GNOSIS: Global Network Operations Status Information System

Jessica A. Kornblum and Jonathan M. Smith
{jkornblu,jms}@dsl.cis.upenn.edu
Distributed Systems Laboratory
Computer and Information Science Department
University of Pennsylvania

September 10, 2001

Abstract

Monitoring the global state of a network is a continuing challenge for network operators and users. It has become still harder with increases in scale and heterogeneity. Monitoring requires status information to be near real-time and displayed continuously. A particular challenge is obtaining status information for each node and constructing the global picture at a monitoring point. GNOSIS, the Global Network Operations Status Information System, achieves a global view by careful extraction and presentation of locally available node data. The GNOSIS model improves on the traditional polling model of monitoring schemes by 1.) collecting accurate data 2.) decreasing the granularity with which network applications can detect change in the network and 3.)displaying status information in near real-time.

We define the *Network Snapshot* as the basic unit of information capture and display in GNOSIS. A Network Snapshot is a visualization of locally available state collected during a common time interval. A sequence of these Network Snapshots over time represent the evolution of network state.

In this paper, we motivate the need for a network monitoring system that can detect global problems, in spite of both scale and heterogeneity. We present three design criteria, *Accuracy, Continuity and Timeliness* for a global monitoring system. Finally, we present the GNOSIS architecture and demonstrate how it better detects network problems which are currently of concern. The goal of GNOSIS is to present a stream of consistent, accurate local data in a timely manner.

1 Introduction

The complexity and scale of modern networks have increased, while the capabilities of tools for monitoring and managing them have not. This has a variety of causes, but our belief is that the local autonomy which makes the Internet scalable serves to make a scalable monitoring system difficult. In addition, the heterogenous nature of network devices provides a difficult platform for extracting a stream of consistent, accurate local data as well as representing this data.

A network is a set of *nodes* joined together by *links* over which they communicate. The primary functions of the nodes may differ as well as the link technologies that connect them. At any given time, many nodes and many links are active, moving packets for a wide variety of applications. The behavior of nodes is complex, and can affect performance of applications as well as other nodes in the network.

Network applications use networks as if they are a utility charged with moving data. Our monitoring, however, is of individual nodes and links rather than the holistic view taken by applications. *Traceroute*, for example, derives the route packets travel to a specific destination by causing TTL expiry reports to be returned from nodes on the path by iteratively incrementing the TTL. The expiry reports consist of an ICMP TIME_EXCEEDED response to the sender, indicating the address of the node. From a network-wide perspective, traceroute is used to detect where packets are being dropped along a specific path, but neither provides information about other paths, nor any causal data for the packet losses. Thus, traditional methods for monitoring networks are unacceptable for detecting and diagnosing global phenomena in the network's behavior because they do not visualize a global view. In addition, the granularity with which they represent the network is not acceptable for detecting current network problems.

The most important principle in designing a network monitoring system is the choice of its correctness criteria. We define three major criteria: Accuracy, Continuity and Timeliness, abbreviated ACT. A system which meets these criteria can then be optimized to meet scalability or overhead requirements.

The remainder of this paper is organized as follows. The next section motivates the need for global monitoring and explains the ACT criteria. Section 3 discusses previous work, and Section 4 provides an example of a traditional Network Management System to outline its approaches. We then present the GNOSIS architecture, designed to satisfy the ACT criteria while scaling to large networks in Section 5. Section 6 highlights a particular element of GNOSIS, the *network snapshot*, while Section 7 covers some GNOSIS implementation strategies and applications. Section 8 concludes the paper.

2 Global Monitoring Principles

This section presents definitions that differentiate global monitoring from local measurement and local monitoring. It outlines desiderata for a global monitoring system, and discusses architectural issues for ACT criteria.

2.1 Local versus global properties - an example

The scale and heterogeneity of modern networks such as the Internet creates a difficult environment to monitor globally (network wide). However, with increases in scale, the need to achieve a global view becomes even more important because problems arise that are better dealt with on global-scale.

A useful analogy is that of air traffic control systems. If air traffic controllers monitored the airspace at the same level we monitor our networks, at "airplane-level", the radar systems would determine positions of each plane in the air space, but never realize their (more important) relative positions. The latter is necessary to detect a possible collision and is accomplished by constructing a global view of the airspace.

