NOVA IMS Information Management School

MDSAA

Mestrado em

Data Science and Advanced Analytics

Enhanced Network Infrastructure Visualization

A Graph-Based Approach to Asset Management and Security Monitoring

Luca Rodrigues de Sousa

Master Project Work

presented as partial requirement for obtaining a Master's Degree in Data Science and Advanced Analytics

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

Enhanced Network Infrastructure Visualization

A Graph-Based Approach to Asset Management and Security Monitoring

by

Luca Rodrigues de Sousa

Master Project Work presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics, with a specialization in Data Science

Supervised by

Flávio Luis Portas Pinheiro, PhD, NOVA Information Management School

November, 2024

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism, any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledged the Rules of Conduct and Code of Honor from the NOVA Information Management School.

Lisbon, 20 November 2024

Luca Sousa

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Prof. Dr. Flávio Luis Portas Pinheiro for his guidance throughout this research. My deepest thanks to my family and my partner Gabriela for their constant support and encouragement. I am also grateful to Sepio for providing the opportunity to conduct this research, and to NOVA IMS for the academic excellence and resources that made this work possible.

ABSTRACT

This project addresses the growing challenges in managing and monitoring complex enterprise network infrastructures through an innovative graph-based visualization system. Traditional approaches to network asset management often result in significant visibility gaps, leading to increased security incidents across organizations. Our solution combines graph database technology with human-centered design principles to improve network topology visualization and analysis.

We implement a dual-focused approach: first, leveraging graph database capabilities for efficient storage and analysis of complex network relationships; second, developing an intuitive visualization interface that reduces cognitive load in network analysis tasks. Using a modified Design Science Research methodology, we conducted three iterative development cycles with different stakeholder groups to refine the solution.

Evaluation results demonstrate significant improvements in network analysis efficiency. Key metrics show reduced task completion times and actions across critical operations: topology understanding improved from 1 minute 55 seconds with 8 actions to 44 seconds with 3 actions, cluster identification from 1 minute 33 seconds with 7 actions to 38 seconds with 2 actions, path tracing from 2 minutes 13 seconds with 10 actions to 41 seconds with 4 actions, and identifying affected infrastructure from 3 minutes 5 seconds with 12 actions to 35 seconds with 3 actions. The visualization system successfully addresses fundamental challenges in network management while maintaining performance with increasing network complexity.

This research contributes to both theoretical and practical aspects of network infrastructure management, offering insights into graph-based visualization approaches and their impact on operational efficiency in enterprise networks. Future work opportunities include integration of advanced graph algorithms, enhanced user interface capabilities, and expanded real-time processing for larger network infrastructures.

KEYWORDS

Network Infrastructure; Graph Visualization; Asset Management; Security Monitoring; Human-Computer Interaction

Sustainable Development Goals (SDG):

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AP	Access Point
CPS	Cyber-Physical Systems
DBMS	Database Management Systems
DSR	Design Science Research
HCI	Human-Computer Interaction
ID	Identifier
JSON	JavaScript Object Notation
SQL	Structured Query Language
UI	User Interface
UML	Unified Modeling Language
VM	Virtual Machine
WLC	Wireless LAN Controller

1. INTRODUCTION

Modern enterprise network infrastructures have evolved into complex cyber-physical systems (CPS), presenting significant challenges in asset management and security monitoring. According to research by Enterprise Strategy Group (ESG) and Axonius (2021), 79% of organizations acknowledge significant visibility gaps in their IT infrastructure, with visibility gaps leading to 3.3 times more security incidents.

These challenges stem from two main factors: technology limitations in precise asset identification and the data challenge of maintaining complex interconnected information in real time while enabling comprehensible insights. Traditional relational or key-value databases compound these challenges, making data maintenance, processing, and querying increasingly difficult as network complexity grows (Rodrigues et al. 2023).

Leveraged by an innovative technology that enables almost complete asset visibility (Sepio, n.d.), and sitting at the intersection of graph theory, database technology, and human-computer interaction this project focuses on two primary objectives: First, to implement graph database capabilities to store and analyze complex network information efficiently, moving beyond traditional database limitations, the approach enables efficient relationship traversal and analysis that scales with network complexity; Secondly, to create an intuitive user experience grounded in Human-Computer Interaction (HCI) principles, transforming complex network data into comprehensible visual representations.

Key stakeholders such as system administrators and Chief Information Security Officers (CISOs) require practical tools to understand and analyze their network's behavior, topology, and structural characteristics. The visualization approach emphasizes cognitive ergonomics and user experience principles to reduce the cognitive load associated with complex network analysis.

As enterprise networks continue to grow in complexity, the ability to effectively visualize and analyze network topology becomes crucial for security management and operational efficiency. The proposed approach addresses a fundamental gap in network management tools, offering a solution combining sophisticated data management with intuitive visualization capabilities.

The remainder of this document is organized as follows: Chapter 2 presents the literature review, covering graph theory, database technologies, and visualization approaches. Chapter 3 details the methodology and the data. Chapter 4 describes the implementation and technical solutions. Chapter 5 presents results and discussion. Finally, Chapter 6 concludes with findings and future work recommendations.

2. LITERATURE REVIEW

2.1 Graph Theory and Network Modeling

Graph theory provides a fundamental framework for understanding and analyzing networks through a simple yet powerful concept: modeling objects as vertices and the representation of relationships as edges (Majeed & Rauf, 2020). A vertex (or node) represents any entity we wish to study, while edges (or relationships) show how these entities connect to each other.

The image below helps with drawing the basic Graph Theory, as described by Wilson (1996). Graph G consists of a nonempty finite set V(G) of segments called vertices (nodes) and a finite set E(G) of distinct unordered pairs of distinct elements of V(G) called edges. A graph is a join G = (V,E) of sets where $V \cap E = \emptyset$ to avoid ambiguity of notation. The elements of V are the vertices or nodes of the graph. The elements of E are the edges. In the example below, the graph on V = {A,B,C} and E = {(A,B), (A,C), (B,C)} with weights 2, 3, and 4 respectively.

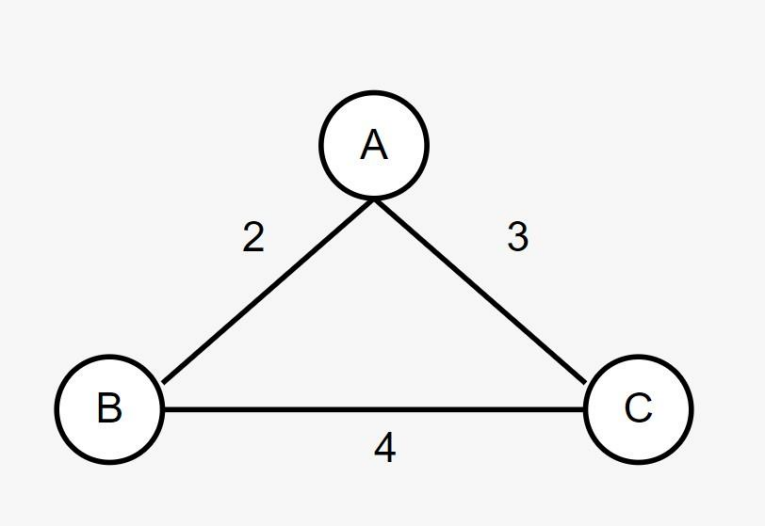


Figure 2.1: Example of a Graph.

By extrapolating this example to concrete cases, it is easy to see how graphs can represent interconnected data. Take for example Lisbon's Metro system, where each station is a Node with Edges (paths) to other stations. Clearly, representing the subway system like this makes it much easier for a user to understand how to get by. Not only that, but also helps planners define where the intersections should be, improving efficiency and user experience.

Figure 2.2: Lisbon Subway Map (Source: Metro Lisboa, retrieved from www.metrolisboa.pt/en/wp-content/uploads/sites/3/2024/07/Diagrama-da-rede-.png, November, 25, 2024)

Graphs are also useful for many other tasks, from sports competitions like football tournaments (Baratela et al., 2024) to the complexities of social networks interaction (Vanderstay, 2023). Researchers have noted that graph theory is extensively used in computer science for tasks like data mining, image processing, network design, and clustering (Mevegar et al., 2024). These methods also support route optimization, troubleshooting network problems, and strategic LAN planning, as highlighted by Ahmed (2012) and Makeri (2019).

More specifically for this project, network topology analysis benefits leveraging graph theory because it naturally captures relationships that traditional data models struggle to represent. Just like a subway System, Computer Networks function as object transferring networks systems, instead of people, it carries internet data, and the way it is distributed and designed is very important to the users and to the engineers and architects that design it. The topology plays a crucial role in determining overall network functionality and efficiency. The

configuration of nodes and links directly impacts various aspects of network performance, security and management. A well-structured topology can help promote efficient data transmission, minimize downtime and simplify troubleshooting (Khan, Jackson, and Goodwin, 2024).

In the cyber security domain, existing platforms explore graph capabilities to display Networks. Datadog, for example, analyzes network traffic across applications, containers, availability zones and on-premises servers.

