The Genesis Project: Network Decomposition in Monitoring and Simulation for Network Management and Intrusion Detection

Boleslaw K. Szymanski, Alan Bivens, Yu Liu, Kiran Madnani, Anand Sastry
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY USA 12180-3590

Abstract— **Genesis proposes a novel approach to scalability and efficiency of parallel network monitoring, modeling and simulation for network management and intrusion detection. The basis of our approach is network decomposition that creates separate network domains. Each domain is independently monitored, modeled and simulated by separate software components. Domain monitoring involves a repository of data and collection agents while models and simulations are run by dedicated processor clusters allocated to each domain. Repositories and simulators collaborate on exchanging data and models to keep the network simulations consistent between domains.**

One of the desirable features of our design is its independence from the underlying simulators and repositories running in the individual domains. Hence, our architecture can be integrated with a number of existing technologies thereby supporting system interoperability.

Our primary application is network management. The simulation in this application predicts changes in the network performance caused by network parameter tuning. Growing security threats to networks increase the importance of intrusion and anomaly detection. Using data collection agents, we create unique signatures for attacks or legitimate user activities (monitoring a stream of network packets at the server). Signatures are represented as probabilistic or time-dependent finite state automata and their parsers can easily be embedded into mobile agents.

Genesis is useful in all applications in which 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 for computer networks. In addition to computer and communication networks, Genesis can also be used to simulate networks of flows of goods and products, vehicles, populations, etc.

Keywords— **Network Monitoring, Network Management, Network Simulation, Parallel Simulation, Intrusion Detection.**

I. Introduction

The major difficulty in simulating large networks at the packet level is the enormous computational power needed to execute all events that packets undergo in the network [1]. Packets crossing the boundaries of parallel partitions impose tight synchronization between parallel processors, thereby lowering parallel efficiency of the execution [2]. For mobile networks, nodes crossing the parallel partition boundaries during the simulation introduce additional challenge.

In this paper, we describe a novel approach to scalability and efficiency of parallel network simulation and monitoring based on network decomposition. The system, called Genesis, can integrate different network simulators as components of the same large-scale simulation supporting interoperability and scalability. The described method can be seen as a variant and modification of a general scheme for optimistic simulation referred to as Time-Space Mappings [3]. Although all optimistic simulations can be viewed as variants of this scheme, very few apply, as we do, iterations over the same time interval to find a solution.

Our approach partitions the network into parts, called domains. The simulation time is partitioned into disjoint intervals. Each domain is simulated independently of and concurrently with the others over the same simulation time interval. At the end of each interval, metrics of inter-domain flows are exchanged between domains. The domain simulators iterate over the same time interval until the exchanged metrics converge to constant values within the prescribed precision. After convergence, all domains simulators move simulation to the next simulation 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 [4].

This distributed simulation method is complemented by the similarly designed network monitoring system, called DOORS. The main result of this paper is to demonstrate that by judicious design of the domains and information exchange, the proposed approach can efficiently parallelize network simulations and inter-operate with data collection system.

II. Parallelism, Interoperability and Synchronization of Genesis Simulators

Although our approach has been described earlier as applied to ns [5] and ssfnet [6] simulators and TCP traf-

This work was partially supported by the DARPA Contract F30602-00-2-0537 and by the grant from the University Research Program of CISCO Systems Inc. The content of this paper does not necessarily reflect the position or policy of the U.S. Government or CISCO Systems—no official endorsement should be inferred or implied.

fic [7], we provide a brief summary here, to make the paper self-contained. The system enables integration of different simulators into a coherent network simulation, hence we called it *GE*neral *NE*twork *S*imulation *I*ntegration *S*ystem, or *Genesis* in short.

