SCALABLE AND EFFICIENT TECHNIQUES FOR NETWORK MANAGEMENT AND NETWORK SIMULATION

By

Rashim Gupta

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Approved:

Boleslaw K. Szymanski, Ph.D.
Thesis Adviser

Rensselaer Polytechnic Institute
Troy, New York

April 2003
(For Graduation May 2003)

CONTENTS

LIST OF TABLES iii

LIST OF FIGURES iv

ACKNOWLEDGMENT v

ABSTRACT vi

1. Introduction and Historical Review 1
   - 1.1 Network Management: Previous Work 1
   - 1.2 Network Simulation: Previous Work 4

2. Network Management using DOORS 6
   - 2.1 Introduction 6
   - 2.2 The DOORS System and Architecture 6
     - 2.2.1 Repository 7
     - 2.2.2 Mobile Agents and the Polling Station 8
     - 2.2.3 Other Components 8
   - 2.3 Results 8
     - 2.3.1 Isolated Network Tests 9
     - 2.3.2 The Standard SNMP Transaction 12
     - 2.3.3 The DOORS Transaction 12
     - 2.3.4 Preprocessing Case 13
     - 2.3.5 Comparisons of all methods 14
     - 2.3.6 Extended Scalability Evaluation through Simulation 14
     - 2.3.7 Example 16
     - 2.3.8 Single AS Simulation 17
     - 2.3.9 Multiple AS Simulation 20
   - 2.4 Contributions 20

3. Network Simulation using GENESIS 23
   - 3.1 Introduction 23
   - 3.2 Typical Memory usage in network simulation 23
   - 3.3 Concept and Idea of GENESIS 25

3.4 Simulation with Distributed Memory and Proxy Links 27
3.5 Performance Evaluation 28
3.6 Contributions 32
4. Discussions and Conclusions 34
4.1 DOORS: Conclusion and Future Work 34
4.2 Genesis: Conclusion and Future Work 35
LITERATURE CITED 37

LIST OF TABLES

3.1 Memory Usage in SSFNet vs. Topology(Frequency interval=0.0005 sec) . . 24

3.2 Memory Usage in SSFNet vs. Topology(Frequency interval=0.0001 sec) . . 24

LIST OF FIGURES

2.1 DOORS Architecture 7

2.2 Sample Autonomous System (AS) Topology 9

2.3 3 second interval bandwidth usage plot 10

2.4 3 client isolated network case, standard deviation plot 11

2.5 3 second interval autoregressive case, bandwidth usage plot 14

2.6 Joint bandwidth usage plots of 3 second intervals 15

2.7 Architecture and flow diagrams for CDN application. 16

2.8 Illustration of DOORS connection architecture 17

2.9 Standard deviation plot for AS simulation 18

2.10 Average Interval / Packet Drops in AS simulation 19

2.11 Sample USA Internet Topology 20

2.12 Standard deviation plot for USA simulation 22

2.13 Average Interval / Packet Drops in USA simulation 22

3.1 Memory Usage in the SSFNet Simulator 25

3.2 Simulation Execution Scheme in Genesis 26

3.3 Proxy Hosts and Inter-domain Traffic 28

3.4 One campus network 29

3.5 Memory usage of SSFNet and Genesis for simulating different sizes of BGP networks. Memory sizes of Genesis shown above are the requirements for single AS simulator 30

3.6 Memory usage of SSFNet and Genesis for 20-AS BGP network simulations with different send-rates 30

3.7 Speedup achieved for simulations of different BGP network sizes 31

3.8 Speedup achieved for 20-AS BGP network simulations with different send-rates 32

ACKNOWLEDGMENT

I would first like to thank my research advisor Prof. Boleslaw Szymanski. It is his support and encouragement that has made this thesis possible. Working under his guidance has been a great experience and has also increased my maturity level in the field of research.

I would next like to thank Alan Bivens. He was both a mentor and a friend and provided me with guidance when I needed it most and it was great working with him on the DOORs project. Also working with me on the DOORS project were Ingo Mclean and Jerome White and working with both was not only a pleasure but also a lot of fun.

Next, I would like to thank Yu Liu, who has helped me a lot in the Genesis project. A special thanks goes to Dave Kotfilla and Dave Cross for helping with all the lab equipment required.

I would also like to state that a lot of this work is based on my research group's papers and I have used them with the co-authors' permissions. Specifically, I have re-used parts of [7], [8] and [37]. My individual contributions are mentioned in the appropriate chapters.

Last, but definitely not the least, I would like to thank my parents, Rajesh and Renu Gupta, and my sister Richa Gupta. I am really lucky to have such a great family and this thesis is a result of their motivation and support that they have provided me.

ABSTRACT

This work consists of two parts both of which are based on the SSFNet simulator and its extensions. The first part describes DOORs, which is a distributed framework for network management middleware. The goal is to easily distribute functional agents to locations where they may carry on the actions of the management application closer to the managed node. In large networks with multiple managers, problems in a network usually draw attention and management traffic to the problem location. This added management traffic only exaggerates the problem. We show and quantify the benefits of the proposed distribution by implementing several real-time network managers using our distributed framework. We also propose and describe several management techniques, including congestion control and network parameter optimization, which use the distributed framework.

The second part describes Genesis, which is a network simulator based on network decomposition. Genesis combines simulation and modeling in a single execution. It decomposes a network into parts, called domains, and simulation time into intervals, and simulates each network partition independently and concurrently over one interval. After each time interval, information is exchanged by each of the domain simulators. Each domain simulator will model the traffic external to its own domain based on the flow data received from other domains. Domain simulators iterate over the same time interval until they reach a global convergence with controllable precision. Thus, Genesis is a more scalable alternative to simulators which demand huge memory resources to simulate large network topologies. Genesis distributes the memory amongst various machines and thus is a distributed approach to simulation.

CHAPTER 1
Introduction and Historical Review

Efficient Network Management has always been a challenge for networks that scale to the size of the Internet. The traditional protocols like SNMP [23] work well in small networks. However, for larger networks, SNMP acts as a bottleneck and does not work as an efficient tool. The two main problems associated with SNMP are scalability and reliability. SNMP is not scalable since SNMP requests are usually not allowed to be made outside of its own network. Further, there is presently no tool in SNMP by which we can poll for the same data without sending the request again and again. This adds a lot of overhead on the underlying network. SNMP also uses the unreliable UDP which does not offer a guarantee that the packets will be delivered.

To simulate networks to the scale of the Internet is also a challenge to the networking research community. Most of the simulators that are used today, like ns, SSFNet and JavaSim, perform well for small networks but when used for large networks like the Internet tend to take a lot of computing power. ns typically cannot scale to the size of the Internet and specifically cannot simulate the BGP protocol. SSFNet while scalable adds a lot of overhead requesting a lot of primary memory to run a network the size of the Internet.

In this thesis we propose scalable and efficient solutions to both the above problems - Network Management and Network Simulation. The first part of the thesis discusses DOORS that is an efficient Network Management Middleware. The second part of the thesis discusses Genesis that is a network simulator that can scale to networks the size of the Internet. We believe that both these tools provide powerful techniques to administer and simulate large networks.

1.1 Network Management: Previous Work

Some of network management's oldest and most known protocols are SNMP [23] and CMIP[40]. SNMP uses a simple request and reply paradigm where an SNMP client contacts (through UDP) a networking device running an SNMP server with a request. The

sever, in turn, replies to the client with the requested information. SNMP also contains very limited support for a push paradigm by using SNMP traps. SNMP, partially because of its simplicity, has gained a very large share of the network management market share. More information about SNMP can be found at [23]. CMIP [40] is much like SNMP but it is used for an OSI network instead of TCP/IP networks. Because OSI networks are not as popular as TCP/IP networks, standards were created to also use CMIP over TCP, called CMOT [40].

SNMP can be inefficient as a network management protocol when many SNMP request must be made. In many cases, when problems occur, management traffic in the problem area increases, worsening the problem. Problems with the protocol have been addressed by IETF and ISO through modifications of their management architecture. SNMPv2 introduced hierarchical decentralization through the concept of proxy agents [9]. The proxy agent simply acts as a client to a group of devices on behalf of a network management station. Another protocol derived from SNMP, RMON (Remote Monitoring), provides network administrators with an additional level of statistics kept by the RMON agent or probe and retrieved by an SNMP client [28]. The RMON specification defines a set of statistics and functions that can be exchanged between RMON-compliant console managers and network probes. This functionality of RMON to involve other RMON agents, provides network administrators with a more comprehensive and global network-fault diagnosis, planning, and performance-tuning information. Although these decentralized features improve the state of SNMP, they do not provide the desired level of decentralization and functionality needed to cope with large networks [14].

We have just discussed above that SNMP can be inefficient as a network management protocol when many SNMP requests must be made. SNMPv2 introduced hierarchical decentralization through the concept of proxy agents [9]. The proxy agent simply acts as a client to a group of devices on behalf of a network management station. Another protocol derived from SNMP, RMON (Remote Monitoring), provides network administrators with an additional level of statistics kept by the RMON agent or probe and retrieved by an SNMP client [28]. The RMON specification defines a set of statistics and functions that can be exchanged between RMON-compliant console managers and network probes. This functionality of RMON to involve other RMON agents, provides network

administrators with a more comprehensive and global network-fault diagnosis, planning, and performance-tuning information. Although these decentralized features improve the state of SNMP, they do not provide the desired level of decentralization and functionality needed to cope with large networks [14]. RMON uses UDP for its underlying communication and it also requires repeated requests to be sent to get the same information from the device to be polled. The DOORS system, which is discussed in detail in Chapter 2, is better than RMON since it uses TCP for communication and also the request has to be sent only once in the form of a mobile agent. Moreover, some algorithms can also be added at the agent side which can compute the important results and send only the final results instead of the polled data back to the client. This helps reduce the bandwidth usage.

