NetFlow: Information loss or win?

Robin Sommer and Anja Feldmann
Saarland University, Saarbrücken, Germany
{rsommer,anja}@cs.uni-sb.de

I. Introduction

Operating a network without accurate traffic statistics is not desirable. Commonly used data sources are SNMP [1], flow-level data like Cisco's NetFlow [2], and packet level data. The first data source provides low volume, coarse-grained, non-application specific data. The latter one provides high volume, fine-grained data and application specific information. NetFlow[1] lies somewhere in between both in terms of volume and in its level of detail.

In this work we ask the question, how much information do we lose with NetFlow compared to packet level traces, and how much information do we gain with respect to SNMP-like data. More specifically, we are interested in the generation of TCP connection summaries and SNMP-like byte count aggregations based on NetFlow data, and we present approaches for solving both problems.

Due to specific constraints introduced by Cisco's informal and ill-specified definition of NetFlow, it is not obvious that we are at all able to hope for deriving accurate results from NetFlows. Therefore, we carefully evaluate our approaches by comparing results from NetFlow to data gathered from full packet traces. Our data sets consist of NetFlows and packet data collected simultaneously at Saarland University, Germany, in March and April 2002.

II. Analysis of TCP Connections

One common use of packet level data is to extract TCP connection summaries. Therefore we first examine how accurately one can infer TCP connection summary information from NetFlows. TCP connection summaries are typically computed using tools such as TCP-Reduce [3] or Bro [4] on packet traces. These tools simulate the TCP state machine of the endpoints and therefore represent the most accurate way of generating summaries. In contrast, NetFlows only provide a very limited view of the actual network traffic: They are uni-directional, already aggregate packets, and their time constraints are only informally defined [2]. We overcome these limitations by building a NetFlow post-processor called Flow-Reduce. It first combines all NetFlows which contribute to a particular TCP connection. In a second step Flow-Reduce is generating all information usually provided by traditional packet-based tools, including the originator of the TCP connection, the duration, and even the TCP state.

Although Flow-Reduce naturally has to rely on a set of heuristics, it performs remarkably well. We verify it by comparing its output to the summaries generated from packet traces. We see that Flow-Reduce produces very close approximations for over 90% of all connections. In particular, NetFlows corresponding to Web traffic seem to be perfectly suited for this post-processing step with an agreement range of over 97%.

But our detailed analysis of mismatches between flow-level and packet-level summaries reveals several problems as well: the router may aggregate packets from several TCP connections into one NetFlow; NetFlows do not allow us to differentiate between transmitted data and goodput; and NetFlows' limited time resolution (only ms) can lead to associating a NetFlow with an incorrect TCP connection. But only the first problem emerges as a significant factor. It implies that we end up with fewer NetFlow connections than TCP connections. This is especially the case for connections initiated by file sharing applications like Gnutella. It occurs much more rarely for other, more traditional, applications such as Web, telnet, and FTP. The reason is that file sharing clients often need several connection attempts using the same socket within a few seconds before a connection is established.

III. Analysis of Byte Counts

One common use of Cisco's NetFlow is to extract application specific byte counts[2] of the traffic passing through a router. These values may, e.g., be used for accounting [5], [6] or visualization [7]. While SNMP data provides byte counts only on a per router and per interface basis, NetFlows contain additional information that can be used, e.g., to aggregate counts per subnet or per applica-

[1]The term NetFlow is used for the concept and the individual records.
[2]The following discussion applies to packet counts as well.

tion. There is only one major problem with this approach: The time granularity of NetFlows is much coarser than that of SNMP-like data. After all NetFlow is already aggregating bytes. Recall that the duration of a NetFlow may be bigger than the SNMP time step granularity. Even if its duration is smaller than the time granularity a NetFlow may start in one time slot and end in the next one. If we want to derive counts at regular time intervals from NetFlows, we therefore have to find a way to distribute the bytes associated with a NetFlow over time.

