Experiences with a Continuous Network Tracing Infrastructure *

Alefiya Hussain[†‡] Genevieve Bartlett[†] Yuri Pryadkin[†]
John Heidemann[†] Christos Papadopoulos[†] Joseph Bannister[†]

† USC/Information Sciences Institute
4676 Admirality Way
Marina del Rey, CA, USA 90292

‡ Sparta Inc
3415 S. Sepulveda Blvd, Suite 800
Los Angeles, CA, USA 90034

alefiya.hussain@sparta.com, {gbartlet,yuri,johnh,christos,joseph}@isi.edu

ABSTRACT

One of the most pressing problems in network research is the lack of long-term trace data from ISPs. The Internet carries an enormous volume and variety of data; mining this data can provide valuable insight into the design and development of new protocols and applications. Although capture cards for high-speed links exist today, actually making the network traffic *available* for analysis involves more than just getting the packets off the wire, but also handling large and variable traffic loads, sanitizing and anonymizing the data, and coordinating access by multiple users. In this paper we discuss the requirements, challenges, and design of an effective traffic monitoring infrastructure for network research. We describe our experience in deploying and maintaining a multi-user system for continuous trace collection at a large regional ISP. We evaluate the performance of our system and show that it can support sustained collection and processing rates of over 160–300Mbits/s.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network monitoring*

General Terms

Measurement, Design

Keywords

Continuous trace collection, trace infrastructure

*This material is based on work partially supported by the United States Department of Homeland Security contract number NBCHC040137 (”LANDER”). All conclusions of this work are those of the authors and do not necessarily reflect the views of DHS.
Alefiya Hussain’s work was done primarily while she was at USC/Information Sciences Institute.

SIGCOMM’05 Workshops, August 22–26, 2005, Philadelphia, PA, USA.

1. INTRODUCTION

As the Internet continues to grow rapidly in size and complexity, it has become increasingly clear that its evolution is closely tied to detailed understanding of its traffic. The enormous volume and mixture of data carried over the Internet provides a rich complexity. This richness has made it difficult to characterize, understand, and model Internet traffic from small-scale laboratory or testbed experiments, and has made it challenging to design simulation models.

Traffic monitoring and measurements can provide an invaluable insight into the dynamics of network traffic, traffic engineering and capacity planning, congestion and fault diagnosis, and security analysis. However, monitoring traffic on the Internet is a challenging task for a wide range of reasons. Many network events that are of interest to the researcher are unpredictable and ephemeral: if tracing is not enabled at the time when they occur, the opportunity of capturing the event is lost. Even when captured, following causality of such an event may be impossible unless enough history, that is network dynamics prior to the event, is captured as well. Privacy and confidentiality concerns limit the willingness of many stakeholders to participate in data collection and constrain the release of data to a wider research community. At times, the data is subjected to extensive anonymization rendering it unusable to a wide variety of network studies. Further, since the Internet is organized as a set of interconnected networks, measurements taken at any local network may not be representative of the whole Internet.

The resulting paucity of representative data has severely limited network research. Currently network dynamics studies that evaluate the effects of new application and protocol deployment, or long-term studies that observe the effect of incremental changes on the Internet, and the change in the overall stability of the Internet under various security threats are not possible. For example, in the past few years, the research community has explored many defense tools and methodologies to make the infrastructure more robust [9]. But lacking real-world data, testing the efficacy of the tools and evaluating their readiness for widespread deployment has been a continuing challenge.

To analyze a large range of network behaviors, one would need to collect network traffic on an ongoing basis rather than as a one-time event to capture transient behaviors that provide insight into network problems. Long term storage and mining of such measurement data will provide a valuable step forward toward improving the understanding of actual network dynamics.

In this paper, we describe a *passive* and *continuous* monitoring infrastructure to collect, archive, and analyze network traffic. *Passive* indicates the non-intrusive nature of the operational monitoring

system that does not interfere with the network traffic. *Continuous* refers to the ability of the collection system to capture network traces uninterrupted for analysis and storage. Further, network traffic is stored and analyzed at *different levels of anonymization* to accommodate user research requirements and privacy restrictions of network data. Our system can provide network traces for a large range of research projects that utilize long term data for the analysis and modeling of Internet traffic. In Section 3 we explore the requirements and challenges of building such a monitoring system and in Section 4 we discuss our design choices in order to address these requirements.

