International Journal of Network Management, vol. 14, 2004, pp. 131-146

Scalability and Performance of an Agent-based Network Management Middleware

Alan Bivens Rashim Gupta Ingo McLean
Boleslaw Szymanski Jerome White
{bivenj, guptar, mcleai, szymansk, whitej8}@cs.rpi.edu

Contact: Boleslaw Szymanski szymansk@cs.rpi.edu
Dept of Computer Science,
Rensselaer Polytechnic Institute, Troy NY
Tel: 518 276 2714, Fax: 518 276 4033

Abstract

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 paper 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. The described middleware operates between the network protocol layer and the application layer and uses standard TCP protocol and SNMP probes to interface the network. By aggregating requests from many users into a single agent, our system allows multiple managers to probe problem areas with minimal management traffic overhead. We discuss and quantify the benefits of the described middleware by implementing real-time network managers using our system.

The main result of this paper 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 separating simulated application flow into separate subflows to simplify design of simulations.

This work was partially supported by the grant and fellowship from IBM Corp. to the Rensselaer Polytechnic Institute. The content of this paper does not necessarily reflect the position or policy of the IBM Corp.; no official endorsement should be inferred or implied.

I. INTRODUCTION

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.

Large scale data collection presents serious problems for companies with subnetworks spread geographically. Typically, such companies have no control over paths between their subnets. Most often, ISPs do not allow packets from management applications to travel across their network for security reasons. These difficulties can all be avoided by moving data collection closer to the managed devices.

II. WHAT IS DOORS?

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]. It is truly a middleware between the managers and the network that they

manage, as it uses the standard SNMP queries and TCP protocol to interface routers. 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.

A. DOORS Architecture

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. 1. In this section, we will briefly discuss the purpose of these components.

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) 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 does not detect any data losses during communication. The reliability benefits of TCP are desirable but come at a cost of increased bandwidth needed for acknowledgments (see Section III-B for an explanation of how we counteract it). SNMPv2 defines the PDU GetBulkRequest [21] to enable retrieval of a large amount of information in a single request. However, GetBulkRequest still uses unreliable UDP as an underlying communication protocol. In an application of interest to this paper, network management, using GetBulkRequest does not simplify acquisition of data over time compared to SNMPv1. Collecting data periodically over regular intervals of time requires sending the GetBulkRequest request repeatedly, much like it is done with the SNMPv1. In conclusion, although the results presented in this paper were collected under SNMPv1 queries, using GetBulkRequest would not make a difference. The main conclusions of the paper, that DOORS, compared to SNMP, provides higher reliability thanks to use of TCP and smaller bandwidth overhead thanks to request aggregation and consolidation, hold also for the version 2 of SNMP.

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.

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.

III. 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. 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 [15]. 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). 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.

A. 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. 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. 3. Moreover, the bandwidth used under DOORS remains effectively 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. 4.

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 [16]. 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. 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.

In the case mentioned above, using the GetBulkRequest introduced by SNMPv2 [21], would not result in any more efficiency than when using traditional SNMP. For both, the requests for data must be sent repeatedly after each interval of time. In contrast, the DOORS mobile agent, which is very close to the router, sends the repeated requests for data. The client has to send only single request to the mobile agent, thus, saving bandwidth.

B. Analysis of Measurements

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.

1) 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 [21]. 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) The DOORS Transaction: Part of TCP's overhead is its three-way handshake [23]. 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.**

C. 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 [24]. 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. 5 shows, the autoregressive agent uses very little bandwidth, far less than the bandwidth used by standard SNMP polling. In Fig. 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.

D. Comparisons of all methods

The differences in bandwidth usage across the many methods including our preprocessing case are shown in Fig. 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.

IV. 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 [10]. 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

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 [13]. 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 [17]. An extensive discussion of separable queuing network models, including their requirements, limitations, and extensions, can be found in [18].

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.

A. Example

Fig. 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. 7(b and c).

The Internet path (Fig. 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. 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 simulations. If the total delay is needed, it can be calculated as follows.

$$Delay_{total} = Delay_A + (1 - hit\_ratio) * Delay_B \quad (1)$$

B. Methodology

A formal description of separability follows.

