Streaming Data Visualization for Network Security

Huilian Sophie Qiu
Advisors: Walter Willinger, Jennifer Rexford

Abstract

The emergence of streaming data or "data in motion" has motivated the development of new "streaming" algorithms that provide up-to-date answers to continuous queries; that is, queries that are issued once and then run continuously as new data streams in. For example, in the context of network traffic management, continuous queries over streaming Netflow data may be used to detect anomalies in the network as they happen (e.g., performance degradation, onset of an attack). One of the most popular approaches for detecting unusual patterns in the network is frequent itemset mining (FIM). Answers produced by many FIM algorithms are often high-dimensional and packed with rich information. As the rate of data arrival may be rapid, interpreting the output in real time can be challenging. The main objective of this thesis is to introduce a new visualization method that can visualize the continuous stream of answers produced by existing streaming algorithms in an intuitive and meaningful manner. The visualization method is designed independent of the choice of FIM algorithms. It is able to capture frequency of each itemset, different relationship between network traffic attributes, and the changes in frequent itemsets over time. Ultimately, users should be able to leverage this visualization to respond to an ongoing attack in real time.

1 Introduction

As the complexity of computer network has grown, the quest for real-time or close-to real-time solutions for managing these networks has remained elusive. For example, in the area of network security, such solutions would enable the timely detection of the onsets of different types of network attacks followed by swift and effective mitigative actions. With respect to network performance, the sought-after solution would be recognizant of user-experienced service degredations as they happen and instruct the network to perform corrective steps in a timely and purposeful manner. Whether the concern is network security or network performance, the development of new solutions to make (close-to) real-time network management a reality relies critically on our ability to capture, process, and analyze large quantities of high-quality network traffic measurements. However, not only does the sheer volume of traffic that traverses many of today's large backbones, Internet exchange points (IXP), and interconnects create serious challenges, but at Gbps to Tbps link speeds, the velocity of the collected data is such that any attempt at developing (close-to) real-time solutions has to treat the measurements as streaming data where one pass over the data is all that can be afforded. Note that such *streaming data* is representative of many recent "big data" occurrences in numerous different application domains (e.g., smart cities, IOT).

Since popular approaches that rely on offline batch processing of such streaming data are counterproductive in view of the desired real-time nature of the envisioned solutions, the streaming data model has motivated the development of a large number of different queries, in essence, a streaming algorithm simply transforms input in the form of a continuous data stream into a continuous stream of output data that consists of up-to-date answers to the posed query. Unfortunately, the output data generated by most streaming algorithms is typically only amenable for manual inspection which makes processing these answers and extracting detailed information from them a time-consuming and often tedious endeavor. In this thesis, we present a visualization design that can automatically process the output produced by certain streaming data algorithms and display the information in an intuitive and meaningful way. In particular, we focus in this thesis on a class of algorithms called FIM algorithms for

streaming data, a generalization of the well-known algorithms for finding frequent items (e.g., top-k) in streaming data.

1.1 Frequent itemset mining

FIM is often explained using a *market-basket model*. This model of data is used to describe a many-to-many relationship between two kinds of objects, *items* in the market and *baskets*, or *transactions*. An *itemset* is a set of items that may appear in many transactions. The *support* of an itemset I is defined as the number of transactions for which I is a subset. In other words, the support of itemset I is the percentage of transactions that contain I. An itemset is considered to be *frequent* if its support is higher than a support threshold.

In the context of network traffic, an attribute value, such as an IP address, a port number, or a protocal type, is an item. Each individual network traffic record is called a transaction. As in the market-basket model, a transaction can contain several attribute values. For example, a single record in Netflow data consists of values of many attributes, such as source IP address, destination IP address, port number, protocol, packet size, etc. An itemset may contain values of some or all attributes in a transaction. It is considered to be a frequent itemset if the joint appearance of all its items is above a threshold. *Frequent items*, or sometimes called *heavy hitter* can be considered as a special case of frequent itemset whose number of items is one.

1.2 Requirements for the visualization

A number of different FIM algorithms for streaming data have been developed in the past two decades (e.g. see [9] and references therein). While frequent itemsets are in general costly (i.e., memory, CPU) to find in real time, the output of existing FIM algorithms for streaming data contains usually both very detailed information and useful meta-data. At the same time, processing this data and unpacking the obtained information typically requires manual inspection and analysis. Therefore, one of the requirements of our visualization method is to automate the unpacking of the output and then display it in an intuitive way so that users can easily identify patterns in the data and observe how they change over time. The visualization method should also be able to show the sizes of the different frequent itemsets, exploit the relationships between different network traffic attributes, and depict how frequent itemsets change over time as new data streams in. Ultimately, users should be able to leverage this visualization to respond to an ongoing attack in real time.

There also exist many variations of the FIM approach. A special case is the class of frequent item mining algorithms. These algorithms can identify what attribute values appear frequently. For example, an IP address is a frequent item because it may be hosting a search engine and receiving many queries. However, frequent items only reflect a single attribute in the network data and only reveal items that are globally popular. Therefore the output of frequent item algorithms for streaming data may not contain enough information to identify traffic pattern of interest. Among the algorithms that are able to identify slightly richer patterns than the frequent item algorithms are the hierarchical heavy hitter and correlated heavy hitter algorithms. Intuitively, the hierarchical heavy hitter algorithm takes the result of the frequent item algorithm applied to, say the IP address which has a strong hierarchical structure [11] and performs aggregation on various levels. Correlated heavy hitters are interested in items that are locally popular. For example, for a globally popular destination machine, the correlated heavy hitter algorithm identifies source machines that contribute to a large portion of these connections [8]. Given the wide variety of FIM algorithms, our visualization method should be not only flexible enough to represent different types of relationship among the data, but also independent of the streaming algorithm that produces the output data.

1.3 Proposed visualization method

The contribution of this thesis is to show how the output generated by different FIM streaming algorithms can be displayed in an intuitive and meaningful manner using our visualization method. Our visualization method takes the output of a chosen streaming algorithm as input. The generated diagram is ever-growing towards the right when new output is produced by the algorithm as a result of new input data streaming in. Each new stream of results is represented as a column of nodes. For each new output

data, a column of nodes is appended to the right of the existing diagram. Each node corresponds to an item. Nodes in two adjacent columns are connected by flows (equivalent to "alluviums" in alluviam diagram) that represent individual itemsets. The height of a flow is proportional to the support of the itemset. Nodes connected to the same flow belong to the same itemset. We also provide an interactive interface that allows users to further inquire into a particular itemset or an item.

We introduce our method in more details in Section 2 using the example of visualizing frequent itemsets in Netflow data. Section 3 shows how our method can be used with different algorithms for different network traffic scenarios. In Section 4 we relate our efforts to previous work and discuss future work in Section 5.

2 Frequent Itemset Visualization Method

This visualization model takes the continuous output stream from FIM algorithms as input and generate diagrams in real time. The way that the model works makes no assumption on the choice of streaming data algorithm. However, the attributes displayed in the diagram depend on the algorithm. Figure 1 is a snapshot of a short segment of a diagram generated using Borgelt's split and merge algorithm, SaM [1], which is implemented by Barthelemy Dagenais in Python. Source code was found on his github repository [2]. In this diagram, we chose to examine pairs of source IP and destination IP that jointly appear to be frequent. In this section, we will use Figure 1 as an example to first describe basic components in our design and the interactive interface implemented in JavaScript using D3 library. Then we use the same example to show how an output from a FIM algorithm can be effectively displaying using our method.

2.1 Basic components

The entire scheme is unbounded and ever growing towards the right. It is divided into successive columns. Each newly arrival stream of data occupies one column. Each column contains a number of nodes. Associated time-stamp is printed below each column. When new output streams in, a new column of nodes can be easily appended on the right. Four of such columns are shown in Figure 1.

Each column can be further divided vertically into sub-columns. In Figure 1, each column has two sub-columns. Each sub-column corresponds to one attribute. Which attribute does this sub-column represent is decided by the user and the chosen algorithm. The relationship between sub-columns can be hierarchical. For example, one column can be IP addresses and another IP prefixes. It can also be associative with one column being source IP the other being destination IP. It is also possible to have more than two sub-columns and map each one of them to an attribute in the itemset. We call the left most sub-column as the first dimension sub-column, and the next as the second dimension sub-column, so on and so forth. Sub-columns are distinguished with different brightness of grey. All nodes within the same sub-column have the same color. Using the color grey is to minimize the distraction of various hues, which are needed for flows to distinguish itemsets.

Each sub-column is divided into individual rectangle nodes horizontally. Each individual node is associated with an item. The value of an item is written on the node. For example, the node in the upper left corner has value {41.43.163.74}, which, in our case, is a destination IP. The heights of nodes will be discussed later.

Inspired by alluvial diagram, which is good at tracing network changes over time, adjacent columns are connected with a number of flows, each of which represents an itemset. Different itemsets are distinguished by different colors. The same itemset appears in consecutive time-stamps uses the same color to enable users to trace its changes. Nodes that are connected by the same flow belong to the same itemset. In Figure 1, one flow directly connects to one node at each ends. Note that each window is divided into two sub-columns. Therefore, items in the itemset connected by the yellow flow is {41.43.195.44} in the first dimension sub-column and {216.84.74.70 41.43.195.44} in the second dimension sub-column. In other words, to see what items are contained in a frequent itemset, one should look at nodes from all sub-columns, unless otherwise specified, for example, one of the sub-columns may be aggregated data instead of real items.

Sometimes, two flows may merge to one node, such as the green and blue flows at timestamp

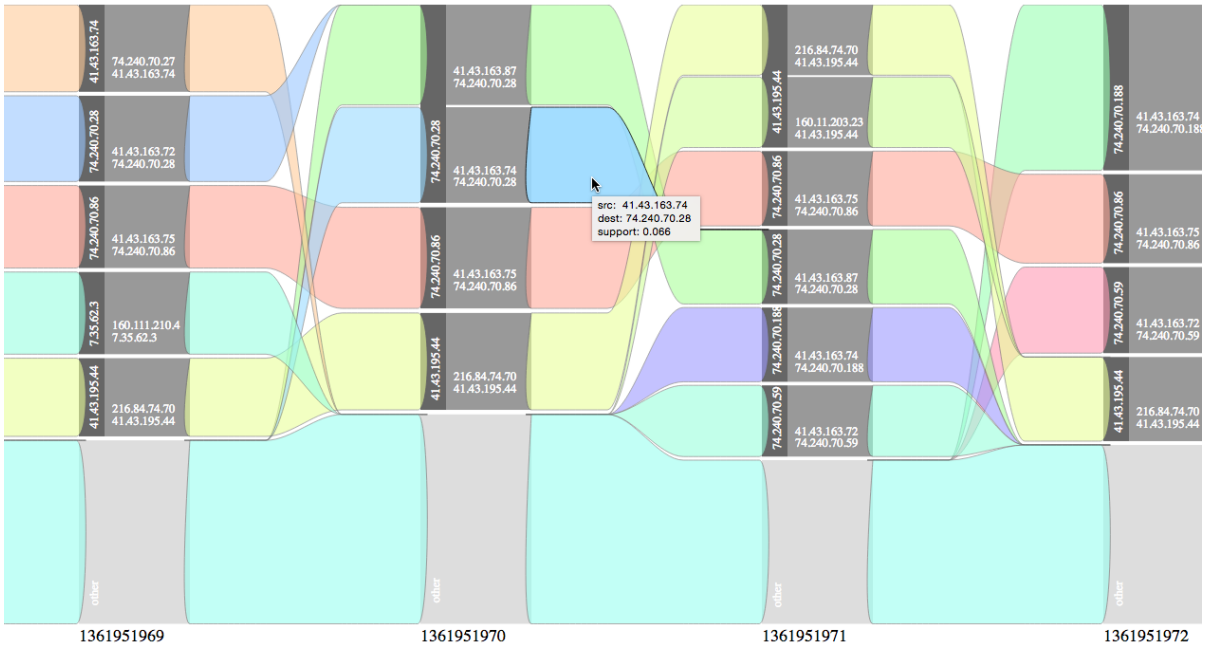

Figure 1: Visaulizing results from a frequent itemsets mining algorithm. Grouping results with the same destination IP address.

1361951970. This feature is inspired by Sankey diagram, which is designed to show network structure changes. The merging of flows means that these two itemsets contain a common item. In the case of green and blue flows, they both contain the item {74.240.70.28}.

The height of each flow is proportional to the log value of its support. Because usually frequent itemsets may only take up a very small fraction of the dataset, using log values instead of exact values can make each flow have reasonable height. A light grey node's height is the sum of all the flows that are connected to it. The height of nodes in each dimension is the aggregated sum of correspondence nodes from the lower dimension. The ordering of nodes starts from the first dimension. All but the bottom ones are ordered by height in descending order. Within each nodes in the first dimension, second dimension nodes are sorted by height again in descending order.

We implemented our visualization method using JavaScript and its D3 library. A demo can be found on http://cs.princeton.edu/~hqiu. There are two features that we want to mention here. First is the animation. When a new stream of data arrives and a new column appends on the right, we make the flows gradually expanding from the previous column as if they are "flowing" into the current column. We added this animation just to mimic the streaming nature of network traffic data. The other feature is the interactive interface. When users hover their mouse over a node or a flow, they will be able to see more information about this item or itemset, including values and support. For example, since the large node at the bottom represents all transactions, its support is 100%. From here, we can tell that, at time-stamp 1361951970, the blue flow's height is roughly 1/3 of the bottom node's. Hovering our mouse to one of the blue flows, we find out that its support is 6%.

2.2 Illustration with Netflow data

Let us now explain how does Figure 1 reflect the result of a FIM algorithm. This diagram is generated by running a FIM algorithm on pairs of source and destination IP addresses. Therefore, the frequent itemsets here can also be considered as heavy hitters.

As we mentioned earlier, each column can be further divided up vertically into sub-columns. The attributes shown in each column is up the users. In Figure 1, we decided to display the item in the second dimension sub-column in lighter grey color. We call these nodes item nodes. The largest node at the bottom represents all transactions during this period of them. We then decided to aggregate these heavy

hitters by their destination IP addresses and shown these aggregation nodes in the first dimension sub-column with darker grey color.

Each flow in Figure 1 represents a frequent itemset, which, in this particular case, contains only one item, a connection between two IP addresses. Tracing the yellow flow from time-stamp 1361951969, we can see that this itemset contains the item {216.84.74.70 41.43.195.44}. The aggregated destination IP address is {41.43.195.44}. At time-stamp 1361951971, we see that the yellow flow still connects to the same item node and aggregation node, but there is another flow connecting to the same aggregation node. This is because at time-stamp 1361951971, two connections, {216.84.74.70 41.43.195.44} and {160.11.203.23 41.43.195.44}, share the same destination IP. We can also see that the height of the yellow flow changes at timestamp 1361951971. Note that the height corresponds to the log value of the itemset's support within at the current time-stamp. The change in height here does not necessarily mean that the exact number of transactions containing this particular itemset has changed. The change may suggest a change in the support. It may also be affected by the increasing number of frequent itemsets.

3 Application Examples

In this section, we demonstrate that out visualization design is capable of visualizing outputs from different FIM algorithms with different choices of attributes. We applied our visualization method on two sets of data: a Netflow data collected at University of Oregon in Feburary 2013 and a DNS record data collected at Princeton University in Janurary 2017.

3.1 Frequent itemsets of different sizes

While the frequent itemsets used in Figure 1 contain only one item, which is a connection between two IP addresses, it is often the case that frequent itemsets may contain different numbers of items. Figure 2 shows how this diagram can visualize frequent itemsets of different sizes.

Attributes shown in Figure 2 are source IP, destination IP, destination port number, and an aggregated source IP prefix. Here, each column is divided into three sub-columns. The right most sub-column shows the destination IP; the middle sub-column shows the source IP; the left most column shows the prefix of source IPs. Numbers shown on some of the flows are port numbers. Note that not all frequent itemsets contain all three of these attributes. The missing attributes are represented by {-}. Port numbers could also be shown by adding another sub-column to the right. For example, the pink flow contains items {248.207.38.29} as destination IP shown in light grey node and {53} as port number shown on the flow. Hovering the mouse over one of the flows as shown in Figure 2, we can see that the bright green flow is a frequent itemset with only one item, which is the destination port number {443}. We see that this flow does not show up in the previous window, meaning that the itemset {443} has newly become frequent at time-stamp 1361951968 and its support is 5.3%. Moving our eyes to the next column, we can see that it persists to be frequent and the support remains at the same magnitude.

This diagram contains two levels of aggregation, aggregating itemsets with the same source IP and aggregating source IPs by prefix. Aggregations are done by the visualization method and users can choose to aggregate attributes in other ways. We can see that at time-stamp 1361951968, two itemsets {248.207.38.29} and {248.207.48.41} had the same source IP prefix in common.

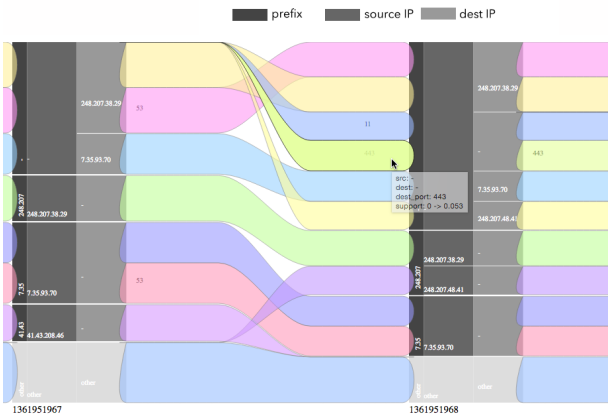

Figure 2: Visualization of frequent itemsets of various sizes

3.2 Correlated heavy hitters

There are many discussions on the trade-offs between frequent itemsets, the information contained

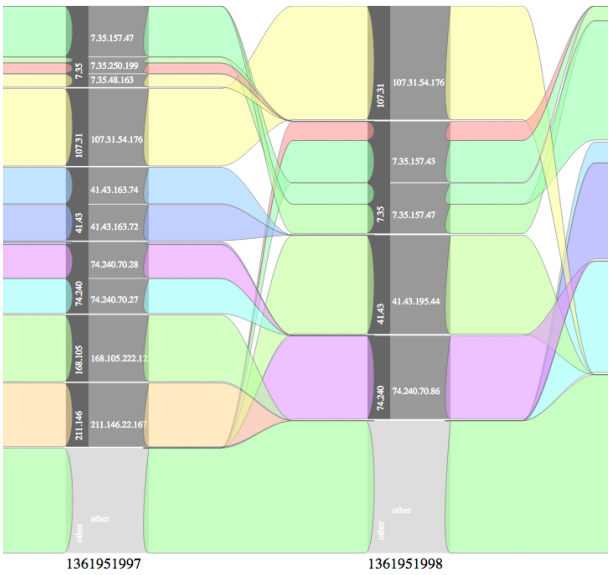

Figure 3: Visualization of correclated heavy hitters

by which may be too rich and too costly to find, and frequent items, which may not reveal enough information. Several models fall between the spectrum of frequent itemsets and frequent items have been proposed. Many of them have to do with aggregating data based on correlations between items or identifying conditionally frequent items.

The diagram in Figure 3 uses the idea of *correlated heavy hitter* proposed by Lahiri and Tirthapura in [6]. The correlated heavy hitter algorithm concerns not only on the support of a single attribute, but also on the correlated support of an attribute. For example, suppose an itemset has two items, p (**p**arent) and c (**c**hildren). If the support of the item p is above a certain threshold and the correlated support $Pr[c|p]$ of c is higher than a threshold, c is considered to be a correlated heavy hitter.

Figure 3 shows correlated heavy hitters among triples of attribute values (*destination IP prefix, destination IP, source IP*). In this figure, dark grey nodes represent destination IP prefix, lighter grey nodes in second sub-column represent destination IP and flows represent source IP. At the bottom of the column, a node {other} is added to represent all the other transactions.

In the context of correlated heavy hitter algorithms, items within an itemset have hierarchical relationship. It is required that the first attribute, destination IP prefix in our example , should be globally popular, i.e., its support should exceed some threshold. The support of the second attribute, destination IP, is the percentage of transactions that contain itself among all the transactions containing its predecessor. Therefore, unlike diagrams for frequent itemsets in Figure 1 and Figure 2 where the a node's height is the aggregated sum over its children in the lower dimension, we decided to first determine the height of the first dimension sub-column, which has the highest hierarchy, then let the height of nodes in the next sub-column be proportional to its correlated support. The heights of flows are also proportional to their correlated support.

The height of the prefix nodes and the {other} node is proportional to the log value of their support. Observe that transactions containing values in the prefix nodes and those in the {other} node partition the dataset. Therefore, the height of the {other} node no longer serves as a reference for the size of entire dataset in the current window. Rather, it shows the support of all non-frequent items in the dataset. It might make sense to linearly map the exact value of support instead of its log value to the height. The reason we chose to use the log value instead of the exact value is the same as we discussed before: popular items may only take up a small fraction. The trade-off here is that we decided to sacrifice the intuitive level in order to display all items in reasonable sizes.

3.3 DDoS attack pattern

Figure 4 shows how our visualization method can capture potential DDoS attacks. Here we manually in-planted a DDoS attack pattern in our data. Figure 4 is a short segment of the entire diagram. The second dimension sub-column with lighter grey nodes represents destination IPs. The first dimension sub-column with darker grey nodes are aggregated data on destination IPs' prefix. Since this diagram uses the output generated by a FIM algorithm, the height of each flow is proportional to its support and the {other} node at the bottom of each column represents all transactions within the same time frame. It is apparent from the diagram that the machine with IP address {74.240.70.28} was receiving a huge amount of traffic during at least within these two seconds. Users can then inquire more information on these connections.

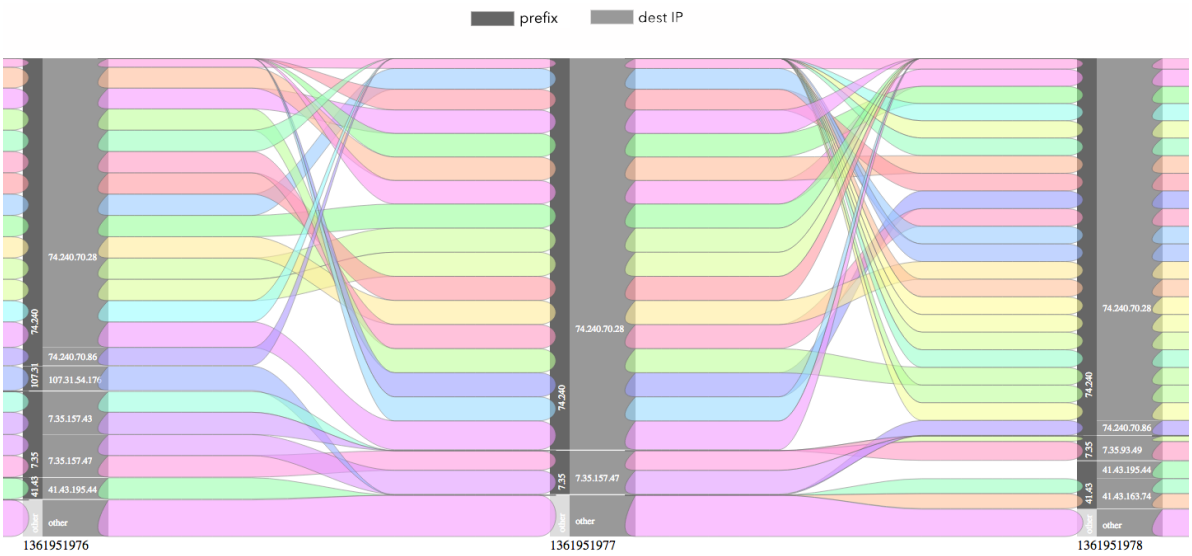

Figure 4: Visualization of frequent itemsets showing potential DDoS attack

3.4 Port scan attack pattern

We also injected a port scan attack pattern into our dataset (Figure 5). Basically, it shows up the same as diagrams for DDoS attack. Since port numbers are short, we decided to display it on each flow so that users do not need to hover their mouse over each individual flow to see what ports are being scanned. We did not do this for DDoS attack because IP addresses can be long, especially IPv6. Showing all source IP can make the diagram messy and difficult to parse.

3.5 DNS-specific pattern

Our visualization method can also be used for datasets other than Netflow data. Figure 6 shows how frequent itemsets of DNS data can be visualized using the same method. The first dimension sub-column represents source IP prefixes; nodes in the second dimension sub-column are source IPs; nodes in the third dimension sub-column are destination IPs. From the diagram, we can see that at time 1361951968, {exchange.Princeton.EDU} became a frequent DNS query name, which is expected as the DNS data was capture at Princeton University.

4 Related Work

Many visuliaztion schemes for frequent itemsets have been proposed previously. One large family of such schemes relies on frequent pattern tree (FP-Tree) algorithms proposed by Han et al. [4]. FP-Tree consists of a set of item prefix sub-trees and shows hierarchical relationships in the dataset. FP-Viz by Keim et al. utilizes such relationships and visualizes outputs from FP-Tree algorithms in a Radial Hierarchical Layout [5]. The root of a FP-Tree, which has a value *null* is placed by a circle in the middle of the visualization. Each segment in the diagrams represents a node in the FP-Tree. The frequency of an item decides the order of these circle segments within each level. Different colors are used to distinguish the support of each itemset with red associated with higher support and green with lower. Users can also choose a frequent item as root and generate a new diagram of items that frequently appear together with the root item. However, this scheme is not designed for streaming data. Each diagram is generated for a bounded dataset. Therefore, it does not reveal how frequent itemsets change over time.

Another visualization method is frequent patterns visual analytic tool (FpVAT) by Leung et al. It consists of two modules: raw data visualization (RdViz) and frequent pattern visualization (FpViz). RdViz displays raw data (i.e. the input data) in a two-dimensional diagram. The x-axis is the items and

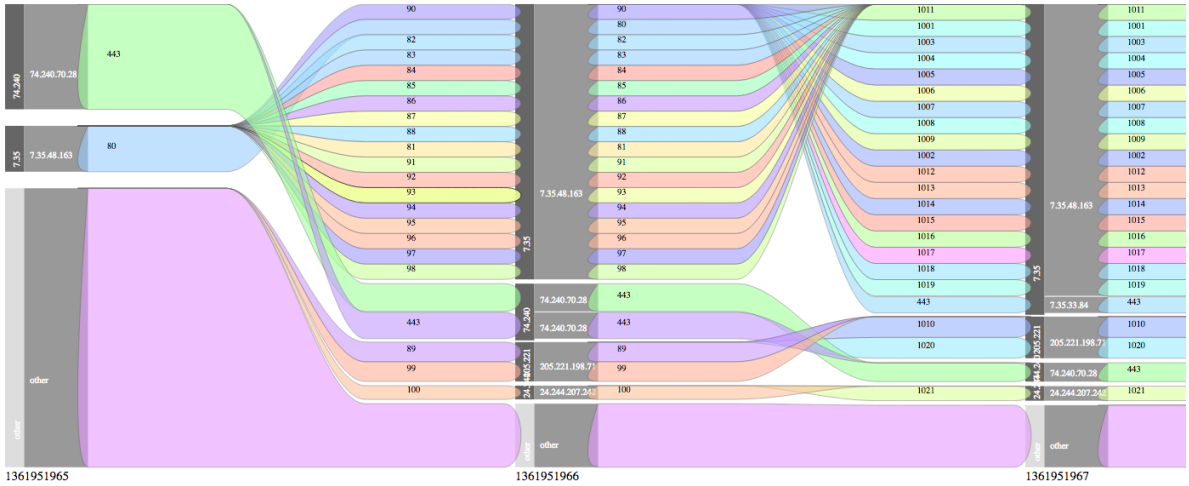

Figure 5: Visualization of frequent itemsets showing potential port scan attack

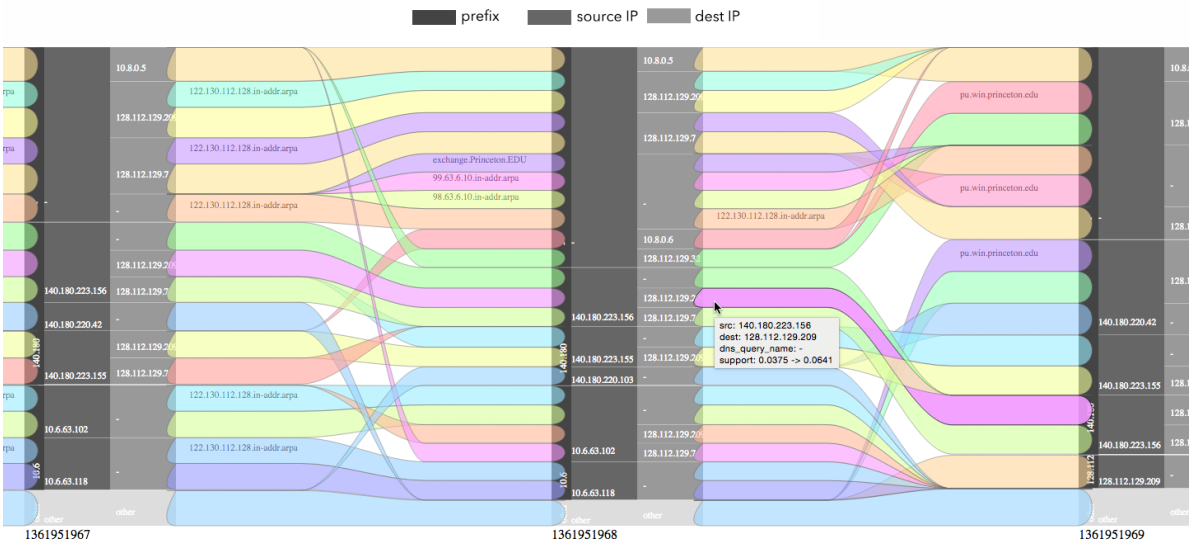

Figure 6: Visualization of frequent itemsets of DNS data

y-axis is the transactions. Each transaction is represented by a horizontal line connecting k filled circles, one for each item in the transaction. RdViz can reveal what transactions hold the same items. FpViz is similar to RdViz but is applied on frequent patterns from mining algorithms. The x-axis is the same as in RdViz. The y-axis is the frequency of an itemset [7]. Both FpVAT and FP-Viz show aggregations of some frequent itemsets on shared items, which is similar to the aggregation we have in our method. However, like FP-Viz, FpVAT is also designed for offline, bounded dataset and is not suitable for streaming data.

Glatz et al. proposed a method of hypergraphs in [3]. A hypergraph consists of three basic components: arrows, circles, and rectangles. Each rectangle corresponds to an attribute value, e.g. IP address, port number. Rectangles from the same frequent itemset all point the same circle, which shows the frequency of this itemset. A rectangle can belong to several frequent itemsets and thus can connect to several circles. This form of aggregation is designed for network data. However, the version they presented in [3] is static and applied on bounded data. They later release an animated version that can display how frequent itemset changes over time [12]. Graphs generated for consecutive time stamps place

the same items at the same place, allowing users to track what items stay frequent but may belong to different itemsets in the next time-stamp. The problem with their animated approach is that after a new diagram is generated for the new stream of data, the previous one disappears. Although users are able to see what items remain frequent from last time-stamp to present, it is hard for them to keep track of how an item or an itemset evolve in a longer period.

There exist two closely related diagrams, sankey diagram and alluvial diagram, that are designed for showing network changes. Sankey diagram was first created by Charles Joseph Minard to show the number of Napoleon's soldiers going to and back from Russia. Later it was used by Captain H Riall Sankey for energy flow of a steam engine in 1898. Although its first usage by Minard showed how an event changes over time, Sankey diagram is currently used more often for showing many-to-many mapping between two domains or the structure changes in a system. Therefore it lacks the sense of how the changes progress over time.

Alluvial diagram is designed to illustrate how the structure of network changes over time. In [10], Rosvall and Bergstrom use significance clustering method to cluster bootstrap network at different time-stamp. This is done by repeatedly sampling links in the network and clustering bootstrap network along the way. Comparing the clustered bootstrap network with the original network gives the degree of support that the data provide in assigning a node to each cluster. Significant clusters are those clustered together in at least 95% of the 1000 bootstrap networks. These records of significant clusters at each time stamp are then plotted using alluvial diagram. Each alluvium represents a cluster. The height of alluviums corresponds to the volume of the flow in the cluster. Different colors are assigned to different alluviums. The alluvial diagram they proposed is capable of tracing the history of network structure changes. They applied their method on data of changes in science and revealed how Neuroscience has gradually become an independent field of studies combining Neurology, Psychology, and Molecular & cell biology. Our work is more related to this approach. However, the diagram presented in [10] is not ideal for frequent itemset visualization, especially frequent itemsets of network data. Their diagram is capable of showing data of three dimension: time, clusters, and connections between clusters. However, network data may have higher dimension. For example, their diagram may be capable of showing popular destination machines over time and the sources of the traffic. Nevertheless, its current design does not allow one to show further details of these network activities, such as protocol or port number.

5 Future Work

One future direction can be extending the same visualization approach to persistent itemset mining and rare itemset mining. All diagrams in this report are generated using FIM algorithms. They demonstrated its ability of showing traffic patterns that appear frequently and may require attention. However, frequent itemset is not the only type of patterns we are interested in. Some attacks may be identify by looking at persistent itemsets or rare itemsets. Some attacks may disguise themselves by making fewer connections at each time-stamp but persisting for a long while. The way we use flows to represent the changes of itemsets as time passes enables users to easily tell what itemsets have stayed frequent over a period of time. However, we also need a way of showing what are the itemsets that have persistently existed for the longest period of time.

Most graphs we shown here do not reveal sufficient information about how two machines interact with each other. Two directions of the connection between a pair of machines may show up in different nodes and there is no indication of their relationship. Therefore, in future work, we may explore how we can incorporate the interactive aspect into our diagram. This can be helpful for detecting DoS attack by looking at incomplete three way handshake, where there is a huge discrepancy between the number of SYN and ACK.

While it is easy to trace how things progress, our current design does not provide an easy way of examining the history of the traffic. Users can scroll back in time to look at what happened before, but this is inefficient. As new data keeps arriving, the diagram grows rapidly. When the users go back and examine a segment of history, they will lose track of the current updates. It is desirable if a condensed view can be provided.

6 Conclusions

In this thesis, we introduced a visualization method that can take the output from a FIM or heavy hitter mining algorithm and display it in an intuitive way. We implemented our method in JavaScript and applied our method on outputs produced by several different algorithms. We also showed that our design is capable of revealing potential attack in real time.

References

[1] C. Borgelt. *Simple Algorithms for Frequent Item Set Mining*, pages 351–369. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-05179-1. doi: 10.1007/978-3-642-05179-1_16. URL http://dx.doi.org/10.1007/978-3-642-05179-1_16.

[2] B. Dagenais. pymining. https://github.com/bartdag/pymining, 2015.

[3] E. Glatz, S. Mavromatidis, B. Ager, and X. Dimitropoulos. Visualizing big network traffic data using frequent pattern mining and hypergraphs. *Computing*, 96(1):27–38, Jan. 2014. ISSN 0010-485X. doi: 10.1007/s00607-013-0282-8. URL http://dx.doi.org/10.1007/s00607-013-0282-8.

[4] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pages 1–12, New York, NY, USA, 2000. ACM. ISBN 1-58113-217-4. doi: 10.1145/342009.335372. URL http://doi.acm.org/10.1145/342009.335372.

[5] D. A. Keim, J. Schneidewind, and M. Sips. Fp-viz: Visual frequent pattern mining. In *InfoVis*, 2005.

[6] B. Lahiri and S. Tirthapura. Finding correlated heavy-hitters over data streams. In *2009 IEEE 28th International Performance Computing and Communications Conference*, pages 307–314, Dec 2009. doi: 10.1109/PCCC.2009.5403820.

[7] C. K.-S. Leung and C. L. Carmichael. Fpvat: a visual analytic tool for supporting frequent pattern mining. *ACM SIGKDD Explorations Newsletter*, 11 (2):39–48, 2010.

[8] K. Mirylenka, G. Cormode, T. Palpanas, and D. Srivastava. Conditional heavy hitters: Detecting interesting correlations in data streams. *The VLDB Journal*, 24(3):395–414, June 2015. ISSN 1066-8888. doi: 10.1007/s00778-015-0382-5. URL http://dx.doi.org/10.1007/s00778-015-0382-5.

[9] B. Mozafari, H. Thakkar, and C. Zaniolo. Verifying and mining frequent patterns from large windows over data streams. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ICDE '08, pages 179–188, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-1-4244-1836-7. doi: 10.1109/ICDE.2008.4497426. URL http://dx.doi.org/10.1109/ICDE.2008.4497426.

[10] M. Rosvall and C. Bergstrom. Mapping change in large networks. *PLoS ONE*, 5(1):e8694, 2010.

[11] D. Tong and V. Prasanna. High throughput hierarchical heavy hitter detection in data streams. In *Proceedings of the 2015 IEEE 22Nd International Conference on High Performance Computing (HiPC)*, HIPC '15, pages 224–233, Washington, DC, USA, 2015. IEEE Computer Society. ISBN 978-1-4673-8488-9. doi: 10.1109/HiPC.2015.30. URL http://dx.doi.org/10.1109/HiPC.2015.30.

[12] R. Vogt and P. Frick. Animated big data visualization. https://deniaz.github.io/animated-big-data/, 2015.