We can engineer a network monitoring system to provide a global view of the network as a real air traffic control system does, by integrating data (*e.g.,* positions and velocities for individual airplanes) together in a global coordinate system. Thus, global network monitoring systems must monitor the "network airspace" to determine how nodes and links interact and behave as a system. Monitoring a single point in the system does not provide a global perspective. We must collect a set of data from various points in the network. Data can be gathered using both active and passive monitoring techniques [32]. However, the following criteria must apply to a set of data to reflect the whole network.

2.2 Principles for Global Network Monitoring

Independent of any optimizations such as scalability and overhead requirements, three principal criteria define a global network monitoring system:

1. **Accuracy**, meaning that the data are precise, complete and consistent when treated as a set from a point in time.
2. **Continuity**, meaning that a sequence of data sets are presented as a stream, to reflect the time-varying nature of network status.
3. **Timeliness**, meaning that the sequence of data sets presented is "near" real time (while this could as well be considered an accuracy property, we prefer to separate time from other data set properties).

Reflecting their initials, we call these the *ACT criteria*.

We can use these criteria to differentiate global network monitoring from other tasks such as local network monitoring and network measurement. Local monitoring may satisfy timeliness and continuity constraints (*e.g.,* a typical use of the `ping` command) but are not easily composed into a network wide picture without (significant) off-line analysis. Network measurements might have requirements of precision, and if they are global, they may have other consistency requirements which allow them to achieve accuracy. However, since the measurements are generally intended to be used offline, they need not meet our timeliness requirement.

A global network monitoring system must present collected status information in a near real-time manner, and do so continuously. Three basic steps are necessary for a system to meet this requirement: (1) gathering status information from the network; (2) converting gathered information into a logical set; and (3) representing this logical set visually and displaying it to the system user. Depending upon the user's needs, the metrics for accuracy, continuity, and timeliness may vary. For example, if a global network monitoring tool is to be used by managers to detect and diagnose problems with a sudden onset, such as the increasingly common denial of service attacks, timeliness may be emphasized, with the system optimized accordingly. Other network problems may require metrics more suited to rare events, such as exception monitoring, rather than aggregate measures. In GNOSIS, we focus on optimizing for rapidly changing network environment.

2.3 Architectural Issues and the ACT Crieria

This section discusses how the correctness criteria of ACT can be achieved while maintaining acceptable performance.

2.3.1 Accuracy

Precision is a measure of the detail at which monitoring information could be collected. Units might include bits, bytes, packets, flows and connections, for example, or other sorts of structured events. When timestamps are used, the precision of the clock will be important. Status information is considered accurate if it deviates only within acceptable limits.

Completeness is a measure of whether we gather status information from every node in the network or from a representative subset. A monitoring system might require that each visual representation include status information from every node in the network. This might pose significant barriers to the timeliness and continuity criteria due to the latency to collect data from every node. A subset of nodes might represent a better design. The former approach may be necessary for applications that want to detect complex behaviors among nodes in the network that are configuration-determined. Thus information is needed from each node to completely understand how the nodes are interacting. Alternatively, an application requiring sampled network characteristics needs a less complete set of information.

Heterogeneity, the diversity of the nodes and links in the network, is a major challenge to gathering of precise status data, since not all devices will have equally precise clocks and equally complete data gathering capabilities. Variation in both interfaces and information available introduces additional complexity into the monitoring system. Heterogeneity is simply a fact of life, *e.g.,* specialized devices such as [9, 20, 30, 21].

To provide accurate global information, status data gathered at various points in the system (i.e. individual nodes and/or links) must be *correlated* into a whole. If this information is to be displayed as a picture, it must be correlated in some functional or qualitative way, such as being of a common type, gathered in a particular locale, or from a single point in time. Otherwise the picture is useless for detecting and diagnosing global phenomena.

2.3.2 Continuity

The *consistency* of status information is a measure of the interarrival delay of consistent data (that is, for example, time correlated). Network status information is constantly changing over time. The level of consistency determines what kinds of behavior can be detected by the monitoring system. Status information that changes infrequently, such as node IP Addresses, requires little work by the monitoring system to keep consistent. On the other hand, if the system wanted to display process load, a rapidly changing metric, a monitoring station would need to quickly gather and collate this information to achieve consistency. Consistency is closely related to the granularity with which a network is monitored (described below).

