Know Your Enemy, Know Yourself: Block-Level Network Behavior Profiling and Tracking

Esam Sharafuddin, Yu Jin, Nan Jiang, Zhi-Li Zhang

University of Minnesota
{shara,yjin,njiang,zhzhang}@cs.umn.edu

Abstract. Gaining a better knowledge of one's own network is crucial to effectively manage and secure today's large, diverse campus and enterprise networks. Because of the large number of IP addresses (or hosts) and the prevalent use of dynamic IP addresses, profiling and tracking individual hosts within such large networks may not be effective nor scalable. In this paper we develop a novel methodology for capturing, characterizing, and tracking network activities at the *block-level*. To characterize block-level behaviors, we carefully select a port feature vector and capture the port activities of individual hosts within a block using a *block-wise (host) port activity matrix (BPAM)*. Applying the SVD low-rank approximation technique, we obtain a low-dimensional subspace representation which captures the significant and typical host activities of the block. Using these subspace representations, we cluster and classify blocks to provide high-level descriptive labels to assist network operators and security analysts to gain a "big-picture" view of the network activities. We also develop novel methods to track and quantify changes in blocks' behaviors over time, and demonstrate how these methods can be utilized to identify major changes and anomalies within the network.

1 Introduction

Due to its scale and complexity, managing and securing today's large campus or enterprise networks is a challenging task. The scale and complexity comes not only from the number of heterogeneous hosts and devices on the network (e.g., various servers, desktop office client machines, laptops, lab machines, wireless access points, routers and so forth), but also from a wide range of diverse applications running on these machines. Traditionally, network security has largely focused on identifying and preventing attacks, e.g., through attack signature generation or anomaly detection (i.e., the focus is on understanding the attacks and attackers). However, the scale, complexity and diversity of large campus and enterprise networks render such an approach *alone* less efficient, scalable and manageable. For instance, in the case of anomaly detection, what constitutes "anomalous" activities in one network or part of it may be considered to be "normal" in another network (or subnet). As an example, "unauthorized" peer-to-peer file sharing applications are not allowed on a departmental subnet of our campus network (unless they are for the purpose of research); on the other hand, such peer-to-peer activities are considered legitimate on student residential hall subnets. Hence, to more effectively manage and secure a large, diverse network, one must also build a good knowledge of one's own network, e.g., by understanding the range of applications, usage patterns and user behaviors in various parts of the network. Such knowledge will enable network operators and security analysts to better tailor their monitoring system and detection tools (e.g., firewall configurations), and focus their attention on specific vulnerabilities or areas of problem.

Along this new direction of *understanding oneself*, several research studies [1–3] have developed algorithms and tools for (primarily) *host-level* traffic classification and behavior profiling. While these studies offer innovative methods for classifying traffic or host behaviors, the analysis at the granularity of individual hosts (or individual IP addresses) has two major drawbacks in practice. First, the prevalent usage of dynamic IP addresses makes tracking individual hosts an infeasible task in most networks [4, 5], since dynamic IP addresses are frequently reassigned to different hosts. Furthermore, the large number of IP addresses (e.g., our campus network has 3 class-B subnets, with 3×10^{16} potential hosts) also make applying host-level traffic profiling to every host quite expensive.

To address these limitations, in this paper, we propose and develop a novel methodology for *block-level* network traffic behavior profiling. An IP address block constitutes a set of consecutive IP addresses, typically in size of 2^k, say, $k = 8$ (i.e., a /24 or class-C block), a unit used by a network administrator for IP address assignment to a subnet. More often than not, many hosts within the same block would be used for similar usage, e.g., a department block for office desktop and laptop machines, a block for lab machines, a block for student residential hall, a block with one or two wireless access points for wireless access (i.e., a wireless block), and so on. As shown in [5], dynamic IP addresses are generally assigned in a block of consecutive IP addresses. Hence, by analyzing and profiling network activities at the *block-level*, we can circumvent the issues caused by dynamic IP addresses. Further, exploiting the similar user activities and usage patterns within a block, we can obtain a more compact block-level behavior profile which captures and summarizes the significant and typical behaviors of hosts within the block. Finally, the block-level analysis is far more scalable: in the case of our campus network, using /24 blocks we only need to profile and track at most 768 $(= 3 \times 256)$ blocks as opposed to 3×10^6 IP addresses.

In this paper, we employ flow-level data (i.e., Netflow data) captured at the campus border router, and utilize the *port* information thereof to characterize and profile traffic behaviors and host activities at the block level. By considering well-known service ports, popular application ports and other dominant ports extracted from our flow data, we form a *port feature vector* consisting of 2000 source and 2000 destination ports. Using this port feature vector, a straightforward way to summarize the behavior of a block is to simply compute the aggregate port distribution of the block: namely, for each source or destination port in the port feature vector, the fraction of flows using the port that are generated by any IP address (or host[1]) within the block. However, while the aggregate port distribution captures the overall activities of hosts within the block, it fails to provide adequate information to capture, characterize, and distinguish significant and typical host behaviors within the block. For example, we would like a

[1] For simplicity, in this paper we use the term a *host* to denote a specific IP address (although an IP address may be assigned to a router, a printer or some other devices).

block-level behavior profile to enable us to meaningfully answer questions such as the following: i) Do all hosts in the block behave similarly, e.g., most of them are client machines that are used to primarily access the web? Thus, the overall port distribution would represent the "typical" behavior of the hosts in the block? ii) Does the block contain one or a few dominant hosts (e.g., web servers, or "heavy-hitter" client machines) that generate a majority of the flows? In other words, the overall port distribution of the block is skewed mostly by these dominant hosts, while obscuring the activities of other "typical" hosts within the block. iii) Or, does the block consist of several groups of hosts with distinct behaviors or activities, e.g., web/email servers, client machines with heavy web and P2P activities?

The ability to answer these and similar questions is important to characterize, summarize and distinguish the behaviors of various inside (campus) blocks within a network, and therefore help network operators and security analysts to understand and monitor the block-level activities, detect sudden changes and anomalies, and identify policy violations, security breaches and malicious attacks. For this purpose, we introduce the *block-wise (host) port activity matrix* (BPAM), which records the activities of each host within the block on these ports, e.g., the number of flows using each of the ports. Hence, the BPAM represents the key port activities of individual hosts within each block. By applying the Singular Value Decomposition (SVD) method (to an appropriately normalized and re-scaled version of BPAM), we obtain a compact, low-dimension *subspace representation* of the behaviors of each block. We show that as a low-rank approximation to the original BPAM, this subspace representation captures the *significant* and *typical* activities of individual hosts within the block, and therefore can be used to answer the questions listed above.

In addition, by introducing a subspace distance metric, we employ the subspace representations to cluster and classify the behaviors of various blocks within a network. The block-level behavior clustering allows us to assign *interpretive* labels to various blocks as to assist network operators and security analysts in understanding the overall block-level activities within the network. Furthermore, we demonstrate how to use the subspace representations to track changes in block-level behaviors over time, and develop two methods to quantify and classify such changes. We also show how these methods can be explored to identify major changes and anomalies within a network. The efficacy of our proposed block-level network behavior profiling methodology has been extensively evaluated and validated using a month-long netflow data collected at our campus network.

The remainder of this paper is organized as follows. In section 2 we describe how the port feature vector is selected. In section 3 we introduce the BPAM and the SVD-based subspace method to extract and summarize significant and typical host behaviors within each block. In section 4 we develop a clustering method to classify block-level behaviors using the subspace representations, while in section 5, we develop methods for tracking and quantifying block-level behavior changes, and show how they can be used to identify anomalies. The paper is concluded in section 6.

