National College of Ireland

# Real Time Network Mapping On Service Level

MSc Internship
Cyber Security


Dmitrijs Starcenko
Student ID: 17148618

School of Computing
National College of Ireland

Supervisor: Ross Spelman

**National College of Ireland**
**Project Submission Sheet**
**School of Computing**

| | |
|---|---|
| **Student Name:** | Dmitrijs Starcenko |
| **Student ID:** | 17148618 |
| **Programme:** | Cyber Security |
| **Year:** | 2020 |
| **Module:** | MSc Internship |
| **Supervisor:** | Ross Spelman |
| **Submission Due Date:** | 23/04/2020 |
| **Project Title:** | Real Time Network Mapping On Service Level |
| **Word Count:** | 4148 |
| **Page Count:** | 19 |

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

**ALL** internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

| | |
|---|---|
| **Signature:** | Dmitrijs Starcenko  |
| **Date:** | 23rd April 2020 |

## PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies). | ☐ |
| **Attach a Moodle submission receipt of the online project submission**, to each project (including multiple copies). | ☐ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | ☐ |

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Real Time Network Mapping On Service Level

Dmitrijs Starcenko
17148618

**Abstract**

Network mapping is one of the key pieces of methods which helps to generate information that allows us to see the setup of current network topology residing on the premises. Having such information can help to figure out issues on the network by identifying the potential location and its cause on the network by referring to the network maps. This paper presents a solution for network map which is generated on the service level i.e. it allows to see through which service and port each end device is connected to each other in order to exchange information and perform certain tasks.

## 1 Introduction

Network state visibility is an important aspect and information source for an Network Administrator or other parties that look after network state. As organizations and any other facilities introduce more and more devices on their network that can run numerous amounts of software and services in the background it can become dangerous by accidentally allowing devices to interconnect with each other over those services.

Typical services that run on end devices which are part of an organisation are FTP, SMB, HTTP/S, SQL, SSH, RDP and many others use logical port numbers which allow communication between numerous devices to exchange and gather information as needed. End devices that reside on the network and users with privileges which grant ability to install different software and services for their needs potentially brings risk of data breach as tracking the activity of these services becomes close to impossible without identifying their activity on the network, but with the help of network mapping generation in real time it becomes possible.

Clark (2006) discusses how important it is to know your social environment in order to make appropriate decisions and avoid potential big unforeseen problems with such knowledge. Same idea applies to network maps. Once having the knowledge about your network topology it will be easier to diagnose communication issues between end devices and other potential issues.

Another reason why such network maps should be generated in real time is because any kind of unauthorised access or data breach can be completed in an instant and real time data availability will allow to react to such issues in timely manner and potentially minimise the potential damage that could be done if acted upon it in much later time. Another reason is the instant availability of the view of the network. Techniques which rely on active or passive data gathering can take a substantial amount of time and even miss some end devices which will return an incomplete network topology view.

As network topology can be both big and small it is important to choose an appropriate technique to represent such information in the best readable way for an observer to consume that data for further analysis. A brief look at the History Of Data Visualization done by Friendly (2008) and the Value of Information Visualization by Fekete et al. (2008) shows that the ability to properly visualize data can show the big picture behind large quantities of complex textual data as well as break the language barrier in order to get the message across.

By examining latest state of the art solutions reported in Gartner Magic Quadrant 2019[Figure 1] that monitor network status and produce real time network maps and provides other data of the network state it was noticed that all the network maps are very similar in nature except for their the visual design. These network maps display end to end device connections over the network with visual representation of routers, switches and firewalls but lack the extra information which port and service is used when the connection is established to perform certain tasks on the network between the end devices.

This paper will describe and implement an approach to be able to see how end devices which reside on the network are connecting to each other over the exposed active ports and determine the service being used. This will help to understand what is actually happening on the network in that given moment to potentially be able to spot abnormalities on the network or identify access to services which should have not been active.

# 2 Related Work

## 2.1 Simple Network Management Protocol (SNMP)

SNMP was introduced as an Internet Standard protocol which is used for collecting and organizing information about IP network devices. The initial version of SNMP is SNMPv1 which is described in Hare (2011) memo the need for it to be standardised by The Internet Activities Board (IAB).

Once SNMPv1 has been standardized it started to get integrated into vast variety of devices which can be situated on the network and give it an IP address. SNMP started to get widely used in network management for network monitoring which exposed management data in management information base (MIB) variables which describe the configuration and system status. Current version of SNMP is SNMPv3 which introduces further enhancements to performance, security and flexibility.

Along came a solution for network mapping from Li (2011). His approach utilizes the ability of SNMP to give access to an object which contains a table with currently connected child devices to a parent device as well as an ARP table with devices on same sub network. Such tables can be seen in devices that support SNMP protocol such as routers, switches, hubs, firewalls and others that support SNMP and can assign IP to a child device and detect other on the same sub network.

Li (2011) algorithm is a great addition to network mapping tools/techniques as it does not rely on the knowledge of IP address range, thus the entire network scan will be completed in record time. Although the biggest drawback is that it relies on the network devices to support SNMP, having it enabled and have enough security privileges to access required MIB variables. Also the only information we can extract from this approach is the device IP address which can be further scanned by other available methods.

## 2.2 Deep Packet Inspection (DPI)

DPI is a process which allows to inspect network traffic packets in detail the data being sent over computer network. Most common cases for DPI is to inspect the packets and block them if needed, make a log entry or re route depending on the header data. In the field of security DPI often analyzes packet data to ensure there is no malicious code, the sent data is in correct format and that there is no eavesdropping.

With help of DPI Cai et al. (2017) implemented a network traffic scanning application which examines network traffic on central Open-Flow network server. This allowed to inspect outgoing traffic and determine the protocol used in order to access online resources. Applications intention was to determine traffic bursts and construct a network map in order to visually identify host that causes the sudden burst. It successfully uses a Fruchterman and Reingold (1991) force directed graph technique implemented by a D3 graph library[1] which improves its readability when networks have lots of endpoints and require automatic positioning in 2D or 3D space.

[1] D3: `https://observablehq.com/@d3/force-directed-graph`

## 2.3 Active VS Passive Scanning for Network Mapping

Active scanning is method which sends out a request to a destination IP in order to get a particular information whether its a simple ping to see if destination host is alive and reachable or to determine the services running on the destination host. This method of network mapping even though useful has drawbacks such as:

- Time - significant time can be taken in order to scan a particular sub-net of network IP's if the range is not known or is too large
- Missed Devices - while scanning is in progress device that could have been running or accessing data have already disconnected from the network

Passive scanning compared to active scanning is a method where a traffic scanner is situated in a particular end device which has the ability to examine the traffic flow on the network and in such way being able to examine the packets data. Disadvantages of passive scanners:

- Detection - if a particular device only acts as an observer on the network then passive scanner wont detect its presence as its not generating any traffic, thus making the network map in complete
- Location - depending on where the passive scanner is situated on the network it will get varying results

Review on Active versus Passive mapping done by Webster et al. (2006) highlighted these disadvantages and recommends to use both techniques in order to maximise to coverage of the network device detection although results can still vary depending on circumstances.