We consider the *monitoring granularity* to be a measure of the smoothness achievable in a global network monitoring system. This is limited by either the rate at which consistent samples of the system status can be constructed and displayed, or the rate of change in the system. In a real network, the former will almost certainly be the limit to system performance. In the latter case, however unlikely, a Nyquist-like sampling rate limit could optimize polling behavior. The bottom line is that a global network monitoring system must gather and display status information efficiently to be able to detect rapidly changing network behaviors.

2.3.3 Timeliness

An essential question in achieving acceptable timeliness is the *overhead* imposed by global network monitoring on the monitored system. A network monitoring system that is used to detect problems in real-time must impose a small overhead on network resources. Often, network resources are a scarce commodity when the network exibits problematic behavior. Such is the case during a denial of service (DoS) attack. A DoS attack aims to consume network resources, thus rendering the network unable to provide services. Obviously, a monitoring system geared towards detecting this kind of problem must consume a low amount of bandwidth and processing time. On the other hand, network monitoring systems that have high overhead (Complex Event Processing [22], for example) can be used while the network is under-utilized, since the processing and bandwidth required have little effect on the ultimate timeliness with which data are delivered.

An issue strongly related to overhead is the *cost of infrastructure* necessary to gather information in a timely fashion. Monitoring systems can have a severe impact on network resources. One approach to limiting this impact is to construct a dedicated network for the monitoring system. This could be done by simply dedicating a processor in each node and connecting every node by an additional link. However, the cost of adding such dedicated resources is high.

Finally, there is the important issue of *scalability*, which is the limit in the number of nodes monitored accoring to the ACT criteria. As an example, assume the system overhead is a constant factor of the number nodes in the system. A small constant would render our system more scalable than a larger constant if we were trying to achieve a monitoring system with low overhead. The required scalability is derived from the metrics required of a specific system.

3 Previous Work

In the following text, we categorize existing monitoring tools and systems into two categories: 1.) *Node and Link level*; 2.) *Network level*. For those systems that do monitoring at the network level, we describe how they fail to achieve the ACT criteria for global monitoring.

Node and Link Level *Ping* and *Traceroute* are the most widely used tools for detecting problems. Ping is used to determine host reachability. Traceroute exports the route packets take from a source to a destination. Round trip time values are given for each intermediate node along the path. Both of these tools are useful for determining node level characteristics but do not provide insight into the health of the global network because they collect data from only one point in the network. Similarly, packet sniffers used for link measurement, such as [3, 31, 24, 17], capture and analyze link characteristics. This information is not plotted into a global framework. However, it can be used to extract statistical models of traffic characteristics.

Network Level Recently, several Internet monitoring and management projects have created architectures and frameworks for aggregating data compiled from the node and link monitoring tools mentioned above to discern end-to-end performance measurements in the Internet. PingER sends a succession of ping requests to a pre-determined number of hosts, collects the RTT of each and computes the average end-to-end delay [23]. Surveyor project is similar, but synchronizes each end host to determine unidirectional delay and loss [15]. Other projects archive data generated from monitoring and measurement tools include [8, 27, 38, 26]. Data collected by these projects are correlated together and sometimes visualized, but not in near real-time.

The time spent gathering status information affects the granularity with which the network can be monitored. Active network [10] and mobile agents [34, 35] research have reduced the latency to gather status information from nodes in the network and thus improved the monitoring granularity. ABLE, SmartPackets and Distributed Management by Delegation delegate management functions to the managed network element [12, 18, 37]. However, the set of data collected does not have the accuracy criterion needed to correlate the data points into a global space. Data cannot be collect at the same point in time with any precision.

The NIMI infrastructure was designed as a control framework for large-scale data collection in a secure environment. Access control policies determine what monitoring tools are allowed to execute. The architecture separates management tools for collecting data from the important aspect of managing which tools *can* gather information in the network [32]. We are considering the NIMI architecture for control and security in the GNOSIS implementation.

HP Openview uses a more centralized approach to gathering status information. The application sends a series of SNMP requests to a set of managed nodes in the network. State is collected and visually represented to the user [13]. This method lacks the timeliness criterion. Latency for gathering status information, collating it into a global space and displaying the global space is too high to monitor rapidly changing network characteristics and problems, and collected data lacks temporal accuracy.