Let T denote the vector describing the network topology and its link capacities (hardware). Let $S_A = < S_1^A, S_2^A, ..., S_k^A >$ denote activities of the sources of the application traffic. Finally, let $S_B = < S_1^B, S_2^B, ..., S_m^B >$ denote activities of the background traffic. The simulation is a function σ, that for a given vector $< T, S_A, S_B >$, uniquely defines the application and background flows denoted by $F_A =< f_1^A, f_2^A, ..., f_k^A >$ and $F_B =< f_1^B, f_2^B, ..., f_m^B)$. Hence,

$$< F_A, F_B >= \sigma(T, S_A, S_B).$$

A simulation with a separated flow F_i is simply run with all the application sources, except i^{th} one, turned off. Thus, source activities for such simulations are defined by the vector $S_i =< \phi, ..., S_i^A, ..., \phi >$ and result is $< F_i, F_B^i >= \sigma(T, S_i, S_B)$.

Let M(F) be the metric of measurements obtained for flow F. For the given background flows B the relative error of separation of flow F_i's from F_A is

$$e = \max_{1 \le i \le k} \frac{|M(F_A) - M(F_i)|}{M(F_A)}$$

In general, as discussed in the next section, the relative error is small if the separated flows do not interact with each other, i.e., if the do not share network nodes at all or use them at different time intervals. If the metric of interest is a flow delay, the relative error is also small (and the application flows are separable) in the following two cases of sharing.

- **Case (1) Utilization on network nodes shared by the separated flows is low.** In this case, the networking nodes which the flows share are underutilized and therefore the interaction of flows on these nodes does not significantly affect either flow. Indeed, for an M/M/1 system, such as a router queue with Poisson inflow, the delay for a server with the processing rate μ and the in-flow λ is $1/(\mu - \lambda) = 1/\mu(1 + u + u^2 \ldots)$, where u is the server utilization. For small utilization, the relative error of approximating application delay by a separated flow delay is at most u, hence, small.
- **Case (2) Utilization on network nodes shared by the separated flows is medium, but the application flow rate is small compared to the background traffic flow rate.** In this case, a network node that the flows share can be well utilized, but the impact of flows F on the queue size of this node is negligible, so the queue contains mainly background flow packets. For an M/M/1 system, repeating the above analysis, we can establish that the relative error of approximation is about $\lambda_i/(\mu + \lambda)$, where λ_i is the rate of one of the separated flows and λ is the rate of the background flow. Since $\lambda_i < \lambda < \mu$, this relative error is small. However, it is important that the server utilization is not high, otherwise even a small change in the inflow will create

a large increase in the total delay. To illustrate this effect, in Figure 8, we plotted a maximum ratio of the application flow to the background flow at which a separation has a precision *e* at the given level of utilization u.

The curves in Figure 8 are a family of hyperbolae. When the utilization is low, the flows are separable. When the utilization is high, the ratio $\frac{\lambda_i}{\lambda}$ must be low for the flows to be separable. All points below the curve are separable. Fig. 9 shows a three-dimensional graph in which *e* changes along a vertical axis. The points on the surface and above shows the values of the strictness parameter *e* for which the flows are separable for the given values of $\frac{\lambda_i}{\lambda}$ and u. When the utilization is minimal, even for the small values of *e*, the flows are separable. Interestingly, even at the relative error of 100% (i.e., when the flows are considered separable even if they change the traffic by 100%) no combinations of high utilization and high values of $\frac{\lambda_i}{\lambda}$ can make the flows separable.

C. Impact of Flow Interactions on Separability

Flows, for the purpose of separability, can be described as sequences of annotated packets. Fig. 10 shows the contents of a flow as a list of q annotated packets and their locations at m discrete times.

In Fig. 10, $P_a(t, c)$ describes the location "c" of packet "a" at discrete time "t" and each location is in the set of all network nodes (including hosts, routers, and links) that a packet may reside at any point in time. Therefore, the interaction of one flow on another can be found by performing a conditional "join" ($\bowtie$) of the two flows, revealing the packets which are at the same network node within the same or adjacent time periods (Fig. 11).

The interaction defined on the packet level is negligible unless packets from one flow enter a network node within $(q_c * x_c)$, where q_c is the size of the queue at node c, and x_c is the average service time at node c. In essence, the interaction between the two is

negligible, if the device has had enough time to totally empty its queue. Formally, this interaction can be defi ned with respect to annotated packet fbws in Equation (2).

$$P_x(t_{a,i}, comp_{x,c}), P'_y(t_{b,j}, comp_{y,c}) \; and \; |t_{a,i} - t_{b,j}| < q_c * t_{p,c} \tag{2}$$

Hence, in the above two parts we have demonstrated that separation can be justified either by proper utilization of shared resources or by separation of interactions in time and space.

