博士論文

Unsupervised Structural Pattern Analyses for the Management of Network Traffic and Configurations

(ネットワークのトラフィックおよび構成のマネジメントにおける
教師なし構造パターン分析)

肥村 洋輔

Abstract

Computer networking, particularly the Internet, has become an essential platform for human life and many aspects of industries. Such the successful growth of the computer networking, instead, insists upon the importance of its efficient management such as traffic management and configuration management. On the other hand, one of recent expectations towards efficient network management stems from the recent fashion of statistical data analytics such as Artificial Intelligence (AI), whereas one difficulty in applying statistical data analyses is derived from its low-level numerical nature differing from actual domain-specific data set. An insight is that the widespread of computer networking has been contributing to data of inter-similar objects (e.g., a number of hosts and/or a number of multiple virtual networks over the shared infrastructure), which can be compared each other to uncover knowledge (e.g., typical patterns and atypical anomalous patterns); This indicates the potential usefulness of unsupervised analyses (e.g., cluster analysis), which extract similar patterns inside data itself without established knowledge database pre-defined by human experts. However, orthodox unsupervised approaches based on extracting and clustering numerical values (i.e., feature vectors) face the difficulty in interpretation of resulting outputs, which is important in management domain in order to take appropriate actions against the outputs. In this research, we study unsupervised approaches with structural patterns (e.g., graph structure), which is more interpretable (e.g., by visualizations) than only using numerical features.

In Chapter 2, we present structural pattern analysis on network traffic data. This analysis is demonstrated with the traffic data obtained in a measurement point in the Internet. We particularly focus on end-host profiling by analyzing network traffic, which is a major stake in traffic engineering. The use of *graphlet* for end-host traffic analysis is efficient for interpreting host behaviors, which essentially consists of a visual representation as a graph. However, graphlet analyses face the issues of choosing between supervised and unsupervised approaches. The former can analyze a priori defined behaviors but is blind to undefined classes, while the latter can discover new behavioral patterns at the cost of difficult a posteriori interpretation. This work aims at bridging the gap between the two. First, to handle unknown classes, unsupervised clustering is originally revisited by extracting a set of graphlet-inspired attributes for each host. Second, to recover interpretability for each resulting cluster, a *synoptic graphlet*, defined as a visual graphlet obtained by mapping from a

cluster, is newly developed. Comparisons against supervised graphlet-based, port-based, and payload-based classifiers with two datasets demonstrate the effectiveness of the unsupervised clustering of graphlets and the relevance of the a posteriori interpretation through synoptic graphlets. This development is further complemented by studying *evolutionary tree* of synoptic graphlets, which quantifies the growth of graphlets when increasing the number of inspected packets per host.

In Chapter 3, we present structural pattern analysis on network configuration data. This analysis is demonstrated with the configuration data in a multi-tenant datacenter network, where multiple customer (tenant) networks are virtualized over a single shared physical infrastructure. The use of multi-tenancy is cost-effective but poses significant costs on manual configuration. Such tasks would be alleviated with configuration templates, whereas a crucial difficulty stems from creating appropriate (i.e., reusable) ones. In this work, we propose a graph-based method of mining configurations of existing tenants to extract their recurrent patterns that would be used as reusable templates for upcoming tenants. The effectiveness of the proposed method is demonstrated with actual configuration files obtained from a business datacenter network.

In Chapter 4, we also present future perspectives for structural pattern analysis for the management of actual field domain. Contrary to the previous chapters (Chapters 2 and 3) that deal with traffic and configuration data in the computer networking domain, this chapter discusses the possibility of extending those analytical approaches to the field domains (e.g., building), which have become more important as more field devices become connected. We consider that the spirit of using data analytics for the management of end-hosts (by traffic analysis) and virtual network configurations (by configuration analysis) should have commonality to some extent for the management of field domains such as to manage end-devices (e.g., sensors and actuators) and their inter-link configurations. However, analytics on data obtained in field domains is relatively difficult due to the differences in the characteristics of those domains (e.g., open systems for field domains, rather than closed systems in computer networking domain), particularly leading to the difficulty in interpretation of analytical results. In addition, characteristics of data in computer networking fields are rather aggregated view of discrete and relative information (i.e., aggregated cardinality), which would be suitable for the discrete structural model (e.g., graphlet), whereas field data additionally suggests the importance of individual point-by-point time-series view with non-discrete and absolute values. We discuss the need for trial-and-error approach of analyzing field data (e.g., per-room temperature in a building), interpretation of which is gradually complemented by introducing multiple types of contexts (e.g., switch states of air-conditioning appliances in rooms) and even other data sources (e.g., external temperature outside the building) that are considered as structurally related. This approach is demonstrated with an unsupervised pattern analysis over an actual dataset about Electrical Heat Pump (EHP) equipped in a number of rooms in a building.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computer networking, particularly the Internet, has become an essential platform for human life and many aspects of industries. The Internet has been still growing, allowing the participation and intercommunication of a variety of devices (including end-hosts and central-hosts of different domains), and growing number of applications are deployed to process and exchange a variety of data over the Internet, having been resulting in playing the significant role of supporting diverse human activities. Such the successful growth of the Internet, instead, insists upon the importance of its efficient management such as traffic management and configuration management. On the other hand, one of recent expectations towards efficient network management stems from the recent fashion of statistical data analytics, as recently represented with Artificial Intelligence (AI), usefulness of which has been studied in prior arts, whereas a difficulty in applying statistical data analyses is derived from its low-level numerical nature differing from actual domain-specific data set. An insight is that the widespread of the computer networking has been contributing to similar accessible data (e.g., obtained from a number of hosts and/or multiple virtual networks over the shared infrastructure), which can be compared each other to uncover knowledge (e.g., typical patterns and atypical anomalous patterns), which will then be used for efficient managements; This indicates the potential usefulness of unsupervised analyses (e.g., cluster analysis), which in general extract similar patterns inside data itself without established knowledge database pre-defined by human experts. However, conventional unsupervised approaches of extracting and clustering numerical values (i.e., high-dimensional feature vectors) face the difficulty in interpretation of resulting outputs, which is important in management domain in order to take appropriate actions against the outputs. In this research, we study unsupervised approaches with structural patterns (e.g., graph structure), which is more interpretable (e.g., by visualizations) than only using numerical features.

1.1 Computer networks and their management

1.1.1 Computer networks

Computer networking, particularly the Internet, has been growing with tremendous number of computing devices used in human daily life and many aspects of industrial activities. Starting from connecting computers in a few universities, over the course of several decades, the Internet has been accepting connections from computing servers in facilities and personal computers in offices and homes; In addition to those computing devices with fixed location and mostly wired connections, the Internet has been still growing along with emerging computing devices with new communication media such as mobile and smart devices with mobile wireless connections in outside field domains (and possibly wearable devices, sensor devices, and actuator devices for the near future).

The usage of computer networking has also been diverse. The networked domains include the one connecting servers and terminal computers in an office (i.e., office networks), inter-connecting different offices within a campus (i.e. campus networks), inter-connecting multiple remote sites (i.e., wide-area networks, or carrier networks) such as for the use of business applications, file sharing, remote conferencing. Other examples include connecting a number of computing servers in datacenter (i.e., datacenter networks) such as for large scale data processing, and providing connectivities to mobile computing devices to the Internet (i.e., mobile networks). In addition, upcoming expectation with the recent fashion of Internet of Things (IoT), is to provide Internet connectivities to fixed and/or mobile sensing/actuating devices (i.e., field networks) associated with appliances that had not been previously networked, to provide new opportunities to produce unmet values.

Another aspect of recent trends in computer networking is stemmed from virtualization; Here, virtualization is in general a form of logically realizing multiple separated networks over a single shared physical network. Well-known Virtual Local Area Network (VLAN) and related virtualization technologies has been widely used for office networks and campus networks, and Virtual Private Network (VPN) for enterprise wide-area networks. Also, there has recently been a number of researches for Network Function Virtualization (NFV) especially for carrier networks, and Software Defined Networking (SDN) especially for office and datacenter networks. The primary benefit of such the virtualization is to provide flexibility of access controlling and communication qualities, as well as to be cost-effective by consolidating multiple networks in a single shared infrastructure.

1.1.2 Network management

Such the successful growth of the Internet, instead, insists upon the importance of its efficient management, which is an activity of maintaining desired running status of networks in charge. As mentioned above, the Internet has been broadly

used by tremendous number of computing devices, for a variety of usages, over the shared infrastructure (e.g., by virtualization); In such the situation, the impact of incidents such as the ones in capacity, security, configurations would be significant, simultaneously affecting a number of computing devices and applications. Network engineers have been put their efforts on network managements.

Representative activities of network management are to deal with traffic and configuration[1]; Here, traffic is an aggregated collection of data flowing over the computer networks (i.e., the packet data generated from end-host devices participating in the Internet), and configuration is relationship among intermediate networking devices (e.g., switches, routers, firewalls, load balancers, security appliances), specifically wirings and settings of those devices. Dealing with traffic is essential for understanding trends in network usage (e.g., applications) for capacity planning; Handing configuration is also important for providing correct connectivities among devices and efficient traffic engineerings (e.g., access controls).

In particular, an important task in management of traffic and configurations is to detecting misbehaviors by profiling network activities. One of the important aspect of traffic management is to find such the misbehaviors in end-hosts, examples of which include the ones significantly consuming network resources (e.g., high volume of traffic over limited link capacity), using inappropriate applications (e.g., prohibited ones inside a managed domain), and performing malicious activities (e.g., port scanning and flooding attacks), which can possibly be observed in aggregated traffic data. On the other hand, one of the important aspect of configuration management is to find misbehaviors in intermediate networking devices, representative example of which is inappropriate configurations (e.g., unstandardized one and possibly misconfigurations – to proactively avoid this network engineers have put significant efforts on configuration managements). The sharing and virtualization of networkings has made network management more complicated, and efficient management of network configurations is required.

1.2 Statistical analyses for network management

1.2.1 Expectations

Data analytics is one of the recent expectations. This expectation is often noted as BigData as well as Artificial Intelligence (AI). This expectation is supported with recent advancement of Machine Learning technologies, as well as the increasing amount of data obtainable in individual domains.

[1] A major classification is derived from FCAPS, which represents fault management, configuration one, accounting one, performance one, security one. This classification is based on individual tasks, while our classification is based on data (e.g., traffic data and configuration one) dealt with network engineers to conduct those tasks. For example, measuring and analyzing traffic data will contribute to efficient management for security management by detecting malicious activities.

It has been considered that rich amount of data with appropriate statistical analytics will enable machines to acquire knowledges and skills for intellectual tasks such as image recognition, natural language processing, and possibly system managements. Data analytics with automated computation are characterized with a few advantages against the past literature (i.e., manual tasks); One is the ability to automatically analyze data, the amount of which cannot be dealt with by humans; Another is its statistical nature of finding knowledge from data themselves, which enables to find unknown and new behaviors.

In addition, the use of statistical data analytics is often considered effective with the context of network management (and especially traffic management); Statistical approaches are capable of handing encrypted or no-payload traffic, analyses of which have been difficult with the conventional payload inspection methods. Possible use cases of data analytics in network management includes, for example, detecting malicious behaviors of end-hosts and failures in networking devices by analyzing network traffic, and extracting best-practices and inappropriate configurations by analyzing know-hows of human network engineers.

1.2.2 Potential usefulness of unsupervised analysis

On the other hand, it is in general difficult to apply statistical analyses to actual domains including network management. Event though network managers have started to measure and store related data (e.g., traffic and configurations), such the data-driven approaches often suffer from the gap between the low-level numerical spaces of general-purpose statistical methods and the domain-specific actual data. This gap will poses a hurdle on the first step of analysis of determining how to analyze the data with which methods.

One insight towards this issue would be the characteristics of data that there become more similar objects within individual domains. For instance, as computer networks are connected from increasing number of devices, traffic data measured over the shared network can be regarded as the activity of individual end-hosts. For another instance, as increasing number of virtual networks are deployed over a single shared physical infrastructure (e.g., virtualized system networks in a datacenter), configuration data in the shared infrastructure can be regarded as the configurations of individual virtual networks.

