along the way. We maintained two completely separate simulators: the packet-level simulator *ns*, and a separate simulator for the numerical investigations of SRM. The advantage of this approach was that the specialized simulator for SRM is able to run the large-scale simulations that we required. The disadvantage was that, because the SRM algorithms are not incorporated into the packet-level simulator, we are unable to explore the SRM algorithms in smaller-scale environments with the less predictable influences of congestion caused by competing traffic in the network. What is needed is a single environment in which the user can move between fine-grained packet-level simulations and more coarse-grained simulations. In this context the experimenter can choose to speed up the simulation studies by ignoring the effects of congestion, for example.

Visualization Front end

Both detailed simulations and larger scale high level simulations will generate enormous amounts of data. Radically new visualization tools are essential to allow for identification of interesting and relevant phenomena, as well as detailed investigation of protocol behavior associated with particular protocol mechanisms, or their interaction.

Visualization will be added to all phases of the simulation process: simulation inputs such as topology definition, traffic/call patterns, network "events"; dynamic state monitoring such as link loads, node states, server and end system statistics, protocol message tracing; simulation statistics collection such as distributions of link, node, and end system information; and abstraction methods for varying the level of detail in data analysis.

Very large topologies will be created using a tool that generates random topologies subject to user specified parameters, i.e., number of levels of hierarchy, connectivity degree at various levels, number of bypass and bilateral links, total number of nodes, etc. Other topologies will be hand configured to represent specific operational networks or networks with very specific characteristics; a graphical user interface will be provided for user definition and modification of the input network topology. In all cases, the user should be able to visualize the resulting topology to identify particular features, and patterns. Map placement is a non-trivial problem and several heuristic approaches have been developed[17, 70]. In addition, because of our interest in large topologies and various levels of abstraction, we will explore the use of abstraction and focus techniques such as fish-eye views.We hope to both employ and contribute to the state of the art in this area.

Visualization tools will also assist in defining traffic inputs. We are interested in the arrival and departure processes and location of sources, the flow characteristics of these sources, the arrival and departure processes and location of receivers, and their reservation characteristics (if any). In some cases the user will be interested in directly related flows (e.g., multicast groups used as part of a single conference, or local recovery groups used as part of a reliable multicast). As with topology, the user should be able to inspect automatically-generated or hand-configured inputs to identify patterns.

Some simulations run with only topology and traffic as input with the purpose of viewing the transient and steady state conditions. Other simulations explore the response to dynamic network events, such as link and node failures, that are not a function of the offered load. These events can be defined and then animated graphically.

The use of colors, line widths, sizes and various charts and graphs will be used to provide a high level view of congestion, traffic flows and route changes, etc. For example color changes could identify traffic loads on a link, flashing links could identify flaps (route changes) and the color of a router could indicate its overall load/congestion. Bar and strip charts can be used for viewing specific numbers for a given link or node.

For detailed testing researchers and developers may be concerned with specific packets and their contents. Graphical tools will be provided for the manipulation of data packets at the bit level. The user will be able to graphically edit protocol headers and even data content for replay to quickly create specific events in a network that would otherwise be length or difficult to recreate. The inputs can be derived from scratch or from packets from a live capture.

Dynamic states are the key output of simulations. One component of visualization assistance will be based on network animation. We will start out by building upon the LBNL tool, *nam*. *nam* is a tcl/tk animation tool for viewing network simulations and packet trace data. A user-created tcl procedure defines the topology and layout and the animation proceeds based on trace data from either a simulation or a traffic monitoring tool. This tool can be used to explore the flow of protocol messages and the modification of node state and packet forwarding processes. End system and server status can also be viewed using this type of animation.

We will extend the current instantiation of network animation to display dynamic, but aggregated information to assist in the detection of patterns. For example, the user will be able to flexibly associate different types of state or events with different colors and will thus be able to view the network at different levels of detail. Moreover, *ns* and the network animator need a unified method for describing the network topology and layout. Currently, the user must supply two separate topology descriptions by hand and further augment the animator topology with layout information (i.e., how the nodes are oriented with respect to each other). Ideally, *ns* would simply communicate the topology to *nam* and invoke an automatic graph layout algorithm. We plan to do exactly this by adding a command to *ns* that will automatically layout the network (with optional hints from the user) and synthesize the topology description syntax required by the animator. Visualizing network statistics will also be supported by new abstraction tools.

The visualization front end will provide a common interface and toolset to both the detailed and session level simulators. This will facilitate the transition of particular studies between the two granularities. In addition, we will apply visualization techniques directly to the issue of facilitating transitions between session and packet level simulations

Libraries

Both the detailed and session level simulators will be driven off of libraries representing topologies and load.

