ARL-TN-0765 ● July 2016

US Army Research Laboratory

Network Visualization Project (NVP)

by Terry Wen, Lisa M Marvel, and C Sean Morrison

NOTICES

Disclaimers

ARL-TN-0765 ● JULY 2016

US Army Research Laboratory

Network Visualization Project (NVP)

by Terry Wen
American Society of Engineering Education, Washington, DC

Lisa M Marvel
Computational and Information Sciences Directorate, ARL

C Sean Morrison
Quantum Research International, Inc., Bel Air, MD

REPORT DOCUMENTATION PAGE		*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* July 2016	**2. REPORT TYPE** Technical Note	**3. DATES COVERED (From - To)** 1 May–31 August 2015
4. TITLE AND SUBTITLE Network Visualization Project (NVP)		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Terry Wen, Lisa M Marvel, and C Sean Morrison		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) US Army Research Laboratory ATTN: RDRL-CIN-D Aberdeen Proving Ground, MD 21005-5067		**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TN-0765
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The purpose of the Network Visualization Project is to construct an interactive application for real-time playback of network activity based upon packet capture data. As data networks continue to expand as an integral part of modern information systems, the importance of the organization and clarity of said networks grows as well. The goal of this project differs from those of similar existing tools in its attempt to optimize both content and clarity, as the product is meant to maintain aesthetic appeal without sacrificing any details of the data meant to be portrayed. The project uses Dshell for back-end data retrieval and Processing for front-end presentation and construction of the application itself.

15. SUBJECT TERMS
computer network traffic, computer network security, computer network visualization, network traffic analysis, network forensics

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Lisa Marvel
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	UU	18	**19b. TELEPHONE NUMBER (Include area code)** 410-278-6508

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

1. Introduction and Background

As part of the ever-expanding age of information, data networking continuously grows in importance. With the quantity of data transferred across networks increasing daily, the monitoring of such networks becomes increasingly difficult. The Network Visualization Project (NVP) tackles issues regarding the display, monitoring, and inspection of network traffic and allows such data to be presented in an easily explored and understandable fashion.

NVP is useful for troubleshooting, application development, protocol analysis, and a myriad of other network analysis purposes. With other commonly available network visualization tools, such as WireShark,[1] users lack direct control of and interaction with the network data. They are simply presented with an itemized display of packet capture information. Other visualizers, such as Commetrix, tend to prioritize aesthetics with visually interesting illustrations that make it difficult or impossible to access technical detail. Tools that lack interactivity or limit access to technical detail are generally unusable for in-depth network analysis work. NVP aims to address these issues with the construction of an interface that more optimally balances content, interactivity, and clarity.

Because of the need for attention to detail, NVP is intended to monitor small-scale networks, watch for specific issues, and trace events in traffic. A primary goal of the project is to create an interface that can help locate potential issues, identify weaknesses in a network, and provide an understandable illustration of network communication that is adequate for a generally untrained eye. In accomplishing this, the project will provide a new tool interface supporting improved network analysis and network communication visualization.

2. Application Design

NVP consists of 2 parts: back-end data handling and front-end presentation. Data are first processed through Dshell, an open source packet-decoding tool developed by the US Army Research Laboratory.[2] These data, either streamed or input through a packet capture file (commonly called a “pcap” file), are output in JavaScript object notation (JSON) format. This JSON is provided as input to the front-end application of the project. This interaction of the user with the back-end interface allows for full manipulation of the data and the ability to directly filter data to the user’s needs.

The data are then processed by the front-end application, which is coded using the Processing programming language.[3] Here, the user is able to visually interact with a time line that encloses the duration of imported packet data. This playback presents the data visually through nodes and connections that can be interactively manipulated fully by the user or automatically arranged by the application. The primary attributes of the data shown are as follows:

- Source and Destination IP addresses
- Ports
- Protocols
- Time
- Packet length
- Other details in packet header (transmission control protocol (TCP) flags, etc.)