This characteristics of data suggests the potential usefulness of unsupervised analysis. In a broad view, statistical data analysis can be categorized as supervised analysis and unsupervised one. Both are common in that each objects in dataset are abstracted as a set of numerical values (i.e., feature vector) that will represent the characteristics of object (e.g., traffic of hosts are processed into feature vectors), which is then discussed through statistical manipulations. The difference between the two is: The supervised analysis relies on pre-fined dataset about the statistical characteristics of objects (e.g., learning dataset) to classify unlabeled objects (that are statistically similar to one of pre-defined) and/or to detect anomalies (that are

extracted as outliers from any of the pre-defined objects); The unsupervised analysis extracts similarity and dissimilarity among objects inside the data itself (without pre-defined knowledges) to detect patterns (clusters of object that are statistically similar) and anomalies (objects that are not similar to any of the others) in data. As the supervised approach requires pre-defined dataset (and rather purpose-specific such as classifications with known labels), the unsupervised approach should be suitable for handling with the data of such the characteristics as it can extract patterns and anomalies only from the data, each of which will be for instance used for trend analysis and also misbehavior detection (including unknown or emerging behaviors).

1.2.3 Difficulty in interpretations

However, an issue in the unsupervised approach is the difficulty in interpretation of resulting outputs. The common type of outputs is a set of clusters of numerical feature vectors that are considered as similar in the feature space. Interpreting the meanings of each cluster (e.g., type of end-hosts in traffic) requires to investigate the features. Different from image analysis and natural language processing fields, data obtained in network management domain (e.g., traffic, or dump of network packets), in general, is low-level and not suitable for human recognitions, as those networking data are originally for computer processing. Special-skilled network engineers may be able to deal with such the low-level data but those data are not the same as the original traffic but the one abstracted as numerical feature vectors losing meanings in the original context that the original dataset might retain.

Interpretation in the network management domain is important. Most of use cases in network management (e.g., misbehavior detection) requires appropriate actions such as to stop illegitimate traffic and to detaching failure networking devices. Indeed, there should be uses cases where interpretation has relatively less priority, examples of which include product recommendation (e.g., collaborative filtering analysis) in Electric Commerce (EC) sites that would put priority on the increase in sales rather than analytical interpretations, and include first increasing accuracy of classification in the machine learning research domains (e.g., earlier stage of image recognition with deep learning). We consider that those use cases are characterized with specific purpose, or Key Performance Indicator (KPI) and/or objective functions, whereas data-driven approach for network management is rather open-ended and requires to talk with data and its analysis results.

1.3 Overview of this research

In this dissertation, we deal with the issues of the difficulty in interpretation with unsupervised data analysis for network management. Specifically, as stated above, primary tasks in network management include traffic management and configuration

management (especially trend analysis and misbehavior detection), and correspondingly we assume that engineers (nor necessarily network engineers) analyze traffic data measured in network link(s) and configuration setting data obtained from networking devices. We assume that those engineers have domain knowledge about computer networking (to a certain extent, and not necessarily specialists) as well as about the use of data analytics software (e.g., R language).

Our basic approach is to use structural information obtained from individual object in the management domain. Here, that structural information is characterized with data model representing interrelation among feature values with networked structure (e.g., graph representation) that are visually accessible to human having domain knowledge. Namely, we consider that domain-specific graph representation should be appropriate in providing the effective visualization for interpretation while preserving analytical capability with existing methods due to its abstract mathematical model.

Specifically, as structural information, we make use of the graphlet model (widely-recognized through Ref. [1]) in traffic analysis, and graph-based virtual network topology representation in configuration analysis. The set of per-host traffic graphlets are analyzed through extracting feature vectors representing its structural aspects (e.g., average number of edges per node), performing unsupervised cluster analysis to produce clusters of hosts that are similar in the structural sense, and then recovering and visualizing graphlet for each cluster based on its cluster centroid that describes the visual characteristics of cluster itself instead of individual hosts in the cluster (named synoptic graphlet). The set of per-virtual network graph representations are analyzed through cluster analysis on the basis of Graph Edit Distance (GED), which represents an aspect of direct distance between a pair of structural graphs (instead of discussing distance between extracted feature vectors), which enables to visually discuss the similarity and dissimilarity among different virtual network configurations.

Scope of this research is as follows. As analysis methods, we study unsupervised one (rather than supervised one). We discuss the application of unsupervised analysis to the network management domain (rather than newly developing the algorithms of machine learning techniques). Data we deal with are derived from networkings management domain such as traffic and configurations (rather than media data such as images, movies, sounds, texts).

1.4 Summary and organization of this dissertation

In summary, we study the use of structural unsupervised analysis for the management of network traffic and configurations. The computer networking, particularly the Internet, has become an essential platform for human life and many aspects of

industries, resulting in the importance of its efficient management such as traffic management and configuration management. Such the management will be supported with the potential usefulness of unsupervised analyses (e.g., cluster analysis) due to the recent trends in network usage (i.e., similar objects are deployed and can be measured within a single management domain). A crucial issue in conventional unsupervised approaches relying on numerical feature vectors stems from the difficulty in interpretation of resulting outputs, which is important in management domain in order to take actions against the outputs. In this research, we study unsupervised approaches with structural patterns (e.g., graph structure), which is more interpretable (e.g., by visualizations) than only using numerical features.

The reminder of this dissertation is as follows. Chapter 2 presents structural pattern analysis on network traffic data, particularly end-host profiling by analyzing network traffic with the graphlet model. Chapter 3 presents structural pattern analysis on network configuration data, particularly finding recurrent patterns in configurations in virtual networks to construct configuration templates. In addition Chapter 4 presents future perspectives for extending the spirit of the structural pattern analysis to actual field domains (in addition to the computer networking domain) by discussing the resemblance and differences in data characteristics between the two domains with actual case study through unsupervised comparative analysis of per-room sensory data obtained in a building. Chapter 5 summarizes this dissertation.

Chapter 2

Structural Traffic Pattern Analysis – Synoptic Graphlet : Bridging the Gap between Supervised and Unsupervised Profiling of Host-level Network Traffic

End-host profiling by analyzing network traffic comes out as a major stake in traffic engineering. Graphlet constitutes an efficient and common framework for interpreting host behaviors, which essentially consists of a visual representation as a graph. However, graphlet analyses face the issues of choosing between supervised and unsupervised approaches. The former can analyze a priori defined behaviors but is blind to undefined classes, while the latter can discover new behaviors at the cost of difficult a posteriori interpretation. This work aims at bridging the gap between the two. First, to handle unknown classes, unsupervised clustering is originally revisited by extracting a set of graphlet-inspired attributes for each host. Second, to recover interpretability for each resulting cluster, a *synoptic graphlet*, defined as a visual graphlet obtained by mapping from a cluster, is newly developed. Comparisons against supervised graphlet-based, port-based, and payload-based classifiers with two datasets demonstrate the effectiveness of the unsupervised clustering of graphlets and the relevance of the a posteriori interpretation through synoptic graphlets. This development is further complemented by studying *evolutionary tree* of synoptic graphlets, which quantifies the growth of graphlets when increasing the number of inspected packets per host. [1]

[1]Contents of this chapter have been published as the following article: (c) 2013 IEEE. Reprinted, with permission, from [Yosuke Himura, Kensuke Fukuda, Kenjiro Cho, Pierre Borgnat, Patrice Abry, and Hiroshi Esaki, "Synoptic Graphlet: Bridging the Gap Between Supervised and Unsupervised Profiling of Host-Level Network Traffic," IEEE/ACM Transactions on Networking, Volume

2.1 Introduction

An essential task in network traffic engineering stems from host-level traffic analyses, where the behavior of a host is characterized based on traffic (i.e., packet sequence) generated from the host. Host-level traffic analyses enable to find users of specific applications for the purpose of traffic control, to identify malicious or victim hosts for security, and to understand the trend of network usage for network design and management. Flow analysis, which also constitutes an important networking stake, can be fruitfully complemented by host profiling (e.g., by breaking down host behaviors into flow characteristics).

Numerous attempts have been made to develop statistical methods for host profiling. Such methods aim at overcoming packet encryption, encapsulation, use of dynamic ports, or dataset without payload – situations that impair the classical approaches relying on payload inspection [2, 3, 4] and port-based rules [5]. The most recently proposed ones are based on heuristic rules [1], statistical classification procedures [6, 7, 8], Google database [9], or macroscopic graph structure [10, 11, 12, 13].

In particular, an effective yet heuristic approach to host profiling is based on *graphlets* [1, 14, 15]. A graphlet is a detailed description of host communication patterns as a graph, as illustrated in Figure 2.1. For each flow, the 5-tuple defining it (proto, srcIP, dstIP, srcPort, dstPort) gives a set of attributes $(A_1, A2, \ldots)$ and the communication pattern of a host is the union, for all flows, of edges connecting nodes associated to flow's attributes. This leads to diverse visual shapes of graphlets depending on the host's flows. The graphlet representation facilitates the intuitive analysis of differences and resemblances among host behaviors, whereas conventional approaches directly handle numerical values of statistical features, which are difficult to interpret.

However, as for any host-profiling approach, the use of graphlets faces the classical issue of choosing supervised versus unsupervised procedures. Supervised approaches rely on a priori determined classes or models of graphlets [1], pre-defined by human experts in a necessarily limited number, and these approaches cannot substantially classify new or unknown host behaviors. Unsupervised approaches are adaptive insofar as the data directly define the output classes of graphlets and can discover behaviors never observed before. These approaches, however, potentially produce clusters composed of a large number of numerical features that cannot receive easy meaningful or useful interpretation.

The present work aims at bridging the gap between the two types of approaches. The main idea for this is the combination of two techniques: To avoid the limitation of supervised approach, we use an unsupervised clustering of graphlets that is able to capture previously unknown classes; To ease the difficulties of unsupervised approach, the resulting clusters are re-visualized into *synoptic graphlets* that allow us to interpret the clusters obtained. Our approach is evaluated with two large datasets

21, Issue 4, August 2013]. (DOI: 10.1109/TNET.2012.2226603)

of traffic collected on two different links (Sec. 2.3). The article is organized along three contributions.

First, the classical problem of supervised classification is revisited (Sec. 2.4). This investigation comprises two aspects: a list of graphlet-based features is proposed to quantify in a relevant way the visual graphlet shape associated with each host; An unsupervised clustering method is applied to these features to yield classification in terms of graphlet shapes. Comparisons against a supervised graphlet-based classifier (BLINC [1]), a port-based one, and a payload-based one allow us to check that most clusters match well-known host behaviors. This result shows that our method makes it possible to discover unknown graphlets, which avoids the problem faced by supervised approaches.

Second, the issue of automatically providing interpretation of the output of the unsupervised clustering is addressed (Sec. 2.5). We solve the *inverse problem* of reconstructing a *synoptic graphlet*, defined as a graphlet inferred from each obtained cluster, by using an original mapping of the cluster attributes (cluster centroid) into a graphlet. Our development of synoptic graphlets shows that an interpretable meaning can be associated automatically to each cluster without any a priori expertise. The effectiveness of synoptic graphlets, which successfully provide interpretability for unsupervised approaches as shown in this work, ends up in bridging the gap between supervised and unsupervised methods.

Third, the nature of host behavior is further studied via synoptic graphlets (Sec. 2.6). The use of synoptic graphlets is expanded to creating an *evolutionary tree*, which explores the visually intuitive growth of a set of synoptic graphlets as a function of the number P of inspected packets per host. This study is useful in integrating host-level traffic characteristics of different P in an interpretable manner, and in quantifying the order of magnitude P beyond which further increase does not lead to substantially more relevant host profiling, i.e., how many packets P we need to profile hosts.

2.2 Preliminaries

Before turning to the method itself and the datasets used in the next sections, we recall the definition of graphlets in the context of Internet traffic and discuss related work. Then we propose an overview of our approach.