In the next section (4), we further motivate the need for a global monitoring system that satisfies the ACT criteria by presenting a traditional centralized polling model for network monitoring. We describe two network problems, *ICMP sweeps* and *Denial of Service* attacks and demonstrate the weakness of the traditional model for detecting them.

4 Centralized Polling Model

Current network monitoring systems attempt to gather and visualize status information at the network level, but fail because the models do not optimize for Accuracy, Continuity and Timeliness (as described in Section 2). There are three basic steps required to visually display network status information into a global view: *gathering data*, *converting the data to a picture*, and *displaying the picture*. A common model for achieving this is called the Centralized Polling Model and is shown in Figure 1.

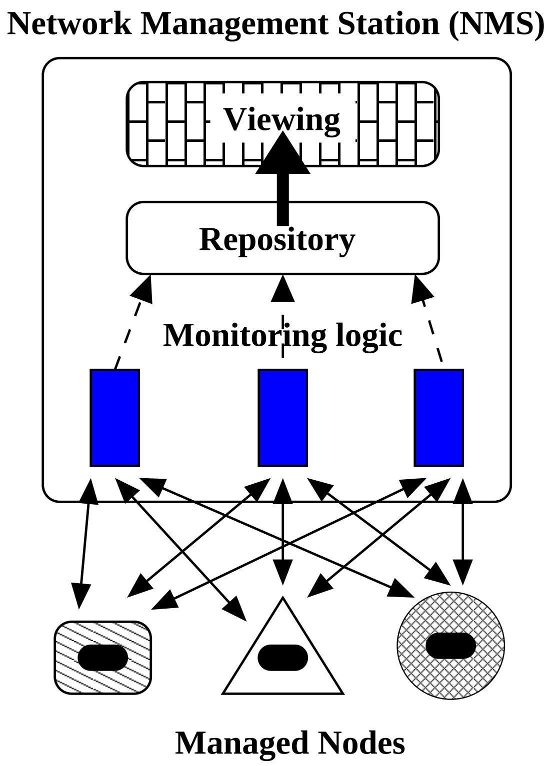

Figure 1: Centralized Polling Model

A Network Management Station (NMS) gathers status information (usually stored in a MIB [25]) by sending a series of SNMP *get* commands [6] to the node individually. Local status information is retrieved, collated and displayed to the user. HP OpenView is an example of such a system [13]. This model has a number of drawbacks when expected to detect current network problems such as *ICMP Sweeps* [14, 4, 33] and *Denial of Service* attacks [1, 36, 7].

ICMP Sweeps The Internet Control Message Protocol (ICMP) was designed to send network control information between nodes. It can be used to learn and study a target network, gaining topology and configuration information [33]. Such information is ammunition for serious network attacks [4]. For example, determining the IP address of a webserver could be used later in a denial of service attack.

During an ICMP Sweep, a series of ICMP packets are sent to various nodes in the network. Simply by using *ping* in broadcast mode, a sweep can be done with a single packet. Discerning the difference between normal ICMP traffic and Sweep traffic is impossible without some mechanism to correlate data temporally. In the Centralized Polling Model, a NMS requests ICMP packet counters from each node, detects that each packet counter, supported by MIB-II ICMP table, has incremented by 1 (from receiving the single broadcast ping packet), but cannot determine if the increase is due to a sweep. This model is missing a mechanism to correlate data into a global space.

Denial of Service Attacks Denial of Service attacks are launched with little effort [7] with the goal of depleting node or system resources thus rendering the system unusable. A common approach, called SYN Flooding, sends a series of connection requests to a victim's machine from a spoofed source address. The target machine allocates data structures to handle the false attacks. Since resources are inherently limited, a flood of SYN requests can quickly consume all available resources [36].

Network and system characteristics during a DoS attack change from "normal" behavior to extreme utilization in a quick time interval. Traditional monitoring techniques are helpless in detecting such a rapid change because the polling data extraction mechanism incurs too much latency to detect the change in a timely way.