The visualization is meant to illustrate macro activity through a network as well as the individual details of individual packets. The interface will also include other utilities, such as playback tools and data filters.

The NVP process is shown in Fig. 1. Where network data is collected in a .pcap file. It is then transformed using Dshell into a JASON format. This is the input into the NVP processing module to construct the visualization.

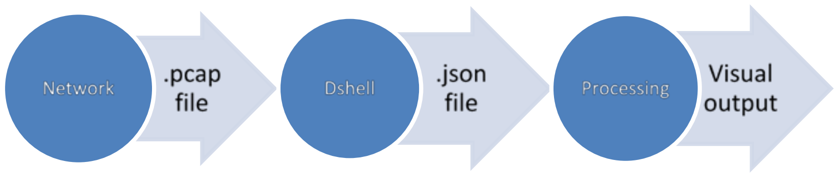
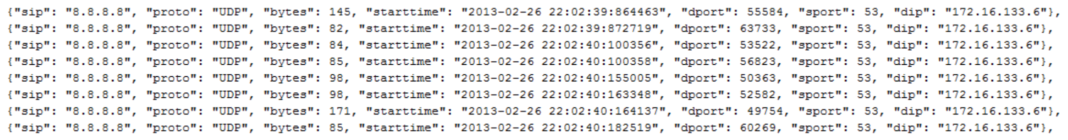


Fig. 1 Application data flow

2.1 Dshell: Back-End Data Handling

Dshell, or Decoder shell, is a command-line framework used for network forensic analysis. Dshell processes existing pcap files and filters output information based on both prebuilt and user-constructed modules. The Dshell portion of NVP involves the construction of an output module that will parse the necessary data from a pcap file into a JSON format file. Dshell allows the user to directly choose the modules to use for full control of the data portrayed in the visualization. Dshell provides a variety of tools and filters necessary to extract the desired information from pcap data and is flexible enough to adapt to a variety of network analysis goals. A sample of the JSON data file is shown in Fig. 2.



```
{"sip": "8.8.8.8", "proto": "UDP", "bytes": 145, "starttime": "2013-02-26 22:02:39:864463", "dport": 55584, "sport": 53, "dip": "172.16.133.6"},
{"sip": "8.8.8.8", "proto": "UDP", "bytes": 82, "starttime": "2013-02-26 22:02:39:872719", "dport": 63733, "sport": 53, "dip": "172.16.133.6"},
{"sip": "8.8.8.8", "proto": "UDP", "bytes": 84, "starttime": "2013-02-26 22:02:40:100356", "dport": 53522, "sport": 53, "dip": "172.16.133.6"},
{"sip": "8.8.8.8", "proto": "UDP", "bytes": 85, "starttime": "2013-02-26 22:02:40:100358", "dport": 56823, "sport": 53, "dip": "172.16.133.6"},
{"sip": "8.8.8.8", "proto": "UDP", "bytes": 98, "starttime": "2013-02-26 22:02:40:155005", "dport": 50363, "sport": 53, "dip": "172.16.133.6"},
{"sip": "8.8.8.8", "proto": "UDP", "bytes": 98, "starttime": "2013-02-26 22:02:40:163348", "dport": 52582, "sport": 53, "dip": "172.16.133.6"},
{"sip": "8.8.8.8", "proto": "UDP", "bytes": 171, "starttime": "2013-02-26 22:02:40:164137", "dport": 49754, "sport": 53, "dip": "172.16.133.6"},
{"sip": "8.8.8.8", "proto": "UDP", "bytes": 85, "starttime": "2013-02-26 22:02:40:182519", "dport": 60269, "sport": 53, "dip": "172.16.133.6"},
```

Fig. 2 Sample JSON data

2.2 Processing: Front-End Data Presentation

The Processing application is a graphic user interface (GUI) that first loads and processes the input JSON data into a visual representation. Loaded into memory, the JSON array contains network packet objects with packet information as attributes. It is loaded into memory via Processing's Java-based JSON input methods. The packets are then added into individual topological data arrays, representing the nodes and connections that they travel between and through during data transmission. A time line is also generated based on the timestamps of the imported packet data, and the packet data are accessed through the individual states of the time line.