2.2.1 Graphlet

A *graphlet* is defined as a graph having the following characteristics in the context of network communication: (1) the graph is composed of several columns $(A_1, A_2, \ldots)$ of nodes, where each column represents one attribute of packets or flows, (2) a node (vertex) in a column is a unique instance of the attribute, and (3) there is an edge between two nodes of neighboring columns if at least one packet/flow has

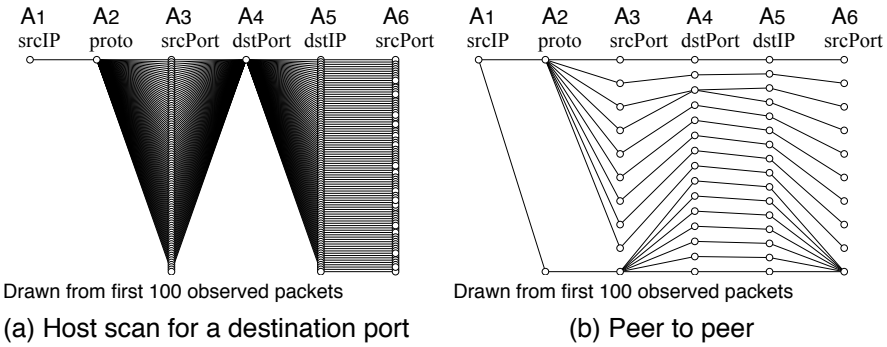

Figure 2.1: Examples of graphlets. Traffic from a single source host is represented as a graph connecting attributes such as proto, srcPort, dstPort, and dstIP.

the two corresponding attributes. Columns of a graphlet are usually related to flow attributes (5-tuple): proto (protocol number), srcIP (source IP address), dstIP (destination IP address), srcPort (source port number), and dstPort (destination port number), which are specified in the header field of every packet.

Figure 2.1 illustrates two manually annotated examples of graphlets drawn with $P = 100$ packets per source host. Figure 2.1(a) shows that the source host, which is represented as the single node in srcIP column, sends packets to a specific destination port of many destination hosts (almost one packet per flow); This suggests that the source host is a malicious scanner aiming to find hosts running a vulnerable application corresponding to the port. Figure 2.1(b) displays a host communicating with several hosts without any specific source/destination port, and hence this host is a peer-to-peer user (not server or client). As shown in these examples, a strong merit of graphlets is the visual interpretability of host characteristics as compared to examining a large number of raw packet traces or directly handling a set of numerical statistics.

We draw a graphlet from host-level traffic. Here, host-level traffic is defined as the sequence of packets sent from the host; Headers in those packets contain source IP addresses equivalent to the host's address. Note that this measurement does not necessarily capture initiation of communication (e.g., TCP hand shake). Each graphlet is drawn from a certain number of observed packets P sent from each host.

The graphlet we use is composed of six columns $A_1, \ldots, A_6$, which represent srcIP-proto-srcPort-dstPort-dstIP-srcPort[2]. The order of columns is different from the original definition [1]. We consider that srcIP-srcPort-dstPort-dstIP should be more comprehensive, because it clarifies the activity of computer processes inside end-hosts (IP-Port pairs) and network-wide inter-process communication among hosts (srcPort-dstPort pairs). We place srcPort at the right side again to cap-

[2]We define 'pseudo' source and destination ports for ICMP to be $srcPort = dstPort = icmp\_code$ in order to consistently draw graphlets.

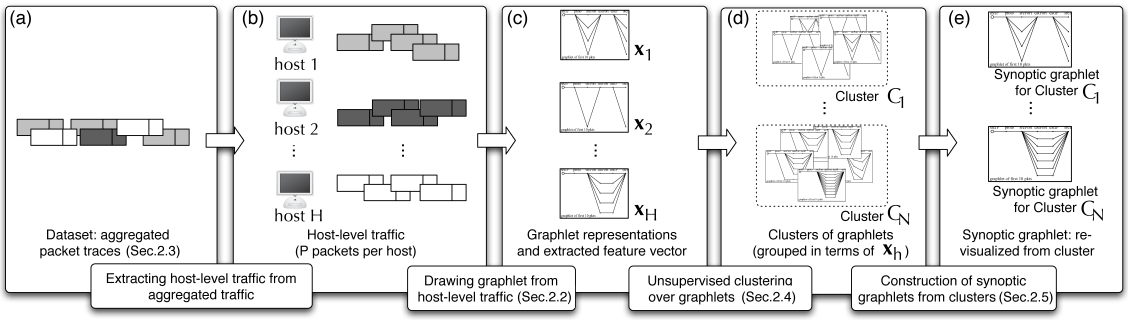

Figure 2.2: Overview of our approach.

ture the relation between dstIP and srcPort (inspired by [14]). Since we draw one graphlet per source host, there is only one point in the left column (srcIP).

2.2.2 Related work and open issues

Here the standpoints of the graphlet-based works and of this work are presented in the context of network traffic classification conducted over the course of a decade.

Many statistics-based methods for traffic analyses have been proposed to classify flows and host characteristics by means of supervised and unsupervised methods. These studies have made use of various supervised machine learning methods such as nearest neighbors [16, 17, 18], Bayesian statistics-based techniques [19, 16, 20, 18], decision tree [16, 20, 18], Support Vector Machine (SVM) [16, 18], or even natural language processing on Google search results [9]. The others have leveraged unsupervised ones including K-means clustering [7, 21, 22], or hierarchical clustering [7]. Both the approaches have been applied to traffic features from various aspects – packet sizes only [21, 17], combinations of packet sizes, flow sizes, inter-arrival times, flow durations etc. [19, 22, 20, 18], and/or entropy regarding the number of related hosts/ports [7, 10]. Those statistics-based methods are capable of overcoming packet encryption, encapsulation, use of dynamic ports, or dataset without payload, which are limitations on conventional approaches relying on payload inspection [2, 3, 4] and port-based rules [5].

Several recent studies particularly focused on large-scale host-to-host connections [6, 8, 11, 12, 13, 23], the use of which promisingly enables to visualize how hosts communicate with one another and enables to find groups of hosts communicating with each other. These works leverage existing graph-based analytical capabilities such as feature extraction regarding complex networks [12], community mining techniques [13], or block identification in communication (adjacency) matrix [11, 23].

Different from those previous works, the approach described here focuses on graphlets – detailed aspects of host behaviors including the usage of protocols and source/destination ports. The use of graphlets has been motivated by their visual

interpretability (as shown before), and has been conducted in a few works. For example, Karagiannis *et al.* perform supervised classification of flows based on graphlet models pre-determined by human experts [1]. Other works characterize graphlet-based host behaviors in unsupervised manners as follows: Karagiannis *et al.* discuss in-degrees and out-degrees of nodes and average degrees of graphlets in [14], and focus on manual finding of typical graphlets as well as on time transition of those features; In [24], Dewaele *et al.* classify hosts, making use of various features (some of them inspired by graphlets) applied to an unsupervised clustering technique.

To overcome the various limitations of supervised/unsupervised approaches that were discussed previously, and in contrast to previous works, the present article aims at bridging the gap between the two analytical approaches on graphlets by proposing a new framework for graphlet manipulation.

2.2.3 Overview of our approach

The three contributions of this work are: (1) the automation of finding typical graphlets via unsupervised clustering in an interpretable manner, (2) a method to re-visualize graphlets from clustering results, and (3) an analysis on evolution of typical graphlet shapes while increasing the number of packets per graphlet, which is complementary to analyses on time-transition of graphlet features. Each contribution is an important step of our method. Steps (1) and (2) are depicted in Figure 2.2 and step (3) is represented in Figure 2.9. Our method is organized as follows.

As a preprocessing step, aggregated traffic traces are first computed. (Figure 2.2(a)). The traffic is measured in a backbone link and composed of packets sent from hundreds of thousands of hosts (Sec. 2.3). We identify per-host traffic (Figure 2.2(b)) according to the source IP addresses specified in the packets, and draw graphlets from the first P measured packets sent from each host (Figure 2.2(c)).

Step (1): An unsupervised clustering over graphlets is conducted to find typical graphlets (Sec. 2.4). A numerical feature vector $\mathbf{x_h}$, which represents shape-based characteristics of a graphlet, is extracted from the graphlet of P packets sent from host h. The set of feature vectors $\mathbf{x_1}, \ldots, \mathbf{x_H}$, representing a set of H hosts, is used for hierarchical clustering to produce clusters of hosts $C_1, \ldots, C_N$ (Figure 2.2(d)). Cluster C_i consists of hosts that are similar in terms of their feature vector in the feature space. For each cluster, we obtain the components of a representative feature vector $\mathbf{x}$ which will be converted to graphlets in the next step.

Step (2): Resulting clusters are visualized to recover interpretability (Sec. 2.5). Since unsupervised clustering handles numerical features and thus loses visual information of graphlets, we re-visualize a representative graphlet associated with each cluster (Figure 2.2(e)). The reproduced graphlet, called synoptic graphlet, is derived from the feature vector $\mathbf{x}$ of the centroid of a cluster. We develop an original method to re-visualize synoptic graphlets in a deterministic manner, since conventional probabilistic ways of graph rewiring are not suitable for highly-structured

graphlets.

Step (3): Additionally, the evolutionary nature of synoptic graphlets is studied (Sec. 2.6). The key observation is that our knowledge of hosts may evolve as P increases from 1 to larger numbers. To study the evolution of the associated synoptic graphlets, we build an evolutionary tree of synoptic graphlets that evolve from the single-line graphlet (the only existing shape for $P = 1$) to the diversity of synoptic graphlets. This evolutionary tree is obtained by combining the clustering results of increasing P (see Figure 2.9). It provides intuitive understanding of the divergences and convergences in the growth of host characteristics as P increases.

2.3 Datasets

This section describes first the two datasets used for the validation of the proposed method, and, second, discusses how combining three different and classical traffic classifiers produces surrogates for real traffic ground-truth.

2.3.1 Traffic traces

We analyze traffic traces stored in the MAWI repository [25, 26] and traces measured at Keio University (used in [16] as Keio-I and Keio-II). MAWI traffic [25, 26] is measured on a transpacific IPv4 link between the U.S. and Japan for 15 minutes everyday. The public repository removes packet payloads, while the private repository retains payloads, up to the first 96 bytes. Results are reported here based on 12 MAWI traces collected once a month (on the 14th) in 2008. Keio traces used here are those presented in [16] and measured for 30 minutes, for two different days in 2006, on a bi-directional edge link in a campus of Keio University. Packet payloads up to 96 bytes were also preserved. We first removed the packets related to protocols other than TCP, UDP, and ICMP.

In the results reported below, we use the source hosts[3] sending at least 1000 packets for MAWI trace (respectively, 100 packets for Keio trace). This choice balances the trade-off between (a) having a lower reliability when hosts do not exchange enough packets, and (b) not keeping enough hosts when the required number of packet is too high. It has been checked that this arbitrary choice is not crucial; Results (e.g., the evolution of the number of clusters) similar to those obtained with $500 < Q < 1000$ were drawn with $200 < Q < 500$, or with $100 < Q < 200$ for MAWI traces, where Q denotes the number of packets sent by a host (observed in a trace). We will quantitatively evaluate the differences of results regarding the choice of Q in the future.

[3]It should also be meaningful to analyze destination hosts; With this analysis, for instance, we will be able to capture hosts receiving lots of attack packets. As a first step, we selected to analyze source hosts because of the easier interpretation of results; Packets sent from a host can be well explained by the application of the host, compared to packets received by a host.

Each of the 12 MAWI traces contains about 1,700 analyzed hosts, yielding approximately a number of analyzed hosts $H = 20,000$ in total for the 12 traces, and the 2 Keio traces contain about $H = 10,000$ hosts in total (H is the number of analyzed hosts). Those analyzed hosts for MAWI data account for 1.1% (19K out of 1.7M) regarding the number of hosts, 86% regarding the number of packets (207M out of 239M), and 93% regarding the number of bytes (1.43T out of 1.52T).

2.3.2 Pseudo ground-truth generators

Traffic analysis methods generally have to be evaluated with dataset annotated from ground-truth. A crucial issue raised in the recent literature, however, lies in designing a procedure to obtain ground-truth on actual traffic traces. Most of researches indeed have regarded ground-truth as the labels put by a single payload-based packet classifier. However a lot of packets are labeled as unknown by payload classifiers (as exhibited in this paper). Also, payload-based methods do not necessarily produce correct outputs. To improve the ground-truth coverage and accuracy, we carefully create three sets of pseudo ground-truth from different methods detailed here.