The contribution of this paper is the design, development, and deployment of a monitoring and analysis infrastructure for data mining that supports long term network traffic storage. We describe our experience in deploying and maintaining a system for continuous trace collection at Los Nettos, a large regional ISP. We evaluate the performance of our system in Section 6 and show that it can support sustained collection and processing rates of over 160Mbits/s with peak rates of 300Mbits/s. while accommodating the requirements of different users. Additionally, we demonstrate how our analysis system supports multiple users and permits the extraction of various types of analysis data from the network traffic.

2. RELATED WORK

Popular passive measurement systems include Simple Network Management Protocol (SNMP)-based network traffic measurement tools, tcpdump and NetFlow. SNMP is the most widely used network management protocol in today's Internet [1]. Agents and remote monitors update a management information base (MIB) within network routers, and management stations retrieve MIB information from the routers using UDP. While, SNMP provides some aggregate statistics, it does not permit detailed network dynamics studies which is the aim of the system proposed in this paper. Another common network monitoring tool is *tcpdump* [7], a network sniffer with built-in filtering capabilities. However, in high–volume traffic monitoring environments such as in ours, tcpdump performs poorly due to inaccurate timestamps and packet drops during large bursts. NetFlow is a monitoring system available on Cisco routers. NetFlow collects flow statistics on a network interface [11] and provides additional details about flows than SNMP. NetFlow requires an external recorder of captured data, it incurs a significant performance hit on routers (and thus is typically used on high-end routers) and in order to keep up with line rate it must switch to sampling.

Few other efforts have been made in capturing network behavior and standardization of metrics. The AT&T trace collection project relies on packet-level information collected by packet sniffers called PacketScopes, flow statistics collected using Cisco's NetFlow tools, and routing information for network-wide traffic analysis [2]. Similarly, the IPMON network designed by Sprint uses a large number of of trace collection points emulate a fault and capture the effect of the network [5]. In both these projects, the network traffic data is for the exclusive use of the parent company and the data is not available to the community at large for research. The monitoring system described in this paper is unique in that it facilitates continuous, passive trace collection and the data is available to the research community through the PREDICT project.

3. CHALLENGES

The design of an infrastructure to monitor, analyze and make Internet data publicly available to research organizations presents both policy and engineering challenges. In this section we outline these challenges and requirements for such a system and, in the next section we discuss how effective design choices in the architecture of the system can resolve most of these challenges.

3.1 Policy Challenges

There are a number of policy challenges which stem from the requirements of developing a *multiple users* system with support for *long-term* network studies that may require *neighbor identification* information. In this section, we discuss each of these requirements and the challenges associated in incorporating them into a monitoring system. Finally, we discuss the strategy of associating *meta-data* with network measurements.

Multiple Users: In order to support many different research projects, the system must seamlessly support multiple users, which presents several challenges. First, since users typically have different data requirements, each user will require *custom processing* of the data to extract information relevant to their research. For example, a user interested in studying long-term flow behavior of the network will need more complex processing to correlate network traces while a user interested in DNS traffic can use simple packet filters to extract DNS packets. Additionally, some types of network studies are more sensitive to packet sanitization and anonymization than others. For example, techniques that retain only TCP/IP headers, renumber IP addresses and scrub packet payloads are especially crippling for network intrusion detection research and worm epidemiology studies, which require network prefix information to analyze infection patterns. Thus, the monitoring system needs to support a range of custom anonymization levels such that important traffic characteristics are not abstracted away.

Given different privacy restrictions and project requirements, each user will have different access privileges which map to data with different anonymization levels. Some research projects such as projects involving network security may still require access to *raw*, un-anonymized traffic, while other projects can tolerate anonymization of most network traffic. Pang and Paxon describe techniques to anonymize and transform network traffic in order preserve privacy while retaining important traffic characteristics [10].

Finally, the system should have all the necessary access control and job isolation mechanisms to ensure data security. Since network data will be available at various levels of anonymization, these mechanisms will prevent unauthorized access to network data.

Long-term Studies: Some users will want to study long-term characteristics of the network—inconsistent anonymization will prevent such a study. The monitoring system needs to provide selective support for consistent IP address scrambling over extended periods of time. This is especially useful for projects such as traffic engineering, long-term network forecasting, and traffic evolution analysis. The utility of renumbering consistency must be balanced against the privacy concerns in inverting mappings.

