Electronics and Information Technologies, 2025, 32, 5–18
http://publications.lnu.edu.ua/collections/index.php/electronics/index

UDC: 004.9

GRAPH VISUALIZATION OF TRACEROUTE UTILITY RESULTS FOR COMPUTER NETWORK ANALYSIS

***Ivan Danych**, Kvitoslava Obelovska, Zoreslava Shpak*
Lviv Polytechnic National University
12 Stepan Bandera St., Lviv 79013, Ukraine

Danych, I.M., Obelovska, K.M., Shpak Z.Y. (2025). Graph Visualization of Traceroute Utility Results for Computer Network Analysis. *Electronics and Information Technologies*, 32, 5–18 https://doi.org/10.30970/eli.32.1

ABSTRACT

Background. Monitoring destination reachability, path discovery, and latency analysis is essential for network performance, reliability, and security. The traceroute utility is widely used for path diagnostics, but its text-based output limits the ability to perform complex analysis of network capabilities. ICMP traffic filtering, rate limiting, and load balancing cause missing hops in route displays, non-monotonic delays, and artifacts that are difficult to detect in tabular form. Existing visualization methods rely on geolocation databases and emphasize geographical aspects rather than network topology. Therefore, developing a geolocation-independent graph-based visualization tool for traceroute is relevant.

Materials and Methods. The proposed graph-based approach converts single and multiple traceroute results into a unified weighted directed graph, where vertices represent IP-defined nodes. Unknown nodes are handled using two strategies: explicit labeling or skipping with a direct connection between known nodes. The following edge-weighting criteria are introduced: occurrence frequency, used to identify critical links, and average round-trip time (RTT), used to detect latency segments. Graph visualization is performed using the Fruchterman-Reingold algorithm. The system is implemented in Java with JavaFX, Spring Core, and GraphStream, supports both archived results import and real-time tracing via a multithreaded producer-consumer model, ensuring graph integrity.

Results and Discussion. Combining multiple traceroutes into a single graph enables quick identification of shared segments and critical nodes. Systematic RTT non-monotonicity, likely caused by rate limiting, supports using absolute rather than incremental RTT values. In the experiments conducted, the multithreaded mode accelerates graph construction fourfold with seven threads, but it also increases the number of timeouts. Using five threads achieves an optimal balance between speed and data quality.

Conclusion. The proposed approach improves network analysis efficiency by constructing a geolocation-independent graph for diagnostics. The developed system reduces cognitive load compared to tabular logs analysis and speeds up detection of bottlenecks, common segments, critical nodes, and routing anomalies.

Keywords: traceroute, graph visualization, network topology, network monitoring, weighted directed graphs

INTRODUCTION

Monitoring the routes to destination hosts and analyzing their timing characteristics are key tasks for ensuring the performance, reliability, and security of network

infrastructure. The traceroute utility is among the most widely used diagnostic tools for tracking the sequence of intermediate nodes along the path from source to destination [1]. However, as the number of traceroutes and the complexity of routes increase, the text-based representation of traceroute results becomes cumbersome, complicating the identification of potential bottlenecks, shared segments, critical nodes, and network behavior anomalies.

Furthermore, the results are significantly influenced by various technical factors of modern network infrastructure, such as Internet Control Message Protocol (ICMP) traffic filtering by firewalls [2], response rate limiting [3], lower processing priority of ICMP packets compared to main application traffic [4], as well as load balancing mechanisms and multipath routing [5]. These factors lead to omissions in the sequence of displayed nodes, non-monotonic RTT (round-trip time) values, and the appearance of artifacts such as loops and diamonds in the routes, which greatly complicate the interpretation of even single traceroutes [5]. All these issues are difficult to detect in a text-based output, as it requires focused analysis to establish relationships between different traceroute results.

Existing solutions for visualizing traceroute results primarily employ geographic approaches that display routing information on world maps or three-dimensional globes. Among such tools are Open Visual Traceroute for local use and various web-based services for online tracing [6-9]. Specialized systems such as VisTracer are designed to detect routing anomalies [10]. Although geographic visualization is intuitive, it has several drawbacks: dependence on external geolocation databases with potential gaps and inaccuracies, and limited functionality in cases where geographic information is unavailable. Moreover, most of the reviewed tools do not support loading archived traceroute results or simultaneous visualization of multiple routes.

This paper proposes a graph-based visualization approach for analyzing traceroute results in network analysis tasks. Single and multiple traceroute runs are transformed into a unified weighted directed graph, where vertices correspond to unique IP addresses and edges represent sequential transitions between nodes. To handle incomplete data, two strategies for processing unknown nodes are implemented: explicit representation of such nodes with appropriate labeling, or skipping them with a direct connection between adjacent known nodes. Two schemes of edge-weighting are introduced: edge occurrence frequency across multiple traceroutes to identify critical links, and RTT-based statistics (average value) to detect segments with increased latency. Visualization is performed using the classical force-directed Fruchterman-Reingold algorithm [11]. To accelerate traceroute execution, a multithreaded approach is supported. It was found that aggressive parallelism may amplify artifacts related to rate-limiting mechanisms and ICMP processing policies.