## 2.4 Force Directed Graph Drawing

Everything where there are at least 2 devices that can communicate with each other is considered a network. These networks can span across thousands of interconnected endpoints and if they would be represented as dots on the screen it would look like unorganised chaos which will be extremely challenging for a human eye to consume and filter such amount of data in a meaningful way.

Where conventional ways to organise unstructured node data is extremely challenging to implement Fruchterman and Reingold (1991) introduced a method called Graph Drawing by Force Directed Placement. With this method each node is connected to each other with an edge which automatically maintains distance from all other nodes allowing for a more structured representation of the network. Additionally this method is extremely helpful as due to proposed idea of mapping device connections through destination port where each device has a range from 0 to 65536 logical ports.

With such vast amount of nodes that can be potentially discovered on the network Force Directed Graphs is the viable solution for this proposed solution idea and is available in a 3rd party library called D3[2] which allows to create such graphs in web applications.

[2]D3: `https://observablehq.com/@d3/force-directed-graph`

# 3 Methodology

Research was conducted by consulting Gartner Magic Quadrant(GMQ)[Figure 1] in are of Network Monitoring. GMQ gives a chart with latest state of the art solutions available on the current market.

Where public documentation for network mapping for mentioned solutions was available, it was accessed and examined to determine their capabilities in area of producing Network Mapping On Service Level. Solutions that did not provide any documentation were researched using search engine with keywords which would return back results around network mapping visualisations with ports and services in use.

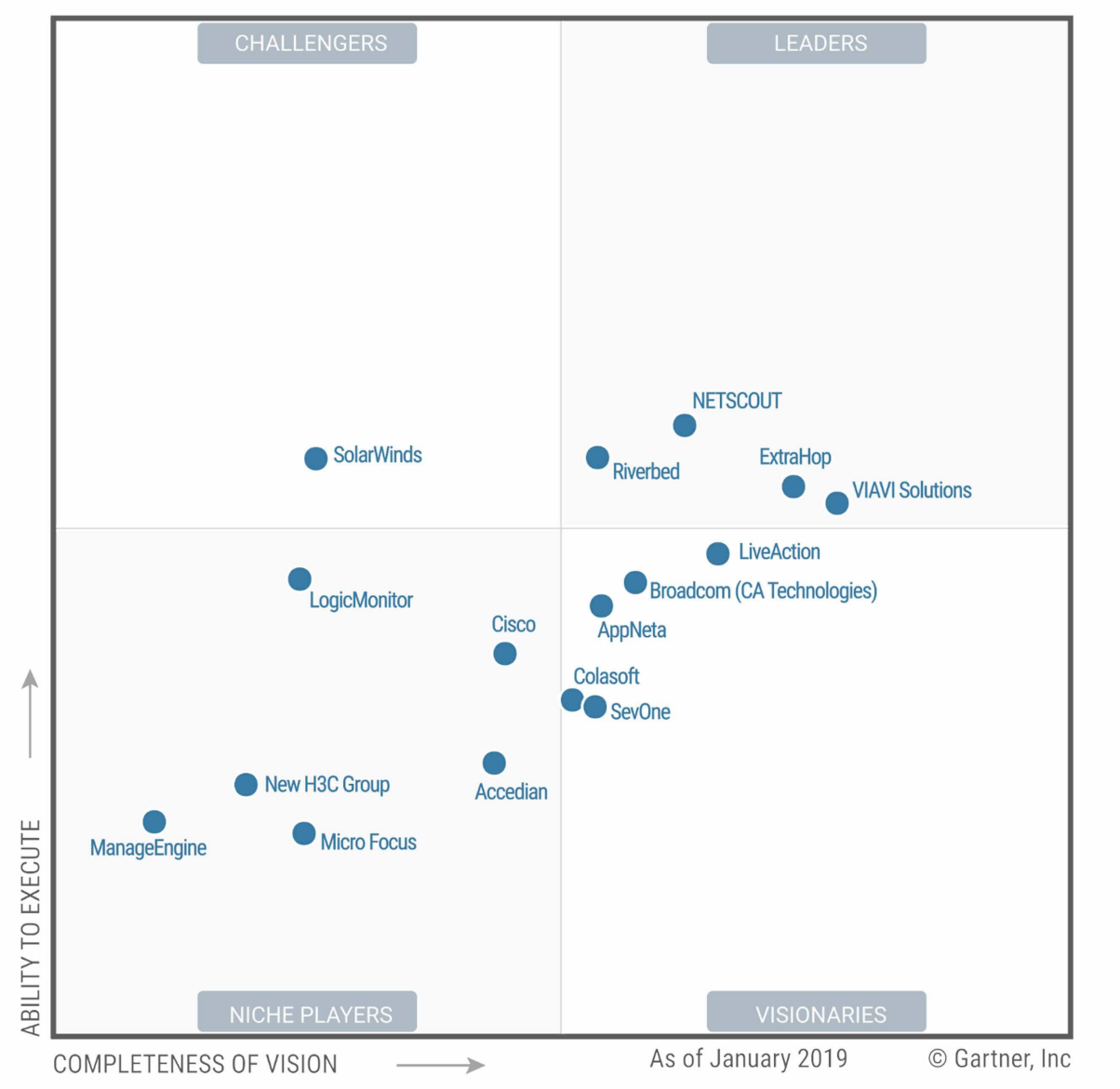


Figure 1: Gartner Magic Quadrant - Network Monitoring

Table 1: Means Used To Find Solutions Network Mapping Capabilities

| **Solution Provider:** | **Documentation Available** | **Other Sources** |
|---|---|---|
| SolarWinds | YES | YES |
| Netscout | YES | NO |
| Riverbed | NO | YES |
| Extrahop | YES | NO |
| Viavi | NO | YES |
| LiveAction | YES | YES |
| Broadcom | YES | YES |
| AppNeta | NO | YES |
| Colasoft | YES | NO |
| SevOne | NO | YES |
| LogicMonitor | YES | YES |
| Cisco | YES | YES |
| Accedian | NO | NO |
| Micro Focus | NO | YES |
| New H3C Group | NO | YES |
| ManageEngine | YES | YES |

After reading through available documentation and other resources which helped to complete the picture of network mapping solutions available on the market, not one of them demonstrated the ability to determine which port and service was used in order to establish a connection between end devices.

Although network mapping on service level is not available these solutions implement advanced techniques which allows them to identify and show following information:

- Access points (Wi - Fi, Internet)
- End devices (Printers, Laptops etc.)
- Protocols used (TCP, UDP, etc)
- Data transfer speed on particular connection
- Switches, Routers, Firewals and many other

# 4 Design Specification

To successfully implement the proposed solution a number of technologies would need to be utilized and combined to ensure the real time experience of data exchange, graph generation and ability to extract information from network packets. The entire solution will be split into three parts:

1. Server Side Application [Figure 4]
2. Client Side Application [Figure 5]
3. Graph Rendering Application

Each application will serve a specific purpose to make the whole solution work as proposed.

Server side application [Figure 4] will be the heart of the entire solution and will be performing multiple roles:

- Serve a real-time connection method for Client Side applications to connect to
- Maintain connection between all Client Side Applications running on host devices
- Retrieve, process, store data from Client Side Applications and send processed data to Graph Rendering Application
- React to events from Client Side and Graph Rendering application

Client Side Application [Figure 5] will be running on clients host machines as a standalone application. Main purpose of client side application is to sniff for incoming packets, extract source IP, destination port and report to Server with that extracted data. Client side application will connect to the Server using a method to maintain real time data exchange and react for incoming events from server as well as reply back to them with requested data if necessary.

Graph Rendering Application sole responsibility is to represent the gathered data from all the Client Side applications in a clear visual representation of the communication of end devices through services running on every host machine on the network.

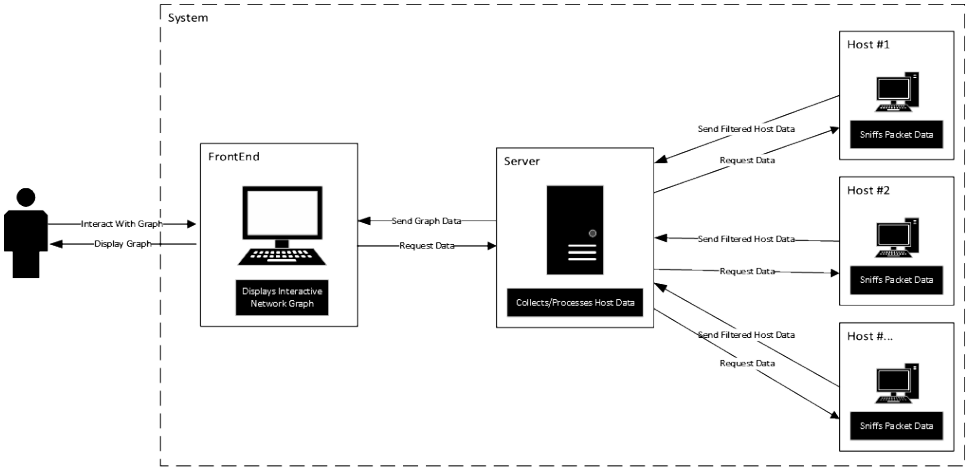


Figure 2: Top Level View Of System Architecture

# 5 Implementation

As previously discussed this implementation will be built in 3 separate applications which combined will produce the desire result of creating the network map on service level.

Both Server Side Application and Client Side Application are implemented as standalone applications which collect, process and transform data for visual presentation.

Visual Graph Application is built as a web application with a help of a powerful graph rendering framework. Further subsection will explain in greater detail how major parts of the applications were built and frameworks/modules used in order to achieve proposed solution.