In addition to detecting a DoS attack after it has already occurred, an ideal situation would be to detect network behavior before an attack reaches the target destination. In the case where the attack is launched from a set of sources, a Distributed Denial of Service (DDoS) attack [5, 29], we could expect a smaller increase in resource consumption at many points in the network, where the aggregate load exceeds total resources available at the target. To detect a DDoS attack, a monitoring system must collate the individual resource usage metrics into the global network framework and determine if the aggregate exceeds the total available resources further down the path. To detect such a change, the monitoring granularity must be very sensitive and able to quickly calculate an accurate view.

The NMS architecture also fails to detect this kind of change in the network because it cannot correlate data collected from distributed points into a global representation. We need to know what the load is at different points in the network during the same time interval. In addition, the global representation should be displayed quickly.

In the following section, we describe the GNOSIS, (Global Network Operations Status Information System), architecture and describe how it achieves the ACT criteria and improves handling network problems.

5 The GNOSIS Architecture

Figure 2 illustrates the architecture of our Global Network Operations Status Information System, GNOSIS.

We have extracted the functions of the NMS and distributed them throughout the network to eliminate the scalability problem with the centralized approach. We have four basic components in our architecture: *Monitoring Node*, *Set of Managed Nodes*, *Repository* and the *Set of Viewing Nodes*. Each play a separate functional role in taking a Network Snapshot. [1]

Monitoring Nodes manage and control the monitoring logic. They set the timer for each snapshot and distribute the data collection logic to the set of managed nodes.

Managed Nodes are the set of network devices being visualized at the Viewing Stations. The specific type of device may vary; however, each managed node must accumulate some local measurements, shown as black ovals, and provide an interface to access them. A monitoring program (dashed line) is deployed to the managed node, essentially setting a timer for capturing local state. State (solid line) is sent to the Repository.

A **Repository** collects local measurements from the managed nodes, sorts by type of state and the time it was extracted from the set of nodes. State is transformed into a picture, encoded in the format expected by the visualization software and sent to the viewing nodes (dotted line).

Viewing Nodes collect a stream of pictures of the network and present them to the user. Viewing stations should display the same sequence of pictures.

[1] A *Network Snapshot* (discussed further in Section 6) is the basic unit of capture and display in time. A sequence of Network Snapshots visually represent the change in the network as time progresses.

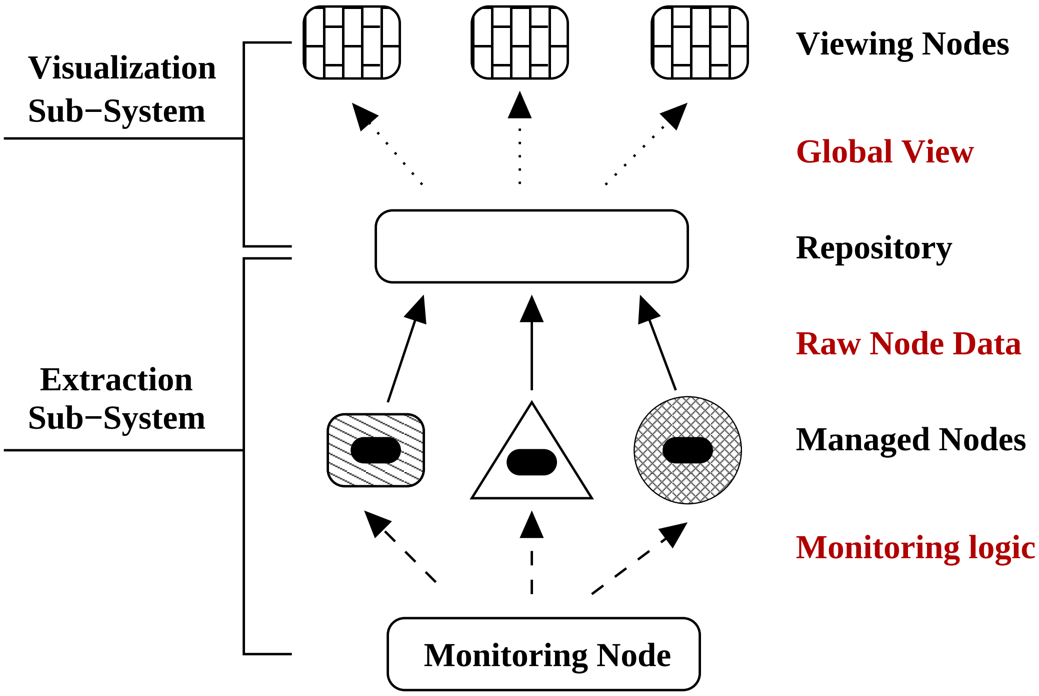

Figure 2: GNOSIS Architecture

Data flow in GNOSIS is indicated by arrows in Figure 2 and is described by the following sequence:

1. Management logic is distributed to the Managed Nodes,
2. At a specified time, t, managed nodes capture local status information and send it to the Repository,
3. The Repository sorts the data by type and time, t, constructing a coherent picture of the nodes,
4. Global views are exported to the Viewing Nodes and displayed.

