On Measuring the Similarity of Network Hosts: Pitfalls, New Metrics, and Empirical Analyses

Scott E. Coull
RedJack, LLC.
Silver Spring, MD
scott.coull@redjack.com

Fabian Monrose
University of North Carolina
Chapel Hill, NC
fabian@cs.unc.edu

Michael Bailey
University of Michigan
Ann Arbor, MI
mibailey@eecs.umich.edu

Abstract

As the scope and scale of network data grows, security practitioners and network operators are increasingly turning to automated data analysis methods to extract meaningful information. Underpinning these methods are distance metrics that represent the similarity between two values or objects. In this paper, we argue that many of the obvious distance metrics used to measure behavioral similarity among network hosts fail to capture the semantic meaning imbued by network protocols. Furthermore, they also tend to ignore long-term temporal structure of the objects being measured. To explore the role of these semantic and temporal characteristics, we develop a new behavioral distance metric for network hosts and compare its performance to a metric that ignores such information. Specifically, we propose semantically meaningful metrics for common data types found within network data, show how these metrics can be combined to treat network data as a unified metric space, and describe a temporal sequencing algorithm that captures long-term causal relationships. In doing so, we bring to light several challenges inherent in defining behavioral metrics for network data, and put forth a new way of approaching network data analysis problems. Our proposed metric is empirically evaluated on a dataset of over 30 million network flows, with results that underscore the utility of a holistic approach to network data analysis.

1 Introduction

Network operators face a bewildering array of security and operational challenges that require significant instrumentation and measurement of their networks, including denial of service attacks, supporting quality of service, and capacity planning. Unfortunately, the complexity of modern networks often make manual examination of the data provided by such instrumentation impractical. As a result, operators and security practitioners turn to automated analysis methods to pinpoint interesting or anomalous network activities. Underlying each of these methods and their associated applications is a fundamental question: to what extent are two sets of network activities similar?

As straightforward as this may seem, the techniques for determining this similarity are as varied as the number of domains they have been applied to. These range from identifying duplicates and minor variations using cryptographic hashes and edit distance of payloads, to the use of distributions of features and their related statistics (*e.g.,* entropy). While each has been shown to provide value in solving specific problems, the diversity of approaches clearly indicates that there is no generally accepted method for reasoning about the similarity of network activities.

With this in mind, the goal of this paper is to make progress in developing a unified approach to measuring the similarity of network activities. In doing so, we hope to encourage a more rigorous method for describing network behaviors, which will hopefully lead to new applications that would be difficult to achieve otherwise. While a complete framework for rigorously defining distance is beyond the scope of any one paper, we address two key aspects of network similarity that we believe must be considered in any such a framework. That is, the *spatial* and *temporal* characteristics of the network data.

Here, *spatial* characteristics refer to the unique semantic relationships between the values in two identical or related fields. For example, if we wish to cluster tuples of data that included IP addresses and port numbers, we would have two obvious ways of accomplishing the task: (1) treat the IPs and ports as numeric values (*i.e.,* integers) and use subtraction to calculate their distance, or (2) treat them as discrete values with no relationship to one an-

other (*e.g.,* in a probability distribution). Clearly, neither of these two options is exactly correct, since the network protocols that define these data types also define unique semantic relationships.

Temporal characteristics, on the other hand, describe the causal relationships among the network activities over time. One way to capture temporal information is to examine short n-grams or k-tuples of network activities. However, this too may ignore important aspects of network data. As an example, changes in traffic volume may alter the temporal locality captured by these short windows of activity. Moreover, network data has no restriction on this temporal locality, which means that activities can have long-term causal relationships with each other, extending to minutes, hours, or even days. Successfully building robust notions of similarity that address these temporal characteristics mean addressing similarity over large time scales.

Contributions. In this paper, we explore these spatial and temporal properties in an effort to learn what impact they may have on the performance of automated network analysis methods. To focus our study, we examine the problem of classifying host behaviors, which requires both strong notions of semantically-meaningful behavior and causal relationships among these behaviors to provide a meaningful classification. Furthermore, we develop an example unified metric space for network data that encodes the unique spatial and temporal properties of host behavior, and evaluate our proposed metric by comparing it to one that ignores those properties. We note that it is not our intention to state that current analysis methodologies are "wrong," nor that we present the best metric for computing similarity. Instead, we explore whether general metrics for network activities can be created, how such metrics might lead to improved analysis, and examine the potential for new directions of research.

More concretely, we begin by defining metric spaces for a subset of data types commonly found within network data. We design these metric spaces to encode the underlying semantic relationship among the values for the given data type, including non-numeric types like IP addresses and port numbers. Then, we show how to combine these heterogeneous metric spaces into a unified metric space that treats network data records (*e.g.,* packets, network flows) as points in a multi-dimensional space. Finally, we describe host behaviors as a time series of these points, and provide a dynamic time warping algorithm that efficiently measures behavioral distance between hosts, even when those time series contain millions of points. In doing so, we develop a geometric interpretation of host behavior that is grounded in semantically-meaningful measures of behavior, and the long-term temporal characteristics of those behaviors. Wherever possible, we bring to light several challenges that arise in the development of general metrics for network data.

To evaluate our proposed metric, we use a dataset containing over 30 million flows collected at the University of Michigan. Our experiments compare the performance of our metric to that of the L_1 (*i.e.,* Manhattan distance) metric, which ignores semantics and temporal information, in a variety of cluster analysis tasks. The results of these experiments indicate that semantic and temporal information play an important role in capturing a realistic and intuitive notion of network behaviors. Furthermore, these results imply that it may be possible to treat network data in a more rigorous way, similar to traditional forms of numeric data. To underscore this potential, we show how our metric may be used to measure the privacy provided by anonymized network data in the context of well-established privacy definitions for real-valued, numeric data; namely, Chawla et al. 's (c, t)-isolation definition [6].

2 Related Work

There is a long, rich body of work related to developing automated methods of network data analysis. These include, but are not limited to, supervised [18, 8, 2, 30, 9, 31] and unsupervised [17, 15, 29, 14, 10, 32, 3] traffic classification, anomaly and intrusion detection [11, 28, 27, 24, 13], and data privacy applications [7, 25, 16, 4]. Rather than attempt to enumerate the various approaches to automated data analysis, we instead point the interested reader to recent surveys of traffic classification [22] and anomaly detection techniques [5]. Also, we reiterate that the purpose of this paper is to explore the role of semantics and temporal information in developing a framework for defining similarity among network activities – a task that, to the best of our knowledge, has not been thoroughly examined in the literature thus far.

Of the network data analysis approaches proposed thus far, the work of Eskin et al. [11] is most closely related to our own. Eskin et al. address the problem of unsupervised anomaly detection by framing it as a form of outlier detection in a geometric space. Their approach uses kernel functions to map arbitrary input spaces to a high-dimensional feature space that encodes the distances among the input values. They show how to use these kernel functions to encode short windows of k system calls for host-based anomaly detection, and fields found

within network data for network-based detection.

Clearly, Eskin et al. share our goal of describing a unified metric (*i.e.,* feature) space for measuring the similarity of network data. However, there are two very important distinctions. First, while Eskin et al. show how to map non-numeric types (*e.g.,* IP addresses) to a common feature space, they do so in such a way that all semantic relationships among the values is removed. In particular, their kernel function treats each value as a discrete value with a binary distance measure: either the value is the same or it is different. Second, their approach for encoding sequences of activities (*i.e.,* system calls in their case) is limited to relatively short windows due to the exponential growth in dimensions of the feature space and the sparsity of that space (*i.e.,* the "curse of dimensionality"). By contrast, we seek to explore the semantic and temporal information that is missing from their metrics by developing semantically-meaningful spatial metrics and long-term temporal metrics based on dynamic time warping of time series data.

3 Preliminaries

For ease of exposition, we first provide definitions and notation that describe the network data we analyze in a format-independent manner. We also define the concepts of metric spaces and product metrics, which we use to create a foundation for measuring similarity among network hosts.

Network Data Data describing computer network activities may come in many different forms, including packet traces, flow logs, and web proxy logs. Rather than describe each of the possible formats individually, we instead define network data as a whole in more abstract terms. Specifically, we consider all network data to be a database of m rows and n columns, which we represent as an $m \times n$ matrix. The rows represent individual records, such as packets or flows, with n fields, which may include source and destination IP addresses, port numbers, and time stamps. We denote the i^{th} row as the n-dimensional vector $\vec{\mathbf{v}}_i =< v_{i,1}, \ldots, v_{i,n} >$, and the database as $V =< \vec{\mathbf{v}}_1, \ldots, \vec{\mathbf{v}}_m >^T$. For our purposes, we assume a total ordering on the rows $\vec{\mathbf{v}}_i \leq \vec{\mathbf{v}}_{i+1}$ based on when the record was added to the database by the network sensor. Furthermore, we associate each column in the matrix (*i.e.,* field) with an atomic data type that defines the semantic relationship among its values. More formally, we define a set of types $T = \{t_1, \ldots, t_\ell\}$ and an injective function $F : [1, n] \rightarrow T$ that maps a column to its associated data type.

