Treemap Visualization of Wireless Network Activity

Andrew Korzeniewski*
Michigan Technological University

Richard D. Pringle II†
Michigan Technological University

ABSTRACT

Network traffic monitoring of available and utilized bandwidth is important to regulate network usage and to allocate sufficient bandwidth to the hosts that need it. Common usages of network traffic monitoring are for traffic prioritization and determining hosts with unnecessary amounts of traffic. System administrators may want the convenience of seeing exactly who is utilizing what portion of the available network bandwidth. A treemap is an excellent visual aid for displaying massive amounts of categorical data into a limited visual area. This tool intercepts wireless network traffic to create a near real-time display of the amount of network bandwidth that each host is consuming.

1 INTRODUCTION

Using treemaps is a popular method for visualizing hierarchical data. They can effectively display the hierarchy by recursively dividing the display area into rectangles [1]. We liked the treemap concept as it allows an easy-to-read, first-glance view of massive amounts of categorical data. This is important for a monitoring tool because constant, dynamic network activity can be considered a large-scale dataset. As such, this type of viewing can be useful to occasionally verify network integrity.

Network traffic monitoring can be a difficult and resource-consuming, but necessary task. We thought that an improved visual paradigm for network monitoring would be beneficial to make analyzing network traffic faster and easier to read.

We wanted to explore the effectiveness of treemaps as a method to visualize network traffic data. Being familiar with networks, we thought that the ability to see who is using network bandwidth at a given time would be useful. This may be useful to determine bandwidth resource allocation across a given network to improve performance and reduce congestion. Furthermore, this type of visualization might aid in first response of network security attacks, such as dynamic denial of service (DDoS). This can be used to quickly determine which external services are most involved in the overall activity of the network.

2 OVERVIEW

2.1 Motivation

We ultimately wanted to create a tool to easily explore and view network traffic data. One attribute to ease the viewing of a changing medium is stability. The importance of stability is covered more in Section 5.1. We also wanted to add functionality to make it easier to view the important aspects of network data which are bandwidth usage and the ratio of upload/download bandwidth. Finally, in order to use this program for monitoring purposes, we wanted to make it real-time.

*e-mail: apkorzen@mtu.edu
†e-mail: rdpringl@mtu.edu

2.2 Implementation

A perl script is used to drive the program. It first calls the Unix utility tcpdump for network packet gathering. It hands off the data, retrieved from tcpdump, to a java program for grouping and display. Thus, the tool is broken up into the following work flow steps: packet gathering, data grouping, and treemap construction and visualization.

3 PACKET GATHERING AND PARSING

A wireless adapter is first configured to operate in monitoring mode. This allows the adapter to be disassociated from any wireless networks, which enables passive viewing of any wireless data packets on a given frequency in range of the adapter.

Tcpdump is used to sniff all available wireless traffic within range, disassociated from any specific network. The information that is available is interpreted by the perl script and only the relevant pieces of information are then included in the output that will be handed off to the java program. This output data includes a raw unix timestamp, the source IP address, and the destination IP address. The perl script filters out all other information associated with each packet, however, adding port numbers to the information or possibly other packet information requires trivial changes to the perl script and involves changes to the java program. Changing the output data to include more information would allow the implementation of more filtering options, but this is discussed more in Section 6.

4 DATA GROUPING

4.1 Design

The java program reads from standard input to interpret the parsed data from tcpdump. Since our goals included accepting real-time input, this portion of the program runs in its own thread. Its purpose is to fill a buffer with standard input data. When the buffer becomes full or if the event dispatch thread requests data, the data is then pushed into the main java program and analyzed so that it can be added to the data structure.

The basic data structure we implement can be thought of in terms of a dialog between people. Figure 1 shows the idea behind our data structure. One host starts a conversation with another host, where a given host may have conversations taking place simultaneously with any number of other hosts.

A pair of hosts may only have at most one conversation between them, and each conversation consists of one or more sentences. Each host keeps a list of its conversations with other hosts. Each sentence object carries a time window value and sent and received traffic counts associated with it. This is shown in Figure 2.

A conversation object in practice can be thought of as a symmetric relation, where given two hosts, A and B, and a conversation between them, A's sent traffic to B is going to be B's received traffic from A. To get the accumulated traffic counts for a host, the host will query all of its conversations for their respective totals within a user-specified time window based on a start and stop time. Upon receiving a start and stop time from a host in a request for traffic counts, each conversation uses the start and stop times to filter out irrelevant sentences. This is also shown in Figure 2.

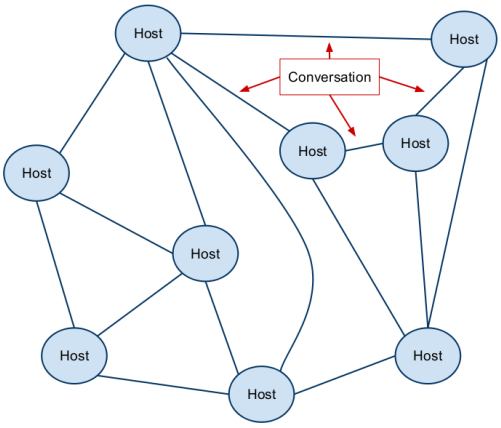

Figure 1: High-level design of data structure. Vertices are unique Host objects and edges are unique Conversation objects

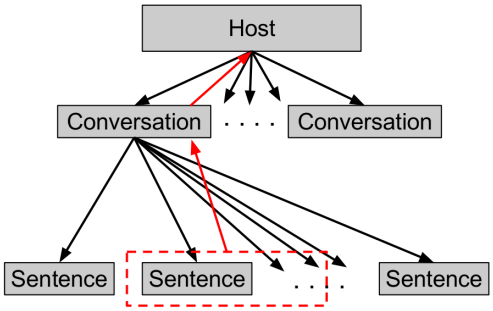

Figure 2: Conversations are a collection of sentences bounded by a time window, shown in red. Figure shows conversation filtering based on a user-specified time window.

4.2 Justification

This structure allows great flexibility in the design and customization of the treemap representation. Flexibility is achieved because a host's data can be shown based on arbitrary time constraints for the conversations it is having with other hosts. Customization is achieved because the program is able to change the display of the dataset to show it in various ways, including filtering hosts using insignificant amounts of bandwidth. More details about flexibility and customization are discussed in Sections 5 and 6.

5 Treemap Visualization

5.1 Layout

One of our design goals was to provide a stable layout for the network traffic data. A layout is considered stable if during a redraw the layout presents as little change as possible to the relative position of elements from the previous display. Because of the random nature of the data, we thought it was important for the layout to be as stable as possible. The data is presented in a treemap by using a spiral rendering algorithm that features acceptably low aspect ratios for readability and more importantly has better spacial continuity between sequential renderings to aid smooth transitions between renderings [1].

Determined by the view, the treemap is partitioned into a set of IP addresses. Partition size is based on the amount of bandwidth in use by each IP address shown, where the larger the area the more bandwidth that host is utilizing. Color is based on shades of red and green, where a deeper shade of red signifies more data has been uploaded and a deeper shade of green signifies more data has been downloaded. For more detail about the color scale, see Section 5.4.1.

Each host is sub-divided into partitions. Assume a host, A. Each of A's partitions represents a different host that has used bandwidth with A for a connection. Since these sub-partitions are not initially shown when a host is selected in the treemap, all of the opposite IP address partitions related with that IP address will be displayed appropriately. This is described more in Section 5.3.3. The sizing and coloring of a particular sub-partition follows the same rules as its parent partition.

5.2 Learning Experience

Our original design as shown below in Figure 3 and Figure 4 show the need for more functionality.

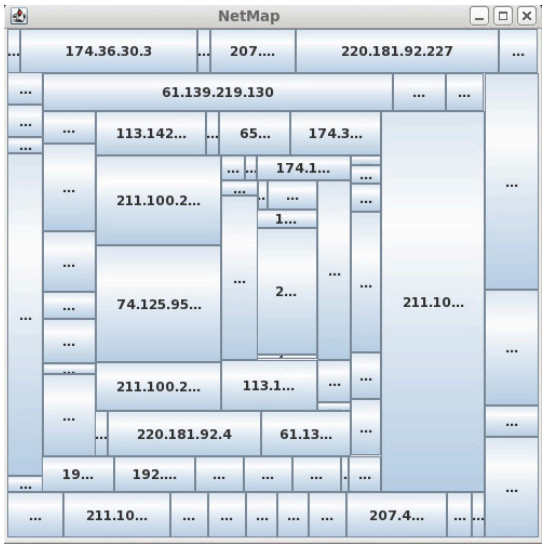

Figure 3: Spiral treemap without skewed hosts.

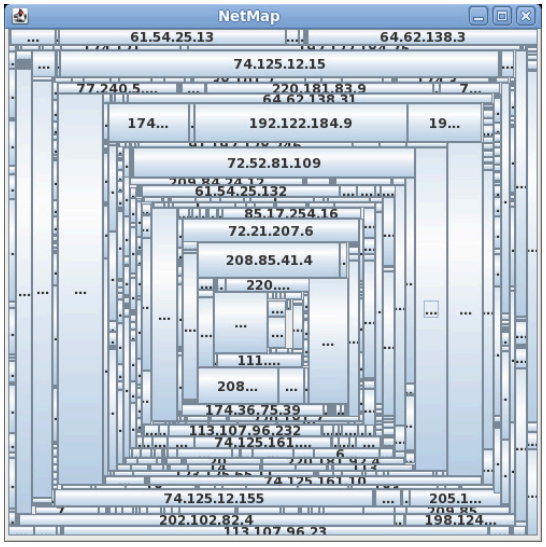

Figure 4: Spiral treemap with skewed hosts.

Figure 3 shows our basic implementation of the spiral treemap as it is discussed in [1]. However, as discussed in Section 5.4, we

needed more functionality to better achieve our goals. This was not due to the inadequacy of the spiral treemap design but rather because of how we wanted to interpret the same dataset multiple ways using the spiral treemap design. For example, Figure 4 shows the treemap with very many hosts that may or may not require a significant portion of the screen area.

Figure 4 implies the need for the ability to filter out particular hosts that may be too small in traffic percentage to be able to be easily clicked on. In this case, sometimes the hosts will take up valuable screen space while adding minimum benefit to the user. This is why we add the *Hide Insignificant Hosts* checkbox as described in Section 5.4.2. Also, Figure 3 shows us only one piece of the information we would like to display and gives us incentive to provide the functionality of the color scale as described in Section 5.4.1.

5.3 Views

A local IP address is defined as an IP address that falls within the group of IP addresses that is associated with the network that is being monitored. For the purposes of our project, our local IP addresses are those addresses that belong to Michigan Technological University. A world IP address is an IP address that is not included in the set of local IP addresses. We present the different views because each view allows the treemap to display the data in a different fashion, thus allowing users to view network traffic data using the view that is best suited for their needs.

5.3.1 Local

The local view displays *all* of the hosts that have local IP addresses whose packets have been captured. We provide this view because it is important to be able to view the network activity of the hosts that are utilizing bandwidth, including identifying whether it is sent or received data. This view is useful because all of the local users' traffic will be shown in this view, which may be used as a quick snapshot into how a network's local users are utilizing bandwidth. This view is shown in Figure 10.

5.3.2 World

The world view displays *all* of the hosts that have world IP addresses whose packets have been captured. We provide this view because it is important to know what outside IP addresses are being accessed by the local individuals that are using network resources. This is useful because unlike the local view, the world view displays all of the world hosts that the local hosts are communicating with. For example, if there are world hosts that require monitoring such as either being prohibited all together or have upper limits on usage then this view provides a means of detecting the IP addresses or the amount of bandwidth that is being used to communicate with them. This view is shown in Figure 5.

Due to security implications, the world view is the default view. We believed that showing the world view would give a user faster access to network problems than the local view for problems like participating in a DDoS, where showing a single world IP with a more-than-expected amount of traffic would be more useful than showing hundreds of local users with equal amounts of traffic. Having the default view as the local view would prevent a user from seeing if the local hosts are contributing to such a DDoS type of assault.

5.3.3 Tunneling

The tunneling view displays either local IP addresses or world IP addresses depending on the host that is selected. When a host is selected to tunnel into, the window redraws the list of IP addresses that are linked by conversations to the selected host. We provide this view as a means to explore which hosts are connected to other hosts and to be able to interactively search through the connections

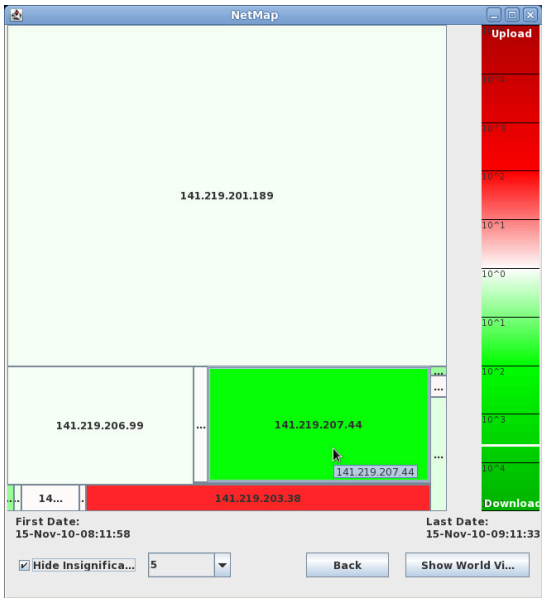

Figure 5: World view without insignificant hosts.

associated between each successive sets of hosts. Successively clicking on hosts will create a *tunnel* as you traverse through the structure. This tunneling functionality is useful because a user can select any host on the screen and see all of the connections associated with that host.

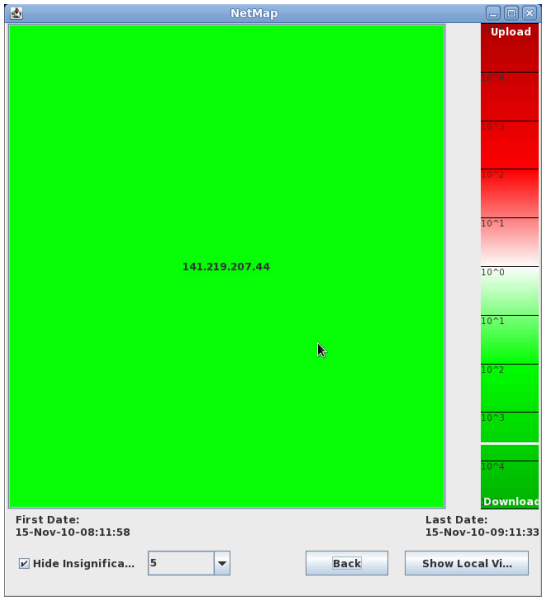

Figure 6: Example of Local Hosts in the Tunneling View

For example, starting from the world view, as shown in Figure 5, clicking on a host will bring up a view containing the local hosts that the clicked host is connected to, as shown in Figure 6. Then proceeding to click on a host in that view will result in a view containing the world hosts that the local host that was just clicked on is connected to, as shown in Figure 7.

Also, the Back button is used only with the tunneling functionality. It allows a user to back up to the previous level as if the click

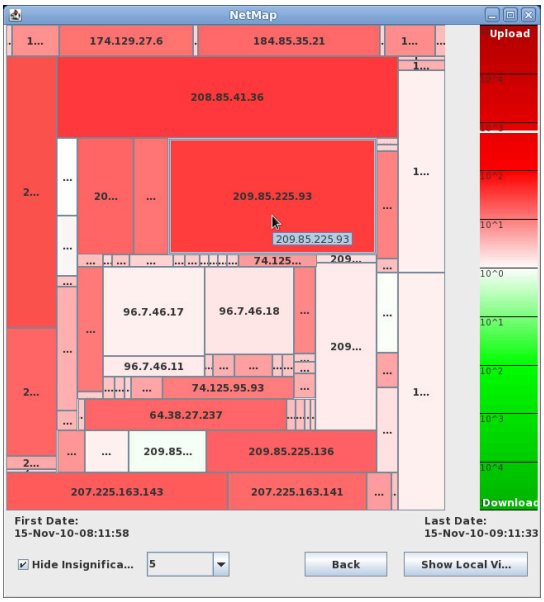

Figure 7: Example of World Hosts in the Tunneling View

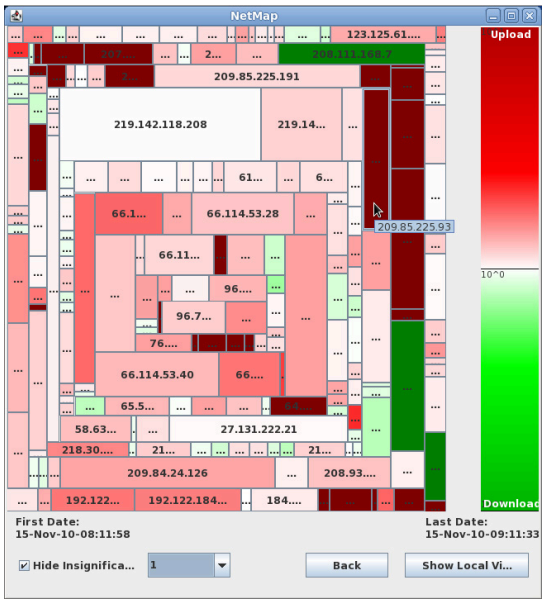

Figure 8: Color Scaling order of 1.

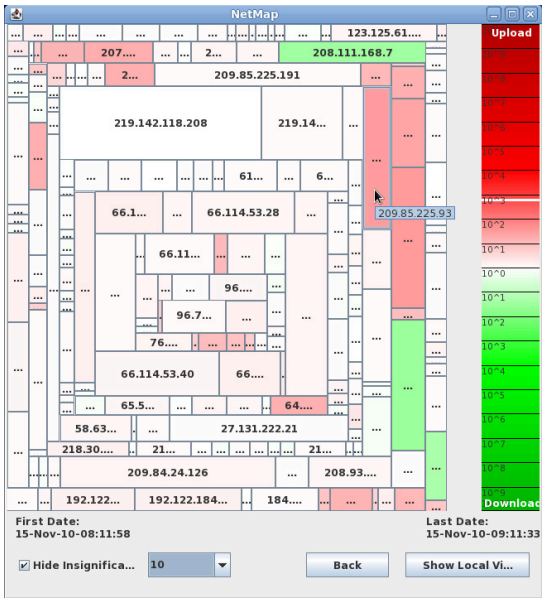

Figure 9: Color Scaling order of 10.

never happened. Clicking will keep a *Tunneling History* such that the Back button will be able to back a user up as many levels as was clicked, even if the clicks were cyclical.

5.4 Other Features

5.4.1 Color Scale and Changing its Range

An important attribute of network data is the two way communication. With these conversations between two different hosts, conversations can be weighted in three different ways. For example, consider the two hosts A and B. A could do a majority of the talking while B just listens. A and B could also do near-equal amounts of talking, or B could do the majority of the talking while A just listens. Assume a host that is talking significantly less than it is listening. If we recall from Section 5.1, the more talking (or uploading) is associated with red and listening (or downloading) is associated with green. This is an important piece of information that our first spiral treemap iteration was not capable of displaying. Therefore, we created the color scale on the right side of our tool. This can be seen in Figures 8, 9 and 10. The color scale uses a logarithmic scale to determine the color gradient. Out of these colors, each host will get the proper color when the treemap is drawn. However, using a static scale would not be very helpful, so we allowed a range of 10 orders of magnitude to demonstrate the difference between the different hosts. This range allows the possibility of a huge contrast between different hosts that may seem similar but are very different with respect to the amount of upload traffic versus the amount of download traffic. Figure 8 shows the scale with 1 order of magnitude and Figure 9 shows the scale with 10 orders of magnitude.

5.4.2 Hiding Insignificant Hosts

Screen real estate is very important when trying to display the volume of data that is associated with network traffic. One of the important aspects was to monitor the percentage of bandwidth usage to determine network congestion issues or possible threats that may arise from general use of a network. We found it difficult to monitor if there were many hosts with very little bandwidth percentages. Therefore, we define insignificance as having an area less than 151 pixels squared or approximately a 12 pixel by 12 pixel box. If the *Hide Insignificant* checkbox is checked and if a host has an area that fulfills the insignificant criteria then the host is considered insignificant and that host will not be drawn. This can be applied to any of the three possible views. The default action is to hide insignificant hosts.

5.5 Effectiveness

In terms of stability, the spiral treemap gave us the best compromise between a stable display and reasonable aspect ratios for the scope of our data. Our data is also ordered by IP address to ease in searching for a specific IP address. However, our approach was not perfect. There was no easy way to visualize the *changes* in the data without using a contrast treemap [1]. Stability was also an issue when the change in the number of nodes between display snapshots was large. Having the list of hosts ordered before drawing helped,

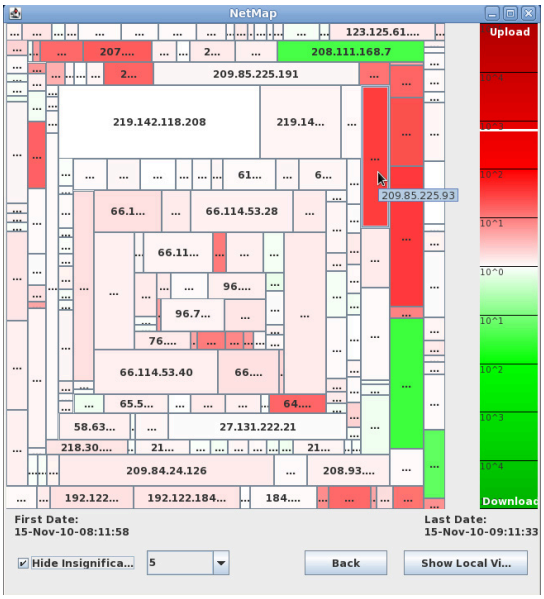

Figure 10: Local view hiding insignificant hosts.

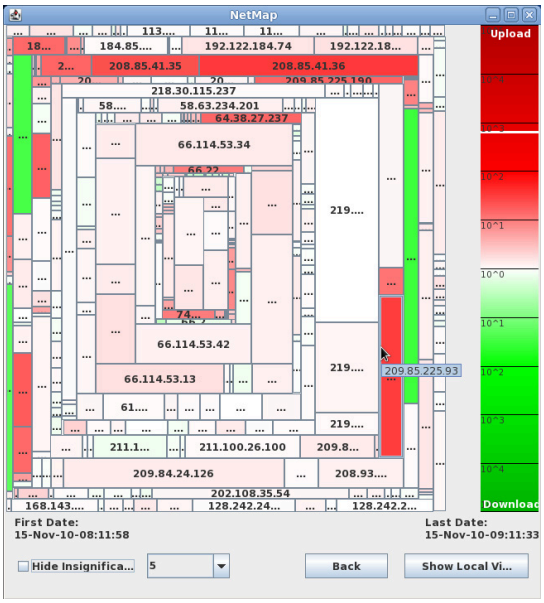

Figure 11: Local view showing insignificant hosts.

but it seemed too difficult to see minor changes within the hosts displayed. However, the program was developed with the intention of seeing the largest contributors toward the data rather than the smallest. More discussion on these can be found in Section 6.

Hiding insignificant hosts helped with *cleaning-up* the data, which made viewing the hosts much easier since there was more room to show more relevant data. Though we wanted to show all of the data, it was just too difficult to select a host whose area was too small to click or see. Because of this, we believe this was a good addition to our design.

Tunneling seemed effective at determining with whom each host was communicating. It was sometimes confusing to not be able to tell which host we were currently examining, as that information does not get displayed in our implementation. Despite this, the relationships between hosts was still easy to see and follow.

The world and local view was a great tool for a *start over* feature when viewing the data. As selecting the World/Local View button would always bring a user back to the highest overview of hosts, this proved useful when a user got lost in the hosts.

The ability to change the order of magnitude scale as it applies to the color scale is especially helpful since a user can better inspect those hosts that either have relatively equal amounts of upload and download data or have significantly skewed amounts of upload or download data. This was justified since users might have difficulty deciphering two white-looking hosts when 10 orders of magnitude are shown. Furthermore, users might have difficulty interpreting two red-looking hosts when only 1 order of magnitude is shown. In addition to the orders of magnitude selector, the color-pinpointing of the color scale also significantly enhances comparing the colors between nodes by actually displaying where a host's color is located on the color scale.

6 Future Work

6.1 Tunneling View History

In addition to just tunneling through hosts to view their connections, there was a desire to also display the current stack of hosts as a breadcrumb list or a scrollable list. This would help to keep track of where the user has clicked and would allow the user to go back to any point in the selection history.

6.2 Time Window

The program was designed with the intention of utilizing user-specified time window filtering (via sentences and conversations). Adding a sliding time window or a user specified start and stop date to filter the data could be a very useful feature for displaying trends in data over specific timer periods.

6.3 Resolving IP Addresses to Hostnames

The ability to resolve hostnames proved difficult as there was no easy means to gain this information due to our implementation. We had to explicitly reserve our wireless adapter to capture data by dis-associating it from any network. Therefore, we required another adapter to resolve IP addresses. Ideally, this functionality should also be included, especially for world hosts so a user can find relationships in how their local hosts use the network.

6.4 Improved Stability and Visualizing Changes

Though our spiral treemap has relatively good stability, it is far from perfect since a host can easily be skewed into a much different area of the window with very little change in the data. Quantifying changes in the data is also very difficult without other tools, such as contrast treemaps [1].

6.5 Domain Groupings

Building on resolving hostnames, grouping hosts of like domains would be very helpful in the world view to see which organizations or companies have the most traffic. This would have the added benefit of making the treemap truly hierarchical, where treemaps excel.

6.6 More Detailed Rollovers

There are no hard numbers for displaying *how much* data was actually transferred in our implementation. Adding in data values in KB/Kb/MB/Mb, etc. would be useful to see data throughput. Including information such as IP Addresses (once resolving hostnames is implemented), extra company information, actual usage percentage, total upload count, total download count, or the number of hosts in that group (when domain grouping is implemented) would be beneficial to a user.

7 Conclusion

In this paper, we introduced idea of a network monitoring treemap project. We presented our reasoning for the project, which included exploring the effectiveness of treemaps as a method to visualize network traffic data. As such, we aim to use a treemap to show network traffic data in a variety of ways because classifying this data differently provides different interpretations to yield more valuable information about a particular network. Overall, the program met its goals and although the design and implementation had weaknesses, the program was sound, stable, easy-to-use, and informative.

References

[1] Y. Tu and H.-W. Shen. Visualizing changes of hierarchical data using treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 13:1286–1293, 2007.