D. 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 fbws (shown in Fig. 12).

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.

In this case:

- $A \bowtie B$ involves both network and host level interaction because fbw A shares a small part of the network path of fbw B and both fbws share the repository. However, we rate the network interaction as minimal because the client is only sent the data after fbw B completes its path. The only way the two fbws would signifi cantly interact is if the one way trip time of fbw B's path + polling interval is less than the one way trip time of fbw 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.

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) are shown in Fig. 13 and Fig. 14. 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 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 con-

gestion 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. 13 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. 14 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.**

E. Multiple AS Simulation

To continue our scalability evaluation in larger topologies, we simulate our system in a USA Internet topology, shown in Fig. 15, consisting of 25 virtually identical Autonomous Systems, each containing 1,300 hosts, 27 internal OSPF routers, and one AS BGP boundary router.

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. 16 and Fig. 17 that show the same type of relationship between DOORS and SNMP as Fig. 13 and Fig. 14, 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.

V. Related Works

We have just discussed how 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 [7]. 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 [26]. 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 [12]. 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 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.

In an attempt to improve SNMP latency and bandwidth overhead, SNMPv2 introduced GetBulkRequest [21] which enables grouping together several requests into a single message. However, in an application that requires periodical data collection for management purposes the PDU GetBulkRequest does not help. In such an application, requests need

to be sent to the routers repeatedly in regular time intervals, hence they cannot be sent in a single bulk request. The benefits of using DOORS for this kind of applications are the same for the traditional SNMP queries and GetBulkRequest. Thanks to using TCP, DOORS is more reliable than both SNMP and GetBulkRequest that use UDP. Thanks to distribution of repositories and mobile agents, DOORS is also more scalable than the other two methods that are basically centralized. Finally, DOORS agents can be programmed with functionality (including data filtering or compression) that far exceed basic functionality of either traditional SNMP request or a new GetBulkRequest.

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. [12], 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. [14] 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.

VI. Current and Future Work

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 [20]. 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 III-C). 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.

VII. Summary

Comparison of DOORS and SNMP leads to the following conclusions. In a single-client scenario, we see a cost benefi t 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 fi rm 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. 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.

REFERENCES

[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 Mnagement*, 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 Associaton 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] 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.

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

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

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

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

[19] 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.

[20] 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.

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

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

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

[24] 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.

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

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

Lead Author: Boleslaw Szymanski

Figure 1

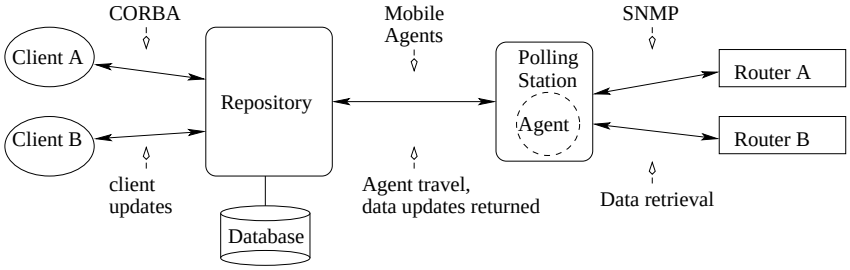


Fig. 1. DOORS Architecture

This page on top. This page Intentionally left blank.

Lead Author: Boleslaw Szymanski

Figure 2

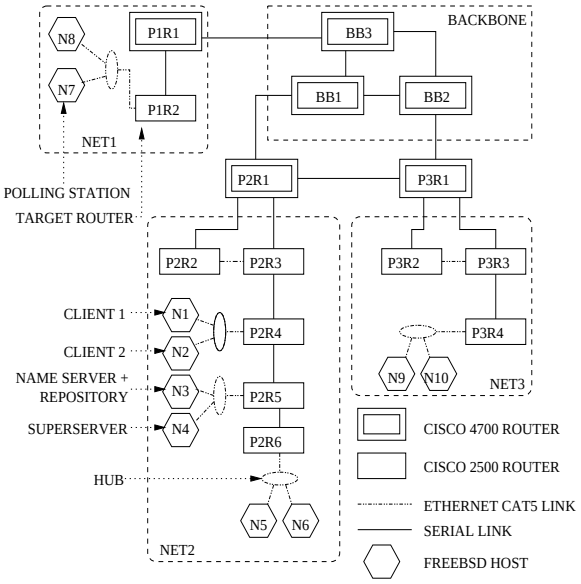


Fig. 2. Sample Autonomous System (AS) Topology

This page on top. This page Intentionally left blank.