2 Block Behaviors and Port Feature Vector

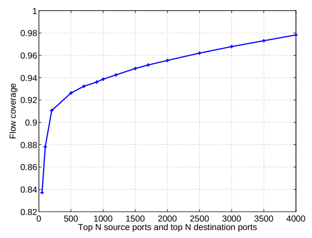
Port numbers are the most widely used packet/flow level features for identifying network activities. Certain (IANA reserved or registered) ports are almost synonymous with the well-known services associated with these ports, e.g., web with TCP 80, email with TCP port 25, and DNS with UDP port 53. Although these reserved ports may be misused by other applications (e.g., for penetrating firewalls), or the well-known services may also use other ports (e.g., TCP 8080 for web), for a majority of the hosts the dominant activities observed on these ports represent the well-known services. Furthermore, many popular "non-standard" applications that do not have officially "reserved" or "registered" ports, such as many P2P and other emerging applications, often use certain fixed ports, or have a default range of ports. More importantly, because of the socket layer programming interfaces used by networked applications today, source and destination ports are generally used in certain manners that distinguish ports that provide *services* (in the sense of an "application server") from many random ports (used by "application clients"). For example, an application providing a service (an "application server") often must listen on a fixed (source) port, while an application requesting such a service (an "application client") must send a service request using the said port as the destination port and a typically randomly generated source port. Hence when examining the flow-level data, the service ports used by popular applications (even though these ports may not be fixed) tend to occur more frequently than a randomly selected port. In this section, we utilize this observation to select a set of frequently used ports to form a *port feature vector* for characterizing block-level network activities based on ports. Before presenting this method, we first describe the datasets used in our study.

Dataset: Our study is based on a one-month data collected at the border router of our campus network. The data includes bidirectional Cisco NetFlow records corresponding to traffic between inside (campus) hosts (3 class B IP blocks with 2^{16} IP addresses) and outside hosts. We focus on the *outgoing* TCP, UDP and ICMP traffic which account for more than 99% of all the traffic from our campus network to the outside Internet hosts. The outgoing flows represent traffic activities either initiated by inside hosts, or in response to outside service requests, thus they are indicative of true activities of the inside hosts. In contrast, incoming traffic may contain a significant amount of "noises" (such as various scanning, backscatter and other activities) generated by the outside hosts towards our campus network, many of which do not even pass the campus network border firewall. In the following, when referring to specific source or destination port number, e.g., source or destination port 80, we will also use the shorthand, *srcPort 80* or *dstPort 80*.

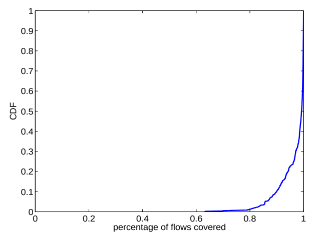
We note that we choose the block size to be 2^8 (class C IP block) throughout the paper, which is the most commonly used block size for our network administrators to assign IP addresses to different departments/subnets [5].

2.1 Port Feature Vector Selection

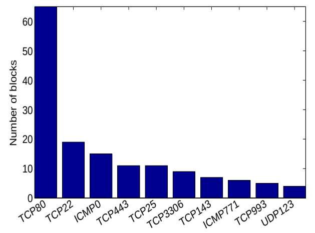
One simple way to select ports to characterize host activities is to utilize a list of well-known or registered service ports from IANA. IANA defines the port number range 0-1023 as the well-known (or reserved) ports, the port number range 1024-49151 as



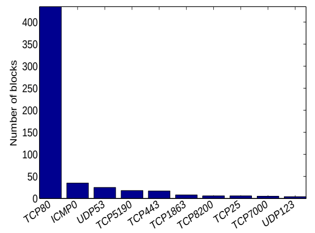
(a) Top N ports vs. flow coverage



(b) CDF of per block flow coverage



(c) Top 10 most popular source service ports



(d) Top 10 most popular destination service ports

Fig. 1. Properties of top service ports

registered ports and the remaining ports as dynamic/private ports. Using such an approach has two obvious drawbacks. First, many ports used by popular applications such as most instant messenger (IM or chat) and p2p applications use port numbers above 1024 or 5000 as service ports. Furthermore, services or applications associated with some well-known or registered service ports may not be provided or used by any hosts within the campus network (or even if they are available, they may be restricted to internal use, thus have no associated traffic activity across the campus border router). Including these reserved/registered service ports in the port feature vector is therefore unnecessary. To address these issues, we propose a method to extract service ports based on the frequencies of ports occurring in the observed traffic of the entire campus network. As described earlier, the basic intuition is that service ports used by popular or common applications by many hosts in our campus network will likely occur more frequently than a randomly selected port. We therefore select the most frequent ports (in terms of number of flows/hosts associated with them) as likely candidates for service ports. The port selection method is described below.

Let F_t be the set of flows observed within our campus network during a time interval t, say, a day (this is the time interval used in this paper). We rank all the source (resp. destination) ports that appear in F_t in terms of both the number of flows and the number of hosts *covered*: we say a *srcPort* (resp., *dstPort*) p covers a flow f if the flow contains p as the source (resp., destination) port. Likewise, we say the *srcPort* or *dstPort* p covers an inside host h (an IP address) if h appears as the source IP address in at least one of the covered flows. We pick the top ranked N source ports and N destination ports in such a manner that they cumulatively cover at least, say 95%, of all flows as well as of all "active" hosts (an IP addresses which generates at least one flow during the time interval t). The intuition is that the (source or destination) service ports selected are expected to contribute a significant traffic volume and be used by a large number of hosts. Hence the choice of N should be sufficiently large so that we can identify nearly all service ports used by popular applications. However, it should not be too large to avoid misclassifying some random ports as service ports.

Through experiments using our flow datasets, we decide on $N = 1999$, which yields 1999 top source ports as well as 1999 top destinations[2]. Fig. 1(a) shows the combined flow coverage (y-axis) of these top source and destination ports (x-axis). From the figure, we see that they cover nearly 98% of all flows. These top source/destination ports also cover nearly 100% of all "active" hosts. Due to the space limitation, we do not include the corresponding figure here. We group all remaining source ports as if they were a special "virtual" source port, referred to as "all other source ports" (or *aoSrcPort* in short), and all remaining destination ports as if they were a special "virtual" destination port, referred to as "all other destination ports" (or *aoDstPort* in short). Together with the top 1999 source ports and 1999 destination ports, we define a 4000-dimension *port feature vector* $PFV = [port_1, \ldots, port_{4000}]$, where for $1 \leq j \leq 1999$, $port_j$ refers to one of the top 1999 source ports (ordered in the increasing number of the port numbers), and for j=2000, $port_{2000}$ refers to *aoSrcPort*; and for $2001 \leq j \leq 3999$, $port_j$ refers to one of the top 1999 destination ports (ordered in the increasing number of the port numbers), and for j=4000, $port_{4000}$ refers to *aoDstPort*.

2.2 Characterizing Blocks using Ports

In this section, we show some statistics regarding the block-level flow coverage using the selected service ports and illustrate how these ports provide some basic information regarding the activities of various blocks. In the following, we use S_{src} and S_{dst} to denote the sets of 1999 source service ports and 1999 destination service ports, respectively.

In Fig. 1(b), we show the flow coverage in each block using the selected service ports, where x-axis represents the percentage of flows in each block that are covered by either S_{src} or S_{dst}, and y-axis is the CDF. We observe that for 90% of the blocks, the flow coverage is at least 90%, indicating that these selected service ports also have good flow coverage at the block level. However, there are five blocks with flow coverage less than 75%. Investigation on the domain names of the IP addresses in these blocks indicates that they contain mostly wireless addresses and other dynamic IP addresses. Hosts using these address blocks may run some non-typical applications, perhaps even inflicted with certain "suspicious" activities.

Figs. 1(c) and (d) show the top 10 source ports and top 10 destination ports that are most popular in all 768 blocks (which accounts for at least 10% of the traffic in each block), where x-axis represents the port numbers and y-axis stands for the total number of blocks with the associated port number. The most popular server activity is web (80 and 443), ssh (22) and email (25 and 993)[3]. Investigation on these blocks with dominant

[2] We refer to ICMP type numbers as the port numbers for ICMP traffic, e.g., ICMP0 represents ICMP type 0 echo reply traffic.

[3] We note that, unless otherwise mentioned, we refer a port number as a TCP port by default

port numbers showing dominant activities, we find that most of these blocks belong to certain department subnets which maintain their own web servers and email servers. On the other hand, the most popular client activities are associated with web (80 and 443), DNS (UDP53), messenger (AOL for 5190 and MSN for 1863). The number of blocks associated with web client activities is much larger, agreeing with the fact that most of IP addresses in our campus network are used by client machines. Investigating the blocks that contain most (web/chat) client activities indicates that many of these blocks belong to residential hall networks, or wireless and other dynamic address blocks that are typically dominated by client machines.