In Genesis, a network domain consists of a subset of network sources, destinations, routers and links that connect them. Each domain is simulated concurrently with other domains and repeatedly iterates over the same simulation time interval, exchanging information with other domains after each iteration. In the initial iteration, each domain assumes either zero traffic flowing into it from outside (when the entire simulation or a particular flow starts in this time interval) or the external traffic characterization from the previous time interval. External traffic into the domain for all other iteration steps is defined by the activities of the external traffic sources and flow delays and packet drop rates computed from the data received from the other domains in the previous iteration step.

Each domain simulator generates by itself all flows whose sources are within this domain, but needs to approximate flows with the sources that are external to its domain. This approximation is achieved as follows. In addition to the nodes that belong to the domain by the user definition, Genesis creates also *domain closure* that includes all the sources of flows that reach or pass through this domain. Since those are copies of nodes active in other domains, we call them *source proxies*. Each source proxy uses the flow definition from the simulation configuration file and the native traffic generator.

The flow delay and the packet drop rate experienced by the flows outside the domain are approximated by the random delay and probabilistic loss applied to each packet traversing in-link proxies. These values are generated according to the average packet delay and its variance as well as the observed packet loss frequency communicated to the simulator by its peers at the end of simulation of each time interval. Each simulator collects this data for all of its own out-link proxies when packets reach the destination proxy.

Each delay at the router is the sum of constant processing, transmission and propagation delays and a variable queuing delay. If the total delay across all external routers is relatively constant in the selected time interval, the actual delay can be approximated by randomly generated delay from the distribution with the same average value and variance as observed in the other domains. Likewise, packet loss can be applied randomly with the probability defined by the observed frequency of the actual packet loss on the external path. These three values (average packet delay and its variance and the frequency of packet drop) are sent to the source proxy to be used in generating the flow. Thanks to the aggregated effect of many flows on queue sizes, this delay changes more slowly than the traffic itself, making such model precise enough for our applications.

Our experience indicates that communication networks simulated by Genesis will converge thanks to monotonicity of the path delay and packet drop probabilities as a function of the traffic intensity (congestion).

The efficiency of our approach is greatly helped by the non-linearity of the sequential network simulation. It is easy to notice that the sequential simulation time grows faster than linearly with the size of the network. Some of our measurements [4] taken over the hierarchical networks indicate that the dominant term is of order $O(n^2)$ even for small networks. The impact of this observation on superlinear speedup of Genesis simulation is shown in Figure 1.

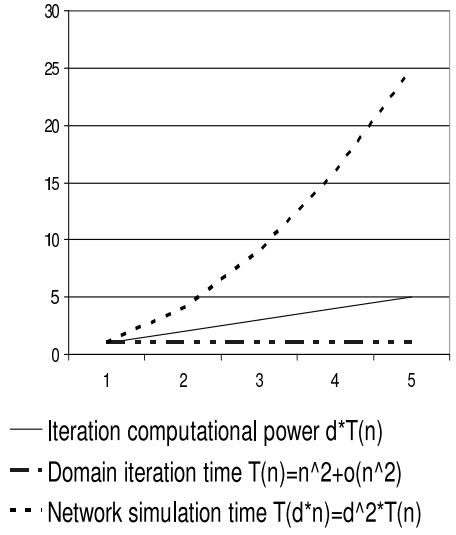

Fig. 1. Analysis of Superlinear Speedup: with d domains of size n superlinear speedup will be achieved if the number of iterations $i < d$.

We conclude that it is possible to speed up the sequential network simulation more than linearly by splitting it into smaller networks and parallelizing the execution of the smaller networks. With modest number of iterations the total execution time can be decreased by an order of magnitude or more. Example of the superlinear speedup for 4 and 16 domain simulations of mixed TCP and UDP traffic is shown in Figure 2.

The extensions of a wired network simulator needed for Genesis interface are quite standard and include the following components.

Domain Definition: that must be introduced by the user in the native simulation configuration file. It simply enumerates nodes belonging to each domain or labels each nodes with the domain identifier.

Source and Link Proxies: that are introduced by Genesis based on the domain definition. Proxies belong to the domain closure. Source proxies represent sources that produce flows crossing or directed into the domain. Link proxies approximate packet delays and packet drops on the path from the source to the domain boundary.

