Manifold Learning Visualization of Network Traffic Data *

Neal Patwari
npatwari@eecs.umich.edu

Alfred O. Hero III
hero@eecs.umich.edu

Adam Pacholski
apachols@umich.edu

University of Michigan
Dept. of Electrical Engineering and Computer Science
1301 Beal Avenue, Ann Arbor, MI, USA

ABSTRACT

When traffic anomalies or intrusion attempts occur on the network, we expect that the distribution of network traffic will change. Monitoring the network for changes over time, across space (at various routers in the network), over source and destination ports, IP addresses, or AS numbers, is an important part of anomaly detection. We present a manifold learning (ML)-based tool for the visualization of large sets of data which emphasizes the unusually small or large correlations that exist within the data set. We apply the tool to display anomalous traffic recorded by NetFlow on the Abilene backbone network. Furthermore, we present an online Java-based GUI which allows interactive demonstration of the use of the visualization method.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Network Monitoring. **General Terms**: Algorithms, Measurement. **Keywords**: Internet traffic anomaly detection& forensics, data mining.

1. INTRODUCTION

Statistical intrusion and anomaly detection methods allow networks to be monitored for attacks for which attack signatures have not yet been developed. However, the huge quantity and high-dimensionality of internet traffic data are significant challenges which research must overcome in order to achieve high reliability and low false-alarm rates. Recently, subspace-based analysis of traffic data by Lakhina, Crovella, and Diot [1, 2] has shown that high-dimensional Abilene traffic measurements can be well-represented within a very low-dimensional subspace. The 'curse-of-dimensionality' can be avoided when high-dimensional data can be represented well in a low-dimensional subspace.

*This research was supported in part by National Science Foundation ITR Grant No. CCR-0325571.

In this paper, we use a manifold learning method to take very high-dimensional NetFlow traffic measurements from the Abilene backbone network and reduce their dimensionality to two dimensions. The resulting 2-D 'map' of the measurements provides a means for visualization of the relationships which exist in a set of traffic data. These relationships may be spatial, eg., between measurements taken across a backbone network or between IP addresses, autonomous systems (AS), or origin-destination (OD)-flows; temporal, eg., measurements taken at different times; or between different applications, as indicated by port numbers.

Such visualization is complementary to detection methods which rely on dimensionality reduction. Subspace-based detection [2] has been successfully used infer the presence of spatial traffic distribution anomalies in network-wide traffic measurements. This inference is done by quantifying the amount of traffic which *cannot* be represented within a low-dimensional subspace. The method we present in this paper allows the visualization of the traffic which *can* be represented within a low-dimensional subspace. Furthermore, this work uses a non-linear dimensionality reduction method rather than a linear subspace method.

1.1 Manifold Learning

Manifold learning encompasses a variety of methods to approximate in a low dimensional space (such as 2-D) the relationships between data points which lie on a manifold in a high dimensional space [3, 4, 5]. Classical multi-dimensional scaling (MDS) and principal components analysis (PCA) may be the most commonly recognized dimensionality reduction methods, but they assume that the high-dimensional data points lie on a linear subspace (for example, on a 2-D plane) of the high-dimensional space. Manifold learning algorithms are more general - data may lie on a curved subspace. For example, a 2-D manifold could be a portion of a sphere, or a 'swiss roll'. When high dimensional data points are random but highly correlated with their close neighbors, data points don't tend to fall into linear subspaces. As a result, manifold learning methods can be more effective than linear methods like MDS.

In particular, in this paper, we use a manifold learning method called the distributed weighted MDS (dwMDS) method [5]. Its key features are:

1. A distributed implementation that allows router maps (see Section 3.3) to calculated in a distributed manner across the network with minimal communication requirements.

2. Consideration of prior information, which allows use of

a 'typical' map to be used as a baseline, thus allowing for easy comparison of data maps over time.

3. A weighted cost function that allows neighbor relationships that are believed to be more accurate to be weighted more heavily.

4. A majorization method which has the property that each iteration is guaranteed to improve the value of the cost function.

1.2 Visualization

In order to facilitate human moderation, we limit ourselves to a 2-D, or possibly 3-D, representation of the data. We believe that data visualization will complement statistical detection methods, and help provide information to help a human moderator make a decision regarding whether or not an anomaly has occurred, and if so, to determine its temporal and spatial characteristics.