The initial design of the GUI visualization used a direct animated visual for the individual packets of an exchange, allowing for a full illustration of the amount of traffic on a connection between nodes. However, multiple issues arose from this, including the inaccurate portrayal of travel time for the data packets, which generally travel faster than can be interactively visualized clearly. This initial version used a time line of visual states where the visualization directly reflected the time-line state, but packet transmission was visualized as an animation. The illustration ended up with inconsistencies, such as response packets being drawn while the initial packets were still portrayed as "travelling" and packets being "frozen" while the time line is paused when they were likely already at their destination (Fig. 3). While the idea of viewing individual packet movements is appealing, the design was ultimately impractical and was readapted to a new model.

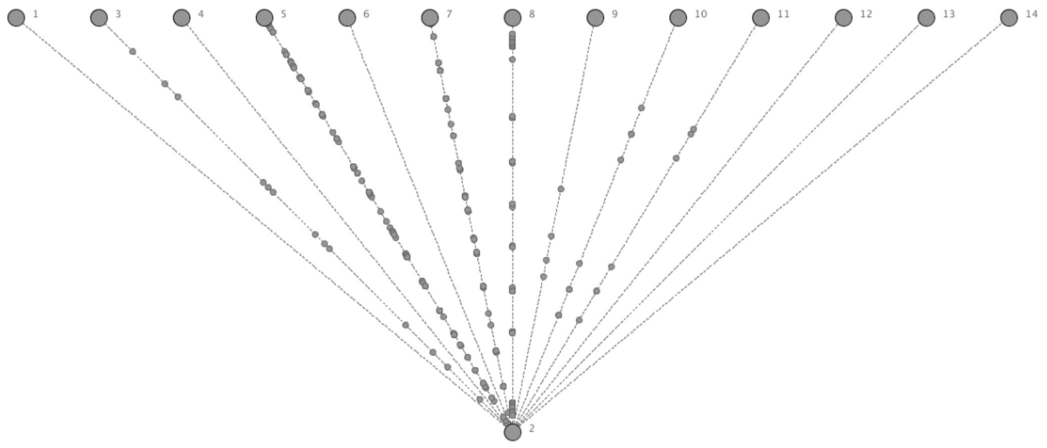


Fig. 3 Original visualization model

The new model uses flow visualization rather than individual packet visualization. A screenshot of the complete interface is found in Section 2.3. Instead of animating individual packet movements, the visualization portrays general activity along a connection, which can then be selected to show more details about a given flow of packets, including information on individual packets. This model allows the visual to base itself on the data instead of the time line directly, and the time line instead influences the state of the data. By doing this, independent animations of nodes and connections can be manipulated far more easily as the model adapts itself to the data rather than the timestamp on the time line. This allows for a simpler layout that does not sacrifice clarity or detail. In the following screenshot (Fig. 4), active data transfer is denoted by a solid connection, and the currently selected connection is denoted by the highlighted connection.

