Challenges and Opportunities in Internet Data Mining

David G. Andersen
Carnegie Mellon University

Nick Feamster
Georgia Institute of Technology

CMU-PDL-06-102
Jan 2006

Parallel Data Laboratory
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract

Internet measurement data provides the foundation for the operation and planning of the networks that comprise the Internet, and is a necessary component in research for analysis, simulation, and emulation. Despite its critical role, however, the management of this data—from collection and transmission to storage and its use within applications—remains primarily ad hoc, using techniques created and re-created by each corporation or researcher that uses the data. This paper examines several of the challenges faced when attempting to collect and archive large volumes of network measurement data, and outlines an architecture for an Internet data repository—the datapository*—designed to create a framework for collaboratively addressing these challenges.*

Keywords: network monitoring, databases, data management, data mining

1 Introduction

Communications networks produce vast amounts of data that both help network operators manage and plan their networks and enable researchers to study a variety of network characteristics. The data is rich, and it varies from the minute (*e.g.*, kilobits per hour of periodic average link utilization data) to the massive (*e.g.*, gigabits per second of live traffic captures). Its uses are equally diverse: On short timescales, the data can facilitate network monitoring, troubleshooting, and reactive routing [9]. Over minutes to days, the data is useful for network traffic engineering, in which operators attempt to shift the flow of traffic away from over-utilized links onto less busy paths. Over months, the data is useful for capacity planning; on even longer timescales, the data helps researchers perform longitudinal studies.

Network data is essential for the operation and planning of the networks that comprise the Internet; it is also vital for analysis, simulation, and emulation. Despite its critical role, the management of this data—from collection and transmission to storage and its use within applications—remains disconcertingly ad hoc, using techniques created and re-created by each corporation or researcher. As one example, consider the storage formats used for node-to-node loss and latency ("ping") data. A number of different projects collect such data, in formats ranging from:

- A hierarchy of directories, one per day, with an *N*-line text file containing a matrix of ping data between a set of *N* nodes, with errors recorded in a separate file. (PlanetLab all-pairs pings data [21].)
- A generalized binary network data storage format (Skitter's ARTS++ format [5]).
- Stored internally using a ROOT-based file [4] (RIPE's Test Traffic Machines [11]).
- A MySQL database, with one row per pair of probes and a non-standard text format for export upon demand (The RON testbed all-pairs probe data [2]).

This paper examines several challenges in collecting and archiving large volumes of network measurement data, and outlines an architecture for an Internet data repository—the *datapository*—designed to create a framework for collaboratively addressing these challenges. Rather than attempt to solve all of these problems, we hope that the datapository will catalyze sharing of tools, formats, and data among researchers and operators. We envision that the datapository will be useful both for storage and analysis, and as a tool for real-time network monitoring.

The heterogeneity of data formats and the lack of an infrastructure to analyze it efficiently harm both network operations and the science of network measurement. The data's unwieldy nature prevents network operators from harnessing the vast amounts of data that could save them time and money (*e.g.*, via network provisioning and traffic engineering) and help defend their networks from attack (*e.g.*, by alerting operators to routing or traffic anomalies).

From a scientific perspective, the lack of an analysis facility means that every network measurement researcher must develop "home brew" analysis tools and encounter the same set of traps and pitfalls as previous researchers. Worse yet, after a study is published, the authors' analysis tools may go unused for long periods of time before another researcher tries to study the same published results years later—at which point the data may have moved to a different location or been re-organized entirely. Even if a researcher successfully locates the data and tools from a particular study, he is left with the challenge of re-discovering how to reproduce the results ("Which parameters did they use to generate that graph?", "What subset of this data was used?", etc.). Indeed, for Internet measurement, repeatable experimentation—a major cornerstone of science—requires significant advances in the storage and management of data.

The remainder of this paper discusses the challenges that arise in constructing a network data analysis facility (Section 2), presents the initial design of the datapository (Section 3), and presents our vision for the opportunities that we hope the datapository will ultimately provide (Section 4). We warn the reader in advance that this paper raises more questions than it answers, though we hope it shines light on a possible path through the measurement minefield.

2 Challenges

This section briefly surveys the challenges in realizing a network data collection and storage facility.

2.1 Robust, distributed data collection

Collecting network measurement data is by nature distributed; this collection must be resilient to network and server outages, providing the datapository with an *eventually* complete data set. [1] In our experience, measurement nodes may be offline for days to months before coming back online and may be partitioned from the network for hours to days while still collecting measurements.

