Processing and Analysis of Large Data Sets Using High Performance Computing: Beyond Experimental Data

Brian Panneton[1], Brian Henz[2], Pritesh Patel[3], James Adametz[4]

[1]Technical and Project Engineering, Army Research Laboratory, Aberdeen Proving Ground, MD, USA
[2]Simulation Sciences Branch, Army Research Laboratory, Aberdeen Proving Ground, MD, USA
[3]Ad hoc Research Associates, Aberdeen Proving Ground, MD, USA
[4]QED Systems LLC, Aberdeen Test Center Test Technology Directorate, Aberdeen Proving Ground, MD USA

Abstract - *Efficiently processing and analyzing big data is gaining importance within the military setting. As systems-of-systems evaluation produces an expanding set of data in terms of volume, source, and range of types, the processing and analysis becomes more complex. The Army Research Laboratory (ARL) and Aberdeen Test Center (ATC) previously partnered to employ High Performance Computing (HPC) to address this problem. This paper describes the expansion of the distributed processing framework which now allows processing and analysis of data from simulation and emulation in addition to live experimentation and real-world sources. New modules have been added to the distributed framework to enable the processing of data from a variety of sources, which allows for a more in-depth analytical view of the event.*

Keywords: Parallel data reduction, Network Analysis, High Performance Computing, Emulation Data Reduction

1 Introduction

Communication systems deployed on the battlefield are critical for tactical operations. These systems must maintain stability and usability within challenging environments, more so than in the commercial field. The systems must go through stringent verification and validation testing before being deployed. In order for this process to occur, large-scale experimental field tests are conducted daily, recording multiple terabytes of many types of data, including network, geospatial, video, and temperature. The experimental data must then be reduced and restructured into a usable data model for analysts to examine and determine the success of the systems under test.

Dealing with a month-long event means the data size scales upwards quickly. Typically, analysts and evaluators want data in a queryable form ready by the next morning. Next-day turnaround time for processing and analytics is crucial, allowing problems to be addressed quickly to prevent further loss of required information. A collaboration between the Army Research Laboratory (ARL) and the Aberdeen Test Center (ATC) produced a distributed, python-based data reduction framework (Data Parser) which runs on High Performance Computing (HPC) resources at ARL's DoD Supercomputing Resource Center (DSRC) [1]. This system is able to achieve the necessary results in a reduced time frame, often in less time than the required time-table for the experimental test.

The Data Parser framework was designed to allow virtually any type of data to be processed in a scalable and parallel way. Originally, the focus was on tactical communications data, but as test types and evaluation goals changed, the framework evolved. The addition of Controller Area Network (CAN) [2], Analog/Digital sensor, video, and other types of data allowed for many new types of analysis to be performed. Newer tests such as Joint Land Tactical Vehicle (JLTV) [3], Autonomous Mobility Applique System (AMAS) [4] and various robotics systems are evaluated based on how the vehicular system interacts with the operator and the environment, more than how the vehicular communications system performs.

Currently, deployed systems are validated and verified by analyzing data from experimental results. However, many of these deployed systems have software models that can be studied within simulation and emulation environments. Comparing data from software models and deployed systems allows simulation and emulation testing to aid in design for future live experimentation and real-world events. The Data Parser has been updated to allow cross-validation between the models in the virtual realm and the physically deployed systems. Simulation and emulation drastically reduce costs in comparison to live

experimentation. Proving virtual tests using models are comparable to the physical tests will open the way for more testing to be conducted in a controlled and repeatable virtual environment. The tester is able to manipulate the tests in ways that are cost prohibitive or otherwise difficult to achieve in physical tests.

The following sections will detail the current advancements and capabilities of processing live experimental data, the work required to transition from processing experimental and real-world data to processing emulation data, the current state of analytics for both emulation and experimentation and finally where this work is heading in the future.

2 Capabilities and advancements

This section details the current state of the Data Parser as well as additions that have been made to it. Due to the initial design choices, the Data Parser framework is easily adaptable to new projects. As the requirements change the Data Parser is able to grow to accommodate them.

2.1 Framework overview

Processing big data in an HPC environment is slightly different than processing on commodity hardware. The HPC clusters generally use a large, highly optimized, distributed file system. This allows easy access to the data from each node and reduces the complexity of moving data. In addition, the HPC environment is meant to be shared among a variety of projects and problem spaces, so a scheduler, such as PBS,[1] LSF[2], or Slurm[3] is employed to queue and manage the system. Taking this into account, most modern big data frameworks will not run efficiently within an HPC environment. The development of a custom data processing framework allows data processing to be run on such a system.