(a) Reverse BLINC. BLINC was originally proposed in [1] and extended to Reverse BLINC in [16], which is now state-of-the-art. BLINC profiles a pair of a source address and a port, and once the pair is matched with one of the heuristic rules based on the graphlet models, all pairs connected to that pair are classified. We used the default setting of Reverse BLINC as in [16]. BLINC's classification framework is WWW, CHAT, DNS, FTP, MAIL, P2P, SCAN, and UNKN (unknown). Since this classifier reports classification results as flow records, we need to convert them into a host-level database. For each source host, we collect a set of flows generated from the host and select the category (except for UNKN) that is the most frequent among the flows. For example, if ten flows from a host are classified into three DNS, one WWW, and six UNKN, then the type of the host is identified as DNS.

(b) Port-based classifier. We use another classifier, which was originally developed in [27] and also used in [28, 29]. This tool inspects a set of packets sent from a host, considering port numbers, TCP flags, and the number of higher/lower source/destination ports and destination addresses. The classification categories are WWWS (web server), WWWC (web client), SCAN, FLOOD (flooding attacker), DNS, MAIL, OTHERS, and UNKN [29]. This tool reports host-level classification results by itself.

(c) Payload classifier. We also use the payload-based classifier developed in [16] [4]. This classifier inspects the payload string of each packet by comparing it with its signature database. The classification categories we select are WWW, DNS, MAIL, FTP, SSH, P2P, STREAM, CHAT, FAILED (when the packets have

[4]In our preliminary experiment, we examined l7-filter [3] and found that the tool generated rather unreliable outputs because of loose payload signatures that are represented as regular expressions with a few bytes. Also, we found that OpenDPI [4] produced mostly unknown reports because it uses strict rules.

no payload), OTHERS (minor flows such as games, nntp, smb, and snmp), and UNKN. Since this tool also generates outputs in the form of flow tables, we merge them into host-level reports by the same means used to aggregate outputs from Reverse BLINC.

The hosts annotations given by the three classifiers of different perspectives are used to evaluate the unsupervised analysis on graphlets that is presented in the next section.

2.4 Unsupervised graphlet analysis

We detail the first step of the method, which is an unsupervised classifier for typical behaviors of hosts that does not rely on predefined models. However, it will still allow us afterwards to provide visual interpretation of the behaviors found.

2.4.1 Methodology for unsupervised graphlet analysis

2.4.1.1 Extracting shape-based features from graphlets

We first extract numerical feature values from graphlets, because visual graphlets cannot be used directly as input to conventional statistical methods (except for image processing). We choose afterwards several types of features related to shapes. We note $\mathbf{x_h}$ the feature vector for the graphlet of host h.

Notations on graphlets. We denote the six attributes (srcIP-...-srcPort) as column $A_1, \ldots, A_6$. In column A_i, the total number of nodes is n_i, and nodes are $v_{1,i}, \ldots, v_{n_i,i}$. We define $i:j$ as the direction from A_i to A_j, which is used to define the in-degree and out-degree of nodes in column A_i ($j = i+1$ or $i-1$). The in-degree of node $v_{k,i}$ is defined on direction $i : i-1$ as $d_{k,i:i-1}$, namely, $d_{k,i:i-1}$ is the number of nodes in A_{i-1} that are connected to node $v_{k,i}$ in A_i. The out-degree is similarly defined on direction $i : i+1$ as $d_{k,i:i+1}$. As a consequence, node $v_{k,i}$ is characterized by the pair of the in-degree and out-degree $(d_{k,i:i-1}, d_{k,i:i+1})$. We define the array of in/out degrees for direction $i : j$ as $D_{i:j} = (d_{1,i:j}, \ldots, d_{n_i,i:j})$ where n_i is the number of nodes in column A_i. $D_{i:j}$ gives the empirical distribution measured from an observed graphlet. Table 2.1 summarizes these notations.

Feature extraction. The proposed features are based on five types of shape-related information, described formally as follows and visually in Figure 2.3 (the relevance of the features is discussed later).

(1) n_i is the number of nodes in column A_i. Note that it is equal to the size of arrays $D_{i:i+1}$ and $D_{i:i-1}$. (6 columns)

(2) $o_{i:j} = \sum_{d_{k,i:j} \in D_{i:j}} I(d_{k,i:j} = 1)$, where $I(\cdot)$ is the indicator function, is the number of nodes that have degree 1 in direction $i : j$ (with $j = i \pm 1$). (10 directions)

Table 2.1: Notations for graphlet description. An attribute has two different degree distributions based on direction (e.g., A_2 is separated into $2:1$ and $2:3$). See Sec. 2.2.1 for details.

A_i	i-th column (or attribute) of graphlets (from left to right)
$v_{k,i}$	Node (vertex) in A_i
$i:j$	Direction from A_i to A_j ($j = i \pm 1$)
$d_{k,i:j}$	In/out-degree of node $v_{k,i}$: in-degree for $i:i-1$ (left half of $v_{k,i}$) and out-degree for $i:i+1$ (right half of $v_{k,i}$)
$D_{i:j}$	Empirical distribution of in/out-degrees in A_i

Table 2.2: Notations for graphlet clustering.

$\mathbf{x_h}$	Host h's graphlet feature vector, composed of the five degree-based features (Figure 2.3)
Dim	Dimension of $\mathbf{x_h}$ (44-dimensional for 6 columns)
H	Number of hosts analyzed
P	Number of packets per graphlet
C_i	Cluster of label i obtained
N	Total number of clusters obtained
θ	Distance-based threshold for clustering

(3) $\mu_{i:j} = \frac{1}{n_i} \sum_{d_{k,i:j} \in D_{i:j}} d_{k,i:j}$ is the average degree of direction $i:j$. (10 directions)

(4) $\alpha_{i:j} = \max_{d_{k,i:j} \in D_{i:j}} \{d_{k,i:j}\}$ is the maximum degree of direction $i:j$. (10 directions)

(5) $\beta_{i:i+1} = d_{k,i:i-1}$, where $k = \arg\max_l \{d_{l,i:i+1}\}$ is, for the node having maximum degree in $i:i+1$ (i.e., Feature 4), its degree in the backward direction $i:i-1$. If more than one node has the maximum degree for Feature 4, the pair with the highest degree is selected from among the candidates. A similar definition holds for the reverse direction $\beta_{i:i-1}$. (8 directions, since the edge columns have degree for only one direction)

As a result, from the graphlet for host h, we obtain a feature vector $\mathbf{x_h} = (x_{h,1}, \ldots, x_{h,44})$ $= (n_1, \ldots, n_6,\ o_{1:2}, \ldots, o_{6:5},\ \mu_{1:2}, \ldots, \mu_{6:5},\ \alpha_{1:2}, \ldots, \alpha_{6:5},\ \beta_{2:1}, \ldots, \beta_{5:6})$ of dimension of $Dim = 44$ $(= 6+10+10+10+8)$. We examine packet traces or flow lists (input) to compute these features (output). The index $i:j$ is omitted when not needed.

Examples. Figure 2.3 shows an example of features. For direction $2:3$, there are four nodes ($n_2 = 4$) and three nodes of one-degree ($o_{2:3} = 3$), and the average degree is 1.5 ($\mu_{2:3} = 1.5$). The second bottom node has the highest degree of three ($\alpha_{2:3} = 3$) and the degree of the node for the other direction is one ($\beta_{2:3} = 1$).

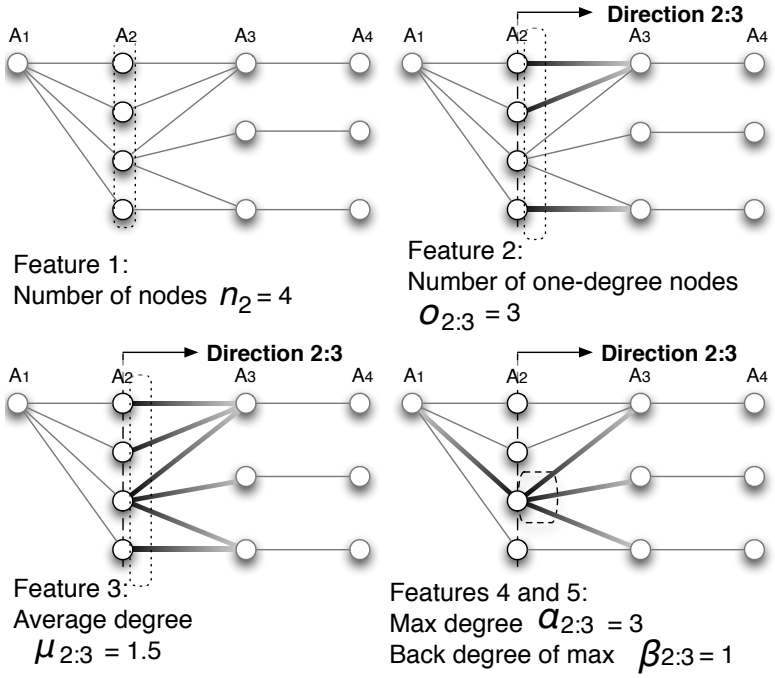

Figure 2.3: Shape-based features for a graphlet (i.e., behavior of a host).

Practical meanings. Even though these features are selected from the viewpoint of graphlet re-visualization (Sec. 2.5), a few of them can also be interpreted as traffic characteristics in a practical sense. n_i is the number of unique instances of the flow attribute (e.g., the number of destination addresses). $\mu_{i:j}$ and $\alpha_{i:j}$ are respectively the average and maximum number of unique flows of an instance of the attribute among all the instances.

Relevance of features. The selection of the five types of features is empirically motivated by two objectives: (*i*) the expected ability to obtain relevant clustering results because a few of the features are already well-known and well-studied [24] and (*ii*) the ability to re-visualize graphlets from the resulting clusters as explained in Sec. 2.5. Also, the relative importance of the five types of features is evaluated by a feature selection method in Sec. 2.4.2.4. Macroscopic degree-related features such as betweenness, the assortativity coefficient, or eigenvalues, are not used because graphlets are microscopic and highly structured. We only use graph-based features to evaluate the interpretability of graphlet clustering results, although there are many other well-studied features such as TCP flag, packet size, and flow size. Such features and ours are not exclusive but complementary; Using both types would enhance host profiling schemes.

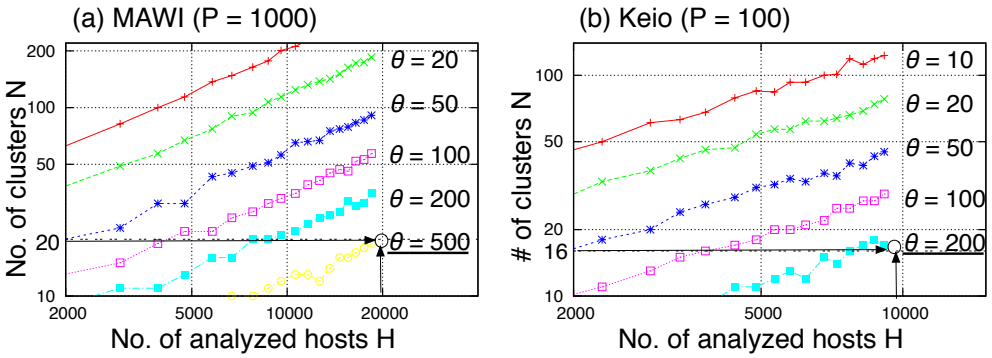

Figure 2.4: Clustering threshold θ characterized by the dependency on the number of analyzed hosts H and the number of resulting clusters N.

2.4.1.2 Applying graphlet features to unsupervised clustering

Here, we establish a method to find typical host behaviors in terms of graphlet shapes. At a high-level view, a set of hosts $\mathbf{x_1}, \ldots, \mathbf{x_H}$ are grouped into clusters $C_1, \ldots, C_N$ (clusters are disjoint sets of the hosts). Table 2.2 lists the notations used for the graphlet clustering.