Lead Author: Boleslaw Szymanski

Figure 3

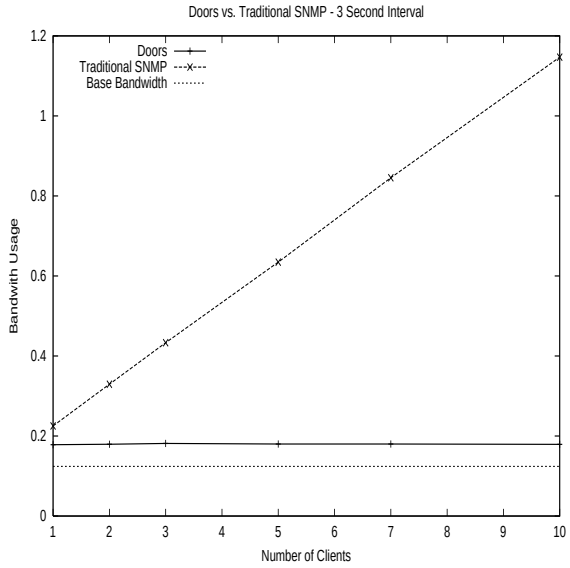


Fig. 3. 3 second interval bandwidth usage plot

This page on top. This page Intentionally left blank.

Lead Author: Boleslaw Szymanski

Figure 4

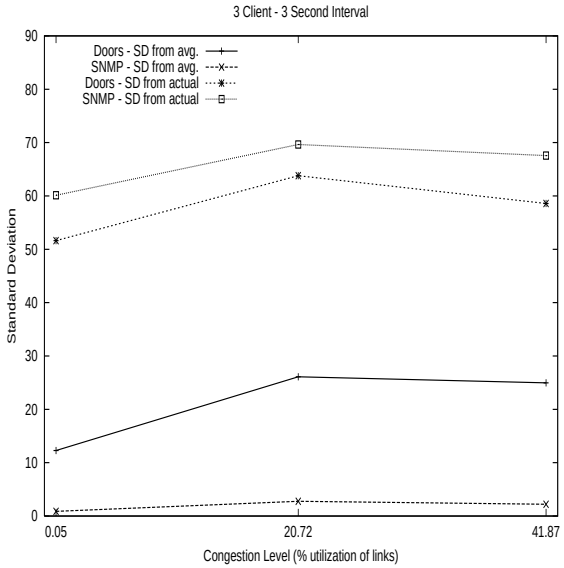


Fig. 4. 3 client isolated network case, standard deviation plot

This page on top. This page Intentionally left blank.

Lead Author: Boleslaw Szymanski

Figure 5

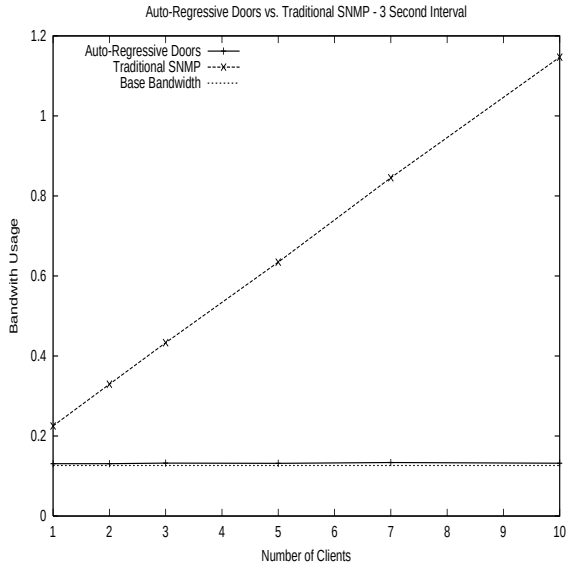


Fig. 5. 3 second interval autoregressive case, bandwidth usage plot

This page on top. This page Intentionally left blank.