The practical usefulness of the proposed approach is demonstrated through examples of single and multiple traceroutes: combining them into a weighted graph simplifies the detection of potential bottlenecks, shared network segments, enables the localization of critical nodes, and allows for rapid identification of anomalies. The RTT-oriented weighting highlights segments with increased latency, while the frequency-based weighting emphasizes the “backbone” of routing paths. Additionally, this approach presents pedagogical value, allowing students to explore network routes and understand routing dynamics through interactive visualization. Combined with simulation-based educational methodologies explored in our previous work [12], this creates opportunities for comprehensive computer network education platforms.

Research aim: to improve the efficiency of network analysis based on the results of the traceroute utility by constructing a weighted, geolocation-independent graph suitable for broad application in diagnostics and monitoring across various practical domains.

MATERIALS AND METHODS

This study proposes an approach for converting the textual output of the traceroute utility into weighted directed graphs to enable geolocation-independent analysis of routes within a given network topology. The approach includes a data parsing and processing algorithm, two strategies for handling unknown nodes, and two edge-weighting models – RTT-oriented and frequency-based.

Principle of operation of the traceroute utility

The traceroute utility is a fundamental and widely used diagnostic tool for analyzing network paths between a source node and a destination node. Along such a path, in addition to the source and destination, a sequence of intermediate nodes is identified, along with the timing indicators for reaching each node [13]. The operation of the utility is based on the sequential manipulation of the Time To Live (TTL) field in the IP packet header, as defined in RFC 792 [14] and RFC 1812 [15]. The utility sends packets with incrementally increasing TTL values (starting from 1). Each intermediate router decreases the TTL field by one, and when the value reaches zero (TTL = 0), it sends an ICMP “Time Exceeded” message. For each node, typically three probe measurements are performed, each reporting the round-trip time (RTT), which allows for the estimation of latency and connection stability. In Windows, the utility by default limits tracing to 30 hops, while the /d parameter disables DNS resolution to speed up the process [13].

The standard output of the traceroute utility is presented in a tabular format, where each row corresponds to a single node along the route (**Fig. 1**). The structure includes: the sequential hop number, three RTT values in milliseconds (or the "*" symbol that indicates a timeout when no response is received within the waiting period, typically 5 seconds), the IP address, and optionally the domain name of the corresponding router. This format allows for assessing the path length using the number of hops to the destination, observing the list of networks through which the packet traversed, and analyzing their sequence. A more detailed examination enables identifying segments with increased latency and common path sections when comparing multiple traceroutes.

```
C:\Users\dynyc\OneDrive\Desktop>tracert -d lpnu.ua

Tracing route to lpnu.ua [178.212.110.31]
over a maximum of 30 hops:

  1     3 ms     2 ms     3 ms  192.168.0.1
  2    14 ms     6 ms     6 ms  100.65.0.1
  3     5 ms     4 ms     5 ms  193.93.218.124
  4     *        *        *     Request timed out.
  5    11 ms    13 ms    11 ms  184.104.202.190
  6    14 ms    11 ms    21 ms  178.212.110.31

Trace complete.
```

Fig. 1. Example of the standard output of the Tracert utility in the Windows operating system.

The accuracy and completeness of traceroute results are significantly affected by the technical characteristics of modern network infrastructure, which complicate data interpretation. The main influencing factors include:

- ICMP traffic filtering. Many routers and firewalls block or ignore ICMP packets for security reasons, resulting in “* * *” entries in the output instead of IP addresses and RTT values [2]. This can conceal actual intermediate devices along the route.

- Response rate limiting. Modern network equipment may employ mechanisms that limit the number of ICMP replies, potentially leading to false timeouts or non-monotonic RTT values even under normal router operation [3].
- Traffic prioritization. ICMP packets may be processed with a lower priority compared to primary application traffic (TCP/UDP), which can distort measured RTT values and may not accurately reflect the real latency experienced by user applications [4].
- Load balancing. In complex networks, traffic can be distributed across multiple paths, resulting in topological artifacts such as loops, diamonds, or parallel branches in traceroute results [5].

Conversion of traceroute textual results into a graph structure

The purpose of this transformation is to convert the sequential textual output of the traceroute utility into a directed graph $G = (V, E)$, where the vertices (V) represent unique IP addresses of network devices, and the edges (E) correspond to transitions between consecutive nodes along the route.

The process of converting a single traceroute $T = \{node_1, node_2, \dots, node_n\}$ into a graph structure is carried out according to the following algorithm:

1. Source initialization. The initial node is set as prev = "local", representing a virtual vertex corresponding to the traceroute source (the user's local machine).
2. Data preprocessing. The parser processes the input text, discarding service lines such as "Tracing route to…", "over a maximum of 30 hops", and "Trace complete". For each hop line, the following parameters are extracted: hop number, three RTT samples, and the IP address.
3. Node processing.
   a. RTT parsing: three RTT values are extracted; if at least one measurement is available, the average RTT for the node is computed.
   b. Node identification: the current node identifier is determined as node = IP (if known) or node = Unknown (if not known).
   c. Graph structure update: node is added to the vertex set V (if not already present); a directed edge (prev → node) is created and added to the edge set E.
   d. State update: prev = node is set for the next iteration.
4. Completion. The target host is recorded as the final vertex of the route.

