Data Mining for Autonomic System Management: A Case Study at FIU-SCIS

Technical Report: FIU-SCIS-2006-03-01

Tao Li, S. Masoud Sadjadi, Juan Carlos Martinez, Lokesh Sasikumar, Manoj Pillai
Autonomic Computing Research Laboratory
School of Computing and Information Sciences
Florida International University
11200 SW 8th St., Miami, FL, 33199, USA
{taoli,Sadjadi,jmart054,lsasi001,mpill001}@cs.fiu.edu

ABSTRACT

Over the years, the advancements in science and technology have led to the increased complexity in computing systems. The systems are thus becoming increasingly more complex with growing number of heterogeneous software and hardware components, increasingly more difficult to monitor, manage and maintain. As a result, it is not a trivial task to provide high performance, high dependability, and high manageability for such computing systems. In this paper, we first present an integrated data-driven architecture for computing system management and then present a case study on Autonomic System Manager: a software system we developed that is currently being used by the system and network administrators of School of Computing and Information Sciences (SCIS) at Florida International University (FIU).

Keywords

Data mining, autonomic computing, self management, self protection, network and system management, anomaly detection

1. INTRODUCTION

With advances in science and technology, computing systems are becoming increasingly more complex with a growing number of heterogeneous software and hardware components. They are thus becoming increasingly more difficult to monitor, manage and maintain. The increasing complexity also worsens system dependability as the failure rate for a system is much higher than before. Therefore, it is not a trivial task to provide high performance, high dependability, high scalability and high manageability that are demanded by enterprise customers. There is thus a pressing need for automatic and efficient approaches to monitor and manage complex computing systems [16].

Traditional approaches to system management have been largely based on domain experts through a knowledge acquisition process that translates domain knowledge into operational rules, policies and dependence models [3,4,11,14,15,19,23,28,34]. This has been well known and experienced as a cumbersome, labor intensive, and error prone process. Moreover, it is extremely difficult, if not impossible, to keep up with the rapidly changing environments. It has been estimated that, in medium and large companies, anywhere from 30% to 70% of their information technology resources are used as administrative (maintenance) cost [25].

Generally system management includes root cause analysis, anomaly and intrusion detection, failure prediction and diagnosis (e.g., loosely-coupled network of computer workstations in a department, large-scale clusters, and grid computing environments). We identify several challenges in managing large-scale computing systems:

- The heterogeneous nature of the computing system makes the management task more complex and complicated. A typical computing system contains different devices (e.g., routers, processors, and adapters) with different software components (e.g., operating systems, middleware frameworks, and user applications), possibly from different providers (e.g., Cisco, IBM, and Microsoft). The heterogeneity increases the likelihood of poorly understood interactions and unexpected interactions/dependencies.
- Large and complex computing systems usually show surprisingly unexpected behavior during failures, system perturbations, and even normal operations. The scale and complexity of these systems greatly surpass what can be understood by human operators. It is difficult for any one person to understand the system at the level of details necessary for system management.
- Current computing systems are dynamic and rapid changing with a growing number of software and hardware components. The fast rate of change worsens

system dependability and exacerbates the difficulty of understanding system behaviors.

- In large-scale computing systems, there is usually a gap between the low-level local view of individual components in the system (such as their states, resource usages, performance metrics, and other details) and the high-level global view of the system behaviors as a whole (such as end-to-end performance metrics, workloads and other aggregate information). Effective system management (e.g., fault detection and localization) needs to link the local and global views and bridge the gap.
- In a grid computing system, with the potential for thousands of resources at geographically different sites and tens of thousands of simultaneous users, real-time fault localization and problem diagnosis become significantly more challenging. As a result, more sophisticated tools are needed that can assist in performing system management by responding quickly and accordingly to the increasing volume of system measurements.

To cope with these challenges, the IBM Autonomic Computing (AC) initiative aims to build autonomic systems that are able to manage themselves with minimum human intervention [18]. To realize the goal of autonomic systems, the underlying assumption is the ability to define and maintain knowledge base and to adapt it with the ever-changing environment [24]. Modern computing systems are instrumented to generate huge amounts of log/trace data. The data in the log files indicate the status of each component and record system operational changes, such as the starting and stopping of services, detection of network applications, software configuration modifications, and software execution errors [27,32]. To enable self-management capabilities, a system needs to automatically characterize its behavior and acquire the needed knowledge from historical log data [31].

In this paper, we first present an integrated data-driven architecture for computing system management and then present a case study on Autonomic Network System Manager: a software system we developed that is currently being used by the system and network administrators of School of Computing and Information Sciences (SCIS) at Florida International University (FIU) (we often use FIU-SCIS for short). The rest of the paper is organized as follows: Section 2 provides an overview of some examples of system log data; Section 3 describes the integrated data-driven framework for autonomic computing system management; Section 4 presents the case study on Autonomic System Manager. In particular, Section 4.1 introduces the current system management practice at FIU-SCIS and summarizes the limitations, Section 4.2 describes the prototype implementation of Autonomic System Manager, Section 4.3 discusses applying data mining techniques on traffic feature distribution, and Section 4.4 presents several application scenarios of anomaly detection using Autonomic System Manager. Finally Section 5 concludes and discusses our future work.

2. Examples of System Log Data

Modern computing systems are instrumented to generate huge amounts of log or trace data. The data in the log files indicate the status of each component and record system operational changes, such as the starting and stopping of services, detection of network applications, software configuration modifications, and software execution errors. To enable self-management capabilities, a system needs to automatically characterize its behavior and acquire the needed knowledge from historical log data. In this section, we describe some examples of the system log data. The data can be collected from the distributed computing components including the network as follows.