Network topology libraries will provide a wide range of specific and random topology types of various scale and configuration. We will construct some topology libraries that represent real operational networks, as well as automatic topology generation tools with many configuration inputs such as connectivity degree, hierarchical levels, bilateral and bypass links, etc.

Load libraries will model particular application traffic, as well as group membership distributions and call patterns. As with the topology libraries, some of these will be derived from empirical data or analytic models, and others will be randomized to allow "what-if" investigation of particular parts of the operating region, even if that region is not currently observable in operational networks.

Network element libraries will be constructed to represent the different characteristics of common link and node types, such as multiaccess LANs, switched multiaccess subnets, ethernets, point-to-point links, wireless links (with corresponding loss rates), and satellite links (with corresponding delay).

Problematic test cases will also be represented in libraries to capture those topology and load configurations that are often the source of protocol bugs. For example, multihomed multi-access LANs often create problems for protocol designs that were conceived thinking only about point-to-point links. Similar examples exist for unicast routing (i.e., those topologies and network events that tend to generate multi-node loops for example). In the case of reservation protocols the cases might be a particular configuration of source and receiver location and the reservation style requested. Whereas for WWW simulations, it might be a particular access pattern on the part of clients combined with particular data location.

Distributed Simulator

We will explore the possibility of employing distributed simulation.

Network layer protocols are modular and composable. Each node running a particular protocol maintains its own state and only interacts with other nodes running that protocol using messages. A simulation of the protocol on a multi-node network can be partitioned to satisfy the criteria observed in [30]. By this criteria, the simulation state variables for the protocol must be partitioned into a set of disjoint states, such that every event of the protocol modifies at most one state. Therefore, the protocol state at a particular node would be in at most one state of the simulation. Likewise, the state associated with a multi-access link, such a broadcast LAN, would have to be in at most one state of the simulation in order to satisfy the criteria.

Each state of the simulation could be executed independently in a separate process; with each process communicating with the other processes using inter-process communication messages. We will first attempt to run detailed simulations in real time, possibly scaled by some factor to allow processes to proceed in synchrony. In this context, network protocols should be good candidates for distributed simulation techniques, with very little added synchronization mechanisms. However, if the distributed simulation cannot operate in real time, *i.e.*, the simulation time is independent of the real time, it is likely to introduce causality violations due to variations in processor speeds, message latencies, etc. In this situation, different synchronization techniques are needed, such as [14, 44].

Methodology

Some of what we have already described could be classified as methodology, e.g., moving between levels of abstraction and validating protocol components. In addition to these two efforts, we will develop new methods for:

- Using network protocol animation to identify protocol interactions of interest. We might be able to learn enough about the types of topologies and traffic patterns that create problems that we could characterize them in simulation libraries or analytic models.
- Using visualization of network dynamics to identify and investigate dynamic phenomena.
- We will incorporate and extend work by Zegura et al. that characterizes network topologies.
- Given the very large state space, we will look to other similar disciplines for techniques used in identifying those parts of the state space most in need of analysis and that help to generalize from studies of one part of the state space to others; the latter is referred to enumeration proofs Such techniques are applied in the area of VLSI design, for example.

Conclusions of detailed approach

In this section we described the components of our proposed VINT project. The various components were described in relation to the rationale presented in the previous section. This section should also serve to better explain and justify the task items proposed in our statement of work.

E: Previous Results

Simulation and emulation have been used extensively in previous research. The research we propose here extends the current state-of-the-art in simulation techniques for networking protocols. We briefly review some of the related work in the areas of network simulation tools, parallel and distributed simulation techniques, testing methodologies, and integrated services protocols.

Simulation in Protocol Research and Development

Many network simulators have been constructed to support protocol research and development. Two such discrete event simulators are *MaRS* [2] and *netsim* [35]. *MaRS* was designed to study network routing protocols. Thus, it allows new routing protocol modules to be added. However, because of its focus on routing, its support for transport layer protocols and application source models is very limited.

In contrast, *netsim* focuses on transport layer issues. It (and its derivatives) have been used in many studies of TCP performance. However, it contains no support for dynamic routing, relying instead on static routing tables. Both *netsim* and *MaRS* have primitive graphical user interfaces.

ns is a centralized simulator that supports transport layer modules. At the network layer, it also supports multicast forwarding. The front end of *ns* can be configured using a tcl parser. *nam* is a visualization tool that can interpret the output from *ns*. We have proposed extensions to these tools in the earlier section on our "Detailed Approach".

The development of simulation tools has not been limited to research environments. Among a number of commercial simulation packages are SimuNet [69], Prophesy [63] and CPSim [18]. SimuNet is a centralized network simulator, that has been tested for a 50 node network running OSPF. Prophesy is a generic centralized simulation tool for queuing analysis. CPSim is a parallel simulation tool that can be extended for computer networks; the freely available version is limited to 256 simulated objects.