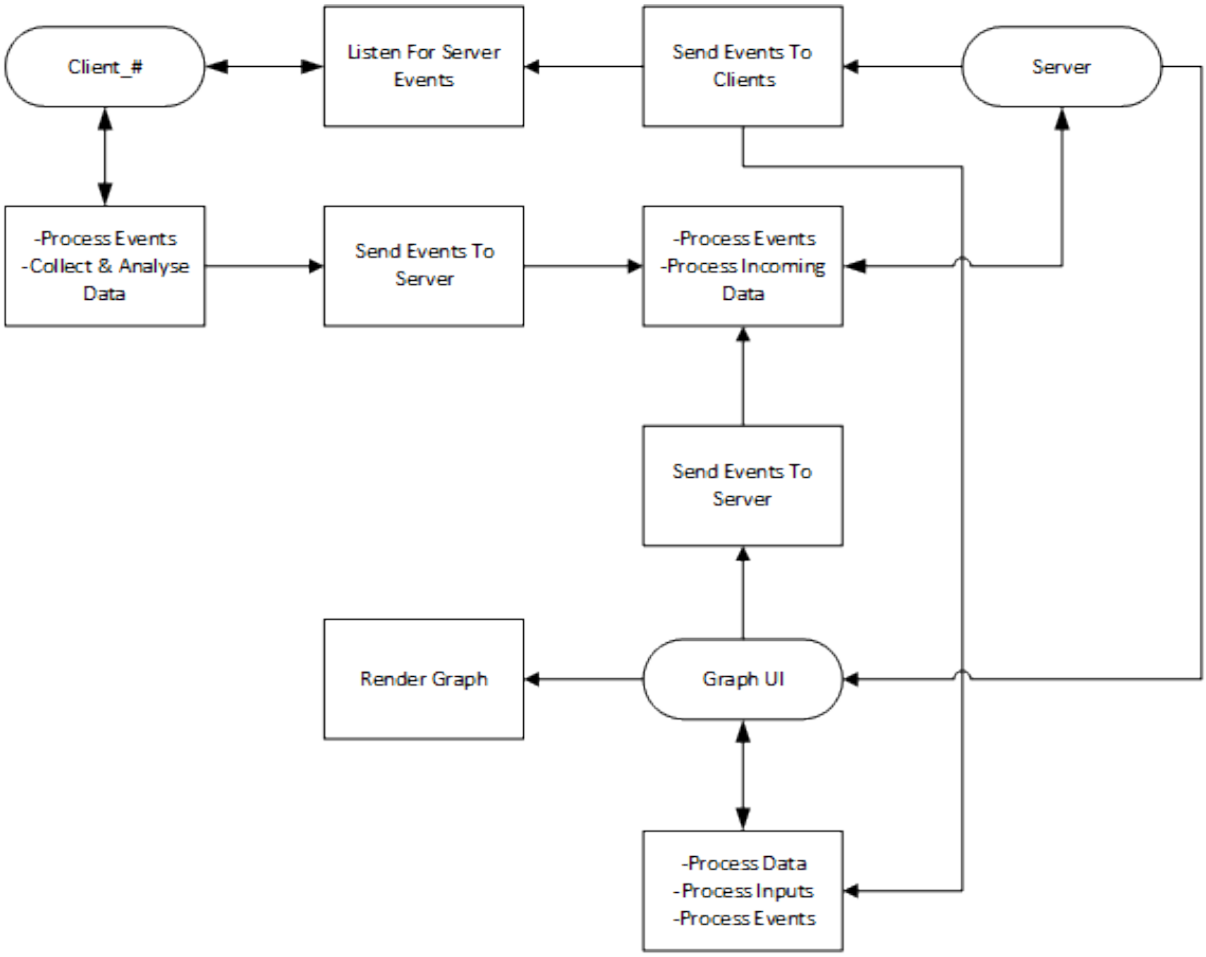


Figure 3: Top Level Overview Of The Solution

## 5.1 Server Side Application (SSA) [Figure 4]

SSA is the central piece of the system. It is implemented as a back-end server which allows remote clients to connect to it and send collected data to be processed in order to view the visual result in the Graph rendering Web Application. SSA is implemented with Python which is a high-level general-purpose programming language that serves two purposes:

- Act as a real time server for bi-directional communication between clients and SSA.
- Act as a hosting server for Graph Rendering Web Application.

SSA makes use of a powerful library called Flask-SocketIO [3] which combines two separate libraries Flask and Socket.IO.

- Flask - creates a web server that gives ability to serve web-pages and implement REST API endpoints
- Socket.IO Server - gives ability to create real-time bi-directional connection between server and client

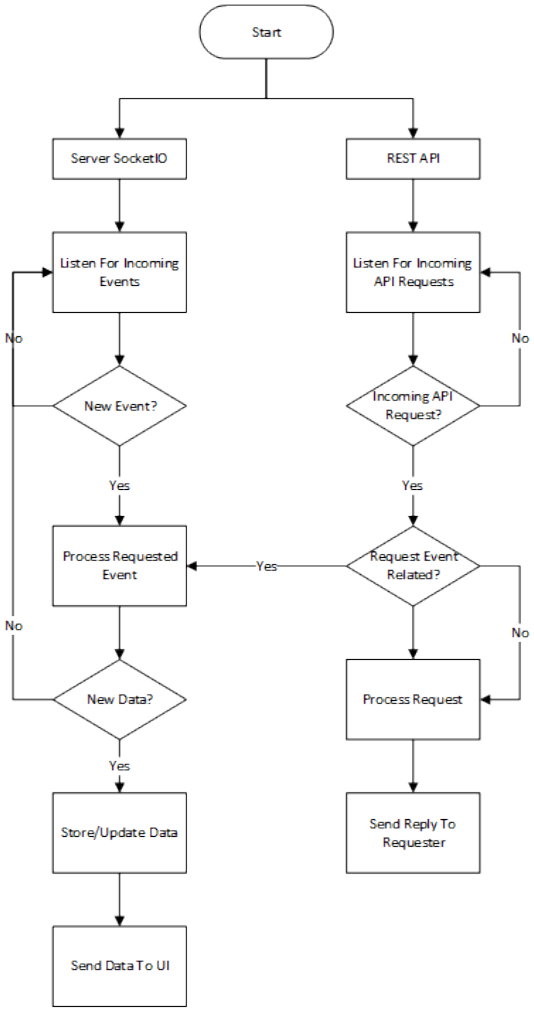


Figure 4: Server Side Application Flow Chart

[3]Flask-SocketIO: `https://flask-socketio.readthedocs.io/en/latest/`

## 5.2 Client Side Application (CSA) [Figure 5]

CSA is built using Python like SSA. Two core responsibilities of CSA are:

- Sniffing - CSA examines all incoming traffic on the host machine
- Reporting - CSA connects to SSA and reports sniffed data

CSA achieves packet sniffing using a popular library called Scapy [4]. Scapy is a packet manipulation library for computer networks. The main feature used in this library is capture and examination of packet in order to determine the original IP of the requester that is trying to connect through a certain port of the destination IP end device.