Barotto et al. [2] designed a network information retrieval tool using Java, CORBA, and SNMP in which web clients use CORBA to submit SNMP requests to an object representing the network. These requests are translated to standard SNMP requests and then retrieved from the managed node. After successful retrieval, the values are sent back the the web client. The system was designed as a CORBA proxy to allow administrators to monitor or configure SNMP parameters from a web browser.

Fuggetta et al. [14], in a case study show the benefits and disadvantages of different mobile code paradigms, discuss the advantages of an agent-based network management system over the client-server model of SNMP. The authors conduct a mathematical analysis of the network traffic used between SNMP, code on demand, remote execution, and mobile agent systems and determined that, while the effectiveness of code mobility depends heavily on the characteristics of the task, mobile code paradigms such as mobile agents can avoid bandwidth consumption problems in cases when management functionality is most important. Typically, these cases include problem situations where the manager will increase its interaction with the devices and possibly upload configuration changes, increasing the congestion present. Consequently, congestion as an abnormal status, is likely to trigger notifications to the management system, which worsens network load.

Bauer et al. [4] use a repository for management of distributed applications in the MANDAS (Management of Distributed Applications and Systems) project. The authors

concentrate on an area other than network management, but there are many similarities with their work and ours. They use a Management Information Repository (MIR) to hold information about their distributed applications. However, they have only one centralized repository. Our implementation uses distributed repositories with advanced communication methods for transferring data between the repositories.

Harista et al. [16] describe MANDATE (MAnaging Networks Using DAtabase TEchnology), which is a proposed MIB (Management Information Base) to support network management. The authors propose to have operators interact solely with their MIB for network management. Their MIB holds information about the network, similar to our network of repositories. Implementation of MANDATE is client-server based with sophisticated client caching. Our implementation is based on distributed agents with caching between repositories for performance and availability.

1.2 Network Simulation: Previous Work

The Internet is unique in its size, support for seamless interoperability, scalability and affinity for drastic change. The collective computational power of all Internet routers involved in network traffic routing makes the Internet the most powerful computer in the world. Network packets are processed and routed in a very short time in the order of a fraction of a second. These very characteristics make the Internet hard to simulate efficiently. The current state-of-the-art Network Simulators are ns, SSFNet and PDNS. While each of these have been successful in simulating specific topologies, none of these are truly scalable as either they cannot simulate networks the size of the Internet or if they can scale, then these simulators need a lot of memory and computing power.

To overcome these problems, we propose Genesis, which combines simulation and modeling in a single execution. Genesis uses a high granularity synchronization mechanism between parallel simulations of the parts of a network. This mechanism uses checkpointed simulation state to iterate over the same time interval until convergence. It also replaces individual packet data for flows crossing the network partitions with statistical characterization of such flows over the synchronization time interval. We had achieved significant performance improvement over the sequential simulation for simulations with TCP and UDP traffic. However, this approach can not be used directly to simulate dy-

namic routing protocols that use underlying network for exchanging protocol information, as no packets are exchanged in Genesis between simulated network parts. We have developed a new mechanism to exchange and synchronize BGP routing data among distributed Genesis simulators. The extended Genesis allows simulations of more realistic network scenarios, including routing flows, in addition to TCP or UDP data traffic.

Large memory size required by simulation software hinders the simulation of large-scale networks. Based on our new support of distributed BGP simulation, we developed an approach to construct and simulate networks on distributed memory using Genesis simulators in such a way that each participating processor possesses only data related to the part of the network it simulates. This solution supports simulations of large-scale networks on machines with modest memory size.

Genesis offers a novel approach to scalability and efficiency of parallel network simulation and we demonstrate that it can be applied to protocols that use traffic feedback to adjust the source traffic rate. The described method can be seen as a variant and modification of a general scheme for optimistic simulation referred to as Time-Space Mappings. Our approach partitions the network into domains and the simulation time into intervals. Each domain is simulated independently of and concurrently with the others over the same simulation time interval. At the end of each interval inter-domain flow delays and packet losses are exchanged between domains that iterate over the same time interval until the exchanged information converges to a constant value within the prescribed precision. After convergence, all domains start working on simulating the next time interval. This approach is particularly efficient if the simulation cost grows faster than linearly as a function of the network size, which is the case for computer networks in general and the Internet in particular. Genesis approach is independent of simulators used for simulating domains. It is useful in all applications in which the speed of the simulation is of essence, such as: on-line network simulation, network management, ad-hoc network design, emergency network planning, or Internet simulation.

CHAPTER 2
Network Management using DOORS

2.1 Introduction

Rapid growth of computer network sizes and uses necessitate analysis of network application middleware in terms of its scalability as well as performance. In this chapter we analyze a distributed network management middleware based on agents that can be dispatched to locations where they can execute close to the managed nodes. Our system allows multiple managers to probe problem areas without adding management traffic to them. We discuss and quantify the benefits of such distribution by implementing real-time network managers using our distributed framework.

The main result of this chapter is a comparison of scalability and efficiency of our agent based management middleware and traditional SNMP based data collection. To this end, we measured traffic in both real and simulated networks. In the later case, we designed, used and described here a method of segmenting simulated applications flows into smaller "separable" flows to simplify design of simulations.

2.2 The DOORS System and Architecture

To support the decentralization of network monitoring tools, we propose DOORS (Distributed Online Object Repositories) that facilitate scalable collection and manipulation of several forms of network data. The DOORS system manages and schedules client data requests at its repositories. The repositories then configure mobile agents to travel to a node very close to the managed device. Once the agent arrives at its destination, it polls the managed device, performs client requested procedures, and sends the result back to the repository to be forwarded to respective clients. The use of agents allows us to place more functionality into what the client perceives as the "request." A DOORS client may ask for various forms of direct network data, as well as any function $f(t, x_0, x_1, ..., x_n)$ of network data and time, where the argument x_t denotes data collected at discrete time t for $0 \leq t \leq n$. Typically, the function f is a statistical manipulation of network monitoring data. As the agent executes some functions at the remote location, the DOORS system

effectively moves the computation closer to the data. This solution drastically reduces the total bandwidth used by any tools which monitor large networks.

At the core, the DOORS system provides an efficient, fault tolerant method for the acquisition, management, manipulation, aggregation, and caching of network data and objects [5]. We aim to make DOORS scalable to the largest existing networks with hundreds of thousands of nodes that cannot be managed using traditional, strictly hierarchical approaches.

One of the main goals of the DOORS system is to provide its services with an absolute minimum impact on the network. The repository itself assists in this goal by the functionality it passes to the client through the requesting language. In the cases discussed in this paper, each client request requires collection of a certain set of SNMP variables $S = \{v_1, v_2, ..., v_n\}$ every $t_{interval}$ seconds for a total duration of t_{total} seconds. To satisfy such a request, DOORS system uses a push method of the client requesting data only once but having the data delivered to the client many times.

DOORS system uses several components to retrieve and process network data. A simple graphical representation of these components working together for a local data request (involving only one repository) is given in Fig. 2.1. In this section, we will briefly discuss the purpose of these components.

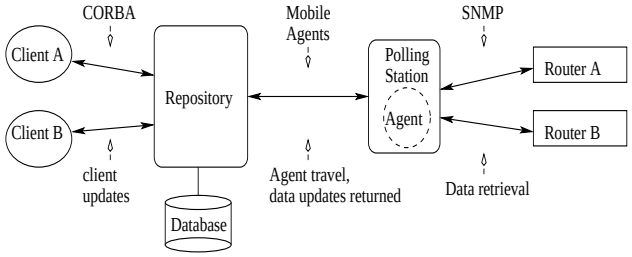


Figure 2.1: DOORS Architecture

2.2.1 Repository

The repository controls agents and coordinates requests from different clients as well as other repositories. Depending on the appropriate action for a particular client request, the repository will either consult the database, create a new agent, send a message to expand the role of an existing agent, or simply add another client to the list of already

distributed data [6].

2.2.2 Mobile Agents and the Polling Station

The mobile agents travel from the repository to a polling station to request the actual data from the destination object at a close range. Agents reduce the network load by passing back only the data necessary for the client and the repository. This is especially useful when additional processing functionality is added to the agent. In such a case, the agent returns only the result of some computation or manipulation of data.

DOORS agents communicate with their sender using the TCP protocol that provides acknowledgment based reliability. In contrast, traditional SNMP uses UDP which allows for undetected data losses during communication. The reliability benefits of TCP are desirable but come at a cost of increased bandwidth needed for acknowledgments (see Section 2.3 for an explanation of how we counteract it).

The polling station is a critical component, because it is difficult to send an agent to sit on the router itself. Many routers use proprietary operating systems. In addition, running the collection programs directly on the routers may introduce unacceptable load on the routers. The polling station simply needs to run an agent server which can receive the agents and allow them to execute their tasks close to the managed node.

2.2.3 Other Components

We also use two minor components on an infrequent basis. We use a CORBA nameserver to resolve CORBA object references for client-repository communication. We also use a superserver to correlate polling stations and managed nodes.

2.3 Results

As previously stated, we retrieve SNMP data from the routers that are targeted in client queries. In this section, we compare the bandwidth used by clients requesting data through the DOORS system with the bandwidth needed by the same requests executed using the traditional SNMP method. We implemented our design on an isolated computer lab configured to represent a small autonomous system (AS) topology, shown in Fig. 2.2. This topology involves three logical networks joined by a backbone consisting of three

core routers. All of the routers in the AS are connected by Ethernet or serial links and run the dynamic routing protocol OSPF [17]. Hosts are connected to some of the routers through 10Mb Ethernet connections.

