Walks in Cyberspace: Improving Web Browsing and Network Activity Analysis with 3D Live Graph Rendering

Lionel Tailhardat
Orange
France
lionel.tailhardat@orange.com

Raphaël Troncy
EURECOM
France
raphael.troncy@eurecom.fr

Yoan Chabot
Orange
France
yoan.chabot@orange.com

ABSTRACT

Web navigation generates traces that are useful for Web cartography, user behavior analysis (UEBA) and resource allocation planning. However, this data still needs to be interpreted, sometimes enriched and appropriately visualized to reach its full potential. In this paper, we propose to explore the strengths and weaknesses of standard data collection methods such as mining Web browser history and network traffic dumps. We developed the DynaGraph framework that combines classical traces dumping tools with a Web application for live 3D rendering of graph data. We show that mining navigation history provides useful insights but fails to provide real-time analytics and is not easy to deploy. Conversely, mining network traffic dumps appears easy to set up but rapidly fails once the data traffic is encrypted. We show that 3D rendering allows to highlight navigation patterns for a given data sampling rate.

CCS CONCEPTS

• **Networks → Network performance evaluation**; • **Human-centered computing → Visualization**; • **Information systems** → *Decision support systems.*

KEYWORDS

Web Browsing Traces, User and Entity Behavior Analytics (UEBA), Network Resource Allocation Planning, Network Visualization, Multilayered Graphs

ACM Reference Format:
Lionel Tailhardat, Raphaël Troncy, and Yoan Chabot. 2022. Walks in Cyberspace: Improving Web Browsing and Network Activity Analysis with 3D Live Graph Rendering. In *Companion Proceedings of the Web Conference 2022 (WWW '22 Companion), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3487553.3524230

1 INTRODUCTION

Comparing the Web to physical libraries makes location, distance and time considerations vanish for Web browser users when accessing content on the Web: locations become URLs even if hosting servers are spread throughout the world; distances become response-time and latency measures; and time is stored as access records in the browsing history of Web browsers. When the idea of "knowledge memorization and organization while reading and with the help of a technical system" was introduced with the MemEx machine [5], dismissing *path* and *intent* from Web navigation miss the point for deeper contextualization (live or retrospective) of the data being accessed. Analyzing the browsing history and data from intermediate systems (e.g. logs, packet inspection) provides contextualization information (e.g. source/destination hosts, network protocol, content size). However, taken as it is, we observe that this data and its inherent representation does not directly or sufficiently reveal details about the virtual travel that occurs. Firstly, analyzing network traffic traces gives insights about data exchange domains on a host to host basis. However, providing insights about *"why"* a given packet is sent between some hosts, and *"how"* this is included into some higher level activity (e.g. Web navigation for online shopping, video on demand streaming, peer-to-peer beaconing of worms) is not available per se without additional knowledge (i.e. combination of time, trigger and accessed resource indicators). Secondly, providing network connectivity maps from traffic dumps is an appreciable option for high level activity analysis. However, static and planar representations hinder finer grain analysis by compressing dynamics and complexity of network traffic.

This has practical impacts on day-to-day experience of the Web with respect to its underlying infrastructure: 1) Web citizens willing to understand their Web browsing traces may find Web browser's history side panel impractical for (live) evaluation of their (complex) browsing path, 2) cyber security analysts working on malicious activity assessment (e.g. forensics, live application usage conformance checking) may get lost in long lists of fast paced network transactions without system topology contextualization, 3) network/platform designers working on optimal data placement projects may miss visual guess about how data is accessed in their system.

We hypothesize that a tool enabling semantically-enriched and live graphical analysis of a Web browsing session could: 1) enable live map-making of accessed resources and assets with timestamped path taken by users or applications, 2) ease identifying access patterns by disentangling map dimensions with 3D representation, 3) enable identifying multi-user/application access patterns by centralizing multiple trace sources in a single map. From the application perspective, such tool would improve situations in need of user behavior analysis (UEBA) for resource allocation planning or access conformance checking. We propose to explore strengths and weaknesses of standard data collection methods such as mining Web browser history and network traffic dumps. We develop the DynaGraph framework, a system combining classical traces dumping tools (i.e. the `tshark` tool and Firefox's Network Monitor component[1]) and an adhoc Web application that provides live 3D rendering of graph data derived from traces.