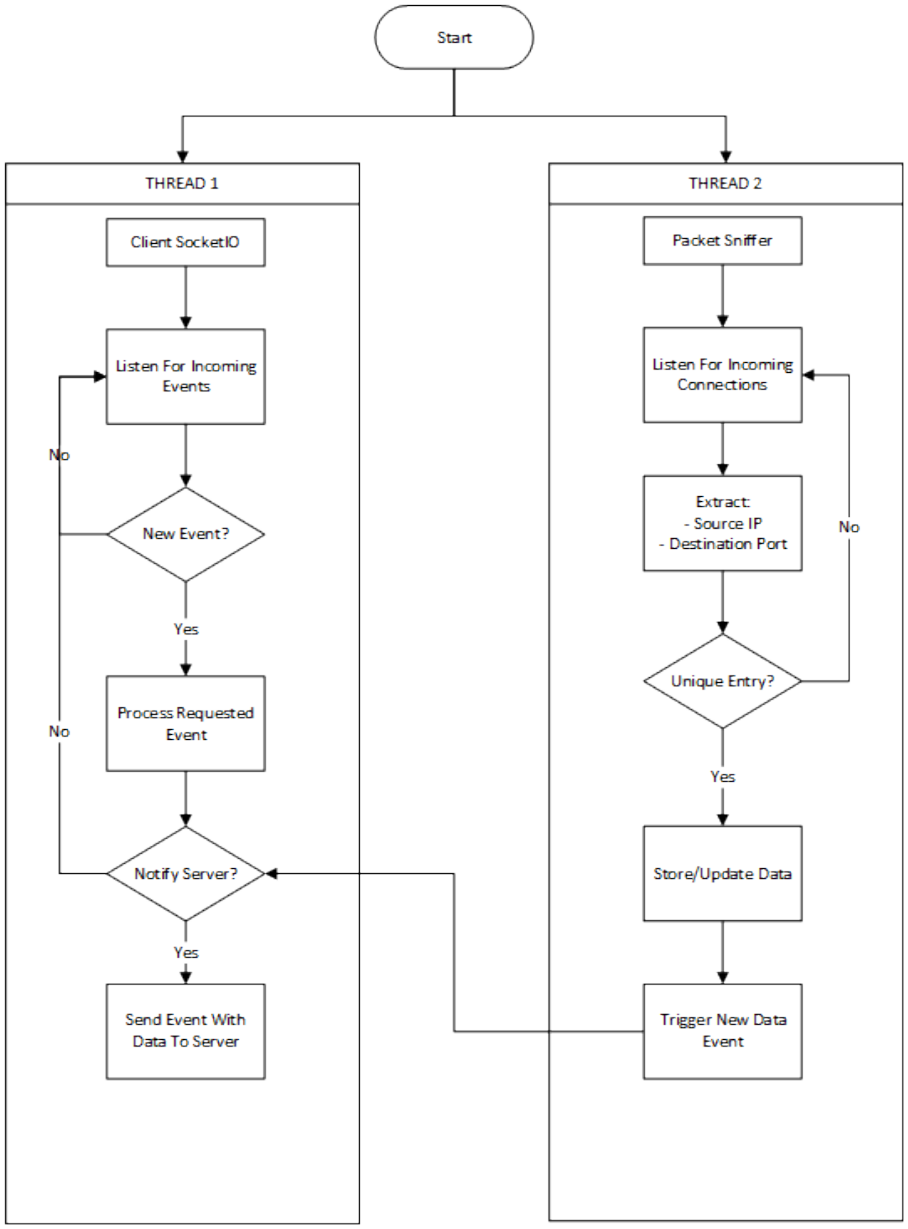


Figure 5: Client Side Application Flow Chart

[4]Scapy: `https://scapy.net/`

## 5.3 Graph Rendering Web Application

For visualisation purposes of the graph this part of the solution is built as a Web application. Web technology allows for quick visual implementations and provides vast variety of open source technologies that help to accomplish various tasks. The three core components of the Graph Rendering Application are:

- React[5]
- D3.js[6] [Data Driven Documents]
- Socket.IO Client[7]

### 5.3.1 React

React is an open-source framework developed by Facebook to create high performing HTML web pages. This framework is used in order to implement a highly responsive web application with advanced methods of React Hooks which allows web applications to react to changes as data is received and minimize unnecessary HTML DOM re-rendering for optimal performance. Implementing a web application gives the freedom to access graphs on any system which can run a web browser without the needed of complicated 3rd party software installations and configurations.

### 5.3.2 D3

D3.js is an open-source JavaScript library that allows to create dynamic, interactive data visualisations in web browsers. It's integration allows to simulate force directed graph where each interconnected node will automatically place and distance itself from other nodes for better readability.

### 5.3.3 Socket.IO Client

Socket.IO is a JavaScript library for web applications which enables real-time, bi-directional communication between web clients and servers. This library integration is essential as it will create the real time data exchange between clients and the server to keep up to date the rendered graph with changes detected on the network which are collected by CSA.

[5]React: `https://reactjs.org`
[6]D3: `https://d3js.org`
[7]Socket.IO: `https://socket.io`

# 6 Evaluation

Implemented solution was tested in a virtual medium sized enterprise network topology built in Graphical Network Simulator 3 (GNS3[8])[Figure 6]. GNS3 allows to simulate network topology by running virtual appliances such as Routers, Firewall, Switches and link Virtual Box instances running different Operating Systems. Test was conducted in 2 different approaches, manual and automated for different purposes which will be outlined in following experiment sections.

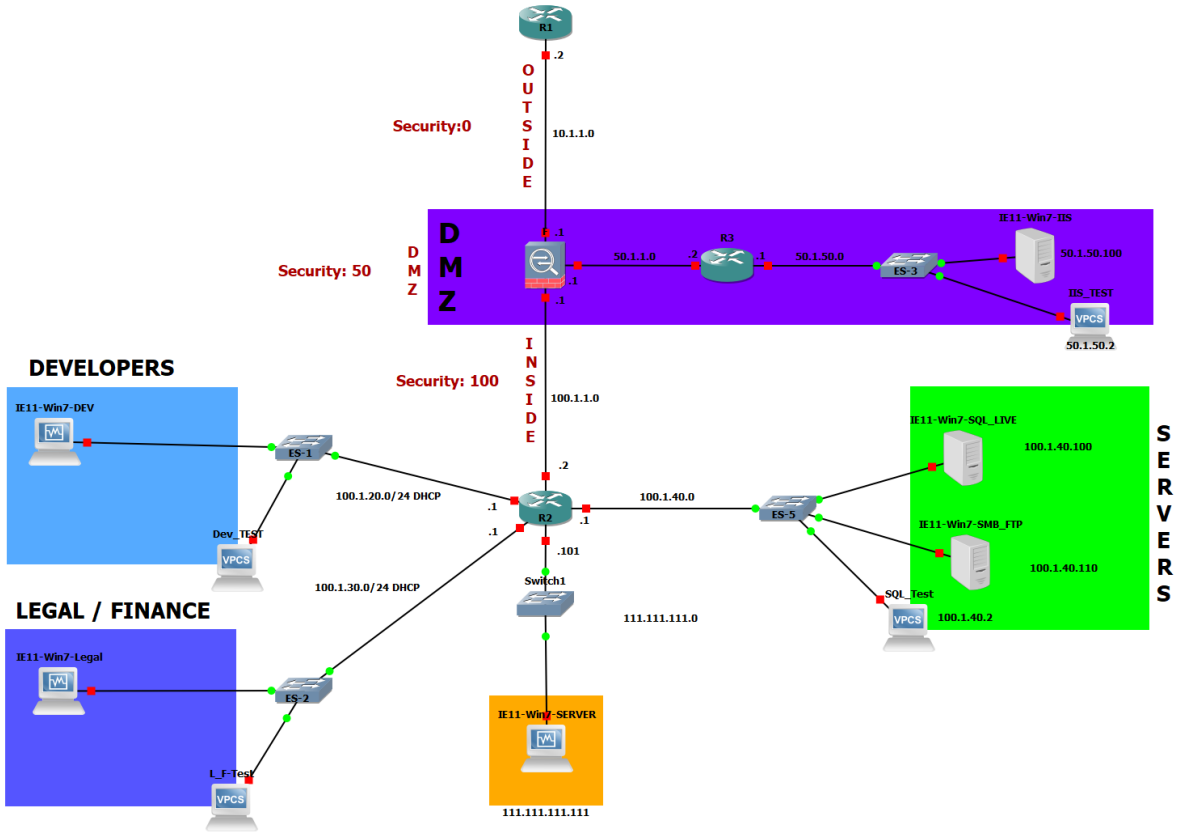


Figure 6: Medium Size Enterprise Network Topology in GNS3

[8]GNS3: `https://www.gns3.com`

## 6.1 Experiment 1

This experiment was conduct in a manual manner. Multiple Virtual Machines (VM) will be manually accessed and try to establish a connection to a multiple services across multiple VM's in order to see how the real time network mapping implementation will generate and present the graph.