Data Collector: that is added to each flow leaving a domain. It collects data about the packet delays and packet drop rates from the flow source to the domain boundary.

Checkpointing and Freeze Event: that enable domain simulation synchronization. Freeze event causes all the simulators to stop at the same simulation time and it is added

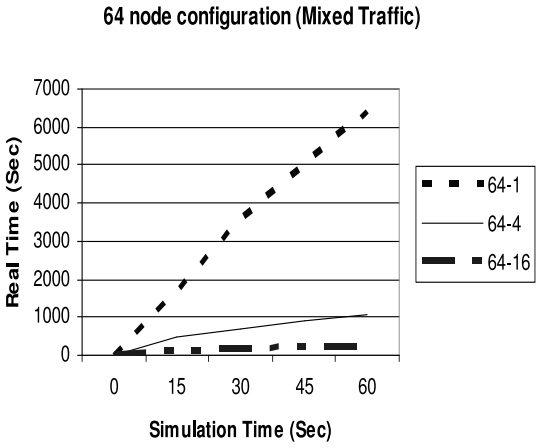

Fig. 2. Simulation Times for 2,000 UDP and 1,000 TCP Flows in the 64-router Network Split into 4 and 16 Domains.

to the future event list by the Genesis system (this is the only new event introduced by Genesis). Checkpointing uses fast, diskless and application independent fork-based memory copy [8] to create a copy of each simulator at the beginning of each simulated interval. At the end of simulated interval, all domains either reactivate a copy (re-simulating the same interval but with new data about external flow) or delete the copy and continue simulation into a new time interval.

III. Some of the Challenges in Integrating GloMoSim with Genesis

The rapid advancement in portable computing platforms and wireless communication technology has led to significant interest in the design and development of mobile networks [9]. However, the management and evaluation of such networks presents unique challenges. Some of these include the shared broadcast medium between thousands of nodes, the mobility of these nodes and the unpredictable nature of the wireless channel with problems such as fading, obstacles and interference. The design of the Genesis interface to mobile network simulator, GloMoSim [10] is described in this section.

As in the previous Genesis implementations, the network is decomposed into domains that contain network nodes, that, however, in this case can be mobile. Thus, a domain is defined by the geographical area that it covers. The user's domain definition is a Genesis specific part of the GloMoSim configuration file. It also contains the radio-range of a node and using these parameters, Genesis computes the *closure* of the domain defined as the conjunction of the domain proper and its boundary regions of the radio-range width. The closure enables the system to account for activities of nodes outside the domain that can interfere with the nodes inside it. Based on the domain closure, Genesis identifies the nodes active in each domain.

As in the Genesis interface to wired networks, domains are simulated concurrently with each other over the same time interval. The domains freeze [5] at user-specified intervals. At the time of freeze the inter-domain data exchange takes place. In GloMoSim, a node can schedule events (transmit and receive packets) while mobile. The current Genesis extension to GloMoSim accounts for the "mobility-trace" defined mobility in which the user specifies the speed, start and destination locations of the nodes in a configuration file. Knowing the above parameters, Genesis computes time and location at which the node crosses the domain boundaries even before the simulation starts. Using this information, each domain simulator knows when and where the mobile node will be active in its domain.

Introduction of domain closures creates regions in the network topology that overlap with at least two domains. Thus, a node in such a region is active in both domains at the same time. The Genesis domain simulators which simulate activities of such a node must include the same events for the node. To achieve this, the inter-domain messages include information about communication events (packets received and sent) by nodes present in the domain-closure. Each domain receiving this information checks if the same events were executed for its copy of the nodes in question. If not, the time interval is re-simulated with the modified list of events for the offending node.

Each domain has at most eight domains as neighbors. Thus, each domain needs to communicate information about the activity of domains lying in its closure to its corresponding neighbor only. We achieve this by establishing a peer-to-peer connection between domains. In other words, each domain receives data from at most eight of domains during the freeze event. Since all the domains must simulate the same time interval, we synchronize them by overlaying a simple farmer-worker architecture over all domains. Synchronization is based on messages sent to the farmer, which identifies the state of the simulation.