- *Network traffic data*: Network traffic data measure the response and characteristics of the traffic. There are four types of network traffic monitoring data being widely used: SNMP link-level byte counts, Netflow link-level flow statistics, PMA link-level packet header traces, and tcpdump host-side packet traces.
- *Performance data*: Performance data report the performance observations of a component at some time intervals (e.g., CPU utilization of a component every five minutes). Typical performance metrics include CPU utilization, memory utilization, swap utilization, workload average, and response time. In particular, performance data are usually numeric.
- *Application-level log*: Application-level log records the application behaviors as well as the generated messages. Examples include Windows NT system and application log, WebSphere application log, Oracle activity log and Linux system log.
- *Failure data*: Failure data contain the system and application crash dumps as well as the error messages.
- *Reports from Operators*: Reports form operations are also referred as trouble ticket data and contain the problems and reports described by human operators. In addition, they may also contain possible causes and symptoms for failures.
- *Request data*: Request data reports the request (such as time, machine, user id and application) executed in the system. Examples include Apache and Microsoft IIS logs.
- *Other data*: Other examples of log data include network-based alert logs, program traces, and probes.

3. Design Overview

To realize the goal of autonomic system management, the underlying assumption is the ability to define and maintain knowledge base and to adapt it with the ever-changing environment. We propose an integrated data-driven framework for autonomic system management. The architecture of the proposed data-driven framework is summarized in Figure 1. The key components of the framework are listed as follows:

- *Log Data Management and Organization*: Network and host sensors, along with instrumented applications provide the ability to collect log data in the computing system. Log Adapter enables generic data collection from multiple heterogeneous data sources by converting individual records and events into the Common Base Event (CBE) format [7] and Situation Construction [13] standardizes the semantics of heterogeneous log data.
- *Real-time analysis*: Real-time analysis processes incoming events in real time and performs online operations/actions based on knowledge obtained from off-line analysis. Typical real-time analysis techniques include anomaly/intrusion detection, problem determination, fault diagnosis.
- *Offline Analysis*: Offline analysis derives and constructs knowledge bases (e.g., correlation and dependency knowledge) from historical data. Typical offline analysis techniques include temporal pattern discovery, correlation rule construction, summarization, and knowledge management.
- *Planning/Action:* Planning/Action includes administer notification, active data collection, sensor/actuator deployment or deactivation etc.

Rather than removing the human from the loop, the proposed data-driven integrated framework exploits the synergy of human expertise and automatic intelligent techniques and establishes the pioneering and practical solutions for system management. In addition, the proposed research and resulting framework are complementary to existing practices in the area of computing system management and can be easily incorporated into existing systems such as HP's Openview [9], IBM's Tivoli [17], and Microsoft's Operations Manager [10].