Recall our design criteria for a global monitoring system: ACT. GNOSIS architecture satisfies each criterion.

Accuracy In GNOSIS, we monitor problems in the network by analyzing how a set of local status information changes over time. Accuracy is a property of this set. Coordinating when data is extracted from the node based on an expiry provides precision and consistency amongst the set of data in relation to time. The Repository ensures visualizations of a set are consistent with respect to a time deviation.

Continuity GNOSIS continually extracts and visualizes sets of local data. We have optimized our architecture to achieve a rapid flow of visualizations by decreasing the number of in band tasks to only include basic mechanisms for data extraction, interpolation and visualization. Distributing management code, subscriptions for receiving visualizations and authorization mechanisms are handled out of band. Completing the extraction and visualization cycle as quickly as possible enables a finer granularity of network monitoring and thus the ability to notice rapidly changing network characteristics.

Timeliness The GNOSIS architecture uses active network techniques to move management logic closer to the Managed Nodes thus reduce the latency for the gathering process. The cost to timeliness of gathering status information is the propagation delay between the monitoring points and the managed nodes. In a simplistic serial polling model, the cost is the sum of the link delays. In a concurrent polling model, the timeliness cost is proportional to the slowest link. In GNOSIS, we have reduced polling latency by moving the polling logic close to the managed nodes. Timeliness becomes dependent on how quickly the interface between the active network element's execution environment and its programmed monitoring logic can return status information.

In the next section, we describe our basic unit of data capture and display in GNOSIS called the *Network Snapshot*.

6 Network Snapshots in GNOSIS

We define a network snapshot in three dimensions: <*node #, type, time*>. A *type* is, for example, an ICMP event (ECHO packet received) or a node resource (processor load). The *node #* is the identity of a monitored node. *Time* is the local time at which the datum is gathered. We assume local clocks are synchronized with NTP or some other means.

A *network snapshot* is a picture of the network at a specified point in time, T. Let d be our delay limit (*i.e.* how much deviation from a perfect T is allowable). s_i is the status information stored at node i and t_i is the local time at node i. A **Network Snapshot** is the set of triples $\{(1, s_1, t_1), (2, s_2, t_2), ..., (n, s_n, t_n)\}$ produced by the following sequence of events:

- Monitoring node distributes a program to the Managed Nodes with an time expiration value, T (T's value must be greater than t_i for all i). T is the time we want to extract data from each managed node.
- when $t_i >$T, the program executes, collecting s_i from the node.
- the Managed Node sends (i, s_i, t_i) to the Repository,
- the Repository sorts collected data first by s then by t such that all s_i's in a group are the same *type* and $T - d \leq t_i \leq T + d$ holds true for all t_i's.
- Groups of triples are distributed to the Viewing Nodes

Let R be current time. The following equations more precisely define the ACT criteria described in Section 2 with respect to Network Snapshot definitions.

- *Accuracy:* For each value sent to the Repository, data is sorted by *type* and *time* according to the properties mentioned above. This ensures that all s_i in a group has been extracted from the node with acceptable precision, defined by d. Furthermore, we guarantee time consistency with the set.
- *Continuity:* Network Snapshots are sent to the Viewing Nodes continuously. Our goal with GNOSIS is to optimize our architecture for a small value of, d providing a view of the network at a finer granularity. Another axis affecting monitoring granularity is how close together the values of T are between network snapshots. We want to generate network snapshots with close values of T for finer monitoring granularity.
- *Timeliness:* Network Snapshots taken at time T are displayed at time R. R should be as close to T as possible as defined by "near real-time". The difference in the values is the latency in gathering, collating and displaying status information in GNOSIS.