We compare the amount of bandwidth used when single and multiple clients are present. To measure the traffic incurred across these networks, we monitor the traffic seen by the serial1 interface of core router BB3 (the link on the left side of the top backbone router in Fig. 2.2). When standard SNMP methods are used, clients are on n1 and n2 and we poll the Ethernet interface of router P1R2 (target) in Net 1. When the DOORS system is used, clients are on n1 and n2, the naming server and repository both on n3, the superserver on n4, and the polling station (n7) close to the target (P1R2) on Net 1.

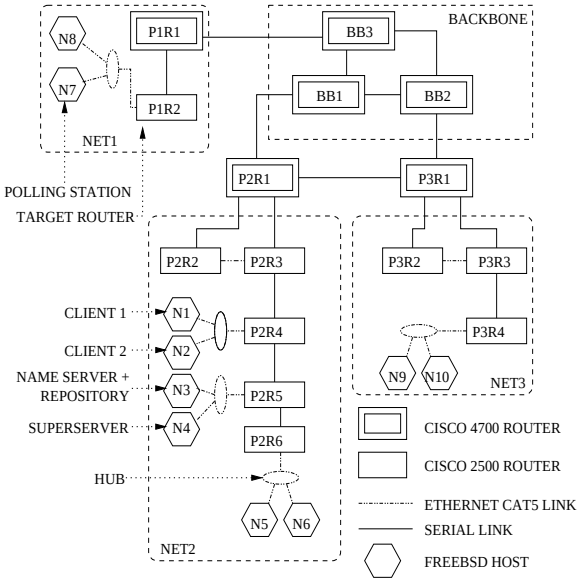


Figure 2.2: Sample Autonomous System (AS) Topology

2.3.1 Isolated Network Tests

To compare the bandwidth usage incurred through data collection under both DOORS and traditional SNMP polling,we collected data at various time intervals including 3, 5, 7 and 10 seconds. We show the savings in bandwidth usage in Fig. 2.3 which shows the data collection at three second intervals under both DOORS and SNMP. We note that for different time intervals, the DOORS system performs much better than SNMP and the results are similar to Fig. 2.3. Moreover, the bandwidth used under DOORS remains ef-

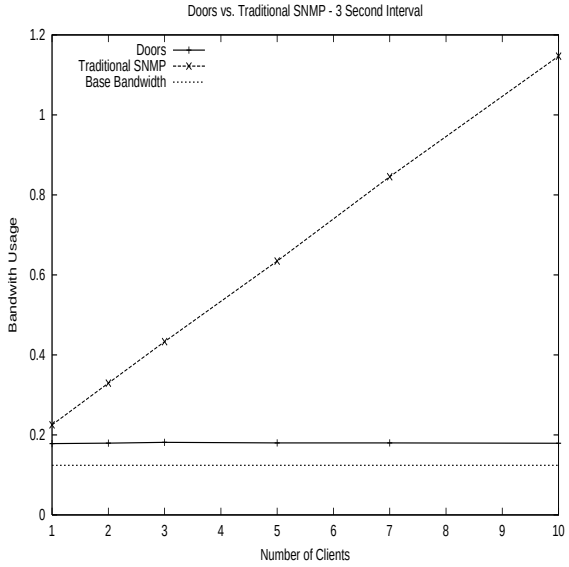


Figure 2.3: 3 second interval bandwidth usage plot

fectively constant regardless of the number of clients, because only one message per the data collection interval is sent back to the repository to be distributed to all clients.

Our goal is two fold, we want to provide an effective way of data collection while putting a minimum strain on the network. At the same time, we want the clients to receive the data in a timely fashion. We have already shown how DOORS meets the first goal. We now examine how the DOORS system impacts the client. We use a simple, first-order statistical analysis of the data received by the client to examine the variance in the times between successive returns of information, or inter-polling time. We compare this variance by calculating the standard deviation of the inter-polling time of the clients. We use two standard deviation statistics, average-based standard deviation and actual-based standard deviation. The average-based standard deviation evaluates the standard deviation through computation of each interval's difference from the sample's average interval (also known as sample standard deviation). This will measure the regularity and smoothness of the inter-polling interval. The actual-based standard deviation involves computation of each interval's difference from the polling interval requested by the client. This deviation statistic measures how close the intervals are to the actual interval requested by the client. The graph describing these statistics under different levels of congestion (currently defined by average link utilization) is shown in Fig. 2.4.

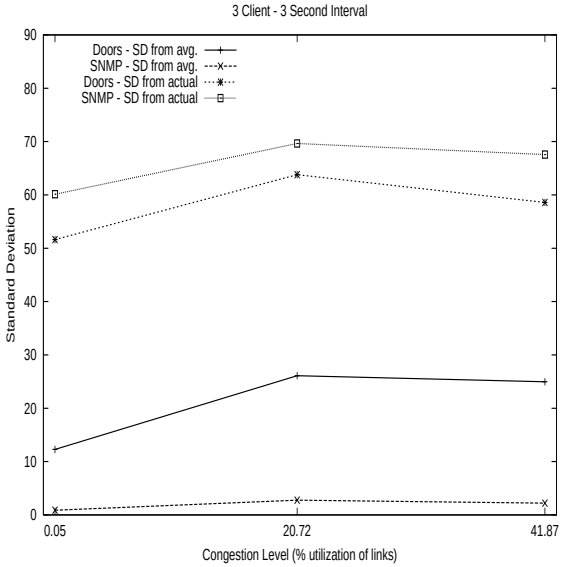


Figure 2.4: 3 client isolated network case, standard deviation plot

The normal SNMP client has a low average-based variance because, using UDP as a transport protocol, its inter-poll time is based only on the network latency and load which are relatively constant in the isolated network.

Because our agents use TCP to send data back to the client, we see a difference in the variance of the actual polling interval perceived at the client. TCP uses many different algorithms, such as control loop based flow control, slow start, congestion avoidance, and its windowing mechanism. The collective use of these algorithms causes delays and oscillations in TCP delivery times compared to that of UDP [18]. However, because the DOORS system sends the configured agent close to the router, the DOORS system does not suffer the delays of traversing the network for the request portion of the request-reply paradigm. Our agent also has an open connection with the repository for long-term message passing. These details allow DOORS to have a fraction of the latency time the normal SNMP client has, and thus reduces its actual-based standard deviation.

As seen in Fig. 2.4, even though the standard deviation of DOORS about the average is higher than that of the SNMP client, the standard deviation about the request, or actual interval is lower than that of the SNMP client. This means that DOORS may be a bit more variable in the exact times it returns the data, but it almost always returns it faster than the SNMP client. The other side of this comment is also true. The SNMP client almost

always takes longer to return the data, but it returns it at a more consistent interval.

The most interesting results come from the single client cases. For each time interval data SNMP needs both a request and a reply (a pull method) while DOORS simply sends the data (a push method). Because of the differences in the underlying protocols and the methods used to get data from the two systems, we will compare the two systems on a transaction basis, where a transaction is simply the client receiving a new instance of data at the polling interval requested. In this section we compare transactions on a packet basis. The number of bytes in each packet were computed by adding 4 bytes for checksum to the measurement obtained from a tcpdump process on a machine in the same network.

2.3.2 The Standard SNMP Transaction

The SNMP transaction involves two packets: $Snmp_{request}$ and $Snmp_{reply}$. According to the SNMP protocol, the same message format, containing SNMP headers followed by name-value pairs, is the basis for both the request and reply. The only difference between the two messages is that value fields are not populated with the actual values in the request packet. Due to SNMP's use of ASN.1 and the BER (Basic Encoding Rules) encoding standards, the difference in bytes between the request and reply depends on the type of data sent and sometimes on the data values themselves [23]. The sizes for the SNMP transaction packets in our experiments are 161 bytes for request and 175 bytes for response for the total of 336 bytes.

2.3.3 The DOORS Transaction

Part of TCP's overhead is its three-way handshake [25]. However, the DOORS agent maintains the connection with the repository over the entire collection period, so the single connection handshake overhead can be amortized over the life of the connection. Therefore, the difference between the bandwidth used by the two methods is in the data messages sent back from the agent. In TCP, data must also be acknowledged with a special packet. In our case this packet is a simple acknowledgment but its size must be 64 bytes (the minimum Ethernet frame size). We have no need to send a request, so the only packet we send is the new data packet, holding the data and some synchronization content

for the agent system. The sizes for the DOORS transaction packets are 122 bytes for data and 64 bytes for acknowledgment, so 186 bytes total, less then 336 bytes required for SNMP. Hence, DOORS is an efficient solution to data collection, lightening the footprint of any network management application.

2.3.4 Preprocessing Case

Many research efforts use SNMP to gather vital network statistics and determine trends in the traffic traveling across the network. DOORS can play an integral role in these applications by reliably collecting the needed data close to the device in question, and doing part or all of the necessary calculations in the agent. To evaluate the contribution of DOORS in such a situation, we use DOORS as a component of the Network Problem Forecasting solution developed by Thottan and Ji [26]. These authors detect changes in traffic patterns using a sequential Generalized Likelihood Ratio (GLR) test by gathering SNMP data in groups of ten, and creating piecewise stationary autoregressive models. The Generalized Likelihood Ratio was then used on the autoregressive coefficients. Once changes are detected using the GLR, the authors correlated the different alarms with (NFS) failures confirmed in system logs.

Thottan and Ji conducted traditional SNMP polls to retrieve the data necessary for the calculations. We can thus modify the DOORS system to implement the windowing and autoregressive calculations in the DOORS agent (called as autoregressive DOORS agent), sending only the likelihood values back to the client for filtering and comparisons. This division of the algorithm saves the cost of $N - 1$ polls across the network per time window, where N is the number of intervals used in each window, because only one set of statistics is sent back per window versus a set of data for every poll. Hence, this solution reduces the bandwidth used by a factor of at least $N - 1$.

As Fig. 2.5 shows, the autoregressive agent uses very little bandwidth, far less than the bandwidth used by standard SNMP polling. In Fig. 2.5, the bandwidth statistic of background traffic caused by the routers using OSPF is 0.126. DOORS cannot fall below that point, but it gets close.