Currently, we are conducting experiments measuring efficiency of the Genesis based GloMoSim simulations.

IV. Data Collection by DOORS

Network decomposition is also a designing principle behind network monitoring system called DOORS (Distributed Online Object RepositorieS) [11]. The system was initially designed to support data collection under such protocols as CMIP and SNMP [12]. We have now extended it to enable support for distributed network problem detection, statistical network data analysis, and intrusion detection. All these application require that DOORS place a minimal footprint on the network itself. Our goal is to make DOORS scalable to the largest existing networks while facilitating a wide variety of specialized clients that simply cannot be managed using traditional, strictly hierarchical approaches.

A. Architecture

With DOORS, any client needing information about any node on the network posts requests for objects and data to

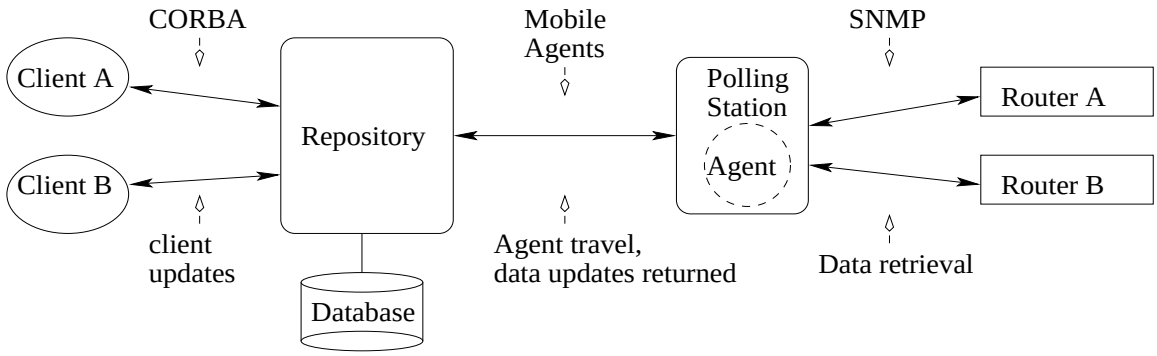

Fig. 3. DOORS Architecture

its local repository. The repository manages and responds to the client requests in a manner virtually transparent to the client. On receiving a client's request, the repository creates an agent encapsulating the request and sends the agent to a node closer to the target. The repository also keeps track of agent locations and activities. If an agent is already dispatched performing a similar function, the repository does not send another agent. Instead, it may either send an update to the existing agent to assume the additional functionality, or, if the request is subsumed by the posted one, the repository simply distributes the results to both clients.

A DOORS client may request various forms of direct network data, as well as any function $f(x_1^t, x_2^t, ..., x_n^t)$ of network data at time t, where an argument x_i^t denotes the data item collected at that time. Function f could represent a statistical manipulation of network monitoring data or an intrusion detection routine. Hence, the agent may execute some functions at the remote location (close to the target), enabling the DOORS system effectively move the computation closer to the data (sending only the answers back), instead of the traditional method of transporting all the data across the network to the client for computation. Moving computation closer to the data reduces the total bandwidth used by any tools which monitor large networks. This extra traffic may noticeably impact network performance, and make the results obtained for large networks useless. Applications such as network problem detection or prediction need large amounts of current data, so minimizing the data collection impact on network performance is of utmost importance for such applications.

To elaborate on the system architecture shown in Figure 3, we explain the steps that the DOORS system executes for simple SNMP queries. We compare this process to that of the standard SNMP clients to differentiate their effects on the network and on the clients. To show the scalability of the system, our example involves several clients requesting similar data. However we also show comparisons on a single client basis.