The remainder of this paper is organized as follows. Section 2 presents some related work. Section 3 presents our approach while Section 4 details the DynaGraph framework. Section 5 describes our experiments and evaluations. Finally, Section 6 concludes the paper and discusses some future work. We release the DynaGraph code at https://github.com/Orange-OpenSource/dynagraph.

2 RELATED WORK

Web browsing and network activity analysis topics are at the cross-border of various research and technical domains, such as: Web usage mining, malicious activity detection, network performance monitoring, process modeling and graph-based analytics. In this section, we review some of this related work from the aspect of Web/IT resources cartography.

The Web cartography approach combines Web crawlers with topic modeling [12] and graph clustering/traversal [4, 10, 14, 16] techniques. It aims at understanding the shape and dynamics of the Web from the lenses of both humanities and social science and network science. Its effectiveness is determined by 1) the number of layers (dimensions) to handle when looking for aggregates in multilayered graphs, 2) the data spatialization and visual indicators chosen by the analyst.

Closer to Web navigation, Web Usage Mining (WUM) *"aims at discovering interesting patterns by exploiting usage data stored during the interactions of users with the Web site"* [6]. WUM can be carried out either on network or user side. Its effectiveness is firstly determined by the ability to access and inspect navigation data, such as Web browser events (e.g. Web Navigation Window add-on[2]), Web navigation session data (e.g. HAR files[3]), Internet Service Provider (ISP) access logs or encrypted network traffic (e.g. TLS protocol). Other effectiveness factors are 1) the ability to identify and distinguish between the users who access Web resources [17], and 2) algorithmic complexity/limitations inherent to the pattern discovery technique (e.g. association rule mining, sequential pattern discovery, clustering, classification) [6].

Detailed knowledge of active IT resources (e.g. routers, servers, virtual machines) is a prerequisite towards high quality of service and operational efficiency for network operators. Anomaly Detection (AD) and Root Cause Analysis (RCA) capabilities of Network Monitoring Systems (NMSs) and Security Information and Event Management systems (SIEMs) directly rely on such knowledge (e.g. host IP address, running software, neighborhood). It is built manually or with the help of automated process such as network discovery (e.g. LLDP protocol, Nmap[4] security scanner), flow monitoring/discovery (e.g. IPFIX protocol, Cisco Tetration platform[5]), or information sharing between stakeholders (e.g. NDL [18], RDF-based infrastructure description [7, 9]).

3 DYNAGRAPH APPROACH

Graphs are convenient tools for studying complex phenomenon from the system point of view: they offer both intuitive representation of relational objects, and mathematical tools (i.e. graph theory) to measure structural characteristics of the system. We observe that ICT systems (IT networks) are typically represented as graphs where vertices are physical/logical resources and edges are physical/logical links between resources. The canonical form of such graph is $(n_A) \underset{\text{connected to}}{\longleftrightarrow} (n_B)$ where $n_{(\cdot)}$ is some IT resource (e.g. server, router, firewall, access point). Similarly, an agent (human or scripted) querying some Web resource (e.g. an HTTP GET for a Web page) is a two-sided relational model where an URL both designate 1) the Web resource object itself, and 2) its location in term of an IP address through name resolution. Canonical forms for such graph are $(agent) \underset{\text{asks for}}{\longrightarrow} (url)$ and $(url) \underset{\text{hosted by}}{\longrightarrow} (n_A)$. Finally, from the agent perspective, triggering a query is done at a specific moment in time with a reference to a specific URL. In addition to this, the same agent may carry out a sequence of queries based on his agenda, thus building an ordered set of actions w.r.t. time. Canonical forms for such graph are $(agent_t) \underset{\text{asks for}}{\longrightarrow} (url)$ with $t \in \{1 \dots T\}$ a time position in the experiment of duration T, and $(agent_{ti}) \underset{\text{followed by}}{\longrightarrow} (agent_{tj})$ where $ti \leq tj$.

We hypothesize that a multilayered graph (i.e. a graph with heterogeneous vertices or edges) with the following minimal set of layers is sufficient for depicting a Web navigation session: 1) hosting IT network resources (e.g. IP addresses), 2) Web resources (e.g. URLs), 3) timestamped agent actions. As we have no prior knowledge of the course of action (i.e. the Web navigation scenario), we propose to capture and process the Web navigation session as a data stream, and use a graph expansion principle for building up the graph simultaneously on the three above mentioned layers through adequate parsing of the data stream.