As argued in the introduction, though the aggregated port distribution of a block provides us with some hints on the overall activities of the hosts within a block, it fails to provide adequate information to capture, characterize, and distinguish significant and typical host behaviors within the block. This is mainly due to the fact that certain extremely active hosts, either popular servers or heavy hitter client machines, can dominate the overall activities of a block, thus masking the typical behaviors of other active hosts. For example, in a block containing different kinds of servers, e.g., web servers, email servers, ssh servers and an IRC server, the activity of the IRC server and ssh servers may not be "visible" due to being masked by the typically huge amount of traffic generated by web and email servers. To circumvent this problem, in the next section, we will introduce the block-wise (host) port activity matrix to represent the port activities of individual hosts within a block.

3 Block-wise Host Port Activities and Subspace Representation

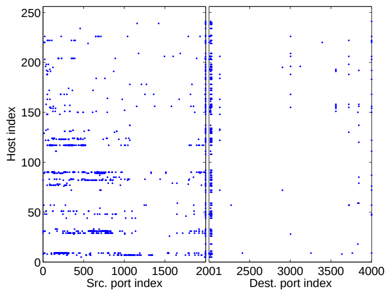
In this section, we introduce the *block-wise (host) port activity matrix* (BPAM) to represent the port activities of individual hosts within a block. Using the Singular Value Decomposition (SVD) method, we derive a low-rank approximation to the BPAM, and thus obtain a low-dimension *subspace representation* of the behaviors of each block. Applying this technique to our campus flowdata, we demonstrate that these subspaces indeed capture the *significant* and *typical* activities of individual hosts within a block, and produce compact and meaningful block-level behavior representations that can be used to characterize and distinguish various block behaviors within a network.

3.1 Extracting Significant Block Behaviors from BPAMs

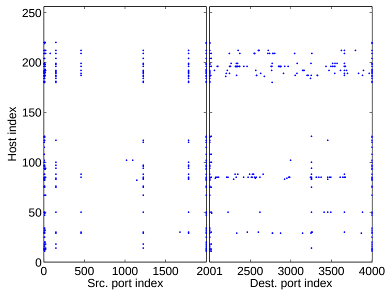
In this section, we formally define the block-wise (host) port activity matrix (BPAM), and apply singular value decomposition (SVD) [4] to A to extract *significant* activities of "typical" as well as "dominant" hosts within a block. Mathematically, as a low-rank matrix approximation to the original BPAM of the block, the outcome of SVD yields a *subspace* representation of the significant block behaviors.

Given a /24 block B, let i, $0 \leq i \leq m-1$ (where $m = 256$), denote a host (more precisely an IP address) within the block. For each host i, let f_i denote the total

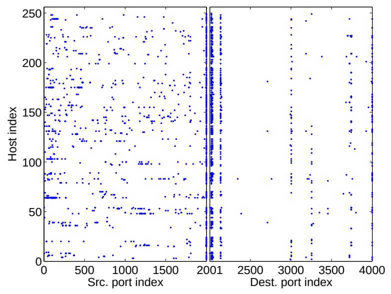
[4] We conducted the same experiments using both SVD and robuts PCA [6, 7], and the results are similar, possibly due to the normalization process which eliminates the effect of outliers.



(a) a client-centered block



(b) a server-dominated block



(c) a residential network block

Fig. 2. Example block-wise host port activities.

number of (outgoing) flows generated by host i during an observation time window W, say, a day. For $1 \leq j < 2000$, we use j to represent one of the 1999 most frequent *source* ports in the *port feature vector* defined in the previous section, and $j = 2000$ to represent the all other source ports. Likewise, for $2001 \leq j < 4000$, we use j to represent one of the 1999 most frequent *destination* port in the port feature vector, and $j = 4000$ to represent the all other destination ports. Then for $1 \leq j \leq 2000$, f_{ij} denotes the number of flows generated by hosts i that uses the source port(s) j, whereas for 2001 $\leq j \leq$ 4000, f_{ij} denotes the number of flows generated by hosts i that uses the destination port(s) j. Note that since each flow has both a source and a destination port, we have $\sum_{j=1}^{2000} f_{ij} = \sum_{j=2001}^{4000} f_{ij} = f_i$.

A straightforward way to represent the block-wise host port activities is to directly use the $m \times n$ matrix $F_B = [f_{ij}]$ (where $m = 256$ and $n = 4000$). Fig.2(a)-(c) visually depict F_B for three representative blocks, where each row in the figure corresponds to a host within the block, and a dot at the jth position of the ith row is plotted if and only if $f_{ij}/f_B \geq 0.005$ and $f_i \geq 100$ flows, where $f_B = \sum_{i=1}^{256} f_i$ is the total number of flows generated by all hosts in the block. The left half (1-2000) of the x-axis represents the source ports, where x=2000 represents *all other* source ports (*aoSrcport*), and the right half (2001-4000) represents the destination ports, where x=4000 *all other* destination ports (*aoDstPort*). The block in Fig.2(a) is a department block with predominantly client machines, Fig.2(b) a department block with many servers (web, email, etc.), whereas Fig.2(c) depicts a block in the student residential hall network with many diverse activities, e.g., frequent p2p, IM activities in addition to typical client activities such web and email. These figures visually illustrate that these blocks indeed exhibit distinct behaviors characteristics.

When applying SVD to extract the "significant" and "typical" behaviors of a block B, using F_B directly has a major drawback. For instance, consider a block that contains one or two popular servers or a few extremely active hosts, but otherwise consists of typical client machines. Because these servers or "heavy-hitter" hosts generate far larger number of flows than an average client machine, their behaviors may mask those of "typical" hosts within the block. In other words, the outcome of SVD may represent only the behaviors of a few dominant "heavy-hitters" but miss the significant behaviors of "typical" hosts. Another alternative is to use the (frequency) matrix $P_B = [p_{ij}]$, where $p_{ij} = f_{ij}/f_i$ is the frequency of flows using port j. This matrix, on the other hand, amplifies the behaviors of "inactive" hosts that generate a few flows in total. For instance, consider a block that contains many active hosts (i.e., frequently generating traffic) but a few "inactive" hosts. These inactive hosts may occasionally respond to outside scanning activities on a few random service ports that are otherwise not used by any of the "typical" active hosts within the block. For these inactive hosts, although there are only a few flows on these randomly scanned ports, because f_i is also very small, p_{ij} is closer to 1. In other words, the corresponding row entries of these inactive hosts would dominate those of "typical" active hosts. But the behaviors of these "typical" active hosts are actually what we are interested in extracting!

To counter-balance the effects of extremely active as well as inactive hosts, we introduce an appropriately *normalized* and *re-scaled* version of F_B (or P_B) using entropy. Recall that $f_B = \sum_{i=1}^{m} f_i$ is the total number of flows generated by all hosts within block B. Define $p_i = f_i/f_B$, the fraction of flows generated by host i. We define the *(flow activity) entropy* of block B, $ent_B := -\sum_{i=1}^{m} p_i \log p_i$. We note that $0 \leq ent_B \leq \log m$: the closer ent_B is to the upper bound $\log m$, the more uniformly distributed are the flows among the hosts; whereas the closer ent_B is to 0, the more skewed is the flow distribution among the hosts. Using ent_B, we define a *scaling factor* for each host as follows: for $i = 1, \ldots, m$, $s_i := ent_B/(-logp_i)$ if $p_i > 0$, and $s_i = 0$ otherwise[5]. We see that the smaller p_i is, the smaller s_i is. On the other hand, s_i only grows inverse logarithmically with f_B/f_i (approximately logarithmically with f_i), thus dampening the effect of extremely active hosts. We are now in a position to formally define the *block-wise (host) port activity matrix* (BPAM) for a given block B:

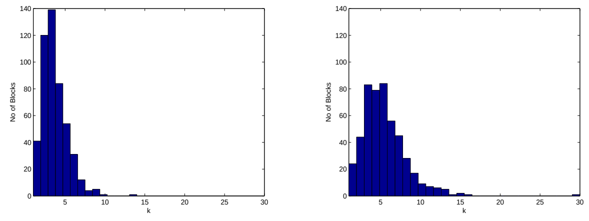
$$A_B := [s_i p_{ij}]_{m \times n} = [s_i f_{ij}/f_i]_{m \times n}.$$

Note that each row i of A_B sums up to s_i, hence, the contribution of each host i to the total "mass" of the BPAM A_B is proportional to s_i. (In the following, we will drop the subscript B when the context is clear.)

