Journal of Network and Computer Applications 124 (2018) 63–79

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

# Anomaly analysis and visualization for dynamic networks through spatiotemporal graph segmentations☆

Qi Liao [a,*], Ting Li [a], Benjamin A. Blakely [b]

[a] *Department of Computer Science, Central Michigan University, USA*
[b] *Strategic Security Sciences Division, Argonne National Laboratory, USA*

A R T I C L E   I N F O

*Keywords:*
Dynamic networks
Anomaly detection
Visualization
Graph segmentation
Spatiotemporal analysis
Analytic tool and applications

A B S T R A C T

With recent technology advance in Internet of Things (IoT) that involves human, sensors, and mobile devices, networks are not only growing much larger but more complex and dynamic in nature. The spatiotemporal dynamics of networks are represented by both topological changes and temporal shifts of attribute information associated with network components. Understanding the pattern and trend of dynamic networks is increasingly important. While data mining approaches are generally useful in analyzing the statistical properties of networks, there has been recent trend to consider bringing human into the loop, and to examine how feedback from visualizations of large-scale dynamic networks can further improve data mining and machine learning. Traditional visualization methods based on animation and sequences of snapshot graphs are also limited by human cognitive capability. We present a dynamic network analysis and visualization (DNAV) tool which explores the spatiotemporal dimensions of graph components. In particular, nodes and edges are augmented with spatiotemporal segmentation based on both topological and attribute dynamics (e.g., time and locations of connectivity). To further facilitate analysis of large dynamic networks, DNAV includes statistical dynamic overviews alongside graph views, as well as data filtering modules for scalable analysis. Using case studies on public datasets, we demonstrate the effectiveness of DNAV in understanding and analyzing anomalies in dynamic networks such as computer communication networks.

## 1. Introduction

Communication networks formed by recent technological advance in Internet of Things (IoT) are not only becoming much larger in size but more complex and dynamic in nature (Atzori et al., 2010). Consequently, the large amount of network traffic log generated by the movement of people and mobile devices are difficult to analyze. Dynamic network analysis (Carley, 2014) is an emergent scientific field to address the challenges and has profound implications for a number of important tasks. For example, a network operator may need to detect anomalies in information technology network for potential faults and security breaches. A military commander may need to gain situational awareness, and understand the relationships in terrorists networks. Advertisers may need to target high-impact users based upon user spatiotemporal actions in social networks. Or, cancer researchers may need to understand interactions in protein networks and detect anomalous tumor cells.

Dynamic network analysis is highly challenging due to spatiotemporal network dynamics in terms of both topological structure and attribute evolution. Dynamic networks are known for their temporally changing topology as a result of on/off patterns of network connectivity. For communication networks, for example, network connections may be established or removed at any moment, making static analysis of a moving target more difficult. Often, dynamic networks are also encoded in higher dimensional space. For example, edges may contain properties such as link quality metrics (e.g., loss rate, latency, or band-

☆ The work presented in this paper was partially supported by the U.S. Department of Energy, Office of Science under DOE contract number DE-AC02-06CH11357. The submitted manuscript has been created by UChicago Argonne, LLC, operator of Argonne National Laboratory. Argonne, a DOE Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.

* Corresponding author.
*E-mail addresses:* liao1q@cmich.edu (Q. Liao), tingli0919@gmail.com (T. Li), bblakely@anl.gov (B.A. Blakely).

https://doi.org/10.1016/j.jnca.2018.09.016
Received 23 March 2018; Received in revised form 17 August 2018; Accepted 27 September 2018
Available online 2 October 2018

width) and the locations where the connections are made. Similarly, nodes may contain attributes such as user profile or computing status (e.g., CPU, memory, or disk utilizations). Determining the most effective manner to analyze the many complex relationships between these attributes, especially in a temporally dynamic context, presents significant challenges.

While classic statistics and graph theories are useful in analyzing static networks, such methodologies may become insufficient with large dynamic networks. There has been research (Lahiri and Berger-Wolf, 2008) to understand and detect anomalies in dynamic networks using methods such as clustering (Bilgin and Yener, 2006) and link prediction (Liben-Nowell and Kleinberg, 2007; Han et al., 2014). However, with rapid growth of dynamic network size, volume, dimension and complexity, data mining approaches *alone* become inadequate. This is due to inherent computational infeasibility when considering all possible prediction outcomes. Alternatively, visualization techniques (Bender-de-Moll and McFarland, 2006) have been proposed to explore dynamic networks for patterns and potential abnormal behaviors. Among them, small multiples and animation (van den Elzen et al., 2014; van den Elzen et al., 2016) are two typical visualization methods for dynamic network analysis. However, it is still difficult to track changes over time due to limitations of human cognitive ability. Other dynamic network visualizations that explore three-dimensional (3D) space (Itoh et al., 2010; Oline and Reiners, 2005) or utilize superimposition and layer juxtaposition (Federico et al., 2011; Beck et al., 2012; Pupyrev and Tikhonov, 2010) have been put forward, but suffered from problems of visual clutter, readability and preservation of mental map (Archambault et al., 2011).

To that end, we designed and developed Dynamic Network Analysis and Visualization (DNAV), which is a relatively lightweight, web-based tool. One major component of DNAV is the dynamic graph view. We adopt the node-link diagram to take advantage of users' familiarity with concept of classic graphs. However, our dynamic graphs differ from traditional graphs in that spatiotemporal information is encoded within the graph components. One of the design principles for DNAV is to try to visually analyze the network dynamics without going to higher-dimensional space or using animation approaches. Therefore, the nodes and edges in our dynamic graphs are divided into segments by their temporal dimensions based on the user's selection of node and edge properties. Each segment can be further divided if there are multiple status values within one time slot. Different colors are utilized to show dynamic node and edge property evolution over time. By dividing basic graph components into segments, we accomplish the goal of showing a dynamic network in one single static view without requiring users to remember changes as in animation schemes.

While dynamic graphs are ideal for detailed investigation, another important component of DNAV are the overviews derived from results of data mining and statistical analysis. For example, temporal trends in connection magnitudes, network entropies, and property values provide a quick overview of networks over the entire time period and can suggest which time periods look suspicious and need further investigation. By combining detailed graph views and overviews, DNAV provides a scalable solution to large-scale dynamic graph analysis. We have implemented multiple interactive functions. First, a time selection bar allows investigators to analyze dynamic graphs only within certain time period. Second, the maximum hop setting enables analysts to limit the number of hops from a selected node. The resulting smaller subgraphs allow scalable visualization of large networks. Lastly, node and edge filtering by their weight attributes is included.

We evaluate DNAV over two publicly-available datasets (Whiting et al., 2015; Grinstein et al., 2009) that contain communication records involving network traffic data, time, and user proximity location information. DNAV is effective in identifying the time and location of anomalous events in network communication patterns.

The rest of the paper is organized as follows: In Section 2, we discuss and compare various data mining and visualization methods in dynamic network analysis. Section 3 describes the key components of DNAV, i.e., dynamic graph construction. Section 4 discusses the dynamic network aggregation overviews. In both sections, we introduce important metrics used for dynamic network anomaly detection and the design principles of visualization in DNAV. Section 5 describes the system architecture and implementations. Two detailed case studies are conducted in Section 6 to illustrate the usage applications of DNAV. Finally, we conclude our work in Section 7.

## 2. Related work

In general, there have been either data mining or visualization approaches in order to understand the dynamics of large complex networks. Data mining of network graphs (graph mining) is historically performed on static graphs. In recent years, researchers have focused on how to understand the evolutionary patterns and structures of networks. In a survey article (Aggarwal and Subbian, 2014), an overview of the main methods used for dynamic network analysis is discussed. In particular, evolutional analysis of dynamic graphs in terms of both the snapshots and streaming scenarios is especially challenging.

Analysis of temporal evolution of communities in dynamic networks is one major area of dynamic network analysis (DNA). Incremental community mining based on the historic community structures rather than independent approaches over static snapshots has been proposed. In particular, the static L-metric method (Takaffoli et al., 2013) was extended to analyze dynamic social networks. In addition, the FacetNet framework (Lin et al., 2009) proposed a probabilistic (Bayesian) model based on the Dirichlet distribution that naturally assigns soft (probabilistic) community memberships to nodes. Since adjacency matrix representation is useful in finding patterns including communities in graphs, Colibri methods (Tong et al., 2008) have been proposed to provide efficient (in terms of computational time and space), low-rank approximations of the adjacency matrices for both static and dynamic graphs.

Anomaly detection in time-evolving networks is another important task in the DNA domain. A recent survey (Ranshous et al., 2015) introduced detailed anomaly types and summarized the main methods used to analyze dynamic networks, such as community based methods, node and edge detection, subgraph detection and compression-based methods. Anomaly detection in time-varying graphs can also be achieved using a correlation matrix of selected features or attributes of nodes (degrees, weights, etc.) and comparing matrices' eigenvectors (Akoglu and Faloutsos, 2010). Trend motif (Jin et al., 2007) has been put forward to uncover significant events within evolving weighted, complex networks. Periodic subgraph mining method (Lahiri and Berger–Wolf, 2008) has been utilized to identify periodically recurring patterns within dynamic social networks. Evolving relational states between the entities of the dynamic networks were analyzed by using an algorithm to detect all maximal non-redundant evolution paths (Ahmed and Karypis, 2012).

The first step in anomaly detection and intrusion detection is to collect data. Various network data collection techniques (Zhou et al., 2018) based on packet, flow, and log are useful. In particular, security-related data collection (Lin et al., 2018) is critical for accurate analysis, and therefore, their functional and security objectives need to be ensured. Security data collection for intrusion detection can be applied in mobile ad hoc networks (MANET) (Liu et al., 2018) and long-term evolution (LTE) networks (He et al., 2018), and various attack types specific to these networks are addressed. Intrusion detection may utilize data fusion technologies (Li et al., 2018) or automata model (Fu et al., 2017) to detect attacks against Internet of Things (IoT) such as jamming, false, and replay attacks. Trust management systems (Zhang et al., 2017) ensure privacy of spam reporting hosts by applying homomorphic encryption.

Visualization techniques have also been proven to be useful for dynamic network analytic processes. The primary objectives of visualization design for DNA are the ability to comprehensively display spatiotemporal network information, to control visual complexity, and to improve computational scalability. The evolution of dynamic networks has been in previous work represented by animation- or movie-like approaches (Grabowicz et al., 2014; Kang et al., 2007). GraphDiaries (Bach et al., 2014) and storyboards (Beyer and Hassan, 2006) are other examples using animated visualization for dynamic network analysis.

Dynamic network visualization can also use small multiples (Burch and Weiskopf, 2014; Alencar et al., 2012). Small multiples provide an overview of network evolution by using a series of similar graphs. Other approaches have used layer superimposition and layer juxtaposition visualizations (Federico et al., 2011). However, the tradeoff between screen pixels and data points prevents the above methods from analyzing dynamic networks with a large volume of data. In addition, even if these approaches decrease complexity by multiplexing the states of a dynamic network in time, they increase human cognitive load to track changes over time.

Networks are historically based on node-link diagrams that play an irreplaceable role in network analysis, thanks to their intuitiveness in showing relational structures within a network. With the appearance of time-varying networks, there have been other structures for representing dynamic networks. For example, MultiPiles (Bach et al., 2015) compares piles of adjacency matrices instead of graphs for visualizing dense dynamic (brain connectivity) networks. Dynamic graphs can be represented with a hierarchial radial tree layout with ThumbWheel sectors (Burch et al., 2011a). Radial tree (Burch and Diehl, 2008), as a new data structure, shows the temporal evolution of relations in a static view. The radius of the outer circle serves as temporal dimension. Radial tree layout has the benefit of reducing visual clutter compared to node-link diagrams, but visualizations based on it are not as easy to comprehend as node-link based visualizations, as they do not have intuitive topological representation.

Other approaches (Itoh et al., 2010, 2012) explore multiple dimension visualizations to demonstrate temporal changes of networks by extending multiple 2D planes into 3D space, or 2.5D representations. These techniques have their advantages in presenting a complete view of dynamic networks. However, it is challenging for interactive analysis as analysts have to constantly zoom or rotate the visualization to explore the dynamic behaviors of each entity. In a parallel edge graph layout (Burch et al., 2011b), each vertical dimensional axis represents a snapshot graph. Parallel edge splatting and rapid serial visual presentation (RSVP) have been combined (Beck et al., 2012). However, similar to parallel coordinates, even with edge splatting, the frequency of edges intersections makes it hard to view the dynamics. Among other approaches in dynamic network visualization, edges have been divided by their connection dynamics and temporal properties (Li and Liao, 2016). Nodes and edges that fit a pattern defined by degree-of-interest (DOI) functions are of primary interest for analyzing local changes while maintaining an abstract overview of the global network (Abello et al., 2014). Temporal changes of social networks have also been analyzed using NodeXL and TempoVis (Ahn et al., 2011).