A.1 Standard SNMP Process

SNMP is a simple request-reply protocol, in which the client uses UDP to send requests for data to a server on the target. The target, in turn, uses UDP to reply to the client. The SNMP protocol itself adds several headers to the UDP and IP headers of the network packets that it sends.

A.2 DOORS SNMP Process

The DOORS process is more complicated than pure SNMP one, but offers many benefits. First, the DOORS clients send requests to their repository for SNMP data from a particular target. When the repository recognizes similar requests arriving, it combines them into one agent, and sends this agent to a node near the target, called a polling station. The SNMP polling is done from this nearby to the target machine. While polling, the agent sends only the results back to the client, saving the cost of the multiple requests going across network links. DOORS agents use TCP to send the data back to the repositories in contrast to SNMP's UDP transport protocol. This choice makes our data collection more reliable than SNMP's because UDP packets may be dropped with no warning or repairing action. Scalability is supported by repository collaboration; requests outside the domain of a repository are forwarded to an appropriate cooperating repository.

B. Experimental Results

As previously stated, we retrieve SNMP data from the routers targeted in client queries. Standard SNMP polls could be used to retrieve data instead of a more complex system such as the one that we have described. However, our slightly more complex system provides efficiency, reliability, and ease of data management while placing the smallest possible footprint on the network.

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 standard SNMP method. Our test-bed consists of six machines running FreeBSD and three Cisco 2500 series routers (topology shown in Figure 4). The test-bed is arranged so that two Ethernet networks are connected by the three routers. In all measurements, the target is the Ethernet interface of router r3.

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 bandwidth seen by router r2 (in the middle of the topology shown in Figure 4).

When the standard SNMP methods are used, there is simply a client running on n7 which polls the Ethernet interface of the target. Under the DOORS system, the client is on n7, the repository is on n8, polling station across the network on n10, and other components of our system are as shown in Figure 4.

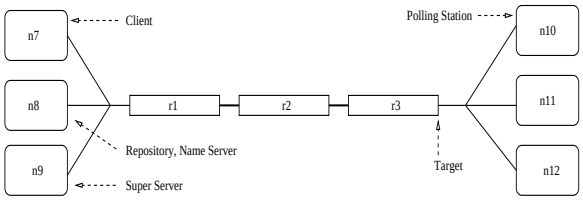

Fig. 4. Topology of the Test-bed for Experiments

We tested up to ten simultaneous clients polling at intervals from three to ten seconds. The results are shown in Figures (5-7).

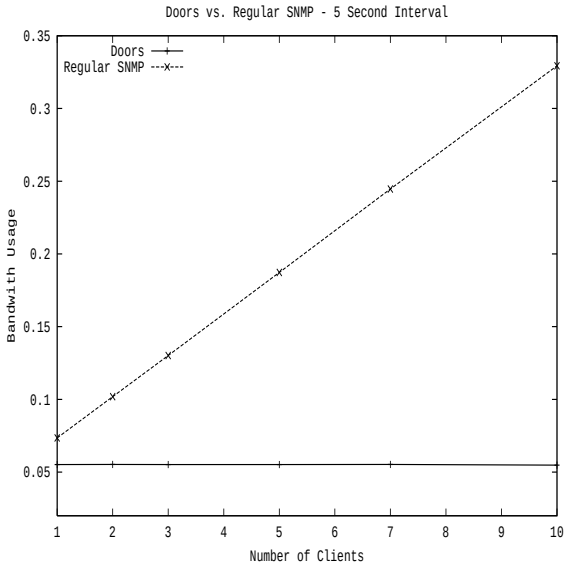

Fig. 5. Five-second Interval Bandwidth Usage Plot

The first graph shows the bandwidth usage incurred through data collection at five second intervals under both DOORS and the standard SNMP polling. The DOORS system achieves better performance regardless of the number of clients used. Moreover, the bandwidth used under DOORS remains effectively constant regardless of the number of clients, because in all cases only one message per the data collection interval is sent back to the repository to be distributed to any number of clients. However, the standard SNMP method must send a request across the network every polling interval for every client. Therefore, as the number of clients grows in data collection under the standard SNMP, so does the collection bandwidth used.