A particular challenge is posed by nodes without ICMP responses (marked as "* * *"). The absence of an ICMP reply should not be automatically interpreted as a physical break in the path, since network devices may intentionally ignore ICMP packets. Two alternative processing strategies for such cases, selectable through configuration, are implemented.

The first approach uses explicit representation of unknown nodes. Each unknown hop is treated as a separate vertex with a specific label and color – yellow and marked as "Unknown" (**Fig. 2a**). This approach preserves route continuity and explicitly visualizes the uncertainty of a node's identity. However, it also increases the graph density, especially considering that during multiple traceroutes, several unknown vertices may physically correspond to the same node.

The second approach skips unknown nodes. An unknown node is skipped, and adjacent known nodes are connected directly (**Fig. 2b**). The presence of an unknown segment is recorded in the edge metadata. This approach improves the readability of the graph by making it more compact. At the same time, the actual route length becomes compressed, introducing a risk of creating physically impossible direct connections.

The developed system enables the transformation of a set of traceroutes into a single unified graph. In this case, the algorithm is similar to the one described above but is applied sequentially to multiple individual traceroutes. The key feature of the merging process lies in the aggregation of corresponding metrics: if the processed traceroute contains an edge

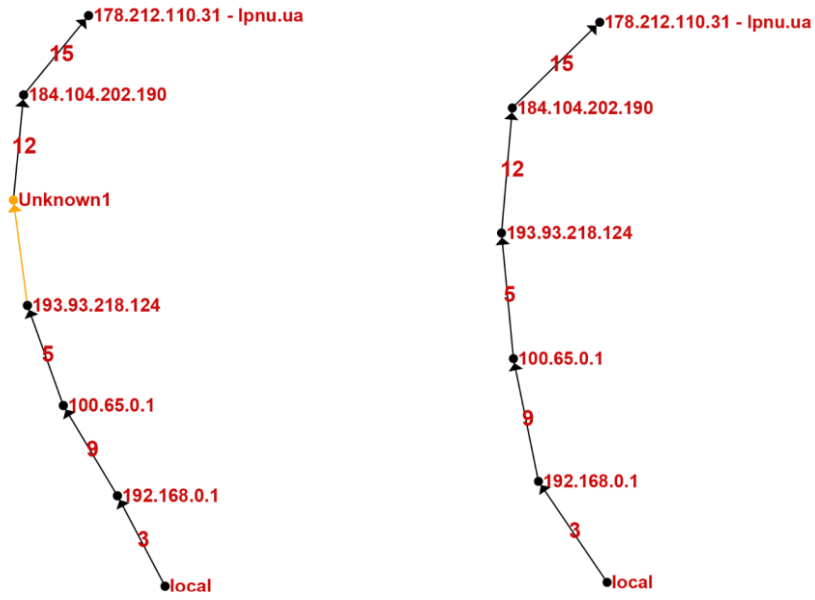

Fig. 2. Example of graph construction for the traceroute shown in Fig. 1: a) explicit representation of an unknown node; b) omission of the unknown node.

that already exists in the graph, the new RTT measurements are aggregated with the existing ones (updating the average value across all traceroutes), thereby providing more accurate data for edge weighting.

To transform the topological graph into an analytical tool for network monitoring, an edge-weighting system was developed that enables quantitative evaluation of various network aspects. The weighting models provide contextual insight for identifying critical nodes, bottlenecks, and anomalous routing patterns that cannot be detected through simple connectivity analysis of the graph. Two basic weighting models are applied in this study (**Fig. 3**).

RTT-based model. For each edge, the weight is determined based on the set of all RTT measurements. The primary estimate is the average value across corresponding samples, which highlights route segments with increased latency (**Fig. 3a**). A fundamentally important feature is the use of the absolute RTT value to the target node

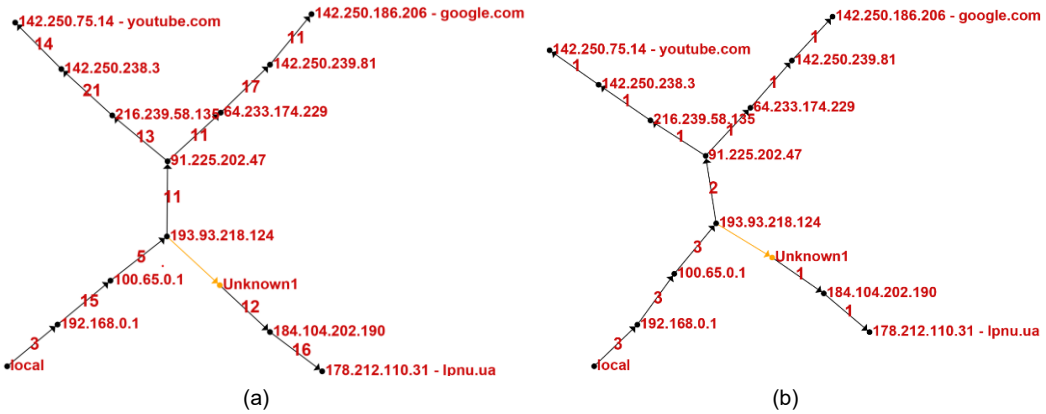

Fig. 3. Example of traceroute visualization to three resources: lpnu.ua, google.com, youtube.com. a) RTT-based model; b) Frequency-based model.

instead of the incremental delay between consecutive nodes. This decision is motivated by practical observations: due to rate-limiting mechanisms and temporary router overloads, it is common for a node at position N to exhibit a higher delay than the subsequent node at position *N* + 1. Using incremental values in such cases would result in negative weights or distortion of the actual network quality representation.