Referring to [Figure 6] of network topology in GNS3 this Table 2 describes VM's running at the beginning of the experiment which generates the initial Network Map On Service Level [Figure 7].

Table 2: VM's Running At Start

| IP | Description |
|---|---|
| 111.111.111.111 | Server which centralises clients data for Network Map generation |
| 100.1.40.100 | Runs SQL server on port 1443 |
| 100.1.40.110 | Runs SMB service to share files on port 445 |
| 50.1.50.100 | Runs IIS server that serves a web page on port 80 |
| 100.1.20.2 | Test VM that will be connecting to various services |
| 100.1.30.2 | Test VM that will be connecting to various services but is not running CSA |

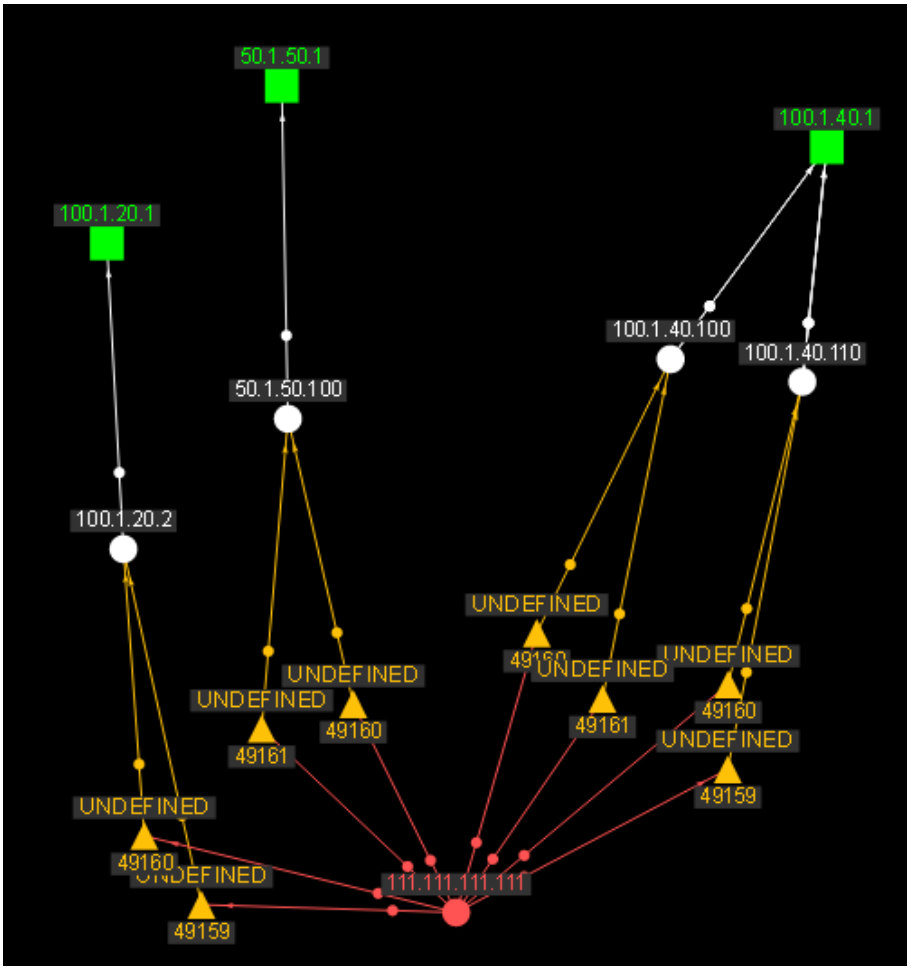


Figure 7: Initial Service Level Network Map

After manually establishing connections from different VM's and referencing Table 3 the following [Figure 8] network map was generated with the current implementation of the solution.

Table 3: VM's Connections

| IP Source | IP Destination | Service + Port |
|---|---|---|
| 100.1.20.2 | 50.1.50.100 | HTTP + 80 |
| 100.1.20.2 | 100.1.40.100 | SQL + 1433 |
| 100.1.20.2 | 100.1.40.110 | SMB + 445 |
| 50.1.50.100 | 100.1.40.100 | SQL + 1433 |
| 100.1.30.2 | 100.1.40.100 | SQL + 1433 |
| 100.1.30.2 | 100.1.40.110 | SMB + 445 |