Feature normalization. Each feature value $x_{h,i}$ from feature vector $\mathbf{x_h}$ is mapped onto a log space as $\log_{10}(x_{h,i}+1)$. For the features related to the ID of the transport protocol, the possible ranges of the values are adjusted to the other features (i.e., addresses and ports) as follows: $\log_{10}(P\frac{x_{h,i}}{\min(3,P)}+1)$, where P is the number of analyzed packets to be drawn as a graphlet, and the value 3 stems from the number of analyzed protocols (TCP, UDP, and ICMP). Hence, this type of feature is distributed into $[0, \log_{10}(P+1)]$ as well as the other features for any P. This normalization onto the log space is motivated by our empirical observation that graphlet shapes can be logarithmically well characterized; For example, by inspecting graphlet shapes with changing P, we observed that difference in graphlet shapes between P = 10 and P = 20 was intuitively similarly significant to P = 100 and P =200 (rather than P =100 and P =110).

Unsupervised clustering. Unsupervised clustering finds groups of hosts that are similar in terms of feature values by analyzing the H hosts $\mathbf{x_1}, \ldots, \mathbf{x_H}$. The hierarchical clustering [7] with Ward's method is used, as it is known to outperform other methods (e.g., single-linkage method). The similarity between a pair of clusters (C_i, C_j) is defined as a merging cost: $\Gamma(C_i, C_j) = E(C_i \cup C_j) - E(C_i) - E(C_j)$, with $E(C_i) = \sum_{h \in C_i} (\gamma(\mathbf{x_h}, \mathbf{c_i}))^2$ the intra cluster variance in Cluster C_i, the Euclidean distance $\gamma(\mathbf{x}, \mathbf{y})$ between vectors $\mathbf{x}$ and $\mathbf{y}$, and the average feature vector $\mathbf{c_i}$ of all hosts in C_i. The distance-based threshold θ is used to separate clusters in this feature space. The clustering produces a set of N clusters $C_1, \ldots, C_N$, depending only on θ (each host is included in a single cluster only). The selection of θ is discussed in

Sec. 2.4.2.

Motivation for distance-based threshold instead of number-based one. The distance-based threshold θ is preferable compared to cluster-number-based thresholds (such as the one for the K-means technique). This is because a consistent value of θ can be used for any P, which mitigates the burden of parameter tuning in analyses with several Ps as performed in Sec. 2.6. Number-based thresholds would have to be appropriately tuned through trial-and-error independently for each P, as the number of typical clusters for each P cannot be known. The consistent use of a single threshold over different Ps is empirically enabled by the normalization of the feature spaces as $[0, \log_{10} P]$, because distance between two clusters of typical behaviors will remain mostly the same for different Ps. Instead, conventional normalization into $[0, 1]$ would induce clusters with different behaviors at larger P to be located closer, requiring θ to be decreased.

Computational load. We used *hcluster* methods in the *amap* R-library. Approximately 1.5 GB memory was required for about $H = 20,000$ instances of $Dim(\mathbf{x}) = 44$ dimensional vectors. It took around 2.4 minutes with a 2.8 GHz Intel Core 2 Duo CPU with 4GB memory. By performing the clustering with changing H, we empirically confirmed that time and space complexities were both $O(H^2)$.

Table 2.3: Resulting clusters (MAWI with $P = 1000$) compared with three classifiers: Reverse BLINC (R-BLINC), port-based classifier (Port), and payload-based classifier (Payload). $N = 20$ clusters are obtained from the analyzed $H = 20,000$ hosts with the selected threshold $\theta = 500$.

			C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}	C_{18}	C_{19}	C_{20}
			4594	1252	986	1283	1526	1427	1134	274	715	451	297	152	950	961	613	652	964	690	221	305
R-BLINC	WEB	10199	1612	672	348	991	825	1361	1031	9	249	5		13	885	233	199	539	681	527	7	12
	DNS	1131	12	54	35	6	50	17	25	208	4	62	56	49	1	216	156	1	38	46	92	3
	MAIL	721	17	21	8	21	25	34	39		7	2		8	16	189	183	17	44	81		9
	P2P	1660	572	14	18	22	222	3	14	33	11	17	3	10	5	285	52	14	107	21	61	176
	SCAN	191										191										
	FTP	253	33				95				5				1	8		17	1			93
	CHAT	24				1													16		7	
	UNKN	5268	2348	491	577	242	309	12	25	24	439	174	238	72	42	30	23	64	77	15	54	12
Port	WWWS	6538	1553	670	710	1178	101	1	6			1	2		892	154	9		709	538	14	
	WWWC	5457	1337			4	722	1351	987	9	255	1	1	19		35	202	532				2
	DNS	807	8	51	33	2	5	5	10	180	3	53	43	32	1	114	74		40	50	103	
	MAIL	632	27	15	5	14	24	33	39		7			11	16	151	162	12	27	84		5
	P2P	361	74	5	3	6	16	2	4			1				160	24		51	2	12	1
	SSH	645	18	466	2	3	13	1	1		2			17	14	86	5	4	3			10
	SCAN	620				2	3	1		1	6	389	19	64		38	30			1	64	2
	FLOOD	709	66	1	111	3	3	5	2	66	173	4	224		4	6	15					26
	PROXY	113	1				5	3			85				2	2	1	12				2
	FTP	129	49	6	3		25		4		4			1	2	7		5	1			22
	OTHER	121	46	4	12	3	19	3	10		3	1			1	4	2				13	
	UNKN	3315	1415	34	107	68	590	22	71	18	177	1	8	8	18	204	89	87	133	15	15	235
Payload	WEB	11392	2620	627	671	1143	810	1316	1003	4	225		1	11	858	177	183	519	694	522	6	2
	MAIL	648	29	15	6	24	24	33	42		8			2	15	169	144	12	38	83		4
	DNS	1171	14	50	33	6	53	14	25	241	4	56	54	52	1	201	169	3	40	50	104	1
	P2P	430	309	2	20	7	68								1	16	4	2				1
	SSH	1023	30	505	12	39	50	29	28		33			16	62	97	10	37	54	20		1
	FAILED	504	128	5	11	7	54	16	7	9	18	21		66	1	29	40	7	2	5	69	9
	FTP	300	28		1	1	124				6				1	12	1	18	1			107
	CHAT	152	3	12	1		3	2			2					18			110			1
	STREAM	107	41		18	3	43				1					1						
	OTHER	106	29	32	8	2	11		1		4			1	3	5	8			1		1
	UNKN	3614	1363	4	205	51	286	17	28	20	414	374	242	4	8	236	54	54	25	9	42	178
#packets to stop separation			500	1000	1000	1000	1000	1000	1000	50	1000	20	50	1000	1000	1000	1000	1000	1000	1000	500	1000

2.4.2 Results: finding typical patterns of host behaviors

2.4.2.1 Threshold selection

The distance-based threshold θ eventually determines the number of extracted clusters N according to a conventional trade-off: a too high θ misses a number of typical host behaviors, while a too low θ produces redundant clusters (i.e., different clusters having similar compositions). By changing the value of θ, we inspected the list of synoptic graphlets (representative graphlets for resulting clusters – details are defined in Sec. 2.5) to identify whether there are redundant clusters (having same shape of synoptic graphlets) and new types of clusters (which cannot be found by large θ). We experimentally found that thresholds that balance this trade-off well are $\theta = 500$ with the MAWI traces (about $H = 20,000$ hosts) for $P = 1000$, producing approximately $N = 20$ clusters, and $\theta = 250$ with the Keio traces (about $H = 10,000$ hosts) for $P = 100$, resulting in $N = 16$ clusters. This trade-off has been manually inspected, because it is quite difficult to computationally identify redundancy of clusters in terms of the shapes of graphlets, which are one of our major focus and are enumerated in Sec. 2.5.

Figure 2.4 addresses the characteristics of θ by showing its relationship to the number of analyzed hosts H and the number of clusters N obtained from (a) MAWI (for $P = 1000$) and (b) Keio (for $P = 100$). Each set of analyzed hosts was selected from a random sample of the total number of original hosts by changing the sampling rate. This figure suggests referential values of θ for each dataset to obtain a certain number of clusters that balances the trade-off well for any H.

We note that this value of θ can be consistently used for other P, and this is the reason why we do not directly use the number-based threshold. Since θ is based on the distance in the feature space, we can compare the clustering outputs from various P with a single consistent criteria. For example, smaller P might lead to fewer numbers of clustering with regard to the feature space. We confirmed that the value of θ is consistently appropriate for other P as shown in Sec. 2.6.

2.4.2.2 Typical patterns of host behaviors

Table 2.3 shows the clustering result, with $H = 20,000$ hosts at $P = 1000$ of MAWI data, obtained from a comparison between the graphlet clustering and the three classifiers, i.e., Reverse BLINC (R-BLINC), port-based classifier (Port), and payload-based classifier (Payload). This table displays the total number of hosts in each category and each cell shows the number of hosts in the intersection between two classes of two classifiers. The first row of the column headings is auto-generated labels. The second row shows graphlets re-visualized from clusters (Sec. 2.5), and the bottom row is discussed in Sec. 2.6.

The sparseness of Table 2.3 indicates that each cluster mostly corresponds to a type of host behavior. For instance, C_6 (containing 1427 hosts) is characterized by one typical category because most of the hosts are labeled as a category of each

Table 2.4: Graphlet features evaluated by FCBF.

MAWI		Keio	
feature	$SU_{i,c}$	feature	$SU_{i,c}$
$o_{i:j}$ of srcPort → dstPort	0.51	$o_{i:j}$ of srcPort → dstPort	0.57
$o_{i:j}$ of dstPort → srcPort	0.48	$o_{i:j}$ of dstPort → srcPort	0.50
$o_{i:j}$ of dstIP → dstPort	0.39	$o_{i:j}$ of dstIP → srcPort	0.41
$o_{i:j}$ of dstIP → srcPort	0.39	$o_{i:j}$ of dstIP → dstPort	0.40
$\mu_{i:j}$ of dstIP → srcPort	0.36	n_i of proto	0.06
$\beta_{i:j}$ of dstIP → srcPort	0.34		
$\mu_{i:j}$ of dstPort → dstIP	0.31		
n_i of proto	0.10		

classifier: 1361 hosts as WEB by R-BLINC, 1351 hosts as WWWC by Port, and 1316 hosts as WEB by Payload. In addition, the overall similarity among the results from the three classifiers cross-validates their effectiveness.

Clusters can show the typical host behaviors hidden in a single category. WEB of R-BLINC, for example, is separated into a few clusters, reflecting the different behaviors of web hosts such as server (C_2, C_3, C_4, C_{13}, C_{17}), client (C_5, C_6, C_7, C_{16}), and P2P user (C_{14}) as suggested by WWWS, WWWC, and P2P of Port, respectively. Moreover, the WWWC (web client) category of Port is clustered into a few groups, and a plausible reason for this is that there are a few typical behaviors of web clients based on the usage of web such as large-file transfer, web browsing, and ajax-based activity. Also, the MAIL category of Port shows the behaviors of only server (C_{18}), only client (C_5, C_6, C_7, C_9) or both server and client (C_{14}). This observation can also be validated by the other categories in the same cluster (e.g., P2P of Port in C_{14}).

In particular, the ability to cluster unknown data is an advantage of the unsupervised approaches. Our clustering method provides key information to profile hosts that R-BLINC classifies as UNKN[5] by separating these hosts into different categories. For example, C_3 separates 577 UNKNs of R-BLINC from the totally 5268 UNKNs of the classifier, and we can speculate that most of the 577 UNKNs are web servers as most hosts in the cluster are classified as web servers (e.g., C_3 mainly consists of 348 WEB hosts labeled by R-BLINC other than the 577 UNKN hosts). The same is true for other UNKNs of the three classifiers. Thus, the results of the classifiers and of our approach complement each other.