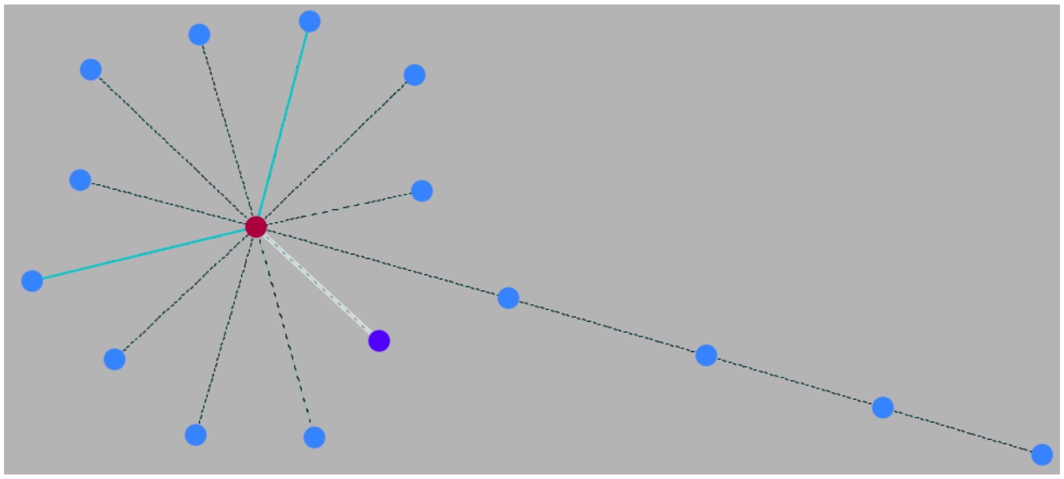


Fig. 4 New flow-based model

This new functionality makes selecting individual packets for details much easier. As stated previously, the selected connection is then illustrated in the interface as a smaller time line, as well as highlighted in the overall time line, to allow viewing of individual packets and their sizes (Fig. 5). The red line (seen in the application screenshot) denotes the current point in time, and packets sent are either drawn above or below the horizontal time line according to whether they originate from source or destination.

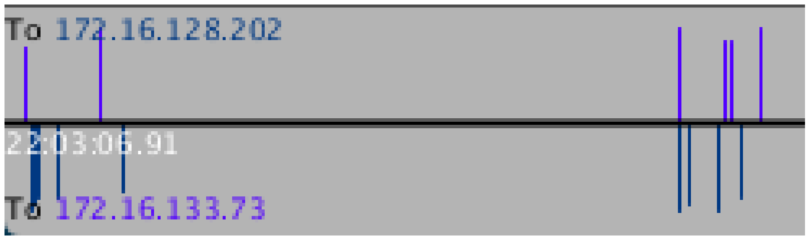
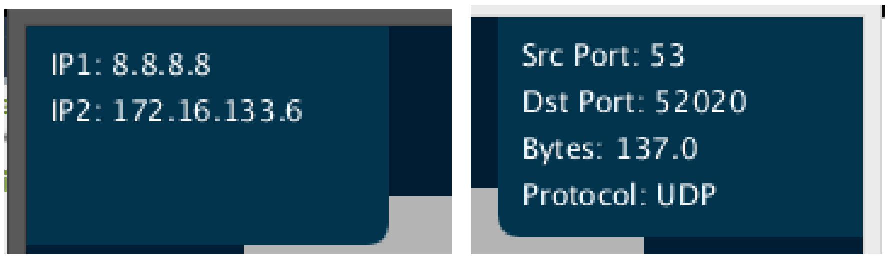


Fig. 5 **Snip of connection time line**

This new model also allows for more flexibility in changing the layout of the visible graph, allowing individual nodes and connections to be visible only when they are actively sending or receiving data to reduce overall clutter. This is particularly important on larger, complex networks with hundreds or thousands of nodes. The GUI allows for easy manipulation of individual node positions so the user can customize the general layout. The program also automatically balances the relative node positions, using a force-directed graph algorithm.[4] This makes for an intuitive and aesthetically pleasing graph drawing that is easy to learn and manipulate.

Aside from the visual aspects of the GUI, selected nodes, connections, and packets also provide direct information, including addresses, ports, bytes, and protocols (Fig. 6). Both TCP and user datagram protocol packet protocols are supported by the visuals through different colors and can also be viewed in the attributes of selected nodes. These data panels can be extended and configured to display additional information as well.



IP1: 8.8.8.8
IP2: 172.16.133.6

Src Port: 53
Dst Port: 52020
Bytes: 137.0
Protocol: UDP

Fig. 6 **Connection and packet information**

2.3 Running a Network Visualization

Currently, the application has not been exported and runs through the Processing integrated development environment (IDE). After retrieving the JSON file from Dshell, it must be named `data.json` and placed into the data directory in the application folder. Then the application file `NetVis.pde` must be loaded and run through the Processing IDE (Fig. 7).

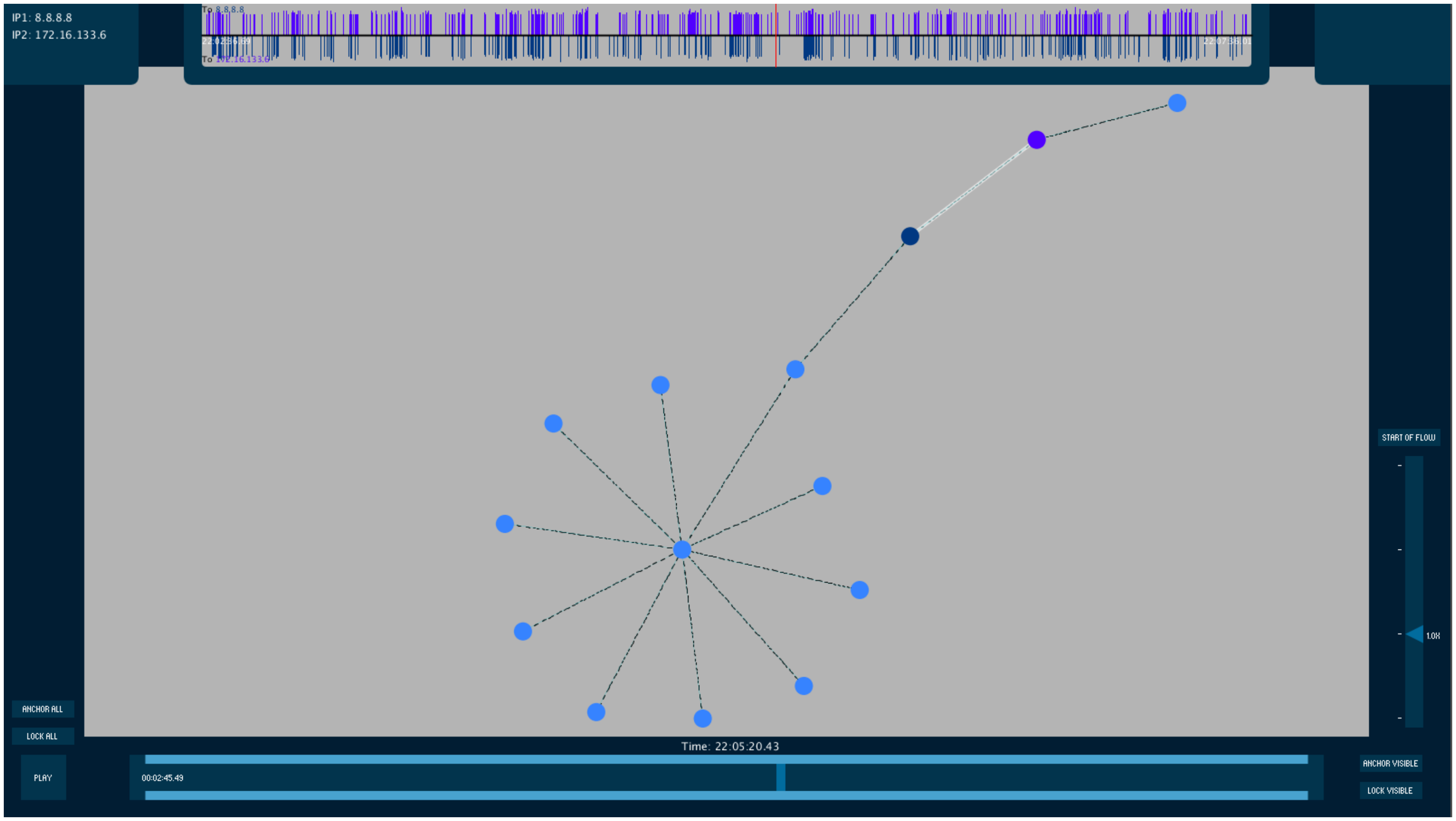
From here, there are multiple controls to interact with as well as the visualization itself. The time line can be manipulated freely, similar to a video player.[5] There are 2 timers, one for direct timestamps and one for the duration of the playback only. On the right side, there is a slider that manipulates playback speed from 0.1× up to 100× speed. There are also multiple buttons that manipulate the layout, with 2 primary functions:

- Lock: Keep node visible at all times, even when inactive (i.e., not transmitting or receiving data)
- Anchor: Keep node anchored at its current position, unaffected by automatic position adjustment

There is one final button labelled *Start of Flow*, which simply moves the time-line cursor to the beginning of the currently selected connection. Nodes and connections can be freely selected individually, and packets can be selected while a connection is active.

Along with the button functionalities, there are also a variety of keyboard shortcuts:

- *Space*: Play/Pause
- *L*: Lock current node/ all visible if no node is selected (Shift+L for all)
- *A*: Anchor current node/ all visible if no node is selected (Shift+A for all)
- *F*: Move to start of selected flow


IP1: 8.8.8.8
IP2: 172.16.133.6

Time: 22:05:20.43

Fig. 7 Application screenshot

3. Conclusion and Future Work

Data networks are a pervasive part of daily life in the modern era, but few are truly aware of what really happens in these systems. Even competent system administrators and network managers struggle to manage the ever-growing complexity. As network size and usage continue to grow, the chances of error and threats become more and more pertinent. Today, even the common household should be able to monitor and make full use of their home network to avoid exploitation and other possible configuration issues. NVP seeks not only to provide a helpful tool for experienced analysts, but also to promote awareness of the importance of how networks are structured and operate.

Though the base of NVP is complete, there is still much that needs to be expanded upon. As stated in the original goals for the project, aesthetics should not sacrifice technical detail in an interactive interface. There remain many details in the network packets (in the original pcap data) that are not currently presented via the GUI. A primary attribute of packets to highlight is the TCP flags to help capture more detail into what is occurring in each of the flows portrayed in the visualization. The GUI also currently functions primarily as a large-scale interactive and animated playback visualization with no simulation capabilities. Other issues in the layout lie in the simplicity of the nodal graph drawing: while the clarity is important, a difficult issue to address is how to distinguish or group similar nodes without having to look into their details.

These issues can be readily accounted for with future improvements to the user customization and interaction, which could use color-coding and node filtering as well as other ways to present information, such as node sizing and clustering. Other ideas discussed included data exportation, which could allow for shared visualizations that include highlighting, annotations, and user-specified or preconfigured layouts that better communicate a given network analysis.

To simplify application deployment, NVP needs to be configured as a stand-alone application that does not require the Processing IDE to run. It may also be incorporated as a web application using Processing.js, a Javascript framework for Processing. All of these functions can be built off of the current base application, which is released as open source to enable further innovation and community-driven development possibilities.

4. References

1. WireShark [accessed 2015 June 3]. http://www.wireshark.org.
2. Dshell [accessed 2015 June 8]. https://github.com/USArmyResearchLab/Dshell.
3. Processing. Documentation [accessed 2015 June 3]. https://processing.org /reference/.
4. Marchi L. Force directed placement [accessed 2015 July 9]. http://openprocessing.org/visuals/?visualID=177.
5. ControlP5. Documentation [accessed 2015 June 20]. http://www.sojamo.de /libraries/controlP5/reference/index.html.

List of Symbols, Abbreviations, and Acronyms

GUI	graphical user interface
IDE	integrated development environment
IP	internet protocol
JSON	JavaScript object notation
NVP	network visualization tool
TCP	transmission control protocol

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIRECTOR US ARMY RESEARCH LAB RDRL CIO LL IMAL HRA MAIL & RECORDS MGMT
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
3 (PDF)	DIR USARL RDRL CIN D L MARVEL J CLARKE RDRL SLB S C MORRISON

INTENTIONALLY LEFT BLANK.