①

Keywords: *Data, Network, Statistics, Analytics, Big Data, Acceleration, Acquisition*

Compressing Test and Evaluation by Using Flow Data for Scalable Network Traffic Analysis

Kevin Buell, Mustafa G. Baydogan, Burhan Senturk, and James P. Kerr

The specialized nature of technology-based programs creates volumes of data on a magnitude never before seen, complicating the test and evaluation phase of acquisition. This article provides a practical solution for reducing network traffic analysis data while expediting test and evaluation. From small lab testing to full integration test events, quality of service and other key metrics of military systems and networks are evaluated. Network data captured in standard flow formats enable scalable approaches for producing network traffic analyses. Because of its compact representation of network traffic, flow data naturally scale well. Some analyses require deep packet inspection, but many can be calculated/approximated quickly with flow data, including quality-of-service metrics like completion rate and speed of service.

« Image designed by Diane Fleischer

Challenges for Acquisition

With two major conflicts coming to an end—Iraq in late 2011 and the expected end of U.S.-led combat operations this year in Afghanistan—it comes as little surprise that budgets throughout the Department of Defense are entering an age of austerity. The earlier enacted across-the-board federal spending cuts, known as sequestration, claimed a percentage of the Defense Department's budget. Despite these budget cuts, the expectation that defense acquisition professionals will field technology-based systems to the warfighter is at an all-time high. Low-intensity conflict operations throughout the world rely heavily on technology and intelligence systems.

Complicating the acquisition of these technology-based systems and programs is the voluminous amounts of data they produce, as observed at the Army's recurring large technology test event, the Network Integrated Evaluation (NIE). NIE produces terabytes of network data in a single day; this amount of data is simply too large to manage and far too large for test and evaluation (T&E) professionals to efficiently analyze the data. Processing a single data set can take as long as 24 to 36 hours; the status quo is grossly inefficient to meet the needs of rapid acquisition.

Because of these inefficiencies, meaningful and effective engineering modifications performed during a test event cannot be done fast enough. The delay between analysis, engineering modifications based on data, and validation can extend the test event—or worse, necessitate a follow-on event. Both scenarios require a longer T&E phase whether in developmental test, operational test, or integrated test, which impacts schedule and cost by extending the T&E phase of the acquisition life cycle. Moving programs that involve complex information and communication networks from the Technology Development Phase (Milestone B) to the Engineering & Manufacturing Development phase of the Integrated Defense Acquisition, Technology and Logistics Life Cycle Management System (Cochrane & Brown, 2010) is notoriously difficult for a variety of reasons, not the least of which is inefficient methods of handling T&E data.

Engineers from the U.S. Army Electronic Proving Ground, with their academic partners at Arizona State University's Security and Defense Systems Initiative, developed a way to compress some of the T&E timeline for defense technology systems that use networks. This partnership realized nominal gains of 75 percent, reducing analysis time from 24 hours to as little as six for data sets of approximately two terabytes. The efficiency gains are a sum of statistical and probabilistic modeling, data reduction, and the use of commercial-off-the-shelf software (COTS) for analyzing network traffic. These gains are a partial answer to the challenge introduced at the beginning of this article: that defense acquisition professionals must manage to field capable technology systems to the warfighter.

Background

Analysis of network traffic has been studied for some time, and a well-established body of research exists on Internet measurement (Crovella & Krishnamurthy, 2006). However, evaluation of networks within military test exercises has some unique characteristics not shared with general Internet measurement. For example, while Internet measurement is largely focused on collecting data at routers, military test exercises often focus more on collecting data at end-nodes. In some ways, this is a luxury enabled by the contained evaluation environment, which allows for accurate point-to-point metrics like speed of service.

On the other hand, collecting data at end-nodes can account only for traffic seen on those nodes. The network may not be large enough for router-based measurements to be useful, so end-node collection may be the only option. This unique environment is not well explored in the current literature on Internet measurement.

Military test exercises present further challenges. Because of the nature of field exercises, harvesting data may not occur as frequently as test officers would like. Interfaces and protocols may not be in place to gather test data, and it often must be copied and physically carried from nodes under test. These unique aspects can lead to data collection inconsistencies and errors.

Nevertheless, providing access to measurement results is essential for providing information that may affect ongoing testing. For example, determining whether a data collection system is working, whether a node is active, and how much data has been collected are all important to know as soon as possible during testing.

As the amount of data flowing over networks increases, the ability to collect, transfer, store, process, and analyze the data becomes more challenging. Many tools and standards from the Internet measurement community can be useful here. Where possible, using open-source software is preferable, thereby decreasing costs and avoiding "reinventing the wheel." Using standard data formats is pivotal to knowledge transfer and tool interoperability. When compared to solutions developed in-house, the open source solution often offers greater performance, and a larger support community, and it is typically far less costly.