Neighbor Identification: Neighbor Identification information provides the last hop information of the network traffic at the monitoring location. For example, at the Internet backbone it would provide last hop peering information indicating which peer originated the traffic. At an ISP, it would provide directionality information that would identify network traffic originating and terminating at the ISP clients. Retaining neighbor identification information is an additional requirement of the system and is especially important in the presence of completely anonymized traffic. It provides important clues for traffic characterization projects.

Meta-data: In order to preserve information about the network traffic during analysis, we need to associate meta–data with each

network traffic measurement. The meta–data can provide basic information regarding the measurement, such as record time, location of measurement, anonymization level, neighbor identity, duration, and packet loss information. In the long run, we anticipate annotating the traces with user–generated meta–data, perhaps describing strange network occurrences, security events, or other traffic features of note.

3.2 Engineering Challenges

Conducting an Internet measurement study is a difficult task since a few minutes of network traffic can contain more than a million datapoints. This is even more challenging when designing an online, continuous monitoring system. The challenges of such a system are primarily *high volume I/O*, *online computation*, *bulk storage*, *reliability and fault tolerance*. In this section, we identify and elaborate on these engineering requirements and challenges.

High Volume I/O: A major problem when monitoring links at high bandwidths is disk I/O. Links of 1Gbits/s capacity are fairly common today between large ISPs, with 10Gbits/s links rapidly gaining ground. Under normal operational conditions, links are utilized at no more than 60% on average. In our network we have observed up to 600Mbits/s or 200Kpkts/s bidirectional traffic on each link. For both performance and privacy concerns we capture only the first 64 bytes of data of each packet. Though on average the captured data stream is about 100Mbits/s, during a large DDoS attack with small packets it is possible that our system will be called to capture at the entire link capacity! Moreover, this estimate is for just one monitoring point. Most networks have multiple connections; in our network, for example, we have four such links we need to monitor. The resulting high disk I/O due to the potentially large volume of network traffic puts a significant strain on the design of the monitoring infrastructure. These network I/O rates are similar to current disk specifications, thus any protocol overhead must be avoided.

Online Computation: In addition to the disk I/O requirement, in order to support the multi-user environment, we need to support custom anonymization and user analysis of network traffic at line rate. This is challenging because different user anaylsis scripts may have different run times, both because the underlying algorithms may differ, and because users' interests may be piqued by different kinds of data. We must therefore accommodate large and variable demands for computation.

Bulk Storage: At normal operational rates capturing 64B of partial packets on each 1Gbits/s link can generate up to 46GB of data every hour. In order to support this high volume of data, we need a large and fast bulk storage system to store the data at line rates from multiple links and support the data analysis requirements of multiple users. Additionally, we need to ensure the storage space is administered effectively to optimize system performance even during mundane tasks such as directory listing and searching and keeping track of per–user disk usage.

Reliability and Fault Tolerance: The system needs to be reliable, robust against failures, and allow remote administration to reduce the need for physical human intervention. Further, to ensure data reliability, network traffic that is identified as useful to a project must be backed-up regularly.

4. ARCHITECTURE

Our goal is to continuously collect, analyze, and archive network data for data mining applications. Three components are used to meet this goal: A set of *collection hosts* (Section 4.1) that capture network packet traces, which are transfered to a *data repository* (Section 4.2), and a *processing cluster* (Section 4.3) that supports pre-processing and per-user processing of the traces.

Figure 1 briefly outlines the interaction of the collection, storage, and analysis phases of the monitoring system. In this section we describe the physical configuration of the various architectural components of our monitoring system and how they assist in meeting the requirements and resolving these challenges. We conclude with the anonymization challenges created by different user requirements.

4.1 Data Collection

The trace collection hosts are responsible for collecting packet traces from the network. In this section we discuss the physical configuration of our system and how it helps meet our high traffic volume requirements. Although some of these techniques have been used by previous trace collection systems [5], we discuss the unique aspects of the system and propose strategies to handle RAID fail-over conditions.

To minimize packet loss at high traffic volumes, we require dedicated, high-performance collection hosts. Our monitoring system employs Red Hat Enterprise Linux machines with dual 2.0GHz Xeon processors, 4GB RAM, 240GB local disk and specialized DAG 4.3GE cards to capture network traffic. Optical splitters are deployed on each monitored link to mirror the traffic to the DAG card. The DAG cards generate a GPS synchronized timestamp for each packet when the beginning of the packet is detected on the link.

