Design and Implementation of an Interactive DBMS-supported Network Traffic Analysis and Visualization System

Hyun-chul Kim, Jihoon Lee

1, First Author
Dept. of Computer Software Engineering, Sangmyung Univ., hyunchulk@gmail.com

2, Corresponding Author
Dept. of Information and Telecommunication Engineering, Sangmyung Univ., vincent@smu.ac.kr

Abstract
Measurement, analysis, and visualization of traffic data are critical day-to-day tasks for effective and efficient network management and operation. In this paper, we design and implement a novel interactive DBMS (Database Management System)-supported network traffic analysis and visualization system. To this end, we first survey and compare existing tools for network traffic measurement and analysis in terms of their supported features and functionalities. Then, we choose to extend FlowScan, one of the most widely-used software tool for network traffic analysis and visualization, with DBMS support as well as Web-based interactive query interface. We describe the overall system architecture and implementation, and demonstrate that our system can be used to easily identify heavy hitters (e.g., malicious attackers), in near real time.

Keywords: Traffic Measurement, Visualization, FlowScan

1. Introduction

Measurement, analysis and visualization of traffic data are very important tasks with regard to effective and efficient network management and administration [1, 2, 7]. The two common approaches for network measurement are the passive and active ones. Passive techniques watch target traffic as it passes by, for example, at routers, switches or hosts, without generating (or injecting) additional traffic packets in the network. On the contrary, active measurement techniques inject packets into the network or transmit packets to other servers or hosts, to test, diagnose, or measure network delay, connectivity, path, etc. For the purpose of measurement of traffic volume, per-application traffic analysis, per-AS traffic analysis, and top-user behavior analysis, passive measurement approach is used. Table 1 shows existing passive measurement tools and their supported useful features. As shown in the table, no tool is currently supporting all the useful features.

While FlowScan [1], though it is one of the most widely network traffic measurement tools, supports many good features like time series graph (as shown in Figure 1) through which users can easily understand overall network traffic trend, it does not provide interactive interfaces for further detailed analysis; e.g., exactly identifying heavy hitters, etc. In this paper, our goal is to design and implement the first passive measurement software where all the important features in Table 1 are supported. To this end, after conducting careful and in-depth analysis of the nine existing tools, we have decided to extend FlowScan with DBMS and Web-based query interface.

The rest of this paper proceeds as follows. In Section 2, we first introduce FlowScan. Section 3 describes the architecture design, implementation, and usage examples of our proposed and developed system. Section 4 concludes the paper.

2. FlowScan

2.1. Architecture

As the first step, we have analyzed FlowScan system architecture and its codeset. FlowScan mainly consists of four different architectural components: (1) Netflow [9] collection module, (2) Netflow analysis module, (3) Netflow storage module, and (4) visualization module. Three of the four modules (i.e., Netflow collection, storage, and visualization modules) had actually been borrowed from other open source software; FlowScan's Netflow collection module uses CAIDA’s cflowd, while storage and visualization module use RRD (Round Robin Database). In
this paper, from the next subsection, we mainly focus on the Netflow analysis module, as we design our to-be-developed system to put the various Netflow analysis results and related data into a database system and handle them. In Figure 2, cflowdmux and cflowd together form Flowscan’s collection module, while flowscan, rrd, and rrdtool is analysis, storage, and visualization modules, respectively.

Table 1. Passive Measurement Tools: Comparison

<table>
<thead>
<tr>
<th></th>
<th>MRTG</th>
<th>Flow-Scan</th>
<th>Cflowd with arts</th>
<th>tcpdump</th>
<th>MADAS</th>
<th>Flow-tools</th>
<th>FlowCollector</th>
<th>Flow Analyzer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw data Collection</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>X</td>
</tr>
<tr>
<td>Data analysis</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Visualize</td>
<td>O</td>
<td>O</td>
<td>X</td>
<td>X</td>
<td>O</td>
<td>X</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td>Time Series Graph</td>
<td>O</td>
<td>O</td>
<td>X</td>
<td>X</td>
<td>O</td>
<td>X</td>
<td>X</td>
<td>△</td>
</tr>
<tr>
<td>Raw flow dump</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>X</td>
<td>X</td>
<td>O</td>
<td>X</td>
</tr>
<tr>
<td>Query Interface</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td>Traffic volume</td>
<td>O</td>
<td>O</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Per-app. Analysis</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Top User</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Top AS</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