Lead Author: Boleslaw Szymanski

Figure 6

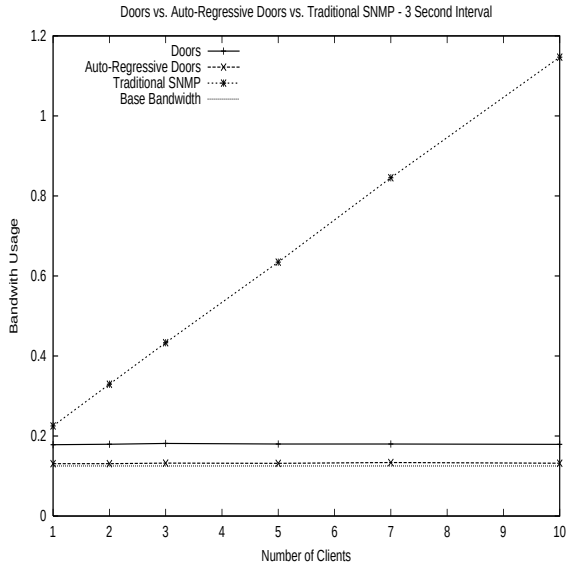
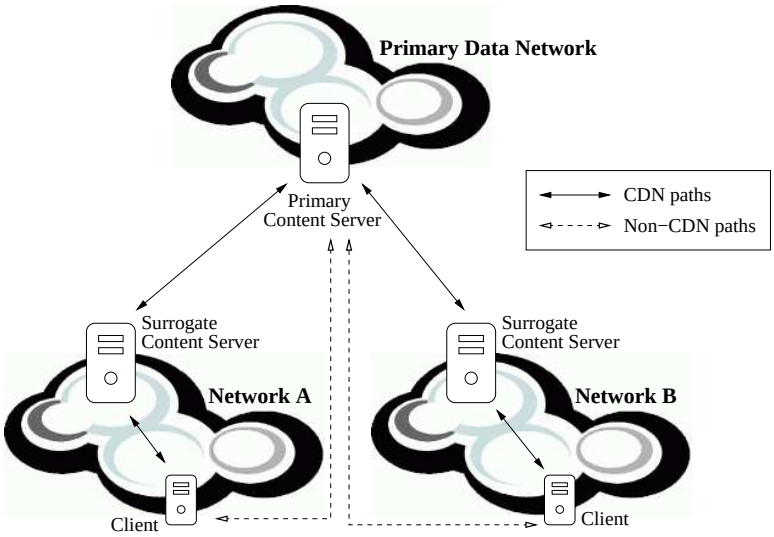


Fig. 6. Joint bandwidth usage plots of 3 second intervals

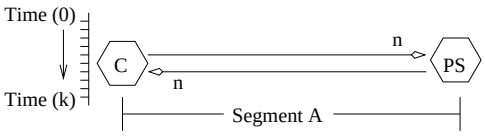
This page on top. This page Intentionally left blank.

Lead Author: Boleslaw Szymanski

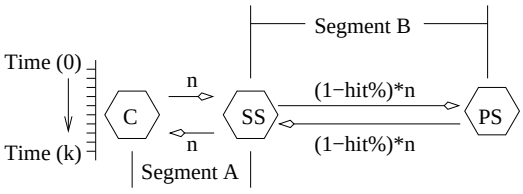
Figure 7



(a) Simple Content Distribution Network(CDN) architecture



(b) Internet path application fbw



(c) Surrogate Server path (CDN) application fbw

Fig. 7. Architecture and fbw diagrams for CDN application.

This page on top. This page Intentionally left blank.