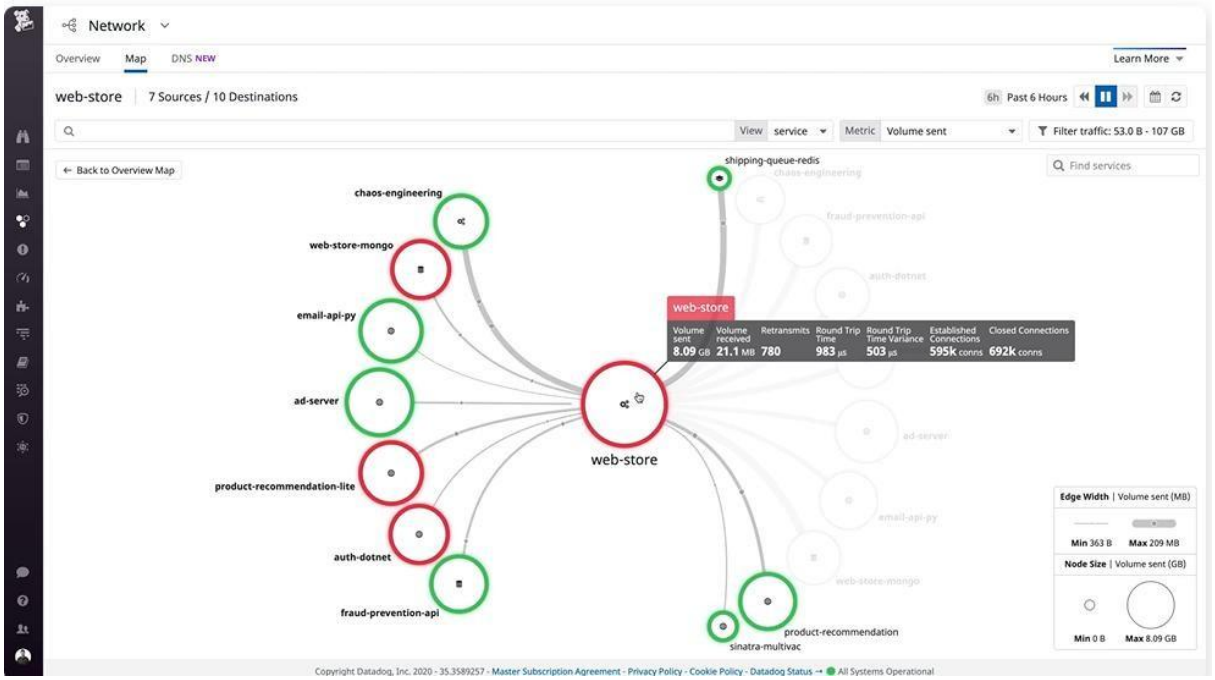


Figure 2.3: Datadog Network Performance Monitoring (Source: Datadog, retrieved from https://www.datadoghq.com/dg/monitor/network/network-new/?utm_source=comparitech&utm_medium=review-site&utm_campaign=dg-network-ww-comparitech-network-topology-and-mapping, November 2024)

This visualization means more than just the User Interface. While a traditional database might need complex joins and recursive queries to find indirect connections, a graph makes such relationships more easily accessible and visible through its edges. When analyzing a computer network, for example, we can easily trace paths between devices, identify bottlenecks, and understand the impact of potential failures (Mevegar, Saraswathi, & Halwegar, 2024).

2.2 Graph Databases

Graph databases are specialized data management systems designed to handle highly interconnected data. Unlike traditional relational databases (RDBMS), which organize data into tables with predefined schemas, or key-value stores, which store simple key-value pairs, graph databases model data as nodes, relationships, and properties. This structure mirrors the way real-world networks—such as social graphs, communication networks, and supply chains—are naturally organized.

In graph databases, relationships between entities are first-class citizens, explicitly stored alongside the data rather than being derived at query time. This property graph model enables efficient traversal of connections, even in dense networks, as relationships are natively represented in the database's storage layer (Stegeman, 2023).

Prominent graph databases include Neo4j, which pioneered the property graph approach, and others like ArangoDB and Amazon Neptune. These systems typically employ specialized query languages such as Cypher, Neo4j's declarative language, which allow intuitive and powerful operations for traversing, matching, and analyzing complex graph structures (Neo4j, n.d.).

This architectural paradigm enables significant performance advantages: while traditional databases require operations proportional to the dataset size for relationship traversal, native graph databases achieve constant-time relationship navigation regardless of total database size (Stegeman, 2023). Empirical evidence supports this theoretical advantage, with Rodrigues et al. (2023) demonstrating significant performance improvements when migrating relationship-intensive queries from traditional databases to Neo4j (detailed query analysis available in Annex 1).

TABLE III
PERFORMANCE COMPARISON BETWEEN
NEO4J & MYSQL

Category	Query	Neo4j	MySQL
Selection	Q1	2ms	31ms
	Q2	8ms	323ms
	Q3	32ms	438ms
Recursive	Q4	2ms	757ms
	Q5	2ms	290ms
	Q6	3ms	305ms
Aggregation	Q7	43ms	146ms
	Q8	18ms	40ms
	Q9	62ms	290ms
Pattern Matching	Q10	5ms	360ms
	Q11	10ms	455ms
	Q12	1ms	68ms

Figure 2.4: Performance comparison of query execution times between Neo4j and MySQL across different query categories including Selection, Recursive, Aggregation, and Pattern Matching operations. Reproduced from Rodrigues et al. (2023).

2.3 Human Computer Interaction (HCI) in Network Visualization

Human-Computer Interaction (HCI) focuses on the study and design of interactive systems that facilitate effective communication between users and computers. In the context of network visualization, HCI principles are essential for creating interfaces that optimize user experience by ensuring users can interact with complex data in ways that promote efficient and accurate decision-making. As network structures grow in size and complexity, designing visualizations that users can intuitively understand becomes an increasing challenge. Effective HCI in network visualization involves reducing cognitive load and enhancing user comprehension through thoughtful design choices (Shneiderman, 1998; Preece, Rogers, & Sharp, 2015).

The core principles of HCI that are particularly relevant to network visualization include **usability**, **feedback**, **control**, and **adaptability**. **Usability** ensures that the system is easy to use and meets the user's needs without overwhelming them with unnecessary complexity (Nielsen, 1993). **Feedback** is critical in HCI, as users must receive immediate and clear responses when interacting with the visualization, helping them refine their mental models of

the data (Norman, 1988). Providing **user control** is another essential principle; it allows users to adjust the visualization to suit their specific needs, such as zooming, filtering, or changing the level of detail displayed (Card, Mackinlay, & Shneiderman, 1999). Finally, **adaptability** ensures that the system can accommodate a range of user preferences, adapting to both expert users who require detailed views and novice users who benefit from simplified representations (Preece, Rogers, & Sharp, 2015).

Cognitive load theory, a foundational concept in HCI, suggests that effective visualizations should balance **intrinsic load**, which comes from the inherent complexity of network data, **extraneous load**, which arises from the design of the visualization interface, and **germane load**, which supports the creation of mental models of the network (Sweller, 1988; Mayer & Moreno, 2003). These insights guide the development of visualizations that reduce unnecessary complexity while enhancing the user's ability to construct accurate mental models of the network's structure.

As users interact with network visualizations, it is crucial that the design facilitates both high-level pattern recognition and detailed data exploration. Interactive features such as filtering, zooming, and highlighting allow users to customize the level of detail presented, ensuring the visualization adapts to the user's specific needs and context (Shneiderman, 1998; Ware, 2012). This balance between completeness and efficiency is key to creating visualizations that help users make faster, more accurate decisions, particularly in high-stakes environments like network security or logistics.

2.4 Business Background

The implementation of real-time network visualization systems presents unique challenges at the intersection of performance optimization and user experience design. Modern network environments generate continuous streams of topology changes, requiring visualization systems to balance update frequency with system responsiveness and user comprehension (Hussein et al., 2023). This balance becomes particularly critical in security contexts, where rapid identification of network changes can have significant operational implications.

Current Cyber-Physical Systems (CPS) solutions for network topology mapping predominantly utilize map visualizations to display network data (Thielemann & Voster, 2024). Leading solutions such as Cisco Thousand Eyes, Nozomi Networks, and Dragos primarily rely on traffic monitoring from Layer 2 upward in the OSI Model[1]. However, these approaches may miss critical infrastructure elements that are only detectable at the physical layer.

The key differentiator in our approach lies in data acquisition methodology. While existing solutions focus on higher network layers, our system captures data from Layer 1 (physical layer), enabled by Sepio's proprietary technology (Sepio, n.d.). This fundamental difference enables complete asset visibility at the hardware level, detection of passive network elements, and identification of unauthorized physical devices.

Turning the raw data into a comprehensive network is where Sepio's core business value resides. The main logic behind the data transformation will be further explored in the Data section. The scope of this project, however, resides on the value added by creating a proper storage layer for our data and finding better ways to present it visually. Additionally, position us to, in the future, explore business opportunities of insights and algorithms leveraged by graph structures.

[1] See Appendix B for detailed information on OSI Model.