*Frequency-based mode*l. For each edge, the weight equals the number of traceroutes in which that edge appears (**Fig. 3b**). This weight is interpreted as an indicator of the segment's commonality and helps highlight the key connections (the "backbone") within the set of routes. For example, in the case of three independent traceroutes, edges shared by all routes would have a weight of 3, indicating their role as a common provider segment.

In the future, the proposed weighting approach for nodes and edges is expected to be extended. Alternative metrics – such as response stability, delay variance, and packet loss frequency – could provide a more comprehensive real-time network analysis.

Software implementation

For the practical implementation of the described algorithms for transforming and visualizing traceroute results, a specialized module was developed to extend the functionality of an existing graph visualization system created in previous research [16]. The base system provides a platform for visual analysis of graph structures with support for various visualization algorithms and interactive exploration tools, which allowed us to focus on the specific aspects of processing textual traceroute data.

The developed module supports two main operating modes:

- Text data parsing mode – designed for analyzing previously collected traceroute results. The module allows importing text files and applies the developed parser to structurally extract data on nodes, RTT values, and IP addresses, after which it executes the selected strategy for handling unknown nodes. The user can dynamically choose the weighting model (RTT-oriented or frequency-based) and adjust visual display parameters.
- The online tracing mode provides interactive execution of the traceroute utility with graph construction in near real time. The user specifies a list of target resources through the graphical interface, after which the system automatically initiates multithreaded tracerouting and gradually integrates the results into the visualization as they are received.

The online tracing mode is implemented using a multithreaded approach, where the traceroute, utility is executed in parallel at the operating system level. The system simultaneously launches multiple tracert processes for different target resources.

To achieve this, the producer–consumer pattern with a BlockingQueue is used. The results of each traceroute (tracing threads) are placed into a dedicated queue, which is processed by a single consumer thread. Formally, the software implementation can be described as follows:

$$Producers: P_1, P_2, \dots, P_n \xrightarrow{Queue} Consumer\ (Sequential) \quad (1)$$

where P_n represents the *n*-th tracing thread, and the arrow denotes the asynchronous data transfer through the queue followed by sequential processing.

This design is motivated by the need to ensure the integrity of the graph structure: since the graph is a shared resource modified when adding new vertices and edges, parallel update operations may lead to race conditions, data loss, or structural inconsistencies. Sequential queue processing guarantees the atomicity of operations and the consistency of the resulting graph.

The development of the online mode was driven by two main factors. First, there is a lack of large publicly available datasets of traceroute results suitable for research and validation of the proposed solution. Second, advancing the concept of real-time network analysis requires periodic data updates, enabling the tool to function as an interactive and efficient monitoring system.

The number of parallel threads n can be configured in the system's configuration file. During practical testing, it was found that beyond a certain threshold of threads, traceroute operations begin to interfere with each other, leading to an increase in nodes with timeouts. This behavior is likely related to rate-limiting mechanisms at the network equipment level and ICMP traffic handling policies.

The software implementation is built on the Java platform, using JavaFX for the graphical user interface, Spring Core for configuration and service management, and GraphStream for working with graph structures. This combination of libraries provides flexibility, modularity, and extensibility of the system.

RESULTS AND DISCUSSION

This section presents the experimental results obtained for the developed system. The study was organized in two main stages: first, the analysis of archived traceroutes to demonstrate the system's basic functionality and its ability to identify shared segments and critical nodes; and second, a series of experiments evaluating the performance of the multithreaded mode and identifying limitations related to rate-limiting mechanisms.

Analysis of multiple traceroutes based on the results of the traceroute utility

To demonstrate the effectiveness of the graph-based approach to network topology analysis, a comprehensive evaluation was conducted using multiple traceroutes to six different resources: google.com, youtube.com, student.lpnu.ua, lpnu.ua, chatgpt.com, and deepl.com, based on previously collected textual traceroute results. The results were integrated into a single weighted directed graph using an RTT-oriented weighting model, where numerical labels on the edges represent the average RTT values in milliseconds, and unknown nodes are highlighted in yellow (**Fig. 4**).

A detailed analysis of individual graph branches reveals an important characteristic of RTT measurements – a systematic violation of monotonic delay growth along the route. On the branch to google.com, the delay to node 142.250.239.81 is 17 ms, while the next node, 142.250.186.206, shows an RTT of only 11 ms. A similar situation was observed for the route to youtube.com, where the intermediate node 142.250.238.3 has an RTT of 21 ms, and the final node 142.250.75.14 shows only 14 ms. Such anomalies may result from rate-limiting mechanisms on intermediate routers [3] or temporary overloads on certain nodes. These observations confirm the correctness of using absolute RTT values to the target node instead of increments between consecutive hops, since the latter could produce negative weights and distort the actual network latency pattern.