Lead Author: Boleslaw Szymanski

Figure 8

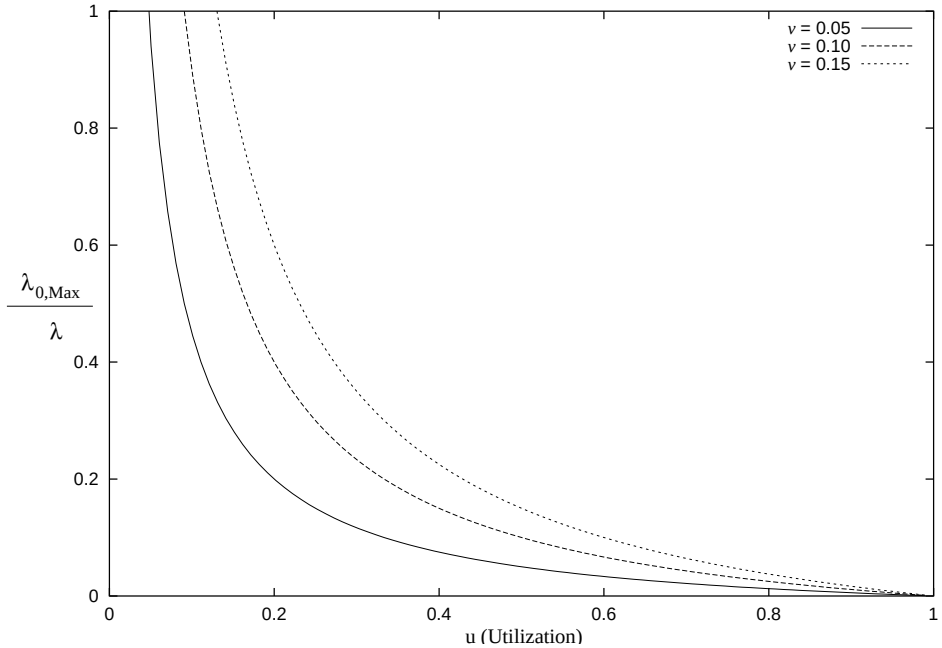


Fig. 8. Plot of $\frac{\lambda_i}{\lambda}$: Points below the curve indicate a separable fbw.

This page on top. This page Intentionally left blank.

Lead Author: Boleslaw Szymanski

Figure 9

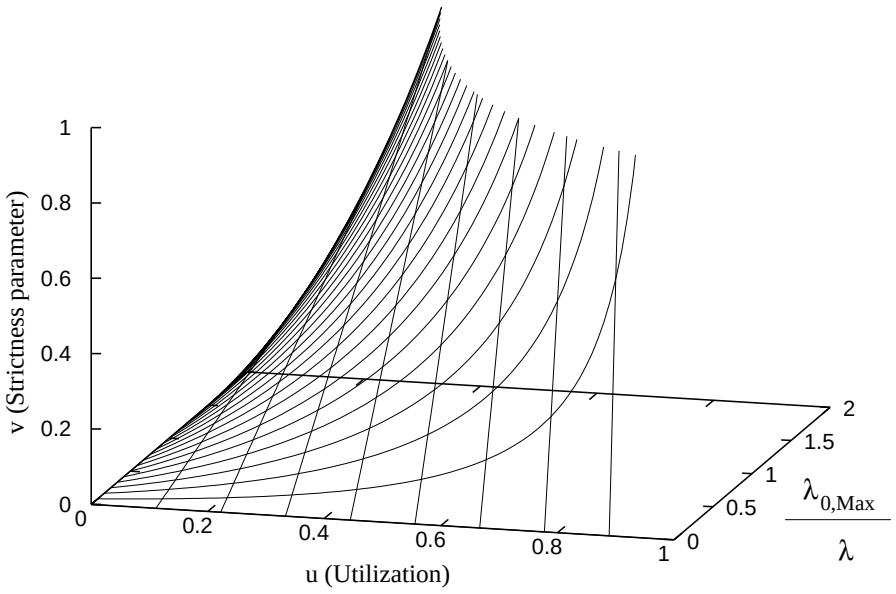


Fig. 9. 3-Dimensional plot of $\frac{\lambda_i}{\lambda} \leq e$: Points above the curved plane defi ne the precision of separation approximation.

This page on top. This page Intentionally left blank.

Lead Author: Boleslaw Szymanski

Figure 10

$$F_A = \begin{bmatrix} P_0((t_{0,0}, c_{0,0}), (t_{0,1}, c_{0,1}), \cdots, (t_{0,m}, c_{0,m})) \\ P_1((t_{1,0}, c_{1,0}), (t_{1,1}, c_{1,1}), \cdots, (t_{1,m}, c_{1,m})) \\ \vdots \\ P_q((t_{q,0}, c_{q,0}), (t_{q,1}, c_{q,1}), \cdots, (t_{q,m}, c_{q,m})) \end{bmatrix}$$

Fig. 10. Annotated packets flow description

This page on top. This page Intentionally left blank.