[5]We provide an example of UNKN hosts labeled by R-BLINC by examining Cluster C_4, which consists of 1283 hosts. This cluster consists of mainly WEB hosts as suggested by the three classifiers and the shape of synoptic graphlet. As mentioned above, this synoptic graphlet can be mapped with BLINC's original WEB graphlet, but this cluster contains 242 UNKN hosts classified by R-BLINC. A plausible reason of the UNKN hosts is as follows. As one of the classification rules, R-BLINC considers WEB hosts to follow "#dstPort - #dstIP $> a$", where a is one of the 28 thresholds and its value with our default setting is $a = 4$. The average and standard deviation of "#dstPort - #dstIP" are 8.12 ± 4.82 for WEB hosts of R-BLINC inside C_4 (991 hosts), and are 3.61 ± 2.39 for UNKN hosts of R-BLINC inside C_4 (242 hosts), which does not follow the above-mentioned R-BLINC's classification rule for WEB.

The effectiveness of a connection pattern-based approach can also be complementarily improved by port- and payload-based approaches. One notable example is C_1, which contains the most of UNKN hosts from R-BLINC. The port and payload classifiers both indicate that this cluster is mainly related to web server and client hosts. Actually, for the 2348 UNKN hosts in C_1, our additional inspection found that 1150 hosts are classified as web server or client by both the port- and payload-based classifiers; This suggests that such cross-validation would reduce the UNKN classification. Another example is that 1404 hosts out of the 1612 WEB hosts for R-BLINC in C_1 are identified as web server or client as well by both the port- and payload-based classifiers, which indicates those hosts can be considered as web-related ones with high 'plausibility.'

2.4.2.3 Inter-cluster distance

We examined the distribution of clusters in the feature space by using the inter-cluster distance metric: $dist(C_i, C_j)$ defined as $\frac{1}{Dim}||\mathbf{c_i} - \mathbf{c_j}||$, where c_i is the centroid vector for C_i. We define $mindist(C_i) = \min_j dist(Ci, Cj)$. The average and standard deviation of $mindist(C_i)$ is 6.63 ± 4.50, with minimum $mindist(C_1) = dist(C_1, C_3) = 0.56$ and maximum $mindist(C_{10}) = dist(C_{10}, C_{11}) = 26.7$ in the log space. This means that the clusters are not uniformly distributed. Our observation was that graphlets with low number of flows (e.g., C_1, C_3, C_5, C_4) have low *dist* between each other, i.e., they are densely distributed yet clustered due to the high number of hosts; Whereas, high *dist* derives from graphlets with high number of flows (e.g., $dist(C_6, C_7)$, $dist(C_{10}, C_{11})$) having similar shape but different typical number of flows.

2.4.2.4 Dominant features

Here we extend the discussion by evaluating which out of the $Dim = 44$ features significantly contributed to the $N = 20$ obtained clusters (Table 2.3). For this evaluation, we use Fast Correlation-Based Filter (FCBF) [30, 19, 16], a feature ranking and selection method. We note that FCBF is used only for evaluating the relative contribution of the features to the clustering results and is not used for other parts of this work.

FCBF selects the most effective and smallest set of features with respect to symmetric uncertainty (SU) $\in [0, 1]$, which measures a form of correlation between two random variables: $SU_{X,Y} = 2\frac{H(X)-H(X|Y)}{H(X)+H(Y)}$, where $H(\cdot)$ is the information-theoretical entropy and $H(\cdot|\cdot)$ is the conditional entropy. $SU_{i,c}$ is the correlation between feature i and clusters (SU against clusters), and $SU_{i,j}$ is that between features i and j (SU against features). A higher $SU_{i,c}$ means that feature i contributes to detecting one or more clusters, whereas a higher $SU_{i,j}$ indicates that joint use of features i and j is redundant. The method first removes irrelevant features (having low $SU_{i,c}$) and then excludes redundant features (having higher $SU_{i,j}$ than $SU_{i,c}$).

Table 2.4 lists the selected features showing their SU against clusters for MAWI and Keio data: $N = 20$ clusters for MAWI with $P = 1,000$, and $N = 16$ clusters for Keio with $P = 100$. The features selected by FCBF are mainly $o_{i:j}$ (the number of one-degree nodes), and this result suggests that this type of feature is more relevant and less redundant than the other features. Our interpretation is that $o_{i:j}$ represents well a part of the graphlet (i.e., the area between $i : j$ and $j : i$) in term of its shape (e.g., a square (parallel line(s) between columns), or a triangle (a knot on a column)) and of its number of lines (i.e., visual complexity) (e.g., one line, a few lines, or many lines). These are basic characteristics of the behavior of hosts, and the features $o_{i:j}$ represent such characteristics better than the other features used here. Figure 2.1 shows examples for $o_{i:j}$. Square shapes such as the area between A_5 and A_6 in Figure 2.1(b) occur when both the values of $o_{i:i+1}$ and $o_{i+1:i}$ are high. On the other hand, triangle shapes such as the area between A_4 and A_5 in the figure appear when one of $o_{i:i+1}$ and $o_{i+1:i}$ is quite low (e.g., zero or one). In particular, $o_{i:j}$ between srcPort and dstPort contributes significantly to the clustering (1st and 2nd ranks in Table 2.4). The relation between the ports represents the detailed behavior of inter-process communication, which is an important aspect of networking.

Even though other features also have discriminative power, such features are not part of the best set of features. For example, we observe that n for srcPort has $SU_{i,c} = 0.43$, and $\alpha_{i:j}$ of srcPort to dstPort has $SU_{i,c} = 0.41$ for MAWI data, indicating that these features are also useful. These features, however, were removed because of their high correlation with corresponding $o_{i:j}$ (e.g., a higher $o_{i:j}$ will be provided by a higher n_i). It means that they have similar but weaker effect on the clustering compared to $o_{i:j}$. In other words, $o_{i:j}$ is a good approximation of the shapes of graphlets. Even so, the other features are also necessary for inferring synoptic graphlets (see next section), and this is why we keep all the features.

2.5 Synoptic graphlet

According to the unsupervised procedure described in Sec. 2.4, graphlets associated with hosts are clustered with respect to their feature vectors. Now, as an inverse problem aiming at associating each cluster with a representative graphlet, as sketched in Figure 2.5, we propose a method to construct a *synoptic graphlet* from the feature vector representing a cluster.

2.5.1 Synoptic graphlet: construction

An original mapping from a feature vector into a set of bipartite graphs that constitute a graphlet is detailed here and illustrated in Figure 2.5. This mapping is applied to the feature vector of the cluster centroid. We will address the motivation to use synoptic graphlets instead of centroid-nearest graphlets at the end of this subsection.

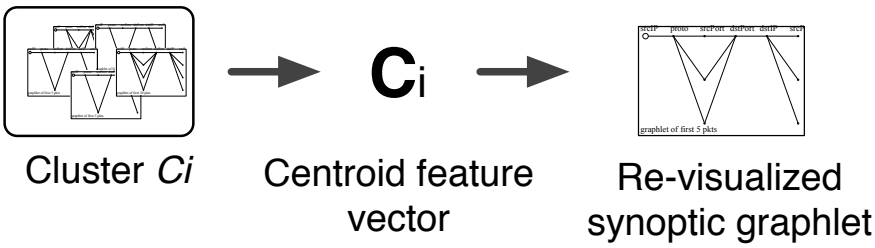

Figure 2.5: Synoptic graphlet. Graphlets obtained from hosts are clustered. In turn, each cluster is associated with a representative a posteriori synoptic graphlet. The second row of Table 2.3 displays the synoptic graphlets re-visualized from the actual clusters of hosts.

Median centroid. Recalling that the feature vector of host h was defined as $\mathbf{x_h} = (x_{h,1}, \ldots, x_{h,Dim})$, let us define $\mathbf{c_k} = (c_{k,1}, \ldots, c_{k,Dim})$ as the centroid features of Cluster C_k, where the $\frac{|C_k|}{2}$-th largest value of $x_{h,i}$ among $h \in C_k$ is selected as the median feature $c_{k,i}$ [6][7].

(1) Considering a graphlet as a set of bipartite graphs. To infer a graphlet from the centroid features of a cluster, we construct a graphlet as a set of bipartite graphs. A_1 and A_2 are a disjoint set of a bipartite graph, A_2 and A_3 are another, and so on. In other words, we break down the graphlet reproduction problem into (a) reproducing the degree distributions of each bipartite graph, (b) rewiring each bipartite graph based on the degree distributions, and (c) merging neighboring bipartite graphs.

(2) Reproducing degree distributions. From a feature vector, we build the degree distribution of direction $i : j$ ($j = i + 1$ or $i - 1$), denoted as $\hat{D}_{i:j} = (d_1, \ldots, d_n)$ where n is the total number of nodes as defined in Sec. 2.4.1.1 ("$i : j$" is omitted from $d_{k,n_{i:j}}$ and $n_{i:j}$ for brevity). We first consider the one-degree nodes as follows: $d_n = d_{n-1} = d_{n-o+1} = 1$. If all the nodes have degree of one (i.e., $n = o$), this procedure ends; Otherwise we rebuild the remaining part of the degree distribution. We define the number of remaining nodes ζ and the remaining degrees ξ as $\zeta = n - o$ and $\xi = \mu \times n - 1 \times o$. The degrees are estimated as follows: $d_1 = \alpha, d_2 = \alpha - \Delta, \ldots, d_\zeta = \alpha - (\zeta - 1) \times \Delta$, where $\Delta = \frac{2}{\zeta - 1}(\alpha - \frac{\xi}{\zeta})$, which satisfies $\xi = d_1 + \ldots + d_\zeta$. This process to distribute the remaining degrees to the remaining nodes is based on the usual appearances of graphlets (e.g., some 'knot' nodes, only

[6] As an example, for n_2, if a cluster contains 100 hosts, the 50th largest value in n_2 is chosen as the median (x_i and x_j ($i \neq j$) do not necessarily derive from the same host).

[7] We note that statistics other than the median could be chosen as a representative. We also tried to use average as representative, but average is not robust to outlier features, and more critically taking the averages lead to decimal values, which are difficult to deal with for graph rewiring. The *Dim*-dimensional median features are converted from a log scale into a linear scale by inverting the normalization function defined in Sec. 2.4.1.2.

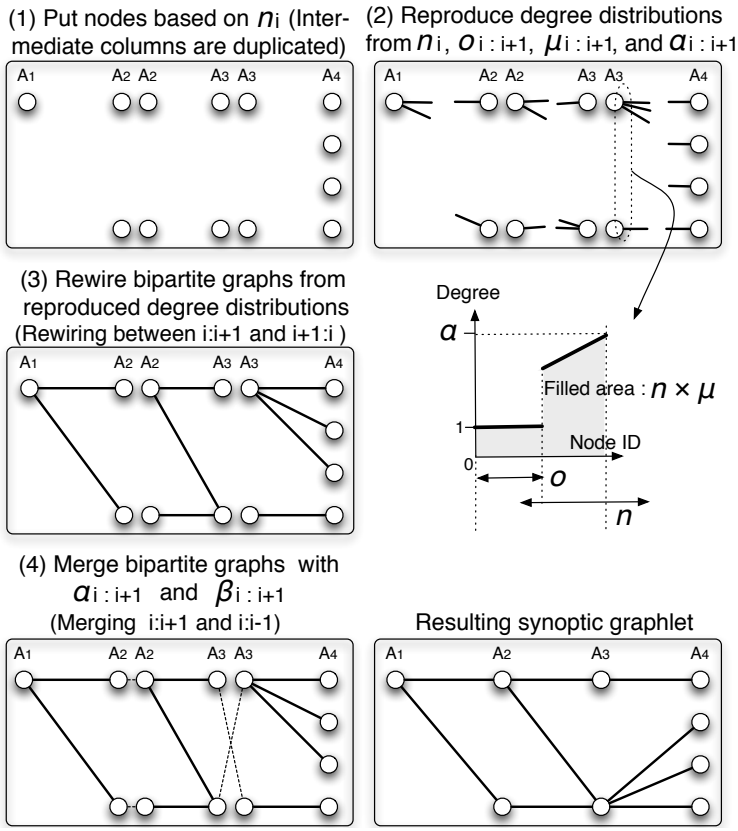

Figure 2.6: Procedure of re-visualizing synoptic graphlets. A synoptic graphlet of a cluster is reproduced from the graphlet features of the cluster centroid. Graphlet features are defined in Sec. 2.4.1.1 and Figure 2.3.

one, etc.).