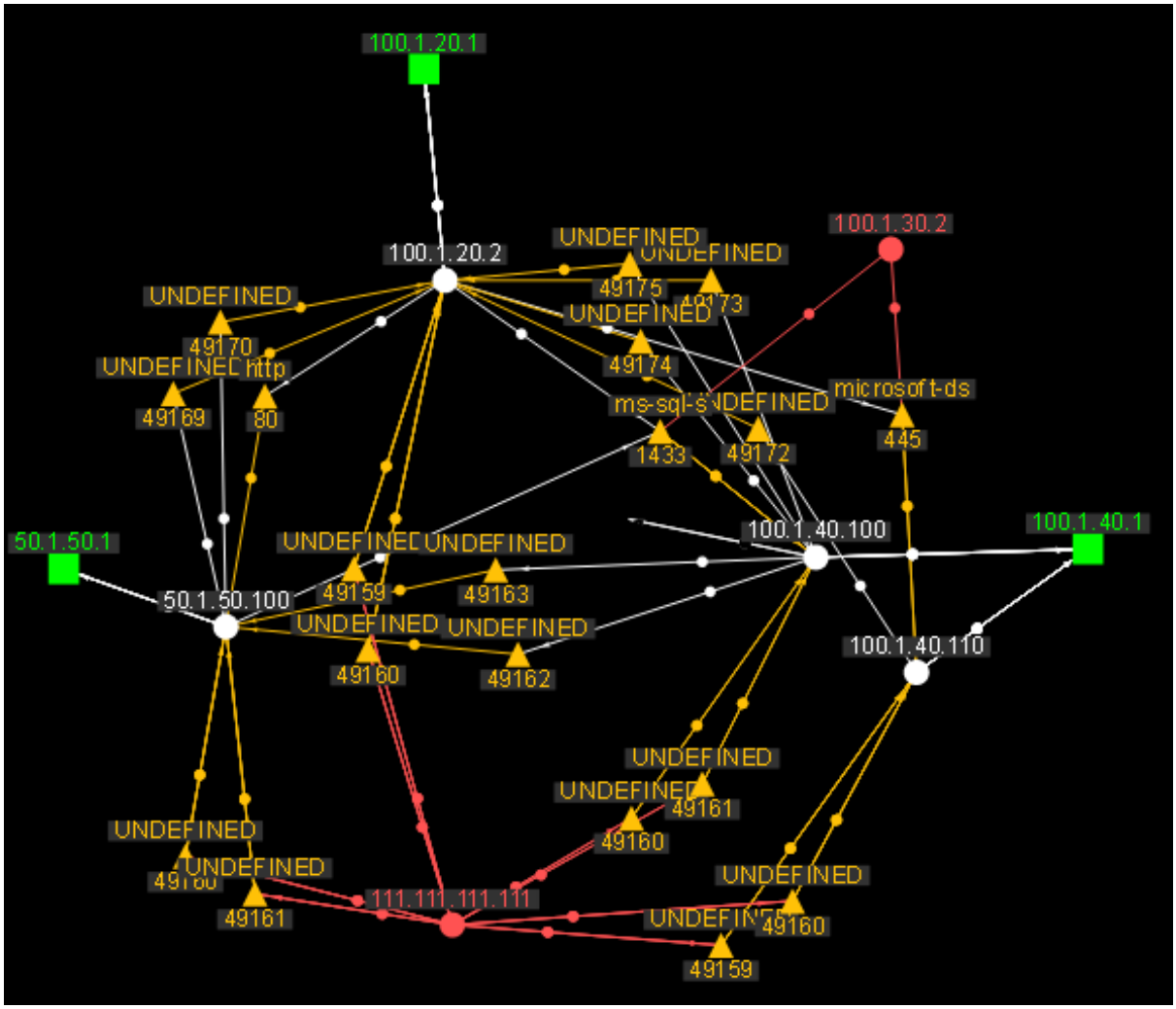


Figure 8: Generated Service Level Network Map

## 6.2 Experiment 2

Experiment was conducted in such a matter to simulate most realistic packet sending and data capture across the network between multiple end devices which are interconnected with routers, switches and firewalls. Referring to [Figure 6] the experiment took following steps:

- In DMZ 50.150.100 will send packets with raw data to 100.1.40.100
- 100.1.40.100 will capture the packet
- 100.1.40.100 will notify server 111.111.111.111
- 111.111.111.111 will forward data to UI
- UI will add single node and return confirmation to Server 111.111.111.111

In Table 4 we see a result of 1 packet statistic traveling from start to end. It was discovered that virtual environments had issues to perfectly synchronise their local time and results were very inaccurate due to these time differences.

Table 4: 1 Packet Send/Receive Time

| **Stage** | **Time** |
|---|---|
| Client1 Sent Packet | 20:47:05.699408 |
| Client2 Captured Packet | 20:47:05.002563 |
| Server Received Data | 20:47:04.979278 |
| UI Received Data From Server | 20:47:04.992 |
| Server Received Ack From UI | 20:47:05.073028 |

## 6.3 Experiment 3

In this experiment the test consisted of sending multiple amounts of packets with raw data on a single main host device which runs both server and client side application to measure applications responsiveness in an ideal environment.

When a test script is executed 5 steps are performed in order to capture event times at each stage of the application:

1. Send packet with raw data which contains time when the packet was sent
2. Client application sniffs the packet, extracts raw data and appends new record of time when packet got captured and sends the data to server
3. Server captures data sent from client and appends new record of received time and send data to UI
4. UI receives data from Server, appends record when data was received, adds single node to the graph, appends new record when process was finished, sends data back to server
5. Server receives data from UI and appends record when data was received

By sending 100 packets as fast as the End Device possibly could we can see on the graph [Figure 9] that over time the response time of application becomes slower and time grows exponentially. The average response time from 100 sent packets is 00:00:02.597 seconds. By continuing sending packets we can see how response time will become slower as UI implementation has to process and simulate tree of the responsive graph.

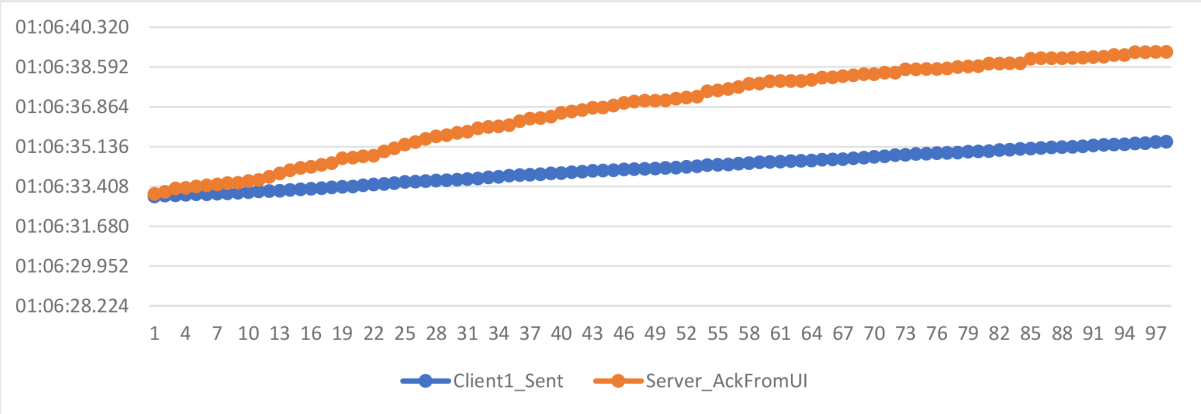


Figure 9: Sending 100 packets without delay

Another test was conducted by sending 100 packets with 0.1 seconds delay to test if this short delay is enough for UI to process data without creating a big hiccup in the process response time. The result can be seen in [Figure 10] and the average response time was 00:00:00.674 seconds. This result is much more satisfying but need to remember that it is done on the same host which runs server and client side application which creates close to ideal environment without much noise on the network.

