A Guide to the Science DMZ Telemetry Study

Tennessee Technological University

2024

Table of Contents

Introduction

Motivation

Science DMZs are widely deployed in U.S. academic campuses, but there is a lack of studies around how DMZs influence actual scientific data and workflows. There are not many comparative research studies examining the performance of the DMZ to an average network. Our goal is to perform an extensive measurement study and quantify improvements brought forward by the new Science DMZ.

Background

A Science DMZ is a portion of a network, built at or near a campus/laboratory network local network, that is optimized for high-performance scientific applications rather than general purpose computing [1]. It is intended to be used by research institutions where extremely large volumes of data (peta- and terabyte levels) need to be transferred with minimal loss or interference. These networks have vastly different security measures from a general-purpose network, using ACLs and limiting users/applications able to access the network, whereas a commodity network can host “endless” users and applications and uses strict firewalls. The difference in security allows for the DMZ to transfer files faster and with less packet loss and jitter.

In this study, we examine the campus and DMZ network at Tennessee Tech. To perform measurements to study the performance of these networks, access to one node on the campus network is provided, as well as four nodes on the Science DMZ. “Leo” is the node on the campus network. “DTN1”, “DTN2”, “Perfsonar1”, and “Perfsonar2” are the nodes on the DMZ. Performance is monitored amongst nodes on the local-area network as well as between local nodes and external nodes on the wide-area network. Sets of RIPE Atlas probes, Google Cloud Virtual Machines, and a collection of perfSONAR nodes hosted at external institutions across the United States are used to examine performance across the wide-area network.

Tools and Methods

RIPE Atlas

RIPE Atlas is a global network with numerous servers, measurement devices, and virtual machines for network measurement. The RIPE Atlas network is made up of “probes” from which a variety of measurements can be conducted to provide a real-time understanding of the condition of the Internet. We utilize the probes to perform ping and traceroutes to and from the servers on campus.

Our test setup:

- We selected 25 probes, sectioned into groups of 5, from across the United States to perform tests.
- Ping tests are set to run every hour and traceroute tests are set to run every 6 hours.
- Tests are set to run from the 5 probe groups to Leo (149.149.2.70), Perfsonar1 (149.149.248.20), and DTN1 (149.149.248.52).
- All settings are left at default aside from the recurrence intervals and the description, which is always set to ‘Chs9LKVsSReijF5iMMGDJISgE6tLvlThcVCPx0sebx4NRbfXOKrFIJUbys@^@Gbu9ZuNq^X8A0FH2ZZmW6Oxb3ITMyB5sz^N8HLv6aspNwqVBPzt5P2PYZp@sPsEGFp4qGTAVWfEQPf6ePPs4s8qT%bx6MQ**mvic9EmyK7eWHJm9Ki%4n@ODTYiUL4O38UMRbyPWjzpk6RjNZ0m%X%nnbOZZk4n’ for ping measurements, and ‘Traceroute’ is added to the end of traceroute measurements.
- A custom command-line tool exists to create these measurements, but it only works on Leo [2]. It is easier to create the measurements from the RIPE Atlas web interface.

Inside Juptyer Notebook, using Python, RIPE Atlas Cousteau is used to analyze the data collected [2]. You will need to take note of the measurement ID in order to access the data. Pandas must be used to format the data so that it can be parsed more easily and analyzed. In many of the graphing scripts, only one dataset can be graphed at a

time and the IP address of the campus node being targeted must be defined in the programming in order for the graph to be accurately plotted.
The IDs of the measurements used most often in the study so far are:

- Ping Measurements
  - 39019527 – Leo
  - 55518116 – Perfsonar1
  - 58881309 – DTN1
- Traceroute Measurements
  - 40016357 – Leo
  - 55517771 – Perfsonar1
  - 58881290 -DTN1