A broad variety of Web navigation context is possible. We summarize usage context of our approach in Table 1. We will also refer, in subsequent sections, to the typical situation where Web navigation 1) is carried on by a user through a Web browser software hosted on a personal computer or equivalent, 2) follows a general search loop scheme. This typical situation entails the following remarks. Firstly, one cannot presuppose any user expertise on both the setup or the usage of a path tracking system. Thus the path tracker should rely as much as possible on already existing technologies and configurations, and maximizes usability and interpretability. A side consequence of this is that far too numerous data may be collected, thus a processing/filtering capability should be taken into account in the design process. Secondly, lack of prior knowledge on the Web navigation course (i.e. general search loop scheme) do not allow to neatly distinguish search sub-sequences within the overall Web browsing session. An illustrative scenario is a student browsing through music fan blogs (i.e. a sequence of hyper-link related Web sites), and then decide to connect to his academic Web site to check for an eventual planning update (i.e. URL-based access to a Web site), and finally goes to a search engine

[1]https://developer.mozilla.org/en-US/docs/Tools/Network_Monitor
[2]https://github.com/vcharpenay/web-nav-window
[3]https://w3c.github.io/web-performance/specs/HAR/Overview.html
[4]https://nmap.org
[5]https://www.cisco.com/go/tetration

to get back on his initial music topic. Unless presence of useful tags denoting activity start/stop in the user traces, the grouping of events will solely rely on the correct choice of a time sampling period. Thirdly, the graph activity diagram may be non-planar due to redundant access to Web content. This scenario gives an example where content may be accessed twice as the user wished to get it twice (e.g. a blog page where details were skipped at first by the student). A more subtle example comes from the fact that complex Web platforms use dedicated servers for some types of contents and refer to the same content from various documents (e.g. a federated service commercial offer where users' avatars are managed on a third-party social network platform). Therefore, accessed resources are composite in the sense that an URL may inherently refer to a set of sub-contents (e.g. images, videos, scripts, etc.) and IT resources (i.e. servers). With both these behavioral and technical complexity in mind, a classical 2D graph representation may not be suitable for finding patterns through visual analysis of such non-planar graphs. A typical override for this is to work with a 3D representation. However, data/graph visualization is a complex topic per se as it combines user expectations about rendering quality with reactivity of user interfaces. Hence, we argue that the development of the path tracking solution should not focus on developing a rendering engine, but will do best at integrating a generic and active rendering engine project.

	Capture source	
Capture context	Net. traffic	Web browser
Browser windows	✓	-
Browser tabs	✓	✓
Multiple browser	✓	-
Browser in incognito mode	✓	-[1]
Browser online mode	✓	✓
Browser offline mode	-	-
Mobile Web browsing	-	-
Local process	✓	-
Browsing session replay	✓	✓
Traffic replay	✓	-

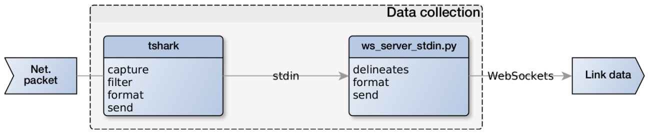
[1] Web browsing sessions cannot be saved to HAR files in Private mode.

Table 1: Scope of the DynaGraph experiments
A check mark (✓) indicates Web navigation context where the DynaGraph framework apply.

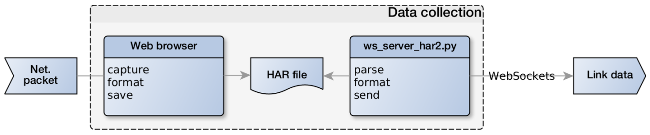
4 DYNAGRAPH FUNCTIONAL ARCHITECTURE

For simple interpretation of our framework, we make use of the *observer* design pattern [8] to form a system from two components categories: 1) *data collection and processing*, and 2) *graph visualization.* We also propose and make use of the criteria from Table 2 as general directions for developing the DynaGraph framework. We detail the DynaGraph components in the following paragraphs.