Metric Spaces To capture a notion of similarity among values in each column, we define a metric space for each data type in the set T. A *metric space* is simply a tuple (X, d), where X is a non-empty set of values being measured and $d : X \times X \rightarrow \mathbb{R}_+$ is a non-negative distance metric. The metric function must satisfy three properties: (1) $d(x, y) = 0$ iff $x = y$; (2) $d(x, y) = d(y, x)$; and (3) $d(x, y) + d(y, z) \geq d(x, z)$. We denote the metric for the type t_j as (X_{t_j}, d_{t_j}).

Given the metric spaces associated with each of the columns via the data type mapping described above, $(X_{F(1)}, d_{F(n)}), \ldots, (X_{F(n)}, d_{F(n)})$, we can define a *p-product metric* to combine the heterogeneous metric spaces into a single metric space that measures similarity among records as if they were points in an n-dimensional space. Specifically, the p-product metric is defined as $(X_{F(1)} \times \ldots \times X_{F(n)}, d_p)$, where $X_{F(1)} \times \cdots \times X_{F(n)}$ denotes the Cartesian product of the sets and d_p is the p-norm:

$$d_p(\vec{\mathbf{x}}, \vec{\mathbf{y}}) = (d_{F(1)}(x_1, y_1)^p + \cdots + d_{F(n)}(x_n, y_n)^p)^{\frac{1}{p}}$$

We note that metrics we propose are straightforward generalizations of well-known metrics [19]; hence, the proofs are omitted for brevity.

4 Metric Spaces for Network Data

To provide a foundation for measuring similarity among network hosts, we define a metric space that captures both the spatial and temporal characteristics of the host's behaviors as follows. We begin by defining metrics spaces that capture semantically rich relationships among the values for each data type found in the network data. For the purposes of this initial study, we restrict ourselves to providing example metrics for four prevalent data types: IP addresses, port numbers, time fields, and size fields. Next, we show how to combine these heterogeneous metric spaces into a single, unified metric space using a p-product metric and a novel normalization procedure that retains the semantic relationships of the constituent metric spaces. This allows us to treat each network data record as a point and capture the spatial characteristics of the host's behavior in a meaningful way. Finally, we model a host's temporal behavior as a time series of points made up of records associated with the host (*e.g.,* flows sent or received by the host), and show how dynamic time warping may be used to efficiently measure distance between the behavior of two hosts.

4.1 Data Types and Metric Spaces

Network data may contain a wide variety of data types, each with its own unique semantics and range of possible values. That said, without loss of generality, we can classify these types as being numeric (*e.g.,* timestamps, TTL values, sizes) or categorical (*e.g.,* TCP flags, IPs, ports) in nature. The primary distinction between these two categories of types is that numeric types have distance metrics that naturally follow from the integer or real number values used to represent them, while categorical types often do not maintain obvious linear relationships among the values. Here, we describe example metric spaces for two data types in each category: time and size as numeric types, and IP and port as categorical.

Numeric Types. The time and size data types contain values syntactically encoded as 16 or 32 bit integers, and have semantics that mirror those of the integers themselves. That is, a distance of ten in the integers is semantically the same as ten bytes or ten seconds[1]. Both types can be represented by metric spaces of the form $(X = \{0, \ldots, M\}, d(x, y) = |x - y|)$, where M is simply the largest value possible in the encoding for that type. In most cases, that means that $M = 2^{32} - 1$ for 32-bit integers, or $M = 2^{16} - 1$ for 16-bit.

An obvious caveat, however, is that one must ensure that values are encoded in such a way that they are relevant to the analysis at hand. For example, when comparing host behavior over consecutive days, it makes sense to ensure time fields are made relative to midnight of the current day, thereby allowing one to measure differences in activity relative to that time scale (*i.e.,* a day) rather than to a fixed epoch (*e.g.,* UNIX timestamps).

Categorical Types. While metric spaces for numeric types are straightforward, categorical data types pose a problem in developing metric spaces due to the non-linear relationship between the syntactic encoding (*i.e.,* integers) and the underlying semantic meaning of the values. Moreover, the unique semantics of each data type prohibits us from creating a single metric for use by all categorical types. Instead, we follow a general procedure whereby a hierarchy is defined to represent the relationships among the values, and the distance between two values is determined by the level of the hierarchy at which the values diverge. While this is by no means the only approach for defining metrics for categorical types, we believe it is a reasonable first step in representing the semantics of these complex types.

To define the distance hierarchy for the port type, we decompose the space of values (*i.e.,* port numbers) into groups based on the IANA port listings: well-known (0-1023), registered (1024-49151), and dynamic (49151-65535) ports. In this hierarchy, well-known and registered ports are considered to be closer to one another than to dynamic ports since these ports are typically statically associated with a given service. Of course, ports within the same group are closer than those in different groups, and the same port has a distance of zero. The hierarchy is shown in Figure 1(a). More formally, we create the metric space (X_{port}, d_{port}), where $X_{port} = \{0, \ldots, 65535\}$ and d_{port} is an extension of the discrete metric defined as:

$$d_{port}(x, y) = \begin{cases} 0 & if\ x = y \\ 1 & if\ \delta_{port}(x) = \delta_{port}(y) \\ 2 & if\ \delta_{port}(x) \in \{0,1\}\ \&\ \delta_{port}(y) \in \{0,1\} \\ 4 & if\ \delta_{port}(x) \in \{0,1\}\ \&\ \delta_{port}(y) \in \{2\} \end{cases}$$

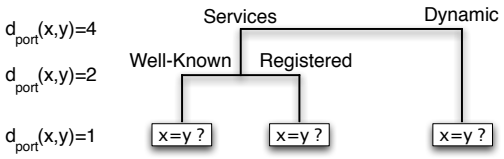
$$\delta_{port}(x) = \begin{cases} 0 & if\ x \in [0, 1023] \\ 1 & if\ x \in [1024, 49151] \\ 2 & if\ x \in [49152, 65535] \end{cases}$$

The choice of distances in d_{port} is not absolute, but must follow the metric properties from Section 3 and also faithfully encode the relative importance of the differences in groups that the ports belong to. One can also envision extending the hierarchy to include finer grained information about the applications or operating systems typically associated with the port numbers. Many such refinements are possible, but it is important to note that they make certain assumptions on the data which may not hold in all cases. For example, one could further refine the distance measure to ensure that ports 80 and 443 are near to one another as both are typically used for HTTP traffic, however there are clearly cases where other ports carry HTTP traffic or where ports 80 and 443 carry non-HTTP traffic. One of the benefits of this approach to similarity measurement is that it requires the analyst to carefully consider these assumptions when defining metrics.

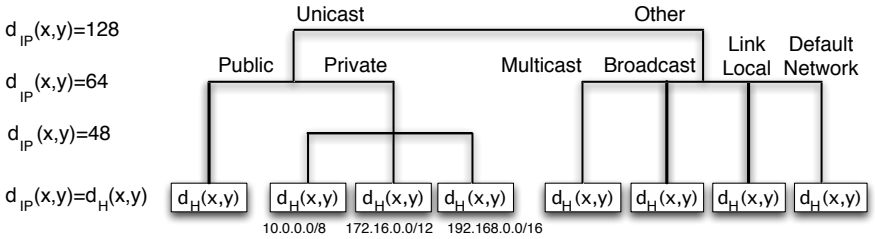
The metric space for IP addresses[2] is slightly more complex due to the variety of address types used in practice. There is, for instance, a distinction between routable and non-routable, private and public, broadcast and multicast, and many others. In Figure 1(b), we show the hierarchy used to represent the semantic relationships among these groupings. At the leaves of this hierarchy, we perform a simple Hamming distance calculation (denoted as d_H) between the IPs to determine distance within

[1]Note that we will address issues arising from different units of measure in the following section.

[2]The provided metric is for IPv4 addresses. IPv6 addresses would simply require a suitable increase in distances for each level to retain their relative severity.



(a) Distance hierarchy for port type.



(b) Distance hierarchy for IP type.

Figure 1. Distance metrics for categorical types of port and IP. Distances listed to the left indicate the distance when values diverge at that level of the hierarchy.

the same functional group. Due to the complexity of the hierarchy, we do not formally define its metric space here. We again reiterate that this is just one of potentially many ways to create a metric for functional similarity of IP addresses. Again, we may imagine a more refined metric that further subdivides IPs by autonomous system or CIDR block, and again we must ensure that that assumptions made by these metrics are supported by the data being analyzed.