As the amount of data flowing over networks increases, the ability to collect, transfer, store, process, and analyze the data becomes more challenging.

Traffic Data Formats

To record network traffic, two main formats are generally available: packet capture and flow. Packet capture formats record everything that goes across the wire (each individual packet, including header and data). Flow formats summarize the traffic and exclude the content. Excluding the content sacrifices the ability to reconstruct true network traffic. However, focusing on flow formats allows for important metrics about the network traffic—like performance, quality of service, and loss—but alleviates the burden of fully constituted packets.

Flow formats are attractive because working with large sizes of data presents bottlenecks, particularly with disk reads and writes. Smaller data sizes result in faster analysis performance. The most popular flow format has been NetFlow from Cisco, but it has been standardized into a nonproprietary flow format called Internet Protocol (IP) Flow Information Export, or IPFIX (Internet Engineering Task Force, n.d.), and several other flow formats are precursors or variations on the flow concept.

Flows normally summarize traffic by recording the number of packets, total bytes, flags, protocols, and other elements over some time period (e.g., 1 minute) from a source IP address to a destination IP address. These are referred to as aggregated flows. An alternative is to produce one flow per packet, referred to as single-packet flows. These enable some quality-of-service and other advanced analyses discussed later.

Some tools also work natively with flow data that have been compressed using standard compression algorithms. Table 1 represents a sample calculation based on various observations when working with flow data. Exact numbers will vary based on traffic characteristics, but this gives some idea of the significant data reduction achieved when using flow data, which in turn eliminates the disk throughput bottleneck.

TABLE 1. SAMPLE FLOW SIZES FOR A GIVEN SET OF PACKET CAPTURES

	Packet Capture	Single-Packet Flow	Single-Packet Flow (Compressed)	Aggregated Flow	Aggregated Flow (compressed)
Size	500 GB	38 GB	2 GB	0.25 GB	0.04 GB

Data Preparation

Network traffic data must be collected and prepared for analysis to support real-time queries and in-depth discovery. Ideally, traffic data are collected natively in flow format (such capabilities are built into most routers). Since some applications require full packet contents, other approaches may be required, such as capturing full packet data and producing flows from it or capturing both full packet and flow data simultaneously.

Reading packet capture data is normally bounded by disk I/O (input/output), but writing it to the significantly smaller flow format is generally not. Optimized open source tools like Yet Another Framework, or YAF (Software Engineering Institute, 2006), convert standard Libpcap-formatted packet captures (Libpcap is a portable library for network traffic capture) to IPFIX format. Conversion time is largely a product of disk read speed. For example, on a current commodity system with

100MB/s read speed, conversions performed from packet capture to flow took about 11 seconds/GB. This means that converting 500GB of packet capture would take about 1.5 hours.

Generally, two approaches are used to produce a final set of flow data. For analyses that do not compare traffic between two traffic collection points (e.g., traffic load, protocol distribution, topology), the following steps are required:

1. Convert all packet capture files to aggregated flow files.

2. Combine the aggregated flow files into a single aggregated flow file.

3. Deduplicate the single aggregated flow file.

The final flow file is deduplicated to avoid double counting traffic seen at a source and destination traffic collection point. An accurate (but approximated) deduplication process is available in some flow-based tools and is accomplished by matching flows based on time, protocol, bytes, and so on within a configurable threshold.

For analyses based on matching packets between traffic collection points (e.g., completion rate, speed of service), the following steps are required:

1. Convert all packet capture files to single packet flow files.

2. Combine the single packet flow files producing one flow file per traffic collection point.

The resulting files are not deduplicated to enable matching of traffic between files for these types of analyses.

Traffic Analysis

An extensive search and evaluation of open source network traffic analysis tools yields several that are particularly noteworthy and useful. Of course, Wireshark and tshark are popular tools providing packet inspection capabilities (Wireshark, n.d.), but their performance in many respects is lacking, particularly for processing many large files. A library called libtrace provides optimal results for working with packet capture files (WAND Network Research Group, n.d.). Argus is another open source tool that provides significant functionality as well as a proprietary flow format that includes some additional information that could be useful for characterizing traffic (QoSient, 2014).

One noteworthy tool that is well-supported, highly optimized, and contains all the basic functionality one would expect for data preparation and traffic analysis is SiLK (Software Engineering Institute, 2006). Traffic analyses enabled by aggregated flow files using SiLK queries include (but are not limited to):

- Topology by generating lists of source/destination pairs;
- Finding all traffic communicating on a given port;
- Separating nodes or pairs into bins based on percentage of total load; and
- Configurable filtering and traffic identification based on any combination of port, protocol, IP address, flow start/end/duration, etc.