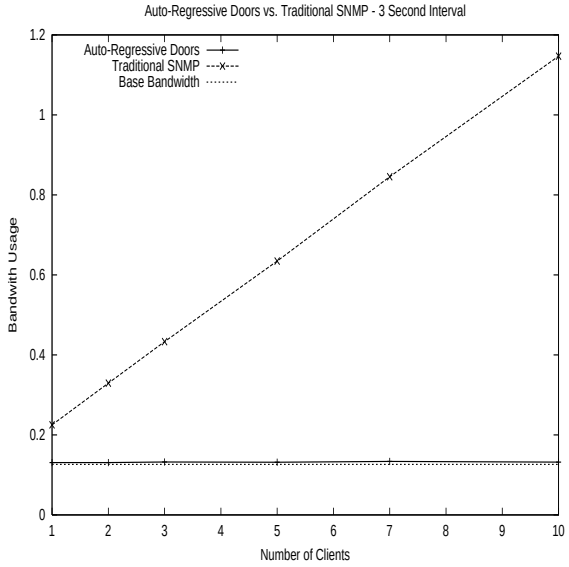


Figure 2.5: 3 second interval autoregressive case, bandwidth usage plot

2.3.5 Comparisons of all methods

The differences in bandwidth usage across the many methods including our preprocessing case are shown in Fig. 2.6. It demonstrates that SNMP clients use considerably more bandwidth than the DOORS client. It also shows that additional savings are achieved when part of the algorithm is assigned to the agent.

2.3.6 Extended Scalability Evaluation through Simulation

To allow more flexibility in our evaluation and extend it to larger topologies (that we cannot test in our isolated lab), we used simulation based on the SSFNet simulator [12]. However, as applications grow in complexity and numbers of connections, the exact behaviors of many simple applications may become very difficult to model. To avoid the hardships of significant simulator code revision, we analyze our application to find appropriate synchronization points whereby we can divide the application flow into smaller, easily simulated component flows. These synchronization points of application cause many applications to execute a sequence of stages. Flows in each stage often happen at non-intersecting times. They frequently happen at separate, non-intersecting physical locations in the network. Even if some flows are active at the same time and possibly at the same place, they can still be modeled separately if their interactions are negligible. Flows

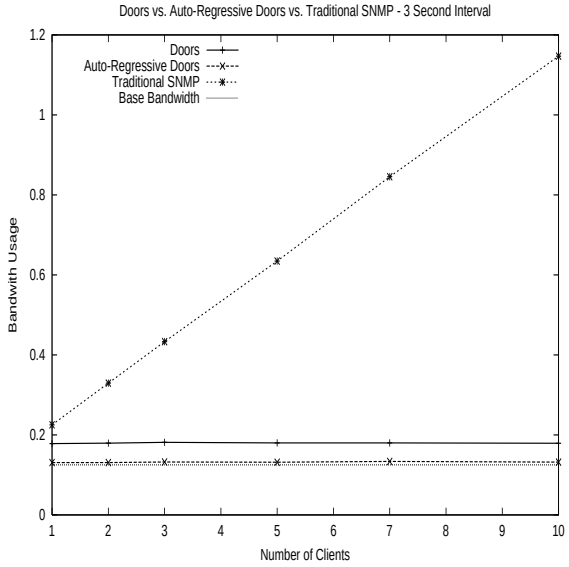


Figure 2.6: Joint bandwidth usage plots of 3 second intervals

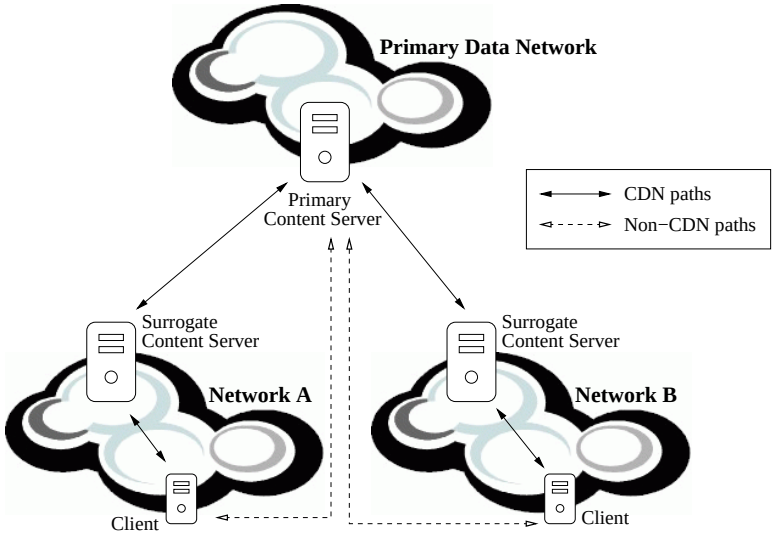
that can be simulated separately from others and still approximate well the behavior of the application are referred to as *separable*.

Separability in networking is not an entirely new concept. To simplify solutions of queuing circuits, queuing separability is used [15]. It enables evaluation of the performance of a complete circuit by evaluating circuits subcomponents in isolation. The performance of the circuit as a whole can then be computed by a combination of these separate solutions [3]. Kleinrock describes the concept of separable performance measures as those that may be expressed simply as a sum of terms, each of which depends only on the flow in a single channel [19]. An extensive discussion of separable queuing network models, including their requirements, limitations, and extensions, can be found in [20].

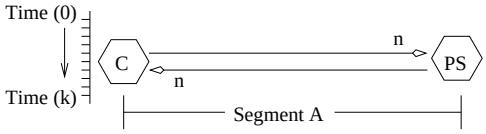
In our approach, separable flows enable the modeler to divide a complex flow of an application into separate, easily simulated flows. Much like in the other approaches, the results of the separate simulations can then be combined to compute overall performance of the entire application. The type of combination needed depends on the type of metrics involved. Often, for flow source-destination delay or loss, it involves a sum or the maximum of the results of separately simulated flows. Combining by a sum is valid whenever the system response is linear for the range of application flows that are of interest.

2.3.7 Example

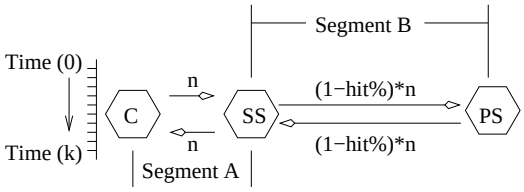
Fig. 2.7(a) shows the architecture of a simple Content Distribution Network (CDN) provider. To quantify the benefit of the CDN provider, the delay of a flow using CDN surrogate servers must be compared to the delay of a flow using Internet paths. These two flows are presented in Fig. 2.7(b and c).



(a) Simple Content Distribution Network(CDN) architecture



(b) Internet path application flow



(c) Surrogate Server path (CDN) application flow

Figure 2.7: Architecture and flow diagrams for CDN application.

The Internet path (Fig. 2.7(b)) represents a simple request/reply communication between a client and a server, so it can be easily simulated directly (without separation). If there are n requests in one direction, there would be n responses in the opposite direction. The CDN path (Fig. 2.7(c)), involving a client, server, and a surrogate server, is a bit more complicated. In the CDN path, the client would issue n requests to a surrogate server, out

of which ($hit\_ratio * n$) would be successfully replied to by the surrogate with no need to go to the primary server. However, the surrogate server would also need to pause its connection with the client while seeking the original information from the primary server $(1 - hit\_ratio) * n$ times. This type of behavior is difficult to model in many generic simulators. Therefore we benefit from dividing the flows at the synchronization point (the surrogate server) for separate simulation

2.3.8 Single AS Simulation

The traditional SNMP data collection uses a single connection at each iteration, so separation in this case is not necessary. For DOORS, however, we divide the frequently occurring parts of our application into three flows (shown in Fig. 2.8).

Flow A is the communication between the client and repository involving one request and n responses where n is the number of iterations. Flow B is the communication between the repository and the polling station involving a possible one-time sending of the agent and n data responses. Flow C is the communication between the polling station and the managed node (router) involving the SNMP request and reply which both happen n times.

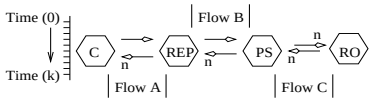


Figure 2.8: Illustration of DOORS connection architecture

In this case:

- $A \bowtie B$ involves both network and host level interaction because flow A shares a small part of the network path of flow B and both flows share the repository. However, we rate the network interaction as minimal because the client is only sent the data after flow B completes its path. The only way the two flows would significantly interact is if the one way trip time of flow B's path + polling interval is less than the one way trip time of flow A's path. The polling intervals are on the order of seconds, while the one-way trip times are on the order of milliseconds, making the above scenario highly unlikely. The interaction the two flows have on the machine only consist of an open connection and has also been deemed minimal.

- $B \bowtie C$ involves both network and host level interactions because they share a small network path and the polling station. Their network interaction is minimal because they are on the same network and both sections' traffic collectively only consist of one more message per time interval than the traditional SNMP method. Their interaction on the polling station is also minimal because it simply conducts the polling and sends the result to the repository.

- $A \bowtie C = 0$ because they are in two totally different networks.

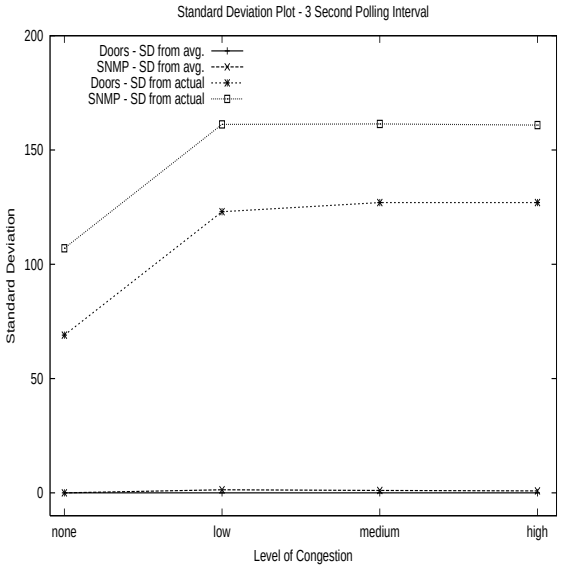


Figure 2.9: Standard deviation plot for AS simulation

In our simulation, we are interested in the effect of the DOORS framework perceived by the client. Therefore, we simply add the delay values from the three separable flows of the DOORS application to compare with the corresponding traditional SNMP requests.