3. METHODOLOGY AND DATA

3.1 Methodology

This research follows a modified Design Science Research (DSR) methodology, based on the framework proposed by vom Brocke et al. (2020). DSR is particularly effective for developing and evaluating IT artifacts designed to address specific organizational challenges. Our adaptation consolidates the traditional six-step process into five components by combining Demonstration and Evaluation phases. The resulting framework consists of 1) Problem Identification, 2) Solution Objective Definition, 3) Design & Development, 4) Demonstration & Evaluation, and 5) Communication.

Following these guidelines, researchers can either iterate back to the design phase to enhance the artifact's effectiveness or proceed to communication, leaving further improvements for future research. In our implementation, we conducted three iterative cycles, each structured in distinct layers to maintain development granularity. This layered approach enabled focused attention on specific aspects during each iteration, ensuring systematic progression and thorough refinement of the solution.

3.1.1 Problem Identification

Iin a computer network system, objects are interconnected, forming large clusters of devices. Understanding these connections often extends beyond first-order links. Consider the scenario depicted in Figure 3.1: a malicious device is connected to a PC, which in turn connects to a switch that interfaces with a larger interconnected network. If we examine only the first-order connections of this switch, we gain insight into its immediate network relationships. However, the switch may connect to multiple other switches, significantly complicating our understanding of the network's overall structure and the potential impact of the malicious device.

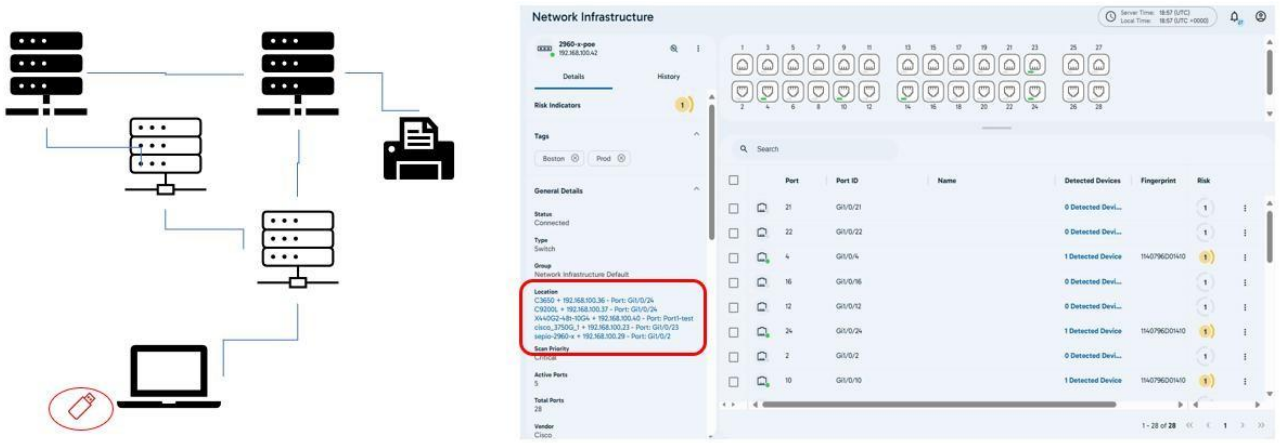


Figure 3.1: Example Network Diagram and Switch details visualization.

User feedback has indicated a desire for improved visualization navigation, which prompted the decision to implement a topology mapping feature. To achieve this efficiently, we must transfer data from our database to the visualization tool in a compatible format. As previously discussed, using relational or key-value databases for this purpose imposes higher computational costs. Addressing this challenge required an engineering decision, leading to the adoption of a node-link diagram as a natural and efficient representation of a network composed of interconnected objects.

Thus, our problem spans two branches. The system engineering aspect, or how to design a system to support the solution to the second problem, which, in turn, is how to enhance user experience on understanding network topology.

Through informal interviews and discussions with relevant stakeholders, namely product designers, customer support engineers and customer representatives, we identified key user requirements which can be validated against Nielsen's Usability Heuristics and the Task-Technology Fit (TTF) model (Goodhue & Thompson, 1995). The findings reveal two fundamental usability challenges in current network visualization approaches. First, users struggle to understand relationships beyond first-degree connections when presented with tabular data, limiting their comprehension of network topology depth. This aligns with Nielsen's "Visibility of System Status" heuristic, highlighting the critical need for clear representation of system relationships. Second, the requirement for multiple clicks to navigate between assets hinders users' ability to maintain a holistic network perspective,

increasing cognitive load and corresponding to Nielsen's principle of minimizing user memory load.

The goals were initially defined by consulting two key sources. First, we referred to Gartner's guidelines. Gartner is a leading research and advisory company that provides insights, advice, and tools for businesses and IT leaders (Gartner, 2024). They offer documentation outlining the features required to be considered a market- reference Cyber-Physical System. Therefore, we focused on the following directive:

- **Topology and Data Flows Mapping**: This involves visualizing linkages and data flows between all assets and sites. Common formats include Purdue model maps, VLAN maps, zone cluster maps, external communications data flow maps, or asset and group maps. Many of these visualizations also offer drilldown capabilities. (Gartner, proprietary document, 2024)

Having this as the defining requirement, we secondly conducted conversations with stakeholders to refine what our final feature would look like.

3.1.2 Definition of Objective

After understanding user's needs. We asked them what features would solve the existing limitations. After some back and forth with customers, conducted in the form of conversations, we came up with the following set of tasks as main objectives.

Table 3.1: Requirements and objectives gathered from stakeholders outlining task characteristics, technology requirements, and their corresponding validation criteria for network visualization system.

Task Characteristic	**Technology Requirement**	**Validation Criteria**
Full network visualization	Interactive graph display	Reduces cognitive load, improves system visibility

Asset Identification	Graph should effectively communicate Asset Type and ID	Enhances information access
Connection path tracking	Path highlighting and tracing	Supports task completion, reduces navigation effort
Risk Score visualization	Visual risk score	Supports decision making

Users require quality through complete network topology visibility, locatability through efficient identification and path tracking, and compatibility with existing security workflows. The requirements effectively address different types of cognitive load in network visualization tasks. Intrinsic load is reduced through clear visualization of network relationships and simplified path tracking. Extraneous load is minimized by eliminating multiple-click navigation and implementing intuitive filtering mechanisms. Germane load is supported through holistic network visualization and enhanced by clear risk highlighting.

This analysis validates that the identified requirements align with established HCI principles while addressing fundamental user needs for network topology visualization. The prioritization reflects both the immediate operational needs and the longer-term strategic value of each feature in supporting effective network management and security monitoring.

3.1.3 Design and Development

The design and development process incorporated three iterative cycles, each targeting different stakeholder groups to ensure comprehensive feedback and progressive refinement. We established distinct milestones with specific stakeholders: beginning with product team feedback, followed by Customer Support Engineers and culminating in direct customer input. This structured approach allowed for systematic evaluation and improvement at each stage,

ensuring that development priorities aligned with both internal requirements and end-user needs.

Table 3.1: Stakeholder feedback cycles.

Cycle	Stakeholder	Feedback
1	Product	1) Show only connected assets. 2) Differentiate Asset Type not only by Icon, but also by color 3) Possibility to turn on and off the feature
2	Customer Support Engineers	1) Graph rendering should be faster 2) Show ID of port in CONNECTED_TO_NET_INFRA relationship when target is a Switch 3) Show only connected assets.
3	Customers	4) Ability to see map in smaller clusters (Not Implemented in this project)

The iterative nature of this process allowed us to address critical functionality and usability concerns progressively, with each stakeholder group providing unique perspectives based on their interaction with the system. Product team feedback focused on core functionality, while Customer Support Engineers provided insights into practical operational needs. The final cycle with end customers was designed to validate the implementation against real-world usage scenarios.

3.2 Techniques and Tools

The implementation of our network visualization system follows a microservices architecture pattern, comprising three main components: a backend processing service, a graph database instance, and a frontend visualization interface. This architecture ensures separation of concerns while maintaining system cohesion and scalability (Hussein et al., 2023).

The system architecture is designed to support efficient data processing, storage, and visualization of complex network topologies. Each component serves a specific purpose while maintaining loose coupling for system resilience and maintainability.

3.2.1 Database Layer

At the core of the system lies a Neo4j graph database instance, chosen for its native graph processing capabilities and optimized query performance for relationship-intensive operations. Neo4j's property graph model aligns naturally with network topology representation, where nodes represent network assets and edges represent their relationships.

3.2.2 Backend Service Architecture

The backend service, implemented in Python, comprises two primary components:

The Data Ingestion and Processing component handles the transformation of raw network data into graph structures, implementing sophisticated proprietary algorithms for relationship mapping and asset classification. This component ensures data consistency and maintains real-time synchronization with network changes.

The API Service exposes RESTful endpoints for data consumption by the frontend, implementing necessary security controls and optimizing query performance. This service layer abstracts the complexity of graph operations from the frontend while providing efficient data access patterns.

3.2.3 Frontend Visualization Engine

The frontend implementation leverages modern web technologies to provide performant and interactive network visualization. The visualization engine employs a two-phase rendering approach:

First, a dedicated service worker performs Cartesian plane calculations for node positioning, optimizing layout algorithms off the main thread. Second, a WebGL-based rendering pipeline,

built on top of D3.js, provides hardware-accelerated graphics rendering, ensuring smooth interaction even with large network graphs.