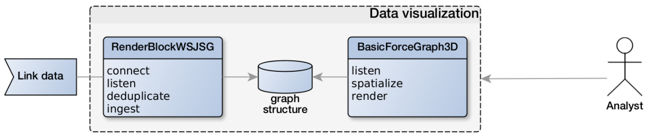
Data Collection: Live packet capture. Assuming that 1) network packets carry useful information at basic fields level, 2) network packets capture should be launched in a programmatic way, we propose a network capture and processing tool chain (Figure 1a) based on the combination of the `tshark` tool[6] and an adhoc script for data processing and forwarding. We provide a standard data forwarding process towards downstream components (e.g. data visualization) by using the WebSockets protocol[7]. This script delineates the `tshark` output stream to the data schemata of Listing 1, and then streams graph data through a WebSockets server instance with the help of the `websockets` package[8].



(a) Data Collection: Live packet capture



(b) Data Collection: Web browser network capture



(c) Data Visualization: Web app

Figure 1: The DynaGraph architecture
Overview of the Data Collection (Figures 1a & 1b) and Visualization (Figure 1c) tool chains.

Data collection: Web browser network capture. Assuming that 1) network packets capture tools may not be available at host level, 2) Web browsers have the capability to analyze Web transactions done by the user, we propose a tool chain (Figure 1b) combining network capture from in-browser monitoring features[9], with an adhoc script for *a posteriori* processing/forwarding saved Web navigation data.

We use the HTTP Archive (HAR) file format for saving Web navigation data. Our script parses HAR entries with the help of the `haralyzer` package[10], and then map event data to three links declarations (Figure 2) following the data schemata of Listing 1. *"url"* access events occurring within the same time period (i.e. `deltaThreshold` parameter defined at script level) are grouped into time buckets (i.e. linked to the same *"browser"* node). As for the above *"Live packet capture"* method, we stream graph data through a standard data forwarding process by using the WebSockets protocol.

Data visualization: WebSocket client and 3D graphs Web application. Assuming that 1) no additional software will be installed on the analyst computer, 2) multiple analysts may wish to observe

[6]https://www.wireshark.org/docs/man-pages/tshark.html
[7]IETF RFC 6455 https://rfc-editor.org/rfc/rfc6455.txt
[8]https://github.com/aaugustin/websockets
[9]See *"Network Monitor"* at https://developer.mozilla.org/en-US/docs/Tools
[10]https://github.com/haralyzer/haralyzer

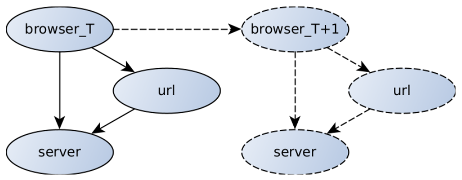


Figure 2: Graph model from HAR entities
Event data from HAR entries are mapped to three links declarations: 1) a time-stamped *"browser"* node → *"url"* link, 2) a time-stamped *"browser"* node → *"server"* link, 3) a *"url"* node → *"server"* link. Time-stamped *"browser"* nodes denote the time period at which *"url"* and *"server"* access events occur. The track of time is materialized by links between *"browser"* nodes of increasing timestamps.

the same stream of data, we propose a Web app (Figure 1c) for fetching and rendering data from the *data collection* components in a one-to-many fashion. For this, we listen to upstream data by using the WebSockets protocol. We apply a per-message processing strategy, where each message is a graph link description in JSON syntax following the schemata from Listing 1. We add data to a graph structure after nodes/edges deduplication, then make use of the `react-force-graph`[11] third party module for client-side 3D graph data spatialization (i.e. force layout) and rendering. Automatic update of the graph rendering component is ensured by the use of a ReactJS *state*[12] on the graph structure storage.

```
{
  "source": "node1_id",
  "sourceGroup": "group1",
  "target": "node2_id",
  "targetGroup": "group2",
  "time": "2021-09-08T10:35:12.138208Z",
  "value": 100
}
```

Listing 1: DynaGraph graph data scheme, with sample data

5 EXPERIMENTS AND EVALUATION

Table 2 summarizes the evaluation of our approach.