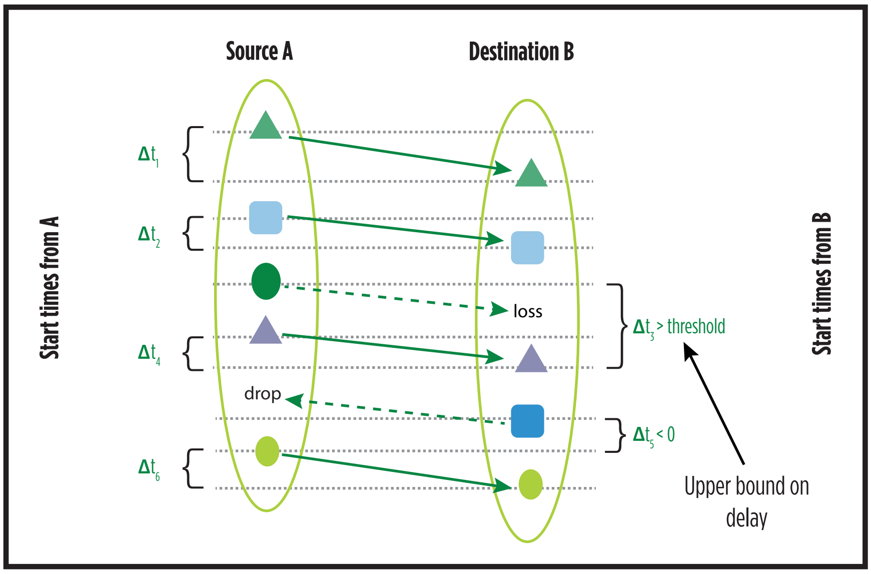
Visualization is not the focus of this article, but two open source tools should be mentioned here. The Ozone Widget Framework (Next Century, n.d.) has proven to be very useful. Also, an extensive, clean, and optimized JavaScript library for visualizing many types of data can be found in D3–Data Driven Documents (Bostock, 2013).

Quality of Service from Flow

Two essential metrics of network traffic are delay and loss. Delay is sometimes referred to as speed of service and loss is sometimes measured by completion rate. Given two files of single-packet flows representing two traffic collection points—each of which is associated with one or more nodes—the traffic loss between the two is determined by simply adding the number of packets in the file representing the destination, and then subtracting the number of packets in the source.

Calculating delay is generally more complicated and subject to some error with flow data. Using files from two traffic collection points, packets can be matched based on characteristics like size and protocol, as well as timestamp within a given threshold as shown in Figure 1. The time-stamps for a given packet will be different in the source and destination precisely because of delay.

FIGURE 1. DELAY AND LOSS BETWEEN SENDER AND RECEIVER



For a given source IP and destination IP pair, reading all single packet flows into memory and matching packets is feasible. To calculate the speed of service, simply subtract the timestamp of the matched packet in the source from that in the destination, and then average the difference over all the matched pairs. This is not matching packets based on content, and working with a timestamp threshold for matching will produce some error. However, packet matching with only flow data—when compared with matching using full packet content—provides nearly identical results.

Two essential metrics of network traffic are delay and loss.

In fact, the average delay over some number of packets (versus delay for an individual packet) can be calculated accurately using a number of approaches. Though lost packets must be accounted for, matching each packet precisely is not essential to produce an accurate average. Instead, ensure that packets are simply matched (uniquely) with some nearby packet in the source and destination. Mathematically, this is because (b1 – a1) + (b2 – a2) is identical to (b1 – a2) + (b2 – a1). In fact, and if it appears more computationally feasible, an alternative is to add all timestamps over a given period from the destination, and then subtract the corresponding timestamps from the source, i.e., (b1 + b2) – (a1 + a2). After dividing by the number of packets, the resultant average delay is the same as exact pairwise matching.

Since flow formats generally provide for some extensibility, one approach to improve matching accuracy is to compute a reasonable, locally unique hash value per packet based on its contents. This hash value is stored as additional information within the flow record. Indeed, flow extensibility can be an important means of bridging the gap between full packet capture and flow data by allowing for small, but critical pieces of data from some packets to be stored with flow records.

Future Work

Some analyses, which are now only accomplished using packet capture data could be accomplished using flow data, but may require some statistical analysis or algorithmic development. For example, reporting quality of service based on message type (e.g., voice versus video versus Web document) might normally require deep packet inspection. However, a flow format could save an indication of message type when converting from packet capture to flow, using extensibility. Also, traffic application mining has been studied extensively and could be used on flows to add some information about message/application type.

Many statistical approaches are available that could provide value in this domain. Finding cause/effect of poor network conditions, finding anomalies, and other problems could be solved with statistical data mining approaches. For example, low quality of service may be caused by many factors including high traffic volume (and associated congestion), proximity of sender and receiver, or physical conditions such as obstacles in the path between sender and receiver.

Advanced analyses may also highlight where significant events occur and which nodes are involved in these events. They may include ways to group and visualize nodes beyond standard clustering like logical topology and geographical display. Node groups may instead be formed based on traffic profile matching and/or quality-of-service similarities.