Our simulation results for the AS (the same AS shown in Fig. 2.2) are shown in Fig. 2.9 and Fig. 2.10. In these graphs we show statistical results of simulation under varying degrees of traffic created by background communication agents sending continuous streams. The levels of congestion [none, low, medium, and high] correspond to an increasing number of background communication agents. These background communication hosts exchange large amount of data, adding a lot of queuing delays to the

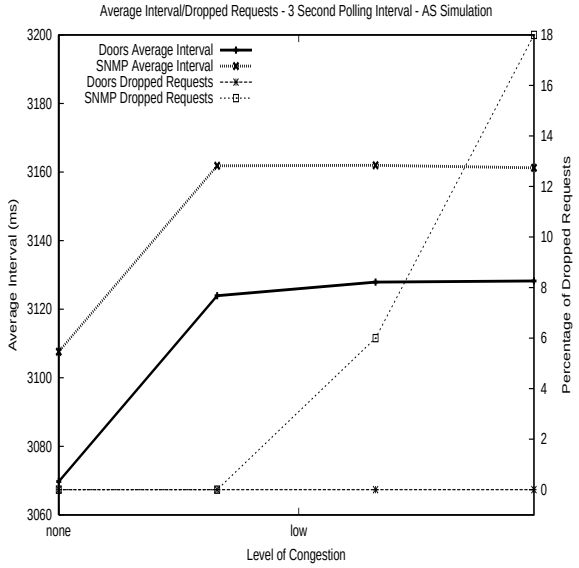


Figure 2.10: Average Interval / Packet Drops in AS simulation

routers and leading to congestion. In case of “none” level of congestion, there are no background hosts in the network and hence the only traffic that flows in the system is that of the DOORS transaction/SNMP polling requests. We then increase the number of background hosts in the subsequent levels of congestion. Thus, at congestion level “high” the network is highly congested resulting in a lot of dropped packets. Using traditional SNMP worsens the problem since it uses UDP for data communication which does not implement congestion control. Also SNMP requires repeated transmission of requests after regular time intervals. DOORS on the other hand uses TCP which does implement congestion control. DOORS also reduces the number of requests since the mobile agent has to be sent only once to the agentserver, which then sends back repeated responses to the client. Fig. 2.9 is similar to the standard deviation plot shown in the isolated network case and shows the variance in the inter-polling interval under varying degrees of traffic. Fig. 2.10 shows the average inter-polling interval on the left-side y axis, while displaying the percentage of dropped request on the right. A dropped request means the client did not receive data for that particular interval due to network load. It is important to note here that the traditional method dropped 6% of the SNMP requests in the medium congestion case and 18% in the high congestion case. In the DOORS case, this does not happen because our TCP layer insures safe arrival.

2.3.9 Multiple AS Simulation

To continue our scalability evaluation in larger topologies, we simulate our system in a USA Internet topology, shown in Fig. 2.11, consisting of 25 virtually identical Autonomous Systems, each containing 1,300 hosts, 27 internal OSPF routers, and one AS BGP boundary router.

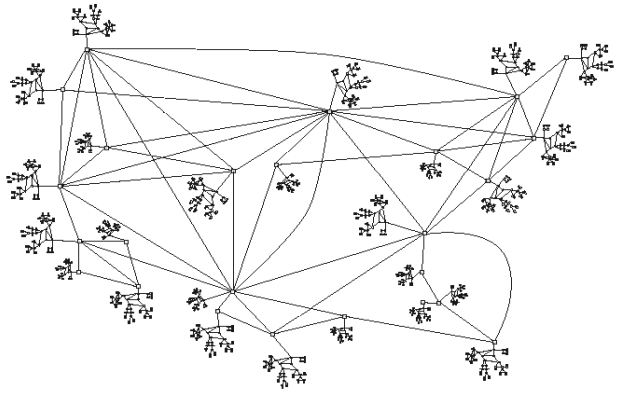


Figure 2.11: Sample USA Internet Topology

We simulate our architecture assuming that multiple managers may exist in the same AS with interest in a trouble spot in another AS. A client analysis of the results is described in Fig. 2.12 and Fig. 2.13 that show the same type of relationship between DOORS and SNMP as Fig. 2.9 and Fig. 2.10, respectively. In the USA topology, the traditional method dropped 4% of the SNMP request in the medium congestion case and 33% in the high congestion case.

2.4 Contributions

My role in this project started with investigating how secure the DOORS system is. This involved a lot of background research in the area of security of mobile agents. I also investigated the source code of the DOORS system and was responsible for quite a few changes in the DOORS system. Gradually my work shifted towards network simulation and I started investigating how to show the scalability of the DOORS system. We first constructed an Autonomous System in our laboratory and my role was to administer the running of the DOORS system and to write the scripts for efficient running of the system. I also wrote a lot of programs to create congestion in the network. The congestion was

monitored using the program tcpdump and DOORS was found to be much better than traditional SNMP.

To continue the scalability of evaluation I next constructed a topology similiar to the DOORS system and ran this under the SSFNet simulator. The topologies varied from one Autonomous System to a network consisting of 25 Autonomous Systems, approximating the network size across the United States. I also ran a lot of tests to congest this network to show that DOORS is especially suitable for networks which are typically congested. I also showed that while there are packet drops in the SNMP protocol, the DOORS system efficiently and successfuly sends the packets across the network.

I also modified the SSFNet simulator to make it suitable for the DOORS system. In the current SSFNet framework, for each response that is to be received by the client, a request has to be sent by the client. However, in the DOORS system, we need multiple responses corresponding to each request. This change was done by me by changing the actual SSFNet simulator code. Changes were also made to the simulator to reflect the separability theory proposed in [7].

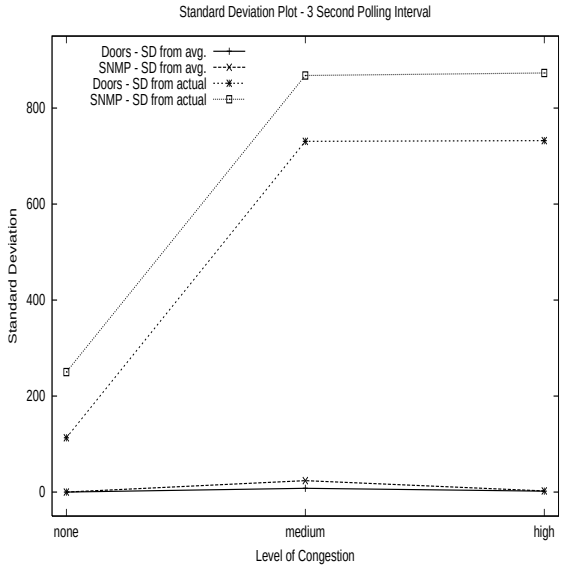


Figure 2.12: Standard deviation plot for USA simulation

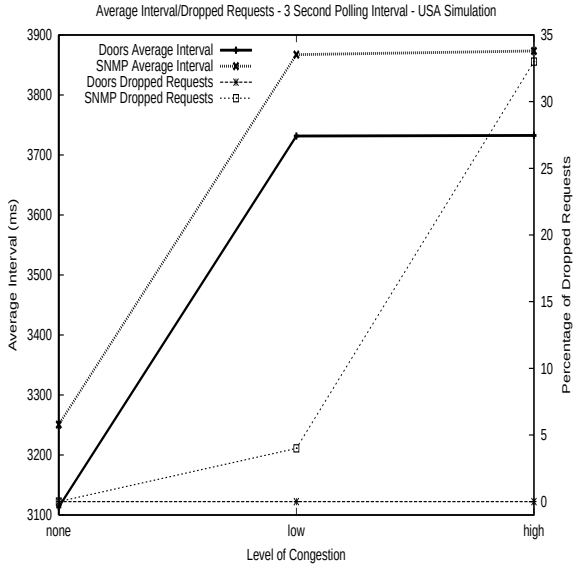


Figure 2.13: Average Interval / Packet Drops in USA simulation

CHAPTER 3
Network Simulation using GENESIS

3.1 Introduction

In simulating large-scale networks at the packet level, a major difficulty is the enormous computational power needed to execute all events that packets undergo in the network [31]. Conventional simulation techniques require tight synchronization for each individual event that crosses the processor boundary [29]. The inherent characteristics of network simulations are the small granularity of events (individual packet transitions in a network) and high frequency of events that cross the boundaries of parallel simulations. These two factors severely limit parallel efficiency of the network simulation execution under the traditional protocols [29].

Another difficulty in network simulation is the large memory size required by large-scale network simulations. With the emerging requirements of simulating larger and more complicated networks, the memory size becomes a bottleneck. When network configuration and routing information is centralized in a network simulation, large memory is needed to construct the simulated network. Moreover, memory size used by the simulation increases also with the intensity of traffic loads that impact the average size of the future event list. Although memory requirements can be tampered by the good design and implementation of the simulation software [32], we believe that to simulate truly large networks, the comprehensive, distributed memory approach needs to be developed.

3.2 Typical Memory usage in network simulation

To have an idea of a typical usage of memory by the popular SSFNet simulator [34], which is used by the networking research community, we ran a few tests with different topologies with the number of Autonomous Systems ranging from 15 to 35 and different amount of traffic flowing in the network. The result is as shown in the tables 3.1 and 3.2 and Figure 3.1.