	Experiments phases	
Requirements/Criteria	Live Packet	Web Browser
Loosely coupled process	✓	✓
Stream rendering	✓	X
Objects grouping	✓	✓
Reproducibility	✓	✓
Secure solution	✓	X
Minimal user involvement	✓	X

Table 2: General approach and DynaGraph evaluation
Qualitative evaluation of experiments. "Experiments phases" correspond to the *live packet capture* and *Web browser network capture* phases, respectively. Symbols: ✓ stands for criteria observed/positive evaluation, X stands for negative evaluation.

Live packet capture. This experiments phase consists in analyzing the output of an arbitrary Web navigation session through the *"live packet capture"* and *"Web app"* components. Correct live graph rendering is carried out by visual observation, including the delay between notifications of new nodes/links and the addition of nodes/links to the graph rendering panel. The Web application reactivity is tested by visual observation of camera rotation smoothness when manipulating the rendering panel. In addition, the usefulness of Firefox *"history"* records is checked in order to allow for tshark/Firefox history comparison.

This experiments phase put forefront two kinds of limitations for both the DynaGraph system and the overarching goal of network/activity analysis: 1) the capacity of the Web application to manage high-rate data bursts (*stream rendering*), 2) the overall capacity to get insights about the Web user activity when network traffic is encrypted (*objects grouping*).

Firstly, the Web application User Interface (UI) keeps fluid for a network capture without capture filters or when navigating to a Web sites with complex content. For example, navigating to the http://scihi.org/vannevar-bush-memex/ Web page is a 327 requests process (11.24 Mb, 15.34 seconds before the *"load"* event), which in turn equates to the processing of ≃ 1380 network packets[13]. However, more complex navigation schemes induces a partial freeze of the UI at graph rendering component level until links/seconds rate decreases.

Secondly, most of the Web traffic is TLS encrypted, and thus only *"server"* nodes are added to the graph as no inner packet inspection can be conducted (i.e. using `Request URI` and *"fetch metadata request header"* HTTP headers[14]). No live side channel help can be taken from the Firefox history records as its corresponding SQLite database is locked while Firefox instances are open.

Besides this, evaluation of the *reproductibility* criteria is positive as the `tshark` tool can replay recorded captures (`-r` commutator) to the DynaGraph Web application through the `ws_server_stdin.py` script. Finally, evaluation of the minimal user involvement is also positive as the data collection tool chain 1) can be launched programmatically, 2) can be launched outside of the Web user computer (e.g. running `tshark` with the *"promiscuous mode"* on another host located in the same network, or running the data collection tool chain on an intermediary host such as a proxy server).

Web browser network capture. Similarly to the *"live packet capture"* experiment, this experiment phase consists in analyzing the output of an arbitrary Web navigation session, saved in a HAR file, through the *"Web browser network capture"* and *"Web app"* components. We reiterate the processing of the same HAR file with changes to the `deltaThreshold` parameter in order to evaluate its effect on rendering and Web navigation pattern emergence (Figures 3a to 3g). In addition, we also export TLS keys of the Web navigation session for comparing the rendering of the equivalent `tshark` traffic dump.

This experiments phase shows a highly positive evaluation degree on the objects grouping criterion as *"browser"*, *"server"* and *"url"* nodes are naturally produced by the `ws_server_har2.py` script. Evaluation of the *reproductibility* criterion is also positive as HAR files can be stored, shared and replayed at a later time. Finally, evaluation of the loosely coupled process criterion is also positive as data processing is a sub-part of the data collection process.

[11]https://github.com/vasturiano/react-force-graph
[12]https://reactjs.org/docs/state-and-lifecycle.html

[13]Thus ≃ 90 links/second rate, and 1380 links without deduplication.
[14]IETF RFC 7230 https://rfc-editor.org/rfc/rfc7230.txt and
W3C fetch-metadata https://www.w3.org/TR/fetch-metadata/

However, evaluation of the stream rendering, secure solution and minimal user involvement criteria is negative as 1) analysis is carried out *a posteriori* (i.e. on a network traffic dump), 2) HAR files include full content of the Web navigation session in a clear access fashion, 3) the Web user is highly involved in the data collection setup and restrained to the use of a single navigation panel (unless more complex setup is developed).