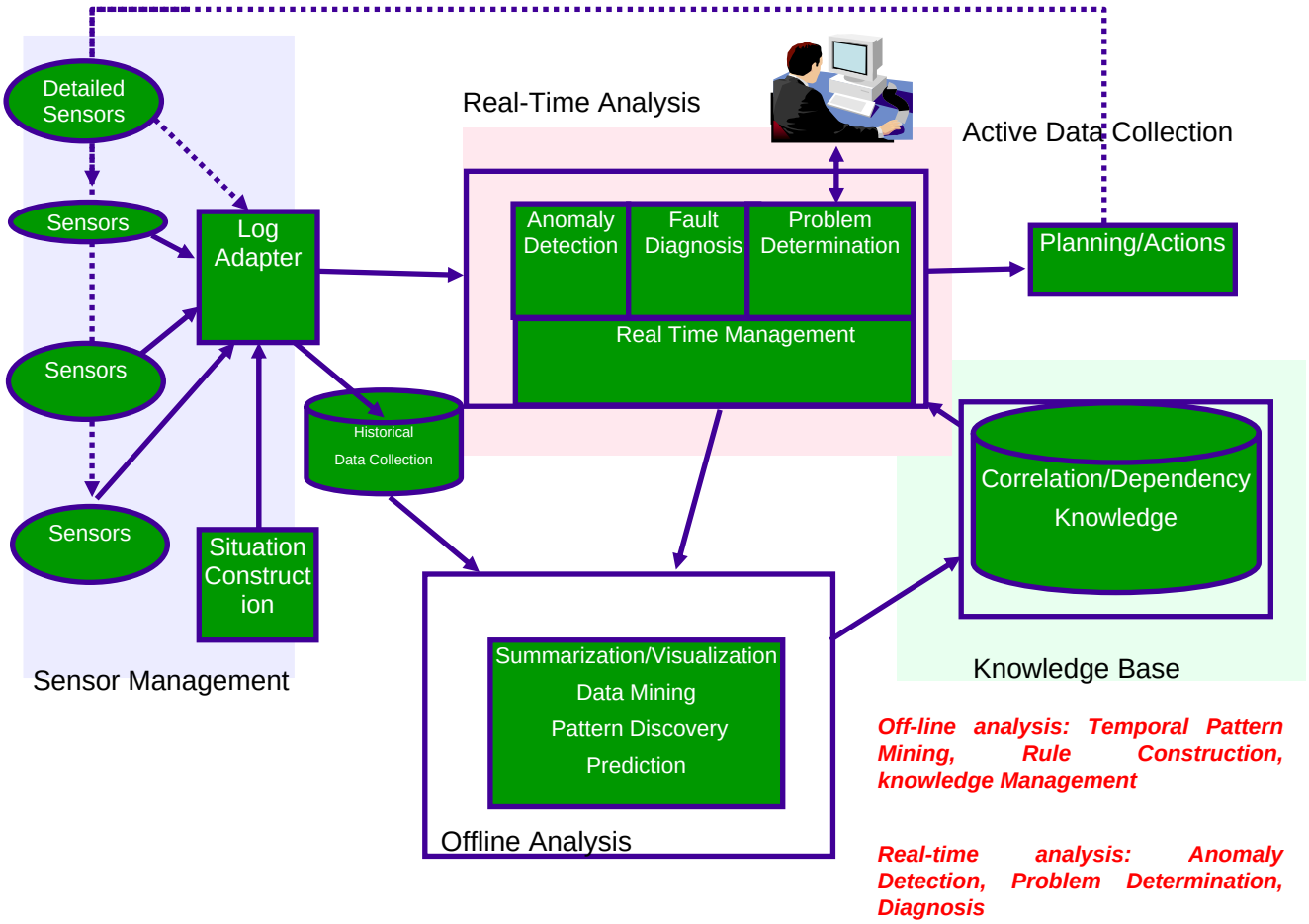

Figure 1: Architecture View of the Data-Driven Autonomic System Management

Other advantages of this framework are the following. First, this framework is complementary to the current knowledge-based approaches, which are based on the elicitation of knowledge from domain experts. Automated log data analysis can be performed without much domain knowledge and its results provide guidance for system managers to perform their jobs more effectively. Moreover, the available domain knowledge can be used to validate, improve, and refine data analysis. Second, the data-driven framework can benefit and extend the existing tools of system monitoring and analysis. Our architecture can interpret monitoring data of the same type but with different formats and contents, and convert them into a common form for further analysis. For example, we can capture the relationship between traffic anomalies and routing activities, i.e., routing policy changes or routing mis-configurations often causes abnormal traffics.

4. A Case Study on Autonomic System Manager at FIU-SCIS

In this section, we present a case study on Autonomic System Manager that we developed as a proof of concept in the School of Computing and Information Sciences at Florida International University. Autonomic System Manager is a software system to be used by system and network administrators. It automatically monitors Net flow data and responds to anomalies in an adaptive fashion. Net flow is short for "network flow", which is defined as a unidirectional sequence of packets between given source and destination endpoints. Network flows are highly granular; flow endpoints are identified both by IP address as well as by transport layer application port numbers. Net flow also utilizes the IP Protocol type, Type of Service (ToS) and the input interface identifier to uniquely identify flows.

The current focus of this prototype implementation is on anomaly detection from network traffic data. Anomaly detection is a major task for system management. The purpose of anomaly detection is to detect and categorize situations that deviate from the normal baselines. Network and system administrators receive a torrent of IP requests everyday. As is the case with any network, some events are malicious. In order to secure a network system effectively administrators need to use tools that have the capability to classify malicious activities and respond to threats in real time. The principle challenge in automatically detecting insidious intruders is that the range of anomalous activities is huge, ranging from port scans, denial of service attacks, ping of death, SYN flooding, UDP flooding, worms, viruses, and disconnecting models remotely.

4.1 System Practice at FIU-SCIS

As illustrated in Figure 2, FIU-SCIS hosts a variety of servers that includes Web servers, file servers, database server, and Netscape mail servers. All data is routed to the relevant servers first go through the Cisco Linux internal router, Saagwa (labeled S in Figure 2). A diagrammatic representation of the logical network map is given below. All of our analysis method and implementation were tested on this network. We first describe the current system management practice in the school of computer science and then summarize the limitations.