Given this definition of BPAM, we apply SVD to A to extract "significant" and "typical" host behaviors of a block. As $m \leq n$, the SVD decomposition of A is given by $A = U_{m\times m} \Sigma_{m\times m} V^T_{m\times n}$, where $\Sigma = [\sigma_i]$ is a diagonal matrix containing the *singular values* in the decreasing order, $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_m$, $U = [u_1, \cdots, u_n]$ is an

[5] Here we implicitly assume that each block has at least two active hosts, thus $p_i < 1$ for all i's. As $-\log p_i$ is the entropy of an individual host i, intuitively s_i measures the contribution of individual hosts' entropies to the (average) entropy of the block.

orthonormal matrix (i.e., $UU^T = I$), the columns of which are the *left singular vectors* of A, and $V = [v_1, \cdots, v_m]$ is also an orthonormal matrix, where the columns are *right singular vectors* of A.



(a) $\epsilon = 10\%$ (b) $\epsilon = 5\%$

Fig. 3. No of singular values for different values of ϵ

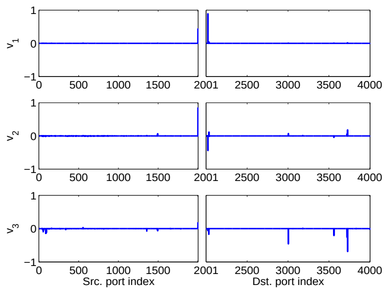
Intuitively, the (n-dimensional) right singular vectors, v_i's, provide an *orthogonal* representation of the port activities of the m hosts within a block. In particular, v_1 captures most dominant port activities (those with most variances or energy) across all hosts, v_2 the second most dominant port activities, and so forth. In a sense, each v_i can be viewed as a *virtual* host, where $|v_{ij}|$ (or v_{ij}^2) measures the magnitude (or fraction) of activities on port j by this virtual host (note that $\sum_{j=1}^{n} v_{ij}^2 = 1$, hence, $\{v_{ij}^2\}$ can be viewed as a probability distribution). Unlike the port activities of the m original (*real*) hosts, the m *virtual* hosts have *orthogonal* port behaviors (i.e., $v_i^T v_j = 0$ for $i \neq j$). As a result, we can decompose the (rescaled) port activities of each original host h, $1 \leq h \leq m$, as a linear combination of the m virtual hosts, where u_{hi}^2 measures the contribution of the virtual host i. Since the magnitude of the overall virtual host i activities is measured by σ_i, those with largest σ_i (say, the first K) capture and represent the most significant and typical behaviors of the hosts within a block. For an appropriately chosen K, the first K singular values and their associated left and right singular vectors provide a low-rank approximation to A, namely,

$$A \approx U_K \Sigma_K V_K^T,$$

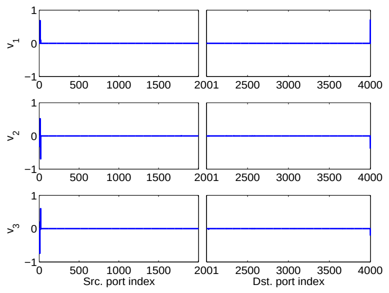
where $U_K = [u_1, \cdots, u_K]$, $\Sigma_K = diag[\sigma_1, \cdots, \sigma_K]$, and $V_K = [v_1, \cdots, v_K]$. Hence, extracting the significant and typical host behaviors of a block boils down to obtaining a low-rank approximation to the BPAM A via SVD. We refer to the *subspace* spanned by the first K dominant virtual hosts, $v_1, \ldots, v_K$, as a subspace representation of the (significant and typical) behaviors of a block. In the next section, we will discuss how we decide on K, and provide some interpretations of extracted "dominant" (virtual) host behaviors (v_k's), where $1 \leq k \leq K$.

3.2 Significant Block Behaviors and Interpretations

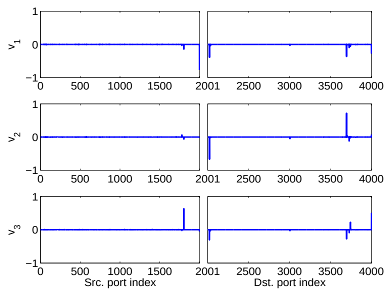
We employ the standard scree plot method to choose K: given a small threshold $\epsilon > 0$, we deem a singular value σ_k significant if $\sigma_k/\sigma_1 \geq \epsilon$, in which σ_1 is the largest singular value. Using this method, we apply SVD to the BPAMs of the /24 blocks in our campus



(a) A client-centered block



(b) A server-centered block



(c) A residential hall block

Fig. 4. Port distribution for the top 3 singular vectors.

network. Figs. 3(a)-(b) show the histogram of the resulting K's for all the blocks for $\epsilon = 0.1$ and $\epsilon = 0.05$, respectively. We see that a majority of the blocks require only a small K, say, ≤ 5. In fact, we have validated that the low-rank approximations thus obtained indeed capture at least 95% of the energy in the original BPAM (or, with squared errors less that 0.05). Hence, for the majority of blocks, their significant and typical behaviors can be captured and represented by a few (largest) singular vectors v_k's. However, there do exists a few "outlier" blocks which require far large number of singular vectors (e.g., one with 30 and another one with 15).

We now use the three representative blocks shown in Fig. 2 as examples to illustrate and interpret the behaviors captured by the top (right) singular vectors v_k's, and show that they indeed capture the significant and typical behavioral characteristics of these blocks. Using $\epsilon = 0.05$, we obtain $K = 3$ for each of the three blocks in Fig. 2(a)-(c). In Figs. 4(a)-(c), we plot the "energy" of the ports, v_{ij}, in the top three singular vectors, v_1 (the top panel), v_2 (the middle panel), and v_3 (the bottom panel), for each of the three blocks. Recall that the range $[0, 2000)$ in the x-axis represents the 1999 most frequent source ports and $j = 2000$ all other source ports (compactly denoted as *aoSrcPort* below); and the range $[2001, 4000)$ the 1999 most frequent destination ports and $j = 4000$ all other destination ports (compactly denoted as *aoDstPort* below).

For the client-centered block in Fig. 4(a), we first note that most points with high energy (i.e., with large non-zero value v_{ij}^2) in v_1-v_3 are concentrated in the destination port range [2001-4000], except for one major point at $j = 2000$ (in v_1) corresponding to *aoSrcPort*. In addition to *aoSrcPort*, the other two largest points in v_1 correspond to destination port 80 ($j = 2020$) and port 443 ($j = 2033$). Hence, v_1 captures the web-related client activities of the hosts. Whereas, v_2 contains a number of considerably large nonzero points corresponding to various destination service ports such as Instant Messaging (IM) for AOL port 5190, ($j = 3728$), in addition to destination ports 80 and 443 (when the same point appears in both v_1 and v_2, it has opposite signs). v_3 also corresponds to other IM, such as Yahoo Messenger port 5050 ($j = 3721$), MSN Messenger port 1863 ($j = 3004$) in addition to remote desktop port 3389 ($j = 3561$). Therefore, the subspace spanned by the top three singular vectors captures the prevalent client behaviors of the hosts within the block, where web-related and IM activities are most significant and typical.

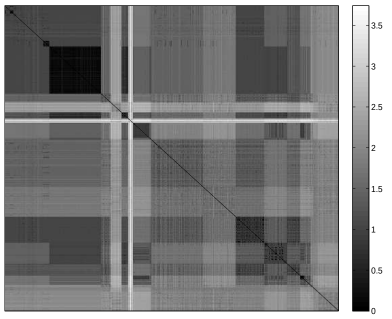
In contrast, for the server-dominated block in Fig. 4(b), most points with high energy in both v_1 and v_2 are concentrated in the source port range [1-2000], except for the point at $j = 4000$ corresponding to *aoDstPort*. In v_1, the other major nonzero points correspond to major source service ports such as source ports 80 ($j = 11$) in v_1.