Visual and statistical analysis of the `deltaThreshold` parameter effect (Figure 3) lead to several additional remarks: 1) resource nodes (i.e. *“url”* and *“servers”*) are invariant although their degree vary with `deltaThreshold`, 2) the numbers of nodes and links appear to be lower bounded (Figure 3g) by the distribution of *“url”* nodes over *“server”* nodes, 3) higher bounds relate to operating system’s timers resolution, 4) it appears that a correct choice of `deltaThreshold` relates to cognition (e.g. user’s reading speed) unless network traffic is generated by non-human agent standard activity (e.g. scripted Web browsing, concurrent Web navigation traces in the same capture), 5) canonical access patterns (e.g. Figure 3h) emerge with high resolution analysis (e.g. *“varied url from same server”* or *“many time same url from same server”*). It can be tentatively concluded that the general model of such graph is a trajectory in a virtual space with multiple scales of constraints.

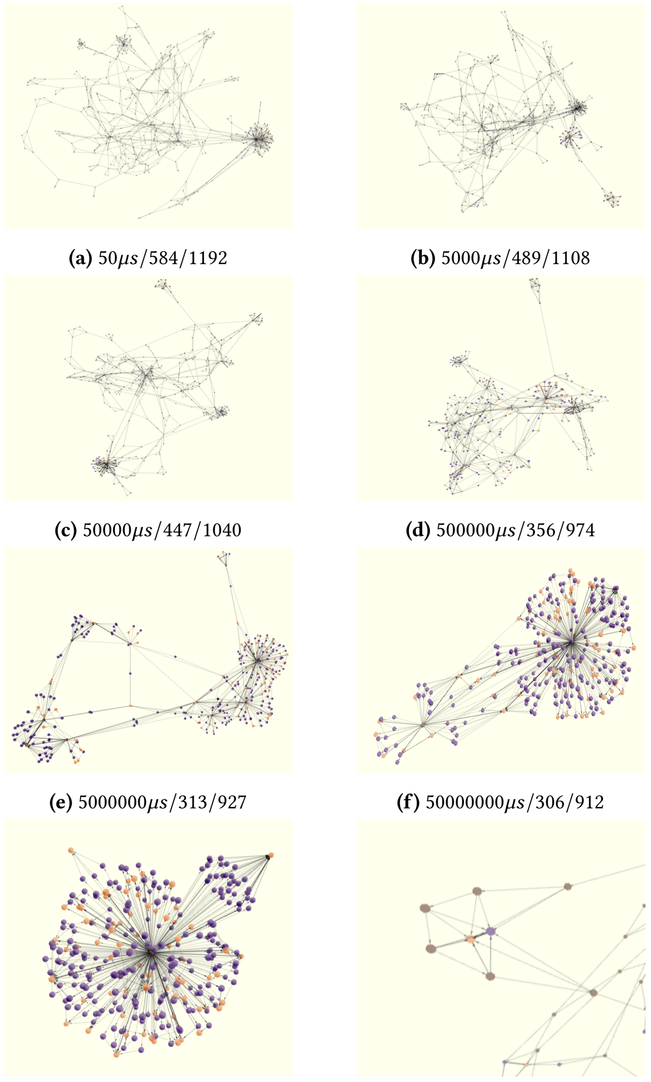

(a) $50\mu s/584/1192$ **(b)** $5000\mu s/489/1108$ **(c)** $50000\mu s/447/1040$ **(d)** $500000\mu s/356/974$ **(e)** $5000000\mu s/313/927$ **(f)** $50000000\mu s/306/912$ **(g)** $500000000\mu s/305/815$ **(h)** $50\mu s$ **pattern example**

Figure 3: deltaThreshold parameter influence on graph rendering, pattern emergence and graph statistics
Renderings derived from experiments on the DynaGraph *“656 requests HAR”* dataset: the same data set is parsed with increasing *deltaThreshold* values for rendering and exploration on separate DynaGraph instances. Node colors: violet for *“url”*, orange for *“servers”*, beige for *“browser”*. Figures’ caption: deltaThreshold/vertices number/edges number.

6 CONCLUSION AND FUTURE WORK

In this work, we aimed to understand and make use of *the form of Web navigation traces*. We firstly hypothesized that a 3-layered graph (multilayered graph) data representation would be relevant for depicting a Web navigation session. Next, we explored 1) strength and weaknesses of network traffic capture techniques at user-side on both operating system network interface and Web browser process levels, 2) feasibility and interest in having live 3D graph visualization of data derived from navigation traces.