Combining six traceroutes into a single graph demonstrates the key advantage of the visual approach: the shared provider segment local → 192.168.0.1 → 100.65.0.1 → 193.93.218.124 can be visually identified instantly, whereas in a text format this would require careful comparison of six separate result tables, each containing five to eight rows. This segment represents the backbone of the local network infrastructure. It is also worth noting that in this format, it is easy to interpret latency times to different resources – in our case, all of them exhibit a similarly low delay of 11-16 ms. Overall, this approach greatly facilitates the detection of potential bottlenecks.

Graph-based visualization effectively reveals more complex network relationships. The routes to student.lpnu.ua and lpnu.ua branch after the node 193.93.218.124. Interestingly, both routes contain unknown nodes in similar positions, and the node

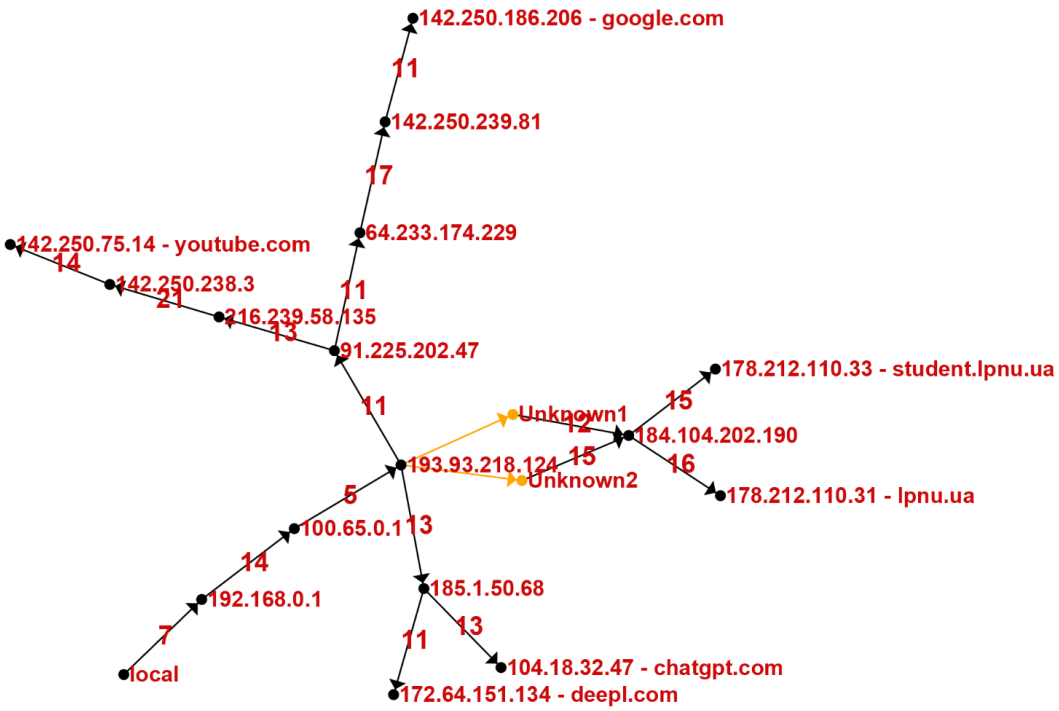

Fig. 4. Visualization of multiple traceroutes to six web resources: youtube.com, google.com, student.lpnu.ua, lpnu.ua, chatgpt.com, and deepl.com. RTT-oriented weighting model is used.

following these undefined ones (184.104.202.190) is shared. This suggests that Unknown1 and Unknown2 may physically correspond to the same router or a group of routers that block ICMP responses for security reasons.

Additionally, the graph demonstrates the logical cohesion of routes to google.com and youtube.com at node 91.225.202.47, which reflects their belonging to the Google infrastructure.

Analysis of the online tracing and visualization mode

The second experiment consisted of a series of studies aimed at evaluating the efficiency of the multithreaded online mode by performing simultaneous traceroutes to ten resources – google.com, lpnu.ua, dou.ua, youtube.com, rozetka.ua, chatgpt.com, wikipedia.org, deepl.com, facebook.com, and instagram.com – using different numbers of parallel threads. The main evaluation criteria were the graph construction speed (including the total traceroute execution time) and the quality of the obtained data, measured by the number of timeouts (**Table 1**).

Table 1. **Dependence of multithreaded mode efficiency on the number of parallel threads**

Number of threads	Graph construction time, s	Number of timeouts (unknown nodes)
1	65	2
3	26	3
5	16	4
7	15	5

The results demonstrate a clear trade-off between data collection speed and quality. As the number of threads increases from 1 to 7, the graph construction time decreases by more than fourfold (from 65 to 15 seconds), confirming the effectiveness of the parallel approach for real-time network monitoring.

The experiment with seven threads is particularly important: among five unknown nodes, four anomalous edges appear, which degrades the topological assessment quality. A detailed analysis of the resulting graph reveals a characteristic anomaly in the local network exit segment (**Fig. 5**).

In **Fig. 5**, a cluster of unknown nodes (Unknown1, Unknown2, Unknown3, Unknown4) can be clearly observed in close proximity to the local router 192.168.0.1. According to the principles of NAT routing, all devices within the local network should forward traffic through a single gateway 192.168.0.1, so the appearance of multiple "unknown" paths in this segment is an artifact.