3.3 Data

The data acquisition process relies primarily on network sensors that provide information in JSON format. This data undergoes comprehensive processing through specialized logic blocks to determine three fundamental aspects: first-degree connections (termed Asset Parent), risk assessment, and asset classification.

In our architecture, 'Asset' represents the highest-level object instance, encompassing all network entity types. The system primarily differentiates between Base Network Elements and Ports. While Ports are implemented as Assets to maintain logical consistency in relationship mapping, they are distinct from Network Elements as they fundamentally serve as interfaces for Switches rather than independent network entities.

Within network elements, the system distinguishes between supervised and unsupervised assets. Supervised Assets represent directly monitored network elements that provide primary scan data, enabling comprehensive attribute collection. Conversely, unsupervised Network Elements are discovered through their connections to supervised assets. These unsupervised assets can represent any type of network element - switches, WLCs, APs, or PCs - though initially identified only by MAC address from sensor data. Through internal logic and knowledge base integration, the system derives additional attributes including asset type, model, and vendor information. Furthermore, by analyzing relationships with first-degree parents (identified in current scans), the system determines network parameters such as SSID, IP Address, and VLAN information.

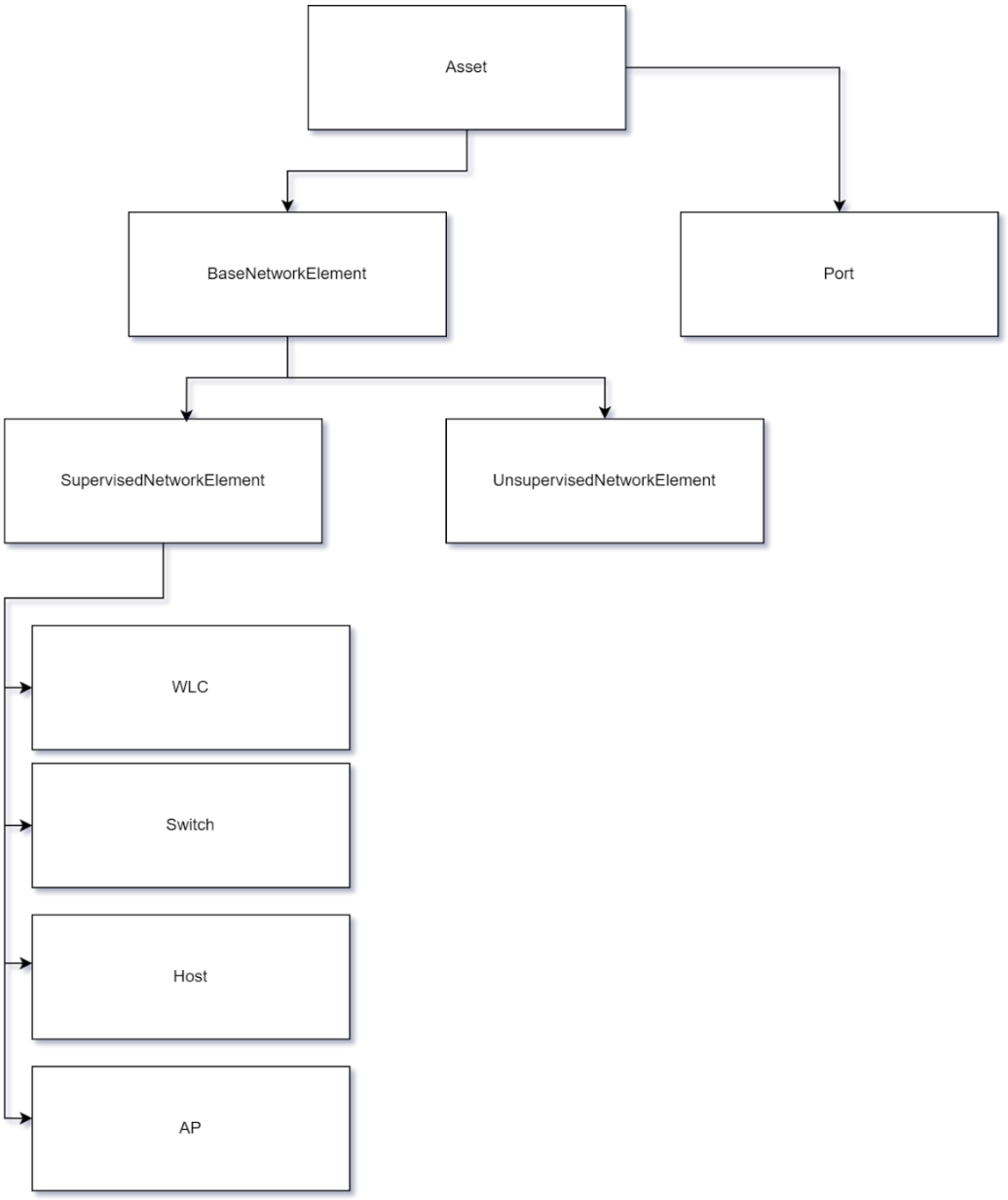


Figure 3.2: Network asset classification hierarchy showing the relationship between supervised and unsupervised network elements. The diagram uses standard UML (Unified Modeling Language) notation with boxes representing classes and arrows indicating inheritance.

We process structured data representing Network Infrastructure Devices. For instance, a Switch entity contains identification attributes and an array of associated Ports, each potentially linking to other network elements. This structure extends to other device types, such as WLCs with their associated Access Points. From this data, beyond extracted and calculated intelligence (risks and classification), our project emphasizes location determination. The system can identify specific asset connections, for example, determining that certain network elements connect to a specific switch through ports.

```
{
    "SwitchID": "E069BA5FDE37_FDO25480VDF",
    "Ports": [
        {   "PortID" "Gi/0/1",
            "LinkPartnerData": " 6DF01ACBD0543",

        }
    ]
}
```

Figure 3.3: JSON representation of a switch configuration showing the switch ID and port information with its corresponding link partner data.

Raw JSON payloads undergo transformation into a graph-based data model where:

- Nodes represent network assets with their associated attributes
- Edges define relationships between assets based on connection types
- Classification metadata enables asset categorization and risk assessment

```python
class AssetNode(BaseModel):
    """Maps asset query results to a Pydantic model."""

    asset_id: str
    hashed_id: str
    asset_display_name: str | None
    classification_id: int
    tags: list[Tag] = []
    asset_type: AssetType
    asset_parents: list[ParentData]
    connection_status: str
    last_seen: str
    is_asset_supervised: bool = False
    risk_score: int = 1
```

Figure 3.4: Asset Node Object definition containing all relevant attributes to be stored per node.

```python
class Relationship(BaseModel):
    """Maps relationship query results to a Pydantic model."""

    model_config = ConfigDict(
        alias_generator=alias_generators.to_camel,
        populate_by_name=True,
        from_attributes=True,
    )
    source: str
    target: str
    source_display_name: str | None
    target_display_name: str | None
    via_port_id: str | None = None
    updated_at: str | None
```

Figure 3.5: Relationship object

At its core, this fundamental data structure consists of Asset-type nodes, each encapsulating domain-specific attributes for analytical and visualization purposes. The attributes can be seen in figure 3.4.

The data model implementation reflects the inherent structure of network topologies while optimizing for graph database operations. Asset nodes are implemented as primary entities with flexible attribute systems, supporting multiple asset types while maintaining strict type constraints. Relationship implementations carry specific semantic meaning within the network topology context.

Node properties are dynamically inherited from Stream message consumption and undergo periodic updates during scanning intervals. Within this framework, relationship topology is characterized by two primary edge types: CONNECTED_TO and CONNECTED_TO_NET_INFRA. Such differentiation of relationship classifications stems from the network topology's hierarchical structure, wherein network assets establish connectivity through switch port interfaces. While ports function as subordinate components of switches rather than autonomous assets, this architectural decision enables dual representation models.

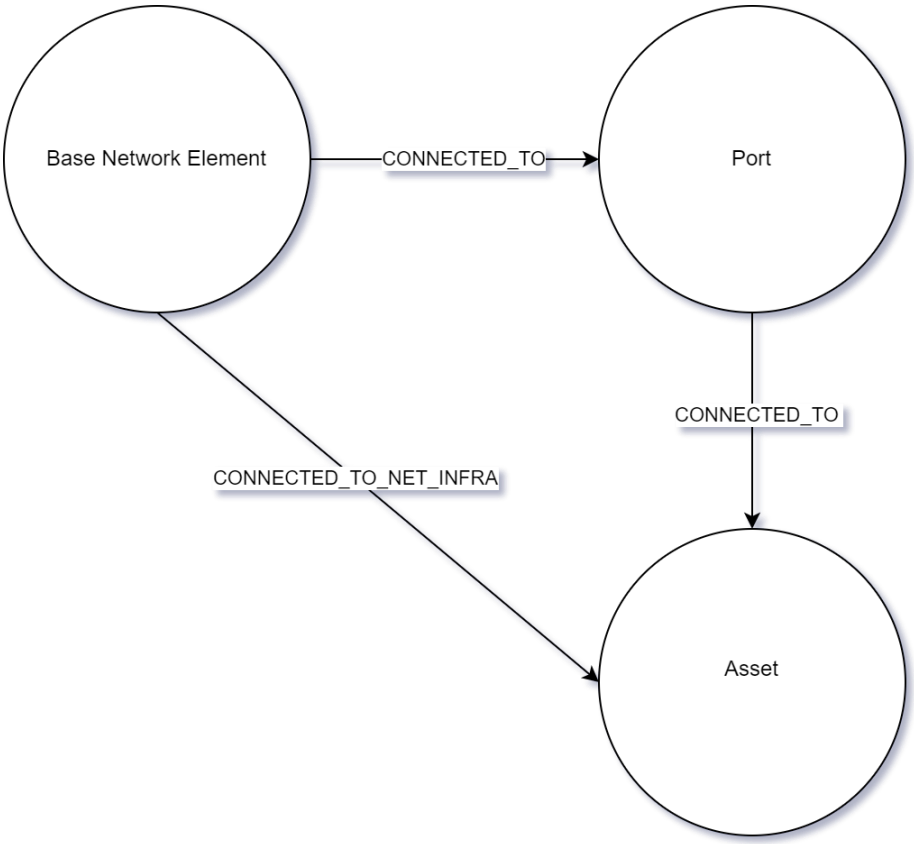


Figure 3.6: General representation of network element relationships showing the connection types between base network elements, ports, and assets. The diagram illustrates two distinct relationship types: CONNECTED_TO and CONNECTED_TO_NET_INFRA.