We developed the DynaGraph framework, a system combining classical traces dumping tools (i.e. the `tshark` tool and Firefox’s Network Monitor component) and an adhoc Web application for graph data visualization. The proposed framework follows the observer design pattern with the help of the WebSockets protocol. Live 3D graph spatialization and rendering is done within the Web app with the help of the `react-force-graph` client-side third-party module. This framework proved to be effective for carrying out our experiments on streamed data, although some real-time rendering performance limitations arise on our setup for complex navigation schemes with mean links/seconds rate above 90.

Data collection at operating system network interface level appears to be easy to set up, but raises the pitfall of low tracking details whenever data encryption is present. Conversely, data collection at Web browser process level brings up expected details about Web navigation history. However, this second technique in its present form fails on live streaming and low user involvement expectations as it highly rely on Web browser features. Additional experiments with 3D spatialization and various data sampling periods reveal the emergence of graph patterns about both Web navigation events and Web content/IT architecture. For instance, the force layout spreads nodes over the representation space in such way that it gets easy to immerse into a Web session navigation path and visually identify access patterns as spatial structures. We observe prevalent structure kinds such as *“various urls from same server”*, *“several times same url from same server”* and *“many sub-resources from many servers”*.

Next step we envision is to build a graph-based navigation pattern catalog related to Web content/IT architecture archetypes. With such catalog, we expect that, beyond Web cartography concerns, improvements could be made for Web navigation with respect to performance and security. For example, combining topic discovery techniques on Web content with navigation pattern learning could lead to topic-based resource allocation algorithms. Similarly, conformance checking and policy-based approaches could help infer user intent and alert on anomalous behavior with navigation pattern comparisons.

Future works will consider the following approaches. Firstly, data collection components will benefit from additional processing in a centralized platform. Merging multiple streams in a message bus platform (e.g. *kafka2websocket*[15]) and semantic mapping/filtering/processing with Stream Reasoning Processors (e.g. StreaminMASSIF [3], RMLStreamer [11]) are interesting options towards this path. In addition, further considerations will be done on adding connectors at Web app level for RDF data handling (e.g. rdfjs/N3.js[16]), or migrating to standard graph analytics products with streaming and rendering capabilities (e.g. Gephi[17]). Secondly, data collection at the Web browser level can be made simpler and automated using a Web browser extension: the main option here is to develop a Web browser plug-in capable of catching Web request events (e.g. with Chromium's `chrome.webRequest` API[18]) and sending key informational Web navigation data (e.g. request URI, remote address, HTTP fetch metadata request headers) as an RDF stream (or alike) over a secured channel. Thirdly, activity identification and architecture classification will benefit from advanced mathematical/logical techniques for topos learning on multilayered graphs. Emphasis will be set on overcoming (potentially) missing data, such as in the case of encrypted data with the lack of HTTP headers inspection. Concerns here will fall into finding, first, an adequate semantization of the user's trace. Using HTTP RDFS/OWL models [1] is an interesting basis for taking advantage of logic-based inference capabilities over heterogeneous data. We will also consider the use of more general mathematical frameworks such as graph embeddings [13, 19], clustering on graph streams [15] and spatial analysis and reasoning [2].

REFERENCES

[1] Noam Ben-Asher, Alessandro Oltramari, Robert F. Erbacher, and Cleotilde González. 2015. Ontology-Based Adaptive Systems of Cyber Defense. In *STIDS*.

[2] Isabelle Bloch. 2021. Hybrid AI Models: Some Opportunities for Explainability Application to Spatial Reasoning and Image Understanding. https://gitlab.lip6.fr/maudetn/ia2-explainability/-/blob/master/slides/4-slides-bloch.pdf

[3] Pieter Bonte, Riccardo Tommasini, Emanuele Della Valle, Filip De Turck, and Femke Ongenae. 2018. Streaming MASSIF: Cascading Reasoning for Efficient Processing of IoT Data Streams. *Sensors* 18 (Nov. 2018), 3832. https://doi.org/10.3390/s18113832

[4] Rodrigo A. Botafogo and Ben Shneiderman. 1991. Identifying Aggregates in Hypertext Structures. In *Proceedings of the Third Annual ACM Conference on Hypertext - HYPERTEXT '91*. ACM Press, San Antonio, Texas, United States, 63–74. https://doi.org/10.1145/122974.122981

[5] Vannevar Bush. 1945. As We May Think. https://www.theatlantic.com/-magazine/archive/1945/07/as-we-may-think/303881/.