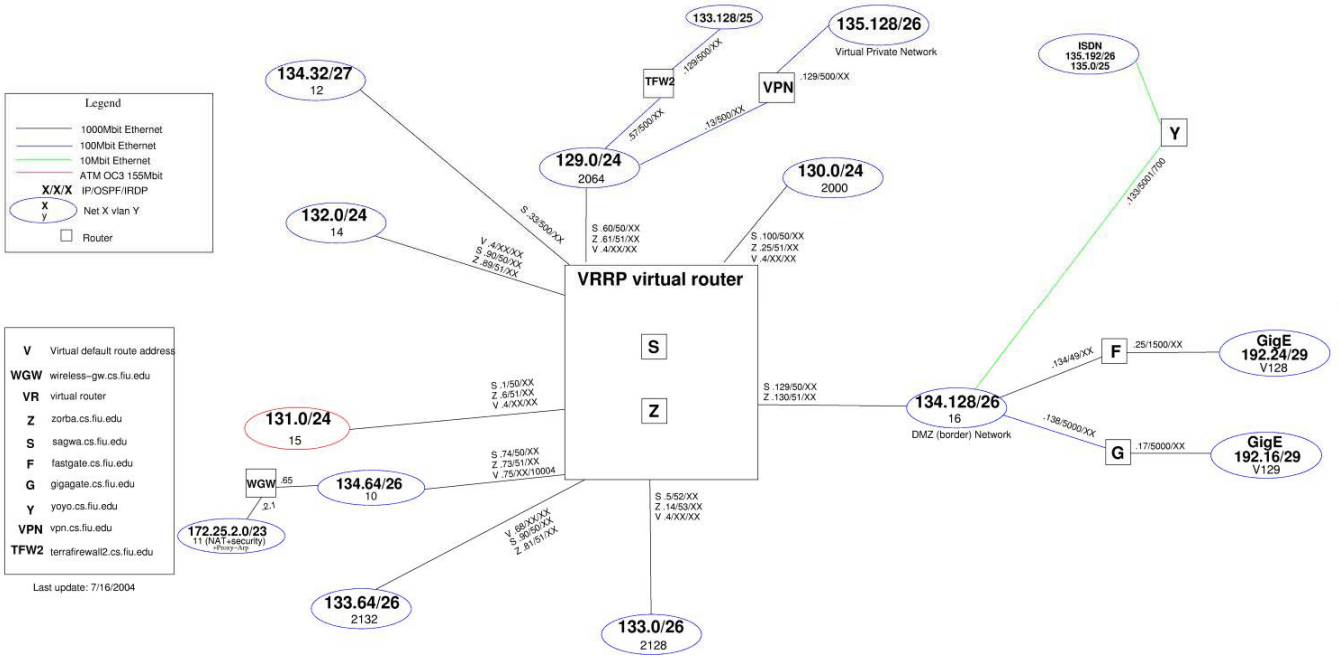

Figure 2: FIU-SCIS high-level network configuration.

Cisco IOS NetFlow efficiently provides a key set of services for IP applications, including network traffic accounting, usage-based network billing, network planning, and security. Current tools used by the network administrators are Fprobe and flow-tools. Fprobe is a libcap based tool that collects network traffic data and emit them as Net flow towards a specific collector. Fprobe is configurable as in all the IP requests (both incoming and outgoing) are together archived in files every 15 minutes.

Flow tools are software packages for collecting and processing Net flow data from Cisco routers. They are primarily used to sift through the data and to look for anomalies. Flows are exported from a router in a number of different configurable versions. A flow is a collection of key fields and additional data. Current statistic from tools (Fprobe and Flow-tools) used by the FIU's system and network administrators reveal the details of traffic flow such as the total octets, total packers, total time, duration of data, average flow time, average packet size, average flow per second etc.

Lysdexic, as shown in Figure 2, is a tap installed between the internet and the FIU-SCIS server that provides us with the raw data (traffic flow) based on commands executed available in flow tools. The tools currently used run on Linux platform and every action triggered is a function of an extensive command that has to be typed out on a command line interface of the Linux. The commands have to be explicitly typed out. Every action that has to be performed is command based. The output of the commands is not user friendly. There is a ton of data to deal with that is not sorted appropriately. Too much data has to be filtered using specific and tedious commands and or sorted with the 'eye'. There is no automated procedure for alert or action**.** Most importantly, such anomalies are revealed if and only if the administrator has the program running and use specific commands to probe and find out whether there is malicious activity.

To summarize, current system management practice in the School of Computing and Information Science at FIU has the following limitations:

- Most network and system management tasks are performed based on command line interface, which is usually not user-friendly.
- In addition, the output of the commands is not easy to interpret and understand.
- Most management tasks are largely based on the experience of the system administrators. In particular, the detection of anomalies is largely based on human manual effort and domain knowledge.
- Current management tools are not adaptive.

4.2 Prototype Implementation of Autonomic System Manager

We developed a prototype implementation of the Autonomic System Manager to help system and network administrators at FIU-SCIS with automatic monitoring of Net flow data and automatic response to anomalies in an adaptive fashion. The current focus of this prototype is on network anomaly detection. A screen shot of the system is shown in Figure 3.

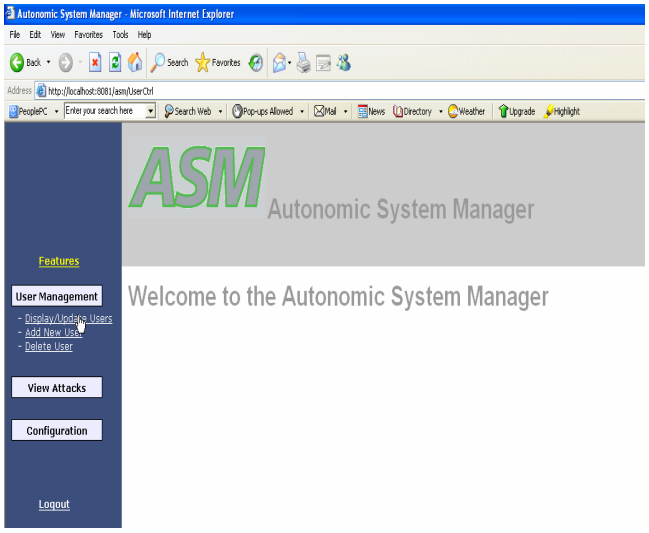

Figure 3: A screen dump of the ASM prototype implementation.

This prototype has a Web-based interface so that the administrators can easily access the information over the Internet. For security purposes, this Web application is password protected and only the users with enough privileges can access different functionalities provided in this system. We used MySQL database to store the data coming from network resource. This database is the core of the Autonomic System Manager, which gathers the data to be processed later. The rest of the system is developed in Java using the Servlet technology and is deployed on a Tomcat application server.

We have identified the core design goals of this system as follows:

- ***Usability***. The usage of Autonomic System Manager must be intuitive. The graphical user interface must be designed that the administrator understands the status without doing any other operation/lookup in the system. There must be no need for a help document. In other words, the interface and the display of graphics should be self explanatory.

- ***Robustness***. Since Autonomic System Manager deals with a dynamic flow of data from the network, it should be able to store and process those data whatever the size of the data is.

- ***Fault Tolerance.*** If some data is corrupted or in wrong format for some reason, Autonomic System Manager should discard that data, and process the rest of the data without noticing any delay or abnormal activity.