The Data Parser Framework was focused on portability between HPC environments, scalability and ease of development for the user. The Message Passing Interface (MPI) is used for all Data Parser communications. Due to the nature of the shared disk infrastructure, control messages, rather than data results, are generally sent over MPI.

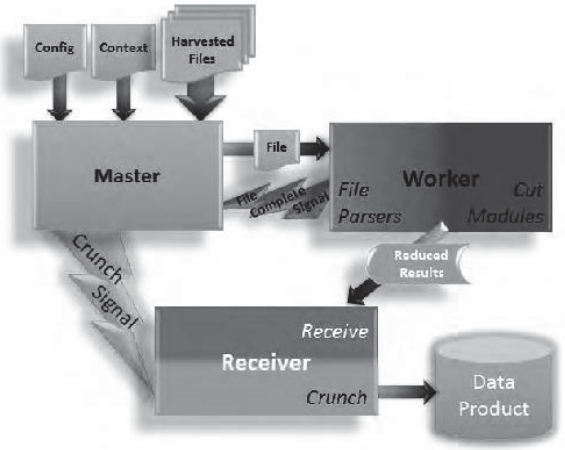

Figure 1 High Level Data Flow in Data Parser

The framework was built using a distributed producer-consumer paradigm. Figure 1 gives a high level view of how the Data Parser processes data. First, the Master parallelizes and distributes the input data to Workers, or MPI rank dedicated to data processing, on a per file basis. On each Worker there are File Parsers and Cut Modules, or producers and consumers respectively. The File Parsers read and break up each file into subsections of data called cuts. Cut Modules subscribe to certain types of cuts and receive them as they are parsed from the file. Cut Modules organize and reduce the data based on what results a user is looking for. These results can then be sent to a Receiver. A Receiver is a Worker dedicated to data aggregation for one Cut Module. The described data flow is referred to as the Process Stage. After the Process Stage, the data can once again be parallelized and reduced in what is referred to as the Crunch Stage. This second stage is where post-processing tasks occur, such as producing graphs or tables for later use. The combination of one Process and one Crunch stage make up a Phase. Phases can depend on each other forming logical data workflows. A more in depth overview of how the framework works can be found by reading the ARL Technical Report titled *High-Bandwidth Tactical Network Data Analysis in a High-Performance-Computing (HPC) Environment: HPC Data Reduction Framework* [5]

1 The Portable Batch System (PBS) is a job scheduler and resource allocation system for HPC created by Altair.
2 The Platform Load Sharing Facility (LSF) is a job scheduler and resource allocation system for HPC created by IBM.
3 The Simple Linux Utility for Resource Management (Slurm) is a job schedule and resource allocation system for HPC created through a collaborative effort between Lawrence Livermore National Laboratory, SchedMD, Linux NetworX, Hewlett-Packard and Groupe Bull.

The amount of time needed to reduce and process data is highly dependent on the type of input, the number of consumers and the size of the HPC cluster. The typical input files produced from testing at ATC are about 1/2 GB in size with the aggregate size being around 1-2 TB. The average test normally runs 25 Cut Modules over 6 phases. In the pre-HPC reduction methods, the reduction and processing time for each terabyte of data took approximately 60 hours. Using the Data Parser on the HPC system with 256 cores has reduced this time frame to under 10 hours. Scaling has allowed processing of data above 40 terabytes using 1024 cores, however this was with a smaller set of Cut Modules. Scalability above this range has not been tested since larger data sets are less frequent.

2.2 Current advancements

The data processing framework has undergone several advancements to enhance its capabilities. New modules have been developed to process additional types of data allowing for more robust cross-correlations and better insight into the observations from any given experiment. Adding new processing to the framework to combine virtual, experimental and real-world data allows for a more in-depth study of the systems under test. In general, new Cut Modules are built independent of the system under test allowing for reuse of the same processing pipelines.