Building upon this foundation, a simplified model abstracts port-level details, thereby optimizing graph visualization by reducing visual complexity and prioritizing switch-level connectivity for end-user comprehension. Conversely, its detailed topology counterpart incorporates port-level granularity.

Retention of port-level information in the data model serves two critical functions: enabling more detailed network topology mapping and enabling future analytical model accuracy through improved connectivity granularity. One possible model to be implemented is network anomaly detection, for example. This model benefits of having more information of how a Switch is usually seen. Having the port layer might increase our depth of understanding of how a network is composed and make such algorithms more effective.

Through this dual representation approach, both high-level network visualization and detailed technical analysis become possible, facilitating context-appropriate information display and analysis.

Asset-type nodes form structural foundation, implementing dynamic property inheritance through periodic scanning updates. To also provide essential for risk assessment and mitigation scenarios, precise asset management, and infrastructure modification planning, without requiring alternating the view, we found ideal to add a property called *via_port_id* in the relationship between a Network Element and a Switch. Additionally, the *updated_at* field incorporates a timeframe scope.

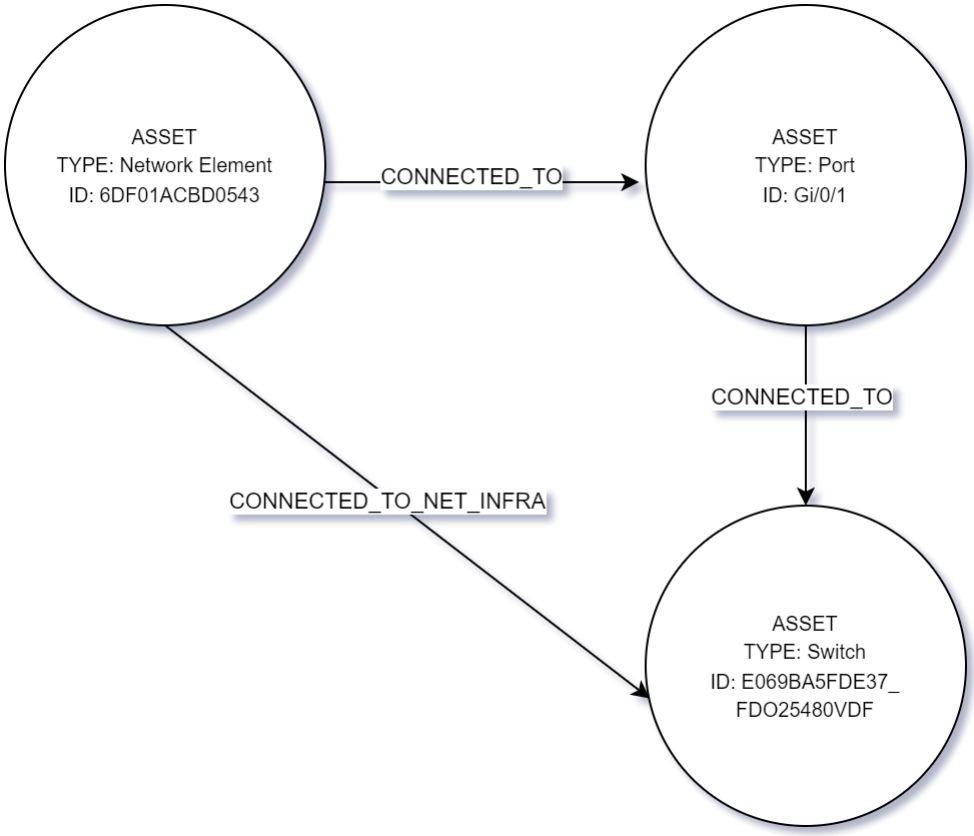


Figure 3.7: Representation of what the Data displayed in Figure 6 would be translated to in our Graph Model

4. RESULTS AND DISCUSSION

4.1 Data Representation

The network data model distinguishes between two types of connections: CONNECTED_TO and CONNECTED_TO_NET_INFRA, a distinction necessitated by the role of ports in network infrastructure. Our approach treats ports differently from regular assets in the default view, while maintaining the capability to expose port-level details when needed. Beyond the visualization benefits, this also improves our querying capabilities. When drilling down to a specific Switch, as seen in image 4.1, we benefit from querying a node ID (that of the Switch in interest) and all assets CONNECTED_TO it, showing up all the Ports. Having a single relationship type would require to always query all relationships and filter Ports in or out, depending on the desired view.

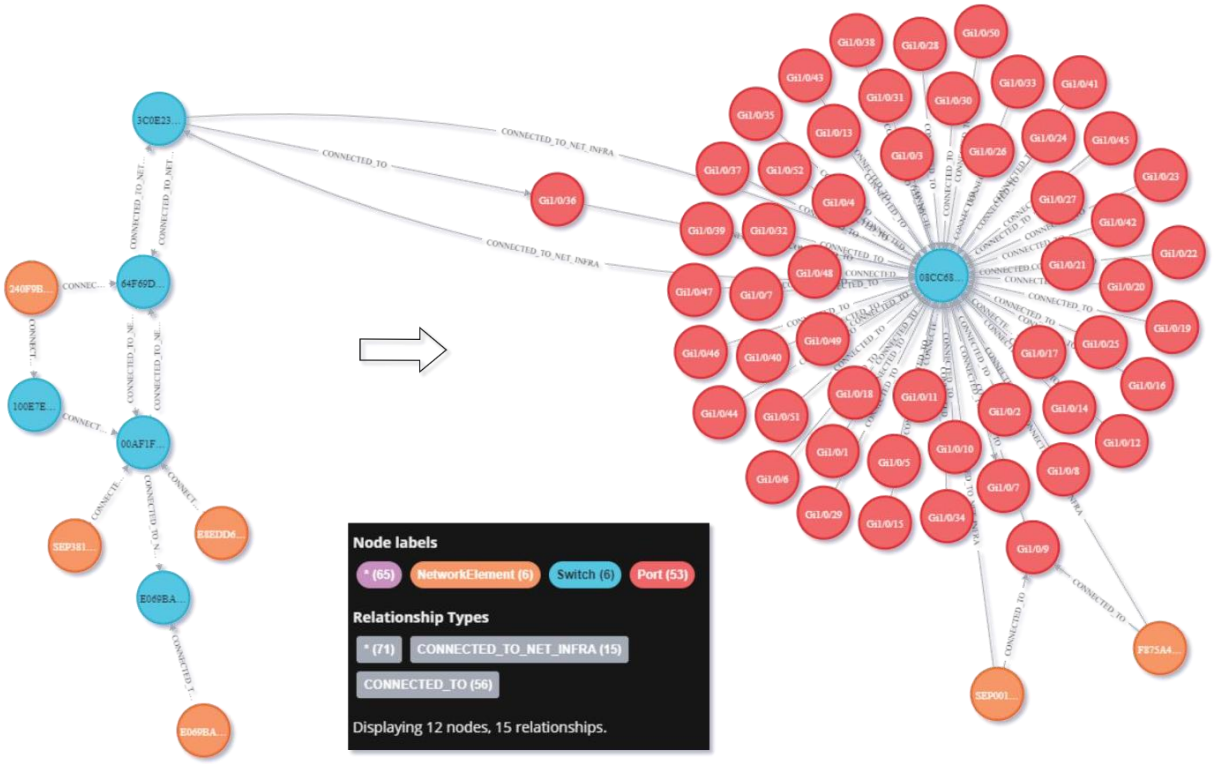


Figure 4.1: Visual comparison of network topology views demonstrating the expansion of switch port connections. The left side shows a condensed network view with basic device relationships, while the right side illustrates the detailed port-level connections of a selected switch in a radial layout. The inset shows the relationship type filtering interface that enables this detailed visualization.