An analysis of the textual traceroute logs reveals the exact mechanism behind these anomalies. In four out of ten traceroutes, timeouts were recorded at the first hop. The most probable cause is the activation of rate-limiting mechanisms on the local router 192.168.0.1 in response to the simultaneous arrival of seven intensive ICMP request streams.

A key advantage of the graph-based approach is its ability to visually identify anomalies quickly. In text format, determining that several unknown nodes physically correspond to the same router is extremely difficult and requires careful comparison of timeout positions across dozens of individual traceroutes. Graph visualization makes such patterns immediately apparent, as unknown vertices naturally cluster around the same transitions.

The experiments performed confirmed the high effectiveness of the proposed graph visualization approach to traceroute results for geolocation-independent network topology analysis. The system enables efficient detection of shared segments and critical nodes. The multithreaded online mode significantly expands the system's capabilities for real-time monitoring, providing a 3-4 times acceleration in data collection when properly configured.

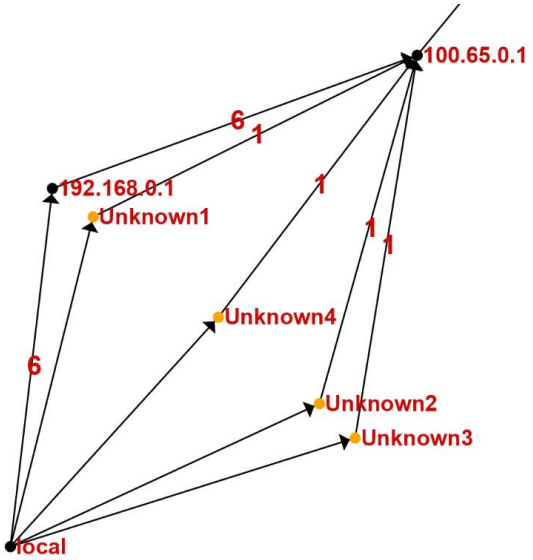

Fig. 5. Local network exit segment when using seven parallel tracing threads. Unknown nodes resulting from timeouts at the first hop are marked in yellow and labeled Unknown1–4.

Comparison with existing tools

To fully evaluate the capabilities of the proposed graph-based visualization approach, it is essential to compare it with existing traceroute analysis tools, both geographic and graph-oriented.

Geographic visualization systems such as Open Visual Traceroute [6] typically demonstrate higher rendering performance due to their reliance on external geolocation databases. Each IP address is directly mapped to predefined coordinates (latitude and longitude). As a result, the placement of nodes is determined by geographic metadata, and rendering complexity grows linearly with the number of nodes – O(n).

In contrast, the proposed method operates without geolocation data and constructs a layout solely from the textual output of the traceroute utility. It relies on a force-directed model – the Fruchterman-Reingold algorithm [11] – whose computational complexity is $O(n^2)$. This leads to lower rendering performance for graphs with a high number of nodes. However, geolocation independence provides an important benefit: the system remains fully functional even when IP-to-location mappings are unavailable or inaccurate. This makes the approach more versatile in research, diagnostic, and educational contexts where the primary interest lies in understanding routing behavior rather than geographic placement.

A direct comparison with VisTracer [10] is more challenging due to the absence of a publicly available implementation or reproducible benchmarks. The paper describing VisTracer indicates that its visualization pipeline partially relies on force-directed layout mechanisms but does not detail the algorithms or their computational cost. Conceptually, VisTracer is designed as a specialized analytical tool focused on detecting routing anomalies – particularly those associated with BGP hijacking, load balancing irregularities, and other forms of path instability. In contrast, the system presented in this work aims to provide a general-purpose graph-based representation of traceroute results.

Thus, the proposed approach complements VisTracer instead of competing with it, as they target distinct analytical objectives. While VisTracer investigates routing during data transmission, our approach focuses on analyzing the output of the traceroute utility. Consequently, our system is designed not to interfere with the traceroute acquisition process, but rather to effectively visualize its data.

Limitations of the proposed approach

The proposed method inherits several fundamental limitations from the traceroute utility itself. The accuracy of RTT measurements is affected by router-specific processing policies, ICMP rate-limiting mechanisms, and transient network conditions, such as increased evening load or temporary congestion on specific segments. These factors may distort the measured delays and reduce the reliability of RTT-based edge weighting. Additionally, ICMP packet filtering performed by firewalls or intermediate routers introduces missing hops and increases the number of unknown nodes. Both implemented strategies for handling such cases – explicit representation or skipping – may lead to incomplete structural representation or to the formation of edges that do not correspond to the actual physical routing path. Consequently, the resulting visualization should be interpreted with awareness of these limitations, particularly in networks with strict ICMP policies or highly dynamic routing behavior.

In addition, several limitations are associated with the visual representation of the resulting graphs. Large graphs may become visually overloaded, with overlapping node labels, edge length indicators, and occasional excessive edge crossings, all of which can hinder interpretability. While basic scaling and zooming mitigate these issues to some extent, they do not fully eliminate them. Further research could therefore focus on improving the layout and rendering algorithms to enhance readability and provide a more effective user experience.