Lead Author: Boleslaw Szymanski

Figure 11

$$\begin{bmatrix} P_0((t_{0,0}, c_{0,0}), \cdots, (t_{0,m}, c_{0,m})) \\ P_1((t_{1,0}, c_{1,0}), \cdots, (t_{1,m}, c_{1,m})) \\ \vdots \\ P_q((t_{q,0}, c_{q,0}) \cdots, (t_{q,m}, c_{q,m})) \end{bmatrix} \bowtie \begin{bmatrix} P'_0((t'_{0,0}, c'_{0,0}), \cdots, (t'_{0,m}, c'_{0,m})) \\ P'_1((t'_{1,0}, c'_{1,0}), \cdots, (t'_{1,m}, c'_{1,m})) \\ \vdots \\ P'_q((t'_{q,0}, c'_{q,0}), \cdots, (t'_{q,m}, c'_{q,m})) \end{bmatrix}$$

Fig. 11. Description of the join operation between two annotated packet flows

This page on top. This page Intentionally left blank.

Lead Author: Boleslaw Szymanski

Figure 12

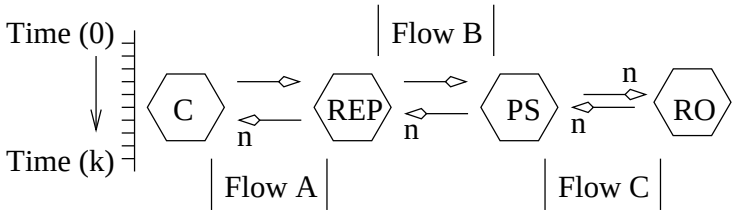


Fig. 12. Illustration of DOORS connection architecture

This page on top. This page Intentionally left blank.

Lead Author: Boleslaw Szymanski

Figure 13

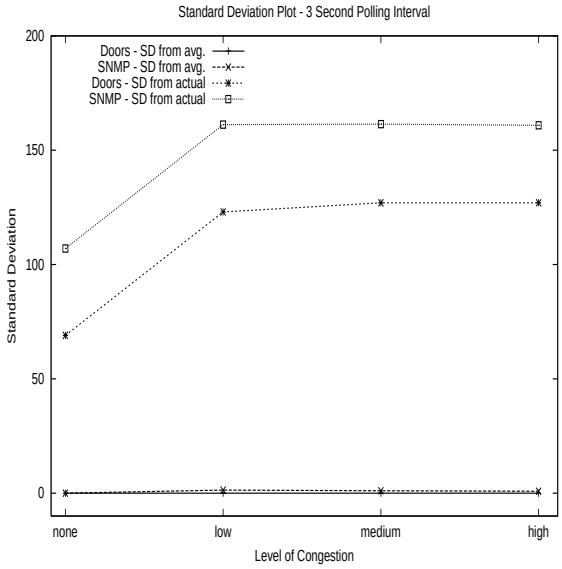


Fig. 13. Standard deviation plot for AS simulation

This page on top. This page Intentionally left blank.

Lead Author: Boleslaw Szymanski

Figure 14

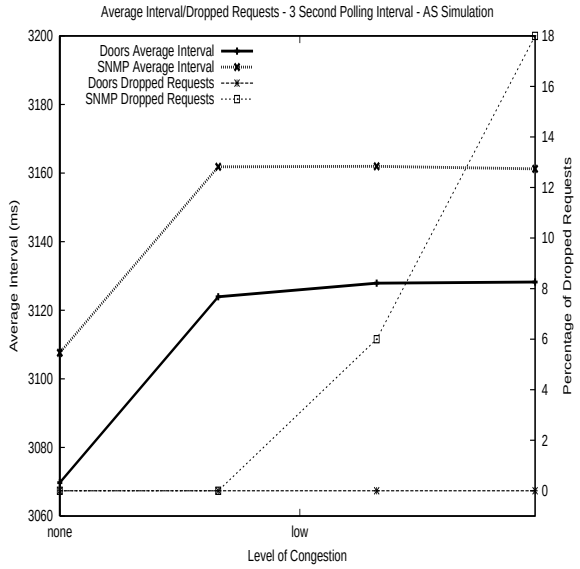


Fig. 14. Average Interval / Packet Drops in AS simulation

This page on top. This page Intentionally left blank.