4.2 End-User Topology Mapping

As demonstrated in Figure 4.1, this flexible visualization approach allows users to transition between different levels of detail. The default view presents a streamlined network topology, while the expanded view reveals detailed port-level connections for specific switches of interest. This implementation uses the Neo4j desktop solution, deployed within system Virtual Machines, providing a powerful tool for detailed network analysis. However, we recognized two key limitations of this approach:

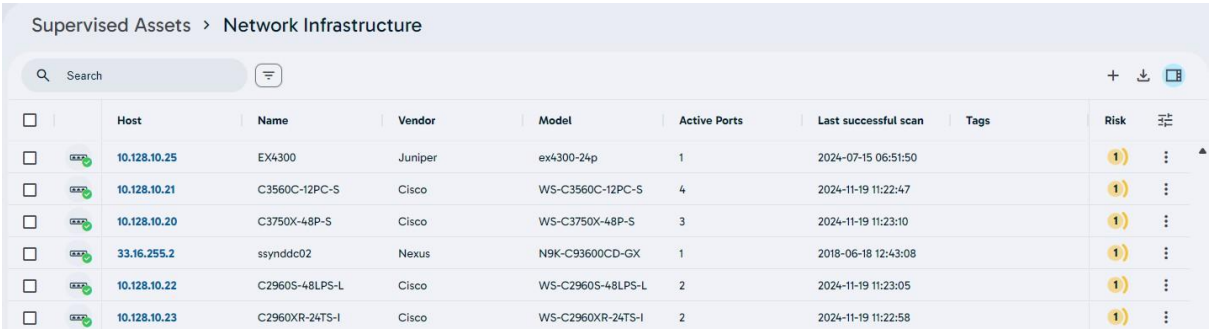
1. The Neo4j interface, while comprehensive, primarily serves advanced users and may not suit the average user's needs for network exploration.

2. Performance constraints become apparent when handling larger networks, with visible degradation occurring at approximately 500 assets.

To address these limitations, we developed a custom front-end visualization solution. While it offers fewer filtering and manipulation options compared to the Neo4j interface, it specifically targets the project's defined objectives. Our evaluation focuses on this custom visualization, though the Neo4j engine remains valuable for data model exploration and detailed analysis of specific network segments.

This implementation strategy allows us to effectively handle enterprise networks that typically contain thousands of infrastructure components, including switches and wireless controllers.

Our testing environment, though smaller in scale, successfully demonstrates both the visualization challenges and their solutions.



Host	Name	Vendor	Model	Active Ports	Last successful scan	Tags	Risk
10.128.10.25	EX4300	Juniper	ex4300-24p	1	2024-07-15 06:51:50		1
10.128.10.21	C3560C-12PC-S	Cisco	WS-C3560C-12PC-S	4	2024-11-19 11:22:47		1
10.128.10.20	C3750X-48P-S	Cisco	WS-C3750X-48P-S	3	2024-11-19 11:23:10		1
33.16.255.2	ssynddc02	Nexus	N9K-C93600CD-GX	1	2018-06-18 12:43:08		1
10.128.10.22	C2960S-48LPS-L	Cisco	WS-C2960S-48LPS-L	2	2024-11-19 11:23:05		1
10.128.10.23	C2960XR-24TS-I	Cisco	WS-C2960XR-24TS-I	2	2024-11-19 11:22:58		1

Figure 4.2: Tabular view of supervised network assets showing device details including host information, vendor, model, active ports, last successful scan timestamps, and risk indicators.

As shown in Figure 4.2, the tabular visualization presents network assets in a structured list format, displaying critical information such as device names, scan history, and status details. This approach excels at providing detailed asset-specific information and is particularly effective as a landing page for network administrators. In our test environment, which consists of six switches (approximately 30 ports per switch) and 120 total assets, this tabular view proves suitable for detailed device investigation but reveals significant limitations when users need to understand network topology.

The current interface implements a hierarchical navigation system where users can view first-degree connections through a sidebar menu. As illustrated in Figure 4.3, clicking on a specific device (e.g., switch 10.128.10.25) reveals its immediate connections via a location dropdown. However, tracing connection paths quickly becomes cognitively demanding. For instance, when following the connection stream to switch 10.128.10.21, users discover additional infrastructure connections. While this relationship may be comprehensible initially, the mental effort required to map these connections increases substantially with each additional hop in the path.

To evaluate these usability challenges, we conducted an internal study with 10 network professionals. Participants were tasked with mapping device connections starting from switch 25, with unlimited freedom to navigate the interface. The results revealed that users needed an average of 50 seconds and 9 distinct clicks to understand the connection paths. More

importantly, participants frequently failed to recall complete connection paths when dealing with more than two connection hops, especially because some connections are bi-directional, highlighting the cognitive burden imposed by the current visualization approach.

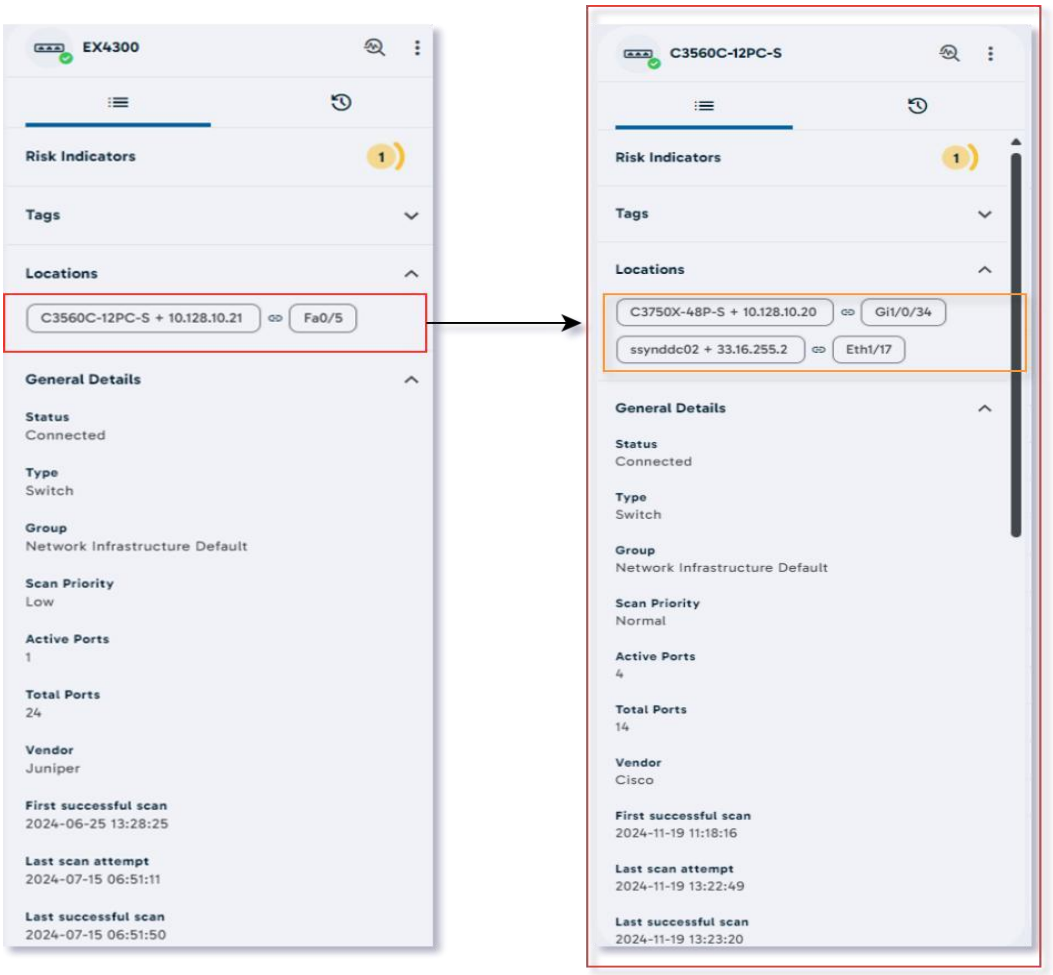


Figure 4.3: User interface comparison showing the progression from device information view (left) to detailed network component view (right), highlighting the navigation path for accessing device status information. Whereas, as seen in figure 4.4, a simple hover over switch 25 will show all the relevant connections. If the user desires to see details of the Asset, beyond the Node, clicking on it opens the page of the asset.