While work such as Paxson's Strategies for Sound Internet Measurement [17] provides valuable guidelines for how to *measure* and *analyze* data about the network, less attention has been paid to the *collection* of this data for subsequent analysis. Unfortunately, when large number of nodes or large volumes of data are involved, the difficulties involved in collecting and organizing can be as large as the difficulties of how to measure the network in the first place.

2.2 Multi-timescale, heterogeneous data

The combination of performing real-time analysis and data mining years of archived data presents challenges to both conventional and streaming database systems:

Non-standard, irregular data. Network data is information-dense, and we cannot predict how users will query the data, which makes indexing a challenge. Some data is best served by custom search techniques. For example, each routing update contains a *network prefix* and a *prefix length*. The prefix length specifies the number of significant bits in the prefix. A common operation in prefix queries is to find a *longest-prefix match (LPM)*: Given an IP address, find the prefix with the longest prefix length whose significant bits are equal to those bits in the IP address. Some data structures (*e.g.*, search tries) can efficiently perform LPM, but the best way to express this query in SQL is as a set of 32 OR'd conditions for each prefix length.

```
SELECT * from table
WHERE (prefix = x AND mask=32)
  OR  (prefix = x & 0xfffffffe
       AND mask=31)
  OR  ...
```

Additionally, network monitoring data is often not amenable to the regular form of standard databases. Packet traces, for instance, have nearly arbitrary content, and a search may examine any of these fields, extract the TCP stream from a series of packets, or examine application-layer headers. RBDMSes do not handle such queries, and existing tools such as `tcpdump` or `ethereal` cannot efficiently process terabytes of packet traces. Other data, such as the popular NetFlow format for flow summarization, has similar limitations for data mining.

[1]The measurement techniques used must also be robust to these events. This lesson is explained well by Paxson, whose original Internet measurements were triggered from a centralized node, and therefore under-counted some failures [16], or whose measurements depended on the DNS [18]. The authors of this paper have seen these mistakes repeated several times since.

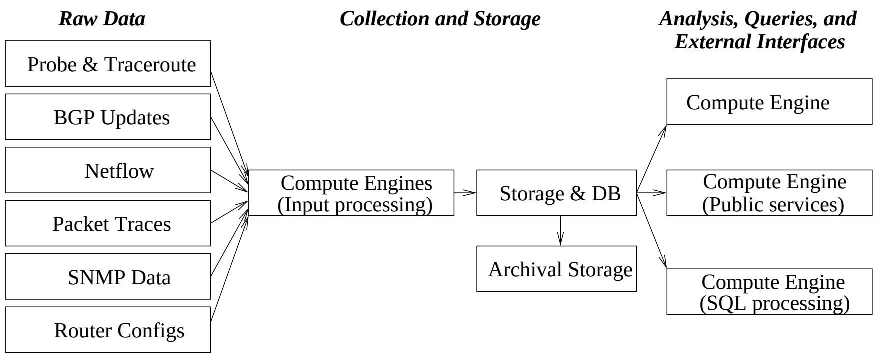

Figure 1: The datapository architecture.

To process non-standard data sources, the datapository may be able to leverage PADS, a system that takes as input a data format description and parses raw data files of arbitrary formats [10].

Large archival data mining. Aurora [3] provides a promising way to process real-time streams of network data, and AT&T's Gigascope [6] supports real-time monitoring of gigabit networks. These systems are excellent building blocks for processing raw data, but they are typically designed for either low-speed streams (*e.g.*, financial transactions) or modest-size "windows" of data. Longitudinal studies and joint analysis of network data may require much large windows over high-speed streams.

Most conventional solutions are unsuited to our application, and are unable to take advantage of many of the simplifying advantages of a data archive: the data is read-only, is easily cached, follows a strict time ordering, and is amenable to both intra- and inter-query parallelism.

Data-mining solutions such as Netezza [14] offer attractive solutions for data mining. Network measurement data offers many opportunities for optimization: the data is read-only, is easily cached, and follows a strict time ordering. Unfortunately, current data mining tools do not cope well with the unique data types present in network monitoring data. We believe that the long-term development of the datapository will spur the development of log-processing database management systems.

2.3 Workflow metadata and privacy

Metadata about data's origins, errors, and modifications—its *provenance*—must be maintained and communicated with the data as it progresses through the analysis workflow [13, 23]. Data producers or the research community must agree upon common metadata formats for data [1] and analysis tools in order to facilitate data sharing.

An important part of the workflow is addressing privacy requirements. The privacy requirements for measurement data range from nearly none (*e.g.*, many BGP routing traces, which are publicly available) to extreme (*e.g.*, an un-anonymized packet trace). In order to gain the efficiencies of scale that we hope to realize with the Datapository, it must be able to host and analyze data at the ends and in the middle of the privacy spectrum. In fact, the steps of "declassifying" this data (*e.g.*, using anonymization "scrubbers" [12] or summarization techniques) will be an important part of the analysis workflow that the datapository enables.

3 The datapository

The datapository consists of a distributed set of *data sources* and a central *storage and analysis* infrastructure, shown in Figure 1. The data is processed by a set of *compute engines* dedicated to input processing, which insert the data into the central storage and database nodes. Once inserted, the data may be analyzed by tools on other compute engines.

3.1 Data Collection

The datapository collects data from many sources, including the RON testbed path monitoring nodes and BGP feeds, Abilene data, and the RouteViews BGP archive. Internally, the datapository understands an orthogonal set of *data types* (which abstract a set of *data formats*), and *access methods*:

Data types are a canonical version of a particular type of measurement (*e.g.*traceroutes, end-to-end probes, and BGP routing updates). The datapository is currently archiving and processing BGP routing tables and updates from RON and Abilene, and spam logs from two spam honeypots. We have immediate plans to archive RouteViews BGP feeds and end-to-end probes from the RON testbed.

Data formats are different storage formats for a particular data type. Each of the formats listed in the Introduction would be one such data format; other examples include the MRT format commonly used to store BGP routing updates, or ASCII files recording traceroutes.

Access methods are implementations of techniques that the datapository obtains data from the data sources. Examples include the code that obtains data (*e.g.*, RouteViews) via FTP, and that which securely copies data from the RON testbed nodes to the datapository via SSH.

Central to the datapository is the notion of a *feed*. A feed represents an (access method, data format, data type) tuple, along with the ancillary data used by the access method or data format (*e.g.*, usernames and passwords, or state corresponding to which objects have already been retrieved). Each feed defines a *workflow* that uses an access method to obtain data in a particular format, converts that format into the canonical representation for that data type, and inserts the data into storage.

3.2 Data Storage

We wish to store data in a cluster of databases that offer an SQL-like interface. Our prior experience and that of others suggests that basing analysis out of a database (we choose SQL out of familiarity) improves consistency, facilitates reuse, and reduces researcher effort [20, 17]. To this end, the present datapository architecture uses a set of MySQL compute engines, each of which has read-only access to the historical data via a distributed filesystem.

This approach has the advantage of simplicity and working with off-the-shelf components, but fails to take advantage of the intra-query parallelism that is available in many of the data-mining queries we issue against the datapository. In addition, the current architecture is limited in its ability to simultaneously query the historical data and newly arrived data, which is maintained read-write by a single database node. We view the development of more suitable storage and mining techniques as a prime challenge for the further development of the datapository.

3.3 Analysis and Queries

We envision three "users" for the datapository: Network operators, researchers performing analysis, and simulation/emulation systems that require traces as input.

Network operators will be able to use the datapository for real-time debugging and network monitoring. By aggregating data from many vantage points into a logically centralized database, operators can efficiently get information from a geographically and topologically diverse set of network locations. Our previous implementation of a public "BGP monitor" was useful to operators in identifying network-wide failures and debugging reachability problems. Such utilities are particularly useful because they grant operators a view of the network from vantage points outside their own network.

Our instance of the datapository holds datasets that researchers and operators choose to make publicly available. Operators may wish to build private datapositories to analyze confidential data and query that data either in isolation or in combination with the publicly available datapository. Analyzing traffic and routing data can help operators with traffic engineering and capacity planning.

The datapository can help network researchers perform long-term longitudinal studies. Network data often exists in two forms: (1) short-term data collected for a particular experiment and (2) long-term data that is collected, but never analyzed *in toto* due to the massive overhead in running a query For example, the RouteViews BGP routing updates comprise BGP data from about 41 routers in 35 different ISPs [15]. An hour of compressed BGP updates from RouteViews requires one megabyte; as a result, longitudinal queries across multiple years and multiple BGP datasets can become prohibitively expensive. As a shared facility, the datapository can amortize storage, processing, and management costs, such as building and maintaining a database and query infrastructure for longitudinal datasets. We hope that lowering the cost of running queries will foster serendipity by making it easier for researchers to issue exploratory queries.

The datapository aggregates multiple data feeds into a single archive, which facilitates performing joint analysis across multiple datasets. Among many examples, our past research has shown the benefits of performing joint analysis across traceroutes, active probes, and BGP routing updates to study the correlations between end-to-end path performance and Internet routing stability [9]. More recently, we have studied the interaction between BGP route hijacking and spam activity [8]. It is our hope that, by providing a common storage and query infrastructure for diverse datasets, Internet researchers will find new ways of finding patterns and correlations across datasets.

3.4 External Interfaces

The datapository makes raw and aggregated data available through standard interfaces. The datapository currently supports an *XMLRPC* export mechanism, and we are implementing *bulk raw data* export. The XML-RPC interface can export data in various formats, including human-readable output and Matlab-friendly matrices. This data export mechanism allows researchers to focus on analysis techniques without being intimately familiar with the myriad formats for network data. In addition to these export formats, we have created several Web interfaces for browsing data and for network troubleshooting.

To facilitate experimentation, we are integrating the datapository with Emulab [22]. Emulab provides hardware resources (nodes and links) that can be configured into a desired network topology. Researchers use this emulated network to evaluate real code and distributed systems. Emulab provides a valuable middle ground between simulation and untamed (and un-repeatable) live networks. However, the accuracy of many emulation experiments—much like simulations—lives and dies by the model and input parameters. Coupling the datapository to Emulab can significantly improve this aspect.

Our goal is to foster a seamless flow of data from network traces to Emulab experiments and back. Emulab users will have access to realistic topologies, background traffic patterns, configurations, and even prior experimental and analytical data. The data produced by these experiments can then be inserted into the datapository for archiving and further processing.

This integration presents a classical *workflow* problem, managing the flow of data from the datapository, into emulation (and perhaps the real world, via Emulab's wide-area nodes), and back. Internet research involves a continual cycle of measuring a system to better understand and evaluate its properties, modeling these properties in a way that makes them easy to integrate into a controlled experiment (*e.g.*, in simulation or emulation), and making changes to the system that is actually deployed. The datapository acts as a critical component in this process by facilitating data collection and management, as well as acting as a driver for trace-based emulation. Section 4.1 describes this process in more detail, as well as its relation to Emulab's scientific workflow management [7].

4 Opportunities

This section explores possible future uses for the datapository. We focus first on how the datapository might help scientific experimentation; then, we explore how it might be employed as a cornerstone of network management—even to the point of *controlling* network operation.

4.1 Repeatable Scientific Experimentation

Two significant challenges in experimentation are repeating a past experiment (under the same or different conditions) and performing new analysis of old data. Without standard techniques for packaging old data and analysis scripts, measurement data is often shuffled about and forgotten or becomes a subset of a continuously collected data stream. At a later date, researchers must recall which subsets of each data stream and which processing scripts they used to produce each set of results.

The datapository can save measurement data from these fates by allowing a researcher to store the details of their analysis workflow. This archived workflow names all of the machinery a researcher would need to reproduce a particular experiment: the data, all scripts needed for analysis, and the process by which these scripts should be executed (*i.e.*, a description of the experiment). The workflow should also refer to any relevant information about how the data was collected and other noteworthy aspects of the data, such as times when the collection machinery was faulty or non-functional, how the data was collected, etc. Our support for workflows is complementary to Emulab, which proposes integrating workflow management into *generating*—not just analyzing—measurements [7].

The datapository's archives may also prove useful for driving trace-driven network emulations. The networking community is developing several platforms for network emulation, which would benefit from the ability to play back traces of actual network data. For example, a researcher might set up an experiment that mirrors a real network topology, process traces at the datapository as they are collected from the real network, and project the failures in these traces onto the mirrored network.

4.2 Network Monitoring and Control

Network operators must ensure availability and good performance in the face of constantly changing conditions (*e.g.*, changing traffic patterns, link failures) and security threats (*e.g.*, worms, denial-of-service (DoS) attacks). Unfortunately, critical data often escapes operators' notice due to both the lack of an infrastructure to collect and analyze this data and the lack of adequate detection algorithms. We hope that the datapository will evolve to support real-time analysis and detection. A real-time analysis facility might even allow operators to automate some aspects of network control. For example, such a system could detect erroneous routing advertisements and prevent them from propagating; it could also detect a DoS or worm attacks and install filters.

5 Discussion and Summary

The datapository is an in-progress facility for shared network data analysis. We believe that a community-based approach to network data analysis can play a critical role in both networking research and operations. In contrast to existing approaches that emphasize cataloging and standardizing Internet data [19], we believe that providing a public computational and storage resource will be an effective stimulus for community involvement and adoption of the standards that evolve.

While building the datapository, we have encountered several conflicts between existing workflow and data management systems and the requirements of network data. We hope that this paper will serve as a catalyst for the improvement of both our data repository, and the available data management techniques.

References

[1] Mark Allman, Ethan Blanton, and Wesley M. Eddy. A scalable system for sharing Internet measurement. In *Passive & Active Measurement (PAM)*, Fort Collins, CO, March 2004.

[2] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris. Experience with an Evolving Overlay Network Testbed. *ACM Computer Communications Review*, 33(3):13–19, July 2003.

[3] Hari Balakrishnan, Magdalena Balazinska, Donald Carney, et al. Retrospective on Aurora. *VLDB Journal*, January 2004.

[4] Rene Brun and Fons Rademakers. ROOT - an object oriented data analysis framework. In *Proc. AIHENP'96 Workshop*, September 1996.

[5] CAIDA's Skitter project, 2002. http://www.caida.org/tools/measurement/skitter/.

[6] Chuck Cranor, Theodore Johnson, Oliver Spataschek, and Vladislav Shkapenyuk. Gigascope: a stream database for network applications. In *Proc. ACM SIGMOD*, pages 647–651, San Diego, CA, June 2003.

[7] Eric Eide, Juliana Freire, Jay Lepreau, Tim Stack, and Leigh Stoller. Integrated scientific workflow management for the Emulab network testbed. Submitted for publication, December 2005.

[8] Nick Feamster. Open Problems in BGP Anomaly Detection. http://www.caida.org/outreach/isma/0411/abstracts.xml#NickFeamster2, Nov 2004. Slides available upon request.

[9] Nick Feamster, David Andersen, Hari Balakrishnan, and M. Frans Kaashoek. Measuring the effects of Internet path faults on reactive routing. In *Proc. ACM SIGMETRICS*, San Diego, CA, June 2003.

[10] Kathleen Fisher and Robert Gruber. Pads: a domain-specific language for processing ad hoc data. In *Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation*, pages 295–304, Chicago, IL, Jun 2005.

[11] Fotis Georgatos, Florian Gruber, Daniel Karrenberg, Mark Santcroos, Ana Susanj, Henk Uijterwaal, and Rene Wilhelm. Providing active measurements as a regular service for ISP's. In *Passive & Active Measurement (PAM)*, April 2001.

[12] Alefiya Hussain, Genevieve Bartlett, Yuri Pryadkin, John Heidemann, Christos Papadopoulos, and Joseph Bannister. Experiences with a continuous network tracing infrastructure. In *Proc. ACM SIGCOMM Workshop on Mining Network Data (MineNet)*, Philadelphia, PA, August 2005.

[13] Jonathan Ledlie, Chaki Ng, David Holland, Kiran-Kumar Muniswamy-Reddy, Uri Braun, and Margo Seltzer. Provenance-aware sensor data storage. In *Proc. Workshop on Networking Meets Databases (NetDB)*, April 2005.

[14] Netezza. Business intelligence data warehouse appliance. Netezza Home Page.

[15] University of Oregon. RouteViews. http://www.routeviews.org/.

[16] V. Paxson. End-to-End Routing Behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615, 1997.

[17] Vern Paxson. Strategies for sound Internet measurement. In *Proc. ACM SIGCOMM Internet Measurement Conference*, Taormina, Sicily, Italy, October 2004.

[18] Vern Paxson, Andrew Adams, and Matt Mathis. Experiences with NIMI. In *Passive & Active Measurement (PAM)*, April 2000.

[19] Colleen Shannon, David Moore, Ken Keys, Marina Fomenkov, Bradley Huffaker, and k. claffy. The Internet measurement data catalog. *ACM Computer Communications Review*, 35(5):97–100, October 2005.

[20] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring ISP topologies with Rocketfuel. In *Proc. ACM SIGCOMM*, Pittsburgh, PA, August 2002.

[21] Jeremy Stribling. Planetlab - all pairs pings. `http://pdos.csail.mit.edu/~strib/pl_app/`, nov 2005.

[22] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An integrated experimental environment for distributed systems and networks. In *Proc. 5th USENIX OSDI*, pages 255–270, Boston, MA, December 2002.

[23] Jennifer Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *Proc. 2nd Biennial Conference on Innovative Data Systems Research (CIDR)*, Pacific Grove, CA, January 2005.