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 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 client inter-polling time. We use two standard deviation statistics, average-based standard deviation and actual-based standard deviation. The former 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 lower this deviation is the more consistent is 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. In our cases, the data integrity was perfect, meaning that all polled values were returned successfully. The graphs describing these statistics are in Figures (6 and 7).

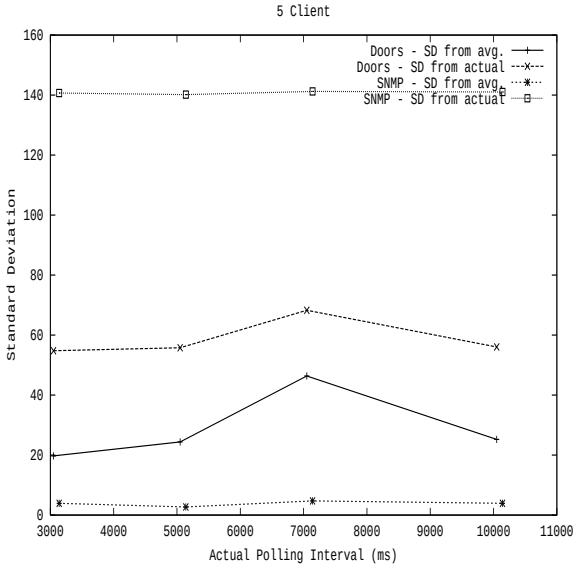

Fig. 6. Five-client Standard Deviation Plot

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

Our agents use TCP to send data back to the client, so there is 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, in its windowing mechanism. The collective use of these algorithms causes delays and oscillations in TCP delivery times compared to that of UDP [13]. On the other hand, the DOORS systems sends the configured agent close to the router, so the DOORS

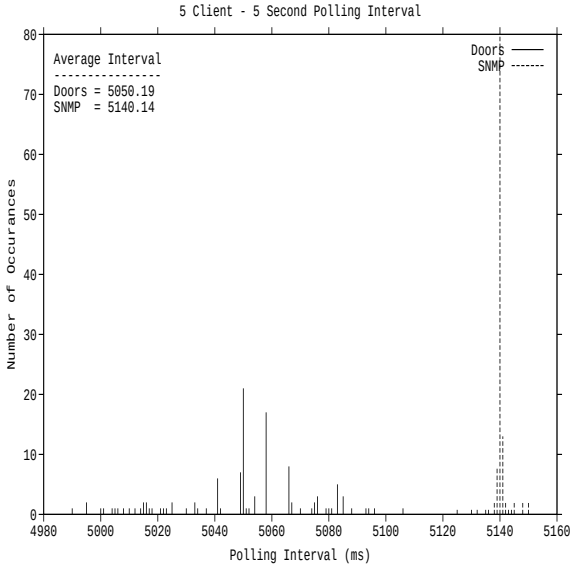

Fig. 7. Five-clients and five-second Interval Histogram Plot

system does not suffer the delays of traversing the network for the request portion of the request-reply paradigm that the normal SNMP client uses. This allows DOORS to have a fraction of the latency time that the normal SNMP client has, and thus reduces its actual-based standard deviation.

As seen in Figures 6 and 7, even though the standard deviation of DOORS about the average is much higher than that of the SNMP client, the average interval of the DOORS client is much lower than that of the SNMP client. This basically means that DOORS may be a bit more variable in the exact times at which it returns the data, but it almost always returns it faster than the SNMP client. The opposite is also true. The SNMP client almost always takes longer to return the data, but it returns it at a more consistent interval.

C. Analysis of Measurements

The most interesting results come from the single client tests. Even though the DOORS system uses TCP while the standard SNMP method uses the inexpensive UDP protocol, the DOORS client still uses less bandwidth than the SNMP client. SNMP needs both a request and a reply (a pull method) while DOORS simply sends the update data (a push method). The doors client also trims some of the extra SNMP protocol headers while sending only the response numerical values back to the client.