Despite existing work trying to effectively analyze dynamic networks, many still fall short when it comes to visualization readability, preservation of mental map, scalability and usability (Archambault et al., 2011). Compared with the above visualization methods, DNAV has notable advantages. For example, DNAV does not require as much human cognitive capability as for the animation-type approaches (Archambault et al., 2011; Grabowicz et al., 2014; Kang et al., 2007). This is due to the fact that DNAV uses one static view for visualizing the dynamics in network graphs. When compared with small multiples or similar approaches (Burch and Weiskopf, 2014; Alencar et al., 2012), which are limited to small subgraphs, DNAV supports much larger graphs as no manual comparison of snapshots of subgraphs is required. Finally, compared with multi-dimensional approaches (Itoh et al., 2010, 2012; Oline and Reiners, 2005), DNAV reduces the visual complexity by staying in the lower dimensions (e.g., 2D vs. 3D) as well as utilizing well-understood node-link architecture.

## 3. Dynamic networks - graph view

In this section, we briefly introduce DNAV and its components. Then, we focus on the most important component, i.e., dynamic network graph design and its underlying algorithms.

### 3.1. Overview

Fig. 1 illustrates an overview of the Dynamic Network Analysis and Visualization (DNAV) tool. The tool is designed to help investigators to understand and identify patterns from general-type dynamic networks and detect the potential anomalies. The basic layout of DNAV includes four primary views:

- Dynamic Graph (DG) view (Fig. 1-(2));
- Magnitude of Connectivity (MoC) view (Fig. 1-(5));
- Network Entropy (NE) view (Fig. 1-(6));
- Spatial-temporal Dynamics (STD) view (Fig. 1-(7)).

There are four additional components in DNAV:

- Node query panel (Fig. 1-(1));
- Time slider bar (Fig. 1-(3));
- Selection pool (Fig. 1-(4));
- Graph summary statistics (Fig. 1-(8)).

Lastly, there are several filtering options (by weight, hops, properties, etc) above the Dynamic Graph (DG) view (Fig. 1-(2)).

The main view is the dynamic graph that intuitively shows the relationships among entities for communication networks. The dynamic graph view presents communication networks' dynamic topologies and properties over time in a static view. This is achieved through node and link segmentation with spatiotemporal property information (to be discussed in detail in following sections).

### 3.2. Dynamic graph

A dynamic network can be represented by a time-varying graph $G_T = \langle V_T, E_T \rangle$, where $T$ represents the whole analysis time period, $V_T$ and $E_T$ signify the network entity set and related connectivity set. In our dynamic graph representation, graph components such as links and nodes are divided into segments by their temporal dimensions. Each segment represents connectivity or an entity's status in one sub-period. Different colors have been used on segments to denote dynamic property values of nodes or edges (see an example of dynamic graphs in Fig. 2). By aggregating these dynamic property values on the related link or node temporal dimensions, we achieve the goal of showing dynamic networks' temporal relations and property values in one single static view.

#### 3.2.1. Segmentation algorithms

The dynamics in large complex networks are reflected not only in their topologies, i.e., entities may come and go and links may be created and torn down constantly, but also in vertex and edges properties ($P_{V_t}$ and $P_{E_t}$), and dynamic changes of values for each property ($P_{V_t i} \in P_{V_t}$, $P_{E_t i} \in P_{E_t}$). To show these dynamics, we merge data with different timestamps but belonging to the same $V_T$ and $E_T$ sets, and then partition the related edges in our dynamic graph into $k$ main segments, where $k > 1$, and the related nodes into $k-1$ main segments (or sectors) according to their temporal dimensions (Fig. 3). Each main segment represents edge or node status in one sub-period $t' = t/(k-1)$,

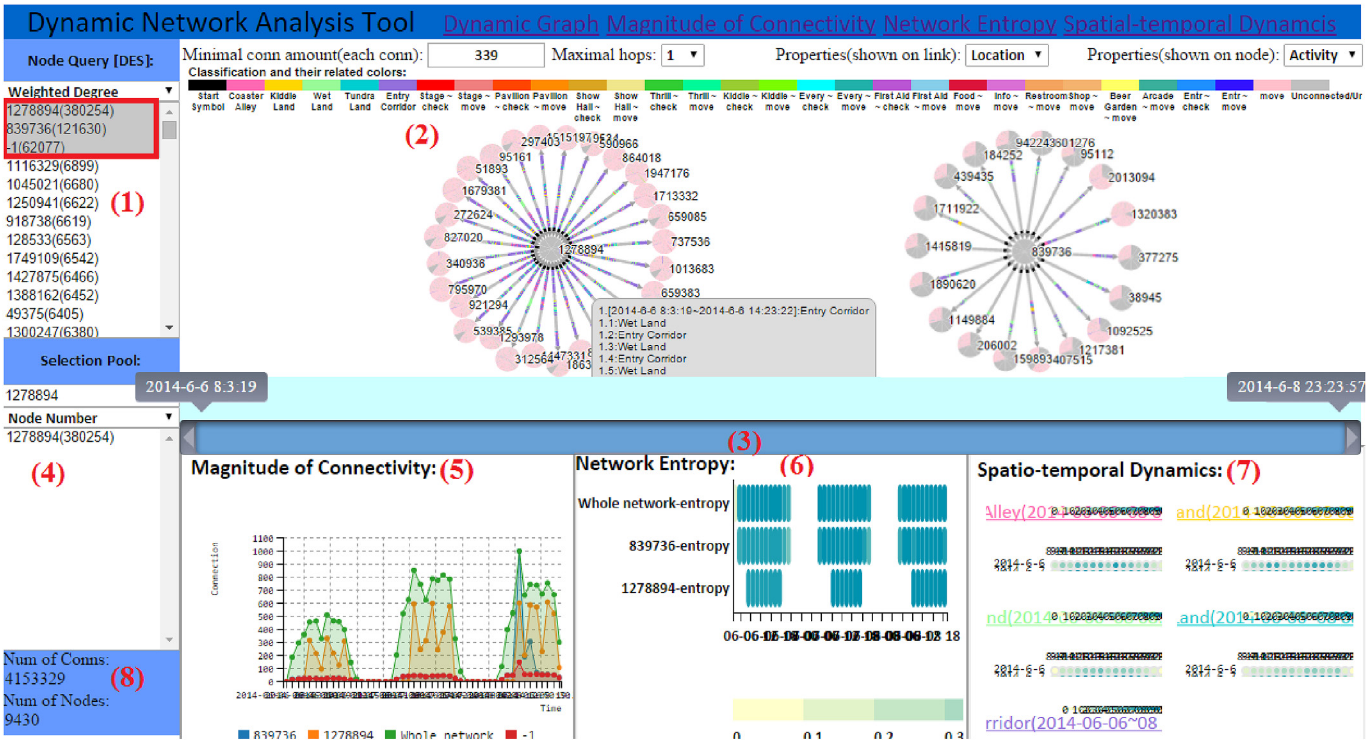


**Fig. 1.** Dynamic Network Analysis and Visualization (DNAV) tool combines multiple views: (1) Node Query; (2) Dynamic Graph; (3) Time Bar; (4) Node Selection Pool; (5) Magnitude of Connectivity; (6) Network Entropy; (7) Spatial-temporal Dynamics; (8) Entity/Connectivity Amount Text Area.

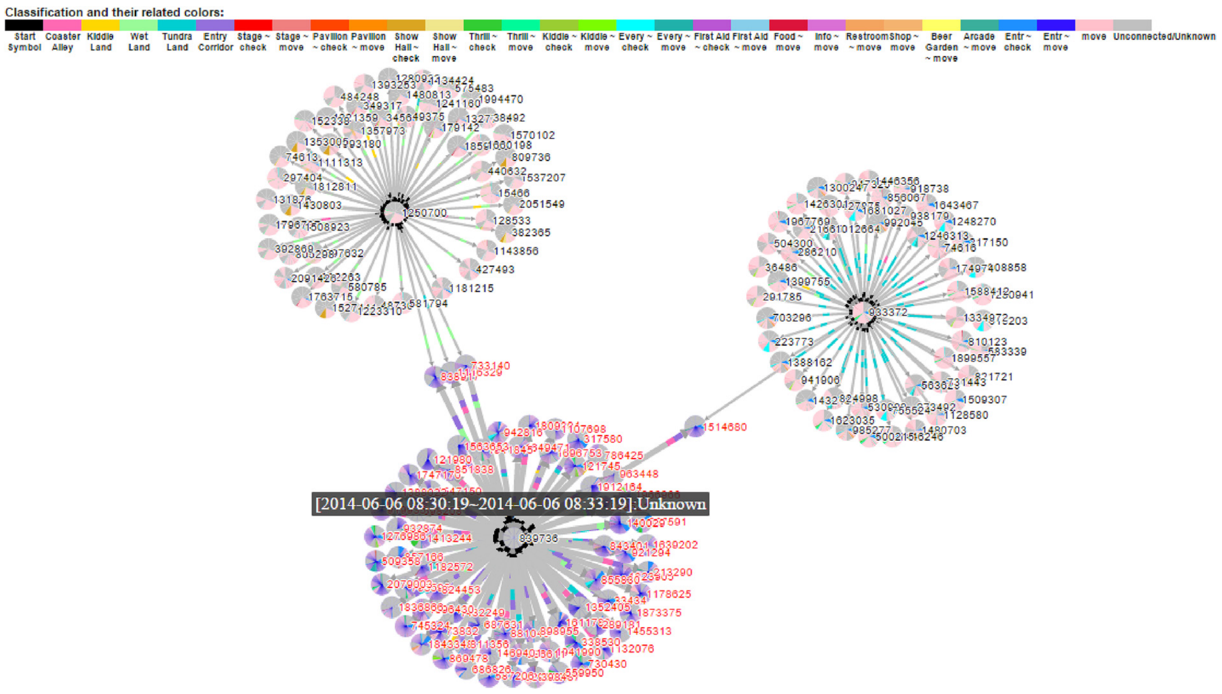


**Fig. 2.** One example of our proposed dynamic graph, in which graph components, such as links and nodes, are divided into segments by their temporal dimensions.

and may be further divided into subsegments. The merged graph can be represented by $G_t^m = \langle V_t^m, E_t^m \rangle$.

As shown in Fig. 3, among the $k$ main segments of an edge, the first (left-most) is used to indicate the initial time interval (denoted with black). Due to the bi-directional links in dynamic graphs, this initial segment helps identify the direction of temporal dimension. Nodes in the dynamic graph are divided to $k-1$ main segments in the order of their clockwise direction, with the 12 o'clock position as the start time.

If during $(i-1) * t'(2 \leqslant i \leqslant k)$ time period, an entity or link property has multiple values, we further divide the $i_{th}$ main segment into subsegments by the number of property values. For example, as shown in Fig. 3, when $i = 2$, the second main segment is further divided into

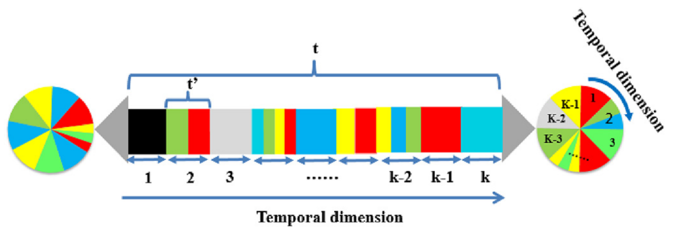


**Fig. 3.** Detailed view of segmentation. Edges and nodes in dynamic networks are divided into $k$ and $k-1$ main segments, respectively. Colors are used to encode both topological and property dynamics, e.g., black for start time, and gray for disconnection. Property values are represented by additional colors, which implicitly indicate presence of connections. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

two subsegments (denoted with different colors, i.e., red and green). Suppose the chosen link property is the location information, the two subsegments indicate that during this time slice, the communication occurred at two different places. Similarly, the second main segment of the right node in the figure can be further divided into two subsegments (denoted with green and blue).

```
Algorithm 1 Segmentation algorithm.
Input: $G_T = \langle V_T, E_T \rangle$, $k$
Output: $G_t^{w'} = \langle V_t^{w'}, E_t^{w'} \rangle$
1:  procedure
2:    $G_t^w = \langle V_t^w, E_t^w \rangle$ $(t_x \leqslant t \leqslant t_y)$ $\leftarrow$ Data filtering ()
3:    for each edge $e$ in $E_t^w$ do
4:      for m $\leftarrow$ 2 to k do
5:        e[m]['property values amount'] = 0
6:      end for
7:      for each record $r$ in $e$ do
8:        $t_i$ = r.connTime
9:        $p_i$ = r.propertyValue
10:        time_ratio = $(t_i - t_x)/(t_y - t_x)$
11:        segment_ratio = time_ratio*(1-1/k)+1/k
12:        for f $\leftarrow$ 2 to k do
13:          if segment_ratio $\geq$ (f-1)/k and segment_ratio < f/k
           then
14:            e[f]['property values amount']++
15:            subsegment_sequence = e[f]['property values
               amount'] −1
16:            e[f][subsegment_sequence]['property value'] = $p_i$
17:          end if
18:        end for
19:      end for
20:      save information of e into $E_t^{w'}$
21:    end for
22:    repeat steps 3–21 for $V_t^{w'}$ with $k-1$
23:    return $G_t^{w'} = \langle V_t^{w'}, E_t^{w'} \rangle$
24:  end procedure
```

The detail of the segmentation algorithm is illustrated in Algorithm 1. Given a raw dynamic graph $G_T = \langle V_T, E_T \rangle$, for each edge of merged segmentations $e \in E_t^w$, $(t_x \leqslant t \leqslant t_y)$, we will iterate over each of its connectivity records. Assuming one of its records has a timestamp value of $t_i$, we will first find out $t_i$'s `time_ratio` (proportion of $t$) value by using the following equation:

$$time\_ratio = (t_i - t_x)/(t_y - t_x) \tag{1}$$

Similarly, for each vertex of merged segmentations $v \in V_t^w$, $(t_x \leqslant t \leqslant t_y)$, we will also iterate over each of its records and calculate their `time_ratio` using the same equation.