Network emulation is another technique for protocol research and development. The Bell Labs Network Emulator [50] emulates a virtual network comprised of point-to-point links. The emulator itself runs on a transputer hardware platform. It has been used to run an emulated version of DARTnet, as well as an x-kernel.

x-Sim [9] runs over the x-kernel; it supports the "direct execution" of transport layer protocol implementations within the x-kernel. The simulator has not been tested for routing and reservation protocols. Furthermore, process emulation in the form of direct execution is restricted to the execution of a single instance of the protocol running on the operating system. We are proposing the execution of protocols on a simulated network environment, rather than in a single OS context.

Nachos [15] and SimOS [64] are other process emulation environments that operate at the granularity of individual instructions. Nachos is a centralized process emulation environment that simulates the MIPS R2/3000 integer instruction set. SimOS is a distributed process emulation environment that supports the emulation of different hardware architectures. It provides the capability to use different hardware components that operate at different granularities and speeds, and permits the emulated process to switch components on-the-fly.

Process emulation at the detail of individual instructions is useful for the study of operating system concepts. On the other hand, network layer protocol design is concerned with the external interactions of any process. Hence, detail at the instruction level is not totally relevant to the design of protocols. It is therefore sufficient for us to capture and emulate only the external interactions of the process when running a detailed simulation.

Visualization of the results of a simulation is a critical component of a simulation environment. A number of different visualization packages are available. Many visualization packages are targeted towards scientific simulations; for instance [48]. *nam* is a visualization tool for networking protocols.

Parallel and Distributed Simulation Techniques

Parallel and distributed simulation techniques have been explored extensively to meet the scaling requirements of large simulations. A survey of the current technology can be found in [30]. Alternatives in the design of a distributed simulation can be found in [58].

Parallel (and distributed) simulations may require synchronization techniques to guarantee consistent results. Conservative approaches for such synchronization require all processes to make progress in unison [13, 11]. Optimistic approaches, such as time warp [44], roll-back a process to an earlier consistent state if causality violations are detected. A number of efficient strategies for synchronization of the processes in a distributed simulation have been proposed [45, 14].

The best possible criteria for partitioning the logical processes in a simulation is identified in [30]. The criteria is that the simulation state variables should be partitioned into a set of disjoint states, such that an event in the simulation modifies at most one state. Iterative improvement methods to automatically partition problems in sequential circuits are discussed in [51, 57]. Computer networks can be trivially partitioned along the links that form the WAN topology to satisfy the criteria above. But for simplicity, our approach will be to let the protocol designer specify the partitions when running a distributed simulation.

There are a number of different techniques for the simulation of communication networks. Most of these are conservative simulation techniques [11, 72]. An optimistic call level simulation of circuit switched networks is discussed in [31]. The largest size network simulation reported is 100 nodes, and a million calls.

Different simulation systems for protocol design have been proposed in the literature. NEST [22] is a simulation and prototyping testbed for distributed network applications. The system consists of a simulation kernel that also acts as a scheduling kernel for the simulation. They have addressed a number of issues, such as synchronization, that we will use in our distributed simulator. Some of the drawbacks in their proposal are that they only model nodes and links. Protocol behaviour is also dependent on the media that the protocol is running over, such as multi-access LANs, ATM networks etc. They provide a list of NEST library functions that the prototype must use. They also coalesce the processes running at each node into a single process, and use a light weight process model to schedule execution of the protocols at different nodes. Hence their simulation server also acts as a scheduling kernel.

Protocol Testing

A review of the various protocol testing literature can be found in [7]. Testing in general, and protocol testing in particular, are being studied extensively.

Estelle [41], LOTOS [42], and SDL [5] are formal languages for protocol specification. Formal specifications are useful for verification [36], test case generation [3], conformance testing [40], etc. Multiple techniques such as simulation, modeling, and formal analysis are applied to study a particular system in [33]. Various tools for specification [23] and analysis of specific properties [1] exist. Issues related to design for testability of software are in [6]. Techniques for different emulation interfaces can be used for conformance testing protocol implementations [74].

The work in these areas is complementary to our focus on the design and development of routing protocols. Protocol specification, testing, and verification techniques are methodological approaches to the design of protocols. As users, as well as developers, of VINT we will draw on such methodologies during the protocol design process.

Integrated Services

Providing support for integrated services in the Internet has been the focus of research efforts for several years. Work in this area can generally be classified into several distinct areas. First and foremost, the Internet service model must be extended to include one or more real-time services[16, 66, 67, 68]. Additional mechanism, in the form of algorithms and protocols, will be needed to support these services. For example, reservation setup protocols are needed to transmit applications' requests for enhanced qualities of service to nodes inside the network[71, 76]. Admission control algorithms operating at the network nodes will determine whether new requests for these services can be accommodated[32, 34, 39, 43]. Once resource reservations are established, packet scheduling algorithms must ensure that applications receive the level of service they requested[25, 38, 46, 59, 73, 75]. Finally, the routing infrastructure may be enhanced to support integrated services. Currently, new multicast routing protocols are being proposed[4, 20], and this work may be extended in the future to support integrated services applications.