DOORS is an efficient and scalable solution to data collection. It can lighten the footprint of any network management application. When the number of clients needing similar data grows, the relative bandwidth used by DOORS remains constant while the standard SNMP method increases bandwidth usage at least linearly with the number of clients. As more functionality is placed in the agent, less data is required to go across the network, causing even greater savings in network resources.

V. Intrusion Detection

It is well-known that a signature of an attack can easily be represented in the form of a DFA and a simple but general parallel parser can then parse the incoming packets to see if any sequence of them matches the attacks encoded in the DFA. We extended this approach by using Time-Dependent Finite Automata. It is easy to include a parser for DFAs within the data collection agents and distribute them to the network nodes. Additionally, for computer system and networks, we investigate use of Probabilistic Finite Automata (PFA) that are capable of accepting all strings (a sequence of commands or a stream of network packets) produced by the legitimate user. PFA's transitions are labeled by the probabilities that such a transition has been taken in the given state by the legitimate user in one of the past sessions. Hence, for each on-line processed string, we can compute the current probability that this string represents activities of the legitimate user. Thus, in our approach user's signatures (Probabilistic Finite Automata) are developed and refined over time automatically.

VI. Conclusion and Future Research

The presented work is in progress and we continue making important extensions and additions. For example, we are currently working on integrating Genesis with the network management system [4]. The results obtained so far indicate that network decomposition is a powerful basis for a scalable and efficient network simulation, monitoring and security.

References

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

[2] R.M. Fujimoto, "Parallel discrete event simulation," *Communications of the ACM*, vol. 33, pp. 31–53, April 1990.

[3] K. M. Chandy and R. Sherman, "Space-time and simulation," in *Workshop on Distributed Simulation'89*. SCS, 1989, pp. 53–73.

[4] T. Ye, 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," in *International Conference on Communication, ICC2001*, Helsinki, Finland, June 2001.

[5] B. Szymanski, Y. Liu, A. Sastry, and K. Madnani, "Real-time on-line network simulation," in *5th International Workshop on Distributed Simulation and Real-Time Applications*, Cincinnati, Ohio, June 2001, IEEE, pp. 22–29.

[6] B. K. Szymanski, Q. Gu, and Y. Liu, "Time-network partitioning for large-scale parallel network simulation under ssfnet," in *Applied Telecommunication Symposium*. SCS, April 2002.

[7] B. Szymanski, A. Saifee, A. Sastry, Y. Liu, and K. Madnani, "Genesis: A system for large-scale parallel network simulation," in *16th Workshop on Parallel and Distributed Simulations*, Washington D.C., May 2002, IEEE.

[8] C. Carothers and B. K. Szymanski, "OS support for checkpointing shared memory parallel programs," *Dr. Dobb's Journal*, 2002, to appear.

[9] M. Gerla and J.T.-C.Tsai, "Multicluster, mobile, multimedia radio network," *ACM/Baltzer Journal of Wireless Networks*, vol. 1, no. 3, pp. 244–265, 1995.

[10] X. Zeng, R. Bagrodia, and M. Gerla, "Glomosim: a library for parallel simulation of large-scale wireless networks," in *12th Workshop on Parallel and Distributed Simulations*, Banff, Alberta, Canada, May 1998.

[11] J. A. Bivens, L. Gao, M. Hulber, and B. K. Szymanski, "Agent-based network monitoring," in *Agent based High Perfor-*

mance Computing, Seattle, Washington, May 1999, Autonomous Agents: ACM SigART.

[12] W. Stallings, *SNMP, SNMPv2, SNMPv3, and RMON1 and 2*, Addison Wesley Longman, Inc., Reading, MA, 3rd edition, 1999.

[13] W. R. Stevens, *TCP/IP Illustrated*, vol. 3, Addison-Wesley Publishing Company, Reading, MA, 1996.