CONCLUSION

Addressing the key tasks of monitoring routes to destination hosts and analyzing their timing characteristics, this work develops and experimentally validates a geolocation-independent approach to network analysis based on transforming the textual output of the traceroute utility into a unified weighted directed graph. A method for building the graph $G = (V, E)$ is proposed, enabling efficient aggregation of multiple traceroutes and accumulation of statistics for subsequent weighting. Two configurable policies for handling unknown nodes are implemented – explicit representation and skipping – along with two weighting models: an RTT-oriented model (for identifying segments with increased latency) and a frequency-based model (for highlighting shared or critical connections). A software module was developed with two operating modes: import of archived traceroute results and online data collection with multithreaded traceroute execution and incremental graph construction.

The obtained results demonstrate that combining multiple traceroutes into a single weighted graph reduces cognitive load compared to analyzing tabular logs, simplifies the identification of shared segments, bottlenecks, and atypical patterns (such as loops or diamonds). It has been shown that absolute RTT values, unlike per-hop increments, yield more stable edge weights in the presence of non-monotonic behavior caused by ICMP rate-limiting. In the multithreaded mode, a practical trade-off between speed and quality was established: using 3–5 threads provides significant acceleration of graph construction with only a moderate increase in timeouts, whereas more aggressive parallelism amplifies artifacts and reduces interpretability.

Unlike geographic systems that rely on external IP-to-location databases, the proposed method focuses on directly representing the routing paths derived from traceroute data and remains fully operational even when geolocation information is unavailable or unreliable.

Furthermore, the study shows that traceroute-based visualization is inherently constrained by the accuracy of RTT measurements and the presence of ICMP filtering, which may introduce missing hops and structural uncertainty in the resulting graph. As a result, the resulting graph structure should be interpreted with consideration of these factors, especially in networks with strict ICMP policies or highly dynamic routing behavior.

The scientific novelty of this work lies in combining the graph-based aggregation of multiple traceroutes with two alternative strategies for processing unknown nodes and implementing a graph weighting mechanism that provides an informative view of network behavior both in historical traceroute analysis and in real-time monitoring.

The practical value of the study is the creation of a network diagnostic and administration tool that enables quick detection of shared segments, critical nodes, and potential bottlenecks, thereby simplifying monitoring and troubleshooting. The method could also serve educational purposes, allowing students to interactively explore network routes and understand topology behavior through visual analysis.

Future work will focus on expanding the set of weighting metrics (e.g., response stability, packet loss frequency, etc.) and implementing automatic anomaly detection and highlighting mechanisms. Furthermore, a promising direction is the exploration of a hybrid RTT weighting scheme that combines both absolute and incremental delay values. Absolute RTT provides a stable representation of end-to-end latency, particularly in scenarios with non-monotonic behavior caused by rate limiting or router overload, while incremental RTT may reveal localized delays between specific hops. Integrating both perspectives could offer a more comprehensive view of latency distribution across the route and enhance the analytical value of the visualization.

ACKNOWLEDGMENTS AND FUNDING SOURCES

The authors received no financial support for the research, authorship, and/or publication of this article.

COMPLIANCE WITH ETHICAL STANDARDS

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

AUTHOR CONTRIBUTIONS

Conceptualization, [K.O., I.D.]; methodology, [I.D]; validation, [K.O., Z.S.]; formal analysis, [I.D.].; investigation [I.D.]; resources, [I.D]; data curation,[I.D.]; writing – original draft preparation, [I.D.]; writing – review and editing, [K.O., Z.S.]; visualization, [I.D.] supervision, [K.O.]; project administration, [Z.S.].

All authors have read and agreed to the published version of the manuscript.

REFERENCES

[1] Caiazza, C., Luconi, V., & Vecchio, A. (2019). TCP-based traceroute: An evaluation of different probing methods. Internet Technology Letters, 2(6), e134. https://doi.org/10.1002/itl2.134

[2] Luckie, M., Hyun, Y., & Huffaker, B. (2008). *Traceroute probe method and forward IP path inference.* In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement (IMC '08)* (pp. 311–317). Association for Computing Machinery. https://doi.org/10.1145/1452520.1452553

[3] Guo, H., & Heidemann, J. (2018, March). *Detecting ICMP rate limiting in the Internet.* In R. Beverly, G. Smaragdakis, & A. Feldmann (Eds.), *Proceedings of the 19th International Conference on Passive and Active Network Measurement (PAM 2018, Berlin, Germany)* (LNCS 10771, pp. 3-17). Springer. https://doi.org/10.1007/978-3-319-76481-8_1

[4] Li, W., Zhang, D., Xie, G., & Yang, J. (2005). TCP and ICMP in network measurement: An experimental evaluation. *Parallel and Distributed Processing and Applications, Lecture Notes in Computer Science*, 3758(3), 870–881. https://doi.org/10.1007/11576235_87

[5] Augustin, B., Cuvellier, X., Orgogozo, B., Viger, F., Friedman, T., Latapy, M., Magnien, C., & Teixeira, R. (2006). Avoiding traceroute anomalies with Paris traceroute. *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement (IMC'06)*, 153–158. https://doi.org/10.1145/1177080.1177100

[6] VisualTraceroute. (n.d.). Visual traceroute. Retrieved from https://visualtraceroute.net/