The topology being used is a modified version of the topology at the SSFNet site [34]. To see the effect of congestion in the network, the number of background

No. of AS's	Background traffic hosts	Memory Used(MB)
15	0	275
15	12	452
15	24	433
20	0	540
20	12	519
20	24	519
25	0	770
25	12	746
25	24	747
30	0	774
30	12	786
30	24	782
35	0	1139
35	12	1148
35	24	1345

Table 3.1: Memory Usage in SSFNet vs. Topology(Frequency interval=0.0005 sec)

No. of AS's	Background traffic hosts	Memory Used(MB)
15	0	275
15	12	732
15	24	1120
20	0	540
20	12	827
20	24	1078
25	0	770
25	12	1157
25	24	1165
30	0	774
30	12	1219
30	24	1219
35	0	1139
35	12	1148
35	24	1345

Table 3.2: Memory Usage in SSFNet vs. Topology(Frequency interval=0.0001 sec)

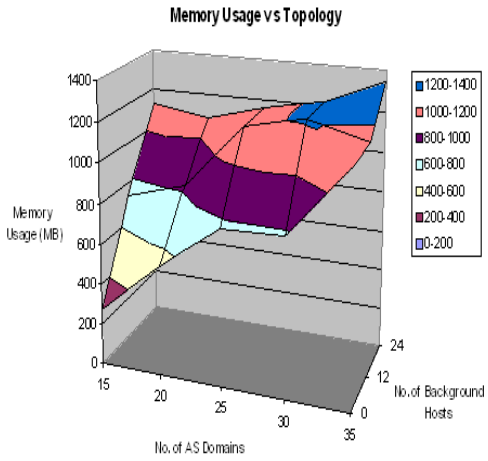


Figure 3.1: Memory Usage in the SSFNet Simulator

hosts in each topology is also changed where each background host represents a UDP session where a file of size 1290000000 bytes is being exchanged by the server and client in packet size of 300 bytes each. Also, the frequency of the sending rate of the server is modified to vary the congestion in the system. As show in the figure, the memory required to perform this simulation is huge and requires a computer with a large amount of primary memory to efficiently simaulte networks to the scale of the Internet.

3.3 Concept and Idea of GENESIS

In this chapter we discuss GENESIS which is a novel approach to both scalability and efficiency of parallel network simulation [35, 36]. Genesis combines simulation and modeling in a single execution. It decomposes a network into parts, called domains, and simulation time into intervals, and simulates each network partition independently and concurrently over one interval. After each time interval, flow delays and packet drop rates observed by each domain simulator are exchanged with others. Each domain simulator will model the traffic external to its own domain based on the flow data received from other domains. *Link proxies* are used in Genesis to represent the external traffic paths. Domain simulators iterate over the same time interval until they reach a global

convergence with controllable precision. An execution scheme is shown in Figure 3.2 that illustrates also synchronization between the repeated iterations over the same time interval and use of checkpoints for the domain simulators [36].

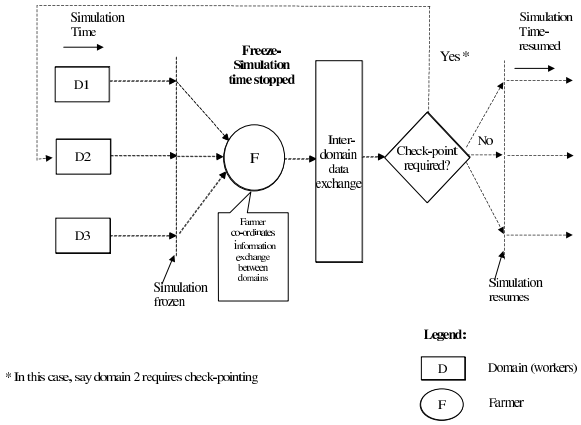


Figure 3.2: Simulation Execution Scheme in Genesis

Genesis achieved performance improvement thanks to simulating smaller number of events and its infrequent, high-granularity synchronization mechanism. No individual packet was synchronized between two parallel simulations; instead, packets were “summarized” on some metrics (delay, drop rate, etc.) and only these data were exchanged between domains at the end of each time interval. This approach was designed to simulate TCP and UDP data traffics, but could not be used to simulate some other flows, for example, data flows providing information for routing protocols. This is because the traffic of a routing protocol cannot be summarized; instead, different content and timing of each routing packet might change the network status. Particularly, our desire to simulate BGP protocol required us to develop new synchronization mechanism in Genesis.

Many parallel simulation systems achieved speed-up in simulation time, however, they also required that every machine involved had big enough memory to hold the full network, the requirement most easily achievable through the systems with shared memory. In Genesis, we have fully distributed the memory usage and we describe in this paper, the new version that overcomes the memory size limitation.

3.4 Simulation with Distributed Memory and Proxy Links

Simulations of large-scale networks require large memory size. This requirement can become a bottleneck of scalability when the size or the complexity of the network increases. For example, ns2 uses centralized memory during simulation, which makes it susceptible to the memory size limitation. The scalability of different network simulators was studied in [32]. This paper reports that in a simulation of a network of a dumbbell topology with large number of connections, ns2 failed to simulate more than 10000 connections. The failure was caused by ns2's attempt to use virtual memory when swapping was turned off. This particular problem can be solved by using machines with larger dedicated or shared memory. Yet, we believe that the only permanent solution to the simulation memory bottleneck is to develop the distributed memory approach.

In the version reported in [35], Genesis did not distribute network information among domain simulators. Each of them had to construct the full network and to store all the dynamic information (e.g., routing information) for the whole network during the simulation. To avoid such replication of memory, we developed a new version of Genesis which completely distributes network information. Thanks to this solution, Genesis is able to simulate large networks using a cluster of computers with smaller dedicated memory (compared to the memory size required by SSFNet simulating the same network).

Memory distribution is particularly challenging in Genesis, because of its special high granularity synchronization approach. In Genesis, within one time interval, one domain simulator is working independently of others, simulating the partial traffics flowing within or through that domain. Other parts of these traffics, which are outside of that domain, are simulated by *proxy links* which compute the packet delays and losses based on flow "summaries" provided by the outside domain simulators [35]. If the network information is completely distributed among the domain simulators, each one has information about only a part of the network. Hence, these simulators cannot simulate global traffics independently because information about a flow source or its destination, or both will not be there. We should notice the difference here from other event-level synchronization systems. In those systems, to simulate distributed network, each individual event crossing the boundary is forwarded to remote simulators regardless of its "semantic meaning", and the parallel simulators do not need to simulate global flows independently.

As a solution, in each domain we introduced traffic proxies that work on behalf of their counterparts in the remote domains. Traffic proxies send or receive TCP or UDP data packets as well as acknowledgment packets according to the produced feedbacks. To simulate inter-domain flows, partial flows are constructed between local hosts and *proxy hosts*. Thus, in the simulation of one AS domain, the simulator just simulates one part of an inter-domain traffic by using *proxy hosts* and *proxy links*, as shown in Figure 3.3.

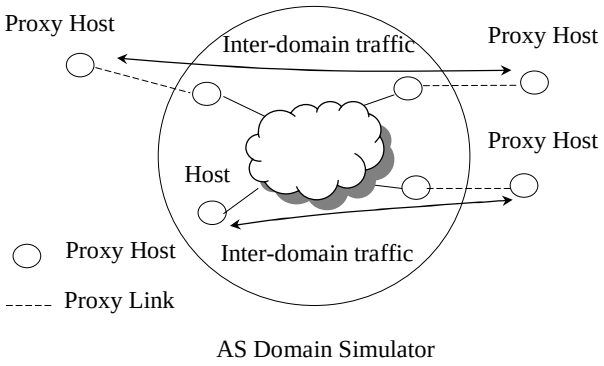


Figure 3.3: Proxy Hosts and Inter-domain Traffic

The actual traffic path between local hosts and remote hosts must be decided by inter-AS routing. For example, inter-AS routing changes can cause remote inbound traffic to enter the current AS domain from different entry points, thus routing the flow through a different path inside the domain. We developed a method, described below, to construct these remote traffic paths and to automatically adjust them to reflect the current inter-AS routing decision.

3.5 Performance Evaluation

To test the performance and scalability of the Genesis extension described here and to compare them to those of SSFNet, we use a modified version of the baseline model defined by the DARPA NMS community [33]. The topology for the model that we are using can be visualized as a ring of nodes, where each node (representing an AS domain) is connected to one node preceding it and another one succeeding it. We refer to each node or AS domain as the “campus network”, as shown in Figure 3.4. Each of the campus networks is similar to the others and consists of four subnetworks. In addition, there are two additional routers not contained in the subnetwork, as shown in the diagram.

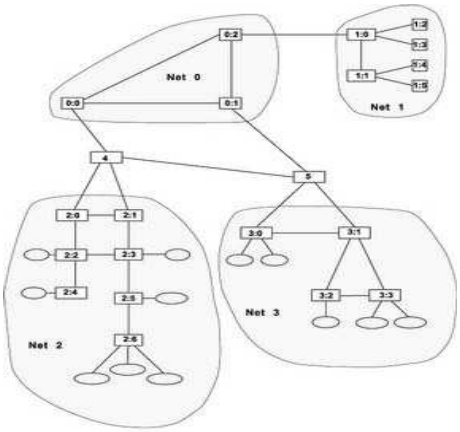


Figure 3.4: One campus network

The subnetwork labeled Net 0 consists of three routers in a ring, connected by links with 5ms delay and 2Gbps bandwidth. Router 0 in this subnetwork acts as a BGP border router and connects to other campus networks. Subnetwork 1 consists of 4 UDP servers. Subnetwork 2 contains seven routers with links to the LAN networks as shown in the diagram. Each of the LAN networks has one router and four different LAN's consisting of 42 hosts. The first three LAN's have 10 hosts each and the fourth LAN has 12 hosts. Each of the hosts is configured to run as a UDP Client. Subnetwork 3 is similar to Subnetwork 2. Internal links and LAN's have the same property as Subnetwork 2.