4.2 Network Data Records as Points

Given the aforementioned distance metrics, the next challenge is to find a way to combine them into a unified metric space that captures the distance between records (*i.e.,* flows or packets) by treating them as points within that space. At first glance, doing so would appear to be a relatively simple procedure: assign one of the available types to each field in the record, and then combine the respective metric spaces using a p-product metric. However, when combining heterogeneous spaces it is important to normalize the distances to ensure that one dimension does not dominate the overall distance calculation by virtue of its relatively larger distance values. Typically, this is accomplished with a procedure known as affine normalization. Given a distance metric d on values $x, y \in X$, the affine normalization is calculated as:

$$\overline{d(x,y)} = \frac{d(x,y) - min(X)}{max(X) - min(X)}$$

However, a naive application of this normalization method fails to fairly weight all of the dimensions used in the overall distance calculation. In particular, affine normalization ignores the distribution of values in the underlying space by normalizing by the largest distance possible. As a result, very large and sparse spaces, such as IPs or sizes, may actually be *undervalued* as a result.

To see why, consider a field containing flow size data. In the evaluation that follows, the vast majority of flows have less than 100 bytes transferred, but the maximum seen is well over ten gigabytes in size. As a result, affine normalization would give extremely small distances to the vast majority of flow pairs even though those distances may actually still be semantically meaningful. In essence, affine normalization procedures remove or minimize the semantic information of distances that are not on the same scale as the maximum distance. Yet another approach might be to measure distance as the difference in rank between the values (*i.e.,* the indices of the values in sorted order). Doing so, however, will remove all semantic information about the relative severity of the difference in those values.

To balance these two extremes, we propose a new procedure that normalizes the data according to common units of measure for the data type. We begin by defining the overall range of distances that are possible given the values seen in the data being analyzed. Then, we divide this space into non-overlapping ranges based upon the unit of measure that most appropriately suits the values in the range. In other words, the range associated with a given unit of measure will contain all distances that have a value of at least one in that unit of measure, but less than one in the next largest unit. In this paper, we use seconds, minutes, and hours for the

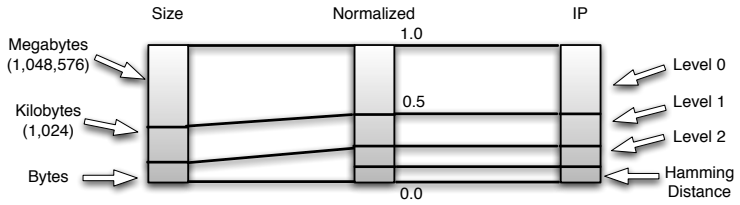


Figure 2. Example piecewise normalization of size and IP types. Mapping each range independently ensures they are weighted in accordance with the relative severity of the distance.

time data type, and bytes, kilobytes, and megabytes for size types. Categorical types, like IPs and ports, are assigned units of measure for each level in their respective distance hierarchies.

Once each distance range is associated with a unit of measure, we can then independently map them into a common (normalized) interval such that the normalized distances represent the relative severity of each unit of measure with respect to units from the same type, as well as those from other data types that are being normalized. It is this piecewise mapping to the normalized distance range that allows us to maintain the semantic meaning of the underlying distances without unduly biasing certain dimensions in the overall distance calculation. For simplicity, we map all metric spaces with three units of measure to the ranges $[0, 0.25)$, $[0.25, 0.5)$, and $[0.5, 1.0]$. Types with four units are mapped as $[0, 0.125)$, $[0.125, 0.25)$, $[0.25, 0.5)$,$[0.5, 1.0]$. Figure 2 shows how this mapping is achieved for size and IP types. Then, by denoting the function that produces the normalized distance for type t_j as $\overline{d_{t_j}(x, y)}$, we can use the p-product metric to define the distance between records in the network data as:

$$d_p(\vec{\mathbf{x}}, \vec{\mathbf{y}}) = \left(\overline{d_{F(1)}(x_1, y_1)}^p + \cdots + \overline{d_{F(n)}(x_n, y_n)}^p\right)^{\frac{1}{p}}$$

The metric space is now $(X_{F(1)} \times \cdots \times X_{F(n)}, d_p)$, and we set $p = 2$ in order to calculate the Euclidean distance between records.

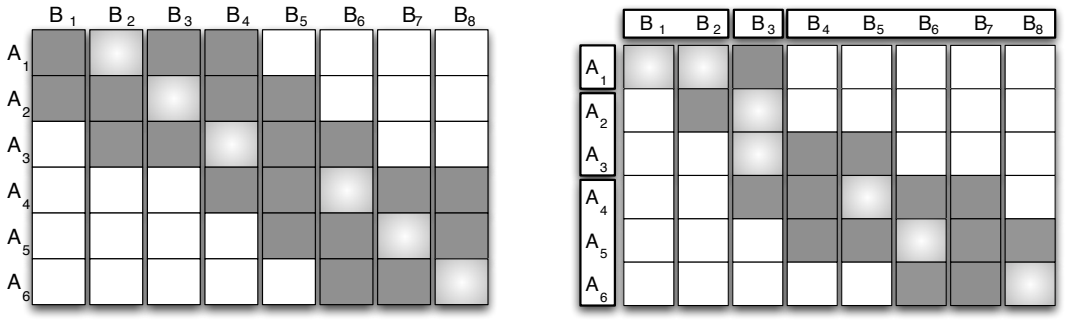
4.3 Network Objects as Time Series

Thus far, we have introduced distance metrics that attempt to capture the semantics that underlie various data types found in network data, and showed how that semantic information may be propagated to metrics for network data records by carefully normalizing and creating a unified metric space. The final step in our study requires us to take these records and show how to use the metrics we have developed to capture the similarity in behavior between two hosts. In effect, we defined a method for reasoning about the spatial similarity of individual records associated with a host, but must also address the concept of temporal similarity to capture its behavior in a meaningful way.

Common wisdom suggests that host similarity should be calculated by independently evaluating records associated with that host, or only examining short subsequences of activities. This is due primarily to the so-called "curse of dimensionality" and the sheer volume of behavioral information associated with each host. Our assertion is that doing so may be insufficient when trying to capture a host's activities and behaviors as a whole, since these behaviors often have strong causal relationships that extend far beyond limited n-gram windows.

In order to capture the entirety of a host's behavior and appropriately measure similarity among hosts, we instead model a host's behavior as a time series of points made up of the records embedded into the n-dimensional metric space described in the previous section. Given two hosts represented by their respective time series, a dynamic programming method known as *dynamic time warping* (DTW) may be used to find an order-preserving matching between points in the two series which minimizes the overall distance. The DTW approach has been used for decades in the speech recognition community for aligning words that may be spoken at different speeds. Essentially, the DTW procedure "speeds up" or "slows down" the time series to determine which points should be aligned based on the distances between those two points, which in our case are calculated using the p-product metric.

Unfortunately, DTW runs in $O(m_1 m_2)$ time and space for two time series of length m_1 and m_2, respectively. Considering that most real-world hosts produce millions of records per day, such an approach would quickly become impractical. Luckily, several heuristics have been proposed that limit the area of the dynamic programming matrix being evaluated. In particular, Sakoe and Chiba [26] provide a technique which reduces the computational requirements of the DTW process by evaluating only those cells of the matrix which lie within



(a) DTW with heuristic for constant sampling.

(b) DTW with heuristic for variable sampling.

Figure 3. Example dynamic programming matrices with the original Sakoe-Chiba heuristic (a) and our adapted heuristic (b) for variable sampling rate time series. Gradient cells indicate the "diagonals" and dark shaded cells indicate the window around the diagonal that are evaluated. In the the variable rate example, points in the same subsequence are grouped together.

a small window around the diagonal of the matrix. Despite only examining a small fraction of cells, their approach has been shown to produce accurate results in measuring similarity due to the fact that most high-quality matches often occur in a small window around the diagonal, and that it prevents so-called pathological warpings wherein a single point may map to large sections of the opposite time series.

4.3.1 Efficient Dynamic Time Warping for Network Data

Ideally, we would apply the Sakoe-Chiba heuristic to make the DTW computation feasible even for very large datasets. Unfortunately, the Sakoe-Chiba heuristic makes assumptions on the time series that prevent its direct application to network data. In particular, it assumes that the points in the time series are sampled at a constant rate, and so the slope of the diagonal that is evaluated in the matrix depends only on the length of the time series (*i.e.,* the time taken to speak the word). For network data, however, the rate at which records are produced is based on the traffic volumes sent and received by the host, which are inherently bursty in nature and dependent on the host's activities. Consequently, a single diagonal with a fixed slope would yield a warping path that attempts to align points that may be recorded at significantly different times, leading to a meaningless measurement of behavior.

To address this, we propose an adaptation where we break the two time series into subsequences based on the time period those sequences cover, and calculate individual diagonals and slopes for each subsequence. The individual subsequences can then be pieced together to form a warping path that accommodates the variable sampling rate of network data, and from which we can extend a window to appropriately measure similarity without evaluating the entire dynamic programming matrix. Figure 3 shows an example of the traditional Sakoe-Chiba heuristic and our adaptation for network data.