As we have to keep the first main segment on an edge to indicate start time, there will be only (k-1) main segments on each link to show topology and property dynamics. Therefore, we need to normalize the `time_ratio` value into `segment_ratio`, i.e., $1/k \leqslant segment\_ratio \leqslant 1$, where $[0, 1/k)$ is kept for start time. The normalization equation is:

$$segment\_ratio = time\_ratio * (1 - 1/k) + 1/k \tag{2}$$

Then we will check which segment the record with $t_i$ timestamp value belongs to, and calculate the property values that the related connectivity or entity has in the segment by using statistical binning.

```
Algorithm 2 Segment color encoding algorithm.
Input: $G_t^{w'} = \langle V_t^{w'}, E_t^{w'} \rangle$, $k$, $Color_S$, $Color_E$, total value #
$\langle N_{P_E}, N_{P_V} \rangle$ of $\langle P_{E_t i}, P_{V_t i} \rangle$
Output: $G_t^d = \langle V_t^d, E_t^d \rangle$
1:  procedure
2:    for each edge $e'$ in $E_t^{w'}$ do
3:      e'[1]['color'][0] = Color.BLACK
4:      for l $\leftarrow$ 2 to k do
5:        if e'[l]['property values amount'] == 0 then
6:          e'[l]['color'][0] = Color.GRAY
7:        else if e'[l]['property values amount'] == 1 then
8:          e'[l]['color'][0] = $Color_S$ + e'[f][0]['property
            value'] * $(Color_E - Color_S)/N_{P_E}$
9:        else
10:          g = e'[l]['property values amount']
11:          for q $\leftarrow$ 0 to g-1 do
12:            e'[l]['color'][q] = $Color_S$ + e'[f][q]['property
               value'] * $(Color_E - Color_S)/N_{P_E}$
13:          end for
14:        end if
15:      end for
16:      save information of e' into $E_t^d$
17:    end for
18:    repeat steps 2–17 for $V_t^d$ with $k-1$
19:    return $G_t^d = \langle V_t^d, E_t^d \rangle$
20:  end procedure
```

### 3.2.2. Segment color encoding algorithm

The color encoding scheme for segments is described in detail in Algorithm 2. Dynamic graph with proper segmentation color encoding for both links and nodes can be represented by $G_t^d = \langle V_t^d, E_t^d \rangle$. As discussed earlier, the first segment is always colored black to indicate the start of time. If one edge or vertex does not have any property value, it indicates the link or node does not appear during such time period. For the rest of $k-1$ edge segments, if a pair of nodes disconnects during $(i-1) * (t_y - t_x)/(k-1)$, $(2 \leqslant i \leqslant k)$ time period, the $i_{th}$ segment will be denoted with light gray. Similarly, if during $i' * t'$ $(1 \leqslant i' \leqslant k-1)$, no status information of an entity is reported, a node segment will be denoted with light gray.

The dynamics of link and entity property values are encoded with colors other than black and light gray that indicate that an entity or connection exists during that timestamp. If one connection or node appears and has only one property value, then the $i_{th}$ main segment will be denoted with one specific color according to its property value. Otherwise if the connectivity has multiple property values, we will further divide the related link's $i_{th}$ segment into subsegments according to the number property values that connectivity has during the $(i-1) * (t_y - t_x)/(k-1)$ time period. The gradient color spectrum is defined in the following equation:

$$Color_i = Color_S + i * (Color_E - Color_S)/n \tag{3}$$

where $n$ depends on the number of different property values after using the binning method; $i$ is in the range of 0 to $n-1$. $Color = \{R, G, B\}$ is a three-tuple containing integer values of red, green and blue, $Color_s$ and $Color_E$ represent the starting color, e.g., yellow (#FFFFCC), and ending color, e.g., blue (#0093AF), respectively. Colors may be customized according to different requirement of dynamic networks analytic tasks.

### 3.2.3. Dynamic graph layout

In dynamic graphs, one edge is drawn connecting two entities, i.e., the source and target entities. The direction of the edge is denoted by arrows (source → target). If during time $t$, the source entity $v_i$ and the target entity $v_j$ of one edge do not change, we will only draw one arrow on the end of link closest to target entity. Otherwise, if two entities swap roles during $t$, we will accordingly add arrows on both sides of the related link.

Once arrows are added, the source position $\left(e_{sx},\ e_{sy}\right)$ and the target position $\left(e_{tx},\ e_{ty}\right)$ of each link must also be updated depending on whether there is an arrow between the edge and the node. To calculate the right position for each edge, we use the following equation to calculate the arc tangent for links in our dynamic graph:

$$\theta = \arctan 2(v_{ty} - v_{sy}, v_{tx} - v_{sx}); \tag{4}$$

In the above equation, $v_{sx}/v_{tx}$, $v_{sy}/v_{ty}$ represent $x$-value, $y$-value for source/target entities, respectively. If there is only one arrow on the link, we use the following equations to calculate links' source position:

$$e_{sx} = v_{sx} + \cos\theta * v_r \tag{5}$$

$$e_{sy} = v_{sy} + \sin\theta * v_r \tag{6}$$

where $v_r$ represents the radius of node in the graph. Otherwise if there are two arrows, we use the following equations to calculate links$'$ source position:

$$e_{sx} = v_{sx} + \cos\theta * (v_r + a_w) \tag{7}$$

$$e_{sy} = v_{sy} + \sin\theta * (v_r + a_w) \tag{8}$$

where $a_w$ represents the width of arrow in the graph. As there is always one arrow at the side of target entity, the equation for links$'$ target position will always be:

$$e_{tx} = v_{tx} - \cos\theta * (v_r + a_w) \tag{9}$$

$$e_{ty} = v_{ty} - \sin\theta * (v_r + a_w) \tag{10}$$

Fig. 2 demonstrates the dynamic graph view, which contains graph-related user controls, such as zoom & pan, drag & drop, and highlights of selected nodes and links. The *mouseover* event on nodes will trigger two actions. First, it will highlight a focus node and its surrounding neighbors with changing size and color of shapes. Second, it will also show detailed information about each individual node and link, i.e., the detailed change pattern of node/link property values over time. This feature helps investigators analyze both the topology and property temporal dynamics. The color legend above the dynamic graph illustrates the color codes used and their meaning for easy lookup.

### 3.2.4. Scalability

To improve scalability, we design and implement five interactive data filtering mechanisms in DNAV. These mechanisms apply to both the dynamic graph (DG) view and the network aggregation view (discussed in the next section). This is possible as the views in DNAV tool are synchronized based on the "Linking and Brushing" (Koytek et al., 2018) design principle.

The first of these filtering mechanisms is a time selection option, implemented as a double-end time slider bar (Fig. 1-(3)). By moving the time slider bar, users can select a customized time slice $t$ from $T$ and generate a subgraph $G_t = \langle V_t, E_t\rangle$, in which $t$ represents the selected time slice, from a given raw dynamic network $G_T = \langle V_T, E_T\rangle$. This can dramatically decrease the computational complexity since the tool only needs to render a view from a subset of entire dataset.

The second mechanism is a maximum hop selection (dropdown menu above Fig. 1-(2)). In our dynamic graph, each selected network entity $v_i(v_i \in v_t)$ is regarded as the root entity. Network entities that connect with a root entity directly are considered to be within one hop of a root entity, and are referred as first-hop entities. Similarly, entities which connect with the first-hop entities directly are within two hops of a root entity and so on. Once analysts set the maximum hop value, the selected root entities along with entitles that are within the maximum hop value of any root entity $v_i$ and their related connections are added into graph $G_t = \langle V_t, E_t\rangle$; $V_t$ includes the selected root entities and entities within the specified maximum hops of all root entities; $E_t$ signifies the related communication set of $V_t$. We apply the breadth-first search algorithm to iterate over network entities in the dynamic network and only analyze a subgraph from the selected nodes rather than the entire graph.

The third mechanism is to provide interactive property selection. The dropdown menus of both link and node properties (above Fig. 1-(2)) enable users to choose one or more link properties $P_{E_t i} \in P_{E_t}$ and node properties $P_{V_t i} \in P_{V_t}$. Complexity is managed by focusing on one property at a time.

The fourth mechanism is node filtering by node weight, e.g., degrees, (as illustrated in the node selection pool on the upper-left of the tool in Fig. 1-(1)). By default, nodes are sorted by their connectivity degrees (either weighted by their connection magnitude or unweighted). Node degrees is a relevant metric to measure the importance of nodes and prioritize the investigation process. With the node selection tool, analysts can either type in one particular node ID or use mouse to select single or multiple nodes (entity set $v_t \in V_t$) to analyze in the dynamic graph.

The fifth mechanism is to perform filtering based on edge weight (as shown above the color legend in Fig. 1-(2)). We add a minimal connection threshold input field for dynamic graphs. Once an investigator input a minimal threshold value, we calculate the communication records that each merged communication $e \in E_t^m$ contains. Only merged communications whose communication records are greater than the user-specified minimal threshold value, along with their related entities, will be added into $G_t^w$.

## 4. Dynamic networks - aggregation view

While dynamic graph visualization provides detailed view of network dynamics, the inherent nature of node-link diagrams limits how much information can be viewed even when efficient layout algorithms are used. In addition to the scalability measures discussed in Section 3-B4, DNAV includes additional aggregation overviews to help analyze the entire network, i.e., Magnitude of Connectivity (MoC), Network Entropy (NE) and Spatial-temporal Dynamics (STD) views. The statistical data and data mining results augment the Dynamic Graph (DG) view and provide a more complete view of dynamic networks based on the linking and brushing design principles.

### 4.1. Magnitude of connectivity

Magnitude of Connectivity (MoC) view captures one essential knowledge, i.e., network connectivity trend. In the MoC view (Fig. 4), the $x$-axis represents time and the $y$-axis represents the normalized connection count of network entities selected from the Node Query or Node Selection Pool (Fig. 1-(1) and (4)). For comparison purpose, the whole network's connection count is also included in the chart. Since the whole network has a much larger magnitude of connectivity, we normalize the data of both whole-network and selected entities to a range of 0 to $N$ by using the following equation:

$$C_n = \frac{C - C_{\min}}{C_{\max} - C_{\min}} * N \tag{11}$$

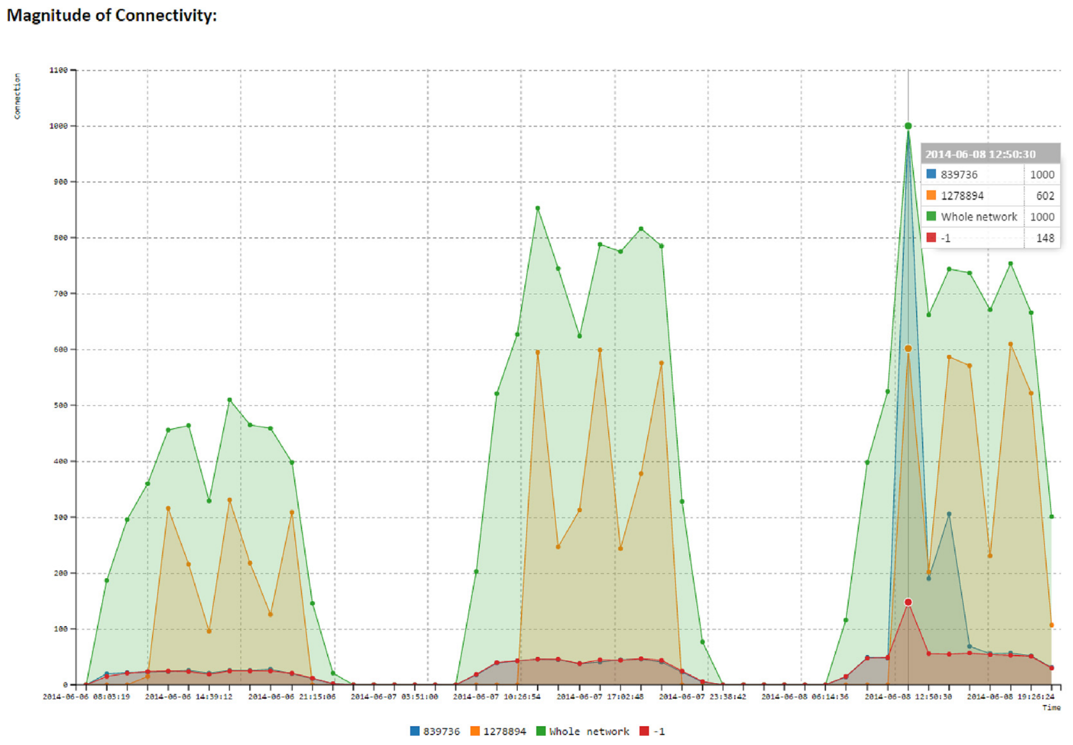


**Fig. 4.** Magnitude of Connectivity (MoC) provides a quick view for the trend of a whole network (or a subnetwork) in terms of network connectivity during the selected time range. For example, at 2014-06-08 12:50:30 (shown in popup), whole network has a significantly large network activities.

where $C_{\min}$ and $C_{\max}$ represent the minimal and maximal connectivity in the original data, respectively.