The traffic that is being exchanged in the model is generated by all the clients in one domain choosing a server randomly from the Subnetwork 1 in the domain that is a successor to the current one in the ring. We used different send-intervals of 0.1, 0.05 and 0.02 second to vary the traffic intensities, and used different numbers of nodes (AS domains) to vary the size of the network. Each simulation was run for 400 seconds of the simulated time.

All tests were run on up to 30 processors on Sun 10 Ultrasparc workstations, which were interconnected by a 100Mbit Ethernet. In the simulations under distributed Genesis, the number of processors used was equal to the number of campus networks.

Genesis distributively constructs and simulates BGP routers in AS domain simulators. To measure scalability of this solution in terms of network size, we simulated BGP networks of 10, 15, 20 and 30 AS domains, each run by a Sun 10 Ultrasparc workstation with 256Mb of memory. As shown in Figure 3.5, when the number of AS domains

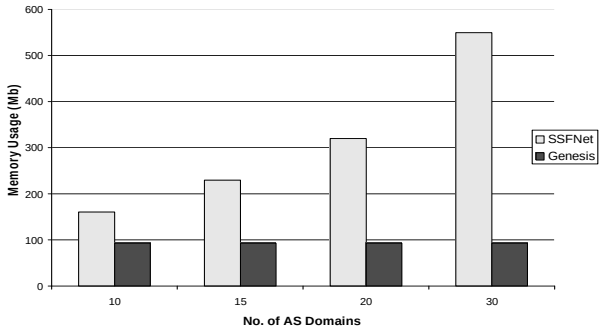


Figure 3.5: Memory usage of SSFNet and Genesis for simulating different sizes of BGP networks. Memory sizes of Genesis shown above are the requirements for single AS simulator

increases, unlike SSFNet, the memory usage of one Genesis AS simulator does not increase. As a result, by utilizing more computers with smaller memories, we are able to simulate much larger networks.

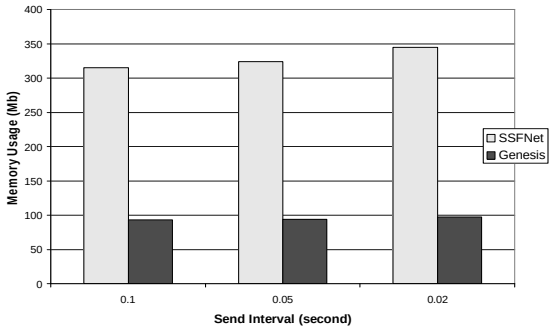


Figure 3.6: Memory usage of SSFNet and Genesis for 20-AS BGP network simulations with different send-rates

Memory usage of simulation is related not only to the static network size, but also to the network dynamics. We increased the traffic intensity by reducing the traffic send-interval from 0.1 to 0.05 and 0.02 second. As shown in Figure 3.6, although we did not observe very big changes in memory usage in SSFNet on this campus network model, Genesis showed even smaller increase in memory size with the same changes in traffic (thanks to its smaller base memory size).

As we have shown in [35, 36], Genesis achieved execution speedups thanks to its high granularity synchronization mechanism. In the described new version of Genesis, despite the extra overheads introduced by distributing the network, good speedups where

achieved for 10, 15, 20 and 30 domain simulators with BGP routers. The Genesis domains were defined by the AS boundaries. Figure 3.7 shows the speedups of simulations for these networks.

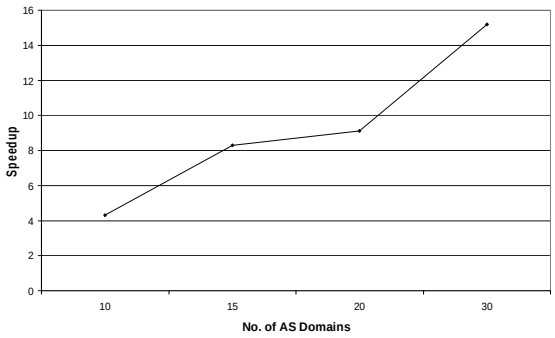


Figure 3.7: Speedup achieved for simulations of different BGP network sizes

Figure 3.8 shows that Genesis achieved higher speedups with higher traffic intensities. This is because with higher traffic intensity, more events need to be simulated in a fixed simulation time. Theoretical analysis tells us that sequential simulation time includes terms of order $O(n * \log(n))$, due to sorting event queues. Genesis distributes the simulation among domain simulators, which reduces the number of events needed to be simulated by one simulator, so it can achieve higher speedups when the traffic increases as well as when the network size increases.

To measure the accuracy of the simulation runs, we monitored the per flow end-to-end packet delays and packet drop rates. We compared the results from distributed Genesis with the results from sequential simulations under SSFNet, and calculated the relative errors. Our results showed that for most of the flows, the relative errors of both packet delay and drop rate were within the range from 2% to 10%, while a small number of individual flows had higher relative errors up to 15% to 20%. Considering the fact that in a simulation with large number of flows, the network condition was mainly determined by the aggregated effects of sets of flows, we calculated the root-mean-square of the relative errors on each set of flows which went through the same domain. These root-mean-squares of relative errors were below 5%, which seems sufficiently close approximation of the sequential simulation for many applications.

Simulation results showed that by fully distributing the simulation in Genesis, we gained the scalability of memory size. In addition, the parallel simulation in Genesis still

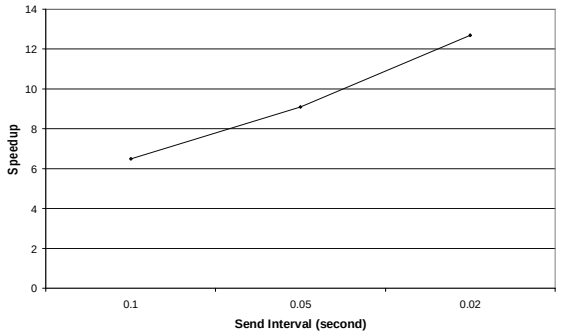


Figure 3.8: Speedup achieved for 20-AS BGP network simulations with different send-rates

achieved performance improvement in this distributed framework, compared to sequential simulations.

3.6 Contributions

I started this project with background research in this area to understand the basics of Network Simulation and how Genesis is different from the traditional simulators. I next tried to understand the memory usage of the SSFNet simulator and how memory is dependent on the traffic network topology. I ran many simulations with different topologies and traffic flows and generated graphs depicting the usage of memory with the traffic description.

I also modified the SSFNet simulator to add the transient traffic functionality in Genesis. For this I was responsible for re-engineering the SSFNet simulator and modifying the data strucutures reponsible for the definition of a traffic node. This required a thorough understandng of the flow of the simulator and how the functions interact.

I was also responsible for writing a program for the breaking up of the NMS scenario topology in such a way that it can be used by Genesis. Thus, one DML topology of the NMS scenario was broken into 20 DML files complete with the IP addresses and flow descriptions and was then used with Genesis to run the simulation. This was then used to show the improvements in memory between SSFNet and Genesis. I am currently working with another student to make an automated program to generate these scripts such that if the given input is a SSFNet DML then the output DML would be Genesis DML files.

My current work is focussed on developing a hierarchial routing protocol, which

is a modified version of the AODV protocol and will be used in the NMS Demo. This protocol is such that it breaks the networks into different hierarchy zones such that one hierarchy can talk to only nodes within the same hierarchy.

CHAPTER 4
Discussions and Conclusions

In this thesis we have focussed on efficient techniques for Network Management and Network Simulation. First, we studied the DOORS system and saw how a network management middleware like DOORS performs much better than traditional SNMP. We performed the tests in both a laboratory as well as the SSFNet simulator to show the improvement in DOORS. Next, we discussed the network simulator - Genesis and how the distributed version uses much less memory than traditional simulators. In this chapter we discuss the future work and conclusions in both these areas.

4.1 DOORS: Conclusion and Future Work

Comparison of DOORS and SNMP leads to the following conclusions. In a single-client scenario, we see a cost benefit of running DOORS for monitoring network data as compared to traditional SNMP polling methods. In a multi-client scenario, DOORS outperforms standard polling methods and this difference grows linearly as a function of the number of clients polling for similar data. DOORS achieves this advantage thanks to the consolidation of multiple client requests into a single aggregated request. DOORS uses TCP connections which make the data transfer inherently reliable while SNMP polling risks dropped requests by using UDP. The added functionality of TCP, comes at the cost of extra bandwidth in the form of added transport layer headers. However DOORS design counteracts this cost by removing the need for the request message used in conventional network polling as well as preprocessing at the polling station. Using DOORS, the client will get its data faster, but may have small deviations in the difference between polls, whereas normal SNMP clients will get the data later than the doors clients, but at a more firm inter-polling interval.

DOORS has been proven useful in cases in which normal SNMP polling is not feasible, and the management application has no control over the networks in-route to the managed networks. [8] DOORS can be extremely effective when encoded with functionality beyond just the simple collection and return of data. When some or all of the

algorithm from the client is placed into the agent, we can see large savings on bandwidth and speed of calculation.

We have shown in two distinct scenarios that agent based network monitoring can achieve great benefits at little costs. Distributing network management applications is a necessity as managed networks continue to grow in size and complexity. Consequently, the effectiveness of the network monitoring and its non-interfering with the network traffic is becoming more and more paramount. We believe that a mobile agent approach is the right solution for providing a middleware for administrators interested in proactive network management with flexible functionality that can be assigned to agents.