Google Cloud

We used Google Cloud to host virtual machines that measurements could be run from and run to.
The Google Cloud VM should now have a static IP address attached to it, but any IP change should be noted as it is used in measurement scripts and inputs to measurement scripts. Always double-check that measurements running to and from Google Cloud are producing expected results.
The current IP address for Google Cloud is 35.207.33.118.
Ping and traceroute measurements are run from Google Cloud to the three main campus nodes (Leo, Perfsonar1, and DTN1). IPerf3 tests are run using Google Cloud as the server and Leo and Perfsonar1 as clients.

perfSONAR

perfSONAR (performance Service-Oriented Network monitoring ARchitecture) is a network measurement toolkit as well as a classification of network nodes that can be used for such measurements. The toolkit provides many resources within one package to test and measure network performance. perfSONAR identifies areas of poor

performance, by both location within the network and by a window of time in which they occur, and flags problem spots. [3] [4]

At the start of our measurement study, we attempted to use the toolkit to perform latency (ping) and traceroute measurements. However, due to data storage changes on an updated version of the toolkit, this became an inviable option, and custom tools were created. Further information about the perfSONAR toolkit and issues encountered can be found in the Lab Notebook.

We continue to utilize the network of perfSONAR nodes for the measurement study. A heavily used node on the Science DMZ is a perfSONAR node and twelve perfSONAR nodes from external academic institutions across the United States were chosen as targets of our WAN experiments.

The following externally located perfSONAR nodes were used for measurements [4]:

- 66.133.111.78
- 66.133.96.78
- 68.142.131.6
- 136.142.202.118
- 141.216.99.254 (This node has since been removed from the perfSONAR Lookup Directory Service [4])
- 192.111.110.77
- 66.99.43.226
- 207.189.117.10
- 209.170.192.2
- 216.58.152.198
- 72.253.66.3
- 67.58.50.74

Cron Jobs

Using cron jobs can save a lot of time when running measurements; it will automatically run tasks at your specified intervals. Access to crontab should be set up for users on the DMZ nodes and can be set up easily on Leo with the sudo command. Using “crontab -e”

will allow you to edit your scheduled jobs and “crontab -l” will allow you to view your scheduled jobs.

Transferring Files to Your Local Machine

Most files encountered in this study can easily be transferred to your local machine using the “copy-paste” technique. However, if you want to transfer files more securely and accurately (pcap files cannot be transferred by copy-paste) then you can use a command such as “scp”. “scp” means “secure copy” and it uses SSH to transfer data. Here is an example for using this command: “scp your_username@leo.ngin.tntech.edu:'*.pcap' ~/leo_pcap”. Utilizing a * in the file name allows for multiple files fitting the specification of “.pcap” to be transferred. If you only want one specific file, then enter the file name verbatim.

Sudo

There are some commands that require root access to use. Students are allowed root access to the Leo machine but not to the DMZ machines. Leo tends to need sudo in front of most basic commands, like ping and traceroute and crontab. Sudo is not needed for those commands on the DMZ nodes, but if you do come across a necessary command that requires root access and that may need to be run regularly, contact Dr. Mike Renfro or Dr. Shannigrahi for assistance.

Routes and Nodes

As mentioned in the Introduction section, experiments are run across local and external network nodes. There are five campus nodes that can be used for experiments:

- Leo (commodity) – 149.149.2.70
- Perfsonar1 (DMZ) – 149.149.248.20
- Perfsonar2 (DMZ)
- DTN1 (DMZ) – 149.149.248.52
- DTN2 (DMZ) – 149.149.248.53

One Campus Gateway node is used for experiments – 149.149.2.1.
As for external resources, twenty-five RIPE Atlas probes are used for measurements and twelve perfSONAR nodes are used (defined previously). One Google Cloud Virtual Machine is used - 35.207.33.118.

*Note regarding access of campus nodes: Leo can be SSH'd into from anywhere, but all DMZ nodes require you to be on campus to SSH into them.