Collectively, these mechanisms represent a fundamental change in the Internet architecture. While individual areas have received considerable attention in isolation, and a vast literature on these subjects exists, experience with integrated services algorithms and protocols is still immature. In particular, interactions between different aspects of these architectural innovations, such as routing and reservation, have received considerably less attention. As the work on integrated services moves from research to deployment, having tools to aid in understanding the interactions and relationship of these components will be vital.

F: Proposers' previous accomplishments

Deborah Estrin: USC/ISI

Deborah Estrin and her graduate students have been primary participants in a range of network protocol design efforts, and as part of this effort have used both simulation and testbeds extensively. In particular they were primary participants in the design of Protocol Independent Multicast routing architecture and protocols, RSVP reservation setup protocol, and Source Demand Routing Protocol for inter-domain routing. In addition, Estrin is co-PI on the NSF sponsored Routing Arbiter project for which visualization tools and other network routing diagnostic tools have been developed.

Lee Breslau and Scott Shenker: Xerox PARC

Lee Breslau developed a session-level simulator to study the impact of alternate path routing algorithms in packet-switched networks employing resource reservations. The session-level simulator enabled the testing of topologies much larger than are conventionally used in network simulations. Lee Breslau has also used detailed simulation to study the dynamics of real-time packet scheduling. Breslau has experience implementing and testing network protocols. He has implemented and tested the Virtual Clock scheduling algorithm and a measurement based admission control algorithm for Predictive real-time service in the Dartnet testbed.

Scott Shenker, in collaboration with colleagues at PARC and the DARTnet community, has used simulations and analysis to study several topics in network behavior. Among these have been epidemic algorithms for replicated database management, the impact of fair queueing algorithms on network behavior, TCP dynamics with two-way traffic, and the asymptotic resource consumption of RSVP's reservation styles. In addition he has been active in the design of the integrated services Internet, and was a principal contributor to both the CSZ service model and the RSVP reservation protocol.

Sally Floyd, Kevin Fall, and Steven McCanne: LBNL

Steven McCanne and Sally Floyd developed the network simulator ns, and Steven McCanne built the network animator nam. Ns is accompanied by a suite of validation tests that can be used to illustrate the congestion control algorithms implemented by the various flavors of TCP, as well as the different gateway scheduling and queue management algorithms. Sally Floyd's research on phase effects emphasized the difficulties in interpreting the results of simulations too literally, particularly when small changes in the initial conditions can results in large changes in the numerical results.

Sally Floyd and other members of the Network Research Group have used simulations, often accompanied with analysis, to investigate a wide range of network dynamics: phase effects in packet-switched gateways, TCP connections with multiple congested gateways, RED (Random Early Detection) gateways, link-sharing and resource management models, the dynamics of TCP over ATM, and long-range dependence in aggregate TCP traffic. Kevin Fall, Sally Floyd, and Steven McCanne have implemented TCP with SACK in ns, and have used ns to compare the performance of different TCP congestion control algorithms. Using a different simulator, Sally Floyd has used simulations to explore the behavior of the request/repair algorithms in SRM in large-scale scenarios of randomly-created topologies.

G: Description of the facilities

USC/ISI

We are requesting three workstations with high end graphics capabilities to support the two programmers and project manager at ISI. In addition we are requesting two compute servers with large amounts of memory to support simulations (both local to ISI and remotely)

ISI maintains a large University research computing center. ISI has built a reputation for excellence and efficiency in both experimental computer services and production services. ISI originally developed, provided, and now helps to support, many mature software packages for the entire Internet community. It contains a variety of computers, from mainframes to microcomputers. ISI is connected to several wide-area networks, including the Internet, SCAN, DARTNET, and Los Nettos.

ISI is the home of the MOSIS project, which provides the academic VLSI research community with low-cost, rapid-turnaround fabrication of VLSI chips and printed circuit boards. Through the FAST project, ISI also maintains an automated broker for standard electronics parts ordering.

ISI is also home to the Integrated Systems Laboratory (ISL), which has demonstrated advanced packaging and electronics system engineering capabilities. This laboratory has established a local technology base for advanced packaging, which has provided characterization and models for new technologies as well as procedures and sources for assembly and testing of prototype and production systems.

Xerox

We are requesting one high-end multiprocessor workstation to support visualization and simulation.

Xerox PARC has a long tradition of important contributions to computer science research. Among recent accomplishments, its network research group has developed significant software for the Internet community, including implementations of multicast routing protocols used in the MBone and nv, a network video tool. It has been an active participant in the development of new standards, including IPv6. In addition, researchers at Xerox have played a major role in the development of an integrated services architecture for the Internet, as well as the necessary protocols and algorithms to support that architecture. Xerox has developed an ATM switch and contributed to ATM standardization efforts.