The most pressing engineering challenge in this system is the requirement to handle large volumes of data I/O and processing continuously. To reduce the volume, we configure the DAG cards to extract the first 64 bytes of each packet and record it into memory. Additionally, we segment traces into pieces for processing. Two approaches to segmenting are *fixed-time* or *fixed-space*, where we break traces into some time period (say, 5 minutes) or some size (say, 512MB). We adopt the fixed-space approach, segmenting data into 512MB files. Since traffic rates vary throughout the day, this approach provides predictable per-segment I/O and processing requirements. Our data processing approach (described in Section 4.3) easily accommodates the varying segment generation rate.

Each DAG card DMA writes captured packet headers into a circular buffer located in shared memory. In order to avoid packet loss the captured trace needs to be moved out of the buffer before it fills. Our *DAG talker* process is responsible for this task. It moves the data out of the buffer and saves it to disk. If the 512MB file size limit is reached, the current trace file is closed and a new one is started. Although, it is possible to transfer the packet headers directly from the card to the data storage, bypassing main memory, our approach helps buffer traffic bursts and reduces packet losses during data transfer.

The data storage location is a fast RAID server accessible via NFS. The RAID may become unavailable to the collection system due to network failures, NFS server failure, or disk failures. In order to support continuous collection, we provide fail-over to a local disk when the RAID server is unavailable. We developed a *DAG mover* program that periodically monitors the availability of the RAID server and triggers data storage on the local disk if the RAID is unreachable.

4.2 Data Repository

The data repository is a DataDirect high-performance RAID networked storage system rated at 1.5Gbytes/s I/O throughput. Cur-

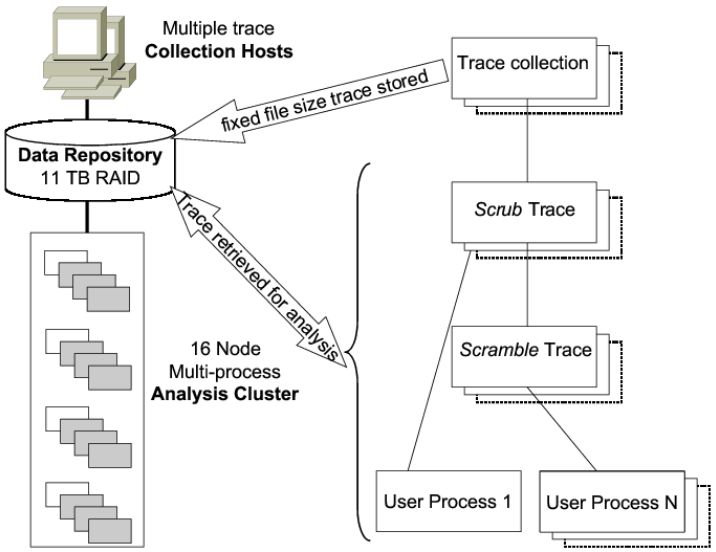


Figure 1: Interaction between various components of the monitoring system

rently, our system accesses storage through NFS over an Ethernet gigabit link. The NFS server adds considerable overhead to our system. After performing extensive client and server side tuning by varying blocksize, timeout, and mounting parameters we were able to improve the throughput from 6Mbits/s to 300Mbits/s. In the near future we will connect the data repository directly to the collection and analysis system using fiber channel. Once the whole system is connected using fiber channel we anticipate throughput in excess of 1Gbits/s.

Our system has 11TB of shared disk space. Since each monitoring location generates over 1 TB of data every day, we need mechanisms that ensure proper use of disk space. We have implemented a *garbage collection* system that expunges old trace files that have been identified as unimportant by all users. The garbage collector cleans up files periodically several times a day or when disk space utilization exceeds 70%. We also have a *quota checker* system that regularly audits usage of the shared disk space among the users. Finally, traces that have been identified as important by the users are backed up periodically onto tape.

In order to preserve information of the network traffic we need to associate *meta-data* with each network traffic measurement. There are several options in associating meta-data with a trace. One technique would be to store meta-data in a separate file, which is often convenient since it facilitates subsequent processing. However, it requires additional separate storage and is prone to loss or disassociation with the primary trace file. Another technique is to encode meta-data information within the trace filename and in the same file. While this technique is more space efficient, it binds the meta-data to the trace and as a result the meta-data is lost when a trace is deleted, making it impossible to revisit the meta-data at a later time.