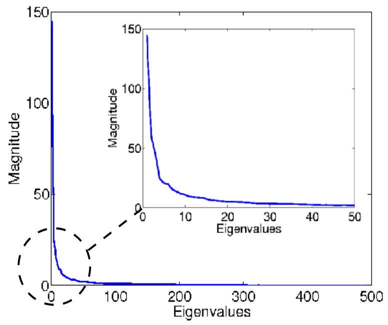
Whereas, v_2 contains a number of considerably large nonzero points in v_2 corresponding to various source ports such as email port 25 ($j = 8$), and ssh port 22 ($j = 7$). v_3 has similar ports (with opposite signs) to that of v_2. Hence, the subspace spanned by the top three singular vectors captures the prevalent server behaviors of the hosts within the block, where web, email, and ssh server activities dominate.

Unlike the previous two cases, the top three singular vectors in the residential network block in Fig. 4(c) contain several points with high energy in both the source $[1, 2000]$ and destination $[2001, 4000]$ port ranges. In addition to *aoSrcPort* and *aoDstPort*, the other two major points in v_1 correspond to destination ports 80 ($j = 2020$), and destination port eMule (p2p) 4662 ($j = 3696$), whereas the points with highest energy in v_2 correspond to destination ports 80 and 4662 (with opposite signs). In v_3, in addition to these two destination ports and the *aoDstPort* port, points with highest energy correspond to source port gnutella (p2p) 6348 ($j = 1805$) and destination IM port 5190 ($j = 3728$). Hence, the subspace spanned by the top three singular vectors captures typical residential hall behaviors in which p2p and IM are prevalent in addition to the web-client behaviors.

We have performed similar analysis of the top singular vectors for other blocks, and found that, overall, these top singular vectors indeed capture the significant and typical behaviors of the hosts within the block, as confirmed by examining the port activities of those individual hosts which generate significant number of flows. As an additional example, we examine the outlier block in Fig. 3 which requires top 30 singular vectors to capture its significant and typical behaviors. These top singular vectors represent diverse behaviors of a mixed group of server (e.g., web, email and ssh) and client hosts which are active on a wide range of source and destination ports. Using information from other sources, we find that it belongs to one of the most diversified school: the school of public health. Blocks belonging to this department are assigned to several other smaller departments: biology, genetics, health school, clinical research and other health-related departments. It is not surprising to find out that this block was originally assigned to the school of public health in general but as the school expanded, portions of the same block were assigned to different departments which put them into a wide range of different usages, from web, email, ftp servers, to office and lab machines as well as wireless access networks.

4 Classifying Block Behaviors


(a) Visual representation of subspace distance matrix.



(b) Eigenvalue scree plot.

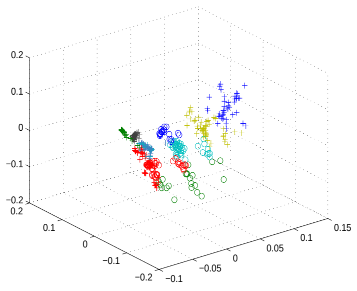

(c) 3-D embedding and K-means clustering.

Fig. 5. Clustering and Labeling of Blocks

The examples in the previous section show that blocks with hosts running different applications exhibit distinct behavioral characteristics, and the principal subspace provides a succinct way to extract, characterize and represent the significant and typical behaviors of a block. In this section, we compare and classify block-wise behaviors based on their subspace representations. By grouping blocks with similar behaviors, we can assign a high-level *interpretive* label to each block, e.g., a client-centered block with prevalent web activities, a server-dominated block with web and email servers, a resident network block with frequent web, p2P, IM activities, and so forth. Such interpretive labels will enable network operators and security analysts to better understand and manage a campus/enterprise network, and profile and track the behaviors of various blocks within the network.

Consider two blocks, B_1 and B_2, where $v_1^{(1)}, \ldots, v_{K_1}^{(1)}$ and $v_1^{(2)}, \ldots, v_{K_2}^{(2)}$ are the top (right) singular vectors extracted from their respective BPAMs. Let $V^{(i)}$, $i = 1, 2$, be the subspace spanned by $v_1^{(i)}, \ldots, v_{K_i}^{(i)}$, i.e., $V^{(i)}$ is the resulting subspace representation of block B_i. To compare the behaviors of these two blocks, we need a *similarity* or *distance* (dissimilarity) metric for the two subspaces, $V^{(1)}$ and $V^{(2)}$. Note that as two subspaces in an n-Euclidean space R^n (n is the number of elements in the singular vectors, $v_j^{(i)}$'s) both pass through the origin, and the set of the singular vectors $v_1^{(i)}, \ldots, v_{K_i}^{(i)}$ forms an orthonormal basis of the K_i-dimensional subspace, $i = 1, 2$. Intuitively, if $K_1 = K_2$ and the two subspaces coincide with each other, then the two subspaces are exactly identical, i.e., their distance is zero. On the other hand, if two subspaces are perpendicular to each other (i.e., $\langle v_i^{(1)}, v_j^{(2)} \rangle = 0$, $1 \leq i \leq K_1$ and $1 \leq j \leq K_2$, where $\langle \cdot, \cdot \rangle$ denotes the inner product of two vectors), then they are least similar, i.e., their distance is largest. In general, as $K_1 \neq K_2$, we would expect that two subspaces where one is entirely contained in the other would have a smaller distance than those where one is only partially contained in, or "cut cross" the other.

These intuitions lead to the following distance metric between two subspaces [8]: For any K_1-dimensional $V^{(1)}$ and K_2-dimensional $V^{(2)}$ subspaces in R^n, let $v_1^{(i)}, \ldots, v_{K_i}^{(i)}$ be an orthonormal basis of $V^{(i)}$, $i = 1, 2$. The *subspace distance* between $V^{(1)}$ and $V^{(2)}$ is

$$d(V^{(1)}, V^{(2)}) = \sqrt{max(K_1, K_2) - \sum_{i=1}^{K_1} \sum_{j=1}^{K_2} (\langle v_i^{(1)}, v_j^{(2)} \rangle)^2}.$$

In [8], it is shown that the subspace distance defined above is a Euclidean distance function, and *is independent of the choices of the orthonormal bases*, $v_1^{(i)}, \ldots, v_{K_i}^{(i)}$. Further, $0 \leq d(V^{(1)}, V^{(2)}) \leq \sqrt{max(K_1, K_2)}$, where $d(V^{(1)}, V^{(2)}) = 0$ if and only if $K_1 = K_2$, and the two subspaces coincide, and $d(V^{(1)}, V^{(2)}) = \sqrt{max(K_1, K_2)}$ when two subspaces are perpendicular to each other[6].

Using the above definition, we compute the pairwise subspace distance between any pair of the 492 blocks[7], and the results are shown in Fig. 5(a): the rows and columns are indexed by the blocks, and a gray scale is used to visually depict the distance: the darker a point (i, j) is, the shorter the distance between the two blocks (B_i, B_j). The rows and columns are sorted so that the blocks with likely similar behaviors are located closer to each other. The figure clearly shows clusters of blocks with likely similar behaviors. To extract the clustering structure and classify the block behaviors, we first apply the

[6] Note that when $K_1 = K_2 = 1$, then $d(V^{(1)}, V^{(2)}) = d(v_1^{(1)}, v_1^{(2)}) = \sqrt{1 - \cos^2 \theta} = \sin \theta$, where θ is the angle between the two vectors $v_1^{(1)}$ and $v_1^{(2)}$ (two 1-dimensional space). Hence, in a sense we can consider $\arcsin d(V^{(1)}, V^{(2)})$ as a *generalized angle* between two subspaces. In statistical literature, another generalization of angles, a series of *principal angles* (typically defined when $K_1 = K_2$) instead of a single metric, are used to measure similarity/dissimilarity between subspaces. In the next section, we will use (a variation of) principal angles to compare, characterize and track similarity/dissimilarity among individual singular vector components of the two subspaces, $V^{(1)}$ and $V^{(2)}$.

[7] We only focus on 492 blocks which contains at least 10 active hosts with at least 10 observed flows originated from each host in a day.

classical scaling method [9] and then the *K-means* clustering algorithm (see, e.g., [10]) for dimension reduction and clustering.

As the subspace distance is a Euclidean distance function, it implies that we can embed the blocks (or more precisely, their corresponding subspace representation) as points in an M-dimensional Euclidean space R^M, where $M = 492$, the total number of the blocks. In this embedding, the Euclidean distance between two points is exactly the subspace distance between the two corresponding blocks. The classical scaling method allows the recovery of the (intrinsic) coordinates of these points in R^M from the *squared distance matrix* of these points (up to a rotation and translation) [11]. The $M \times M$ squared distance matrix is given by $D^{(2)} = [d_{ij}^2]$, where $d_{ij} = d(V^{(i)}, V^{(j)})$ is the subspace distance between blocks B_i and B_j. Let $J = I - M^{-1}ee^T$, where $e = [1, 1, ...,]^T$ is an M-dimensional all-1 column vector[8], and define the doubly-centered matrix $B_D = -\frac{1}{2}JD^{(2)}J$. Applying eigenvalue decomposition to B_D yields $B_D = W\Lambda W^T$, where $\Lambda = diag[\lambda_1, \dots, \lambda_M]$ is a diagonal matrix containing the eigenvalues (in a decreasing order) of B_D, columns of the orthonormal matrix $W = [w_1, \dots, w_M]$ are the corresponding eigenvectors of B_D, and $W^T W = I$. Then the columns of the matrix $X = \Lambda^{1/2}W^T$, where $\Lambda^{1/2} = diag[\sqrt{\lambda_1}, \dots, \sqrt{\lambda_M}]$, are the coordinates of the M points (blocks) in R^M, and the Euclidean distance of any two points, B_i and B_j, computed using this coordinate system X, is exactly equal to d_{ij}.

Using X, we can directly apply the *K-means* clustering algorithm to extract the cluster structure. However, for large M, this may not be efficient and scalable (as its time complexity is $O(M^3)$). Moreover, the well-known "curse of dimensionality" also leads to unsatisfactory clustering results [12]. Examining the scree plot of the eigenvalues of B_D (see Fig. 5(b)), we see that the M points lie mostly in a low-dimensional space, as their coordinates in the higher dimensions are close to zero. The inset in Fig. 5(b) shows that we can apply the spectral clustering method [13] directly. Using $r = 3$ and $K = 10$, the 10 clusters are represented by either "o" or "+" with different colors, where "+" stands for dense or seemingly dense clusters, and "o" stands for loose clusters). From the figure, we see that there are blocks that are more tightly clustered together, while others are somewhat more loosely clustered. A few blocks are "outliers", having a relatively distance to nearly all other blocks. Due to some randomness in the K-means algorithm, some blocks may be assigned somewhat arbitrarily to one or the other loose cluster, as their distances to other blocks in each of them may be similar. Hence, the ordering how blocks get assigned to each cluster has an impact on the overall clustering results. We have performed K-means clustering algorithms with different seeds, with $r = 3, \dots, 6$, and $K = 5, \dots, 25$. The overall observation remains the same: there are about 3 tightly clustered blocks, and a small group of "outlier" blocks, while other blocks belong to somewhat more loosely associated clusters. The number of clusters and membership of blocks hinge on the parameter K, the seeds, but less so on the dimension r.

Table 1 summarizes a sample clustering result with $r = 3$ and $K = 10$, where the intra-cluster distance is the average distance between blocks within the same cluster, and the inter-cluster distance is the average distance between blocks within the same

[8] J is often referred as the cantering matrix, as multiplying a matrix by J on both sides produces a matrix that has $0 - mean$ columns or rows.

ID	Label	Intra-dist.	Inter-dist.	# blocks	Dominant Src. Ports	Dominant Dst. Ports	Details
1	web client-centered	0.28	1.26	83	*aoSrcPort*	80	Academic departments. No servers
2	web server-centered	0.99	1.29	13	80, 25, 443	*aoDstPort*	CSE and ITLabs. with multiple web servers
3	non-web-dominated	1.01	1.42	28	Mail, p2p (no port 80)	*aoDstPort*	CSE, ITLabs and SuperComputing. No web traffic.
4	mixed web clients/web servers	1.1	1.59	51	80, *aoSrcPort*	80, *aoDstPort*	Departmental office client machines with web activities and at least one web server.
5	mixed web clients/servers	1.09	1.34	57	80, 25, 22	80, *aoDstPort*	Departmental office client machines with web activities and different types of servers.
6	diversified web clients	1.32	1.48	69	non-service random ports	80 and non-service random ports.	Web clients along with other client traffic.
7	web and p2p clients	1.31	1.44	79	p2p	80 and p2p	Client machines with more diverse behaviors and non-web-dominance.
8	mixed client behaviors	1.63	1.61	9	p2p and IM	80, p2p, mail and IM	Client machines with more diverse behaviors and non-web-dominance.
9	mixed clients and servers	1.52	1.59	16	80, 25, IM and p2p	80,IM and p2p	Residential halls and wireless blocks with very diversified traffic
10	outliers	1.64	2.13	6	special-service-ports	special-service-ports	Blocks with widely different behaviors.

Table 1. Summary of clustering results.

cluster and those outside the cluster. Each cluster is assigned a "high-level" *descriptive label*, based on the interpretation of common behaviors shared by most blocks in the cluster. The interpretation is derived by manually examining the ports with high *energy* (i.e., v_{ij}^2) in the top singular vectors of the blocks within a cluster. The blocks within the first three clusters are most tightly clustered, exhibiting more cohesive behaviors. The blocks within the *client-centered* cluster are characterized by the fact that most energy is concentrated on two points, *dstPort 80*, and *aoSrcPort*, and the combined energy of these two ports often exceeds 90% of the total energy. Blocks within this cluster often belong to academic departments. In contrast, the *web-server-dominated* cluster is characterized by the fact that highest energy is concentrated on *srcPort 80* and *aoDstPort*. Blocks within this cluster demonstrate somewhat more diversity (compared to those within the client-centered cluster), as some blocks may also exhibit higher energy on other service ports, such as *srcPort 443*, *srcPort 25*. The blocks within this cluster belong to the university and academic departments/colleges (e.g., CSE, IT) where multiple web servers are hosted. Blocks within the third *non-web-dominated* cluster are characterized by *lack of* high energy (often 5% or less) on either *dstPort 80* or *srcPort 80*. Many of the blocks within this cluster belong to lab machines (e.g., CSE and IT labs) and the supercomputing center, where users of these machines do not routinely use them for web surfing.

The fourth cluster, *mixed web clients/web servers*, contains blocks with high energy on both *srcPort 80* and *dstPort 80* (at least 15%) as well as on *aoSrcPort* and *aoDstPort*. These blocks typically comprise client machines with predominantly web surfing activities, together with at least one web server. The next cluster, *mixed web clients/servers*, is similar to the previous cluster, in that they contain blocks with high energy on both *srcPort 80* and *dstPort 80* (at least 15%). They differ from those in the previous cluster in that they have high energy only on *aoDstPort*, not on *aoSrcPort*; in addition, they also contain relatively high energy on a few other source service ports such as *srcPort 25* (email) or *srcPort 22* (ssh), suggesting that these blocks may contain other server-related activities. Most blocks of the previous two clusters belong to academic departments, containing typical office client machines as well as web or other servers.

The sixth cluster is a loosely clustered which, in addition, to having a high energy on dstPort 80, it contains diversified (seemingly random) ports both for destination and source ports. Unlike cluster 1, this cluster does not have *aoSrcPort* as dominant source port. Most probably, this cluster depicts behaviors of blocks for which the member clients initiate web connections using random, yet frequently-used ports in our campus traffic.

Blocks within the next three clusters are more loosely clustered. Their behaviors are characterized by far less dominant web client activities and no dominant web server activities. Cluster 7 (*web and P2P clients*) is characterized by relatively high energy on various source and destination P2P ports, while Cluster 8 (*Mixed client behaviors*) in addition contains high energy on some IM (instant messaging) and other related ports. But blocks in neither cluster contain relatively high energy on any of the standard source service ports, suggesting that they contain client machines with more diverse behaviors, where web activities are no longer dominant. The behaviors of the blocks in Cluster 9 (*mixed clients and servers*) are more diverse, with energy spreads not only across a number of P2P and IM source/destination ports but also on srcPort 25, and so forth. Most of these blocks belong to student residential hall networks, university wireless and library network blocks. The last cluster contains essentially a few "outlier" blocks, where their behaviors are quite distinct due to being used for special purpose. For example, one of the blocks within this cluster contains two very active PlanetLab machines, with a wide range of ports frequently being used. Another example is a block being used to conduct credit card transactions. We conclude this section by emphasizing that the goal of our clustering of block behaviors is not to generate a precise classification, but to produce some high-level "descriptive" labels and provide a "big picture" view of the block-wise behaviors in a campus/enterprise network so as to assist network operators and security analysts to better monitor and manage the network.

5 Tracking Block Behaviors over Time

In this section we show how we can use the subspace representations of block behaviors to track changes in their behaviors over time, and detect major changes that may be indicative of potential attacks or other anomalies.

5.1 Methods for Tracking Behavior Changes over Time

Given a block B, let $V^{(t)}$ denote the subspace representation of its behaviors extracted at the tth time interval (say, the tth day). We can use the subspace distance, $d(V^{(t)}, V^{(t+1)})$, to compare and track the changes in the behaviors of block B over time. For simplicity of notations, we will use V and W instead of $V^{(t)}$ and $V^{(t+1)}$ to denote the subspace representations of a block B at the two consecutive time intervals t and $t+1$, respectively. Let $V = [v_1, \ldots, v_K]$ and $W = [w_1, \ldots, w_L]$, where v_i's and w_i's are the corresponding top singular vectors. Clearly, if $K \neq L$, then $d(V, W) \geq 1$, a relatively large distance. Even when $K = L$, we may still expect a relatively large distance. Fig.6(a) shows an example of the subspace distance of each of the 492 blocks in two consecutive days: the x-axis is indexed by the blocks in the increasing order of the subspace distance between the two days; the solid curve is the subspace distance of the blocks (the left y-axis is the corresponding scale), and the dotted "zigzag-like" curve represents $|K - L|$–the difference in the numbers of dominant singular vectors in two days (the right y-axis is the corresponding scale). We see that about 170 blocks having a subspace distance less than 1, many of which have a distance close to 0, indicating relatively little changes in their behaviors over two days. The majority of blocks (about 300) have a subspace distance between 1 and 2, while a few blocks have a subspace distance larger than 2. There are about 215 blocks (nearly half of the blocks) with $K = L$: the majority of these blocks have a subspace distance less than 1; however, for a few of them, the subspace distance can be as large as 1.5. All the remaining blocks have different numbers of (dominant) singular vectors in the two days, with a subspace distance of at least 1.

Clearly, in general, large distance signifies major changes in its behaviors. However, the subspace distance *in itself* does not tell us what may have changed that causes a large distance from time interval t to $t + 1$ interval. To address this issue, we develop two methods which provide more detailed information to quantify and track the behavior dynamics of blocks over time. In the first method, we consider each individual singular vector, w_j, $1 \leq j \leq L$, at time interval $t + 1$ and compute its distance to the (entire) subspace V of the previous time interval t as follows:

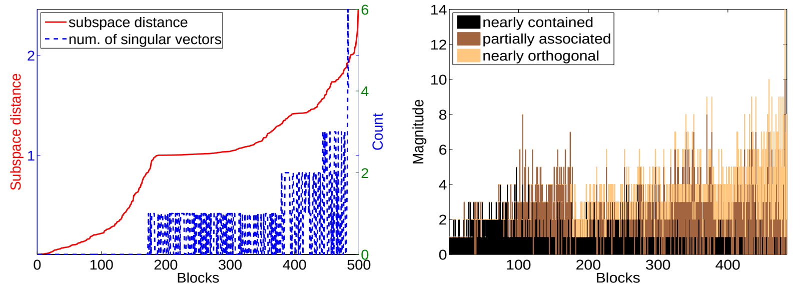
$$d(w_j, V) = \sqrt{1 - \sum_{i}^{K} \langle v_i, w_j \rangle}. \tag{1}$$

Eq.(1) can be viewed as measuring the (generalized) *angle* Θ between the vector w_j and the subspace V, where $\arcsin \Theta = d(w_j, V)$. Hence, if $d(w_j, V)$ is close to 0, say, $d^2(w_j, V) \leq \epsilon$ for some small $\epsilon > 0$ (we use $\epsilon = 0.1$ in all experiments), then w_j is *(nearly) contained* in V. In other words, the behavior captured by w_j (namely, the associated port activities) in the time interval $t + 1$ can be nearly fully represented by those in the previous time interval, i.e., V. On the other hand, if $d(w_j, V)$ is close to 1, say, $d^2(w_j, V) \geq 1 - \epsilon$, then w_j is *(nearly) orthogonal* to V (i.e., $\theta \approx \pi/2$), and thus the behaviors captured by w_j is almost totally different from those represented by V. When $d(w_j, V)$ lies in between (say, $\epsilon \leq d^2(V, W) \leq 1 - \epsilon$), the behaviors captured by w_j contains both “old” port activities that are similar to those in V, but also “new” ones that are not. In this case, we say w_j is *partially associated* with V. Therefore, by considering individual w_j’s and computing their distances to V, we can classify w_j into three categories, *(nearly) contained*, *partially associated*, or *(nearly) orthogonal*, and use this classification to identify and quantify those w_j’s, or *new* behaviors, that cause large changes in the subspace distance.

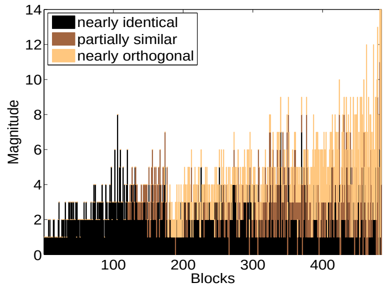
In the second method, we go one step further by comparing individual w_j’s with individual v_i’s. To characterize the relations between w_j’s and v_i’s, we introduce a sequence of *principal angles* [14], $(0 \leq)\theta_1 \leq \cdots \leq \theta_{\max\{K,L\}}(\leq \pi/2)$, defined *recursively* as follows: 1) $cos\theta_1 = \max_{1 \leq i \leq K}$
$\max_{1 \leq j \leq L} \langle v_i, w_j \rangle$; namely, θ_1 is the *smallest* angle formed by any pair of v_i’s and w_j’s. We denote the pair of vectors associated with θ_1 as (v_1^*, w_1^*). 2) More generally, for $k \geq 2$, θ_k is the smallest angle between the (remaining) pairs of v_i’s and w_j’s *after* we have removed the pairs associated with $\theta_1, \ldots, \theta_{k-1}$. Note that if $K = L$, then each w_j is paired with one v_i. If $K \neq L$, then either some w_j’s (when $K < L$) or v_i’s are left. Hence, for $k = \min\{K, L\} + 1, \ldots, \max\{K, L\}$, we define $\theta_k = \pi/2$. For $1 \leq k \leq \min\{K, L\}$, the pairs of vectors (v_k^*, w_k^*)’s that are associated with the

principal angles θ_k's provide the best *matching* between the behaviors represented by v_i's and w_j's: v_1^* and w_1^* represent most similar behaviors in the time intervals t and $t+1$, v_2^* and w_2^* the second most similar, and so forth.

We can therefore classify the relations between v_i's and w_j's using θ_k's. If there exists $\kappa_1 \leq \min\{K, L\}$ such that for $k = 1, \ldots, \kappa_1$, θ_k is close to 0 (say, $\sin^2 \theta_k \leq \epsilon$), then w_k and v_k^* are *(nearly) identical*. Thus they represent behaviors that do not change very much from t to $t+1$. On the other hand, if there exists $\kappa_2 \leq \min\{K, L\}$ such that for $\kappa_1 \leq k \leq \min\{K, L\}$, θ_k is close to $\pi/2$ (say, $\sin^2 \theta_k \geq 1 - \epsilon$), then w_k and v_k^* are *(nearly) orthogonal* and any values between identical and orthogonal are considered to be *partially similar*. Thus these pairs represent nearly distinct behaviors that were present in the time interval t but not $t+1$, and vice versa. In addition, when $K \neq L$, any *unmatched* v_i's or w_j's represent "old" behaviors that have disappeared or new behaviors that have emerged. Hence, the principal angles provide us with a finer grain method to quantify and track the changes in block behaviors over time.