Xerox is connected to both the Dartnet IP network and the BAGNet ATM network testbeds.

LBNL

We are requesting one high-end workstation, a multiprocessor machine that can simultaneously do animation and visualization along with simulation. We are also requesting one low-end workstation for a desk-top machine for one of the researchers.

Lawrence Berkeley National Laboratory is a U. S. Dept. of Energy, Energy Research laboratory with programs in most fields of basic and applied science. LBNL's Information and Computing Sciences Division programs include research on high-speed data transport techniques that are now used in almost every commercial TCP/IP protocol implementation, new methods of congestion control for global-scale networks, and technology for the large-scale use of Internet-based multimedia applications. Other research areas contributing to the development of a National Information Infrastructure that will let scientists use national research facilities from any location include regional health care information systems, energy demand management systems, national "collaboratories", high speed distributed systems, and distributed, on-line systems for analyzing digital video. ICSD also has nationally recognized programs in visualization of scientific data, software engineering and data management.

Physical facilities for network and distributed systems research include numerous workstations, a HIPPI LAN, several ATM LANs, and direct connection to the BAGNet ATM network.

H: Summary of the qualifications of key personnel

Deborah Estrin: USC/ISI

Deborah Estrin is currently an Associate Professor of Computer Science at the University of Southern California in Los Angeles where she joined the faculty in 1986. Estrin received her Ph.D. (1985) and M.S. (1982) from the Massachusetts Institute of Technology from M.I.T. and her B.S. (1980) from U.C. Berkeley. In 1987, Estrin received the National Science Foundation, Presidential Young Investigator Award for her research in network interconnection and security. Estrin is a co-PI on the NSF Routing Arbiter project at USC's Information Sciences Institute. She co-chairs the Source Demand Routing Working Group of the IETF and is an active participant in the Inter-Domain Multicast Routing and RSVP working groups. Estrin is a member of the ACM, IEEE, and AAAS. She has served on several panels for the National Science Foundation and National Research Council/CSTB, and is currently a member of ARPA's ISAT. Estrin was one of the founding Editors of Wiley's Journal of Internetworking Research and Experience and is currently an editor for the ACM/IEEE Transactions on Networks and program co-chair of ACM Sigcomm '96.

Lee Breslau and Scott Shenker: Xerox PARC

Lee Breslau is currently Member of the Research Staff at the Xerox Palo Alto Research Center, which he joined in 1994. He received his Ph.D. (1995) and M.S. (1989) from the University of Southern California, and his B.A. (1981) from the University of Pennsylvania. His graduate research was in the area of policy routing protocols for large internets and routing algorithms to support resource reservations. His current research interests include resource reservation protocols, scheduling and admission control algorithms for real-time service, and priority dropping algorithms. Breslau is active in the IETF Integrated Services and RSVP Working Groups. Breslau is a member of the ACM. He is supported primarily by the Advanced Research Projects Agency under contract DABT63-94-C-0073.

Scott Shenker is currently a Principal Scientist at Xerox Corporation's Palo Alto Research Center. He received his Sc. B (1978) from Brown University, his Ph. D. (1983) in theoretical physics from the University of Chicago, and spent the 1983-4 academic year at Cornell University as a Post-Doctoral Associate. He also is an adjunct associate professor at the University of Southern California. He has been co-chair of the Integrated Services Working Group of the IETF, and he will be program co-chair of ACM Sigcomm '97. He is partially supported by the Advanced Research Projects Agency under contract DABT63-94-C-0073.

Sally Floyd and Kevin Fall: LBNL

Sally Floyd is a staff scientist in the Network Research Group at LBNL. She has worked at LBNL since 1983, and has been a member of LBNL's Network Research Group since 1990. She received her Ph.D. (1989), M.S. (1987), and B.A. (1971) from the University of California at Berkeley, where her research field for her graduate studies was in theoretical computer science. Floyd is active in the IETF in the transport area directorate, in the TCP Large Windows and Integrated Services Working Groups, and in the IRTF's End-to-End Research Group. She is a member of ACM, IEEE, and CPSR, and has served on several panels for the National Science Foundation. She is the vice-chair of SIGCOMM and a program co-chair of ACM Sigcomm '96. She is supported largely by the Director, Office of Energy Research, Scientific Computing Staff, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