Figure 1. FlowScan’s time series graph
2.2. Netflow analysis module

Using cflowd, Flowscan dumps Netflow data, collected at and sent by routers every 5 minutes. Then, flowscan module is called to analyze (for every 5 minutes as well) the received flow data files and then sleeps again for five minutes. The flowscan module reads the dumped data files and applies the analysis module on every single collected flow, classifying them according to the flow’s port number, protocol number, output interface, etc. Among the analysis modules, SubNetIO.pm and CampusIO.pm are the representative ones, through which classification of the flow is performed (e.g., whether it is TCP, FTP, or/and inbound traffic), that put the analysis results into the RRD database. Figure 3 shows the pseudo-code of the CampusIO.pm module, where the wanted function is applied to target flows to classify them according to the destination address, interface number, etc.

3. Proposed system

Figure 4 describes our proposed system, comparing with its predecessor FlowScan. Basically, we have added three additional modules to FlowScan, which are flow database management module, data aggregation module, and query module, as described in this section.
3.1. Database management module

The Netflow analysis module in FlowScan finds whether a flow under processing (i) is of in-bound or out-bound, and (ii) can be classified using its layer-4 port number information or not. Right after this finding process, our proposed system injects all the analyzed information of the flow into database. Our proposed database system records the following information for every flow: source IP, destination IP, input interface, output interface, source port, destination port, ICMP type, the number of packets, bytes volume, next hop, flow starting time, ending time, protocol, source AS, destination AS, TCP flag, inbound or outbound, known application or unknown (by its port number). We used MySQL as DBMS, one of the world’s most popular open source database.

```perl
package CampusIO;

sub profile {
    # Remember the time-stamp from the filename.
    whence = filename2time_t(filename)
}

sub wanted {
    if (exporter_hop(flow::next_hop)) {
        # This flow is destined for another "local" flow-exporting router.
        # We'll catch it later, if and when it's exported by that router.
        return 0
    }

    if (outbound_hop(flow::next_hop)
        or outbound_interface(flow::output_if)) {
        # This flow is outgoing.
        outbound_total++
    } elsif (inbound_address(flow::destination_address)) {
        # This flow is incoming.
        inbound_total++
    } else {
        # This flow (an "intranet" flow) is unwanted.
        return 0
    }

    return 1
}

sub report {
    update_RRD_files(whence, inbound_total, outbound_total)
}
```

Figure 3. CampusIO.pm’s pseudo-code
3.2. Web-based query interface

Once all the flow data are stored at our database system, our system allows users to extract useful information from the database through Web-based (using Perl CGI module) query interface. Through the interface, our system provides a set of predefined queries such as top 10 users, per-AS analysis, per-port analysis, per-protocol analysis, etc., which are typically most often useful and informative tasks for network operators. Once users set parameter values using Web interface, the chosen parameter values are handed over to Perl CGI scripts that (i) parse them, (ii) generate SQL queries based on the parsed results, (iii) submit the SQL queries to the NetFlow database, and (iv) convey back the query results to the users through the Web interface. Figure 5 shows a screen shot of the developed Web query interface, tested with a predefined “Top 10 Users” query. At the time of this writing, our system supports the following predefined queries: (i) overall traffic statistics, (ii) all flows in a specific time period, (iii) trace flows per a user, (iv) per-protocol usage statistics, (v) per-port usage statistics, (vi) next hop statistics, (vii) per-AS information and traffic matrix, (viii) packet size and flow distribution, (ix) top 10 users, protocols, ports, ASes. All these information are extracted from raw NetFlow flows, which contain a variety of information as shown in Figure 6.

3.3. Data aggregation

One notable problem we have confronted building our Netflow database system is, its too large volume. According to our experience, at a mid-sized university campus with O(1,000) people inside, a border router exports around 100 K flows in 15 minutes on average, which means around 9.6 M flows per day and 3.5 G flows per year would have to be stored into and
handled at our database system. In fact, when we tested with a query handling one-day data (dealing with approximately 9.6 M flows), it took 1 minute to get the query results. The larger time span of data a user wants to process, the longer time they would have to wait. Besides, storing all the raw flow information in our database system takes up too much disk space. To alleviate this limitation, our system performs data aggregation.