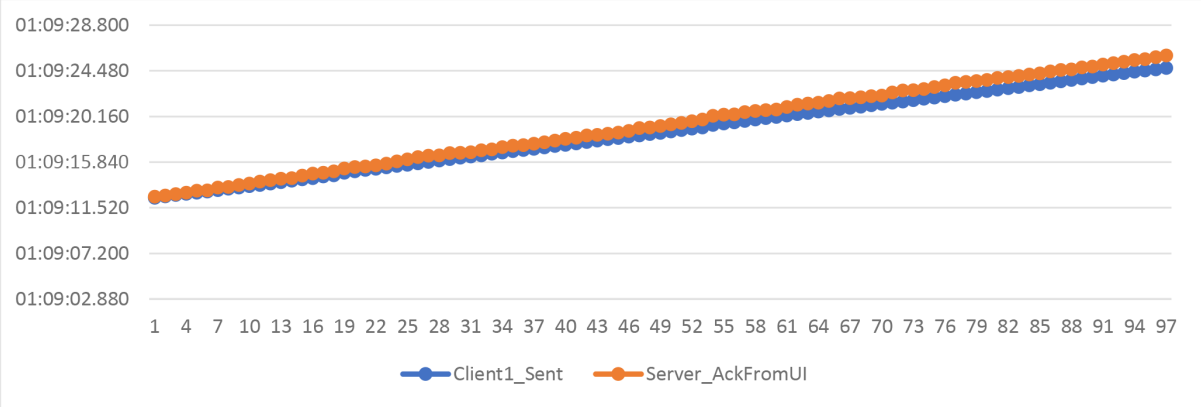


Figure 10: Sending 100 packets with 0.1 seconds delay

## 6.4 Discussion

Experiment 6.1 is the main experiment which attempts to demonstrate the power and possibilities of the proposed real time network mapping on service level solution. Upon examining the generated map [Figure 8] it can be hard at first to distinguish the information you might be looking for but over time it starts to make sense as animations and directions of the arrows help to understand which services and host connects to each other.

Major elements which are missing from this solution is the ability to identify the appliances on the network such as switches, routers and firewalls which would generate a more complete picture of the generated network map. Solution proposed by Li (2011) using a Simple Network Management Protocol(SNMP) would compliment this proposed solution in a great way and would help to complete the entire network picture which would help to scan through device connection tables to identify appliances that help to establish connection for end devices on the network.

It can also be noticed that there is possibly a big amount of unnecessary information showed on ports which are labeled as UNDEFINED. A lot of ports will return no service identified with the requested port as they are either used as automatically assigned by an application during a connection process. In this case when Client Application connects to Server it establishes a TCP connection which results in several handshakes using automated ports starting at 49100. A filtering functionality could be implemented to ignore such port connections but can be another security risk if a hidden service would be running under an undefined service name using one of the automated ports.

Another big limitation with current solution is that it doesn't recognise whether current displayed connection through a port is active or inactive. Once the connection is attempted or established it will show up on the network map but at its current state of implementation does not check for its activity.

When experiment 6.3 was conducted, number of sent packets was increased to 500 which were constantly sent to client to intercept without any delays. During the experiment after examining result set it was noticed that only 400+ packets got returned. This could be due to lack of computation power of a virtual machine resulting in dropped packets. This makes current state of implementation of the solution quite vulnerable as dropping packets might result in missing a critically important connection which will not show up on the generated network map. Another reason is because of the solutions architecture. At its current state it stores all the data in RAM and does not make use of any kind of queues for incoming data, so hick up in the system might result in dropped packets and lose of data since nothing is being stored in any kind of database which could be processed later on as resources become available.

# 7 Conclusion and Future Work

Over all the entire solution is seen as a success as it manages to create interactive graphs that display how devices interact with each other on the service level through both known and unknown services bound to ports that might be not known to both users and network administrators.

Due to the research of this solution is more technical rather than theoretical it comes with very straight forward implications which can disrupt or not create full picture of the network due to few reasons. For graphs to render there requires to be a Server which processes data from End Devices that are running the Client Side Application that sniffs network packets and determines on which port a connection is currently established. In a situation where a certain segment of a network devices do not run the CSA there will be no data sent to Server and thus not display a graph of network representation at its current state.

As a software solution it has limitless potential with the ability for Server to send events to Clients to perform specific instructions. Current version of implementation delivers the proposed solution but can be improved further by adding following features:

- Cross Platform - implement CSA version to support IOT devices
- Collapsible Nodes - limits displayed number of nodes to improve readability
- Filtering - ability to select criteria that displays graph information
- Selective Time Frame - implement ability to select network state at certain time frame or time period
- Graphics - ability to add visually represent node type, Server, Normal PC etc.
- Background Scanning - implement background active scans in order to find isolated block on the network where devices don't run Client Side Application
- Passive Scanning - add a passive scanner at the router which connects to outside world to scan network traffic and report to Server, this can help to find other undetected IP's on the network if no active scanning is in place
- Source Port - current application state displays the destination port to be used, but could also include the source port of the IP connecting to a destination IP port
- Connections Count - display count how many times certain service was accessed / connected to, those that will have unusually high count can trigger an alert of suspiciously high activity

As a software application this solution can become potentially commercially viable. For commercial viability to be achieved it would be required a significant investment to implement the proposed upgrade features and many others in order to compete with current market leaders in this field of network mapping.

# References


Bostock, M., Ogievetsky, V. and Heer, J. (2011). $D^3$ data-driven documents, *IEEE transactions on visualization and computer graphics* **17**(12): 2301–2309.

Cai, Y.-Z., Lin, C.-Y. and Tsai, M.-H. (2017). Application-aware realtime monitoring with data visualization in openflow-based network, *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, IEEE, pp. 385–390.

Clark, L. (2006). Network mapping as a diagnostic tool, *Centro Internacional de Agricultura Tropical, La Paz, Bolivia* .

Fekete, J.-D., Van Wijk, J. J., Stasko, J. T. and North, C. (2008). The value of information visualization, *Information visualization*, Springer, pp. 1–18.

Friendly, M. (2008). A brief history of data visualization, *Handbook of data visualization*, Springer, pp. 15–56.

Fruchterman, T. M. and Reingold, E. M. (1991). Graph drawing by force-directed placement, *Software: Practice and experience* **21**(11): 1129–1164.

Guozheng, Y., Yuliang, L. and Huixian, C. (2011). A new network topology visualization algorithm, *2011 First International Conference on Instrumentation, Measurement, Computer, Communication and Control*, IEEE, pp. 369–372.

Hare, C. (2011). Simple network management protocol (snmp).

Li, X. (2011). A method of network topology visualization based on snmp, *2011 First International Conference on Instrumentation, Measurement, Computer, Communication and Control*, IEEE, pp. 245–248.

Onwubiko, C. (2009). Functional requirements of situational awareness in computer network security, *2009 IEEE International Conference on Intelligence and Security Informatics*, Ieee, pp. 209–213.

Orebaugh, A. and Pinkard, B. (2011). *Nmap in the enterprise: your guide to network scanning*, Elsevier.

Rai, R. (2013). *Socket. IO Real-time Web Application Development*, Packt Publishing Ltd.

Rohith, R., Moharir, M., Shobha, G. et al. (2018). Scapy-a powerful interactive packet manipulation program, *2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS)*, IEEE, pp. 1–5.

Webster, S., Lippmann, R. and Zissman, M. (2006). Experience using active and passive mapping for network situational awareness, *Fifth IEEE International Symposium on Network Computing and Applications (NCA'06)*, IEEE, pp. 19–26.