It has been noted in the network security field that the growth of switched and other highly segmented networks has posed a significant problem for current intrusion detection methods which use sniffed data for detection [22]. To address this problem, and facilitate advanced, efficient data collection for intrusion detection processes, we are currently creating intrusion detection agents to perform detection functions for clients.

We are still encoding more of the client's functionality into our agents. Therefore, detection and collaboration can happen at the platform on which the agent is located, potentially making communication back across the network to the client a rare necessity. Some of this has already been implemented (see Section 2.3.4). As more of these functionalities are given to the agent, we can use collaboration between agents to provide distributed detection based on decisions made by multiple agents using their own collected data.

4.2 Genesis: Conclusion and Future Work

In this thesis, we demonstrated that the described here new Genesis version can work efficiently with fully distributed network memory. This design reduces and makes scalable the memory size requirement for large-scale network simulations. As a result, Genesis is able to simulate huge networks using limited computer resources. One direction of the future work in this ares is to apply Genesis to more real-world applications, to construct more practical network models and to simulate and analyze them in Genesis.

With the new support of BGP simulation in Genesis, study of the performance and stability of BGP can be another direction of future research. The memory size required

by BGP network simulation increases very fast when the number of BGP routers and AS domains increases. As a result, the simulation of large BGP networks was hindered by the memory size limitation. Genesis offers a new approach to simulating BGP on distributed memory that is scalable both in terms of simulation time and the required memory.

LITERATURE CITED

[1] M. Baldi, S. Gai, and G. P. Picco, "Exploiting code mobility in decentralized and flexible network management," in *Mobile Agents*, Berlin, Germany, April 7-8 1997, First International Workshop, MA, pp. 13–26, Springer Verlag.

[2] André Mello Barotto, Andriano de Souza, and Carlos Becker Westphall, "Distributed network management using SNMP, Java, WWW and CORBA," *Journal of Network and Systems Management*, vol. 8, no. 4, pp. 483–497, 2000.

[3] Forest Baskett, K. Mani Chandy, Richard R. Muntz, and Fernando G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *Journal of the Association for Computing Machinery*, 22(2):248–260, April 1975.

[4] M. A. Bauer, R. B. Bunt, A. El Rayess, P. J. Finnigan, T. Kunz, H. L. Lutfiyya, A. D. Marshall, P. Martin, G. M. Oster, W. Powley, J. Rolia, D. Taylor, and M. Woodside, "Services supporting management of distributed applications and systems," *IBM Systems Journal*, vol. 36, no. 4, pp. 508–526, 1997.

[5] J. Alan Bivens, Li Gao, Mark Hulber, and Boleslaw Szymanski, "Agent-based network monitoring," in *Agent based High Performance Computing*, Seattle, Washington, May 1999, Proc. Autonomous Agents99, Seattle, WA, pp. 41-53.

[6] Alan Bivens, Patrick H. Fry, Li Gao, Mark F. Hulber and Boleslaw K. Szymanski. "Distributed Object-Oriented Repositories for Network Management," August 1999, Proc. 13th Int. Conference on System Engineering, pp. CS169-174, Las Vegas, NV.

[7] Alan Bivens, Rashim Gupta, Ingo Mclean, Jerome White and Boleslaw K. Szymanski. "Scalability and Performance of an Agent-based Network Management Middleware," submitted to International Journal of Network Management.

[8] J. Alan Bivens. "Distributed Framework for Deploying Machine Learning in Network Management and Security," PhD Thesis, submitted to Rensselaer Polytechnic Institute, December 2002.

[9] J.D. Case, K. McCloghrie, M. Rose, and S. Waldbusser. Structure of management information for version 2 of the simple network management protocol. RFC 1902, January 1996.

[10] CCITT, "Specification of abstract syntax notation one (asn.1)," CCITT Recommendation X.208, 1988.

[11] CCITT, "Specification of basic encoding rules for abstract syntax notation one (asn.1)," Recommendation X.209, 1988.

[12] James Cowie, David M. Nicol, and Andy T. Ogielski, "Modeling the global internet," *Computing in Science & Engineering*, vol. 1, no. 1, pp. 42–50, January/February 1999.

[13] Anja Feldmann, Anna C. Gilbert, Polly Huang, and Walter Willinger, "Dynamics of IP traffic: A study of the role of variability and the impact of control," in *SIGCOMM*, 1999, pp. 301–313.

[14] Alfonso Fuggetta, Gian Pietro Picco, and Giovanni Vigna, "Understanding Code Mobility," *IEEE Transactions on Software Engineering*, vol. 24, no. 5, pp. 342–361, May 1998.

[15] Neil J. Gunther. *The Practical Performance Analyst: performance-by-design techniques for distributed systems*. McGraw-Hill Series on Computer Communications. McGraw-Hill, New York, NY, 1998.

[16] J. R. Harista, M. O. Ball, N. Roussopoulos, A. Datta, and J. S. Baras, "MANDATE: MAnaging Networks using DAtabase TEchnology," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 9, pp. 1360–1372, Dec. 1993.

[17] Christian Huitema, *Routing in the Internet*, Prentice Hall PTR, Upper Saddle River, NJ 07458, second edition, 2000.

[18] Y. Joo, V. Ribeiro, A. Feldmann, A.C. Gilbert, and W.Willinger, "TCP/IP traffic dynamics and network performance: A lesson in workload modeling, flow control, and trace-driven simulations," *SIGCOMM Computer Communications Review*, vol. 31, no. 2, April 2001.

[19] Leonard Kleinrock. *Queuing Systems: Volume II: Computer Applications*, volume II. John Wiley & Sons, Inc, New York, NY, 1976.

[20] Edward D. Lazowska, John Zahorjan, G. Scott Graham, and Kenneth C. Sevcik. *Quantitative System Performance: Computer System Analysis Using Queuing Network Models*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1984.

[21] G. Mansfield, E. P. Duarte Jr., M. Kitahashi, and S. Noguchi, "Vines: Distributed algorithms for a web-based distributed network management system," in *Worldwide Computing and Its Applications*, Tsukuba, Japan, March 10-11 1997, International Conference, WWCA, pp. 281–293, Springer Verlag.

[22] Stuart McClure and Joel Scambray, "Once-promising intrusion detection systems stumble over switched networks," in *InfoWorld*, vol. 22, pp. 58–58. InfoWorld Media Group, Inc., December 2000.

[23] William Stallings, *SNMP, SNMPv2, SNMPv3, and RMON1 and 2*, Addison Wesley Longman, Inc., Reading, Massachusetts, 3rd edition, 1999.

[24] W. Richard Stevens, *TCP/IP Illustrated*, vol. 3, Addison-Wesley Publishing Company, One Jacob Way; Reading, Massachusetts 01867, 1996.

[25] W. Richard Stevens, *TCP/IP Illustrated*, vol. 1, Addison-Wesley Publishing Company, One Jacob Way; Reading, Massachusetts 01867, 1994.

[26] Marina Thottan and Chuanyi Ji, "Proactive anomaly detection using distributed intelligent agents," *IEEE Network, Special Issue on Network Management*, vol. 12, no. 5, pp. 21 –27, Sept-Oct 1998.

[27] Marina Thottan and Chuanyi Ji, "Adaptive thresholding for proactive network problem detection," in *IEEE International Workshop on Systems Management*, Newport, Rhode Island, Apr. 1998, IEEE, pp. 108–116.

[28] S. Waldbusser. Remote network monitoring management information base. Request for Comments: 1757, February 1995.

[29] Bhatt, S., R. Fujimoto, A. Ogielski, and K. Perumalla, Parallel Simulation Techniques for Large-Scale Networks. *IEEE Communications Magazine,* 1998.

[30] Fujimoto, R.M. Parallel discrete event simulation. *Comm. of the ACM*, 33:31–53, Oct. 1990.

[31] Law, L.A., and M. G. McComas. Simulation software for communication networks: the state of the art. *IEEE Comm. Magazine*, 32:44–50, 1994.

[32] Nicol, D. Comparison of network simulators revisited. *Available at http://www.ssfnet.org/Exchange/gallery/dumbbell/ dumbbell-performance-May02.pdf*, May 2002.

[33] NMS (*Network Modeling and Simulation DARPA Program*) baseline model. *See web site at* http://www.cs.dartmouth.edu/ nicol/NMS/baseline/.

[34] SSFNet(*Scalable Simulation Framework Network Models*). *See web site at* http://www.ssfnet.org/homePage.html.

[35] Szymanski, B., A. Saifee, A. Sastry, Y. Liu and K. Madnani. Genesis: A system for large-scale parallel network simulation. *Proc. 16th Workshop on Parallel and Distributed Simulation*, pages 89–96, May 2002.

[36] Szymanski, B., Y. Liu, A. Sastry, and K. Madnani. Real-time on-line network simulation. *Proc. 5th IEEE Int. Workshop on Distributed Simulation and Real-Time Applications, DS-RT 2001,* August 13–15, 2001, IEEE Computer Society Press, Los Alamitos, CA, 2001, pages 22–29.

[37] Szymanski, B., Y. Liu, and Rashim Gupta. Parallel Network Simulation under Distributed Genesis *To appear in the Proc. 17th Workshop on Parallel and Distributed Simulation*, 2003

[38] Szymanski, B., Q. Gu, and Y. Liu. Time-network partitioning for large-scale parallel network simulation under ssfnet. *Proc. Applied Telecommunication*

Symposium, ATS2002, B. Bodnar (edt), San Diego, CA, April 14–17, SCS Press, pages 90–95, 2002.

[39] Ye, T., D. Harrison, B. Mo, S. Kalyanaraman, B. Szymanski, K. Vastola, B. Sikdar, and H. Kaur. Traffic management and network control using collaborative on-line simulation. *Proc. International Conference on Communication, ICC2001*, 2001.

[40] U. Warrier, L. Besaw, L. LaBarre, and B. Handspicker. The common management information services and protocols for the internet (cmot and cmip). Request for Comments: 1189, October 1990.