Other visualization methods have found use in network monitoring. For example, visualization of flows by application over time is commonly done using FlowScan [6]. Monitoring the number of flows over time using FlowScan is an excellent tool to identify DoS attacks. However, there is an increasing a number of ports at which attacks are possible. As attacks (such as the Slammer worm) exploit smaller user populations, even obscure services' traffic must be monitored. Dimensionality reduction is a means to monitor, separately, hundreds or thousands of traffic statistics but to minimize the complexity of the information display.

Furthermore, graph visualization provides information regarding the physical connections that exist in a network. Visualizations of the global internet, such as CAIDA's Skitter plot [7], are important statements about the interconnectivity of the global network. Other tools developed at CAIDA, such as Otter and Walrus, provide 2-D and 3-D visualization of network graphs. The visualization method presented in this paper provides information not just about the connections that exist, but also the traffic correlations that exist. Connection distances match correlation - when correlation between two nodes is high, they are plotted close together, but when correlation is low, nodes are drawn further apart.

1.3 Framework

We frame the dimensionality reduction problem as a sensor data localization problem. In this framework, 'sensors' are the hardware or software which record data, for example, on each router in a backbone network. The traffic data which they record can be of arbitrarily high dimension. For example, rather than counting the grand total number of flows (just one dimension), they would count the total number of flows from each source IP address (up to 2^{32} dimensions). The key to understanding a particular sensor data map is to know "Where are the sensors?", "What traffic statistic is recorded?", and "What are the dimensions?":

1. *Where are the sensors?*: Sensors can be 'located' at physical computers, *i.e.*, at backbone routers, or at IP addresses; or they can be 'located' at less physical concepts such as source or destination ports, or particular time periods. A sensor attached to a particular source port monitors only traffic which matches its source port, and a sensor attached to a time monitors only traffic which arrives during that time period.

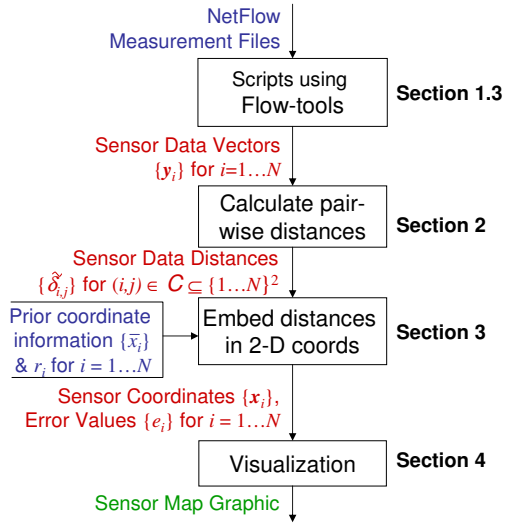

Figure 1: Flow chart of data visualization from NetFlow data input to data map output.

2. *What traffic statistic is recorded?*: Sensors might measure flows, packets, or octets, or some combination of the three.

3. *What are the dimensions?*: Sensors can divide traffic by source or destination IP address, port, or AS; time period; link or router; or some combination of them. Traffic statistics are then recorded for each dimension (port, IP address, AS, time period, link or router) separately.

For example, in Section 4.1, sensors are located at backbone routers, recording the number of flows from each source IP address. As another example, in Section 4.2, sensors are located on destination port numbers, recording the number of flows from each source IP address. As a final example, in Section 4.3, sensors are located on backbone routers, recording the total number of packets received in each of the past T 5-minute time intervals.

We note that this framework can be used to describe the measurements in [1], in which sensors were located at all 10-minute time intervals over the course of a week, and sensors measured total octets on each link across the (Sprint-Europe or Abilene) backbone network.

We denote the data measured at sensor i as $\mathbf{y}_i$, where $i \in \{1, \ldots, N\}$, where N is the total number of sensors. The high-dimensional vector $\mathbf{y}_i$ is defined by,

$$\mathbf{y}_i = [y_i(l_1), y_i(l_2), \ldots, y_i(l_{|\mathcal{L}|})], \quad (1)$$

where $\mathcal{L}$ is the set of possible dimensions (see #3 above) with $|\mathcal{L}|$ elements, and $l_k \in \mathcal{L}$ for all $l = 1 \ldots |\mathcal{L}|$. In many applications, $y_i(l_k) = 0$ for most of its elements l_k, thus $\mathbf{y}_i$ is best stored as a sparse vector. For example, the set of possible IP addresses is much larger than the set of IP addresses observed in a particular traffic stream.

We use the flow-tools package created by Mark Fullmer [8] to process NetFlow files in order to generate the data for the vectors $\{\mathbf{y}_i\}$.

2. MEASUREMENT DISTANCES

ATLA	CHIN	DNVR
130.14.24.0, 1545	129.25.0.0, 13913	129.25.0.0 14331
131.247.224.0, 1487	141.89.48.0, 8738	207.46.104.0 12142
128.61.64.0, 1197	207.46.104.0, 3708	207.46.248.0 7198
198.32.152.0, 1147	204.179.120.0, 3520	207.68.176.0 4968
164.111.192.0, 1139	203.250.224.0, 3441	64.4.16.0 4156
131.247.232.0, 1098	207.46.248.0, 3300	207.68.168.0 3707
⋮	⋮	⋮

Table 1: Example data: Top few lines of $\{\mathbf{y}_i\}$ for 3 Abilene routers, flows by source IP (last 11 bits zeroed) for 5 minutes ending 20 Jan 2005 01:00 UTD.

After sensors record $\{\mathbf{y}_i\}_i$ as described in Section 1.3, the next step is to calculate distances between the data vectors. Often, two sensors will record different levels of total traffic even though their traffic is very correlated. In this paper, unless otherwise noted, we normalize each data vector such that its sum is one:

$$\tilde{\mathbf{y}}_i = \frac{\mathbf{y}_i}{\|\mathbf{y}_i\|_1} \tag{2}$$

where $\|\mathbf{y}_i\|_1$ is the L_1 norm, *i.e.*, the total traffic measured at sensor i. The value $\tilde{\mathbf{y}}_i(l)$ thus is the fraction of traffic measured in dimension l. Rather than calculating the distance between absolute traffic levels, we can compare the distribution of traffic.

Let $\delta_{i,j}$ be the distance between the measurement vectors from sensors i and j. An arbitrary norm can be used; in this paper, we use the Euclidean distance between $\tilde{\mathbf{y}}_i$ and $\tilde{\mathbf{y}}_j$,

$$\delta_{i,j} = \Big[\sum_{l \in |\mathcal{L}|} \left(\tilde{\mathbf{y}}_i(l) - \tilde{\mathbf{y}}_j(l)\right)^2 \Big]^{\frac{1}{2}} \tag{3}$$

Defining $D_{\mathbf{y}} = \sum_{(i,j)\in\mathcal{C}} \delta_{i,j}$, *i.e.*, the sum of all of the distances between neighbors. To ensure that different sensor maps are approximately the same size, we also normalize $\delta_{i,j}$ in order to achieve a desired constant sum, $D_{\mathbf{x}}$, for some $D_{\mathbf{x}} \in \mathbb{R}$. For each $(i,j) \in \mathcal{C}$, we define

$$\tilde{\delta}_{i,j} = \delta_{i,j} \frac{D_{\mathbf{x}}}{D_{\mathbf{y}}} \tag{4}$$

2.1 Neighbor Selection

Using the set of normalized distances $\{\tilde{\delta}_{i,j}\}$, we next determine the sensor neighbor relation. Intuitively, pairs of sensors which are 'close' to each other will consider each other to be neighbors. We use the K-nearest neighbors method to determine the neighbor set, which we denote $\mathcal{C} \subseteq \{1,\ldots,N\}^2$, In K-nearest-neighbors, a pair i and j are neighbors if either j is one of the K nearest neighbors of i, or j is one of the K nearest neighbors of i. Specifically, i and j are neighbors if $\tilde{\delta}_{i,j}$ is one of the K smallest of either set $\{\tilde{\delta}_{i,k}\}_{k\neq i}$ or set $\{\tilde{\delta}_{k,j}\}_{k\neq j}$.

We could also limit $\mathcal{C}$ to be a subset of $\bar{\mathcal{C}} \subset \{1,\ldots,N\}^2$, where $\bar{\mathcal{C}}$ is a set of *allowed* neighbors. When sensors are routers, $\bar{\mathcal{C}}$ could be pairs of routers physically linked (or L-hop neighbors) in the network. This prior limitation is useful to bound the data communication and distance calculation to $\mathcal{O}(qN)$, where q is the average number of physical connections (or L-hop neighbors) at each router, rather than the $\mathcal{O}(N^2)$ necessary for full pair-wise distance calculation.

3. COORDINATE EMBEDDING

Using the calculated distances $\{\tilde{\delta}_{i,j}\}$, for neighbor pairs $(i,j) \in \mathcal{C}$, we calculate the coordinate embedding in two dimensions. In the dwMDS method, this means that we find the two-dimensional coordinates $\{\mathbf{x}_i\}_{i=1\ldots N}$ which best represent the calculated distances and the prior coordinate information in a weighted-least-squares sense. Specifically, we find the 2-D vectors $\{\mathbf{x}_i\}_{i=1\ldots N}$ which minimize the following cost S,

$$S = 2\sum_{i=2}^{N}\sum_{j=1}^{i-1} w_{i,j}\left[\tilde{\delta}_{ij} - \|\mathbf{x}_i - \mathbf{x}_j\|\right]^2 + \sum_{i=1}^{N} r_i \|\mathbf{x}_i - \overline{\mathbf{x}}_i\|^2 \tag{5}$$

where $w_{i,j}$ is the weight assigned to the link (i,j), r_i is the weight assigned to the prior coordinate information for sensor i, and $\overline{\mathbf{x}}_i$ is the prior coordinate given for sensor i. First, we discuss the assignment of $w_{i,j}$, and then, we discuss the assignment of prior coordinate information.

3.1 Assignment of Weights

Weights are assigned to each pair of neighbors to allow pairs which are perceived to have a less accurate measured distance to be down-weighted. Inspired by the weighting frequently used in locally weighted regression methods (LOESS) [9], we use a weighting w_{ij} that is a decreasing function of measured distance $\tilde{\delta}_{ij}$:

$$w_{ij} = \begin{cases} \exp\left\{-\tilde{\delta}_{ij}^2/h_i^2\right\}, & \text{if } (i,j)\in\mathcal{C}, \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

where $h_i = \max_j \tilde{\delta}_{ij}$. This choice of w_{ij} equalizes the (nonzero) weight distribution in all sensors.

3.2 Prior Coordinate Information

Prior coordinate information is an option in the dwMDS method. If there is no prior information for sensor i, we set $r_i = 0$. If all sensors have $r_i = 0$, the calculated output map can arbitrarily be translated, rotated, and flipped without affecting its cost S. The purpose of the map is to show the relationships between sensors' data, and as such, translation, rotation, and flipping do not change the meaning of the map. However, if the user will view many maps, for example, in sequence over time, it would be confusing if each subsequent map was nearly identical but with arbitrary rotation. Prior coordinate information ($r_i > 0$) is a means to provide a stable orientation for a sensor map. For example, when the sensors are on Abilene backbone routers as described in Section 4.1, we use the router coordinates plotted in Fig. 2(a).

If desired, a sensor can be set to be an *anchor*, *i.e.*, a sensor whose coordinates are predetermined and not subject to change, by setting $r = \infty$ [5]. For $0 < r_i < \infty$, the choice of r_i determines how sensitive the sensor's location is to changes in sensor data. A high r_i, that is, $r_i \gg \sum_j w_{i,j}$ will mean that node i acts like an anchor except in very extreme circumstances. In this paper, we typically use a low r_i, that is, $r_i \ll \sum_j w_{i,j}$ will mean that the map will be predominantly attempt to match the measurement distances, and use the prior coordinates only to remove translational and rotational degrees of freedom. Unless otherwise noted, we use $r_i = \epsilon = 10^{-3}$, a value low enough to only affect the translation and rotation, but not the shape of the sensor map.

3.3 Algorithm Overview

The dwMDS algorithm is a distributable, iterative algorithm to minimize S and thus find the desired 2-D coordinate embedding. In this paper, we only present a broad overview of the dwMDS algorithm, and leave the details to the reference [5].

The global cost S is separable by sensor, *i.e.*, $S = \sum_{i=1}^{N} S_i$, where S_i is sensor i's contribution to the cost function. Sensor i can minimize S_i using only distances $\tilde{\delta}_{i,j}$ between itself and its neighbors j, and its neighbors' previous iteration 2-D coordinate estimate. In a distributed implementation, a degree of information privacy is enabled since measured distances and coordinate estimates are only exchanged between nearest neighbors. Each sensor initializes its coordinate estimate to its prior coordinate, $\overline{\mathbf{x}}_i$, if one exists, or a random coordinate, if not. Then, each sensor uses a simple majorization algorithm which guarantees non-increasing cost in each round of the optimization. In a distributed implementation, each sensor iteratively minimizes its own cost and updates its most recent coordinate estimate. Then, it passes its new coordinate estimate to the next sensor, which then performs its own local minimization. The algorithm iterates in order to visit each sensor in turn, and may perform several rounds until a convergence criteria is met.

We use a centralized implementation of the dwMDS algorithm, in which the data is collected and optimized at a single processor [10]. This is used to demonstrate the visualization method's capabilities, but the distributed implementation discussed here and in [5] is future work.

3.4 Error Metric

We also define an error value e_i^2 for sensors $i \in \{1, \ldots N\}$:

$$e_i^2 = \sum_{(i,j)\in\mathcal{C}} w_{i,j} \left(\tilde{\delta}_{ij} - \|\mathbf{x}_i - \mathbf{x}_j\|\right)^2 \quad (7)$$

The value of e_i helps quantify how much information was lost in the low dimensional representation of the sensor coordinates. It represents the quantity of measurement distances $\{\tilde{\delta}_{i,j}\}_j$ which are not represented by the 2-D coordinates of itself, $\mathbf{x}_i$, and its neighbors j, $\{\mathbf{x}_j\}$. The value e_i^2 is analogous to the residual value in PCA and can be used to help decide whether or not the sensor's data is anomalous. In this paper, we 'color' each sensor as a function of e_i: if e_i is low, we shade the sensor light gray, and if e_i is high, we shade the sensor dark gray. Examples are presented in the following section.

We note that the dwMDS method will output coordinates in an arbitrary dimension. Thus, although we limit ourselves to 2-D for the purposes of this paper, 3-D visualization is equally possible with the presented method.

4. CASE STUDIES

In this section we show several examples of how the proposed algorithms can be used to visualize backbone traffic on Abilene. We use NetFlow data recorded during January 2005, available from the Abilene Observatory [11]. Note that, for privacy reasons, only the most significant 21 bits of IP addresses are available - the last 11 bits are zeroed out. Furthermore, NetFlow data is sampled at 1/100, so for each packet reported here, there were likely 99 more unrecorded.

4.1 Router Maps

When sensors are attached to routers in a backbone network, a 'router map' shows the spatial characteristics and correlations of the routers' traffic, rather than just the connectivity of the routers. In the following examples, sensors are routers, and we measure flows in a 5-minute period, separated by source IP address. The size of each sparse data vector is limited to 1000 – flows not from the top 1000 source IP addresses (/21) for a particular router are ignored. For this reason, source IP addresses are typically lost if they have less than 10-50 flows. We set the number of neighbors to $K = 5$.

4.1.1 Typical Activity

First, we attempt to characterize 'typical' router map behavior using the router maps for the four-week period 02-Jan to 29-Jan. Since a router map can be calculated for each 5 minute period (12 per hour), there are a total of $12*24*7*4 = 8064$ maps. We calculate the 8064 maps, and then calculate the sample mean and covariance of $\mathbf{x}_i$ for each router i, and show the results in Fig. 2(a). Although there are certainly attacks active during this 4-week period, averaging maps over a long period of time can provide intuition about what is typical behavior for the router map.

The typical router map in Fig. 2(a) both makes sense geographically and describes typical traffic patterns seen on Abilene. Much of Abilene traffic is East-West or West-East, and Northern routers (especially DNVR, KSCY, IPLS, and CHIN) bear much of this traffic. These routers have very correlated sensor data because a significant proportion of Abilene OD-flows pass through all of them.

4.1.2 06-Jan-2005

First, we show that the router map changes very dramatically with very large traffic changes. On Thursday, 6-Jan-2005, during the 5-minute period ending at 17:55 UTD, NetFlow data recorded on the CHIN router count a total of 9×10^4 single-packet flows (of 40-byte packets) from two source IP addresses in Taiwan to a small range of destination IP addresses in Hungary. This volume corresponds to about 25% of the typical flow volume on CHIN. The traffic from the two Taiwanese source IP addresses was observed on CHIN and no other router, thus distances between sensor data recorded at CHIN and other routers are unusually high, and the 2-D coordinates for CHIN must be kept very distant from all other sensors. Also, because of the normalization done to calculate $\tilde{\delta}$ from δ, the rest of the map distances have shrunk to compensate. This is shown in Fig. 2(b).

4.1.3 20-Jan-2005

On Thursday, 20-Jan at 01:00 UTD, there are a large number (14,000) of 29-byte packets from a 129.25.0.0 (Drexel U.) source IP address sent to a 131.252.120.0 (Portland State U.) destination (see Table 1). The packets are UDP with source port 3095 or 3096 to a wide range of random destination ports > 1024. These packets travel through the WASH, NYCM, CHIN, IPLS, KSCY, DNVR, and STTL backbone routers. Other routers (SNVA, LOSA, HSTN, and ATLA) do not see any flows from this source address at this time. Distances between the listed Northern routers and the other Southern routers are unusually high. In the router map shown in Fig. 2(c), there is a clear split in the map between the two sets of routers.

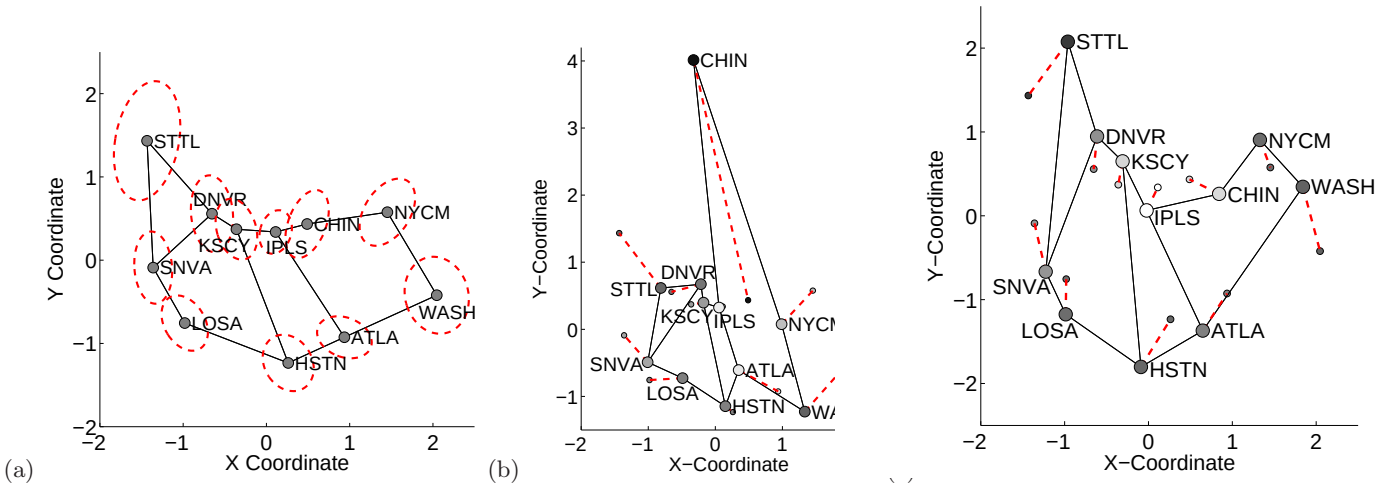

Figure 2: (a) Mean (•) and 1-σ uncertainty ellipse (- - -) of router maps from 2-Jan to 29-Jan. Maps during (b) port scan on 6-Jan 17:55 and (c) attack on 20-Jan 01:00, show router coordinates (•) connected (- - -) to the mean (·) from (a), and shaded by error value e_i. All figures show Abilene backbone links (—).

4.2 Port Maps

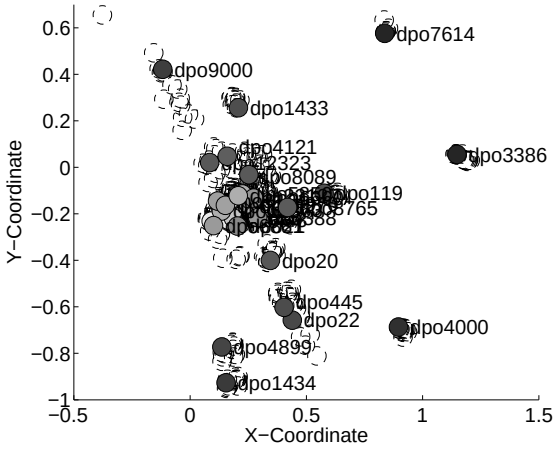

Figure 3: Destination port map for 01-Jan-2005 at 3:35 UTD (•dpo#) along with past 1 hour map history (dotted line circles). Sensors are attached to the top 30 destination ports (by total flows) and measure number of flows per source IP address.

Here, we present 'port maps' which can be used to visually detect ports which exhibit anomalous traffic. For a port map, sensors are attached to destination ports and record the number of flows per source IP address. That is, each sensor is assigned a port and only looks at flows that have that destination port. We would expect that attacks or scanning activity would exhibit very different distributions of source IP addresses as compared to ports that aren't the subject of attacks or scans. Unlike the router map, the port map can be calculated using only the data at one particular router (Fig. 3 uses data from IPLS).

At each time, we attach sensors to 30 ports (we select the ports with the highest number of flows). We measure flows per source IP address within the current 5 minute time period. Then, the port map is calculated with $K = 15$ and no prior ($r_i = 0$ for all i). As comparison, we include port maps calculated over the past hour. Again, to enable easy comparison, we rotate and translate the past port maps to line up with the most recent map. Fig. 3 shows an example for 1-Jan-2005. While most ports fall very close to each other near the origin, a few ports have very different source IP address distributions, thus are placed far from the center and have a higher e_i (as indicated by the darker shade of their circle). Notably, destination ports 9000, 1433, 7614, 3386, 4000, 445, 22, 4899, and 1434 are mapped far from the center, all of which have known vulnerabilities.

Many of these ports are consistently 'abnormal' and shown far from the center of the port map over a wide range of time (days), but destination port 7614 appeared on the port map rarely. Its appearance at 03:55 does correspond to a scan originating from source IP 211.82.216/21 for that destination port using single, 48-byte packet flows. Although 01-Jan is a low-traffic day in general, the port map can emphasize such a small scan (480 flows out of 1.2×10^5 total) because 100% of destination port 7614 traffic corresponds to a unique (and otherwise unrecorded) source IP address.

4.3 Map Web Applet

We have available a Java-based applet which displays temporal and spatial traffic on the Abilene backbone [10], as shown in Fig. 4. The applet has a graph of the traffic for each router, and can plot traffic levels from 21 different sets of ports grouped by application.

In addition to displaying the traffic by port and time, the applet calculates a router map for each time. In this router map, the sensors are routers, and they measure total packets, by port and 5-minute time interval. The user can select which groups of ports, and the number of time intervals to use as dimensions. Rather than normalizing to calculate $\tilde{\mathbf{y}}_i(l)$ as described by (2), we use $\tilde{\mathbf{y}}_i(l) = \mathbf{y}_i(l) - \overline{\mathbf{y}}_i(l)$, where $\overline{\mathbf{y}}_i(l)$ is the median of the past T_m time samples, where T_m is also user-adjustable. Using a filtered traffic stream emphasizes the changes that occur over time. When traffic changes

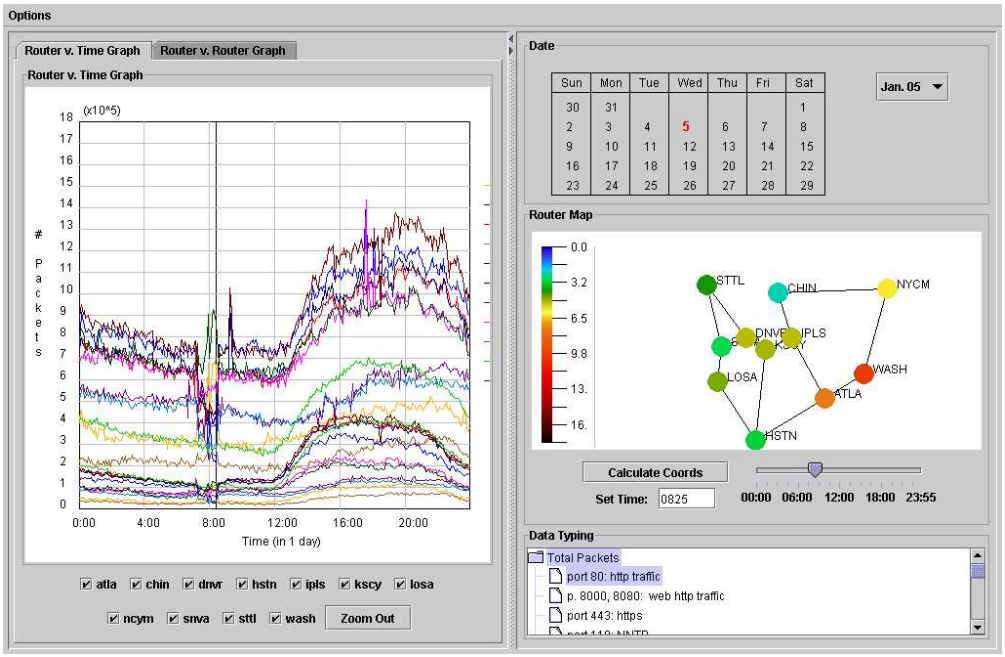

Figure 4: Total traffic and port 80 traffic on 05-Jan-2005, displayed using the visualization applet [10]. The router map is calculated for 08:25 UTD, during scheduled maintenance of the CHIN-IPLS link, during which traffic drops at CHIN and IPLS and increases dramatically at the HSTN and ATLA routers.

over time, the router map shows where (which routers) the change is most dramatic. Details are available online [10].

5. FUTURE WORK AND CONCLUSION

We have introduced a visualization tool which can aid in the discovery of malicious activity and other traffic anomalies in high-dimensional NetFlow data. We urge the reader to visit the online repository to view other sensor maps and to utilize the web-based visualization applet [10]. Clearly, manifold learning methods can be used to provide information about relationships that exist in sets of network traffic data.

Future work will certainly attempt to automate the detection and classification process, similar to the application of subspace-based detection methods in [1, 2]. Further, we hope to investigate other distance metrics which may better emphasize similarities in traffic distributions besides the L_2 norm. Other ML methods such as Isomap [3] and locally-linear embedding (LLE) [4] should also be tested. It is hoped that router maps, port maps, and other types of sensor maps may together serve as a step-by-step investigation aid, iteratively helping to locate a traffic anomaly in a very high-dimensional spatial and temporal data space.

Acknowledgements

The authors thank Panna Felsen, who initiated the visualization applet as part of the NASA SHARP program.

6. REFERENCES

[1] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," in *ACM SIGCOMM '04*, Aug. 2004. [Online]. Available: http://www.cs.bu.edu/faculty/crovella/paper-archive/sigc04-network-wide-anomalies.pdf

[2] ——, "Characterization of network-wide anomalies in traffic flows," in *ACM/SIGCOMM Internet Measurement Conference*, Oct. 2004. [Online]. Available: http://www.cs.bu.edu/faculty/crovella/paper-archive/imc04-characterizing-network-wide.pdf

[3] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, Dec 2000.

[4] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by local linear embedding," *Science*, vol. 290, pp. 2323–2326, Dec 2000.

[5] J. A. Costa, N. Patwari, and A. O. Hero III, "Distributed multidimensional scaling with adaptive weighting for node localization in sensor networks," *IEEE/ACM Trans. Sensor Networks*, May 2004, (submitted). [Online]. Available: http://www.eecs.umich.edu/~hero/comm.html

[6] D. Plonka, "Flowscan: A network traffic flow reporting and visualization tool," in *Large Installation System Administration (LISA) Conference 2000*, Dec. 2000, pp. 305–317. [Online]. Available: http://www.usenix.org/events/lisa2000/plonka.html

[7] Cooperative Association for Internet Data Analysis (CAIDA). http://www.caida.org/.

[8] M. Fullmer and S. Romig, "The OSU Flow-tools package and Cisco NetFlow logs," in *Large Installation System Administration (LISA) Conference 2000*, Dec. 2000, pp. 291–303. [Online]. Available: http://www.usenix.org/events/lisa2000/fullmer.html

[9] W. Cleveland, "Robust locally weighted regression and smoothing scatterplots," *J. American Statistical Assoc.*, vol. 74, no. 368, pp. 829–836, 1979.

[10] Map-tools online supplement. http://www.engin.umich.edu/~npatwari/mnd05.

[11] Internet2: Abilene Observatory. http://abilene.internet2.edu/observatory/.