2.2.1 Transition to emulation data

ARL and US Army Communications-Electronics Research, Development and Engineering Center's (CERDEC) Space and Terrestrial Communications Directorate (S&TCD) both use a modular framework called the Extend-able Mobile Ad-hoc Network Emulator (EMANE) to support Mobile Ad-hoc Network (MANET) emulations. Emulation provides real-time modeling of the data link and physical layers for MANETs in order to predict connectivity and provide a testbed for analyzing network protocols and applications that can be run unmodified in the emulation environment [6]. The emulation testbed provides a repeatable virtual test environment. This repeatability is not realizable under live exercise conditions. Configuration of a MANET emulation experiment includes provisioning a virtual machine, linux container (LXC), or physical hardware for each networked platform or device, configuring and executing daemons (EMANE, GPS, etc), and configuring the communication network. Post emulation analysis of the systems under test requires real time collection of data from multiple processes and collection points within the testbed. Having control of the testbed and its configuration, coupled with knowledge of how the data will be processed within the Data Parser, allows the data to be collected in a way that is easy to ingest for later analysis.

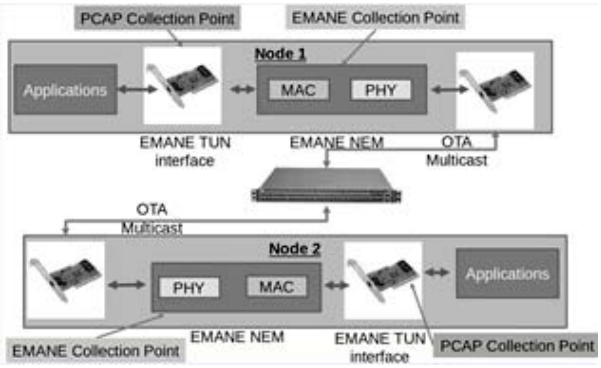

Figure 2 Data Collection Points from MANET Emulation

2.2.1.1 Configuration and data collection from MANET emulation

In Figure 2, the emulation testbed configuration is illustrated with data collection points identified. The collection points are located at the EMANE TUN[4] interface to collect packet capture (PCAP) data. These locations capture all of the ingress and egress traffic to and from applications and traffic generators running on the emulated devices. Depending on the radio model, the captured data may include routing messages for daemons such as the optimized link state routing (OLSR) daemon. These messages may be absent from the packet capture if the routing is configured in the EMANE radio model. The PCAP collection point will capture all UDP and TCP traffic that is terminated or generated by the applications for

```
tcpdump -p -n -B 8192 -x -s 0 -i emane0 -C 500 -Z root
        -w ${TEMP_OUTDIR}/node-${NODEID}-emane0-inbound.pcap inbound
tcpdump -p -n -B 8192 -x -s 0 -i emane0 -C 500 -Z root
        -w ${TEMP_OUTDIR}/node-${NODEID}-emane0-outbound.pcap outbound
```

Figure 3 Sample TCPDUMP Script

[4] A TUN is a software defined virtual network interface [11]

EMANE ”shared code” models, including waveform applications such as routing in the EMANE Network Emulation Modules (NEM). The EMANE Over-the-air (OTA) Manager Channel is not recorded since it is the back-haul network for EMANE itself. Packets are sent and received faster than the configured latency on this channel.

The EMANE collection point can be provided data through the TestPoint software created by AdjacentLink LLC [7]. Testpoint probes the EMANE NEM at set intervals and records a predefined list of counters and tables from the running NEM.

2.2.1.2 Collecting PCAP data

The framework’s PCAP File Parser uses the Device-Tap- Direction (DTD) concept [8] in order to avoid knowing detailed configuration of the system under test. The DTD is relatively easy to create within the virtual machines or LXCs (virtual nodes). Each provisioned virtual node needs to be uniquely identified within the network and is generally identified within the hostname by a unique numeric identifier (ie: thufir-n001). This numeric identifier can be used as the device in the DTD format. Some tests may use network-specific identifiers such as the Virtual Large Area Network (VLAN) tag instead. The collection point, which is mainly used for packet matching within the Data Parser framework, is commonly set to the EMANE TUN interface. In some tactical scenarios, analysts may be interested in how different parts of the overall network topology are performing. Performance can be derived from capturing packet data on both sides of a network device such as an in-line encrypter. The collection point configuration enables unique matching patterns where multiple communication channels may overlap. The packet's direction, inbound or outbound, is determined using a heuristic such as matching the interface’s source Ethernet address or the packet’s destination IP.

Record packets on the system is generally done with tcpdump. The lines shown in Figure 3 separately record 500MB PCAP files using the DTD format. There is a limit set on the size of the PCAPs strictly because it makes processing the data faster within the Data Parser framework due to distributing the data on a per file basis. The 500MB size comes from the normal size of the experimentally collected data files and is not a restriction.