We discuss how to implement GNOSIS to achieve *Accuracy* (small value of d), Continuity (close values of T) and Timeliness (close values of T and R) in the next section.

7 GNOSIS Implementation

Our distributed monitoring code is implemented with the SNAP execution environment. SNAP provides a lightweight processing environment for active packets. [28] shows that SNAP achieves the same processing performance as IP for PC-based routers. In addition, SNAP's in kernel implementation reduces the latency of extracting status information by three orders of magnitude over JAVA-based mobile code [2]. Since we have moved the management code, the precision in which we can gather data is limited by the performance of SNAP's interpreter and the granularity of the clock mechanism (described below). Perhaps most importantly, SNAP's resource bounded execution make it safe to execute.

We correlate the status information collected from different nodes by introducing a global clock mechanism, such as [19, 11, 16], at the managed nodes. Adding synchronization to the nodes provides a means of grouping information into logical pictures. However, we must determine which mechanism will impose the least amount of overhead on the network while allowing us to satisfy the accuracy goal.

Implementing a shared time base is relatively simple. Various tools already exist [19, 11, 16] to synchronize distributed clocks. However it is impossible to achieve full synchronization. The monitoring granularity is bounded by the accuracy of the shared clock. We will choose a mechanism that has the best accuracy and imposes the lowest overhead.

We achieve a fully-consistent representation of the network by sorting data centrally by type and time. Each type of status information, (i.e., load, number of TCP connections, etc.) is grouped together into time intervals based on when the information was extracted in the system. The length of the time interval and how often the metric changes affect our accuracy model.

8 Conclusion

We have made three contributions in this paper.

First, we have developed a model for a global network operations monitoring system, the ACT model, which defines criteria for global network monitoring. These three criteria are Accuracy, Continuity and Timeliness. If these criteria are satisfied, a system can monitor global network operations successfully, and additional performance criteria can be imposed to achieve scalability or low overhead.

Second, we have developed an advanced architecture, the Global Network Operations Status Information System (GNOSIS), which meets the ACT criteria while providing significant performance and flexibility advantages. GNOSIS provides these advantages through use of active network technology placed close to monitored systems. Safe and Nimble Active Packets (SNAP) allows polling to be performed locally in a low latency environment, while pushing data to subscribers wishing to further aggregate or visualize data. This organization permits a clean LAN/WAN separation, and overcomes many challenges from round-trip delays faced by traditional management systems.

Third and finally, we have developed the novel idea of a "network snapshot" as the unit of time-consistent information in GNOSIS. This seemingly simple idea not only creates a locus for data consistency decisions, but permits inter-snapshot data compression techniques to be applied, towards considerable gains in performance.

The main goal of our future work is experimental evaluation of a GNOSIS prototype and refinement of the architecture based on the results of the evaluation.

Acknowledgements: The authors would like to thank Geoff Egnal and Stefan Miltchev for reading early drafts of this paper and E. Christopher Lewis and Jonathan Moore for enlightening discussions.

References

[1] Denial-of-service attack via ping. CERT Advisory CA-96.26, December 1996.

[2] Scalable distributed management with lightweight active packets. Technical Report MS-CIS-01-26, University of Pennsylvania, 2001.

[3] K. G. Anagnostakis, S. Ioannidis, S. Miltchev, and J. M. Smith. Practical network applications on a lightweight active management environment, 2001. To be publish in IWAN 2001 Proceedings.

[4] Ofir Arkin. ICMP usage in scanning. Sys-Security Group, July 2000. `www.sys-security.com` and `www.itcon-ltd.com`.

[5] "carko" distributed denial-of-service tool. CERT Incident Note IN-2001-04, April 2001.

[6] J. Case, M. Fedor, M. Schoffstall, and J. Davin. A simple network management protocol (SNMP), May 1990. RFC 1157.

[7] Denial of service tools. CERT Advisory CA-1999-17, December 1999.

[8] Kenjiro Cho, Koushirou Mitsuya, and Akira Kato. Traffic data repository at the WIDE project. In *USENIX 2000 FREENIX track*, June 2000.

[9] Cisco IP Telephones, June 2000. `http://www.cisco.com/univercd/cc/td/doc/pcat/iptl.htm`.

[10] David L. Tennenhouse and Jonathan M. Smith and W. David Sincoskie and David J. Wetherall and Gary J. Minden. A survey of active network research. *IEEE Communications Magazine*, 35(1):80 – 86, January 1997.