We run experiments to observe the parameters in the table below. Ping Experiments provide insight to RTT (or latency), jitter, and packet loss. Traceroute Experiments provide RTT (or latency), and path lengths. iPerf3 Experiments detail throughput and Data Transfer Experiments provide RTT, Throughput, Window Size, Download Time, and Inter Packet Delay (or jitter). BGP Experiments offer insight to path lengths.

Parameters for Comparative Analysis	
LAN Measurements	**WAN Measurements**
Throughput	Everything Observed on LAN Side
RTT Between Nodes	BGP Routes To/From External Vantage Points
RTT Between Node and Gateway	Path Length Between Campus and External Vantage Points
Jitter	-
Packet Loss	-

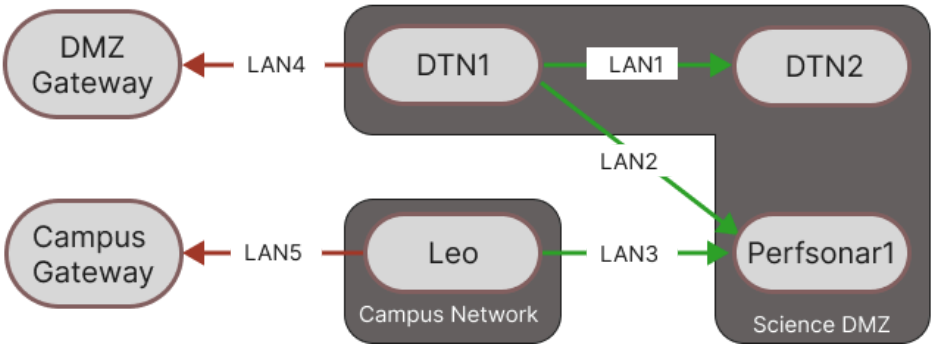

Figure 1: LAN Routes

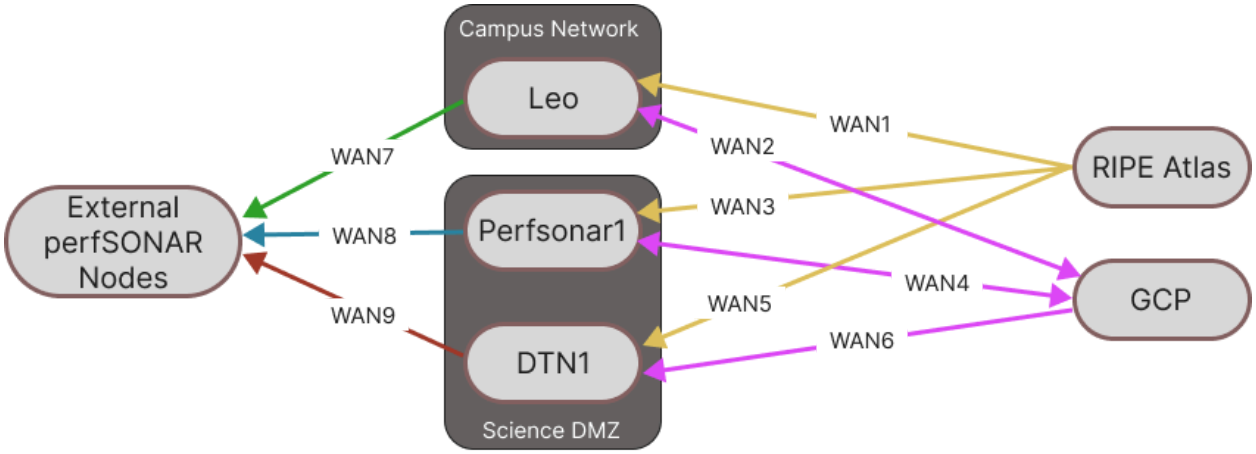

Figure 2: WAN Routes

Ping Experiments

Running the Experiment

Ping experiments are run on all LAN and WAN routes.
WAN1, WAN3, and WAN5 ping measurements are run from RIPE Atlas to campus.
LAN1 – LAN5, WAN2, WAN4, WAN6, and WAN7 – WAN9 are run using a custom command-line tool found on the project GitHub [5]. Currently the command line tool uses a while loop rather than a cronjob to run incrementally, so each measurement must be restarted at the end of the preset end time (this can be changed to whatever length of time is needed but shorter time periods are easier to monitor for mistakes). The tool can be converted to use cronjob instead of a while loop now that crontab is set up on the DMZ nodes.
Command-line ping experiments run every 30 minutes and RIPE Atlas ping experiments run every hour. More information on RIPE Atlas measurements can be found in the Tools and Methods section.
For consistency, the command-line ping tests should be started on the hour or on the half-hour. Using bash for Perfsonar1, DTN1, and GCP, and sudo bash for Leo, the ping test scripts will be started. It will ask for a list for IP input and a filename (start time functionality is disabled so the input for that field does not matter). The IP input should be a list of IP addresses separated only by a space, no commas. The filename should be set to “ping_*date*_” for easier record keeping and the IP address will be automatically appended to the end. The script will run ping to each IP address and store the results in files specific to the destination IP.
Exact IP List inputs for each source node are as follows:

- Leo
    - 66.133.111.78 66.133.96.78 68.142.131.6 136.142.202.118 192.111.110.77 66.99.43.226 206.71.76.62 207.189.117.10 209.170.192.2 216.58.152.198 67.58.50.74 149.149.248.20 149.149.2.1