- ***Response time***. The Autonomic System Manager should be able to get the data from the data source, populate the database and display it in a fairly short acceptable time.

- ***Reliability***. Whatever the input to the system, the Autonomic System Manager should process and display the network status according to the parameters given by the system administrator as precise as possible.

This prototype explores the use of data mining and adaptive software techniques for anomaly detection in system management. The system aims to address those limitations in current system practice. We use data mining techniques to automatically detect abnormal Net flow traffic. Hence for performing real time classification of net flow behavior we collect various packet fields, composite features, and features constructed from multiple packets.

As described in the architecture shown in Figure 1, the major components of the Autonomic System Manager are described as follows:

- *Log Data Organization and Management*: The FIU-SCIS is currently averaging 5 million IP hits per day. We use a dedicated MYSQL database server for experimental purposes and route the network traffic into the database. A database is better suited for handling such vast amounts of data. Since most of the commands are embedded into SQL queries the results of these queries would be much faster rather than if the data were archived in a file.

- *Off-line Analysis*: In Autonomic System Manager, our focus is on anomaly detection. Usually the historical information is first used to establish a normal baseline for dynamic event behavior models via an off-line process [11,33]. Our initial works was to generate all the patterns of attacks found in the historical data and store those patterns in the Knowledge base of the system. These patterns would help us to detect and classify each of these attacks observed in traffic. In addition, a Web based interface is created for the system administrator. Since visual data is the easiest to grasp, a histogram will be used to the current flow of data in an aggregated format and when the administrator clicks on a link related to a particular bar and the administrator is taken to a page that displays the details relevant to that particular bar. The commands listed out in the annexure will be incorporated as command buttons that perform the specific tasks upon a simple mouse click.

- *Real-time Analysis*: In Autonomic System Manager, we perform real-time analysis based on the traffic distributions. The monitoring of the flow of data will be continuous (from the time the application is launched) and alerts (unlike the current system) are flagged immediately based on threshold values set by the administrator. Manual inspection of Netflow data archived in files is now negated and the system is to a greater degree, autonomic.

- *Planning/Action*: our current implementation provides the following functionalities: (1) Configure triggers: the system will allow the system administrator to define the criteria of a possible attack (e.g., over what percentage of usage on the protocols should an alarm should be sent); (2) Alert Admin: The administrator will be able to receive an email about any abnormal behavior that might have occurred in the system.

Upon developing a strong data analysis scheme it become as important to conceive of an adaptable software that would not involve manual configuration by administrator but one which would acting upon the information provided by the data analysis, update the servers / firewall automatically to block or take necessary action on attacks. We are in the process of developing a software system in lieu with the following properties in mind. The system needs to have a detailed knowledge of its components, current status, and ultimate capacity to other systems to govern itself. It will need to know the extent of its owned resources. The system should be able to dynamically sift through data. The nature of the software being built should be 'always on'. The system will perform it's tasks and adapt to the users needs without dragging the user into the intricacies of it's working.

4.3 Data Mining on Traffic Feature Distribution

In computing system management applications, the misclassification of anomalies and systems taking inappropriate action or even worse, missing the anomalies, is a huge problem. So in our Autonomic System Manager, we apply the data mining techniques on the traffic distribution. In fact, recently, there has been lots of research on using data mining and machine learning techniques for system research such as software bug detection, mis-configuration detection, Internet service management [1,2,5,6,8,29,30].

Analysis of traffic distribution is the most effective way to detect the anomalies and classifying the network anomalies [12,20,22]. The intuition behind this is that many important kinds of traffic anomalies cause change in the distribution of addresses or ports observed in traffic and this attacks have their distinct pattern which enable us to classify the type of anomalies in the network. Table 1 lists two common anomalies encountered in FIU-SCIS network.

Anomaly	Definition	Traffic Feature Distribution Affected
SSH Brute-Force	Connect to a remote machine and attempt to login using common user name/password combinations	Destination address, Source address, destination port 22

Attacks	on port 22 using SSH.	
Port Scan	Probes too many destination ports on one or many destination addresses.	Source address, Destination address, Destination port

Table 1: Two common anomalies in our network

The above discussed anomalies have a major affects on certain traffic distribution if seen for a specific period of time. In some cases, when source addresses are spoofed in attacks by an intelligent attacker, the traffic distribution becomes more dispersed and it attacker may go unidentified but still the destination address of the victim would remain the same and would reflect in our system as excess connection to a single destination address and hence we could detect such attacks but inherently we will not be able to locate the origin of the attack. In other cases traffic distribution becomes more concentrated on a small set of values as when a single source sends a large number of packets to a single destination in an unusually high number of connections to the FIU. A traffic feature is a field in the header packet. For our use we currently focus on four fields for doing analysis on traffic distribution: Source address, Destination address, Source port and Destination port.

The distribution of traffic features is a high dimensional object and so can be difficult to do proper analysis and come up with anomalies with fewer false alarms. However we can make the observation that in most cases by extracting useful information from the degree of dispersal or concentration of the distribution using reduction in their dimensionality by means of aggregation on certain factors to detect attack patterns. In some cases, the fact that the degree of dispersal of the destination port but the concentration of the destination address is a strong signature which should be useful for detecting the anomaly. We have implemented several data mining techniques include summarization, visualization and information-theoretic based methods in our Autonomic System Manager.

One thing to note is: the distinct separation among these types of known anomalies suggests that it may be possible to divide these set of anomalies in groups and trigger the software sensors. Thus our architecture provides a strong analysis framework upon which we can build an autonomic system which currently sends the report to the system administrator of any anomaly detected in the system.

4.4 Application Scenarios of Autonomic System Manager

4.4.1 Traffic Summarization

Our Autonomic System Manager provides many summarization and visualization functionalities such as:

1. Top Host/flow/link statistics**:** Provides an analysis of the IP layer traffic statistics collected for each host/flow/link in real time.
2. Flow Size Distribution: Estimates the distribution of flow sizes.
3. Bandwidth Utilization: Charts the number of packets per second and bytes per second flowing across the network in real time, among different applications, users, peering ISPs and etc.
4. Visualization: Displays summary information to visualize relationships among categorical attributes (e.g., IP addresses). It can also assist users in understanding and interpreting various patterns.

A screen shot of the summarization interface of our Autonomic System Manager is shown in Figure 4.

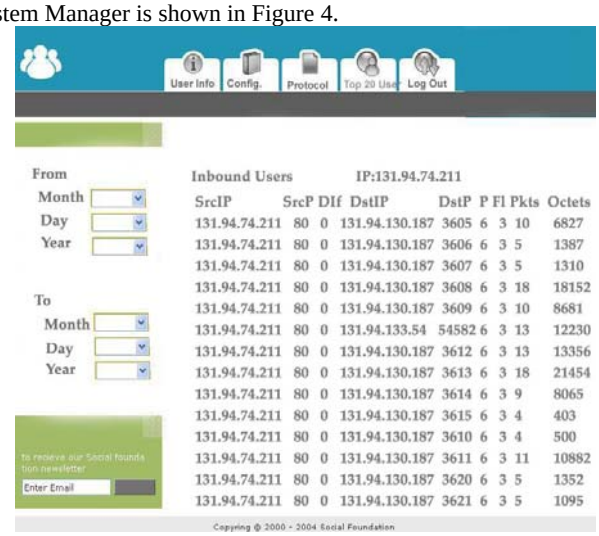

Figure 4: A screen dump of a detailed monitoring interface

4.4.2 SSH Brute Force Attack

A metric that we use to capture the degree of dispersal or concentration of distribution is sample entropy which has been very effective to detect SSH Brute-Force attacks. We depict the graph to represent the change in the traffic distribution during the SSH Brute-Force attacks using the Entropy to calculate the traffic distribution. Entropy formula used in the system to detect SSH Brute-Force attacks is as follows:

$$H(X) = -\sum_{i=1}^{N} \frac{n_i}{S} \log_2 \frac{n_i}{S}$$

where N is the number of top connections considered during the specific period, n_i is the number of connection made by an individual IP to connect to the FIU-SCIS network to log into the system, S is the total number of connections made on destination port 22 during that specific period. The entropy values takes a dip in the value of H(x) when distribution is maximally concentrated i.e., when an anomaly is detected in the system.

Herein we use sample entropy of traffic distribution that are constructed from number of connections made to the system. In the detection process this means that the anomalies showing unusual traffic volume will cause the entropy values to drop. This anomalies detected on the basis on change in traffic volume are also detected when there is a change in the entropy.

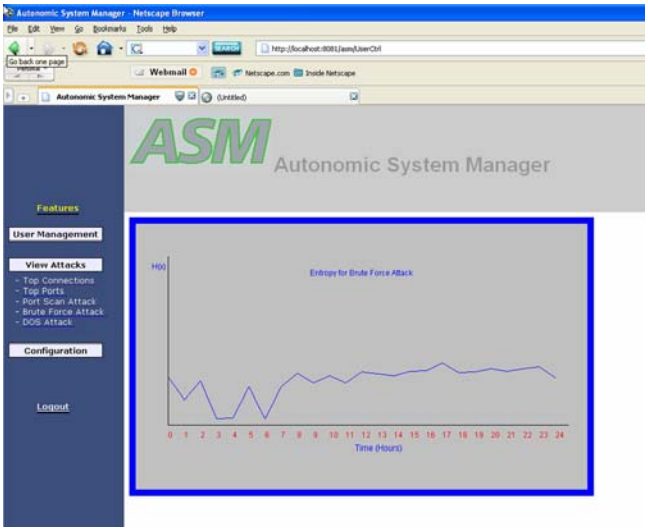

Figure 5: Entropy-Based Anomaly Detection.

Figure 5 shows an example on detecting the SSH Brute Force Attack using the entropy-based method. The entropy graph is obtained based on the traffic data on November 2, 2005. X-axis shows the time (in hours) during the day and Y-axis shows the entropy value. As you can see from the graph, the entropy value takes a dip between 3am to 4am. This can be verified from the data statistics from 3am to 4am on November 2, 2005, as shown in Table 2.

Source IP	**Dest_Port (22)**
140.122.105.156	37642
131.94.129.109	81
68.83.179.203	20
131.94.130.72	9
66.92.248.193	9
131.94.135.12	7
131.94.135.30	6

Table 2: Data statistics from 3 am to 4am on November 2nd, 2005

The statistics in Table 2 display the various source IP communicating with the FIU-SCIS network on the destination port 22, trying to login into the system. Whenever the attacker is trying to login into the system using the brute force attack the amount of connections the particular Source IP exceeds the total amount of connection of any other IP by a huge margin. Hence using the entropy based anomaly detection, the system could easily detect of any anomaly whenever there is a significant drop in the value of entropy calculated. One of our ongoing efforts is to use the statistical significance test to detect the entropy changes.

Entropy based anomaly detection is very sensitive hence it would be effectively working and detecting anomalies during the peak hours and also during the non-peak hours. This anomaly is reported to the system administrator as soon as such SSH Brute Force attack pattern is found in the system.

4.4.3 Port Scan Detection

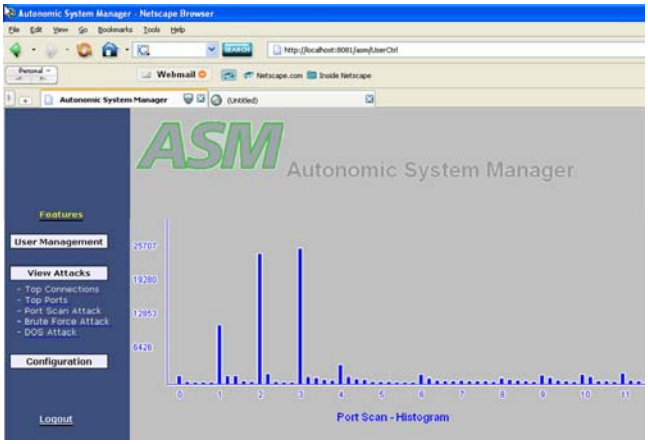

Figure 6: Port-scan Detection from 12:00am to 12:00pm on November 3rd, 2005.

Figure 6 shows an example for detection of the Port Scan done into the FIU-SCIS system to locate any vulnerable ports. The histogram is generated based on the traffic data on November 03, 2005. To further investigate the system behaviors, we take a closer look on the data statistics from 3am to 4am on November 3, 2005 as shown in Table 3.

Source IP	Src_port	Dest_IP	No. of Connections on distinct ports
216.76.30.66	80	131.94.129.76	25707
131.94.128.2	53	131.94.74.219	1333
130.207.108.135	20	131.94.130.174	1165
144.92.104.38	20	131.94.130.174	784
131.94.133.4	53	131.94.191.101	756

Table 3: Data statistics from 3am to 4am on November 3, 2005

The statistics in Table 3display the various source IP communicating with the FIU-SCIS network trying to communicate with the same destination IP on various distinct ports to find any vulnerable open ports on the destination IP targeted by the attacker. This graph displays the entire top five source's IP connected to the same destination IP on multiple

destination port for the duration of 1 hour. Clearly the graph indicates there was a dominant attack on the FIU-SCS network from the IP 216.76.30.66 on the machine 131.94.129.76 on the 25707 distinct destination ports to look out for any Open port on the machine so as to get the control of the system through that open port. This critical data is reported to the system administrator as soon as such port-scan pattern is found in the system.

5. Conclusion

Current computing environments are becoming increasingly complex with a growing number of heterogeneous software and hardware components and it is not a trivial task to provide the high performance, dependability, scalability and manageability in such computing environments. In this paper, we first provide an integrated data-driven architecture for computing system management and then present a case study on Autonomic Network System Manager: a software system we developed that is currently being used by the system and network administrators of School of Computing and Information Sciences (SCIS) at Florida International University (FIU).

We are currently working on enhancing our Autonomic System Managers from the following perspectives:

- More application tasks: our current system focuses on anomaly detection on traffic log data. We are currently implementing problem determination and diagnosis based on application-level log data.

- More advanced data mining techniques: we are currently implementing several advanced classification and clustering techniques for log data analysis.

- Integrated data analysis: system log data collected from different resources may have different characteristics at different granularities. Especially different types of data can complement each other. Analysts need to navigate freely between these levels in order to solve complex problems. The integration of types of data will facilitate cross validations. Furthermore, insights obtained at one level will be used to guide the selection and filtering of information at other levels. We will develop techniques to facilitate integrated data analysis for system management applications.

6. Acknowledgements

The work is partially supported by a 2005 IBM Faculty Award, a 2005 IBM Shared University Research (SUR) Award, and the National Science Foundation CAREER Award under grant no. NSF IIS-0546280. We would like to thank the systems and networking (SYS) group at the School of Computing and Information Sciences (SCIS) at Florida International University (FIU) for their valuable support. Special thanks to Eric Johnson for his constant feedback and involvement on this project.

7. References

[1] Marcos K. Aguilera, Jeffrey C. Mogul, Janet L. Wiener, Patrick Reynolds, and Athicha Muthitacharoen. Performance debugging for distributed systems of black boxes. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, 2003.

[2] Peter Bodik, Greg Friedman, Lukas Biewald, Helen Levine, George Candea, Kayur Patel, Gilman Tolle, Jon Hui, Armando Fox, Michael I. Jordan, and David Patterson. Combining visualization and statistical analysis to improve operator confidence and efficiency for failure detection and localization. In *Proceedings of The 2nd IEEE International Conference on Autonomic Computing (ICAC '05)*, 2005.

[3] A. Bouloutas, S. Calo, and A. Finkel. Alarm correlation and fault identification in communication networks. *IEEE Transactions on Communications*, 1994

[4] Aaron Brown, Gautam Kar, and Alexander Keller. An active approach to characterizing dynamic dependencies for problem determination. In *Proceedings of The Seventh IFIP/IEEE International Symposium on Integrated Network Management*, 2001.

[5] Mike Y. Chen, Emre Kiciman, Eugene Fratkin, Armando Fox, and Eric A. Brewer. Pinpoint: Problem determination in large, dynamic Internet services. In *Proceedings of 2002 International Conference on Dependable Systems and Networks (DSN 2002)*, pages 595–604, 2002.