[11] D.L. Mills. Network time protocol (NTP), September 1985. RFC 958.

[12] German S. Goldszmidt. *Distributed Management by Delegation.* PhD thesis, Columbia University, December 1995.

[13] Hpopenview. `http://www.openview.hp.com`.

[14] `http://www.opensystems.com/support/docs/6332/expsig_2100.html`.

[15] Sunil Kalidindi and Matthew J. Zekauskas. Surveyor: An infrastructure for internet performance measurements. In *INET*, June 1999.

[16] E. Kaplan. Understanding gps: Principles and applications, 1996.

[17] K. Keys, D. Moore, R. Koga, E. Lagache, M. Tesch, and K. Claffy. The architecture of coralreef: an internet traffic monitoring software suite. In *Proceedings of PAM2001*, April 2001.

[18] Jessica Kornblum, Danny Raz, and Yuval Shavitt. Active process interaction with its environment. In *Active Networks: International Working Conference*, pages 114 – 129, October 2000.

[19] D. Loy. Gps-linked high accuracy ntp time processor for distributed faulttolerant real-time systems, 1996.

[20] Enterprise class IP solutions products, 2000. `http://www.lucent.com/enterprise/solutions/eclips/product_3po.html`.

[21] ORINOCO: Your mobile broadband connection, 2000. `http://www.wavelan.com`.

[22] David C. Luckham and Brian Frasca. Complex event processing in distributed systems. Technical Report CSL-TR-98-754, Stanford University, August 1998.

[23] Warrent Matthews and Les Cottrell. The pingER project: Active internet performance monitoring for the HENP community. *IEEE Communications Magazine*, May 2000.

[24] Steven McCanne and Van Jacobson. The bsd packet filter: A new architecture for user-level packet capture. In *Proceedings of the Winter USENIX Conference*, pages 259 – 69, January 1993.

[25] K. McCloghrie and M. Rose. Management Information Base for Network Management of TCP/IP-based internets: MIB-II, March 1991. RFC 1213.

[26] A. McGregor, H-W Braun, and J. Brown. The NLANR network analysis infrastructure. *IEEE Communications Magazine*, 38(5):122–128, May 2000.

[27] D. McRobb, K. Claffy, and T. Monk. Skitter: Caida's macroscopic internet topology discovery and tracking tool, 1999.

[28] Jonathan T. Moore, Michael Hicks, and Scott Nettles. Practical programmable packets. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies*, April 2001.

[29] "mstream" distributed denial of service tool. CERT Incident Note IN-2000-05, May 2000.

[30] NEC announce IP strategy for corporate network solutions in the asian market, June 2000. `http://www.nec.co.jp/english/today/newsrel/0006/2101.html`.

[31] Vern Paxson. Bro: A system for detecting network intruders in real-time. *Computer Networks*, 31(23-24):2435–2463, December 1999.

[32] Vern Paxson, Andrew Adams, and Matt Mathis. Experiences with NIMI. In *Passive and Active Management*, 2000.

[33] J. Postel. Internet control message protocol. Technical report, IETF, September 1981.

[34] Antonio Puliafito and Orazio Tomarchio. Using mobile agents to implement flexible network management strategies. *Computer Communications Journal*, 23(8), April 2000.

[35] Marcelo G. Rubinstein and Otto Carlos M. B. Duarte. Evaluating tradeoffs of mobile agents in network management. *Networking and Information Systesm Journal*, 2(2), 1999.

[36] "Christoph L. Schuba, Ivan V. Krsul, Markus G. Kuhn, Eugene H. Spafford, Aurobindo Sundaram, and Diego Zamboni". "analysis of a denial of service attack on tcp". In *"In Proc. of the 1997 IEEE Symp. on Security and Privacy"*, pages 208–223, May 1997.

[37] B. Schwartz, A. Jackson, T. Strayer, W. Zhou, R. Rockwell, and C. Partridge. Smart packets for active networks. In *OPENARCH'99*, pages 90–97, New York, NY, USA, March 1999.

[38] Rich Wolski, Neil Spring, and Chris Peterson. Implementing a performance forecasting system for metacomputing: The network weather service. Technical Report TR-CS97-540, University of California, San Diego, 1997.