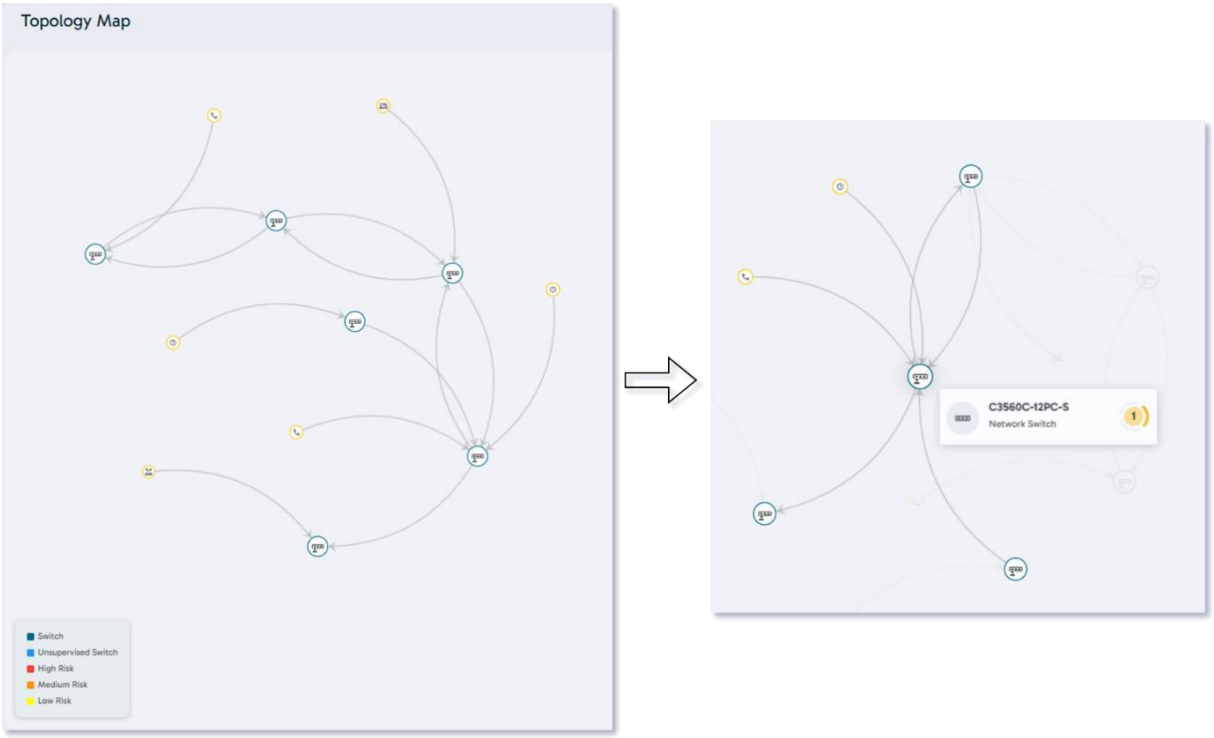


Figure 4.4: User Interface response to hovering the cursor over an Asset of interest. Highlights the Asset and all its first-degree connections.

4.3 Metrics and Evaluation

To quantitatively assess the effectiveness of our visualization solution against the project objectives, we established a structured evaluation framework focusing on four key performance indicators. Each metric was designed to measure specific aspects of user interaction and task efficiency, and the task test suite was conducted with a group of 10 participants (see Appendix C for survey details), with focus on two primary dimensions:

1. Task Completion Time: Measuring the duration required to complete specific network analysis tasks

2. Number of Actions Required: Quantifying the interaction efficiency through the count of necessary user actions (clicks, selections, etc.)

These metrics, explained below, are applied across four critical operational scenarios and measures in both the original and new implementation, enabling a direct comparative analysis of improvements.

1. Topology understanding:
    a. Task: Explore 5 unknown infrastructure network
    b. Success Criteria: user correctly explains network layout
2. Cluster Identification:
    a. Task: Identify two separate 5 infrastructure clusters
    b. Success Criteria: Identify separate clusters and list assets in each
3. Path tracing:
    a. Task: Trace path from Switch A to Switch B
    b. Scenario: Switch A -> Switch C -> Switch D -> Switch B
    c. Success Criteria: List all intermediate Assets
4. Security Impact Analysis
    a. Task: Trace affected infrastructure from malicious device
    b. Scenario: Atack tool -> Switch C -> Switch A -> Switch B
    c. Success Criteria: Identify all affected infrastructure

Table 4.1: Evaluation metrics using tabular data

Task	Average Time	Average Number of Actions
Topology Understanding	1min 55s	8
Cluster Identification	1min 33s	7
Path Tracing	2min 13s	10
Affected Infrastructure	3min 5s	12

Table 4.2: Evaluation metrics using topology map

Task	Average Time	Average Number of Actions
Topology Understanding	44s	3
Cluster Identification	38s	2
Path Tracing	41s	4
Affected Infrastructure	35s	3

The results show a significant improvement in topology understanding and overall network analysis. The main advantage is clearly time. The holistic visualization provided by the map enables much lighter cognitive exercises, allowing users to visualize connections instead of imagining or remembering them. Another point to mention is that, while the number of actions did not show such a decrease, the ratio between this metric and the average task completion shows clearly that each action comes with more information density. In the map, these actions are mainly hovering, occasionally clicking to expand asset information.

5. CONCLUSIONS AND FUTURE RESEARCH

The implementation of a graph-based visualization system for network topology management has demonstrated significant improvements in both technical performance and user experience. Performance metrics show consistent query response times even with increasing network complexity, while user feedback indicates improved comprehension of network relationships and reduced time to identify critical paths. The system successfully addresses the initial objectives: efficient storage and processing of network topology data through graph database implementation, and intuitive visualization of complex network relationships.

User studies particularly highlight three key achievements: reduced time to identify network patterns, improved understanding of asset relationships, and enhanced ability to track connection paths. The graph database implementation shows consistent performance advantages over traditional approaches, particularly in relationship-intensive queries where path finding and connectivity analysis are crucial.

However, several areas present opportunities for future enhancement:

First, the integration of advanced graph algorithms for network analysis could provide additional insights into network behavior and potential vulnerabilities. The current foundation is built to support the implementation of community detection, centrality analysis, anomaly detection algorithms, Deep Walk, Node2Vec and potentially graph based Machine Learning.

Second, the user interface could be enhanced with features like those found in Neo4j Desktop, including advanced query visualization, interactive graph exploration tools, and customizable visualization layouts. These improvements would further reduce cognitive load while providing more sophisticated analysis capabilities.

Finally, the system's real-time processing capabilities could be expanded to handle larger data volumes and more complex relationship types, particularly as network infrastructures continue to evolve and grow in complexity.

BIBLIOGRAPHICAL REFERENCES

Axonius. (2021). *Study shows 79% of organizations acknowledge an asset visibility gap, leading to 3x more incidents*. Axonius. https://www.axonius.com/press-releases/study-shows-79-percent-organizations-acknowledge-asset-visibility-gap

Baratela, E. A., Xavier, F. J., Peron, T., Villas-Boas, P. R., & Rodrigues, F. A. (2024). *Predicting soccer matches with complex networks and machine learning*. *Complex Networks*, cnae043. https://doi.org/10.1093/comnet/cnae043

Cloudflare (n.d.). *What is the OSI Model? Learning Center.* Retrieved November 15, 2024, https://www.cloudflare.com/learning/ddos/glossary/open-systems-interconnection-model-osi/

Card, S. K., Mackinlay, J. D., & Shneiderman, B. (1999). *Readings in information visualization: Using vision to think*. San Francisco, CA: Academic Press.

Exabeam. (2023). 2023 TDIR global report. Retrieved November 15, 2024 https://www.exabeam.com/resources/reports/2023-tdir-global-report/

Gartner. (2024). Insights, Advice, and Tools for Business and IT Leaders. Retrieved November 15, 2024 https://www.gartner.com/en.

Goodhue, D. L., & Thompson, R. L. (1995). *Task-Technology Fit and Individual Performance*. *MIS Quarterly*, *19*(2), 213–236. https://doi.org/10.2307/249689

Hussein, S., Lahami, M., & Torjmen, M. (2023). *Assessing the quality of microservice and monolithic architectures: Systematic literature review* [Preprint]. Research Square. https://doi.org/10.21203/rs.3.rs-3497708/v1

Khan, T., Jackson, G., & Goodwin, M. (2024). *What is network topology?* IBM. https://www.ibm.com/topics/network-topology

Majeed, A., & Rauf, I. (2020). *Graph Theory: A Comprehensive Survey about Graph Theory Applications in Computer Science and Social Networks*. Inventions, 5(1), 10. https://doi.org/10.3390/inventions5010010

Marikyan, D., & Papagiannidis, S. (2023). Task-Technology Fit: A review. In S. Papagiannidis (Ed.), TheoryHub Book. Newcastle University. https://research-

information.bris.ac.uk/files/376433308/Marikyan_and_Papagiannidis_2023_task_technology_fit_Review.pdf

Makeri, Y. A. (2019). Design and Implementation of optimized features in a local area network for improved enterprise network [Conference paper]. ResearchGate. https://rb.gy/kzf6in

Mevegar, Q., Saraswathi, B. A., & Halwegar, W. A. (2024). *A review of graph theory and its applications across various disciplines*. *International Research Journal of Engineering and Technology (IRJET)*, 11(3), e-ISSN 2395-0056, p-ISSN 2395-0072. https://www.irjet.net/archives/V11/i3/IRJET-V11I325.pdf

Neo4j. (n.d.), What is Cypher? Neo4j documentation. Retrieved November 15, 2024, https://neo4j.com/docs/getting-started/cypher/

Preece, J., Sharp, H., & Rogers, Y. (2015). *Interaction Design: Beyond Human-Computer Interaction* (4th ed.). John Wiley & Sons.

Robinson, I., Webber, J., & Eifrem, E. (2013). *Graph Databases: New Opportunities for Connected Data*. O'Reilly Media.