Collecting PCAP data can also be done with the Automated Performance Assessment Framework Innovation (APAFI) [12] sniffer which monitors inbound and outbound packets. The APAFI sniffer writes PCAP files but has other features to perform real-time analysis that may be wanted.

In addition to the APAFI sniffer, data collection can be done using the Riverbed AppResponse Xpert (ARX) appliance to capture all incoming and outgoing traffic between physical and virtual nodes.

Once the PCAP files are collected and in the DTD format, they can be loaded and run through the communications processing Cut Modules which look at the different network protocols in depth including IP, TCP, and UDP. For instance, the TCP modules allow full flow calculations from host-to-host rather than round-trip. The packet data can also be combined with other data types to better analyze the system under test.

2.2.1.3 Collecting GPS data

The majority of the EMANE-related tests rely on geospatial positioning in order to calculate the propagation loss of a network over terrain. EMANE generates time-synchronized mobility events which get fed into the GPS daemon running on the virtual machine. The current experimental data has GPS times, space and position information recorded once a second. To produce data in a similar fashion, a simple process is started that polls the GPS daemon on the virtual node once per second and outputs a comma separated values (CSV) file. The generated CSV files can be ingested by the CSV File Parser and the data can be analyzed.

2.2.2 Processing instrumented emulated systems

TestPoint allows the insertion of probes into an EMANE emulation or the surrounding environment in order to record details about the events that took place. This data is recorded in a database as a look-up table in addition to being saved as a raw data file. Two new Cut Modules and a File Parser were added to the Data Parser Framework to handle TestPoint data.

The Cut Module called TestPoint Emane-Loss collects and plots network statistics from the Physical Layer and Data Link Layer of the Open Systems Interconnection (OSI) model. Statistics collected include forward pathloss, signal-to-noise ratio, data-rate over time and queue overflows. These lower level statistics can greatly improve evaluation of a system-under-test and are generally difficult to record accurately within a live experimental test.

The Cut Module called TestPoint Cpu-Mem collects and plots system statistics from the virtual nodes. The focus of this module is CPU and Memory usage of the overall emulated hardware. This information allows an analyst to assess the performance of the system under test during planned events such as stress testing of the network.

The TestPoint File Parser has been created to parse the database look-up table, extract the raw output for individual Probe Messages, and convert them into cuts. The two new Cut Modules subscribe to these cuts. This automated process allows the Data Parser to be able to processes any new data types that TestPoint can generate.

2.3 Test analytics

The Data Parser framework was originally designed to accommodate the US Army's Network Integration Evaluation (NIE) tests. As detailed in the previous section, many advancements and improvements have been made. The following section gives a quick look into some of the other tests that use the Data Parser framework for their analytics.

2.3.1 Vehicular and robotics test analysis

The original purpose for the processing framework was to perform reduction and analysis of tactical network communications data and render a database model for the analysts and evaluators to derive and extract key performance metrics. With the success of this framework, ATC has decided to enhance its capabilities to include vehicular and robotics test processing. To that end, new modules are being developed to allow the system to interpret and render Controller Area Network (CAN [1]), analog sensors, JAUS [2], video sources, device log files and other types of vehicular data. These will provide new and rich sources of information that can be later correlated with each other.

Much of the vehicular test analysis focuses on reliability of systems and subsystems. Within a vehicle, there can be a multitude of devices that can malfunction. It is difficult to record the precise time of when a malfunction occurs and the malfunction can sometimes remain undetected until a manual inspection occurs. By adding CAN data to the list of information being processed, the behavior of subsystems within each vehicle can be observed as they communicate with each other. Quick determination of when a particular device issues specific error codes or stops communicating with its peers is done by examining the reduced data.