Our extension of the Sakoe-Chiba heuristic for network data proceeds as follows. Assume that we are given two time series A and B, where $|A| \leq |B|$. We begin by splitting the two time series being aligned into subsequences such that all points in the same subsequence were recorded during the same second (according to their timestamps). The subsequences in the two time series are paired if they contain points for the same second. Any subsequences that have not been paired are merged in with the subsequence in the same time series that is closest to its timestamp. Thus, we have k subsequences $A_1, \ldots, A_k$ and $B_1, \ldots, B_k$ for the sequences A and B, with A_i and B_i being mapped to one another. Next, we iterate through each of the k subsequence pairs and calculate the slope of the pair as $S_i = \frac{|B_i|}{|A_i|}$. The slope and subsequence mapping uniquely determine which cells should be evaluated as the "diagonal" in our variable sampling rate adaptation. Specifically, the j^{th} point in the i^{th} subsequence (*i.e.,* $A_{i,j}$) is mapped to $B_{i,j'}$ where $j' = \lceil j * S_i \rceil$.

With this mapping among points in hand, we then place a window around each cell of length at least $\lceil S_i \rceil$ to ensure continuity of the path in the dynamic programming matrix (*i.e.,* a transition exists between cells). For those points that occur at the beginning or end of a subsequence, the window length is set based on the index of the mappings for the adjacent points to ensure that the subsequences are connected to one another. In order to provide a

Algorithm 1
Dynamic Time Warp $(A, B, map, slope, c)$

Require: $|A| \leq |B|$
Require: $c > 0$
  Initialize m as a $|A| \times |B|$ matrix
  $m[i][j] \leftarrow \infty$ for all i, j
  $m[0][0] \leftarrow 0.0$
  for $i \leftarrow 0$ to $|A|$ **do**
    $start \leftarrow max(0, map[i] - slope[i] * c)$
    $end \leftarrow min(|B|, map[i] + slope[i] * c)$
    for $j \leftarrow start$ to end **do**
      if $i \neq 0$ and $j \neq 0$ **then**
        $distance \leftarrow d_p(A[i-1], B[j-1])$
        $left \leftarrow matrix[i][j-1] + distance$
        $up \leftarrow matrix[i-1][j] + distance$
        $diag \leftarrow matrix[i-1][j-1] + distance$
        $m[i][j] \leftarrow min(left, up, diag))$
      end if
    end for
  end for
  return $m[|A|][|B|]/(|A| + |B|)$

tunable trade-off between computational complexity and accuracy of the warping, we extend the window by a multiplicative parameter c so that $c * S_i$ cells on either side of the "diagonal" are evaluated.

Algorithm 1 shows the dynamic time warping procedure using the restricted warping path from our variable sampling rate heuristic[3]. For ease of presentation, we provide as input to the algorithm lists containing the two time series A and B, the mapping between indices in A and their "diagonal" mapping in B, a list containing the slope values to be used to ensure continuity of the matrix $slope$[4], and the multiplicative factor c that controls the number of cells evaluated around the "diagonals." As with any dynamic programming technique, the minimum distance for the allowable warping path is given in the lower-right cell of the matrix. Furthermore, in order to ensure the computed warping distances are comparable among time series of varying lengths, we normalize the distance by dividing by the number of cells in the longest possible warp path $|A| + |B|$, which also ensures that the distance remains symmetric. We use this normalized distance to compare the behaviors of hosts and determine their similarity. The computational complexity of our proposed heuristic is $O(|B| * c)$ in the worst case where both series A and B are of equal length, which represents a significant reduction from the quadratic complexity required when evaluating the entire dynamic programming matrix.

[3]Note that this algorithm can be easily altered so that only two rows in the matrix are maintained in memory at any given time, thereby reducing the space complexity to $O(|B|)$.

[4]In practice, there are actually two lists – a forward and backward list – to accommodate for cases where the point is at the beginning or end of a subsequence.

5 Evaluation

The underlying hypothesis behind the behavioral distance metric proposed in the previous section is that the semantics of the network data and long-term sequences of activities provide a more robust notion of host behavior than measures that ignore such information. To evaluate this hypothesis, we compare our proposed dynamic time warp (DTW) metric to the well-known L_1 distance metric, which does not explicitly capture semantic and sequencing information. The experiments in our evaluation compare these two metrics using a broad range of analysis methodologies on a large dataset collected on a live network segment in an effort to pinpoint the benefits of our approach over naive metrics.

We begin by measuring the trade-off between speed and accuracy of our dynamic time warping-based distance calculation under varying settings of the multiplicative window parameter c. Using the results from this experiment, we choose a window that provides a balance between fidelity and speed, and show that our approach is feasible even on datasets containing millions of flows. Next, we compare our approach to a metric that ignores semantics and sequencing. In particular, we examine the clusters produced by a variety of cluster analysis techniques when using L_1 distance and our proposed DTW metric. Moreover, we look at the consistency of these behaviors across consecutive days of observation to show that our approach captures the most robust notion of network behaviors. Finally, we examine a concrete application of our metric to the problem of quantifying privacy of network hosts, and show how it enables application of well-known privacy definitions, such as the notion of (c, t)-isolation defined by Chawla et al. [6].

Data. In our evaluation, we make use of a dataset containing uni-directional flow records collected within the Computer Science and Engineering department of the University of Michigan. The CSE dataset, as we refer to it throughout this section, was collected over two consecutive twenty-four hour periods, and captures all local and outbound traffic from a single /24 subnet. The data contains 137 active hosts that represent a broad mix of network activities, including high-traffic web and mail servers, general purpose workstations, and hosts running specialized research projects. The relatively small number of hosts allows us to manually verify the behavioral information with system and network administrators at the University of Michigan. Table 2 provides a summary of the traffic within the CSE dataset on each of the observed days.

Window Parameter c	Avg. Time per Warp in seconds (σ)	Avg. Distance Increase in percentage (σ)
25	1.0 (5.5)	5.0% (6.2%)
50	1.8 (10.1)	3.3% (4.5%)
100	3.2 (19.3)	1.9% (3.1%)
200	5.5 (36.2)	0.9% (1.9%)
500	11.1 (79.7)	0.0% (0.0%)

Table 1. Time vs. accuracy comparison, including averages and standard deviations for each window parameter tested. Accuracy measured as percentage increase in distances from window parameter $c = 500$.

	Day 1	Day 2
Number of Hosts:	135	137
Total Flows:	14,400,974	16,249,269
Avg. Flows per Host:	106,674	118,608
Median Flows per Host:	4,670	2,955
Max Flows per Host:	6,357,810	6,620,785

Table 2. Traffic properties for both days of the University of Michigan CSE dataset.

Throughout all of our experiments, we consider six fields within the flow log data: start time, flow size, source IP and port, and destination IP and port. These fields are associated with the appropriate distance metrics and used to measure behavioral distance according to the dynamic time warping procedure outlined in the previous section, except for instances where we explicitly make use of an alternate distance metric. Furthermore, we remove all non-TCP traffic from the dataset in an effort to minimize noise and backscatter caused by UDP and ICMP traffic. We note that since we are examining flow data, the use of TCP traffic should not unfairly bias our analysis since causal relationships among the flows will likely be dictated by application-layer protocols or other higher-order behaviors of the host.

Finally, we performed an initial analysis of the data using k-means clustering in combination with our DTW metric to pinpoint a variety of scanning activities, including pervasive Nessus scans [21] from two hosts within the CSE network and several IPs from China performing fingerprinting on CSE hosts. These scanning hosts were removed from the data to ensure that we focus our analysis on legitimate forms of network behaviors that may be verified via the University of Michigan CSE department's system and network administrators. The specifics of this technique are discussed in Section 5.2, but the fact that our method was able to pinpoint these scanning activities is interesting in and of itself.

5.1 Efficiency

Network datasets collected on real-world networks may contain millions of records. As such, it is important to understand the efficiency of our proposed metric and its ability to reasonably operate on large datasets. Moreover, we must also understand the impact of choosing the window parameter c on the accuracy of the resultant distance and the speed with which it is calculated. To do so, we choose five settings of the window parameter (25, 50, 100, 200, 500) and run distance measures among a random subset of 50% of the hosts from each day of the CSE dataset. The hosts in the sample selected for our experiment had an average of 63,984 flows each, with the largest host having 6,357,811 flows. For each of the window parameter settings, we measure the time it takes to perform the DTW procedure on a single 3.16GHz processor and the increase in distance from that which was calculated by the largest parameter setting (*i.e.,* $c = 500$), which are shown in Table 1.

The results of our efficiency experiments show that most parameter settings can perform DTW-based behavioral distance measures in a few seconds with relatively small changes to the overall distance, even when calculating distances among hosts with thousands or millions of flows. For example, with a parameter setting of $c = 100$, the DTW metric takes an average of only 3.2 seconds with an increase in the calculated distance of just under 2%. For the remainder of our evaluation, we fix the window parameter $c = 100$ since it appears to provide an appropriate trade-off between distance calculation accuracy and speed.