In particular, data aggregation in our system has been designed to speed up predefined queries. For every 15 minutes, it performs the flow aggregation process based on the five guidelines, reflecting the predefined queries most (likely to be) often used by network operators; per port, per protocol, per top user, per AS matrix, per next hop IP, as shown in Table 2. The proposed aggregation scheme has significantly contributed in reducing disk storage usage as well as query response times. For example, it took only a few seconds to handle a query with one week’s flow data. Also, as it takes too much disk space to store and archive full raw flow dump data for a long time, network operators may prefer to choose to store only aggregated data for a long-term data collection. Designing a data aggregation scheme for the largest storage space saving with the smallest information loss is left as our future work.

Figure 5. Web-based query interface: a screen shot (IP addresses and date information are masked out due to privacy concerns)

3.4. Deployment and Usage Examples

ISP or network operators are often interested in finding hosts or flows consuming most of network resources (e.g., heavy hitters, scanners, spammers, etc.). Our system’s user interface makes it easy and convenient for users to quickly identify heavy hitters by providing both time series graph as well as interactive query system, as shown in Figure 7 and 8.

We have deployed our developed system at a border router of a university campus network. While operating, when FlowScan generates a time series traffic plot like Figure 7, indicating
that sudden abnormal traffic spikes are observed, then network operators, using our Web-based
query interface, can easily and successfully identify who they are (i.e., IP addresses), along with
the detailed information such as what protocol and ports they use, how large is the flows, how
many they are, from when to when, etc., as shown in Figure 8. Consequently, using our
developed tool, we were able to exactly identify a DDoS attack destined to a machine (whose IP
address was x.204.97.179) on our network.

<table>
<thead>
<tr>
<th>FLOW</th>
<th>index: 0xctff</th>
<th>router: A.B.C.D</th>
<th>src IP: E.F.G.H</th>
<th>dst IP: I.J.K.L</th>
</tr>
</thead>
<tbody>
<tr>
<td>input fIndex: 13</td>
<td>output fIndex: 12</td>
<td>src port: 16000</td>
<td>dst port: 2036</td>
<td></td>
</tr>
<tr>
<td>pkts: 1</td>
<td>bytes: 58</td>
<td>IP nexthop: M.N.O.P</td>
<td>start time: Sat Dec 8 20:34:41</td>
<td></td>
</tr>
<tr>
<td>end time: Sat Dec 8 20:34:41</td>
<td>protocol: 6</td>
<td>tos: 0</td>
<td>src AS: 9848</td>
<td></td>
</tr>
<tr>
<td>dst AS: 1781</td>
<td>src masklen: 13</td>
<td>dst masklen: 24</td>
<td>TCP flags: 0x18 (PUSH</td>
<td>ACK)</td>
</tr>
<tr>
<td>engine type: 1</td>
<td>engine id: 3</td>
<td>in/out unknown</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 6.** A sample flow information (Netflow ver. 5)

Table 2. Aggregated Data Table: Schema

<table>
<thead>
<tr>
<th>Table Port</th>
<th>srcport</th>
<th>dstport</th>
<th>pkts</th>
<th>bytes</th>
<th>Flows</th>
<th>time</th>
<th>protocol</th>
<th>in/out</th>
<th>unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table Protocol</td>
<td>protocol</td>
<td>pkts</td>
<td>bytes</td>
<td>flows</td>
<td>time</td>
<td>protocol</td>
<td>tcpflag</td>
<td>in/out</td>
<td></td>
</tr>
<tr>
<td>Table Topuser</td>
<td>srcip</td>
<td>dstip</td>
<td>pkts</td>
<td>bytes</td>
<td>Flows</td>
<td>Time</td>
<td>in/out</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Table Nexthop</td>
<td>input_if</td>
<td>output_if</td>
<td>pkts</td>
<td>bytes</td>
<td>Nexthop</td>
<td>Flows</td>
<td>time</td>
<td>in/out</td>
<td></td>
</tr>
<tr>
<td>Table Asmatrix</td>
<td>input_as</td>
<td>output_as</td>
<td>srcport</td>
<td>dstport</td>
<td>Pkts</td>
<td>Bytes</td>
<td>Flows</td>
<td>nexthop</td>
<td>time</td>
</tr>
</tbody>
</table>
4. Conclusion

In this paper, we have proposed, developed, deployed, and tested our own software, which is an extension of the widely-used tool FlowScan with additional DBMS support and Web-based interactive query interface. Future work include further database and query performance optimization,
visualization of query results (in a more graphical way; currently only in a form of tables), platform independence (currently working only on FreeBSD with MySQL), and detection of application flows that passes through dynamically allocated ports, such as those of P2P applications [4, 5, 6, 8].

5. Acknowledgements

This research was supported by a 2013 Research Grant from Sangmyung University.

6. References