- Perfsonar1

    - 66.133.111.78 66.133.96.78 68.142.131.6 136.142.202.118 192.111.110.77 66.99.43.226 206.71.76.62 207.189.117.10 209.170.192.2 216.58.152.198 67.58.50.74 149.149.2.70 149.149.2.1
- DTN1
    - 66.133.111.78 66.133.96.78 68.142.131.6 136.142.202.118 192.111.110.77 66.99.43.226 206.71.76.62 207.189.117.10 209.170.192.2 216.58.152.198 67.58.50.74 149.149.248.20 149.149.248.53 149.149.2.1
- GCP
    - 149.149.2.70 149.149.248.20 149.149.248.52

Processing the Results

Results from RIPE Atlas can be easily accessed with measurement IDs when using the RIPE Atlas Cousteau library in Python. Examples for accessing this data can be found in the two project repositories on Github [5] [2].

Results from command-line ping experiments are stored in log files. They include UNIX timestamps for each measurement run in the time window and raw transcriptions of the ping that was run. In the project repository on Github [5] there are python scripts to convert the raw data to JSON and to graph the parameters we desire. Before graphing, the files must be transferred to your local machine. Files can be transferred using the scp command or by using CTRL+up to copy the file output and paste it into a new text file on your local machine. If you want to append the outputs of more than one file, then the copy and paste method would work best. The raw ping output is formatted differently on Leo than on the DMZ machines, so there are two different JSON conversion scripts that can be used. Always double check that the file is converted to JSON correctly. Pay attention to which machine you retrieved the files from when you are moving them to the local machine, ping does not document source IP addresses and it is easy to mix up results.

Graphs

Many graphs represent the ping experiments. Cumulative Distribution Function graphs are used to categorize latencies observed in the experiments in a percentile format. Jitter graphs are also created to show the average daily jitter and standard deviation on a line plot. The jitter is calculated by noting the difference between consecutive ping results, then they are averaged across a day's span.

Traceroute Experiments

Running the Experiment

Traceroute experiments are run on all LAN and WAN routes, excluding LAN4 and LAN5. This experiment does not yield impactful comparative information for the documentation of the study but does offer insight to the functionality of the campus and DMZ network.

WAN1, WAN3, and WAN5 traceroute measurements are run from RIPE Atlas to campus.