Kevin Fall is a Staff Computer Scientist in the Network Research Group at Lawrence Berkeley National Laboratory and a lecturer for the Computer Science Division at UC Berkeley. He received his Ph.D. (1994) and M.S. (1991) degrees in Computer Science from UC San Diego, and his BA degree in Computer Science from UC Berkeley (1988). He held concurrent postdoctoral positions with UCSD and MIT before joining LBNL in September 1995. His research interests include issues of scale in computer networks, protocol implementation and performance, security, multimedia applications and operating systems. Kevin has participated in a number of large-scale systems and networking projects, including the Berkeley UNIX project, MIT's Project Athena, the CASA Gigabit Testbed, and the Sequoia 2000 Project. He is a member of ACM, IEEE, and the Usenix Association and has consulted for the San Diego Supercomputer Center and the Center for Communications Research. He is supported primarily by the Director, Office of Energy Research, Scientific Computing Staff, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

I: Cost breakdown

References

[1] T. Abe, M. Hayashi, and S. Nojo. A software tool to support the reliability design and evaluation of telecommunication networks. *IEEE Journal on Selected Areas in Communications*, 12(2):345–354, February 1994.

[2] C. Alaettinoğlu, A.U. Shankar, K. Dussa-Zeiger, and I. Matta. Design and implementation of mars: a routing testbed. *Internetworking: Research and Experience*, 5:17–41, 1994.

[3] A. Avritzer and E.J. Weyuker. Generating test suites for software load testing. In *Proceedings of the International Symposium on Software Testing and Analysis*, pages 44–57, 1994.

[4] Tony Ballardie, Paul Francis, and Jon Crowcroft. An architecture for scalable inter-domain multicast routing. In *Proceedings of ACM Sigcomm*, pages 85–95, September 1993.

[5] F. Belina and D. Hogrefe. The CCITT-specification and description language SDL. *Computer Networks and ISDN Systems*, 16, 1989.

[6] R.V. Binder. Design for testability in object-oriented systems. *Communications of the ACM*, 37(9):87–101, September 1994.

[7] G. V. Bochmann and A. Petrenko. Protocol testing: Review of methods and relevance for software testing. In *Proceedings of the International Symposium on Software Testing and Analysis*, pages 109–124, 1994.

[8] Bonomi, F., and Fendick, K., The rate-based flow control framework for the available bit rate ATM service. *IEEE Network*, March/April 1995.

[9] L.S. Brakmo and L.L. Peterson. Experiences with network simulation. *ACM SIGMETRICS*, May 1996. To be published.

[10] L. Brakmo, S. O'Malley, and L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance In *Proceedings of ACM Sigcomm*, 1994.

[11] R.E. Bryant. Simulation of packet communiaction architecture computer systems. Technical Report MIT/LCS/TR-188, Massachusetts Institute of Technology, 1977.

[12] R. Cáceres, P. Danzig, S. Jamin, and D. Mitzel, *Characteristics of Wide-Area TCP/IP Conversations*, ACM SIGCOMM '91, September 1991, pp. 101-112.

[13] K.M. Chandy and J. Misra. Distributed simulation: A case study in design and verification of distributed programs. *IEEE Transactions on Software Engineering*, SE-5(5):440–452, September 1979.

[14] K.M. Chandy and R. Sherman. Space, time and simulation. In *Proceedings of the SCS Multiconference on Distributed Simulation*, volume 21, pages 53–57, March 1989.

[15] W.A. Christopher, S.J. Procter, and T.E. Anderson. The nachos instructional operating system. Nachos is available at `ftp://sprite.berkeley.edu/nachos/`.

[16] David D. Clark, Scott Shenker, and Lixia Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. In *Proceedings of ACM Sigcomm*, pages 14–26, August 1992.

[17] M. Coleman and D. Parker. Aesthetics-base Graph Layout for Human Consumption. *Software – Practice and Experience*, 26(1), 1994.

[18] CPSim: A high-performance discrete-event simulation tool. `http://www.wdn.com/bti-sim/`, 1994.

[19] P. Danzig and S. Jamin. *tcplib: A library of TCP/IP Traffic Characteristics*, USC Networking and Distributed Systems Laboratory TR CS-SYS-91-01, October, 1991, ftp://catarina.usc.edu/pub/jamin/tcplib.

[20] Stephen Deering, Deborah Estrin, Dino Farinacci, Van Jacobson, Ching-Gung Liu, and Liming Wei. An architecture for wide-area multicast routing. In *Proceedings of ACM Sigcomm*, pages 126–135, September 1994.

[21] Alan Demers, Srinivasan Keshav, and Scott Shenker. Analysis and simulation of a fair queueing algorithm. *Internetworking: Research and Experience*, 1(1):3–26, January 1990.

[22] A. Dupuy, J. Schwartz, Y. Yemini, and D. Bacon. NEST: A network simulation and prototyping testbed. *Communications of the ACM*, 33(10):64–74, October 1990.

[23] P. v. Eijk. LOTOS tools based on the Cornell Synthesizer Generator. In *Protocol Specification, Testing, and Verification IX*, pages 275–286. IFIP, Elseiver Science Publishing B.V. (North-Holland), 1989.