The MoC view includes several interactions as well. When users move the mouse cursor over one time point, the normalized connection count in the corresponding time period (the selected time point minus the previous time point) will be shown as in Fig. 4. When users move the cursor over a certain network entity or the whole network in the bottom part of the view, only the selected network/entity's timeline will be highlighted.

MoC is able to reveal the trend of the whole network, or selected network entities. With MoC, we can quickly identify critical inflection points. For example, in Fig. 4, spikes of network activity suggest further investigation is needed at the selected time period.

### *4.1.1. Network entropy*

While MoC shows the direct trend of dynamic networks in terms of connectivity, the Network Entropy (NE) view utilizes statistical and data mining methods to demonstrate the change of behavior as indicated by network entropies. Changes of entropy can imply significant network events (Wagner and Plattner, 2005). Like other views, NE is automatically rendered based on the Node Query/Node Selection Pool and the time-selection slider bar. The $x$-axis in the NE view (Fig. 5) is used to show analytic time, and the $y$-axis represents selected network entities. Like the MoC view, the whole network is automatically displayed for quick comparison. NE also provides detailed time and entropy dynamics once users move their cursor over a certain timestamp.

To calculate the entropy value for the whole network, we first calculate the probability of appearance for each connection by using the following equation:

$$P_i = \frac{C_i}{\sum_{i=1}^{n} C_i} \tag{12}$$

where $C_i$ denotes the frequency of the $i$th edge, and $n$ refers to the total number of connections in the dynamic network.

Next, we calculate the network entropy using the following equation:

$$E = -\sum_{i=1}^{n} P_i \log_2 P_i \tag{13}$$

Methods that we use to calculate entropy values for individual network entities are similar to methods we use to calculate entropy for the whole network. After using the selected nodes as root nodes, we differentiate connections involving the selected nodes from the list of all connections. The resulting subnetwork is then used to calculate the network entropy using the above equations.

For easier comparison, we normalize the entropy values to the interval [0,1] as $E_n$:

$$E_n = \frac{E - E_{\min}}{E_{\max} - E_{\min}} \tag{14}$$

where $E_{\min}$ and $E_{\max}$ are minimal and maximal entropy values, respectively. After getting the normalized entropy values, we then partition them into $B$ buckets. Normalized entropy values which are in the range of [(k-1)*0.1, k*0.1], ($1 \le k \le$B) will be binned to 0.1*k.

Color-coding is applied to represent different entropy values using the gradient color spectrum as defined in Eq. (3), where $n = B$ (in our case 10). With this color model, we assign a color for each binned entropy value. While normalized entropy values range from 0 to 1, the gradient colors change correspondingly from the starting color (e.g., yellow) to the ending color (e.g., blue). If, at certain point, there is a significant change in entropy value of the whole network or selected network entities, potential anomalies are implied in the network or entities.

### *4.1.2. Spatio-temporal dynamics*

Another characteristic of dynamic networks is their spatio-temporal dynamics, e.g., entities may communicate with other entities in the network at different locations over time. To analyze network connectivity

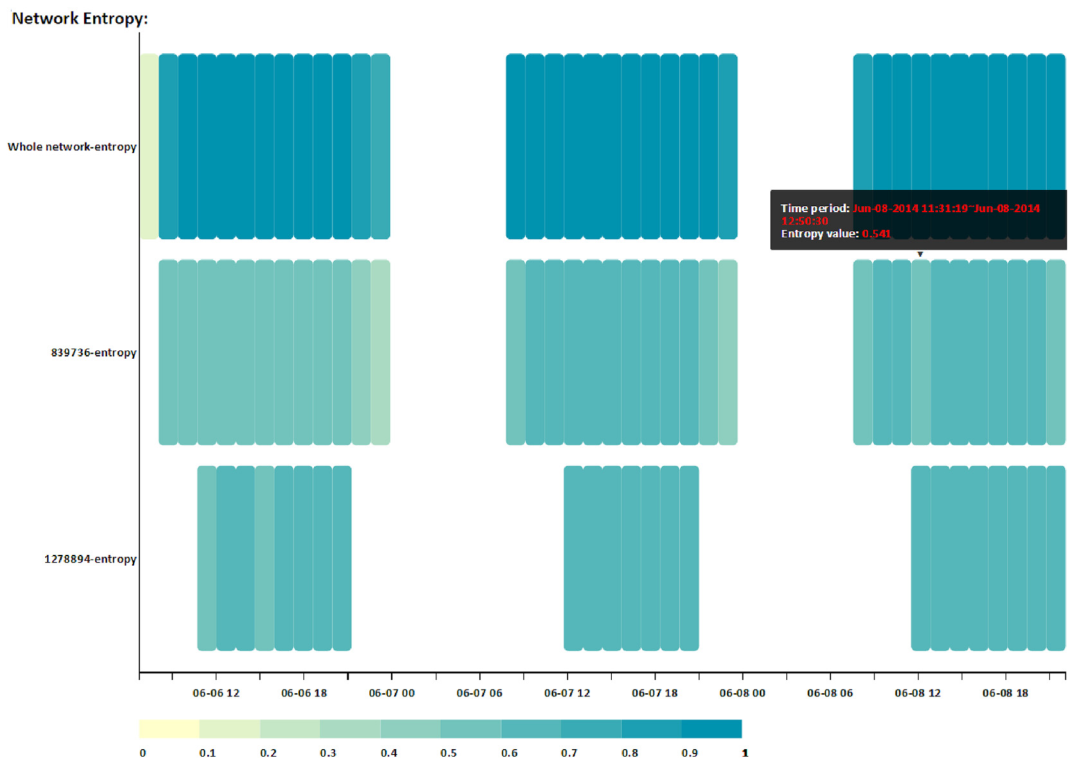


**Fig. 5.** The Network Entropy (NE) view suggests inflection point where the network status shifts. Larger difference of network entropies implies significant network events. For example, from 2014 to 06-08 11:31:19 to 12:50:30, node 839736's entropy changes comparing to the previous time period.

patterns, we categorize all network connections according to one link property, i.e., locations where the connections occur. We show the trend for each of the locations in the network through the Spatio-temporal Dynamics (STD) view (Fig. 6). As with previous views, STD uses normalization and binning for clarity.

In STD, rows and columns represent the date and time, respectively. Each cell is used to show one selected network property that can be derived from node and edge properties. We can use connection count as an example. Since location is a common property associated with connections in dynamic networks, STD allows users to choose multiple locations side-by-side for comparing network property trends at different locations. For example, Fig. 6 shows different peak network activity patterns at five different locations. Shifts in patterns may suggest anomalous time and location values.

The dynamic network aggregation views enable a scalable approach to dynamic network analysis and provide a quick starting point for investigation by observing the trend of network topology and node/link property changes. While each of these views explores different characteristics of the dynamic network, when linked together, they can provide a complete overview of the whole network or network entities. This allows users to further drill down into interesting time periods or potentially anomalous entities by using dynamic graph views.

## 5. System architecture and implementation

This section describes components of underlying system of DNAV and its implementation details. An overview of DNAV system implementation is included in Fig. 7. We develop the DNAV tool based on the server-client model rather than a desktop application by taking advantage of scalability, accessibility and convenience. An investigator can analyze and visualize her network from anywhere in the world with the Internet and a web browser by visiting a URL. In the following section, we describe each component in the server and client side.

### 5.1. Server-side

Data Formatter (DF) is implemented using Python script language to parse raw data into a database for querying from the client side. Given a network dataset, essential attributes for both entities and connections are first extracted. Significant attributes include timestamp, source ID, destination ID, and all relevant property information, e.g., location of the connection. This data is then stored in a database (MySQL). We use a database since DNAV includes multiple interactive functions such as temporal navigation, max hops selection, entity selection, node/edge weight filtering, etc. All of these operations depend on relevant information that can be obtained with queries to such a database. The data format is shown in Tables 1 and 2. There are two main tables, networkConnectity and networkEntity.

In addition, a web (Apache) server services visual analytic requests from a client machine. The web server utilizes Data Processors (DPs) to query the database and generate JSON, CSV or TSV data files. The DPs, written in PHP, extract and integrate information from the networkConnectity and networkEntity tables so that the information can be presented in all visualizations of DNAV.

There are instances of DPs for each major view in DNAV. For the MoC view, once a user selects certain network entities from the Node Query (Fig. 1-(1)), an instance of DPs will be created. This will search all connections involving the selected entities from the networkConnectity table and calculate the total number of network records in each time interval. The final results are then written to data files (e.g., CSV).

For the NE view, users need to select network entities from the Node Query first. Instead of calculating network connections, a separate DP calculates entropy values in each time interval for the selected network

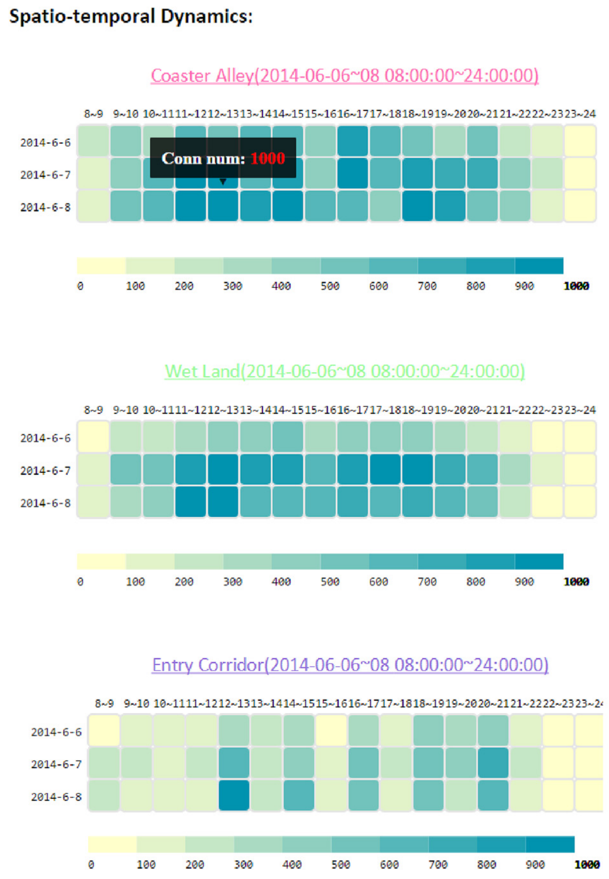


Fig. 6. Spatio-temporal Dynamics (STD) view provides location-based network dynamics. The example shows juxtaposition of network connectivity dynamics at different time of three days at five difference locations.

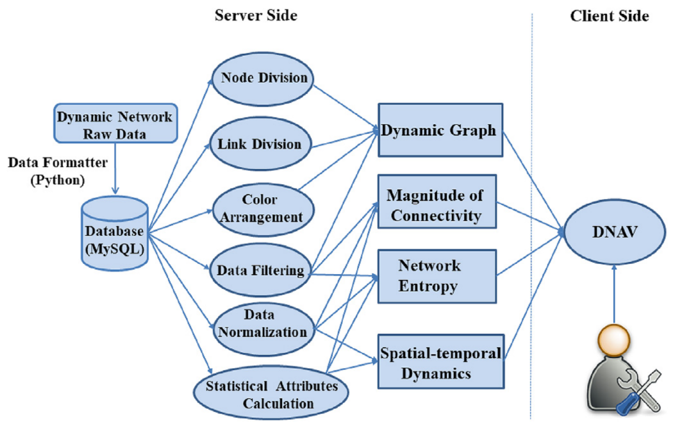


**Fig. 7.** System architecture of DNAV based on client-server model.

**Table 1**
Attributes in networkConnectity table.

| Attribute Name | Example |
|---|---|
| sequenceId | 0 |
| connTime | 2014-06-06 08:03:19 |
| srcEntityId | 439105 |
| destEntityId | 1053224 |
| $\text{property}_1$ | $\text{value}_1$ |
| … | … |
| $\text{property}_n$ | $\text{value}_n$ |

**Table 2**
Attributes in networkEntity table.

| Attribute Name | Example |
|---|---|
| sequenceId | 0 |
| connTime | 2014-06-06 08:03:19 |
| entityId | 1591741 |
| $\text{property}_1$ | $\text{value}_1$ |
| … | … |
| $\text{property}_n$ | $\text{value}_n$ |

entities and the whole network by querying information from the networkConnectity table.