[6] Mike Chen, Alice X. Zheng, Jim Lloyd, Michael I. Jordan, and Eric Brewer. Failure diagnosis using decision trees. In *Proceedings of International Conference on Autonomic Computing (ICAC-04)*, 2004

[7] M. Chessell. Specification: Common base event, 2003. http://www-106.ibm.com/developerworks/webservices/library/ws-cbe/.

[8] Ira Cohen, Moises Goldszmidt, Terence Kelly, Julie Symons, and Jeffrey S. Chase. Correlating instrumentation data to system states: a building block for automated diagnosis and control. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation(OSDI'04)*, pages 231–244, 2004.

[9] Hewlett-Packard Development Company. Hp openview management solutions for your adaptive enterprise. http://www.managementsoftware.hp.com/.

[10] Microsoft Corporation. Microsoft Operations Manager. http://www.microsoft.com/mom/default.mspx.

[11] Frank Feather, Dan Siewiorek, and Roy Maxion. Fault detection in an ethernet network using anomaly signature matching. In *SIGCOMM '93: Conference proceedings on Communications architectures, protocols and applications*,

pages 279–288, New York, NY, USA, 1993. ACM Press.

[12] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred. Statistical Approaches to DDoS Attack Detection and Response. DARPA Information Survivability Conference and Exposition (DISCEX), pages 303--314, April 2003.

[13] Genady Grabarnik, Abdi Salahshour, Balan Subramanian, and Sheng Ma. Generic adapter logging toolkit. In *Proceedings of First IEEE International Conference on Autonomic Computing(ICAC-04)*, pages 308–309, 2004.

[14] B. Gruschke. A new approach for event correlation based on dependency graphs. In *Proceedings of the 5th Workshop of the OpenView University Association: OVUA'98*, 1998.

[15] Cynthia S. Hood and Chuanyi Ji. Proactive network fault detection. In *INFOCOM '97: Proceedings of the INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution*, page 1147, Washington, DC, USA, 1997. IEEE Computer Society.

[16] Paul Horn. Automatic computing: IBM's prospective on the state of information technology. http://www.research.ibm.com/autonomic, 2001. IBM Corporation.

[17] International Business Machines Corp (IBM). IBM Tivoli. http://www-306.ibm.com/software/tivoli/.

[18] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer*, pages 41–50, 2003.

[19] S. Kliger, S. Yemini, Y. Yemini, D. Ohsie, and S. Stolfo. A coding approach to event correlation. In *Intelligent Network Management (IM)*, 1997.

[20] Anukool Lakhina, Mark Crovella and Christophe Diot. Mining Anomalies Using Traffic Feature Distributions. SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications, pages 217-228, 2005

[21] Wenke Lee, Sal Stolfo, and Kui Mok. A data mining framework for building intrusion detection models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, 1999.

[22] W. Lee and D. Xiang. Information-Theoretic Measures for Anomaly Detection. In IEEE Symposium on Security and Privacy, Oakland, CA, May 2001.

[23] L. Lewis. A case-based reasoning approach to the resolution of faults in communications networks. In *Intelligent Network Management (IM)*, 1993.

[24] Tao Li, Feng Liang, Sheng Ma, and Wei Peng. An Integrated Framework on Mining Logs Files for Computing System Management. In Proceedings of The Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (*SIGKDD 2005*)

[25] IBM Market Research. Autonomic computing core technology study, 2003.

[26] J. Pei, S. J. Upadhyaya, F. Farooq, and V. Govindaraju. Data Mining for Intrusion Detection - Techniques, Applications and Systems. In ICDE Tutorial, 2004.

[27] Wei Peng, Tao Li, and Sheng Ma. Mining Log Files for Data-Driven System Management. SIGKDD Explorations, Volume 7, issue 1, pages 44-51, June 2005.

[28] I. Rouvellou and G. W. Hart. Automatic alarm correlation for fault identification. In *INFOCOM '95: Proceedings of the Fourteenth Annual Joint Conference of the IEEE Computer and Communication Societies (Vol. 2)-Volume*, page 553, Washington, DC, USA, 1995. IEEE Computer Society.

[29] Ramendra K. Sahoo, A. Oliner, Irina Rish, Manish Gupta, Jose E. Moreia, Sheng Ma, Ricardo Vilta, and Anand Sivasubramaniam. Critical event prediction for proactive management in large-scale computer clusters. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD'03)*, pages 426–435, 2003.

[30] R.K. Sahoo, M.S. Squillante, A. Sivasubramaniam, and Yanyong Zhang. Failure data analysis of a large-scale heterogeneous server environment. In *Proceedings of the 2004 International Conference on Dependable Systems and Networks*, pages 772–781, 2004.

[31] Brad Topol, David Ogle, Donna Pierson, Jim Thoenscn, John Sweitzer, Marie Chow, Mary Ann Hoffmann, Pamela Durham, Ric Telford, Sulabha Sheth, and Thomas Studwell. Automating problem determination: A first step toward self-healing computing systems. IBM White Paper, October 2003.

[32] Wei Xu, Peter Bodik, and David Patterson. A flexible architecuture for statistical learning and data mining from system log streams. In *Proceedings of ICDM 2004 Temporal Data Mining Workshop*, 2004.

[33] Nong Ye, Syed Masum Emran, Qiang Chen, and Sean Vilbert. Multivariate statistical analysis of audit trails for host-based intrusion detection. *IEEE Transactions on Computers*, 51(7):810–820, 2002.

[34] S. A. Yemini, S. Kliger, E. Mozes, Y. Yemini, and D. Ohsie. High speed and robust event correlation. *IEEE* Communications Magazine, 34(5):82–90, 1996