Various types of meta-data can be recorded with a traces. Some may provide basic information regarding the measurement, such as record time, location of measurement, or duration, while others can be more detailed and could be obtained after post-processing, such as packet loss, number of flows, and protocol mix. Currently, we store meta-data only for files identified as important and encode it within the file to reduce storage space. In the future, we plan to explore archiving meta-data for all network traffic in separate logs.

4.3 Processing Cluster

All data analysis is performed at the USC Center for High Performance Computing and Communications (HPCC) [12]. We have reserved a 16-node partition for exclusive use for our system in order to guarantee service. All analysis processes share HPCC's common Portable Batch System (PBS) queue [6] and our cluster consists of Linux nodes with dual 3GHz Xeon processors and 1GB of memory.

The monitoring system needs to process a large amount of data online and is limited by disk I/O. As discussed in Section 3.2, we maintain *fixed-space* 512MB trace files in order to better manage the I/O bandwidth and processing requirements.

There are two categories of analysis performed on the network traffic:

- Each trace is *pre-processed* to address privacy and security concerns. Pre-processing of the traces consists of *scrubbing* and *scrambling* of the packet headers. Additional details on anonymization techniques are discussed in Section 5.

- A trace can then be analyzed by multiple users by a *per-user process*. The user process is defined independently by each user for each trace.

All the processing on the cluster is done with *dispatcher-worker* applications in order to keep up with the online computational requirements of our system, similar to other systems [4]. The dispatcher application periodically monitors the workload and active processing capability by queuing worker processes through the HPCC batch processing system. When work is available it starts additional processing jobs, up to a configurable maximum level of parallelism.

Workers run independently, processing available trace files and terminating when all files are processed or a time limit is reached.

Traces are kept in different directories corresponding to their processing status (scrubbed, scrambled, user analyzed).

The pre-processing and per-user processing systems can have multiple worker processes analyzing the files at any time, limited only by the number of processors available in the cluster. Once a trace is pre-processed, the system creates hard-links in the user space based on the user's data security clearance (explained in Section 5. We found that the loose coupling between pre-process and per-user process phases using dispatcher-worker applications helps deal with the high volume traffic efficiently and isolates failures, thus enhancing the reliability and fault tolerance of the analysis system. We also have an extensive monitoring system that gathers statistics for deployed worker processes and un-processed files and alerts the operator in case of potential problems.

Our system must be robust to processing errors for several reasons. First, if a trace is not completely scrubbed or scrambled, it would leak sensitive privacy information. Second, because we allow user jobs to post-process traces we cannot assume those jobs are error-free. We therefore consider all processing suspect and examine a number of possible failure scenarios: (1) the job is not run, (2) the job runs and stops in the middle with part of the trace processed, (3) the job runs multiple times sequentially, (4) the job runs multiple times concurrently. To handle a wide range of errors, we have designed our jobs to be *idempotent*, that is, they can run multiple times without any ill-effect. Therefore if the job monitoring system detects conditions (1) and (2) it restarts the job on the trace. In case of repeated failure in processing the trace, the trace is marked and administrator is alerted of the error. Failure condition (3) is less critical as it results in wasted computational power but the trace is processed correctly. In this case we alert the administrator in case of processing the same trace more than twice. Failure condition (4) is the most critical as the trace may be overwritten and not processed correctly. We have taken all measures to ensure that this condition does not occur in our system by moving files to an working directory by the worker process. If such as error does occur, the job monitoring system is designed to alert the administrator at once.

5. CUSTOM ANONYMIZATION

Privacy and security issues are compounded in a multi-user environment. Sharing of raw Internet packet traces must be restricted, since traces contain sensitive user data and control information. Clearly user data must treated carefully since it may contain e-mail, or passwords, credit card, or other private information. Control information is a concern since when combined with external information such as DHCP logs and ISP subscriber databases, IP addresses can be mapped back to individuals' identities. A range of privacy laws govern network trace data, ranging from wiretap laws to student confidentiality requirements. As a result, ISPs must be very careful with how trace data is managed, both from an ethical and a legal standpoint.