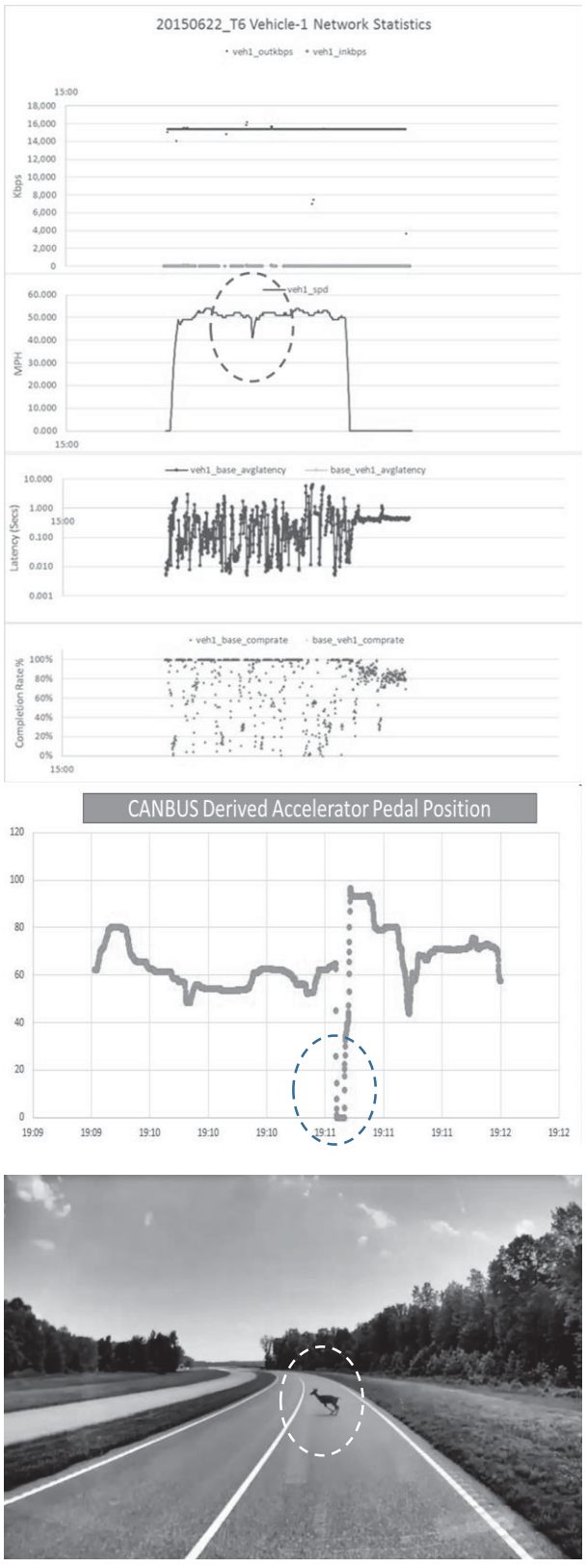

Figure 4 Cross-Domain Vehicular Anomaly Analysis

Future efforts related to robotics test and evaluation will employ a new JAUS module (currently under development). This new module will allow analysis of a vast amount of information on what a robotic entity is sensing and how it is reacting to its environment.

New video processing modules have been added which allow for indexing the video streams recorded from several sources into a queryable form that will allow the analysts and evaluators to quickly view pertinent video sources related to the vehicle at a specified time. Video processing that aides in object motion detection, and other computationally intensive video analysis techniques is also going to be incorporated.

ATC envisions adding new framework modules that will enable correlation over many sources to scan for anomalies in the data. This data can then be investigated in detail by the analysts to extract as much information as is possible for all sources available to support analytical objectives. A sample of this kind of cross-domain analysis is demonstrated in Figure 4. The network statistics charts (top section) includes a vehicle speed section (second line trace down from the top). The vehicle was supposed to be maintaining a relatively constant speed around the test track. The analysis indicates an unexpected momentary drop in vehicle speed (highlighted in the dotted circle). The CAN bus data (middle section of the figure) was queried for the vehicle at that time, and it was shown that the operator removed pressure from the accelerator (middle dotted circle). Querying the database for the corresponding video at the time showed exactly why the operator slowed down (a deer in his path).

2.3.2 MODESTA analysis

An up-and-coming capability from CERDEC S&TCD called Modeling, Emulation, Simulation Tool for Analysis (MODESTA) is being developed for system-of-systems experimentation, validation and assessments in a controlled lab environment. Analytical objectives of experimentations conducted using MODESTA are primarily driven to evaluate Tactical Applications and Waveform behaviors. In the current Army's tactical network architecture there are about six different radio waveforms, three different network security enclaves, numerous application hosts, and various network devices. With such layers of complex networks and devices, there are many data collection points. In order to evaluate the entire system-of-systems, all data collection points must be monitored to accurately analyze, evaluate, and troubleshoot system performance and integration issues.