Lead Author: Boleslaw Szymanski

Figure 15

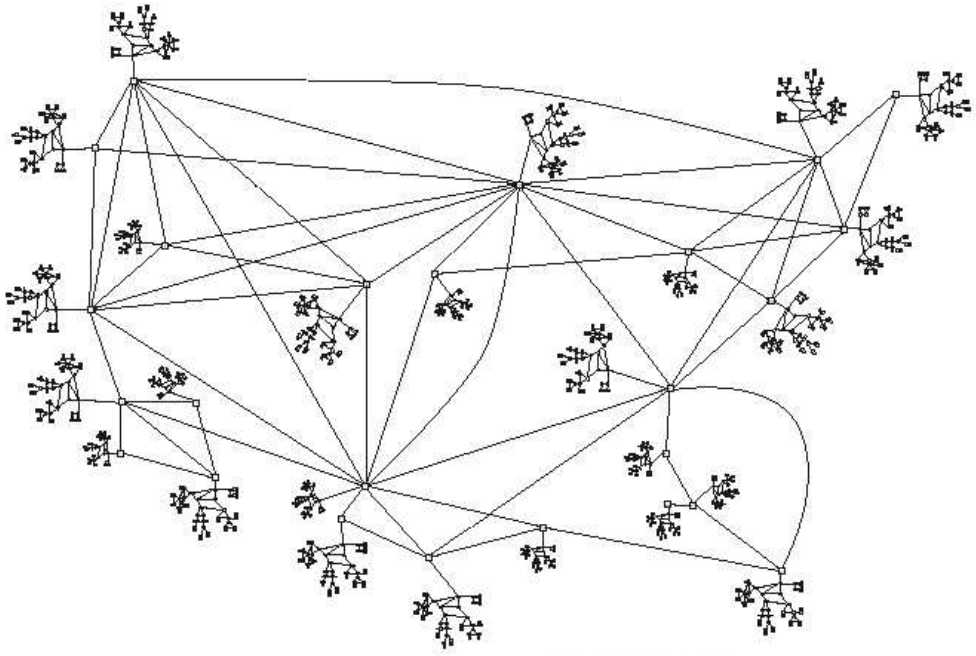


Fig. 15. Sample USA Internet topology

This page on top. This page Intentionally left blank.

Lead Author: Boleslaw Szymanski

Figure 16

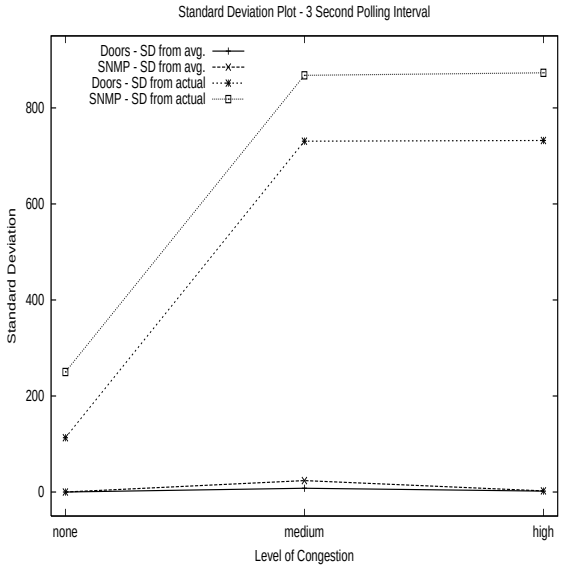


Fig. 16. Standard deviation plot for USA simulation

This page on top. This page Intentionally left blank.

Lead Author: Boleslaw Szymanski

Figure 17

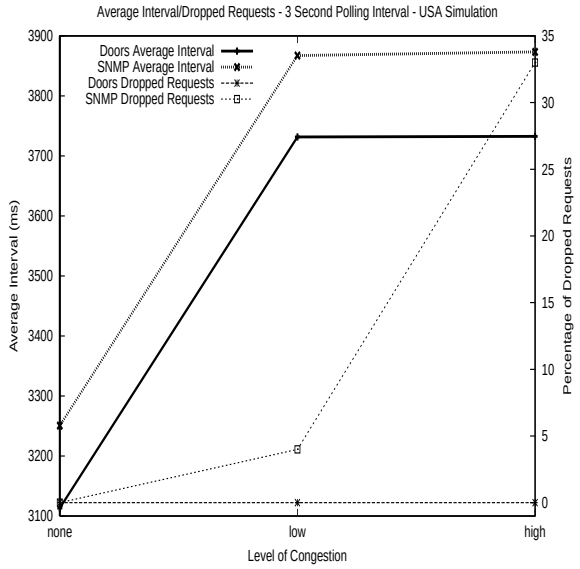


Fig. 17. Average Interval / Packet Drops in USA simulation

This page on top. This page Intentionally left blank.