Unfortunately, current trace anonymization techniques that eliminate all private information severely limit research. Due to the privacy issues with raw traces we propose a multi-level *data security clearance* scheme that allows the user to access real-world traces based on research needs. We store real-world network traces at different levels of anonymization and control access to the traces based on the users' data security clearance. Each trace is subjected to a *scrub* phase, that removes the packet payload information to ensure that user data is not directly leaked. A trace is then passed through a *scramble* phase, that remaps IP addresses to prefix-preserving random values [8]. Additionally, we record the host IP to anonymized address mappings in order to support *custom anonymization*, where a user may want to know the reverse mapping for a few hosts that are under the control of the user, and therefore do not have any privacy concerns.

Each data security clearance level is associated with a set of *data movement rules*, that specify how the traces should be handled. The higher the clearance the more restrictive the the data movement rules. For example, users with access to un-anonymized traces have to perform all analysis on our systems and cannot move the traces away. We discussed this in more detail in Sections 4.2 and 4.3.

In order to support long term studies, we maintain consistent IP address renumbering using hashing [3]. We refer to this as *persistence* of the scrambled traffic. Additionally, it is also important to maintain neighbor identification or *directionality* information within the traces. This is done during the scramble phase where the MAC addresses in the packet are renumbered with a persistence of one month.

6. EVALUATION

We deployed our collection system at two locations at a large regional ISP. In this section, we first describe our monitoring location and the type of traffic mix we observe. We then evaluate our analysis system in a multi-user environment.

6.1 Monitored Network Description

We collect network traffic through a coordinated effort between multiple sites within Los Nettos, a large regional ISP in the Los Angeles area. Los Nettos connects with several large ISPs including Verio, Cogent, Level 3, and Internet II. Los Nettos provides an interesting view of the Internet since it is close enough to the edge to observe regional and end system effects, but also large enough to capture a diverse traffic mix. Los Nettos provides access to several different classes of network traffic. It has a diverse group of academic and commercial clientele, including USC, TRW, JPL, and Aerospace Corporation.

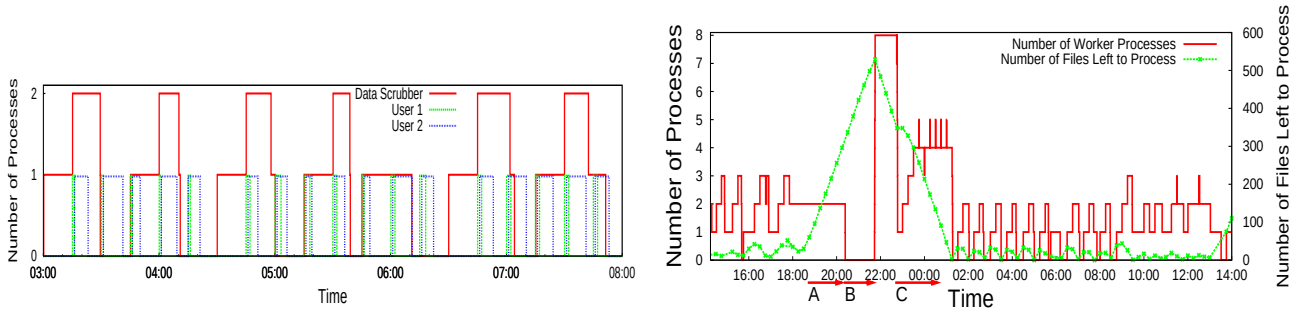
Los Nettos has an aggregate bidirectional traffic volume of over 1Gbits/s. We have strategically placed two tracing hosts that allow our system to capture up to 85% of the network traffic. One collection host monitors a link with 600Mbits/s aggregate bidirectional traffic while the other host monitors a link with 250Mbits/s. After partial packet capture the links generate nearly 160Mbits/s of packet header traces.

6.2 Regular Operation

Our system handles multiple users with different data and sanitization requirements. During peak times a trace file is generated every 30 seconds and the analysis cluster pre-processes and runs per-user processing on these files continuously.

Figure 2(a) shows job duration for a five-hour period. A scrubber-dispatcher schedules jobs every fifteen minutes to pre-process new trace files, and two user-dispatchers spawn jobs to run user-defined processes. User1 handles un-anonymized data and uses anomaly detection techniques to detect denial of service attacks. User2 characterizes P2P traffic and extracts all TCP packets with the SYN or FIN flags set. The scrubbing process keeps up with traces using one worker job but approximately every 45 minutes two jobs are spawned to clear the backlog. User1's worker jobs typically process all files within two minutes and User2's worker jobs run between 5–10 minutes. Thus all processing, scrubbing and two per-user workers, are handled efficiently in real-time by the analysis system.

This demonstrates our ability to handle different user requirements. Since jobs are loosely scheduled by the cluster system we can handle varying computational requirements of different users



(a) Normal Conditions

(b) Failure Conditions

Figure 2: File processing on the cluster under normal and failure operational conditions.

without carefully monitoring and tuning our system to adjust to changes in load or post-processing. In addition, the parallel processing capability of the cluster allows the trace processing to run slower than real-time without building large backlogs.

6.3 Failure Recovery

An important aspect of our loosely coupled architecture is that it is robust in the face of various failures. Figure 2(b) shows the progress of pre-processing data files before, during and after an infrastructure failure. At approximately 18:00 (labeled as time period A) the queuing system stopped accepting jobs and the number of files in need of processing grew. During time period B there was a complete cluster failure; all processing jobs still running were terminated. The job monitoring system discussed in Section 4.3 handled cleaning up after these prematurely terminated jobs. Just before 22:00, the cluster came back on line. Note that our processing system responded to the backlog of work by triggering the maximum level of parallelism (8 concurrent jobs) until the backlog was cleared in period C. We also observe that some intermittent failures and restarts during this time.

This example illustrates the robustness of our system to failures. Because work and computation are loosely scheduled, loss of compute nodes simply delays processing rather than losing data. We believe our system can tolerate temporary loss of any part of the system other than the data collection hosts themselves.

7. SUMMARY

In this paper we described a passive, continuous monitoring system to collect, archive, and analyze network data captured a Los Nettos, a large regional ISP within Los Angeles. We outlined the major requirements and challenges for the monitoring system. Our system supports multiple users, where each user can have different data and anonymization requirements. Further, the system is capable of handling large I/O and processing requirements due to the high-volumes of network data. We described a viable architecture and evaluated it with two-users to show that it effectively handles variable load. Traces collected via this system are available through the PREDICT project (`http://www.predict.org`).

Acknowledgements

We would like to thank Jim Pepin, Walter Prue, and Sanford George of Los Nettos, and Garrick Staples and Brian Yamaguchi of USC, for helping with the tracing infrastructure.

8. REFERENCES

[1] J.D. Case, M. Fedor, M.L. Schoffstall, and C. Davin. Simple network management protocol (SNMP). http://www.faqs.org/rfcs/rfc1157.html, May 1990.

[2] R. Cceres, Nick Duffield, Anja Feldmann, J Friedmann, A Greenberg, R Greer, T Johnson, C kalamanek, B Krishnana, F True, and J Merwe. Measurement and analysis of IP network usage and behavior. *IEEE Communications Magazine*, 38(5):144–151, May 2000.

[3] J. Fan, J. Xu, M. Ammar, and S. Moon. Prefix-preserving IP address anonymization. *Computer Networks*, 46(2):253–272, October 2004.

[4] Armando Fox, Steven D. Gribble, Eric A. Brewer, and Elan Amir. Adapting to network and client variability via on-demand dynamic distillation. In *Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 160–170. ACM, October 1996.

[5] C. Fraleigh, C. Diot, B. Lyles, S. Moon, P. Owezarski, D. Papagiannaki, and F. Tobagi. Design and deployment of a passive monitoring infrastructure. *Lecture Notes in Computer Science*, 2170:556–567, 2001.

[6] Robert L. Henderson. Job scheduling under the portable batch system. In *IPPS '95: Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, pages 279–294, London, UK, 1995. Springer-Verlag.

[7] Van Jacobson, Craig Leres, and Steven McCanne. tcpdump - the protocol packet capture and dumper program. http://www.tcpdmp.org.

[8] Greg Minshall. tcpdpriv - a prefix-preserving anonymization tool. http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html.

[9] Jelena Mirkovic, Sven Dietrich, David Dittrich, and Peter Reiher. *Internet Denial of Service: Attack and Defense Mechanisms*. Prentice-Hall, Upper Saddle River, NJ, 2004.

[10] Ruoming Pang and Vern Paxon. A High-level Programming Environment for Packet Trace Anonymization and Transformation. In *Proceedings of ACM SIGCOMM 2003*, pages 339–351, Karlsrhue, Germany, August 2003.

[11] Cisco Systems. Netflow services and applications. http://www.cisco.com/warp/public/732/netflow.

[12] USC. High Performance Computing and Communications. http://www.usc.edu/hpcc/.