[24] Fall, K. and Floyd, S., *Comparisons of Tahoe, Reno, and Sack TCP*, Technical report, December 1995. URL ftp://ftp.ee.lbl.gov/papers/sacks.ps.Z.

[25] Domenico Ferrari and Dinesh C. Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE Journal on Selected Areas in Communications*, 8(3):368–379, 1990.

[26] Floyd, S., *Simulator Tests*, July 1995. URL ftp://ftp.ee.lbl.gov/papers/simtests.ps.Z.

[27] Floyd, S., *Ns Simulator Tests for Class-Based Queueing*, February 1996. URL ftp://ftp.ee.lbl.gov/papers/cbqsims.ps.Z.

[28] Floyd, S., and Jacobson, V., *On Traffic Phase Effects in Packet-Switched Gateways*, Internetworking: Research and Experience, V.3 N.3, September 1992, p.115-156.

[29] Floyd, S., Jacobson, V., Liu, C., McCanne, S., and Zhang, L., *A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing*, ACM SIGCOMM 95, August 1995, pp. 342-356.

[30] R.M. Fujimoto. Parallel discrete event simulation. *Communications of the ACM*, 33(10):30–53, October 1990.

[31] G. Gaujal, A.G. Greenberg, and D.M. Nicol. A sweep algorithm for massively parallel simulation of circuit-switched networks. *Journal of Parallel and Distributed Computation*, 18(4):484–500, August 1993.

[32] R. Gibbens, F. Kelly, and P. Key. A decision-theoretic approach to call admission control in ATM networks. *IEEE Journal on Selected Areas in Communications*, 13(6):1101–1114, 1995.

[33] R. Groz, C. Jard, and C. Lassudrie. Attacking a complex distributed algorithm from different sides: An experience with complementary validation tools. In *Protocol Specification, Testing, and Verification IV*, pages 3–17. IFIP, Elseiver Science Publishing B.V. (North-Holland), 1984.

[34] Roch Guerin, Hamid Ahmadi, and Mahmoud Naghshineh. Equivalent capacity and its application to bandwidth allocation in high-speed networks. *IEEE Journal on Selected Areas in Communications*, 9(9):968–981, September 1991.

[35] A. Heybey. Netsim Manual. MIT, 1989.

[36] T. Higashino, K. Ninomiya, T. Kimoto, K. Taniguchi, and M. Mori. Automated verification of equivalence of protocol machines. In *Protocol Specification, Testing, and Verification IX*, pages 235–246. IFIP, Elseiver Science Publishing B.V. (North-Holland), 1989.

[37] N.C. Hutchinson and L.L. Peterson. The x-kernel: An architecture for implementing network protocols. *ieeese*, 17(1):64–76, January 1991.

[38] J. Hyman, A. Lazar, and G. Pacifici. A Separation Principle Between Scheduling and Admission Control for Broadband Switching. *IEEE Journal on Selected Areas in Communications*, 11(4), pp 605-616, 1993.

[39] Jay Hyman, Aurel A. Lazar, and Giovanni Pacifici. Joint scheduling and admission control for ATS-based switching nodes. In *Proceedings of ACM Sigcomm*, pages 223–234, August 1992.

[40] ISO. *Conformance Testing Methodology and Framework*, ISO/TC97/SC21 N 2^{nd} DP9646 edition, December 1987.

[41] ISO. *Estelle, a FDT Based on an Extended State Transition Model*, ISO TC97/SC221 DIS 9074 edition, September 1987.

[42] ISO. *LOTOS, a FDT Based on Temporal Ordering of Observational Behaviour*, ISO TC97/SC21 DIS 8807 edition, July 1987.

[43] S. Jamin, P. Danzig, S. Shenker and L. Zhang. A measurement-based admission control algorithm for integrated services packet networks. In *Proceedings of ACM Sigcomm*, September 1995.

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

[45] D. Jefferson and P. Reiher. Supercritical speedup. In *Proceedings of the* 24^{th} *Annual Simulation Symposium*, pages 159–168, New Orleans, Louisiana, U.S.A., April 1991.

[46] C. Kalmanek, H. Kanakia, and S. Keshav. Rate Controlled Servers for Very High-Speed Networks. In *Proceedings of GlobeCom '90*, pp. 300.3.1-300.3.9, 1990.

[47] G. Kane. *MIPS R2000 RISC Architecture*. Prentice Hall, 1987.

[48] Khoros. `http://www.khoros.unm.edu/`. Commercial Visualization Software.

[49] Kung, H. T., and Morris, R., Credit-Based Flow Control for ATM Networks. *IEEE Network Magazine*, March/April, 1995.

[50] A. Lapone, N. Maxemchuck, and H. Schulzrinne. The bell laboratories network emulator, August 1993.