The straight-forward approach for solving the problem is also the common one, e.g., taken by `FlowScan` [7]: The export time of a NetFlow determines the time slot to which all of its bytes are assigned. In addition to its simplicity another advantage of the approach is that it is well-suited for real-time processing: Only one time slot is updated for each new NetFlow, and its number is monotonically increasing. But obviously there is a tradeoff here between simplicity and accuracy. In particular, long-lasting, high-volume NetFlows carry a large fraction of the traffic and are the ones that are likely to be associated with the wrong time slot and therefore have the potential to introduce large errors. A more appropriate aggregation method may be to prorate the bytes of a NetFlow across all time slots that intersect the life-time of the NetFlow.

In order to evaluate the two methods, we generate SNMP-like byte counts from the corresponding packet traces. In addition we applied both aggregation techniques, as well as some others, to our data sets. We observe that prorating NetFlows matches the actual byte counts very well for all examined aggregation intervals (which range from 4 to 64 minutes). But aggregation by export time shows rather large errors for time intervals below 30 minutes. As an example, Figure 1 shows both aggregation methods applied to intervals of 8 minutes using our March data set.

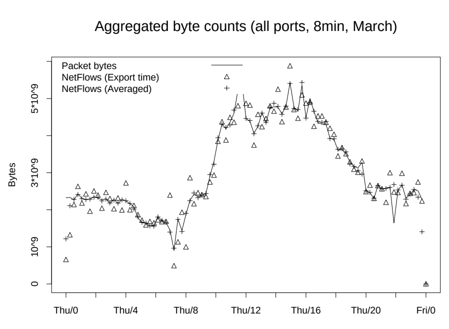

Fig. 1. Byte counts aggregated into 8 minute time slots

We draw two conclusions: First, there exists a method that can accurately aggregate NetFlows even for small time intervals. Second, the commonly used method needs some care in choosing an interval that is sufficiently large to compensate for the deficiencies of the method.

IV. Summary

We set out to gain insights into the capabilities of NetFlow by asking how and with what accuracy one can derive the same or additional information from NetFlow than from finer grained or coarser grained measurements.

While NetFlow obviously cannot provide the detail of packet traces, we do not have to abandon TCP connection summary information. We present a post-processor FLOW-REDUCE that overcomes the limitations of Cisco's unspecific and resource dependent definition of NetFlow. During the evaluation we identify some general limitations of NetFlow. Finally, comparing aggregated NetFlows to SNMP-like byte counts shows that we are able to get accurate results if we either prorate the byte counts over time or if we use large enough time intervals.

Overall we conclude that NetFlow, if used with care, is indeed a valuable tool for network measurement. While one loose some accuracy, the loss is often not significant if weighted against the reduction in measurement data or the information gain. To further understand the limitations and benefits of NetFlow we plan to reverse-engineer the exact process by which Cisco routers generate NetFlows using a dedicated router inside our testbed. Using this insight we plan to simulate the process using the packet traces in order to precisely characterize how often each source of error occurs. In general, our tool FLOW-REDUCE is not limited to Cisco's proprietary flow implementation but it either is already capable of dealing with other flow models, e.g., [8], or can easily be extended to do so.

References

[1] William Stallings, *SNMP, SNMPv2, SNMPv3 and RMON 1 and 2*, Addison-Wesley, 1999.

[2] Cisco Systems Inc., "NetFlow Services and Applications - White paper," `http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm`.

[3] "`tcp-reduce`," `http://ita.ee.lbl.gov/html/contrib/tcp-reduce.html`.

[4] Vern Paxson, "Bro: A system for detecting network intruders in real-time," *Computer Networks*, vol. 31, pp. 2435–2463, 1999.

[5] Mark Fullmer and Steve Romig, "The OSU flow-tools package and CISCO netflow logs," in *Proceedings of the Fourteenth Systems Administration Conference (LISA-00)*, 2000, pp. 291–304.

[6] Simon Leinen, "Flow-based Traffic Analysis at SWITCH," Poster at PAM2001.

[7] Dave Plonka, "FlowScan: A network traffic flow reporting and visualization tool," in *Proceedings of the Fourteenth Systems Administration Conference (LISA-00)*, 2000, pp. 305–318.

[8] Kimberly C. Claffy, Hans-Werner Braun, and George C. Polyzos, "A parameterizable methodology for Internet traffic flow profiling," in *IEEE Journal on Selected Areas in Communications*, vol. 13, no 8, pp. 1481–1494, 1995.