(3) Rewiring bipartite graphs. A bipartite graph is generated from $\hat{D}_{i:i+1}$ and $\hat{D}_{i+1:i}$ computed above. Nodes of higher degrees of A_i are connected with those of lower degrees of A_{i+1}, which reflects an emprical traffic characteristics (one-to-many connection rather than two-to-many). An example of this characteristics is server-client behavior, where (a) a source port is connected with several destination hosts and also (b) a destination host is associated with a set of several destination ports, which are not related to other hosts. By defining $i : i+1$ as r (right) and $i+1 : i$ as l (left), we connect $v_{1,r}$ with $v_{n_l,l}, \ldots, v_{(n_l - d_{1,r}-1),l}$, and then connect $v_{2,r}$ with $v_{k,l}, \ldots, v_{(k-d_{2,r}-1),l}$, where k is the largest label of nodes that have degree remaining after the previous connections. We iterate this connection procedure until $v_{n_r,r}$ is dealt with and consequently obtain a bipartite graph.

(4) Merging bipartite graphs into a synoptic graphlet. A synoptic graphlet is then drawn by combining each pair of neighboring bipartite graphs. We additionally define the direction: $i : i+1$ as f (forward) and $i : i-1$ as b (back-

ward). The two directions have different degree distributions with the same number of nodes: $\hat{D}_f$ and $\hat{D}_b$, and a pair $(d_{k,f}, d_{l,b})$ is merged into a node $v_{m,i}$, where k, l, and m are determined as follows. We first compute the degree correlation between $\hat{D}_f$ and $\hat{D}_b$, which we define as $\gamma = (\alpha_f - \alpha_b) \times (\beta_f - \beta_b)$, with $\alpha_{i:j}$ and $\beta_{i:j}$ of the centroid features. If the correlation is positive ($\gamma \geq 0$), we combine the nodes in the same order of degree value: $v_{1,i} = (d_{1,f}, d_{1,b}), \ldots, v_{n,i} = (d_{n,f}, d_{n,b})$. Conversely, for $\gamma < 0$, the combination order is reversed: $v_{1,i} = (d_{1,f}, d_{n,b}), \ldots, v_{n,i} = (d_{n,f}, d_{1,b})$.

Synoptic versus centroid graphlets. Instead of synoptic graphlets, centroids may have been selected as cluster representative. For clusters with very large number of flows, both choices likely yield close representatives, however, centroids suffer from a number of disadvantages: (i) Centroid graphlets may show a very large variability (hence lacking robustness) for clusters with small number of flows, while synoptic graphlets are less dependent on the actual number of flow per host, because it is regenerated from all the representative features of a cluster; (ii) Centroid graphlets not necessarily result into the typical representative of the cluster. The centroid may occasionally correspond to a specific behavior, even when many of its Dim features are close to the median, to the contrary of synoptic graphlets that somehow make the visualization/interpretation step independent from the classification phase (in a semi-supervised spirit) [8]; Therefore synoptic graphlets should be more effective tools to represent what actually happens in the feature space and thus to profile and interpret host behaviors. More detailed comparisons between centroid and synoptic graphlets are beyond the scope of the present contribution and will be discussed elsewhere.

2.5.2 Synoptic graphlet: interpretation

The second row of the column headings in Table 2.3 shows the synoptic graphlets, re-visualized from the $N = 20$ clusters presented in Sec. 2.4.2 (larger versions are displayed in Figure 2.9).

Effectiveness of synoptic graphlets. One of the advantages of synoptic graphlets is the ability to construct an intuitive understanding of clustering results. The "complexity" of the shapes of synoptic graphlets meaningfully represents the intensity of flows. For example, a graphlet of many lines is derived from the use

[8] We briefly compared the synoptic graphlet and centroid-nearest graphlet for each cluster. Approximately 80% of clusters produced intuitively similar shapes of the two kinds of graphlets; This is plausibly due to the well-tuned threshold θ and enough number of hosts inside a cluster. Such correspondence between the two kinds implicitly validates the overall procedure of rewiring graphlets. We also found the differences in shapes of the two kinds. For example, in Cluster C_2, there were differences in the #nodes in the dstIP column between the corresponding synoptic graphlet and centroid-derived graphlet; Indeed, the collapse in the shape of the synoptic graphlet (Table 2.3) indicates that this graphlet does not represent per-host behavior well but rather represents an aggregated view. We manually inspected the composition of C_2 and found that this cluster contained two types of typical host behaviors. This graphlet suggests that it would be meaningful to further separate C_2 into different clusters.

of many flows, indicating that the corresponding host uses an application for many peers and/or many ports (e.g., DNS and MAIL are the categories of the many-lines graphlets such as C_{15}). In addition, the number of nodes for each column A_i is also meaningful. For instance, if A_3 (srcPort) has only a few nodes, then the corresponding host can be speculated to be a server (e.g., C_3 is mainly labeled as WWWS by Port).

BLINC models validity. Most of the synoptic graphlets in Table 2.3 correspond to most of the BLINC graphlets[9] (listed in [1]), and thus our result validates the intuitions behind the BLINC series. An exception, though, is pointed out by C_{11}; Most hosts are identified as UNKN by R-BLINC, whereas they are mainly identified as FLOOD by Port (probably because of a large amount of SYN packets and few targeted hosts). On the other hand, some clusters having similar shape of synoptic graphlets consist of similar breakdown such as C_{17} and C_{18}. As implied by the different number of lines in the shapes of synoptic graphlets for the two clusters (Table 2.3), this result indicates two typical number of flows of graphlets, which might not easily be found by applying untuned heuristic rules.

One-flow graphlets. C_1 represents synoptic graphlets composed of one flow (4594 in total – about 25% among the analyzed hosts), and the three classifiers unfortunately identify many of them as UNKN. This kind of isolated communication has been observed in prior studies [31, 12, 11] as well. Although one-flow graphlets are classified into various application categories as the three classifiers point out, the one-line shape itself reveals the important information that $P = 1000$ packets from a single host constitute only one flow. In other words, a one-flow graphlet possibly implies large file transfer, because we do not observe any control flows or the other flows. This plausible interpretation is supported by the finding that many of these hosts identified by the three classifiers are web or P2P users, which are occasionally used for host-to-host large-file transfer in some cases.

In summary, synoptic graphlets are effective for an intuitive and visual understanding of the clustering output, and the comparison result indicates the relevance of the overall idea of BLINC, while alleviating the difficulty of manually setting appropriate rules and parameters.

[9]For example, the synoptic graphlet of C_4 can be mapped with WEB graphlet (shown in Figure 5(d) of the original paper), because the two graphlets commonly represent one srcPort, and several dstPort, and a few dstIP nodes. The relevance of this mapping is supported by the fact that R-BLINC classifies most of the hosts in C_4 as related to WEB. For another example, the synoptic graphlet of C_8 can be related to DNS graphlet (shown in Figure 5(g) of the original), because the two graphlets commonly represent many srcPort, and one dstPort, and a few dstIP nodes. The original graphlet represents both client-side and server-side behavior in a single figure, yet the graphlet of C_8 can be mapped with client-side one. The relevance of this mapping is supported by the fact that R-BLINC classifies most of the hosts in C_8 as related to DNS.

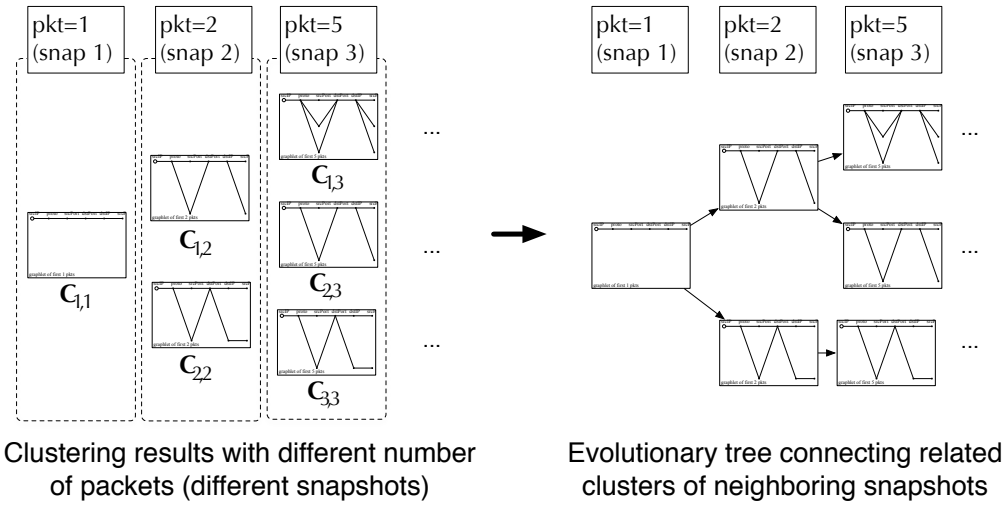

Figure 2.7: Creation of evolutionary tree.

2.6 Evolutionary nature of host-level traffic

Let us further discuss the effectiveness of the new method by introducing *evolutionary tree* of synoptic graphlets, which provides a way to understand the evolution of information about host behaviors when the number P of analyzed packets increases. To achieve this, we analyze the same set of $H = 20,000$ hosts by changing the value of P. This tree can also answer the question "how many packets P do we need to find all typical patterns?" and "how accurately hosts can be profiled with a given P?."

2.6.1 Evolutionary tree: creation

Snapshot. The next key question in the assessment of synoptic graphlets is raised by the choice of the number P of packets that need to be involved in graphlet construction to find all typical patterns and thus permit accurate host profiling. This is addressed via the concept of synoptic graphlet *evolutionary tree* that characterizes host behavior profiling evolution when P increases. For example, a single packet (thus a single flow) produces a single-line graphlet, whereas two packets may result either in a single line if they belong to the same flow or in two lines sharing nodes and edges if they share common attributes. Any graphlet may hence evolve from an identical single-line shape towards a complex pattern as P increases. An evolutionary tree is thus obtained from combining different *snapshots* s, i.e., graphlets obtained from different values of P_s [10]. For MAWI data,

[10]It should also be interesting to analyze this by increasing the number of flows per host (instead of increasing the number of packets P). However, we have to elaborate on the appropriate way to

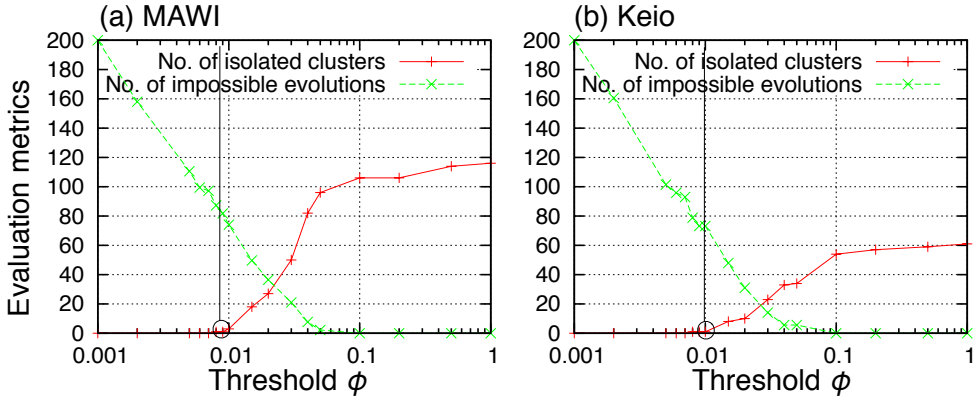

Figure 2.8: Characteristics of the threshold for evolutionary tree ϕ.

$P = 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000$ for snapshots $s = 1, \ldots, 10$; for Keio data, $P = 1, 2, 5, 10, 20, 50, 100$ for $s = 1, \ldots, 7$.