[7] Sundin, S. (n.d.). Traceroute mapper. GitHub Pages. Retrieved from https://stefansundin.github.io/traceroute-mapper/

[8] Traceroute Online. (n.d.). *Traceroute online*. Retrieved from https://traceroute-online.com

[9] GeoTraceroute. (n.d.). GeoTraceroute. Retrieved from https://geotraceroute.com/

[10] Fischer, F., Fuchs, J., Vervier, P.-A., Mansmann, F., & Thonnard, O. (2012). VisTracer: A visual analytics tool to investigate routing anomalies in traceroutes. *Proceedings of the Ninth International Symposium on Visualization for Cyber Security (VizSec '12)*, 80–87. https://doi.org/10.1145/2379690.2379701

[11] Cheong, S.-H., & Si, Y.-W. (2022). *Force-directed algorithms for schematic drawings and placement: A survey. arXiv preprint* arXiv:2204.01006. https://arxiv.org/abs/2204.01006

[12] Obelovska, K., Danych, I. (2022). Adoption of the OMNET++ Simulator for the Computer Networks Learning: A Case Study in CSMA Schemes. In: Hu, Z., Zhang, Q., Petoukhov, S., He, M. (eds) Advances in Artificial Systems for Logistics Engineering. ICAILE 2022. Lecture Notes on Data Engineering and Communications Technologies, vol 135. Springer, Cham. https://doi.org/10.1007/978-3-031-04809-8_21
[13] Microsoft. (n.d.). tracert. In *Windows Server: Windows commands*. Microsoft Learn. https://learn.microsoft.com/uk-ua/windows-server/administration/windows-commands/tracert
[14] Postel, J. (1981). Internet control message protocol. RFC 792. Internet Engineering Task Force (IETF). https://datatracker.ietf.org/doc/html/rfc792
[15] Baker, F. (1995). Requirements for IP version 4 routers. RFC 1812. Internet Engineering Task Force (IETF). https://datatracker.ietf.org/doc/html/rfc1812
[16] Danych, I. M., & Shpak, Z. Y. (2025). Methods and tools for graph visualization in dynamic systems: An experimental study. Ukrainian Journal of Information Technology, 7(1), 131–139. https://doi.org/10.23939/ujit2025.01.131

ВІЗУАЛІЗАЦІЯ НА ГРАФАХ РЕЗУЛЬТАТІВ УТИЛІТИ TRACEROUTE ДЛЯ АНАЛІЗУ КОМП'ЮТЕРНИХ МЕРЕЖ

***Іван Данич**, Квітослава Обельовська, Зореслава Шпак*
Національний університет «Львівська політехніка»,
вул. Степана Бандери 12, м. Львів, 79013 Україна

АНОТАЦІЯ

Вступ. Моніторинг досяжності вузлів, виявлення маршрутів і аналіз затримок є важливими для забезпечення продуктивності, надійності та безпеки мережі. Утиліта traceroute широко застосовується для діагностики шляхів, однак її текстовий формат ускладнює аналіз складних топологій. Фільтрація ICMP-трафіку, обмеження швидкості відповідей і балансування навантаження призводять до пропусків хопів, немонотонних затримок і артефактів, що важко виявити у табличному вигляді. Існуючі методи візуалізації базуються на геолокаційних даних і зосереджуються на географічних аспектах, а не на топології мережі. Тому актуальним є створення геолокаційно незалежного інструменту візуалізації результатів утиліти traceroute у вигляді графа.

Матеріали та методи. Запропоновано підхід, що перетворює одиничні та множинні результати утиліти traceroute у зважений орієнтований граф, де вершини є IP-вузлами. Невідомі вузли обробляються двома способами: явним маркуванням або пропуском із прямим з'єднанням відомих вузлів. Ребра зважуються за частотою появи (для виявлення критичних зв'язків) і середнім RTT (для визначення ділянок із затримками). Візуалізацію виконано за алгоритмом Fruchterman-Reingold. Система реалізована на Java з використанням JavaFX, Spring Core і GraphStream, підтримує імпорт архівних даних і трасування у режимі реального часу через багатопотокову модель producer-consumer, що забезпечує цілісність графа.

Результати. Об'єднання трасувань до кількох цілей в один граф дає змогу швидко знайти спільні сегменти та критичні вузли. Виявлено немонотонність RTT, ймовірно спричинену rate limiting, що обґрунтовує використання абсолютних, а не приростових значень RTT. У проведених експериментах багатопотоковий режим прискорив побудову графа у чотири рази при семи потоках, але збільшив кількість таймаутів. Використання п'яти потоків забезпечило оптимальний баланс швидкості та якості.

Висновки. Запропонований підхід підвищує ефективність аналізу мережі, формуючи геолокаційно незалежний граф для діагностики. Розроблена система зменшує когнітивне навантаження порівняно з аналізом табличних даних і пришвидшує виявлення потенційних «вузьких місць», спільних сегментів, критичних вузлів та аномалій маршрутизації.

Ключові слова: traceroute, візуалізація на графах, мережева топологія, мережевий моніторинг, зважені орієнтовані графи.

Received / Одержано	Revised / Доопрацьовано	Accepted / Прийнято	Published / Опубліковано
06 November, 2025	01 December, 2025	02 December, 2025	25 December, 2025