[6] Giovanna Castellano, Anna M. Fanelli, and Maria A. Torsello. 2013. Web Usage Mining: Discovering Usage Patterns for Web Applications. In *Advanced Techniques in Web Intelligence-2: Web User Browsing Behaviour and Preference Analysis*. Springer Berlin Heidelberg, Berlin, Heidelberg, 75–104. https://doi.org/10.1007/978-3-642-33326-24

[7] Oscar Corcho, David Chaves-Fraga, Jhon Toledo, Julián Arenas-Guerrero, Carlos Badenes-Olmedo, Mingxue Wang, Hu Peng, Nicholas Burrett, José Mora, and Puchao Zhang. 2021. Oeg-Upm/Devops-Infra: First Official Version. (April 2021). https://doi.org/10.5281/ZENODO.4701264

[8] Eric Freeman, Elisabeth Robson, Kathy Sierra, and Bert Bates (Eds.). 2004. *Head First Design Patterns*. O'Reilly, Sebastopol, CA.

[9] M. Ghijsen, J. van der Ham, P. Grosso, C. Dumitru, H. Zhu, Z. Zhao, and C. de Laat. 2013. A Semantic-Web Approach for Modeling Computing Infrastructures. *Computers & Electrical Engineering* 39 (2013). https://doi.org/10.1016/j.compeleceng.2013.08.011

[10] Sharad Goel, Duncan J. Watts, and Daniel G. Goldstein. 2012. The Structure of Online Diffusion Networks. In *Proceedings of the 13th ACM Conference on Electronic Commerce - EC '12*. ACM Press, Valencia, Spain, 623. https://doi.org/10.1145/2229012.2229058

[11] Gerald H, Sitt Min Oo, Gertjan De Mulder, Michiel Derveeuw, Pieter Heyvaert, Wouter Maroy, Vincent Emonet, Kmhaeren, De Ben Meester, and Thomas. 2021. RMLio/RMLStreamer: Release 2.1.1. https://doi.org/10.5281/ZENODO.3887065

[12] Ismail Harrando, Pasquale Lisena, and Raphaël Troncy. 2021. Apples to Apples: A Systematic Evaluation of Topic Models. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*. INCOMA Ltd., Held Online, 483–493. https://aclanthology.org/2021.ranlp-1.55

[13] Thomas Kipf. 2020. *Deep Learning with Graph-Structured Representations*. Ph.D. Dissertation.

[14] Jon M. Kleinberg. 1999. Authoritative Sources in a Hyperlinked Environment. *J. ACM* 46, 5 (Sept. 1999), 604–632. https://doi.org/10.1145/324133.324140

[15] Ryan McConville, Weiru Liu, and Paul Miller. 2015. Vertex Clustering of Augmented Graph Streams. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 109–117. https://doi.org/10.1137/1.9781611974010.13

[16] F. Pfaender, M. Jacomy, and G. Fouetillou. 2006. Two Visions of the Web: From Globality to Localities. In *2006 2nd International Conference on Information Communication Technologies*, Vol. 1. 566–571. https://doi.org/10.1109/ICTTA.2006.1684433

[17] Clearcode S.A. 2020. User Identification. https://adtechbook.clearcode.cc/user-identification/.

[18] Jeroen van der Ham, Paola Grosso, Ronald van der Pol, Andree Toonk, and Cees de Laat. 2007. Using the Network Description Language in Optical Networks. In *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*. IEEE, Munich, Germany, 199–205. https://doi.org/10.1109/INM.2007.374784

[19] Tianxing Wu, Arijit Khan, Huan Gao, and Cheng Li. 2019. Efficiently Embedding Dynamic Knowledge Graphs. (Oct. 2019). arXiv:1910.06708

[15]https://github.com/GenEars/kafka2websocket

[16]https://github.com/rdfjs/N3.js

[17]https://gephi.org/

[18]https://developer.chrome.com/docs/extensions/reference/webRequest/