[51] N. Manjikian and W.M. Loucks. High performance parallel logic simulation on a network of workstations. In *Proceedings of the Workshop on Parallel and Distributed Simulation*, volume 23, pages 76–84, San Diego, California, U.S.A., July 1993.

[52] McCanne, S., and Vetterli, M., *Joint Source/Channel Coding for Multicast Packet Video*, In the Proceedings of the IEEE International Conference on Image Processing, October, 1995, Washington, DC.

[53] S. McCanne and S. Floyd. ns—Network Simulator.
Available at `http://www-nrg.ee.lbl.gov/nrg/nrg.html`.

[54] D. Mitzel, D. Estrin, S. Shenker, and L. Zhang *A Study of Reservation Dynamics in Integrated Services Packet Networks*, Infocom '96.

[55] D. Mitzel, D. Estrin, S. Shenker, and L. Zhang *An Architectural Comparison of ST-II and RSVP*, Infocom '94.

[56] Mogul, J., *The Case for Persistent-Connection HTTP*, SIGCOMM 95, pp. 299-314, August 1995.
development.

[57] W.M. Nandy, B.and Loucks. On a parallel partitioning technique for use with conservative parallel simulation. In *Proceedings of the Workshop on Parallel and Distributed Simulation*, volume 23, pages 43–51, San Diego, California, U.S.A., July 1993.

[58] G.J. Nutt. Distributed simulation design alternatives. In *Proceedings of the SCS Multiconference on Distributed Simulation*, volume 22, pages 51–55, San Diego, California, U.S.A., January 1990.

[59] Abhay K. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. *IEEE/ACM Transactions on Networking*, 2(2):137–150, April 1994.

[60] Paxson, V., *Empirically-Derived Analytic Models of Wide-Area TCP Connections*, IEEE/ACM Transactions on Networking, Vol. 2 No. 4, August 1994.

[61] Paxson, V., *Growth Trends in Wide-Area TCP Connections*, IEEE Network, Vol. 8 No. 4, pp. 8-17, July 1994.

[62] Paxson, V, and Floyd, S., *Wide-Area Traffic: The Failure of Poisson Modeling*, IEEE/ACM Transactions on Networking, Vol. 3 No. 3, pp. 226-244, June 1995.

[63] Prophesy—detailed description. `http://www.csn.net/abstraction`, 1994. Commercial package.

[64] M. Rosenblum, S.A. Herrod, E. Witchel, and A. Gupta. Complete computer system simulation: The SimOS approach. *IEEE Parallel and Distributed Technology*, 1996. To be published.

[65] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. Internet Draft.

[66] S. Shenker, C. Partridge, and J. Wroclawski. Specification of Controlled Delay Quality of Service. Internet Draft, November 1995.

[67] S. Shenker and C. Partridge. Specification of Guaranteed Quality of Service. Internet Draft, December 1995.

[68] S. Shenker, C. Partridge, B. Davie, and L. Breslau. Specification of Predictive Quality of Service. Internet Draft, November 1995.

[69] D. Sidhu, T. Fu, R. Agarwal, A. Crain, and S. Carton. SimuNet: A tool for simulating high speed networks, May 1994. Commercial package; details available at +1 410 750 2253, or by email to telenix@access.digex.net.

[70] M. Thomas and E. Zegura Generation and Analysis of Random Graphs to Model Internetworks. Technical Report GIT-CC-94-46, Georgia Institute of Technology, 1994.

[71] Topolcic, C., Experimental Internet Stream Protocol: Version 2 (ST-II). *Internet RFC 1190*, October 1990.

[72] C. Tropper and A. Boukerche. Parallel simulation of communicating finite state machines. In *Proceedings of the Workshop on Parallel and Distributed Simulation*, volume 23, pages 143–150, San Diego, California, U.S.A., July 1993.

[73] D. Verma, H. Zhang, and D. Ferrari. *Delay Jitter Control for Real-Time Communication in a Packet Switching Network*, In **Proceedings of TriCom '91**, pp 35-43, 1991.

[74] H.K. Zeng and D. Rayner. Gateway testing techniques. In *Protocol Specification, Testing, and Verification IV*, pages 637–653. IFIP, Elseiver Science Publishing B.V. (North-Holland), 1984.

[75] L. Zhang. VirtualClock: A new traffic control algorithm for packet switching networks. *ACM Transactions on Computer Systems*, 9(2):101–124, May 1991.

[76] Zhang, L., Deering, S., Estrin, D., Shenker, S., and D. Zappala, *RSVP: A New Resource ReSerVation Protocol*, **IEEE Network Magazine**, September, 1993.

[77] Lixia Zhang, Scott Shenker, and David D. Clark. Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic. In *Proceedings of ACM Sigcomm*, pages 133–147, September 1991.