LAN1 – LAN5, WAN2, WAN4, WAN6, and WAN7 – WAN9 are run using a custom command-line tool found on the project GitHub [5]. Currently the command line tool uses a while loop rather than a cronjob to run incrementally, so each measurement must be restarted at the end of the preset end time (this can be changed to whatever length of time is needed but shorter time periods are easier to monitor for mistakes). The tool can be converted to use cronjob instead of a while loop now that crontab is set up on the DMZ nodes.

Command-line traceroute experiments run every 30 minutes and RIPE Atlas traceroute experiments run every six hours. More information on RIPE Atlas measurements can be found in the Tools and Methods section.

For consistency, the command-line traceroute tests should be started on the hour or on the half-hour. Using bash for Perfsonar1, DTN1, and GCP, and sudo bash for Leo, the traceroute test scripts will be started. It will ask for a list for IP input and a filename (start time functionality is disabled so the input for that field does not matter). The IP input should be a list of IP addresses separated only by a space, no commas. The filename should be set to “trace_*date*_” for easier record keeping and the IP address will be automatically appended to the end. The script will run traceroute to each IP address and store the results in files specific to the order that destination Ips are entered in the list.

Exact IP List inputs for each source node are as follows:

- Leo

    - 66.133.111.78 66.133.96.78 68.142.131.6 136.142.202.118 192.111.110.77 66.99.43.226 206.71.76.62 207.189.117.10 209.170.192.2 216.58.152.198 67.58.50.74 149.149.248.20 149.149.2.1
- Perfsonar1
    - 66.133.111.78 66.133.96.78 68.142.131.6 136.142.202.118 192.111.110.77 66.99.43.226 206.71.76.62 207.189.117.10 209.170.192.2 216.58.152.198 67.58.50.74 149.149.2.70 149.149.2.1
- DTN1
    - 66.133.111.78 66.133.96.78 68.142.131.6 136.142.202.118 192.111.110.77 66.99.43.226 206.71.76.62 207.189.117.10 209.170.192.2 216.58.152.198 67.58.50.74 149.149.248.20 149.149.248.53 149.149.2.1
- GCP
    - 149.149.2.70 149.149.248.20 149.149.248.52

Processing the Results

Results from RIPE Atlas can be easily accessed with measurement IDs when using the RIPE Atlas Cousteau library in Python. Examples for accessing this data can be found in the two project repositories on Github [5] [2].

Results from command-line traceroute experiments are stored in log files. They include UNIX timestamps for each measurement run in the time window and raw transcriptions of the traceroute that was run. In the project repository on Github [5] there are python scripts to convert the raw data to JSON and to graph the parameters we desire. Before graphing, the files must be transferred to your local machine. Files can be transferred using the scp command or by using CTRL+up to copy the file output and paste it into a new text file on your local machine. If you want to append the outputs of more than one file, then the copy and paste method would work best. The raw traceroute output is typically formatted identically across all of the machines, so there is only one JSON conversion script used to convert the results, but always double check that the conversion is correct as the format could be subject to change. Pay attention to which machine you retrieved the files from when you are moving them to the local machine,

although traceroute does document source IP addresses, it is easy to mix up results in the graphing process.

Graphs

Multiple graphs represent this experiment. There are bar graphs that track the frequency of route lengths and there are bar graphs that track the average latency exhibited by different route lengths.

IPerf3 Experiments

Running the Experiment

iPerf3 experiments are run using Leo, Perfsonar1, and Google Cloud. Leo and Perfsonar1 are designated as clients and Google Cloud as the server. Using crontab, the experiments are set to run every 12 hours, storing the UNIX timestamp and iperf3 results to a log file.

Processing the Results

For easiest processing, the iperf3 log files should be extracted from Leo and from Perfsonar1. The log files can be converted to JSON using python scripts found on the project GitHub. The format of iperf3 output is generally the same across all machines.

Graphs

The graphs that currently represent this experiment show the levels of throughput (both download and upload) over time on a line plot.