(a) Correlation between subspace distance and change in no of singular vectors (b) Percentages of the 3 different categories of pairs



(c) Percentages of the 3 different categories of distances

Fig. 6. Subspace distance between two consecutive days

We apply the two methods described above to classify and quantify the changes in block behaviors over the same two days as in Fig. 6(a). The results are shown in Fig. 6(b) and Fig. 6(c), respectively, where the number of singular vectors (or vector pairs) belonging to each of the three categories is plotted, and the total height of the curve is the maximum number ($\max\{K, L\}$) of singular vectors in each block. The block indices (the x-axis) are fixed in the same order (i.e., increasing subspace distance) as in Fig. 6(a). As is clear from Fig. 6(b) and Fig. 6(c), for blocks with very small subspace distance (say, block 1 - block 100) between the two days, almost all w_j's of the second day are *nearly contained* in the subspace V of the previous day (the dark shaded area under the curve in Fig. 6(b)). Furthermore, most "best-match" vector pairs (v_i^*, w_i^*)'s are *nearly identical* (the dark shaded area under the curve in Fig. 6(c)), with the remaining pairs at least *partially similar*. As the subspace increases further, more w_j's become *partially associated* with V, suggesting that w_j's capture some new activities or changes in behaviors in the underlying hosts of these blocks. For block 171 or higher (where the subspace distance of the two days is larger than 1), we see that at least one w_j becomes *nearly orthogonal* to V, or equivalently, there exists either an unmatched singular vector (when $K \neq L$) or at least a pair (v_k^*, w_k^*) that is *nearly orthogonal* (both are indicated by the light shaded area under the curve in each of the figures). In particular, for blocks where the subspace distance is at least 2 (block 450 and higher), nearly all singular vectors w_j's in the second day are orthogonal to V, and to the individual singular vectors v_i's of the previous day. Hence, they suggest that hosts in these blocks may have almost completely different behaviors in these two days.

The results show that using these two methods, we can quantify and track the change in block behaviors over time, and identify specific activities (e.g., as embodied by the nearly orthogonal w_j's) that cause any major changes. We have performed similar analysis to compare, quantify and track the changes in block behaviors over time using two-week long data, and obtained qualitatively similar results. Due to space limitation, we do not present them here.

5.2 Anomaly and Attack Detection

In this subsection, we show that by tracking and identifying major changes in block behaviors, the methods we have developed in the previous subsection can be used to detect potential anomalies and attacks. We demonstrate this capability through *attack emulation*, where we inject certain types of attacks or other anomalous activities into a block with otherwise "normal" activities. We have performed this study using a range of anomalies. Due to space limitation, however, we briefly describe the outcomes under three common types of attacks/anomalies: *outside scanning*, *back-door trojan activities*, and *DDoS ping flood attacks*.

For the scanning attack scenario, we assume that an outside scanner is sending traffic to all active hosts within a block which tends to trigger a response to this scanning with a small number of flows (one or two flows) using the same incoming port. Hence, the activities of the block have a new source port which reflects an additional activity. Unlike the change of subspace distance caused by activities being modified, i.e. addition of one or more activities or making an activity more or less significant, the change caused by response to scanning changes the activity to a totally different host behavior associated with a drastic increase in subspace distance. We observe that blocks injected with such scanning activity experience relatively high increase in subspace distance. More specifically, we notice that server-dominated blocks which usually experience little (or no) increase in subspace distance ≤ 0.05, their subspace distance suddenly jumps to values greater than 2 since now the dominant source port is no longer the original source port corresponding to the service provided by the host, it is now the scanning port which is more significant and dominant.

We also emulate traffic for an inside client host suddenly acting as a server establishing and accepting connections at some port (e.g. 80). This type of back-door attack is usually used by bots, in which the client host daily registers the same IP address with several different domains chosen from a specific list of domains. Consequently, bots search the whole list to find the IP address to communicate with For client-dominated blocks, we see that they transform into a server-like behavior dominated by the source port corresponding to the bot port which causes a sudden increase in the subspace distance which is far greater than the normal range of change for client blocks which might otherwise have a slight increase in subspace distance caused by additional activities.

Finally, we tested our method for distributed denial of service (DDoS) attacks in which an attacker sends an ICMP ping packets to an inside (potentially server) host and listens for responses. This causes the server to start responding to multiple requests which will now be dominated by both incoming and outgoing ICMP ports. The nature of DDoS attacks involves a large number of ICMP requests (hence the name ping flood) and the inside server now is no longer dominated by the port it is meant to service, but rather by ICMP source and destination ports. Consequently, such (usually server) blocks will experience a sharp increase in the subspace distance caused by DDoS attacks.

The above discussion illustrates the potential utility of our block-level behavior profiling and tracking methodology in detecting anomalies and attacks, in addition to providing better knowledge of the "normal" activities and their changes over time within a network. Clearly, the resulting anomaly and attack detection uncovered by tracking the subspace distance over time of a block is only meant for *post-mortem* analysis or "after-fact" discovery of attacks or anomalies, not *real-time* detection. Thus, it is complementary to firewalls and other IDS (intrusion detection system) and IPS (intrusion prevention system) that are commonly deployed in large campus and enterprise networks. Our technique is particularly useful in uncovering compromised *inside hosts* that are (frequently, periodically or even occasionally) used to launch attacks or other illegitimate activities targeted at the outside Internet.

6 Conclusions

In this paper we have developed a novel methodology for profiling and tracking network activities at the *block-level*. By capturing and characterizing significant and typical host behaviors within a block of contiguous IP addresses associated with subnets where many hosts often have similar usage patterns, the proposed methodology is more scalable and can effectively handle the difficulty in tracking individual host behaviors due to dynamic addresses. We introduced a *block-wised (host) port activity matrix (BPAM)* which represents the activities of individual hosts within a block on a carefully selected port feature vector. Applying the SVD low-rank approximation technique, we obtained a low-dimensional subspace representation which captures the significant and typical host activities of the block. Using these subspace representations, we clustered and classified blocks to provide high-level descriptive labels to assist network operators and security analysts to gain a "big-picture" view of the network activities. We also developed novel methods to track and quantify changes in block's behaviors over time, and demonstrated how these methods can be utilized to identify major changes and anomalies within the network.

References

1. K. Xu, Z.-L. Zhang and S. Bhattacharyya. Profiling Internet Backbone Traffic: Behavior Models and Applications. In *Proc. of ACM SIGCOMM*, August 2005.
2. T. Karagiannis, K. Papagiannaki and M. Faloutsos. BLINC: Multilevel traffic classification in the dark. In *Proc. of ACM SIGCOMM*, August 2005.
3. K. Xu, Z.-L. Zhang, and S. Bhattacharyya. Internet traffic behavior profiling for network security monitoring. In *IEEE/ACM Trans. Netw.*, 2008.
4. Y. Xie, F. Yu, K. Achan, E. Gillum, M. Goldszmidt, and T. Wobber. How Dynamic are IP Addresses? In *Proc. of ACM SIGCOMM*, 2007.
5. Y. Jin and E. Sharafuddin and Z.-L. Zhang. Identifying Dynamic IP Address Blocks Serendipitously through Background Scanning Traffic. In *Proc. of ACM CoNext'07*, 2007.
6. Haakon Ringberg, Augustin Soule, Jennifer Rexford, and Christophe Diot. Sensitivity of pca for traffic anomaly detection. *SIGMETRICS Perform. Eval. Rev.*, 35(1):109–120, 2007.
7. B. Rubinstein et al. ANTIDOTE: Understanding and Defending against Poisoning of Anomaly Detectors. In *Proc. of ACM IMC*, 2009.
8. L. Wang et al. Subspace distance analysis with application to adaptive bayesian algorithm for face recognition. In *Pattern Recogn.*, volume 39, pages 456–464, 2006.
9. I. Borg and P. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer Series in Statistics, 2005.
10. R. Duda, P. Hart, and D. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.
11. S. Lee, Z.-L. Zhang, S. Sahu, and D. Saha. On suitability of euclidean embedding of internet hosts. In *SIGMETRICS Perform. Eval. Rev.*, 2006.
12. F. Korn, B. Pagel, and C. Faloutsos. On the 'dimensionality curse' and the 'self-similarity blessing'. In *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2001.
13. Ulrike von Luxburg. A tutorial on spectral clustering. *CoRR*, abs/0711.0189, 2007.
14. Zlatko Drmac. On principal angles between subspaces of euclidean space. In *SIAM Journal on Matrix Analysis and Applications*, volume 22, pages 173–194, 2000.