5.2 Impact of Semantics and Causality

To evaluate the potential benefits of semantics and long-term causal information in behavioral metrics, we compare the performance of our DTW metric to the L_1 distance metric. In our experiments, the L_1 metric operates by creating distributions of values for each of the six fields, calculating the L_1

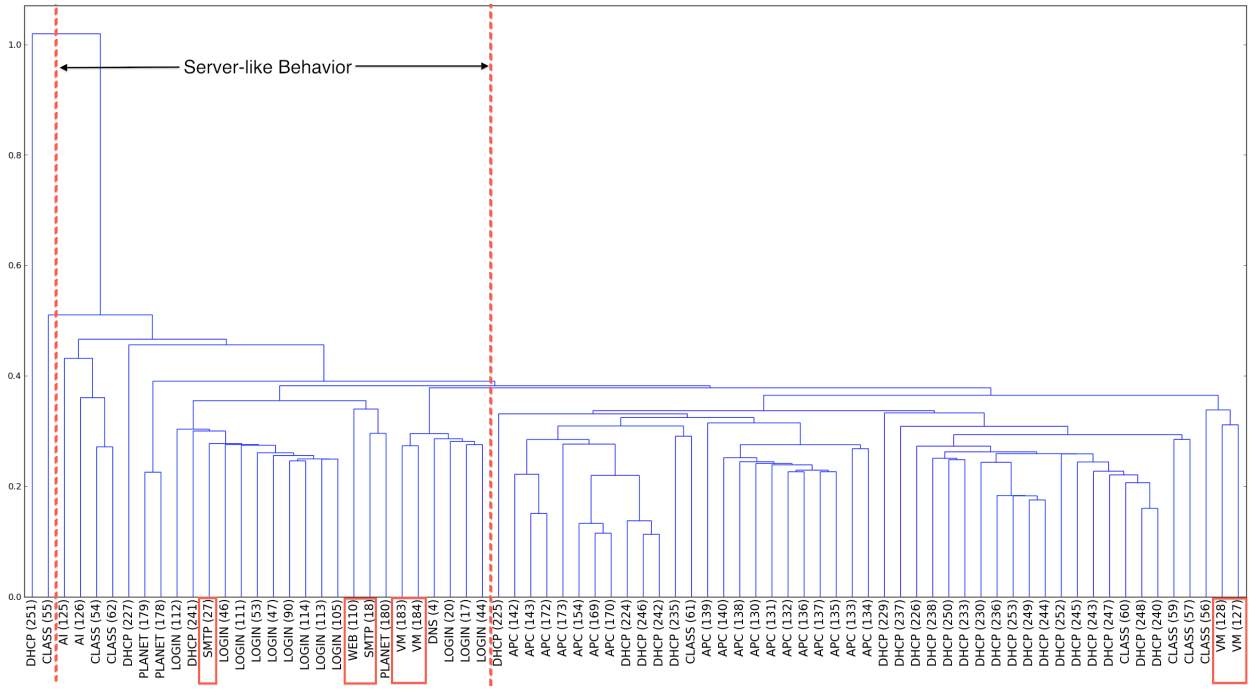


Figure 4. Dendrogram illustrating agglomerative clustering using DTW-based metric on hosts in the first day of the CSE dataset.

distance between the two hosts' respective distributions, and summing the distances. The DTW metric operates exactly as discussed in Section 4.

Our evaluation is broken into three parts. In the first, we use single-linkage agglomerative (*i.e.,* hierarchical) clustering to visualize and examine the behavioral similarity among all hosts in our experiments. The second experiment uses k-means clustering to explore the ability of the two metrics to produce clusters with coherent semantics. The final experiment compares the consistency of the clusters produced by the above techniques, as well as the similarity of the hosts across consecutive days to measure the robustness of the behavioral information captured by the two metrics. In the first two experiments, we examine only the first day of traffic from the CSE dataset, while both days are used in the final experiment.

In all of these experiments, we make use of information obtained from the University of Michigan CSE system and network administrators about the known usage of the hosts in our analyses to provide a general notion of the correctness of the clustering and to highlight specific cases for deeper inspection. This information allows us to label the hosts in our data with one of ten labels that describe the stated usage of the host when its IP was registered with the computer support staff. These labels include: web server (WEB), mail server (SMTP), DNS server (DNS), a variety of host types involving general client activities (LOGIN, CLASS, DHCP), specialized research hosts for PlanetLab (PLANET), an artificial intelligence research project (AI), and auxiliary power units (APC). For the purposes of these clustering experiments, we only examine the subset of hosts that we have labels for (76 of the 137 hosts).

Agglomerative Clustering. The agglomerative clustering of hosts using the DTW and L_1 metric are shown as dendrograms in Figures 4 and 5, respectively. The dendrograms visualize the agglomerative clustering process, which begins with each host in its own cluster and then merges clusters iteratively with their nearest neighbor. The leaves in the dendrogram are labeled with the stated usage of the host obtained from system administrators and a unique identifier to facilitate comparison between dendrograms. The branches of the dendrogram illustrate the groupings of hosts, with shorter branches indicating higher levels of similarity.

At first glance, we see that both DTW and L_1 metrics group hosts with the same label in fairly close groups. In fact, it appears as though L_1 distance actually produces a better clustering according to these labels, for instance by grouping SMTP servers, PlanetLab hosts, and VM servers in very tight groupings. However, there are two

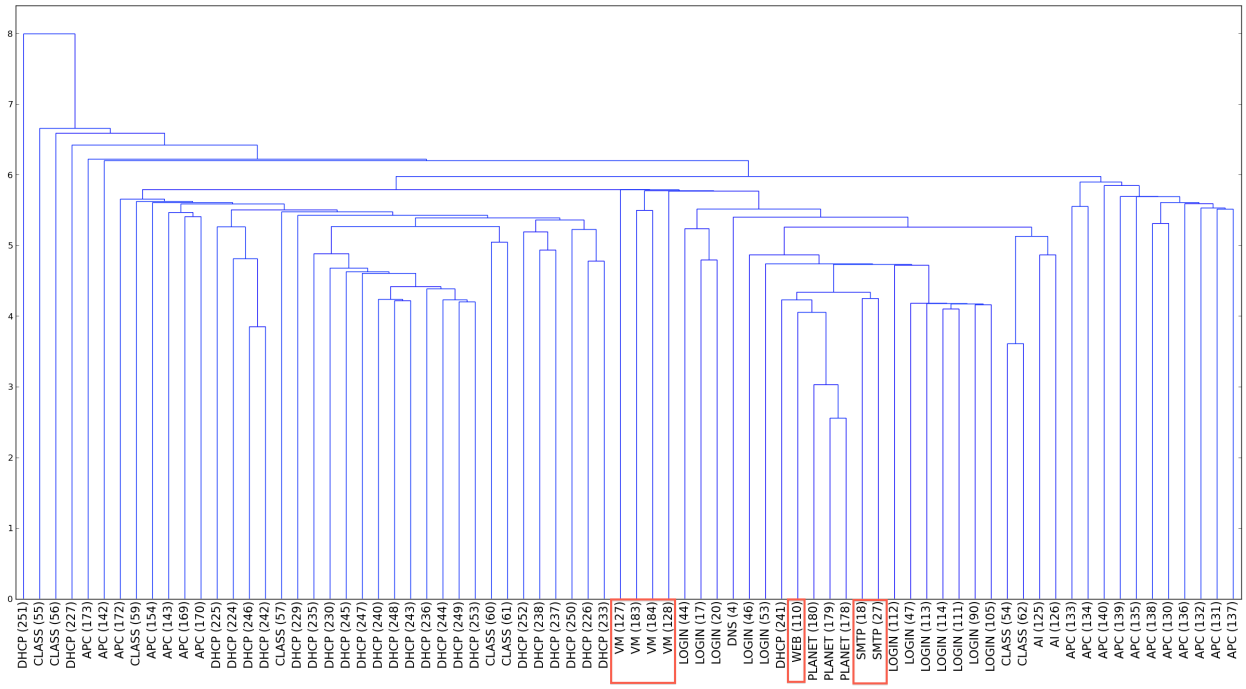


Figure 5. Dendrogram illustrating agglomerative clustering using L_1 distance on hosts in the first day of the CSE dataset.

subtle shortcomings that require deeper investigation. First, the L_1 dendrogram clearly indicates that there is relatively little separability among the various clusters, as evidenced by the small differences in branch lengths in the dendrogram from distances of 5 to 6.5. The impact of this is that even small changes in the underlying distributions will cause significant changes to the groupings. We will investigate this particular shortcoming during our consistency experiments. The second shortcoming is that while the labels provide a general idea of potential usage of the hosts, they say nothing about the actual activities being performed during recording of this dataset. As such, it is quite possible that hosts with the same labels can have wildly different behaviors.

To more closely examine the clustering provided by DTW and L_1 metrics, we manually observe two groups of hosts (highlighted in Figures 4 and 5): virtual machine hosts (VM) and major servers (SMTP and WEB). In the L_1 dendrogram, all VM hosts are grouped together, whereas in the DTW dendrogram the hosts are grouped into pairs. Moreover, these pairs are on opposite sides of the dendrogram indicating significant differences in behavior. When we manually examine these hosts' activities, we see that one pair (127,128) appear to be performing typical client activities, such as outgoing SSH and web connections, with only approximately 2,000 flows each. The other pair (183,184), however, had absolutely no client activities, and instead most of the approximately 6,000 flows consisted of VMWare management traffic or Nagios system monitoring traffic [20]. Clearly, these are very distinct behaviors – client activity and basic system management activity – and yet the L_1 metric was unable to distinguish them.

For the primary servers in the dataset, the L_1 distance metric groups the two SMTP servers together, however our DTW metric ends up grouping one of the SMTP servers (18) with the web server while the second SMTP server is much further away. Upon closer examination, the SMTP servers are differentiated by the fact that one server (18) is the primary mail server that receives about five times the amount of the traffic as the second server (27). In addition, the first server (18) receives the majority of its connections from IPs external to the CSE network, while the second server (27) receives many of its mail connections from hosts within the CSE IP prefix and has a significant number of connections from hosts performing SSH password dictionary attacks. It is this local vs. external preference that causes the first server (18) to be grouped with the web server, since the web server also has a significant amount of traffic, almost all of which is associated with external hosts. Naively, the L_1 clustering appears to make the most sense given these labels, but again its lack of semantic and causal informa-

		DTW Metric	L_1 Distance
Cluster 1	Hosts/Flows:	25 hosts/139.2 flows	12 hosts/165.8 flows
	Host Types:	APC 72%, DHCP 24%, CLASS 4%	APC 50%, DHCP 41%, CLASS 9%
	Activities:	$< SPORT = \{22, 6000\} >$ $< SPORT = 21, DADDR =$Chinese IP$>$ $< SPORT = 443, DADDR =$UMich DNS$>$	$< SPORT = \{22, 6000\} >$ $< SPORT = 21, DADDR =$Chinese IP$>$
Cluster 2	Hosts/Flows:	25 hosts/1,166.0 flows	18 hosts/908.2 flows
	Host Types:	DHCP 72%, CLASS 16%, VM 8%, AI 4%,	DHCP 89%, CLASS 11%
	Activities:	$< SPORT = \{22, 80\} >$	$< SPORT = 22 >$
Cluster 3	Hosts/Flows:	26 hosts/539,438 flows	46 hosts/305,211 flows
	Host Types:	LOGIN 46%, PLANET 11%, CLASS 11%, VM 8%, SMTP 8%, WEB 4%, AI 4%, DNS 4%, DHCP 4%	APC 26%, LOGIN 26%, CLASS 10%, DHCP 8%, VM 8%, PLANET 6%, AI 4%, SMTP 4%, WEB 2%, DNS 2%
	Activities:	$< SPORT = \{22, 25, 80, 111, 113, 993\} >$ $< SPORT = 5666, DADDR =$UMich DNS$>$	$< SPORT = \{21, 22, 80, 111\} >$ $< SPORT = 5666, DADDR =$UMich DNS$>$ $< SPORT = 443, DADDR =$UMich DNS$>$

Table 3. k-Means clustering of hosts in the first day of the CSE dataset using DTW and L_1 metrics. Activities represent the dominant state profiles obtained from applying our extension to the Xu et al. dominant state algorithm [32].

tion has caused it to ignore the actual behaviors.

As a side effect of our close examination of some hosts within the dendrogram, we also gained some insight into the general structure of the dendrogram in the case of the DTW metric. That is, the DTW metric produced a dendrogram where the hosts on the right side of the dendrogram perform general client activities, while most hosts on the left side act as servers. This separation is illustrated by vertical dotted lines in Figure 4. The DTW dendrogram is further refined by the type of hosts (internal vs. external) that they communicate with, the volume of traffic, and other interesting behavioral properties.

k-Means Clustering. Another way to evaluate the performance of the DTW and L_1 metric is to consider if the clusters produced maintain some level of behavioral coherence, and whether the groupings are similar to those that might be produced by a human network analyst looking at the same data. To examine these properties, we use k-means++ clustering [1] to produce clusters from each of the distance metrics and then examine the properties of the resultant clusters. In particular, we use the dominant state analysis technique of Xu et al. [32] to characterize the dominant behaviors in each cluster, examine the distribution of host labels in each cluster, and manually examine a random sample of each cluster to determine the overall behavior being captured.

While a full discussion of the Xu et al. dominant state algorithm is beyond the scope of this evaluation, we provide a high-level notion of its operation and how we extend it to capture behaviors of groups rather than individual hosts. The algorithm begins by calculating the normalized entropy of each field being analyzed, and then ordering those fields from smallest entropy value to greatest. The algorithm then selects all values from the smallest entropy field whose probability are greater than a predetermined threshold t, thereby creating "profiles" for each value. Then, each of those profiles is extended by examining values in the next smallest entropy field whose joint probability of occurrence with the profile values is above the threshold t. The profiles are extended iteratively until no new values may be added from the current field, at which point they are considered to be final profiles.

In the original paper, Xu et al. defined the distribution of values on a per-host basis to quickly determine host activities. To apply the same procedure to groups of hosts, we apply the dominant state algorithm in two levels. At the first level, we run the algorithm on distributions of values for each host in the cluster. This produces dominant state profiles for each of those hosts. From these profiles, we extract the values for each of the present fields to create new distributions. These distribution represent the probability of a value occurring in the dominant state profiles for the hosts in the current cluster. Finally, we apply the dominant state algorithm to these cluster-wide distributions to extract out the profiles that occur most consistently among all hosts in the clusters.

For this experiment, we manually examined a sample of hosts throughout the entire dataset and determined that there were, roughly, three high-level classes of activities: server activities, client ac-

tivities, and noise/scanning activities. This manual classification was supported by the results of our agglomerative clustering analysis, which showed several levels of increasingly subtle behavioral differences within each of these three classes. Given this rough classification of behavior, we set $k = 3$ and see if the resultant clustering indeed captured the intuitive understanding of the activities as determined by a human analyst. Table 3 shows the break down of the three clusters produced by the k-means++ algorithm using DTW and L_1 metrics. The results of the DTW metric clustering shows that the clusters are indeed broken into the three classes of activity found via manual inspection. Cluster 1, which contains primarily the power supply devices (APC) and DHCP hosts, had significant scanning activity from IP addresses in China, and relatively low traffic volumes indicating only sporadic use. By comparison, hosts in Cluster 2 exhibit traditional client behaviors of significant SSH and web browsing activities. Finally, Cluster 3 contains hosts with significant server-like activities. For instance, the hosts related to the artificial intelligence research project (AI) were found to be running web servers that provide statistics to participants in the project, and most of its activity is made up of web requests.

The L_1 metric, on the other hand, produced somewhat incoherent clusters. While there is some overlap in the clusters and observed behaviors, it is clear by the distribution of host types that these clusters mix behaviors. The most prominent example of this is that many of the power supply hosts (APC) that we verified as have low traffic volumes and scanning activity were grouped in with the server cluster (Cluster 3). Moreover, the client cluster (Cluster 2) contains many of the DHCP hosts with scanning activity, which in turn alters the behavioral profile for that cluster to remove web activities. The results of this experiment certainly indicate that the L_1 metric simply does not capture a coherent notion of behavior. Moreover, when the number of clusters is increased, the differences between the DTW and L_1 metrics become more significant. That is, the L_1 metric continues to mix behaviors, while the DTW metric creates clusters that represent increasingly fine-grained behaviors, such as the preference for communicating with internal versus external hosts.

As mentioned earlier in this section, this clustering approach was also used to filter a large portion of the scan traffic found in the dataset. Specifically, when we first ran this experiment, we found that the behavioral profiles produced by Cluster 1 were consistent with a wide range of scan traffic, and that the number of hosts in that cluster were significantly larger than expected. We were then able to use the behavioral profiles to remove traffic from scanning IPs and produce a much cleaner clustering without most of the initial scanning activity. As you can see by the scanning IP from China found in the profile of Cluster 1, it appears that we can continue performing this clustering approach to iteratively identify increasingly subtle scanning activity.

Changes in Behavior Over Time. Perhaps one of the most important properties of any behavioral metric is its robustness to small changes in underlying network activity. That is, we want any changes in the measured distance to be related to some high-level behavioral change and not minute changes in the specifics of the traffic. As our final experiment, we use both days of traffic in the CSE dataset to determine the sensitivity of the DTW and L_1 metrics to changes in behavior over time. To do so, we take the set of all hosts that occur in both days (as determined by IP address), and compare their day one time series to those of all hosts in day two. This produces a list of distances for each host in the day one data to the day two hosts. With consistent behavior and a a behavioral metric that is robust to minute changes in activity, we would expect to find that each day one host is closest to itself in the day two data. Of course, there are also instances where host behavior did significantly change between the two observation periods. Therefore, our analysis looks at both the number of hosts within each label class whose behaviors appear to remain the same and the reasons that some hosts' behaviors apparently change.

To begin, we provide a summary of the above consistency experiment for each host label when using the DTW and L_1 metrics in Table 4. The table shows two values: the number of hosts with perfect consistency and the average consistency rank of all hosts in the group. By perfect consistency, we mean hosts in day one whose closest host in day two is itself. Consistency rank refers to the rank of the day one host in the sorted list of day two distances. Ideally, if the behaviors of the hosts in the group are exactly the same we would have all hosts with perfect consistency and an average rank of 1.0. The most obvious observation we can make from the results of this experiment is that the L_1 metric appears to be more brittle than DTW with a significantly greater average rank and fewer perfect consistency hosts overall. What is more interesting, however, are the cases where DTW indicates change in behavior and L_1 does not, and vice versa.

For the case where DTW indicates change and L_1 does not, we again examine the two SMTP servers

Host Type	Num. Hosts	DTW Metric		L_1 Distance	
		Num. Perfect Consistency	Avg. Rank	Num. Perfect Consistency	Avg. Rank
WEB	1	1	1.0	1	1.0
DNS	1	1	1.0	0	17.0
PLANET	3	2	1.3	3	1.0
VM	4	2	1.5	0	41.3
LOGIN	12	8	2.1	8	14.2
SMTP	2	1	3.5	2	1.0
APC	18	2	16.2	0	39.9
AI	2	0	41.5	0	40.5
DHCP	24	2	48.4	1	31.8
CLASS	8	1	54.4	0	31.1
TOTAL	75	20	17.1	15	21.9

Table 4. Host behavioral consistency for hosts occurring in both days of the CSE dataset.

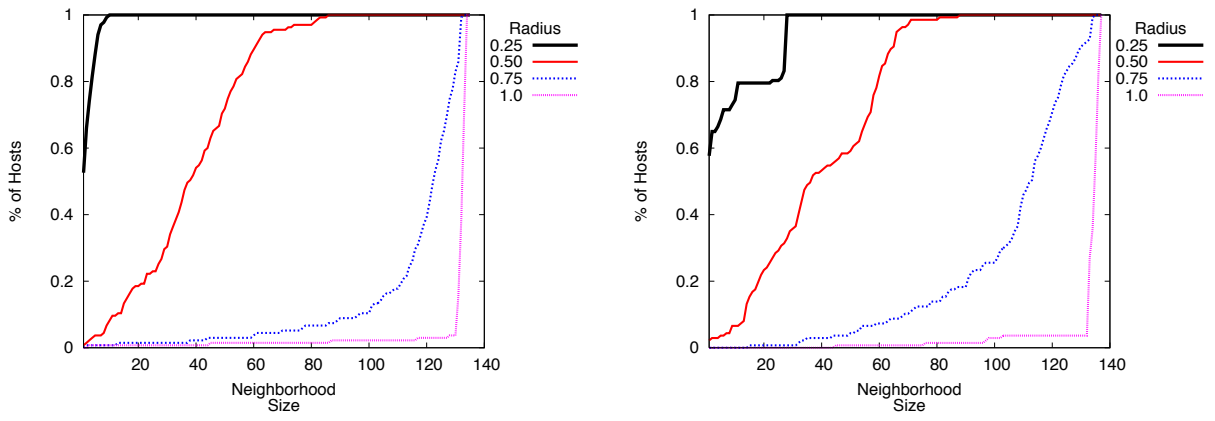
found in our data. Recall from the previous clustering experiments that one SMTP server in the first day of data acts as the primary server with many connections to external hosts, while the other receives many fewer connections and those are primarily to internal hosts. Upon closer inspection of the two servers' behaviors in day two, we find that the primary SMTP server's (18) activities are effectively unchanged. The secondary server (27), however, has a significant increase in the proportion of traffic that is related to mail activities. In particular, the secondary SMTP server (27) in the first day of data has a roughly even split between general mail traffic and an SSH password dictionary attack (*i.e.,* hundreds of consecutive SSH login attempts), while in the second day the SSH attacks stop almost completely and the general mail traffic to both internal and external hosts increase significantly. Intuitively, this does indeed indicate a change in behavior due to the presence of the SSH attack and change in mail activity, although the L_1 metric indicates that no such change took place.

Next, we look at the group of VM servers for the case where the L_1 metric indicates a change and the DTW metric does not. In this case, manual inspection of all four VM hosts indicates that there were no significant changes in traffic volume or activities for any host. The only noticeable change was that one of the client-like VMs (127) was port scanned for a few seconds late at night on the second day. Given this information, our DTW metric's ranking makes perfect sense – the two VMs running management protocols (183,184) were exactly the same as their previous day's activity, while the client-like VMs (127,128) got confused for one another in the previous day which is evidenced by the average ranking of 1.5 for that group (*i.e.,* rank of 1.0 for the management VMs, 2.0 for the client VMs). With the L_1 metric, not only did it confuse the VM behaviors for one another in the previous day, but it also confused it with dozens of other client-like hosts (*i.e.,* DHCP, LOGIN, etc.) and management hosts (*i.e.,* APC), thereby causing the significantly higher average rank for that group. In fact, the L_1 metric only seems to achieve a lower average rank in groups where changes in behavior are expected, such as the DHCP, APC, or CLASS groups, and where the the rankings are likely to improve by chance due simply to tiny changes in activities.

5.3 Applications

To conclude our evaluation of the proposed DTW behavioral metric, we discuss its potential applications. Certainly, the fact that our proposed metric provides a robust, semantically meaningful notion of host behavioral similarity means that it may be a good foundation for a number of network analysis tasks, such as anomaly detection or host classification. In fact, the cluster analysis technique described above can be thought of as a form of host classification and an anomaly identification method. There are, however, some non-obvious forms of network data analysis which may benefit from our more formalized approach. One such application is the use of the DTW metric as the basis for analyzing the privacy of hosts within anonymized network data.

Roughly speaking, anonymized network data is simply a transformation of data collected on a computer network such that certain sensitive information is not revealed to those who use the data, but the data remains generally useful to researchers and network analysts. This sensitive information includes learning the real identities of hosts found within the data despite those identities being replaced with a pseudonym (*e.g.,* prefix-preserving anonymization [23, 12]). One of the most difficult parts of achieving network data anonymization is the lack of applicable privacy definitions upon which these transformations can be based, due primarily to the fact that rigorously defining the behavior of hosts and the ways these behaviors may change remains an open problem. Although prior works (including our own) have attempted to define privacy analysis techniques for anonymized data



(a) Day 1 (b) Day 2

Figure 6. Privacy neighborhood sizes for different distance radius settings.

[7, 25], their methods have been based upon techniques like the L_1 distance metric evaluated above, and consequently are likely to provide a rather loose privacy analysis. We argue that given the DTW metric's unique ability to capture a robust notion of host behavior and the fact that it embeds this behavior in a well-defined metric space, it holds good promise in bringing formal privacy definitions to the field of anonymized network data.

We illustrate this point by examining one way in which a privacy analysis for network data may be built around the definition of (c, t)-isolation put forth by Chawla et al. in the context of multidimensional, real-valued spaces [6]. This privacy notion essentially states that any "anonymized" point should have t real points from the original data within a ball of radius proportional to c centered at the anonymized point. That is, each anonymized point should have a conserved neighborhood of real points that it may "blend in" with, and therefore provide the potential adversary with some level of uncertainty about the point's real identity. The definition can be used as the basis of an analysis methodology simply by looking at the distribution of neighborhood sizes for each anonymized point at increasing radius sizes – such information may be used by a data publisher to, for instance, determine the relative risk of publishing the anonymized data in its current form. The distribution may be visualized via a cumulative distribution function that shows the percentage of points with a number of neighbors (*i.e.,* neighborhood size) less than the given value for a specified radius.

To adapt the definition and analysis methodology for network hosts, we simply consider the entire time series for the hosts to be a "point" and use the DTW metric to calculate the radius. Figure 6 shows examples of this privacy analysis methodology applied to hosts within the the two days of our CSE dataset under the assumption that none of the six fields in our metric have been altered by the anonymization process. One way of interpreting this analysis is that the radius bounds the potential error in the adversary's knowledge of the host's behaviors, while the number of hosts within the neighborhood provides a sense of the privacy of that host. Therefore, if a data publisher assumes the adversary could gain significant knowledge of a host's behaviors, perhaps derived from publicly available information, it would be prudent to consider the neighborhood sizes when the radius is relatively small. As Figure 6(a) shows, for example, even for a radius of 0.5, well over 80% of the hosts in the data have neighborhoods of size 20 or greater, thereby indicating potentially significant privacy for those hosts. Of course, as discussed earlier in Section 4.1, this is contingent upon the assumptions made about the adversary's "view" of the data via the metric definitions.