Rodrigues, Cajetan & Jain, Mit Ramesh & Khanchandani, Ashish. (2023). *Performance Comparison of Graph Database and Relational Database*. https://www.researchgate.net/publication/370751317_Performance_Comparison_of_Graph_Database_and_Relational_Database

Sepio. (n.d.). Asset visibility: See what you've been missing. Retrieved November 9, 2024, https://sepiocyber.com/blog/tackling-asset-visibility-complexity-axonius/

Stegeman, J. (2023, May 8). Native vs. non-native graph database. Neo4j blog. Retrieved November 20, 2024 https://neo4j.com/blog/native-vs-non-native-graph-technology/

Thielemann, K., & Voster, W. (2024). Hype Cycle for Cyber-Physical Systems Security, 2024. Gartner, Inc. https://www.gartner.com/en/documents/5601859

Thielemann, K., & Voster, W. (2023*). Market Guide for CPS Protection Platforms. Gartner, Inc*. https://www.gartner.com/en/documents/4487799

Wilson, R. J. (1996). *Introduction to graph theory*. Pearson Education Limited.

Vanderstay, M. (2023, April 3). *Viewing Twitter influence with network graphs*. https://markvanderstay.com/posts/twitter-analysis-of-retweets-with-r/index.html

vom Brocke, J., Hevner, A., & Maedche, A. (2020*). Introduction to Design Science Research*. In Design Science Research. Cases. (pp. 1-13). Springer. https://doi.org/10.1007/978-3-030-46781-4_1

APPENDIX A: RESUMED QUERY ANALYSIS FROM RODRIGUES ET AL. (2023)

A. Query Definitions and Purpose The following queries were implemented and analyzed in both Neo4j and MySQL:

Selection Queries:

- Q1: Find professionals by tag - enables searching for professionals with specific expertise
- Q2: Find students by group and tag - locates students in specific groups with interests
- Q3: Retrieve professional emails - fetches all email communications for a specific professional

Recursive Queries:

- Q4: Unlimited depth question-answers - traces question-answer chains with unlimited recursive depth
- Q5: Two-level question-answers - traces question-answer chains with maximum depth of 2
- Q6: Three-level question-answers - traces question-answer chains with maximum depth of 3

Aggregation Queries:

- Q7: Professional answer count - calculates total number of professionals who provided answers
- Q8: Tagged professionals count - determines number of professionals with specific expertise tags
- Q9: Most common professional tag - identifies the tag associated with the highest number of professionals

Pattern Matching Queries:

- Q10: Tagged answered questions - matches questions that have answers along with their associated tags
- Q11: Group membership patterns - identifies student-professional pairs who share group memberships
- Q12: Expertise overlap patterns - discovers student-expert pairs who share common interest tags

B. Performance Results Analysis The execution time comparison between Neo4j and MySQL implementations revealed significant performance differences across query categories, as shown in Table III. Neo4j demonstrated notably faster execution times, particularly for recursive queries and pattern matching operations.

C. Testing Methodology Each query was executed three times in both database systems:

- Neo4j: Using the PROFILE command to measure exact execution times
- MySQL: Using EXPLAIN ANALYZE to obtain detailed execution metrics Final measurements represent the average execution time across these three runs to ensure consistency.

Note: This research utilized synthetic or anonymized data for performance testing and did not involve human participants or personal data collection, therefore not requiring an Ethics Committee Report.

APPENDIX B: OSI MODEL REFERENCE

The OSI (Open Systems Interconnection) model divides network communication into seven layers:

7. Application Layer
    - End-user network services (HTTP, FTP, DNS)
    - User interface and access to network services
6. Presentation Layer
    - Data translation and encryption (SSL/TLS)
    - Format conversion and data compression
5. Session Layer
    - Manages communication sessions
    - Dialog control and synchronization
4. Transport Layer
    - End-to-end data delivery
    - TCP (reliable), UDP (fast)
3. Network Layer
    - Routing and logical addressing (IP)
    - Packet forwarding between networks
2. Data Link Layer
    - Physical addressing (MAC)
    - Local network data delivery
1. Physical Layer
    - Binary data transmission
    - Physical network media and signals

Key Concept: Each layer serves the layer above it, allowing modular network design and troubleshooting by isolating problems to specific layers.

APPENDIX C: METRICS QUESTIONNAIRE DETAILS TABLES

As the average number of actions was not included in the questionnaire, we did a 1 to 1 follow up to estimate how many were necessary in the Topology Map phase for each participant.

Table A: Topology Understanding Task

Id	Name	Job Title	Email	Age	Time to Conclude	Number of Actions	Completed
1	João Luís	Product Designer	joaol@sepiocyber.com	31	1m30	5	Yes
2	Thiago Brasil	Software Engineer	thiagob@sepiocyber.com	26	2m	12	Yes
3	Jonatas de Carvalho Pereira	Software Engineer	jonatasp@sepiocyber.com	33	5s	3	No
4	Alexandre Duarte	Data Analyst	alexandred@sepiocyber.com	23	3m	10	Yes
5	André Fernandes	Customer Support Engineer	andrefe@sepiocyber.com	38	3m	5	Yes
6	Pedro Marques	Web Developer	pedrom@sepiocyber.com	50	10s	5	Yes
7	Eduardo Muñoz	DevOps Engineer	eduardom@sepiocyber.com	32	10m	25	Yes
8	André Pinto	Software Engineer	andrep@sepiocyber.com	27	45s	5	Yes
9	Sergio Cordeiro	HR Specialist	sergioc@sepiocyber.com	40	2m	8	Yes

Id	Name	Job Title	Email	Age	Time to Conclude	Number of Actions	Completed
10	Luca Rodrigues de Sousa	Software Engineer	lucar@sepiocyber.com	27	58s	5	Yes

Table B: Cluster Identification

Id	Name	Job Title	Email	Age	Time to Conclude	Number of Actions	Completed
1	João Luís	Product Designer	joaol@sepiocyber.com	31	1m	5	Yes
2	Thiago Brasil	Software Engineer	thiagob@sepiocyber.com	26	1m	8	Yes
3	Jonatas de Carvalho Pereira	Software Engineer	jonatasp@sepiocyber.com	33	15s	5	Yes
4	Alexandre Duarte	Data Analyst	alexandred@sepiocyber.com	23	1m	4	Yes
5	André Fernandes	Customer Support Engineer	andrefe@sepiocyber.com	38	2m	5	Yes
6	Pedro Marques	Web Developer	pedrom@sepiocyber.com	50	2m	5	Yes
7	Eduardo Muñoz	DevOps Engineer	eduardom@sepiocyber.com	32	10m	25	No
8	André Pinto	Software Engineer	andrep@sepiocyber.com	27	40s	5	Yes

Id	Name	Job Title	Email	Age	Time to Conclude	Number of Actions	Completed
9	Sergio Cordeiro	HR Specialist	sergioc@sepiocyber.com	40	90s	5	Yes
10	Luca Rodrigues de Sousa	Software Engineer	lucar@sepiocyber.com	27	1m40s	6	Yes

Table C: Path Tracing

Id	Name	Job Title	Email	Age	Time to Conclude	Number of Acrions	Completed
1	João Luís	Product Designer	joaol@sepiocyber.com	31	1m	5	Yes
2	Thiago Brasil	Software Engineer	thiagob@sepiocyber.com	26	1m	5	Yes
3	Jonatas de Carvalho Pereira	Software Engineer	jonatasp@sepiocyber.com	33	30s	7	Yes
4	Alexandre Duarte	Data Analyst	alexandred@sepiocyber.com	23	1m43s	12	Yes
5	André Fernandes	Customer Support Engineer	andrefe@sepiocyber.com	38	3m	20	Yes
6	Pedro Marques	Web Developer	pedrom@sepiocyber.com	50	3m	5	Yes
7	Eduardo Muñoz	DevOps Engineer	eduardom@sepiocyber.com	32	5s	50	??
8	André Pinto	Software Engineer	andrep@sepiocyber.com	27	45s	5	Yes
9	Sergio Cordeiro	HR Specialist	sergioc@sepiocyber.com	40	90s	5	Yes
10	Luca Rodrigues de Sousa	Software Engineer	lucar@sepiocyber.com	27	15s	5	Yes

Table D: Finding All Relevant Network Infrastructures Affected by a Malicious Device

Id	Name	Job Title	Email	Age	Time to Conclude	Number of Actions	Completed
1	João Luís	Product Designer	joaol@sepiocyber.com	31	1m30	10	Yes
2	Thiago Brasil	Software Engineer	thiagob@sepiocyber.com	26	1m30	6	Yes
3	Jonatas de Carvalho Pereira	Software Engineer	jonatasp@sepiocyber.com	33	10s	5	Yes
4	Alexandre Duarte	Data Analyst	alexandred@sepiocyber.com	23	4m	16	Yes
5	André Fernandes	Customer Support Engineer	andrefe@sepiocyber.com	38	2m	6	Yes
6	Pedro Marques	Web Developer	pedrom@sepiocyber.com	50	10s	12	Yes
7	Eduardo Muñoz	DevOps Engineer	eduardom@sepiocyber.com	32	10m	50	??
8	André Pinto	Software Engineer	andrep@sepiocyber.com	27	20s	6	Yes
9	Sergio Cordeiro	HR Specialist	sergioc@sepiocyber.com	40	5m	7	Yes
10	Luca Rodrigues de Sousa	Software Engineer	lucar@sepiocyber.com	27	14s	3	Yes

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa