Design of an IP Flow Record Query Language

Vladislav Marinov and Jürgen Schönwälder

Computer Science, Jacobs University Bremen, Germany
{v.marinov,j.schoenwaelder}@jacobs-university.de

Abstract. Internet traffic is often summarized by collecting NetFlow/IPFIX flow records. Several tools exist to filter or to search for specific flows in a collection of flow records. However, there is a need for a framework (filter language) which allows certain types of traffic patterns to be defined and matched in a collection of flow records. The goal of this project is to research the various filter/query languages used by tools or proposed in the literature and to extract a common basis for a new orthogonal flow record query language. We present research motivation and state of the art in this paper.

Key words: NetFlow, IPFIX, network traffic analysis, query language

1 Introduction

The analysis of network traffic and more specifically Internet traffic has become an important area of research. Cisco has designed the Netflow/IPFIX protocol [1, 2], which allows to create a summary for the traffic flows that traverse a router. A network flow is defined as an unidirectional sequence of packets between given source and destination endpoints. Flow records include details such as IP addresses, packet and byte counts, timestamps, Type of Service (ToS), application ports, input and output interfaces, etc. Network elements (routers and switches) gather flow data and export it to collectors for analysis.

Although the flow records carried by NetFlow/IPFIX provide aggregated information about the packets traversing a specific router, this information still contains too many details for network administrators and is not useful unless processed by network analysis tools. Most of the existent tools provide mechanisms for selecting specific flows in a collection of flow records. This makes possible some simple tasks like filtering by an IP address or port number or generating Top N talkers reports. However, identifying more complex flow patterns resembles a search for a pin in a haystack. In order to describe complex traffic patterns and match a collection of flow records against the description, one needs a useful flow record query language.

The rest of this paper is structured as follows: Section 2 presents the state of the art in query languages used by network analysis tools. In Section 3, we present the research motivation for a new filter/query language and we conclude in Section 4.

2 State of The Art in Query/Filter Languages

Several early implementations of network analysis tools used a Relational Database Management System (RDBMS) to store the data contained in flow records and therefore they use SQL-based query languages for selecting flows.

B.Nickless [3] describes a system which uses standard MySQL and Oracle DBMS for storing the attributes from NetFlow records. Using powerful SQL queries, the tool was able to provide good support for basic intrusion detection and usage statistics. With the advance of high-speed links, however, network managers could not rely on pure DBMS anymore because of performance issues. There was also a semantic mismatch between the traffic analysis operations and the operations supported by the commercial DBMS. The data used by network analysis applications can be best modeled as transient data streams as opposed to the persistent relational data model used by traditional DBMS. It is recognized that continuous queries, approximation and adaptivity are some key features that are common for such stream applications. However, none of these is supported by standard DBMS. Based on these requirements B.Babcock et al. [4] propose the design of a Data Stream Management System (DSMS). Together with the model the authors also extend the SQL query language so that the DSMS can be queried over time and provide examples of network traffic reports that are generated based on flow data that is stored in such a DSMS. *Gigascope* [5] is another stream database for network monitoring applications. It uses GSQL for query and filtering which is yet another modification of the SQL query language adopted in a way so that time windows can be defined inside the query. *Tribeca* [6] is another extensible, stream-oriented DBMS designed to support network traffic analysis. It is optimized to analyze streams coming from the network in real time as well as offline traces. It defines its own stream query language which supports operations such as projection, selection, aggregation, multiplexing and demultiplexing of streams based on stream attributes. The query language also defines a windowing mechanism to select a timeframe for the analysis.

The *Berkeley Packet Filter (BPF)* [7] specifies simple rules which are widely used among network analysis tools to filter a stream of packets. BPF allows users to construct simple logical expressions for filtering network traces by IP address, port number, protocol etc. and translates them into a small program executed by a generic packet filtering engine. One popular use of the BPF is in the `tcpdump` utility. The BPF rules for constructing filter expressions are also used in `nfdump` [8], which is a powerful and fast filter engine used to analyze network flow records. `nfdump` is currently one of the *de facto* standard tools for analyzing NetFlow data and generating reports. BPF expressions are also used in the *CoralReef* network analysis tool described in [9, 10] in order to generate traffic reports from collected trace files. The *Time Machine* tool described in [11] uses BPF expressions to define classes of traffic and BPF is also part of the query language used by the engine for retrieval of interesting traffic.

The `flow-tools` package [12] is another widely-used collection of applications for collecting and analyzing NetFlow data. Two of the flow-tools applications are responsible for filtering flows and generating reports: `flow-filter`

and `flow-report`. The former application uses the Cisco Access Control List (ACL) format to specify a filter for IP addresses and command line arguments for specifying other filtering parameters such as port numbers, ASes etc. The latter uses a configuration file where reports can be defined by using a number of primitives.

FlowScan described in [13] is a collection of perl scripts which glues together a flow-collection engine such as the `flow-capture` application from `flow-tools`, a high performance RRD database, which is specifically designed for time series data [14], and a visualization tool. FlowScan has the capability of generating powerful high-level traffic reports, which might help operators to detect interesting traffic patterns. However, reports must be specified as separate perl modules, which is not trivial and might involve some heavy scripting.

C.Estan et al. [15] proposes an approach for detecting high-level traffic patterns by aggregating NetFlow records in clusters based on the flow record attributes. Aggregation on several flow attributes results in a multidimensional cluster. Initially all possible multidimensional clusters are constructed. Then an algorithm is executed which selects only clusters that are interesting to the network administrator. It aims at retaining clusters with the least degree of aggregation (so that a bigger number of flow attributes is contained). Interesting activities are considered to be exceeding a certain threshold of traffic volume of a cluster or significant change of the traffic volume inside the cluster. Finally, all clusters are prioritized by being tagged with a degree of *unexpectedness* and presented to the network administrator as a traffic report.

The *SiLK* Analysis Suite [16] is another script-based collection of command-line tools for querying NetFlow data. It provides its own primitives for defining filtering expressions. Unlike other network analysis tools, *SiLK* contains two applications that allow an analyst to label a set of flows sharing common attributes with an identifier. The `rwgroup` tool walks through a file of flow records and groups records that have common attributes, such as source/destination IP pairs. This tool allows an analyst to group together all flows in a long lived session such as a FTP connection. `rwmatch` creates matched groups, which consist of an initial record (a query) followed by one or more responses. Its most basic use is to group records into both sides of a bidirectional session, such as a HTTP request. From the huge collection of tools that we have surveyed *SiLK* is the only one, which is capable of declaring some correlation between flows. Therefore, we believe that it might serve as a good basis for a new flow query language.

A summary of the query languages used by the various network traffic analysis tools is presented in Table 1.

3 Research Issues

Given the large number of flow records collected on high-speed networks, it is necessary to reduce their number to a comprehensible scale using filtering and aggregation mechanisms. Each flow or aggregated flow has a set of properties attached to it that characterize the flow. It is to be expected that flows that

Table 1. Query languages used by network traffic analysis tools

Tool	Query Language	Input Data Format
B.Nickless et. al. [3]	SQL	RDBMS
B.Babcock et. al. [4]	extended SQL	DSMS
Gigascope	GSQL	DSMS
Tribeca	proprietary	DSMS
`tcpdump`	BPF	`pcap` files
`nfdump`	BPF	`nfcapd` raw NetFlow files
CoralReef	BPF	`pcap` and `crl` files
Time Machine	BPF	indexed `pcap` files
Flow-Tools	ACL/proprietary	`flow-capture` raw NetFlow files
FlowScan	perl script	`flow-capture` raw NetFlow files
AutoFocus	proprietary	packet header traces/raw NetFlow files
SiLK	proprietary	raw NetFlow files

correspond to similar network activities (certain applications or certain attacks) have similar properties. In addition to the properties recorded in flow records, one can derive further properties that are even more suitable to characterize the behavior of a flows. One objective when investigating traces is to detect regularities such as repeating patterns. These patterns typically spread over several flows. For example, if an intensity peak in flow X always occurs after an intensity peak in flow Y with a fixed delay, they form a pattern describing a certain network behavior. The goal of network administrators is to detect such patterns of correlated flows.

For example, one would be interested in finding out where, when, and how often a certain Internet service is used. A concrete scenario is a network administrator who wants to detect VoIP applications by finding STUN flows generated by VoIP applications when they try to discover whether they are located behind a Network Address Translator (NAT). If one knew the pattern that is created when a service is trying to establish a connection, one could search for this specific pattern in the selected flows.

The goal of this project is to design a flow record query language, which allows to describe patterns in a declarative and easy to understand way. The language should be able to define filter expressions (needed to select relevant flows) and relationships (needed to relate selected flows). Another requirement is that it should be possible to express causal dependencies between flows as well as timing and concurrency constraints. Existent query languages as discussed in Section 2 are not suitable for detecting complex traffic patterns because of either performance issues (SQL-based query languages) or because they lack a time and concurrency dimension (BPF expressions and the other query languages we discussed). Furthermore, the new query language should provide support for network specific aggregation functions, such as IP address prefix aggregation, IP address suffix aggregation, port number range aggregations, etc. which are not part of many standard query languages.

4 Conclusions

We presented the state of the art in query languages used by network traffic analysis tools and motivated the need for developing a new declarative language that allows to define and identify high level traffic patterns in a collection of network flow records. We plan to collect a comprehensive set of interesting traffic patterns from network operators and base our new query language on the needs to express these patterns.

Acknowledgement

The work reported in this paper is supported by the EC IST-EMANICS Network of Excellence (#26854).

References

1. Claise, B.: Cisco Systems NetFlow Services Export Version 9. RFC 3954 (October 2004)
2. Claise, B.: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information. RFC 5101 (January 2008)
3. Nickless, B.: Combining Cisco NetFlow Exports with Relational Database Technology for Usage Statistics, Intrusion Detection, and Network Forensics. In: Proc. of LISA'00, USENIX Association (2000) 285–290
4. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in Data Stream Systems. In: Proc. of PODS'02, ACM (2002) 1–16
5. Cranor, C., Johnson, T., Spataschek, O., Shkapenyuk, V.: Gigascope: A Stream Database for Network Applications. In: Proc. of SIGMOD'03, ACM (2003) 647–651
6. Sullivan, M., Heybey, A.: Tribeca: a System for Managing Large Databases of Network Traffic. In: Proc. of ATEC'98, USENIX Association (1998) 13–24
7. McCanne, S., Jacobson, V.: The BSD Packet Filter: A New Architecture for User-level Packet Capture. In: Proc. of USENIX'93, USENIX Association (1993) 259–270
8. : nfdump, http://nfdump.sourceforge.net/
9. Moore, D., Keys, K., Koga, R., Lagache, E., Claffy, K.: The Coral Reef Software Suite as a Tool for System and Network Administration. In: Proc. of LISA XV, USENIX Association (2001) 133–144
10. Ken Keys and David Moore and Ryan Koga and Edouard Lagache and Michael Tesch and KC. Claffy: The Architecture of CoralReef: an Internet Traffic Monitoring Software Suite. In: Proc. of PAM'01, CAIDA, RIPE NCC (April 2001)
11. Kornexl, S., Paxson, V., Dreger, H., Feldmann, A., Sommer, R.: Building a Time Machine for Efficient Recording and Retrieval of High-Volume Network Traffic. In: Proc. of IMC'05, USENIX Association (2005)
12. : flow-tools, http://www.splintered.net/sw/flow-tools/
13. Plonka, D.: FlowScan: A Network Traffic Flow Reporting and Visualization Tool. In: Proc. of LISA'00, USENIX Association (2000) 305–318
14. : Rrdtool, http://oss.oetiker.ch/rrdtool/
15. Estan, C., Savage, S., Varghese, G.: Automatically Inferring Patterns of Resource Consumption in Network Traffic. In: Proc. of SIGCOMM'03, ACM (2003) 137–148
16. Michael Collins, Andrew Kompanek, Timothy Shimeall: Analysts Handbook: Using SiLK for Network Traffic Analysis. CERT. 0.10.3 edn. (November 2006)