An obvious downside of this approach is that it is difficult to interpret the semantics of the overall DTW distances. That is, understanding what exactly a radius of 0.5 means with respect to the underlying behaviors may be difficult. One way to address this issue is to provide distributions of distances for each dimension computed during the time warping process, in addition to the Euclidean distance. Additionally, the semantically-meaningful metric spaces for each field may also need to be altered to accommodate for measuring behavioral distance between fields that have been altered by the anonymization process in the anonymized network data and those in the original (*e.g.,* comparing prefix-preserving IP pseudonyms to the original

IPs). However, we believe that the fact that our approach allows for such changes to the underlying semantics is a contribution in and of itself.

6 Conclusion

Many types of network data analysis rely on well-known distance metrics to adequately capture a meaningful notion of the behavioral similarity among network hosts. Despite the importance of these metrics in ensuring sound analysis practices, there has been relatively little research on the impact of using generic distance metrics and ignoring long-term temporal characteristics on analysis tasks. Rather, distance metrics used in practice tend to take a simplistic view of network data by assuming they inherit the semantics of its syntactic representation (*e.g.,* 16- or 32-bit integers), or that those values have no relationship at all. Moreover, they examine network activities in isolation or within short windows (*e.g., n*-gram distributions), which removes much of the long-term causal information found in the data. Consequently, these approaches are likely to provide brittle or unrealistic metrics for host behavior.

In this paper, we explored an alternative approach to defining host similarity that attempts to incorporate semantically meaningful spatial analysis of network activities and long-term temporal sequencing information into a single, unified metric space that describes host behaviors. To accomplish this goal, we developed metric spaces for several prevalent network data types, showed how to combine the metric spaces to measure the spatial characteristics of individual network data records, and finally proposed a method of measuring host behavior using dynamic time warping (DTW) methods. At each stage in the development of this framework, we brought to light potential pitfalls and attempted to explain the unique requirements surrounding the analysis of network data, including the need to carefully define normalization procedures and consider assumptions about the data made in developing the metrics. Our proposed metric was evaluated against the well-known L_1 distance metric, which ignores both semantic and temporal characteristics of the data, by applying cluster analysis techniques to a dataset containing a variety of realistic network host activities. Despite the admitted simplicity of our example metrics, the results of these experiments showed that our approach provides more consistent and useful characterizations of host behavior than the L_1 metric.

As a whole, these results indicate that it is useful to consider long-term temporal characteristics of network hosts, as well as the semantics of the underlying network data when measuring behavioral similarity. Furthermore, our results point toward several potentially interesting areas of future work. In the short term, one may consider the development of more refined distance metrics, including fine-grained metric spaces for a wider range of data types and time warping methods that allow for localized reordering of points. The results also call for a study of the performance of our DTW metric when applied to non-TCP protocols and to network objects other than hosts, such as web pages. More generally, a more formal method for characterizing behaviors, which may be used as the basis for provable network data anonymization techniques or robust traffic generation methods, seems warranted.

Data Access To encourage continued research on general network data similarity metrics, we have made the complete dataset used in our study available to the public via the PREDICT data repository [24] as "Departmental-Netflow-Trace-1" (Hosted By: Merit Network, Inc., Keywords: NetFlow).

Acknowledgments This work was supported by the U.S. Department of Homeland Security Science & Technology Directorate under Contracts FA8750-08-2-0147 and NBCHC080037, and by the National Science Foundation under Grant No. 0937060 that was awarded to the Computing Research Association for the CIFellows Project.

References

[1] D. Arthur and S. Vassilvitskii. k-Means++: The Advantages of Careful Seeding. In *Proceedings of the* 18^{th} *Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, January 2007.

[2] T. Auld, A. W. Moore, and S. F. Gull. Bayesian Neural Networks for Internet Traffic Classification. *IEEE Transaction on Neural Networks*, 18:223–239, 2007.

[3] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian. Traffic Classification on the Fly. *SIGCOMM Computer Communications Review*, 36(2):23–26, 2006.

[4] M. Bezzi. An Entropy-based Method for Measuring Anonymity. In *Proceedings of the* 3^{rd} *International Conference on Security and Privacy in Communications Networks and the Workshops*, pages 28–32, 2007.

[5] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):15, 2009.

[6] S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee. Toward privacy in Public Databases. In

Proceedings of the 2^{nd} Annual Theory of Cryptography Conference, pages 363–385, 2005.

[7] S. E. Coull, C. V. Wright, A. D. Keromytis, F. Monrose, and M. K. Reiter. Taming the Devil: Techniques for Evaluating Anonymized Network Data. In *Proceedings of the 15^{th} Network and Distributed Systems Security Symposium*, pages 125–135, 2008.

[8] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli. Traffic Classification Through Simple Statistical Fingerprinting. *SIGCOMM Computer Communications Review*, 37(1):5–16, 2007.

[9] J. Early, C. Brodley, and C. Rosenberg. Behavioral authentication of server flows. In *Proceedings of the 19^{th} Annual Computer Security Applications Conference*, pages 46–55, December 2003.

[10] J. Erman, M. Arlitt, and A. Mahanti. Traffic Classification using Clustering Algorithms. In *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data*, pages 281–286, 2006.

[11] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A Geometric Framework for Unsupervised Anomaly Detection. In *Applications of Data Mining in Computer Security*, pages 77–92, 2002.

[12] J. Fan, J. Xu, M. Ammar, and S. Moon. Prefix-preserving IP Address Anonymization: Measurement-based Security Evaluation and a New Cryptography-based Scheme. *Computer Networks*, 46(2):263–272, October 2004.

[13] V. Frias-Martinez, S. J. Stolfo, and A. D. Keromytis. Behavior-Profile Clustering for False Alert Reduction in Anomaly Detection Sensors. In *Proceedings of the Annual Computer Security Applications Conference*, pages 367–376, 2008.

[14] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multilevel Traffic Classification in the Dark. In *Proceedings of ACM SIGCOMM*, pages 229–240, August 2005.

[15] T. Karagiannis, K. Papagiannaki, N. Taft, and M. Faloutsos. Profiling the End Host. In *Proceedings of the Passive and Active Network Measurement Conference*, pages 186–196, 2007.

[16] A. Kounine and M. Bezzi. Assessing Disclosure Risk in Anonymized Datasets. In *Proceedings of FloCon*, 2008.

[17] A. McGregor, M. Hall, P. Lorier, and J. Brunskill. Flow Clustering Using Machine Learning Techniques. In *Proceedings of the Passive and Active Network Measurement Conference*, pages 205–214, 2004.

[18] A. W. Moore and D. Zuev. Internet Traffic Classification Using Bayesian Analysis Techniques. In *Proceedings of ACM SIGMETRICS*, pages 50–60, June 2005.

[19] J. Munkres. *Topology*. Prentice-Hall Englewood Cliffs, NJ, 2000.

[20] Nagios IT Infrastructure Monitoring. `http://www.nagios.org/`.

[21] Nessus Vulnerability Scanner. `http://www.nessus.org`.

[22] T. Nguyen and G. Armitage. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys & Tutorials*, 10(4):56–76, 2008.

[23] R. Pang, M. Allman, V. Paxson, and J. Lee. The Devil and Packet Trace Anonymization. *ACM Computer Communication Review*, 36(1):29–38, January 2006.

[24] M. Ramadas, S. Ostermann, and B. Tjaden. Detecting Anomalous Network Traffic with Self-Organizing Maps. In *Proceedings of Recent Advances in Intrusion Detection Conference*, pages 36–54, 2003.

[25] B. Ribeiro, W. Chen, G. Miklau, and D. Towsley. Analyzing Privacy in Enterprise Packet Trace Anonymization. In *Proceedings of the 15^{th} Network and Distributed Systems Security Symposium, to appear*, 2008.

[26] H. Sakoe and S. Chiba. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26:43–49, 1978.

[27] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou. Specification-based Anomaly Detection: A New Approach for Detecting Network Intrusions. In *Proceedings of the 9^{th} ACM Conference on Computer and Communications Security*, pages 265–274, 2002.

[28] Y. Song, A. Keromytis, and S. Stolfo. Spectrogram: A Mixture-of-Markov-Chains Model for Anomaly Detection in Web Traffic. In *Proceedings of the 16^{th} Annual Network and Distributed System Security Symposium*, pages 121–136, 2009.

[29] S. Wei, J. Mirkovic, and E. Kissel. Profiling and Clustering Internet Hosts. In *Proceedings of the 2006 International Conference on Data Mining*, pages 269–275, 2006.

[30] N. Williams, S. Zander, and G. Armitage. A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification. *SIGCOMM Computer Communications Review*, 36(5):5–16, 2006.

[31] C. Wright, F. Monrose, and G. M. Masson. On Inferring Application Protocol Behaviors in Encrypted Network Traffic. *Journal of Machine Learning Research*, Special Topic on Machine Learning for Computer Security(7):2745–2769, December 2006.

[32] K. Xu, Z. Zhang, and S. Bhattacharyya. Profiling Internet Backbone Traffic: Behavior Models and Applications. In *Proceedings of ACM SIGCOMM*, pages 169–180, August 2005.