Data Transfer Experiments

Running the Experiment

Data Transfer Experiments are run exclusively on Leo and DTN1, executed by use of crontab every 4 hours. In these measurements we download files from five sources and capture the packets as the files are downloaded using tcpdump. Tcpdump stores the results in a pcap (packet capture) file and using Wireshark we can examine RTT, throughput, download time, interpacket delay, and window size from the data. Due to the nature of this experiment, storage issues can arise, so storage should be monitored, and files offloaded to an external storage facility when nearing capacity.
On Leo, this experiment is run with a single script. Due to root permissions on the DMZ machines, the experiment must be run in two separate scripts. The student is able to run the script responsible for downloading the files (the cronjob for this should be set to every 4 hours at one minute after the hour). Dr. Renfro runs the cronjob for the script responsible for executing tcpdump and storing the results in a file. The student has access to change the script but any changes in the cron must be performed by Dr. Renfro.

Transferring Files

As detailed in the Tools and Methods section, you should use scp to extract packet capture files from Leo and DTN1. These files are numerous and large in size so unless you have an abundance of free space on your local machine, you should utilize an external drive to store the files in.

Using Wireshark

After transferring the packet capture files to your local machine, you can examine the results in Wireshark. Most of the values for the experiment can be found under the “Conversations” tab and in the Graphs (window size, RTT, throughput). In the “Conversations”, throughput is labelled as “A->B” and “B->A”, the download throughput that should be documented will be the larger of the two. You should also filter the data

inside of “Conversations”, by right-clicking on a conversation and selecting “filter on stream id”. Also, in Wireshark, you can alter the columns of information displayed on the main screen using the “Preferences” tab. You can select and deselect default columns and create custom columns. For the external calculations of RTT and interarrival delay, you should select only columns of delta time and a custom column for RTT.

Processing the Results

The results for each parameter of this experiment are stored in a spreadsheet, and the daily averages are stored in another separate spreadsheet.
To calculate RTT and Interpacket Delay, you should download the filtered data with only Delta Time and RTT columns selected as a CSV. Then you can run the CSV through a python script (DataTF.ipynb) found on the project repository and it will output the values.

Graphs

Only 5 days of measurement data was used for the initial study, so the exact averages are documented in static arrays in the graph script for each parameter observed across both Leo and Perfsonar1. This can be altered to read from the Excel file if the volume of data increases in the next phase of the study.

BGP Experiments

Summary

ITS runs this set of experiments. This set of measurements examines path lengths along the BGP routes of the general-purpose campus network and the Science DMZ. Any further questions about this experiment should be directed to Dr. Shannigrahi.

Graphs

The graph that currently represents this experiment is a bar graph showing the frequency of certain route lengths for the general purpose network and the Science DMZ side-by-side.

Video Experiments

Overview

This experiment is intended to run with Leo, DTN2, and Google Cloud. Leo and DTN2 will use Dash Client to capture streaming data from Google Cloud which will be streaming the videos.

Frequently Asked Questions

Ping/traceroute/other mundane command is not working on Leo.

Try the command again with ‘sudo’ [2] in front of it.

If a basic traceroute command is not working on Leo, even with sudo, add “-I icmp” to the command as TCP/UDP traffic is tricky on Leo.

References

[1] "Science DMZ," 27 April 2022. [Online]. Available: https://fasterdata.es.net/science-dmz/.

[2] "Science DMZ - RIPE Atlas Repository," [Online]. Available: https://github.com/cdsweo42/ScienceDMZ/tree/main.

[3] "perfSONAR," perfSONAR, 2024. [Online]. Available: https://www.perfsonar.net/index.html. [Accessed 04 04 2024].

[4] "perfSONAR Lookup Service," perfSONAR, [Online]. Available: https://stats.perfsonar.net/d/spFwAQi4z/perfsonar-public?orgId=2. [Accessed 04 04 2024].

[5] "Science DMZ Project Repository," [Online]. Available: https://github.com/tntech-ngin/dmz-measurements.