MODESTA utilizes an updated version of the Command, Control, Communications, and Computers (C4) Data Model to store the resulting correlated data. The C4 Data Model was originally designed by Army Test and Evaluation Command (ATEC) and has been updated to enable storage of data generated from the diverse MODESTA environment. The MODESTA analysis team has leveraged the C4 Data Model to conduct deep dive analysis on multi-layered tactical waveform networks. For end-to-end system-of-systems performance assessments, it is important to track every packet as it traverses various ingress and egress collection points at all nodes in its path. Packet correlation of traversal events throughout large emulated virtual networks is computationally intensive. The smart hash-based pattern matching [9] performed in searching through the datasets at all virtual nodes reduces the complexity of end-to-end packet correlation. This end-to-end look enables analysts to deep-dive into performance-related issues. Having the ability to process the data in parallel on a large cluster also enables delivery of data products to analysts within a reasonable time frame.

2.4 Future ideas

The Data Parser updates have improved the framework's capabilities and now allows for processing of simulation and emulation data in addition to live experimental and real-world data. Though the framework works well in its current state, there are more improvements that are planned in the future. Reducing the memory footprint of Cut Modules and optimizing their algorithms will increase scalability. In addition, the possibility exists for a shift to a community-maintained big data ecosystem. Getting it to run well on an HPC cluster in comparison to commodity hardware would be the first hurdle to cross. Luckily, Lawrence Livermore National Laboratory's Magpie [10] is addressing this issue. The benefit would be a significant drop in maintenance and an increase in capabilities. Using a system that includes software such as Apache Spark allows for the use of utilities and libraries like its Machine Learning Library (MLlib) with minimal extra effort.

3 Conclusions

Having the capability to process large data sets of varying data types quickly and efficiently is important to the

Army. The analysis gathered from the tests helps validate and verify the systems in the field that assist the soldiers. Being able to follow the same processing pipeline on all data sources can drastically reduce the time and cost of testing new devices. The big-data field is growing rapidly and will continue to do so in the near future. The amount of data will only increase from this point on. Thus having reduction and processing methods in place will only be beneficial.

4 References

[1] B. Panneton, G. Besack, J. Adametz, B. Tauras K. Renard, "Processing and Analysis of Large Data Sets from High Bandwidth Tactical Networking Experiments Using High Performance Computing," *The International Test and Evaluation Association Journal*, vol. 36, no. 3, Sept 2015.

[2] CAN in Automation. (2016) CAN lower- and higher-layer protocols. [Online]. http://www.can-cia.org/can-knowledge/

[3] Andrew Feickert, "Joint Light Tactical Vehicle (JLTV): Background and Issues for Congress," LIBRARY OF CONGRESS CONGRESSIONAL RESEARCH SERVICE, Washington, DC, PDF CRS-RS22942, 2013.

[4] Joey Cheng. (2014, Sept) Defense Systems. [Online]. https://defensesystems.com/articles/2014/09/02/marines-army-amas-autonomous-convoys.aspx

[5] J. Adametz B. Panneton, "High-Bandwidth Tactical-Network Data Analysis in a High-Performance-Computing (HPC) Environment: HPC Data Reduction Framework," Army Research Labs, Aberdeen Proving Ground, Md, ARL-CR-0777, 2015.

[6] Naval Research Laboratory. (2016, Feb) Extendable Mobile Ad-hoc Network Emulator (EMANE). [Online]. http://www.nrl.navy.mil/itd/ncs/products/emane

[7] Steven Galgano, TestPoint Data Collection Framework, 2014, Rev. 1.2.

[8] B. Panneton K. Renard, "A Standard for Command, Control, Communications and Computers (C4) Test Data Representation to Integrate with High Performance Data Reduction," Army Research Laboratory, Aberdeen Proving Ground, Md, ARL-TR-7329, 2015.

[9] Panneton, Brian C; et. al., "High-Bandwidth Tactical Network Data Analysis in an HPC Environment: Packet-Level Analysis," Aberdeen Proving Ground, MD, 2015.

[10] Robin Goldstone, "Data Intensive Computing Solutions," Lawrence Livermore Nation Laboratory, http://computation.llnl.gov/projects/lc-big-data-leadership. [Online]. http://computation.llnl.gov/projects/lc-big-data-leadership

[11] TCPDUMP.ORG. (2015, Sept) TCPDUMP. [Online]. http://www.tcpdump.org/manpages/tcpdump.1.html

[12] Florian Thiel. (2000) Universal TUN/TAP Device Driver. [Online]. https://www.kernel.org/doc/Documentation/networking/tuntap.txt