Conclusions

As the volume of network traffic data increases, analysis of military systems and networks becomes more challenging. Flow data have been used by the general Internet measurement community for some time to enable scalable traffic analysis for cyber security and traffic engineering. However, flow data have not been widely applied to metrics requiring more precision and advanced analytics. Military exercises are unique in that they are generally a smaller, contained set of traffic, often utilizing end-to-end measurements.

Flow data are much smaller than full packet captures and thus address the I/O bottleneck common in data processing for network measurement. Compressed flows provide even more data minimization. Common flow formats allow for some extensibility so that essential pieces of payload can be kept within a flow when needed.

Traffic flow data directly enable basic traffic characterizations like load and protocol distribution, and these are handled well by open source tools that work with flow data. More advanced analyses like quality of service are also possible using flow data and flow tools with additional algorithmic support. By capturing flows composed of a single packet and matching these flows at source and destination, delay and packet loss can be calculated accurately using flow data.

Statistical approaches could be used for even more high-level traffic characterization such as cause/effect analysis of network traffic conditions. Advanced analyses may also highlight where significant events occur and which nodes are involved in these events. Grouping nodes based not only on logical topology and physical location, but also traffic pattern similarity would also provide additional understanding and enhanced visual analytics. These provide greater insight into network traffic data that can point operators to potential areas for further analysis.

These approaches to increase the efficiency of network traffic analysis are small, but indicative of the trend to meet the big data problem. The length of T&E for technology-based programs will continue to grow as the complexity and interdependencies of these systems grow. The challenge of big data can be mitigated through incremental improvements. Using sensible, pragmatic methods that reduce the data—through statistical and probabilistic modeling coupled with the acceleration of analysis by adopting COTS—is one way to manage this big data challenge. Regardless of the challenge, defense acquisition professionals must look for new ways to enable our increasingly technology-enhanced warfighter.

Author Biographies

Dr. Kevin Buell is a research scientist at Arizona State University Research Enterprise (ASURE) in Scottsdale, AZ. His work focuses on network analysis, cyber security, and software engineering. He received his PhD in computer science from Arizona State University and has an extensive background in software engineering for the defense industry. Dr. Buell is a member of the Institute for Electrical and Electronics Engineers Computer Society as well as ACM's Special Interest Group on Software Engineering.

(E-mail address: kevin.buell@asu.edu)

Dr. Mustafa Baydogan is a postdoctoral Fellow at Arizona State University's Security and Defense Systems Initiative. He received his PhD in industrial engineering from Arizona State University and specializes in data mining and statistical process control. Dr. Baydogan is an active member of the data mining section of the Institute for Operations Research and the Management Sciences.

(E-mail address: mustafa.baydogan@asu.edu)

Mr. Burhan Senturk is a master's student in the School of Computing, Informatics, and Decision Systems Engineering at Arizona State University. He received his BS in computer science from Middle Eastern Technical University. Mr. Senturk's work is focused on data mining and network traffic analysis.

(E-mail address: muhammet.senturk@asu.edu)

Mr. James P. Kerr is a computer scientist at the U.S. Army Electronic Proving Ground at Fort Huachuca, AZ. He specializes in instrumentation development and support for network test and evaluation. He is currently working on next-generation tools and infrastructure for large-scale test events. Mr. Kerr received his MA in mathematics at San Diego State University.

(E-mail address: james.p.kerr.civ@mail.mil)

References

Bostock, M. (2013). *Data-driven documents* [JavaScript library Web site]. Retrieved from http://d3js.org

Cochrane, C., & Brown, B. (2010). *Integrated defense acquisition, technology and logistics life cycle management system* [Training aid chart]. Fort Belvoir, VA: Defense Acquisition University Press.

Crovella, M., & Krishnamurthy, B. (2006). *Internet measurement: Infrastructure, traffic and applications.* New York: John Wiley & Sons.

Internet Engineering Taskforce®. (n.d.). *IP flow information export (IPFIX).* Retrieved from http://datatracker.ietf.org/wg/ipfix/charter

Next Century Corporation. (n.d.). *Ozone Widget Framework* [Web software organization/access Web site]. Retrieved from http://owfgoss.org

QoSient. (2014). *Argus* [Network audit record generation/utilization Web site]. Retrieved from http://www.qosient.com/argus

Software Engineering Institute. (2006). *CERT NetSA security suite: Monitoring for large-scale networks.* Retrieved from https://tools.netsa.cert.org

WAND Network Research Group. (n.d.). *Libtrace* [Trace processing Library Web site]. Retrieved from http://research.wand.net.nz/software/libtrace.php

Wireshark. (n.d.). *Wireshark* [Network protocol analyzer Web site]. Retrieved from http://www.wireshark.org