For the STD view, a DP will first categorize all network connections in the networkConnectity table by one of their spatial properties such as location information. For each spatial property, the DP retrieves other relevant properties from the database such as connNum, requestSize, or responseSize, and calculates the sum of the selected property in each time unit. Last, the DP writes the result into files.

Last is the dynamic graph (DG) view. In addition to selecting entities, users may select a time interval, maximum hops, minimal connection threshold, and edge/node property to analyze. A DP first queries all nodes and edges that appear in the selected time interval from the networkConnectivity and networkEntity tables. The DP will query records to display on links that include the selected entities and meet the user selection parameters. It will then segment the edges and nodes according to the segmentation and coloring algorithms discussed in Section 3. The final result is written into JSON files which are read by the DG view.

### 5.2. *Client-side*

Users utilize a web browser to visually explore network graphs using DNAV. Views are implemented using JavaScript with Data-Driven Document (D3) libraries. The MoC view also uses the D3-based reusable chart library (C3). Additionally, we have used JQuery for data transmission and JQrangslides for the time slider bar.

As users switch between views, choose filter parameters, or make selection, the operation is transmitted via AJAX to the server asynchronously. For example, when users adjust the time slider bar or select entities in the visualization tool, the selected start time, end time and entities will be transmitted to the DPs on the web server via AJAX. Consequently, DPs make queries to the back-end database server. The processed data is then returned by the DP to client side for visualization.

## 6. Case study

In this section, we illustrate the usage of DNAV using two publicly available datasets. One dataset (Whiting et al., 2015) is related to dynamic wireless communication networks including user locations. The other dataset (Grinstein et al., 2009) is comprised of dynamic networks from network traffic logs augmented by movement of employee proximity card logs (e.g., wearable employee ID badges). Through thorough case studies, we demonstrate how DNAV may be used to identify and analyze the causes of potential anomalies in dynamic networks of relatively large size.

### 6.1. *Case: communication network with location information*

In this case study, we utilize the VAST 2015 dataset (Mini-Challenge 2) (Whiting et al., 2015). The location and communication network data are collected from visitors at DinoFun World theme park for an event honoring a celebrity between June 6th and 8th, 2014 (Friday, Saturday and Sunday). DinoFun World Park is equipped with sensor beacons that record visitors' movements and locations within the park. All pathways in the park are covered by these sensors, as are the ride check-in locations. Locations are not recorded while people are on rides or inside attractions (including restaurants, stores, and restrooms).

Attributes of individual's spatiotemporal records include time, visitor ID, type of activity (either check-in or movement), and $(x, y)$ coordinates. The communication network data set contains 9410 IDs (visitors or park services) and 4,153,329 connections. All visitors to the park are assigned an ID, use a park app to check into the park and rides, and to communicate with fellow visitors. Each inter-visitor communication connection record has four attributes: time, sender ID, recipient ID, and location where the communication is initiated. Location values can be one of five zones: Coaster Alley, Kiddie Land, Wet Land, Tundra Land, or Entry Corridor. The need for investigation is due to an incident of vandalism in the park. A soccer celebrity (Scott Jones) was scheduled to appear in two shows located at Coaster Alley, but the event did not go as planned due to this vandalism. Thus, investigators need to analyze and understand the spatiotemporal communication patterns in the park to attempt to determine when & where the vandalism occurred.

Given the dataset, we first categorize the (x,y) coordinates of each data record into Thrill Rides, Kiddie Rides, Rides for Everyone, Food, Restrooms, Shopping, Shows and Entertainment, Beer Garden, and Information and Assistance, based on the location information from the provided park map.

After loading the dataset into DNAV, we found that the communication patterns suggest that the park was opened around 8am and closed around 12am each day. We then used the time bar to select the whole time period (Friday, Saturday and Sunday) and choose the "weighted degree" option. Entities in the selected time period were sorted by their degrees and displayed in the Node Query. There were three IDs which stand out for their large volume of connectivity (highlighted with red rectangle in Fig. 1-(1))). They corresponded to ID 1278894 with a weighted degree of 380254, ID 839736 with a weighted degree of 121630, and the external ID (−1) with a degree of 62076. As a good starting point, we chose these three IDs and analyzed them with the three dynamic network aggregation views included in DNAV. In each of the three views, we found noticeable patterns.

#### 6.1.1. *Patterns in dynamic network aggregation views*

The Magnitude of Connectivity (MoC) view shows the connectivity trend of selected entities and the whole network. We learn from the whole network (green line in Fig. 4) that there are more connections on Saturday and Sunday than Friday. On the third day (Sunday) during 2014-06-08 11:31:19 to 2014-06-08 12:50:30, the whole network's activities reach a peak.

Another pattern that the MoC view makes apparent is that ID 1278894 (yellow line) only appears periodically (12pm–10pm) on each of the three days, i.e., from 12 to 1pm, 4–5pm, and 8–9pm, when it tends to have a peak connectivity. For ID 839736 (blue line) and external (red line), their connectivities have similar pattern. Based on this information, we assumed that there might be significant events during these time periods. To validate our assumption, we looked at the NE view for additional patterns.

In the Network Entropy (NE) view (Fig. 5), we saw that the entropy patterns of IDs 839736 and 1278894 are different from the whole network. The whole network, as well as selected entities, have a large entropy change (the whole network's entropy changes from yellow to blue on 2014-06-06) at the beginning and end of each day. We assessed these changes to be normal as we expected the activity of the network in its entirety to have a significant change between times the park is open, and times it is closed. However, there are other entropy changes for IDs 839736 and 1278894 during 2014-06-08 11:31:19 to 2014-06-08 12:50:30 and 2014-06-06 14:39:12 to 2014-06-06 15:58:23, respectively. The entropy value for ID 1278894 during this period ranges from 0.603 to 0.595, which is less than the change for ID 839736 (0.629–0.541). Information shown in both NE and MoD views lead our investigation to the abnormal 2014-06-08 11:31:19 to 2014-06-08 12:50:30 time period.

From the Spatiotemporal Dynamics (STD) view (Fig. 6), one pattern is that the Coaster Alley tends to have more connectivity during the hour intervals of 11–15 and 16 to 20. We formed a conjecture that these two time periods correspond with two daily showtimes in that area. When visitors gather around the stage during the show time, the communication volume tends to spike. In addition, we could see that the Entry Corridor's connection number has an alternating high and low pattern during the hour intervals of 12–21. It tends to have a higher volume of connectivity from 12 to 13, 14 to 15, 16 to 17, 18 to 19, and 20 to 21. This may indicate the peak times for tourists entering or leaving the park.

#### 6.1.2. *Patterns in dynamic network graph views*

We utilize the Dynamic Graph (DG) view to further analyze the three IDs (1278894, 839736, and external) with the highest degrees of connectivity and drill down into the suspicious time period (2014-06-08 11:31:19 to 2014-06-08 12:50:30) to reconstruct what happened during this period.

As seen in Fig. 8, network connections associated with ID 1278894 always start at Entry Corridor (purple on edges). One example is highlighted with a red circle. After that, the connections disperse among all locations. The network connections also exhibit bidirectional behavior, as shown by arrows on the edges. The node segmentation (gray) indicates there is no movement associated with its location. This pattern implies that ID 1278894 is a fixed entity located at Entry Corridor and may be part of the park's own infrastructure. Visitors appear to use the park's mobile app to check into the park at the Entry Corridor and later the rides at different areas in the park.

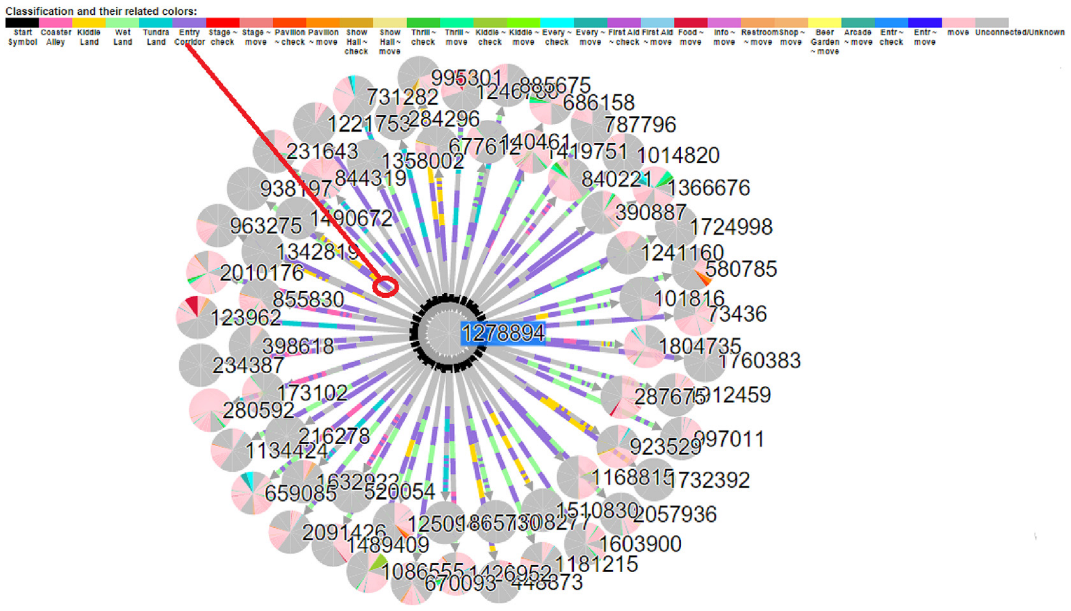
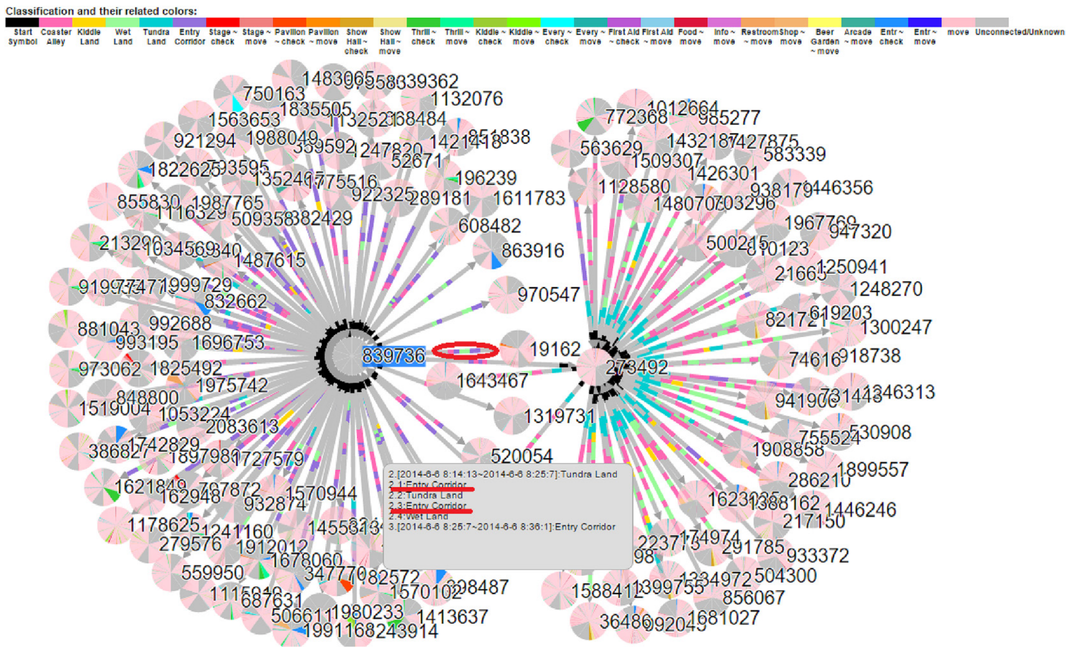


**Fig. 8.** Communication patterns of ID 1278894 suggest connections always start at the location of Entry Corridor (purple link segments) and are bidirectional. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)



```
2.[2014-6-6 8:14:13~2014-6-6 8:25:7]:Tundra Land
2.1:Entry Corridor
2.2:Tundra Land
2.3:Entry Corridor
2.4:Wet Land
3.[2014-6-6 8:25:7~2014-6-6 8:36:1]:Entry Corridor
```

**Fig. 9.** ID 839736's connections initiate at various locations but always terminate at the Entry Corridor (purple link segments) and are bidirectional. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

One distinction between the patterns of ID 839736 and ID 1278894 is that while ID 1278894 only communicates from 12 to 22 each day at the Entry Corridor, ID 839736 communicates with other IDs during the whole period (see Fig. 4). Another pattern apparent in Fig. 9 is that while the initiation point of communications from ID 839736 varies, the termination point is always in the Entry Corridor (purple). Like ID 1278894, connections with ID 839736 exhibit bidirectional communications, and the node segment information indicates a lack of movement. These patterns, along with the high degree of connectivity for ID 839736, support our conjectures. First, ID 839736 is located at the Entry Corridor. Second, visitors always communicate with ID 839736 first, before it sends a reply. Third, ID 839736 is a component of park infrastructure, located in the entry corridor.

The external ID (denoted by −1) stands out for its large connectivity volume as well. When analyzing connectivity of the external ID, we found patterns as shown in Fig. 10. One communication pattern is that all connections are unidirectional. Combined with patterns from the network aggregation views, we conclude that only people inside the park can establish contact with the outside, and the external ID can be any entities outside of the park.

After getting a better picture of communication patterns of the most frequently occurring node IDs, we continued our analysis into the interesting time period from 2014 to 06-08 11:31:19 to 2014-06-08 12:50:30 derived from dynamic network aggregation views. We move the time

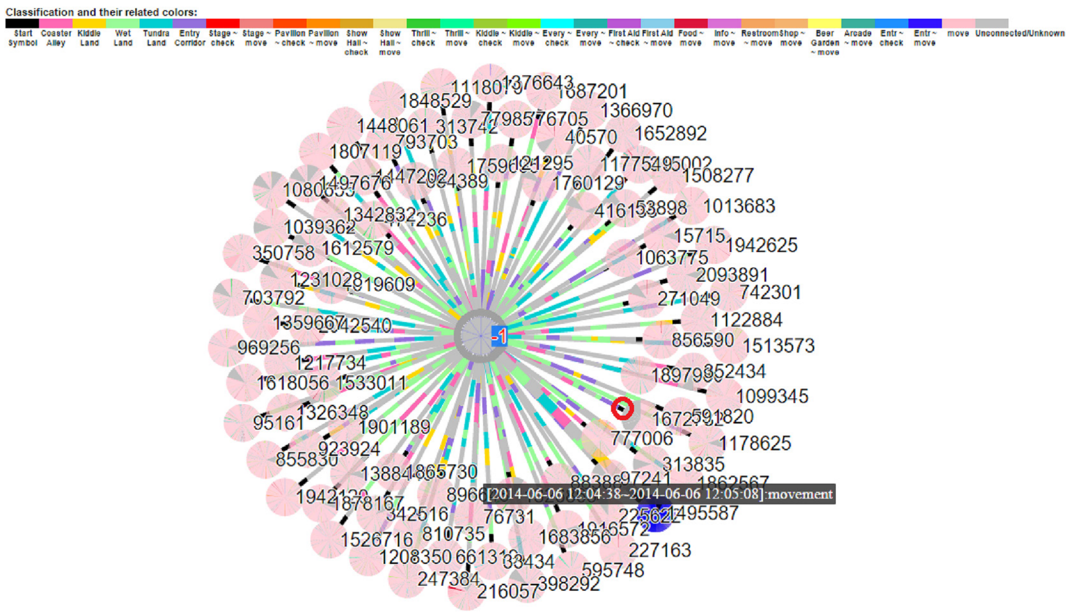
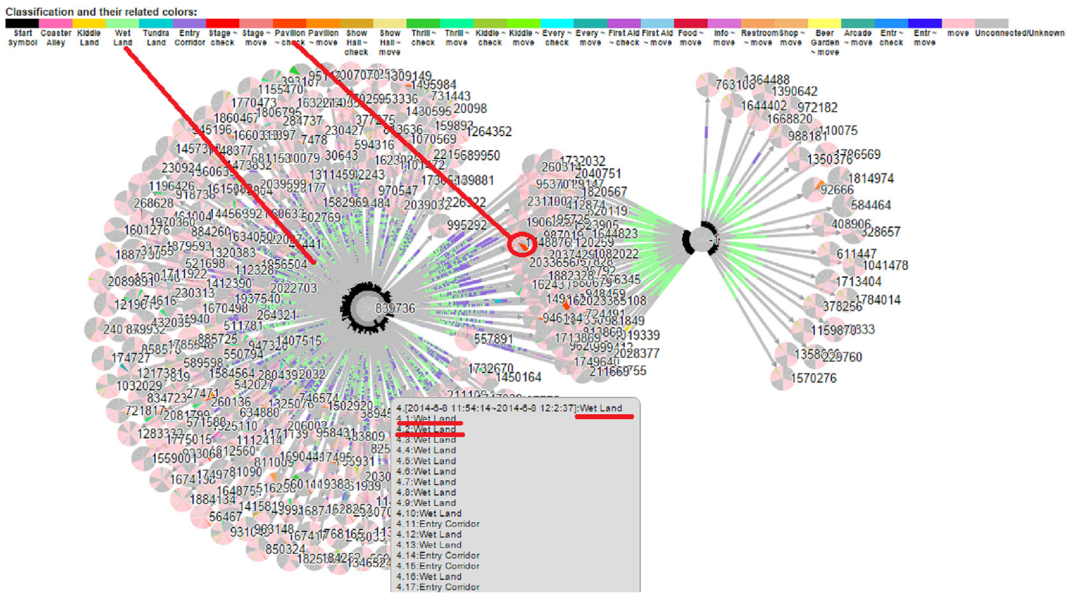


**Fig. 10.** The external ID (−1) is always the target entity of unidirectional connections from inside the park.



```
4.[2014-6-8 11:54:14~2014-6-8 12:2:37]:Wet Land
4.1:Wet Land
4.2:Wet Land
4.3:Wet Land
4.4:Wet Land
4.5:Wet Land
4.6:Wet Land
4.7:Wet Land
4.8:Wet Land
4.9:Wet Land
4.10:Wet Land
4.11:Entry Corridor
4.12:Wet Land
4.13:Wet Land
4.14:Entry Corridor
4.15:Entry Corridor
4.16:Wet Land
4.17:Entry Corridor
```

**Fig. 11.** Since 2014-06-08 11:54:14, IDs start to frequently communicate with ID 839736 and the external ID from the Wet Land, where the vandalism happened. This is indicated by both green (Wet Land) link segments and orange (Creighton Pavilion) node segments. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

bar to set the start and end time as 2014-06-08 11:29:05 and 2014-06-08 12:52:56, select the minimum connection weight as 20, and choose IDs 839736 and the external ID. A dynamic graph view is generated as shown in Fig. 11. As seen from the visualization, many IDs start to frequently communicate with ID 839736 and the external ID at Wet Land (green) starting at 2014-06-08 11:54:14. The status shown on nodes also indicates that many visitors are moving around Creighton Pavilion (highlighted with a red circle).

Comparing the above patterns with Figs. 2 and 10, both IDs 839736 and the external ID do not exhibit similar connection patterns in other time periods. We conclude that the vandalism was discovered around 2014-06-08 11:54:14 at Creighton Pavilion. Once the vandalism was discovered, visitors initiated many communications with ID 839736 (park information desk). In addition, visitors started to communicate with the external ID. We interpret this as communications with absent friends or family members to relay news of an unusual event in DinoFun Park.

### 6.2. *Case: network traffic with employee proximity cards*

In this case study, we will analyze a netflow-like network traffic log, along with logs of employee movement recorded by proximity cards. This data is taken from the two VAST 2009 (Mini-Challenge 1) datasets (Grinstein et al., 2009). The first dataset is proximity card log. Proximity cards are plastic ID cards, often worn as badges, with an embedded RFID chip and antenna. Each embassy employee uses such an identification token to open doors to and within the Embassy. Each data record contains an employee number, proximity card number, date and time

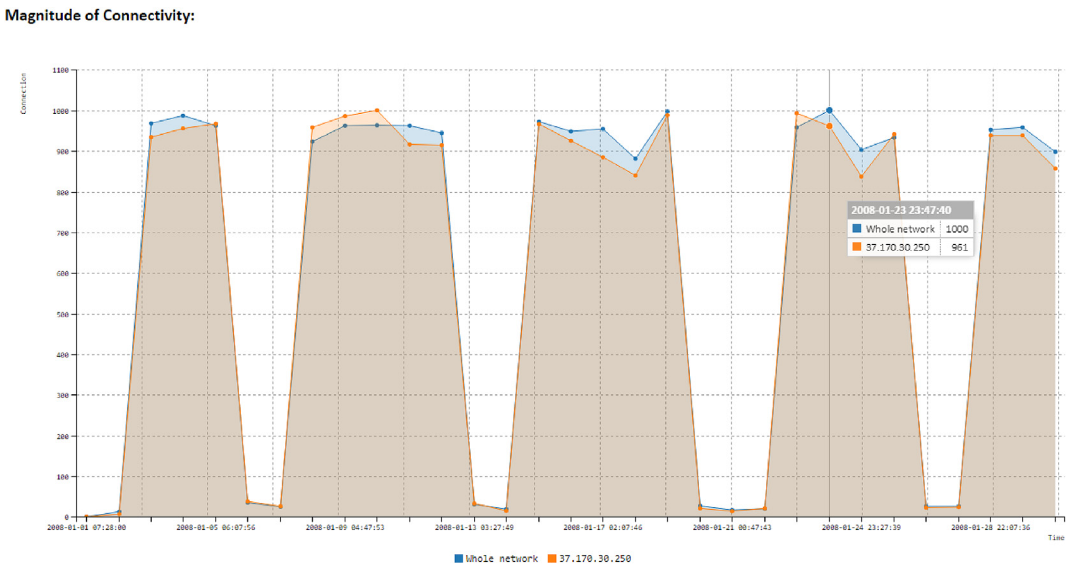


**Fig. 12.** The MoC view of one month's traffic data suggests a peak network connectivity on 01/23/2008 and little connectivity over weekends and holidays.

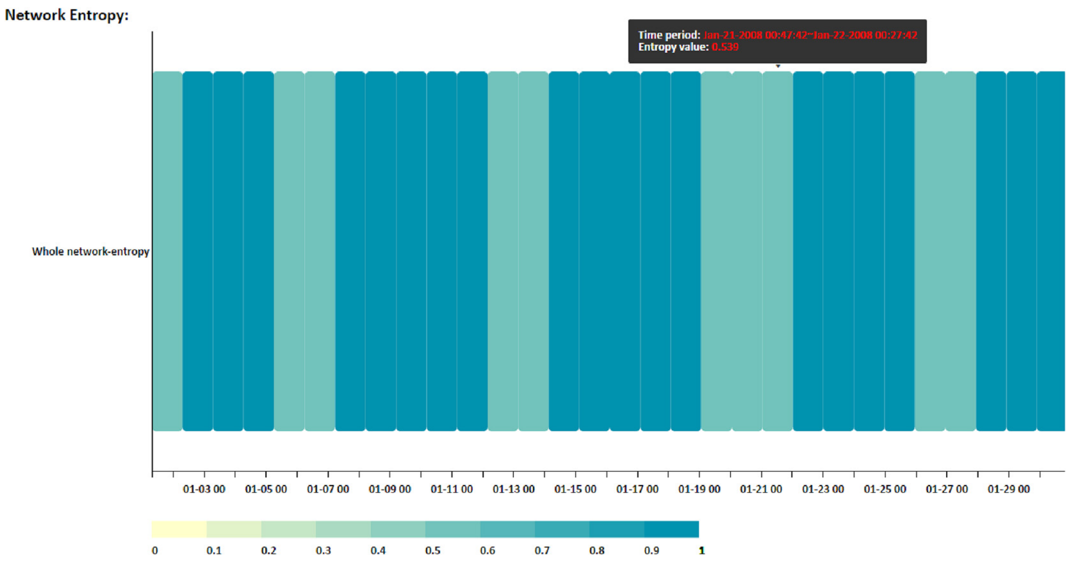


**Fig. 13.** The NE view suggests that during the whole month, except for weekends and holidays, the network entropy of the whole network has no significant change.

of use, and location of use. The second dataset is network traffic logs. Each employee has been assigned a desktop computer with a static IP address for use in their daily duties. The log data consists of the computer IP address, employee ID of the computer's owner, and outgoing and incoming activity from the computer including: destination sites, payload (request and response data), and port numbers. Both datasets cover the same period of one month.

An employee was suspected of sending data to a criminal organization from within the Embassy. We needed to investigate the situation. First, we loaded both datasets into DNAV, and explore potential patterns in the tool's dynamic network aggregation views.

### 6.2.1. *Patterns in dynamic network aggregation views*

The MoC view (Fig. 12) demonstrates an overview of network connectivity over the entire month of January 2008. Weekdays exhibit similar connectivity behavior with connection numbers ranging from 900 to 1000 (normalized) except for weekends and a holiday (01/21) where there is almost no connectivity. The network reaches a peak connectivity of 1000 on 01/23/2008.

We could also see a similar pattern from the NE view (Fig. 13). The entropy value of the whole network changes dramatically between weekdays and weekends but not as much between the weekdays. We concluded that the suspect in the Embassy is able to leak data to outside without changing the normal distribution of connectivity.

Since the suspect employee sent data to a criminal organization, it is reasonable to consider traffic with large request payload (upload traffic) but small response payload (download traffic) to be anomalous. To try another investigative approach, we loaded the STD view (Fig. 14) and observed the request/response rate patterns. In this view, the rows are days of the month and columns are the IDs of each computer in different restricted areas of the Embassy.

Fig. 14 suggests there are several suspicious machine IDs that continue working during weekends or holidays, such as IDs 8, 10, 16, 20 and 32. Additionally, there are a few other machines with high request/response rates that drew our attention. On the other hand, machine IDs 25, 26, 36, 37, 48, and 49 have low request/response rates.

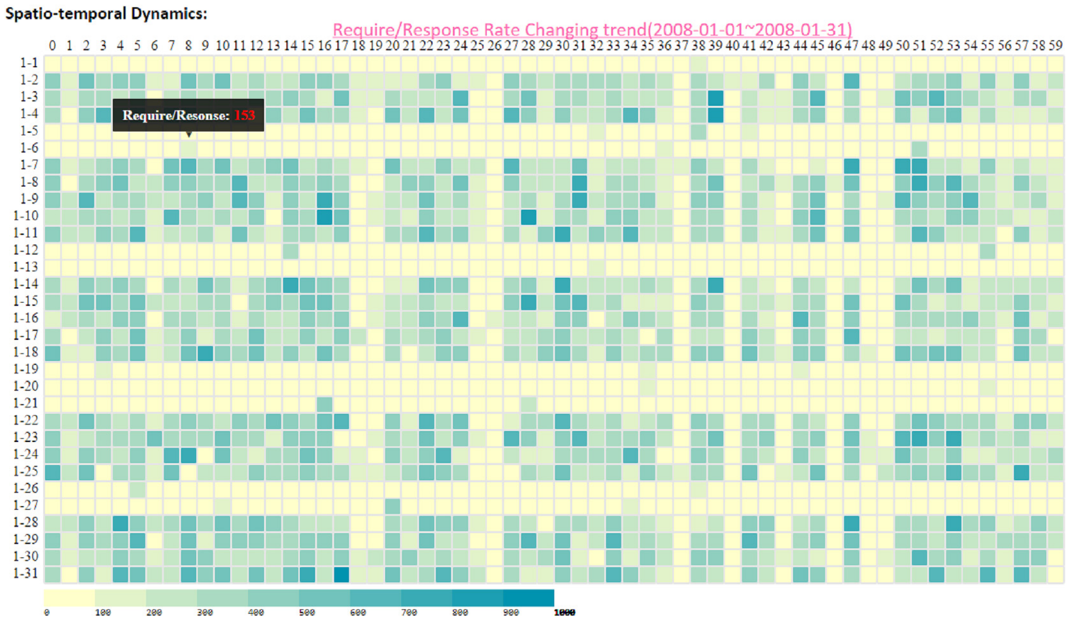
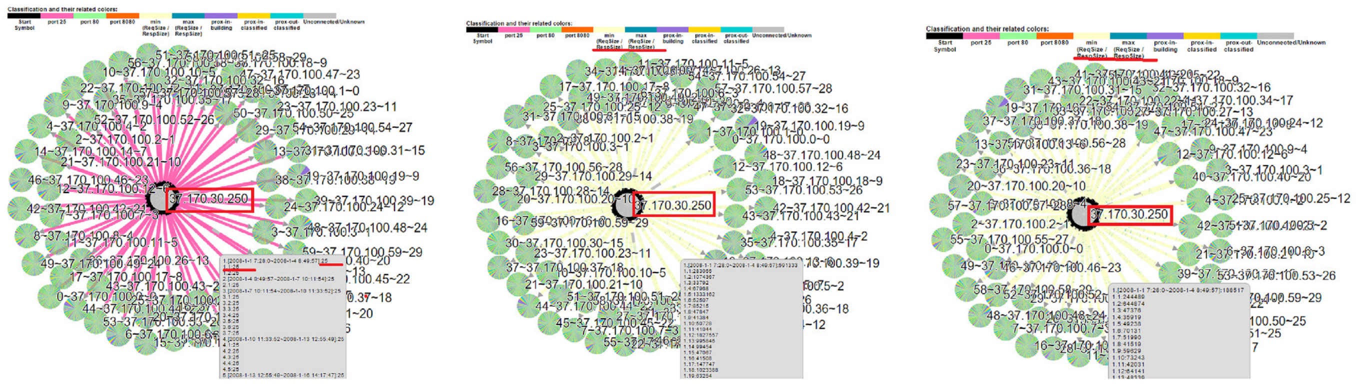


**Fig. 14.** The STD view in which rows are days of the month, columns are machines at difference locations, and cells are request/response payload ratios. Machines with high network connectivity during off-hours or with high request/response ratios are suspicious for leaking data to outside.



(a) Connections are all over port 25 (pink link segments) (b) Small request payloads (yellow link segments) (c) Small response payloads (yellow link segments)

**Fig. 15.** The DG view shows IP address 37.170.30.250’s connections’ temporal trend.

We therefore judged that there is a low probability these were used by the suspect employee.

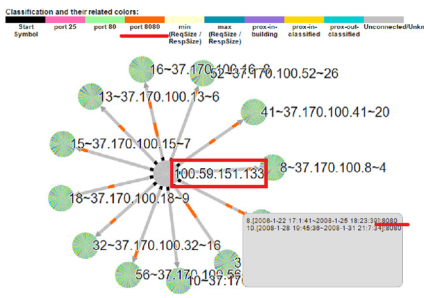
## 6.2.2. Patterns in dynamic network graph views

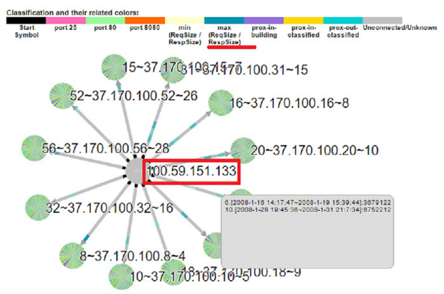
After analyzing patterns in the dynamic network aggregation views of the whole network, we drilled down into specific machines by using the dynamic graph (DG) views. We first moved the time slider bar to include the entire month, and select the “Request” attribute from the Node Query options. IDs were sorted according to the payload requested by internal machines in descending order. We saw from the Node Query the top two IP addresses: 37.170.30.250 (a total request payload of 6015113957) and 100.59.151.133 (a total request payload of 144634785). Since we assumed the machines used by the suspect employee to leak information to his contact will most likely generate a large payload, we chose these two machines to further analyze their temporal properties in our dynamic graphs.

In the DG view (Fig. 15), we choose port numbers and request/response payload sizes as the link properties. This reveals that all internal machines are communicating with 37.170.30.250 through port 25 (pink links) and they tend to have a comparatively small payload of both requests and responses (yellow links). Since port 25 is the standard port for Simple Mail Transfer Protocol (SMTP), the ubiquitous protocol for servers exchanging email, we assumed IP address 37.170.30.250 is an email server for the embassy and considered it as normal.

We then analyzed the machine with IP address 100.59.151.133. As seen from Fig. 16, there were 12 machines connected with IP 100.59.151.133 during the month. First, we selected port number as the link property. These internal machines connect to IP 100.59.151.133 using destination port 8080 (orange links). Port 8080 is commonly used for auxiliary web services.

We then selected payload size as the link property. These internal machines have large request/response ratios, i.e., high request payload (blue) but low response payload (yellow). These patterns drew our attention to machine 100.59.151.133, which we suspected belongs to the criminal destination IP outside of the Embassy. Twelve interior


(a) Connections are all over port 8080 (orange)


(b) Large request payloads (blue)

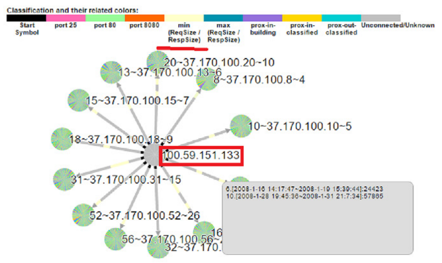
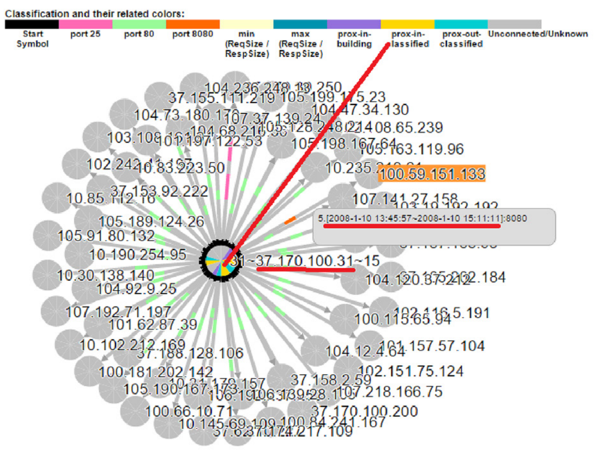

(c) Small response payloads (yellow)

**Fig. 16.** Suspicious outside IP 100.59.151.133 communicates with 12 interior machines.


(a) 2008-1-10 13:45:57 to 2008-1-10 15:11:11, employee 31

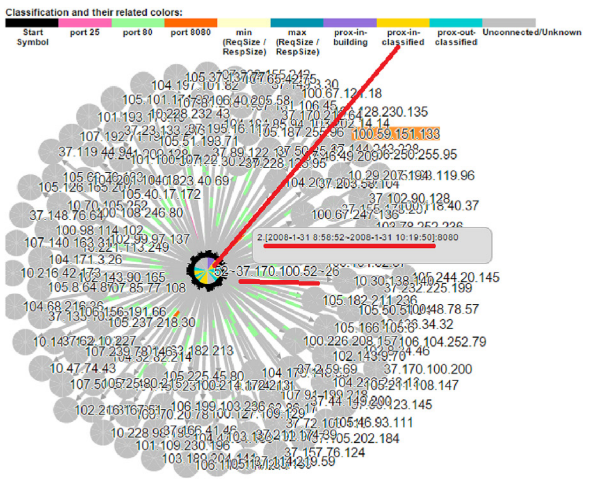
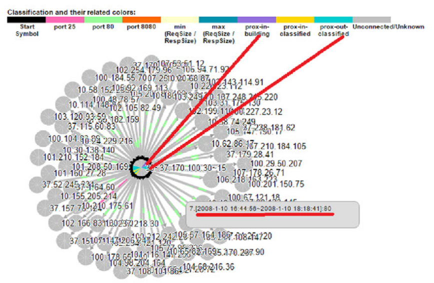
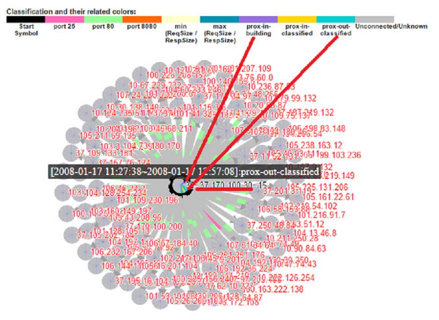

(b) 2008-1-31 8:58:52 to 2008-1-31 10:19:50, employee 52

**Fig. 17.** Machines are used to communicate with outside IP 100.59.151.133 while their owners were not in their offices but in the restricted area (yellow segment on the center node). (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

machines (37.170.100.8, 10, 13, 15, 16, 18, 20, 31, 32, 41, 52, and 56) are thus likely the machines that the spy in Embassy uses to contact with the criminal organization.

To further verify our suspicion and to identify who is the spy within the network, we analyzed each of the 12 machines' behaviors during their period of communication with IP 100.59.151.133. We selected location information in the employee proximity card log data as the node property. We found that all of these machines' communications with IP 100.59.151.133 are made without the machines' owners' presence in their office. Fig. 17 demonstrates two examples.


(a) On 01/10/2008, employee 30 went into the restricted room without prox in.


(b) On 01/17/2008, employee 30 went into the restricted room without prox in.

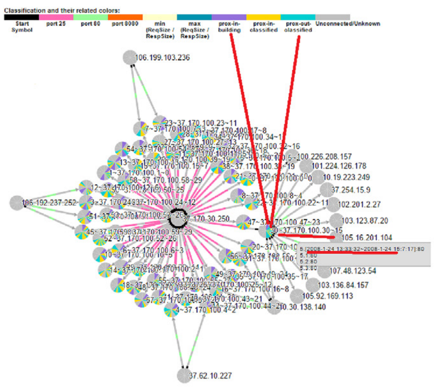

(c) On 01/24/2008, employee 30 went into the restricted room without prox in.

**Fig. 18.** Employee 30 went into the restricted rooms without prox in, which is against the policy.

From 2008-1-10 13:45:57 to 2008-1-10 15:11:11, employee 31 was in a restricted room (prox-in-classified, yellow node), but his allotted machine was used to communicate with outside IP 100.59.151.133. Same with employee 52, his machine was communicating with IP 100.59.151.133 while he was in other restricted area during 2008-1-31 8:58:52 and 2008-1-31 10:19:50. These patterns reminded us that the spy employee might want to hide his illegal behavior by using other employees' computers while these employees are out of their office.

Finally, we narrowed our focus to behaviors of each neighboring employee of the owners of the 12 victim machines. One suspicious employee draws our attention. While 12 machines connect with 100.59.151.133, employee 30 is the only active neighbor that badged into the building but did not badge into the restricted room. As seen in Fig. 18, there were three days, January 10, 17 and 24, when employee 30 went into the restricted room without badging in. This is against the policy that employees are required to prox into and out of the restricted area and not tailgate (enter without badging in by following a coworker who did badge in). All of the patterns found above suggest employee 30 is the spy employee.

In summary, through the case studies, we demonstrate the effectiveness of DNAV in detecting communication patterns in dynamic networks through temporal segmentations in nodes and edges. Although the datasets in the case studies mostly involve spatiotemporal information, DNAV exhibits its powerfulness in terms of generality since the node and edge properties can be any feature of interest, including but not limited to time, location, port number, IP address, user, content, etc. In case of larger datasets, the dynamic graph view alone is cumbersome to solve entire problem. We find it better to combine the dynamic graph view with other visualizations such as entropy statistical views that allow tiered knowledge ranging from overview to details.

## 7. Conclusion

The analysis of large-scale, complex and dynamic networks is important and has many applications. DNAV provides an alternative visual analytic approach to pattern recognition and anomaly detection in dynamic networks. Aggregation overviews and dynamic graph views, when combined, provide better scalability and insight than any individual perspective. In particular, dynamic graphs utilize spatiotemporal segmentations of graph components, such as nodes and links, to encode both topological and property dynamics associated with evolving networks. We demonstrated the usage of DNAV through case studies and suggest potential benefits provided by DNAV in understanding other types of dynamic networks. Future work is needed to explore alternative methods to improve scalability for larger networks.

## References

Abello, J., Hadlak, S., Schumann, H., Schulz, H.-J., 2014. A modular degree-of-interest specification for the visual analysis of large dynamic networks. IEEE Trans. Visual. Comput. Graph. 20 (3), 337–350.

Aggarwal, C., Subbian, K., 2014. Evolutionary network analysis: a survey. ACM Comput. Surv. 47 (1) 10:1–10:36.

Ahmed, R., Karypis, G., 2012. Algorithms for mining the evolution of conserved relational states in dynamic networks. Knowl. Inf. Syst. 33 (3), 603–630.

Ahn, J.-w., Taieb-Maimon, M., Sopan, A., Plaisant, C., Shneiderman, B., March 29-31 2011. Temporal visualization of social network dynamics: prototypes for nation of neighbors. In: Social Computing, Behavioral-cultural Modeling and Prediction, MD, USA, pp. 309–316.

Akoglu, L., Faloutsos, C., November 29-December 02 2010. Event detection in time series of mobile communication graphs. In: Army Science Conference, Orlando, FL, USA, pp. 77–79.

Alencar, A.B., Börner, K., Paulovich, F.V., de Oliveira, M.C.F., March 26-30 2012. Time-aware visualization of document collections. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing, New York, NY, USA, pp. 997–1004.

Archambault, D., Purchase, H.C., Pinaud, B., 2011. Animation, small multiples, and the effect of mental map preservation in dynamic graphs. IEEE Trans. Visual. Comput. Graph. 17 (4), 539–552.

Atzori, L., Iera, A., Morabito, G., October 2010. The internet of things: a survey. Comput. Network. 54 (15), 2787–2805.

Bach, B., Pietriga, E., Fekete, J.-D., 2014. GraphDiaries: animated transitions and temporal navigation for dynamic networks. IEEE Trans. Visual. Comput. Graph. 20 (5), 740–754.

Bach, B., Henry-Riche, N., Dwyer, T., Madhyastha, T., Fekete, J.-D., Grabowski, T., 2015. Small MultiPiles: piling time to explore temporal patterns in dynamic networks. Comput. Graph. Forum 34 (3), 31–40.

Beck, F., Burch, M., Vehlow, C., Diehl, S., Weiskopf, D., September 30-October 4 2012. Rapid serial visual presentation in dynamic graph visualization. In: IEEE Symposium on Visual Languages and Human-centric Computing, Innsbruck, Austria, pp. 185–192.

Bender-deMoll, S., McFarland, D.A., 2006. The art and science of dynamic network visualization. J. Soc. Struct. 7 (2), 1–38.

Beyer, D., Hassan, A.E., October 23-27 2006. Animated visualization of software history using Evolution Storyboards,. In: Proceedings of the 13th Working Conference on Reverse Engineering, Benevento, Italy, pp. 199–210.

Bilgin, C.C., Yener, B., 2006. Dynamic Network Evolution: Models, Clustering, Anomaly Detection. Tech. Rep. Rensselaer Polytechnic Institute, Troy NY.

Burch, M., Weiskopf, D., August 05-08 2014. A flip-book of edge-splatted small multiples for visualizing dynamic graphs. In: Proceedings of the 7th International Symposium on Visual Information Communication and Interaction, New York, NY, USA 29:29–29:38.

Burch, M., Diehl, S., 2008. TimeRadarTrees: visualizing dynamic compound digraphs. Comput. Graph. Forum 27 (3), 823–830.

Burch, M., Höferlin, M., Weiskopf, D., July 13-15 2011. Layered TimeRadarTrees. In: Proceedings of the 15th International Conference on Information Visualisation, London, pp. 18–25.

Burch, M., Vehlow, C., Beck, F., Diehl, S., Weiskopf, D., 2011. Parallel Edge Splatting for scalable dynamic graph visualization. IEEE Trans. Visual. Comput. Graph. 17 (12), 2344–2353.

Carley, K.M., 2014. Ora: a toolkit for dynamic network analysis and visualization. In: Encyclopedia of Social Network Analysis and Mining, pp. 1219–1228.

Federico, P., Aigner, W., Miksch, S., Windhager, F., Zenk, L., September 07-09 2011. A visual analytics approach to dynamic social networks. In: Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies, New York, NY, USA 47:1–47:8.

Fu, Y., Yan, Z., Cao, J., Koné, O., Cao, X., May 2017. An automata based intrusion detection method for internet of things. Mobile Inf. Syst. 2017 (1750637), 1–13.

Grabowicz, P.A., Aiello, L.M., Menczer, F., 2014. Fast filtering and animation of large dynamic networks. EPJ Data Sci. 3 (1), 1–16.

Grinstein, G., Scholtz, J., Whiting, M., Plaisant, C., October 12-13 2009. Vast 2009 challenge: an insider threat,. In: IEEE Symposium on Visual Analytics Science and Technology (VAST), Atlantic City, NJ, pp. 243–244.

Han, K., Zhang, T., Liao, Q., 2014. Data Stream Mining Based Dynamic Link Anomaly Analysis Using Paired Sliding Time Window Data. DTIC Document, Tech. Rep.

He, L., Yan, Z., Atiquzzaman, M., 2018. Lte/lte-a network security data collection and analysis for security measurement: a survey. IEEE Access 6, 4220–4242.

Itoh, M., Toyoda, M., Kitsuregawa, M., July 26-29 2010. An interactive visualization framework for time-series of web graphs in a 3D environment. In: Proceedings of the 14th International Conference on Information Visualisation, London, pp. 54–60.

Itoh, M., Yoshinaga, N., Toyoda, M., Kitsuregawa, M., Febuary 28-March 2 2012. "Analysis and visualization of temporal changes in bloggers' activities and interests. In: Proceeding of the 2012 IEEE Pacific Visualization Symposium, Songdo, South Korea, pp. 57–64.

Jin, R., McCallen, S., Almaas, E., October 28-31 2007. Trend motif: a graph mining approach for analysis of dynamic complex networks. In: Seventh IEEE International Conference on Data Mining, Omaha, NE, pp. 541–546.

Kang, H., Getoor, L., Singh, L., 2007. Visual analysis of dynamic group membership in temporal social networks. ACM SIGKDD Explor. Newslett. 9 (2), 13–21.

Koytek, P., Perin, C., Vermeulen, J., André, E., Carpendale, S., January 2018. Mybrush: brushing and linking with personal agency. IEEE Trans. Visual. Comput. Graph. 24 (1), 605–615.

Lahiri, M., Berger-Wolf, T.Y., December 15-19 2008. Mining periodic behavior in dynamic social networks. In: Eighth IEEE International Conference on Data Mining, Pisa, Italy, pp. 373–382.

Li, T., Liao, Q., July 5-7 2016. Dynamic networks analysis and visualization through spatiotemporal link segmentation. In: Proceedings of the IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), Chengdu, China, pp. 209–214.

Li, G., Yan, Z., Fu, Y., Chen, H., May 2018. Data fusion for network intrusion detection: a review. Secur. Commun. Network. 2018 (8210614), 1–16.

Liben-Nowell, D., Kleinberg, J., 2007. The link-prediction problem for social networks. J. Am. Soc. Inf. Sci. Technol. 58 (7), 1019–1031.

Lin, Y.-R., Chi, Y., Zhu, S., Sundaram, H., Tseng, B.L., 2009. Analyzing communities and their evolutions in dynamic social networks. ACM Trans. Knowl. Discov. Data 3 (2) 8:1–8:31.

Lin, H., Yan, Z., Chen, Y., Zhang, L., March 2018. A survey on network security-related data collection technologies. IEEE Access 6, 18345–18365.

Liu, G., Yan, Z., Pedrycz, W., March 2018. Data collection for attack detection and security measurement in mobile ad hoc networks: a survey. J. Netw. Comput. Appl. 105, 105–122.
Oline, A., Reiners, D., October 26 2005. Exploring three-dimensional visualization for intrusion detection. In: IEEE Workshop on Visualization for Computer Security, Minneapolis, Minnesota, pp. 113–120.
Pupyrev, S., Tikhonov, A., November 29-December 1 2010. Analyzing conversations with dynamic graph visualization. In: 10th International Conference on Intelligent Systems Design and Applications, Cairo, Egypt, pp. 748–753.
Ranshous, S., Shen, S., Koutra, D., Harenberg, S., Faloutsos, C., Samatova, N.F., 2015. Anomaly detection in dynamic networks: a survey. Wiley Interdisciplinary Reviews: Comput. Stat. 7 (3), 223–247.
Takaffoli, M., Rabbany, R., Zaïane, O.R., August 25-28 2013. Incremental local community identification in dynamic social networks. In: Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, New York, NY, USA, pp. 90–94.
Tong, H., Papadimitriou, S., Sun, J., Yu, P.S., Faloutsos, C., August 24-27 2008. Colibri: fast mining of large static and dynamic graphs. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, pp. 686–694.
van den Elzen, S., Holten, D., Blaas, J., van Wijk, J.J., 2014. Dynamic network visualization withextended massive sequence views. IEEE Trans. Visual. Comput. Graph. 20 (8), 1087–1099.
van den Elzen, S., Holten, D., Blaas, J., van Wijk, J.J., 2016. Reducing snapshots to points: a visual analytics approach to dynamic network exploration. IEEE Trans. Visual. Comput. Graph. 22 (1), 1–10.
Wagner, A., Plattner, B., June 13-15 2005. Entropy based worm and anomaly detection in fast ip networks. In: 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise, pp. 172–177.
Whiting, M., Cook, K., Grinstein, G., Fallon, J., Liggett, K., Staheli, D., Crouser, J., October 25-30 2015. Vast challenge 2015: mayhem at dinofun world,. In: IEEE Conference on Visual Analytics Science and Technology (VAST), Chicago, IL, pp. 113–118.
Zhang, L., Yan, Zheng, Kantola, R., July 2017. Privacy-preserving trust management for unwanted traffic control. Future Generat. Comput. Syst. 72, 305–318.
Zhou, D., Yan, Z., Fu, Y., Yao, Z., August 2018. A survey on network data collection. J. Netw. Comput. Appl. 116, 9–23.

**Qi Liao** is an Associate Professor of Computer Science at Central Michigan University. He received his M.S. and Ph.D. in Computer Science and Engineering (CSE) from the University of Notre Dame. He graduated summa cum laude with a B.S. and departmental distinction in Computer Science from Hartwick College, New York, with a minor concentration in Mathematics. He is a lifetime member of Kappa Mu Epsilon, Upsilon Pi Epsilon, and Tau Beta Pi.

Dr. Liao's research interests include computer security, anomaly detection, machine learning, visual analytics, and economics/game theory at the intersection of network usage and cybersecurity. He received best papers at USENIX and IEEE conferences, Emerald Literati Awards for Excellence-Outstanding Paper for Information and Computer Security, IEEE VAST Challenge Award, 2nd place in National Security Innovation Competition, and Center for Research Computing Award for Computational Sciences and Visualization. Dr. Liao received the 2015 CMU College of Science & Technology Award for Outstanding Research.

Dr. Liao has served on NSF panels, conference co-chairs, technical program committees, journal editorial boards. Dr. Liao was a visiting scientist at IBM Research in 2010; the 2014 American Society for Engineering Education (ASEE) Fellow for United States Air Force Research Laboratory; and a guest research faculty at Argonne National Laboratory in 2018.

**Ting Li** graduated from Central Michigan University, Michigan, USA with a M.S. degree in Computer Science. She received her Bachelor's degree in Network Engineering from Hubei University of Economics, Hubei, China. She is currently a Data Access Engineer at Realtor.com. Her research is mainly focused on Data Visualization, Network Security and Big Data Processing.

**Benjamin Blakely** is a cyber security researcher at Argonne National Laboratory. Previously, he has held positions in the private, public, and education sectors, and built an information security program to support growth of a cloud software startup through its initial public offering into the thousands of corporate customers. He earned his PhD and BS degrees in Computer Engineering from Iowa State University, with minors in psychology and political science. He holds the Certified Information Systems Security Professional (CISSP) and Certified Information Security Manager (CISM) certifications.