Tree creation. Let C_{s,N_s} denote the set of N_s clusters obtained at snapshot s (i.e., from P_s packets). For each s, C_{s,N_s} is obtained with a value of the sole threshold θ that remains constant and does not depend on P_s. Thus, θ serves as distance basis in the feature space, and hence does not determine a priori the number of clusters, which permits to compare clustering outputs obtained with different P. The evolutionary tree is created from a single criteria, relying on a threshold ϕ: if the number of hosts in $C_{s,i} \cap C_{s+1,j}$ is larger than $\phi \times H$ (H being the total number of analyzed hosts), the two clusters $C_{s,i}$ and $C_{s+1,j}$ are connected by an edge, which materializes that the typical behavior $C_{s,i}$ at snapshot s tends to evolve into $C_{s+1,j}$ at $s+1$. Finally, an evolutionary tree provides an intuitive overview of the behavioral growth of hosts (cf. e.g., Figure 2.9).

Threshold. Setting the threshold ϕ, which determines whether neighboring clusters are connected or not, results from the following trade-off: Too high ϕ may yield 'isolated' clusters, not connected to any other clusters on any neighboring snapshot; Too low ϕ may yield many 'impossible' evolutions in graphlet shapes. For example, for some synoptic graphlets, α might be reduced from s to $s+1$ because of the changes in the set of hosts within a cluster, despite the fact that this never occurs in the evolution of the graphlet of a single host. Therefore, the connection between Cluster i at snapshot s and Cluster j at $s+1$ is declared *impossible*, if either of parameters n, μ, and α is reduced. Figure 2.8 illustrates the trade-off, plotting the number of isolated clusters and that of impossible evolutions as a function of ϕ. Empirically, the threshold is set to $\phi = 0.0077$ (i.e., about 150 hosts) for MAWI data, and to $\phi = 0.0070$ (i.e., about 70 hosts) for Keio data, which maintain no isolated cluster and a low number of impossible evolutions.

---

deal with hosts with low number of flows (e.g., 25% of hosts have only one flow).

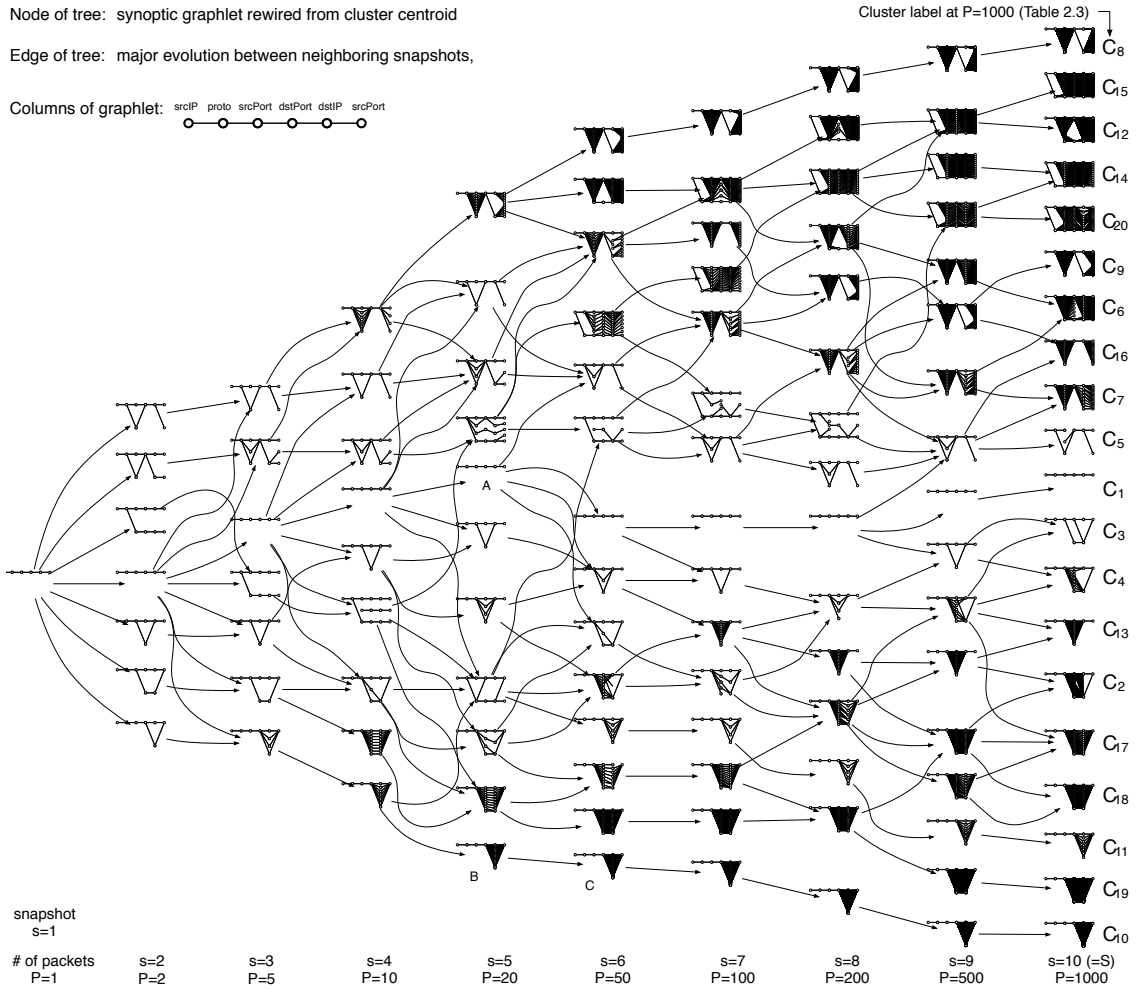

Figure 2.9: Evolutionary tree of synoptic graphlets as a function of P (or s).

2.6.2 Evolutionary tree: interpretation

2.6.2.1 Intuition from evolutionary tree – visual analysis

Global view. Figure 2.9 depicts the resulting evolutionary tree for MAWI data ($H = 20,000$, $\theta = 500$, cf. Sec. 2.4.2.1, $\phi = 0.77\%$, cf. Sec. 2.6.1). Synoptic graphlets at snapshot s are shown in the s-th column, and related synoptic graphlets (from successive snapshots) are linked with arrows. The synoptic graphlets at $s = 10$ correspond to the evaluations presented in Secs. 2.4.2 and 2.5.

Figure 2.9 thus provides an intuitive and comprehensive overview of the evolution of typical host behaviors, from $P = 1$ (origin of graphlets) to large P, permitting interpretation of graphlet changes with P. Interestingly, clusters do not only separate but also merge as P increases. This suggests that there exist different evolution

footprints, even when hosts are clustered into a same group at a given snapshot. Evolutionary trees thus enhance the profiling by providing richer information.

Early stages. For $P = 1$, by nature, there is only one-flow graphlets. For $P = 2$, although there are theoretically $2^4 = 16$ possible graphlets (combination of four attributes: proto, srcPort, dstPort, and dstIP), only 7 are actually observed. Although some graphlets are actually different from the seven synoptic graphlets and have different transitions, these are not typical, and hence do not appear in the figure. Such minor graphlets could be found by finer-grained clustering, with lower θ.

Late stages. The final forms of graphlets become apparent in the late stages. For example, one-flow graphlet A is destined to mostly remain one-flow, after $P = 20$, as indicated by the abrupt increase in predictability discussed in Sec. 2.6.2.2. Other examples are provided by synoptic graphlets B and C, prominent at $P = 20$ and 50, respectively. They are mainly related to scanning activities, which thus indicates that $P = 20$ is large enough to permit separation of scanners from other activities. As a whole, the total number of clusters at $P = 1000$ remains quasi-unchanged compared to that at $P = 100$. Thus, $P = 100$ can be considered as the reference number of packets required for accurately discovering typical host behaviors. Also, this result implies that $P = 100$ provides some longitudinal stationarity of aggregated view of host behaviors. The bottom row in Table 2.3 lists each stage at which each Cluster $C_{S,i}$ stops its evolution along the tree (i.e., becomes predictable).

Keio data case. Similar results were obtained for Keio data, but for the fact that one-flow graphlets continue to evolve at $P = 50$. We interpret that the stagnation of one-flow graphlets for MAWI could stem from the partial view of the traffic, measured at the backbone link, whereas Keio traffic is measured at an edge router.

2.6.2.2 Predictability in evolution – quantitative analysis

To complement the understanding of synoptic graphlet evolution, the evolution predictability of a given host in the tree is quantified. Let us define $P(C_{s_2,j}|C_{s_1,i}) = \frac{|C_{s_2,j} \cap C_{s_1,i}|}{|C_{s_1,i}|}$, which measures the probability that hosts in Cluster i at snapshot s_1 ($C_{s_1,i}$) evolves into Cluster j at s_2 ($C_{s_2,j}$). We define the predictability of Cluster $C_{s,i}$ as $Pred(C_{s,i}) = 1 + \frac{1}{\log_{10} N_S} \sum_{j=1}^{N_S} P(C_{S,j}|C_{s,i}) \times \log_{10} P(C_{S,j}|C_{s,i})$, where S is the final snapshot and N_S the corresponding number of clusters. $Pred(C_{s,i})$ is hence a normalized entropy that characterizes the dispersion of transition probabilities. Thus, if $C_{s,i}$ grows only to $C_{S,1}$ then $Pred(C_{s,i}) = 1$, whereas if $C_{s,i}$ can evolve into any future shapes with equal probability then $Pred(C_{s,i}) = 0$. Note that this predictability is computed considering all possible evolutions (i.e., $\phi = 0$).

Figure 2.10 displays the predictabilities of all clusters $Pred(C_{s,i})$ as a function of P (or s) for MAWI and Keio. Each dot stands for a synoptic graphlet (i.e., a cluster), for a given snapshot. The dashed line represents the transition in the average predictability and shows that the predictability is approximately linear with

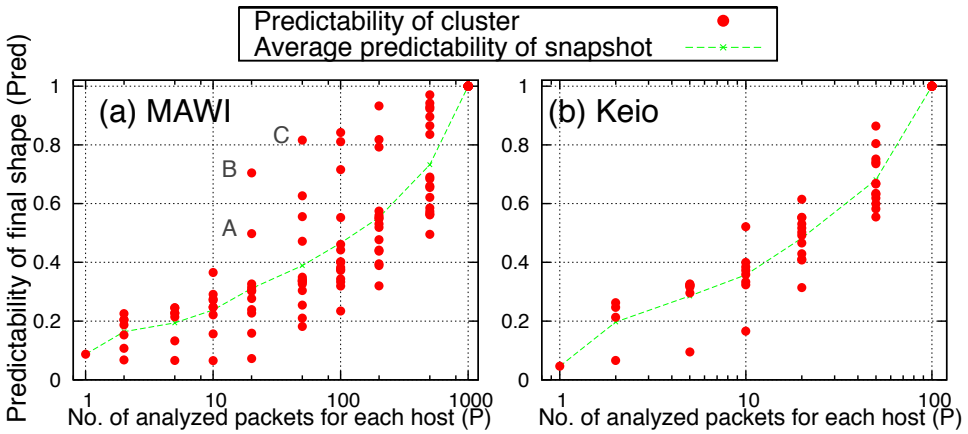

Figure 2.10: Predictability of evolution as a function of P.

$\log P$ (Pearson's correlation coefficient is 0.95). The predictability at $P = 1$ is almost 0, which suggests that the corresponding origin of a graphlet can evolve into any final graphlet. Conversely, this predictability becomes higher with higher P. In addition, predictabilities for some $C_{s,i}$ abruptly become higher than for others, which indicates the end of the evolution for that synoptic graphlet, as shown by Points A and B at $P = 20$ and C at $P = 50$, (that correspond to synoptic graphlets A, B, and C in Figure 2.9). The high predictability value for these synoptic graphlets at low snapshots confirms the observation made from the evolutionary tree that the future of these graphlets is early set and hence that can be easily distinguished with fewer packets than other types of graphlets.

2.7 Discussion

Revisiting BLINC. The results presented in Secs. 2.4.2 and 2.5 validate the concepts at work in BLINC, as most of the auto-generated synoptic graphlets can be related to empirically defined BLINC graphlet models [1]. However, such heuristic model-based approaches face the potential difficulties in (a) designing appropriate rules as indicated by the observed unknown clusters and in (b) determining the relevant values of thresholds for accurate classification as partially implied by a prior work [16], which conducted a number of trials to determine appropriate parameters. Instead, unsupervised approaches can potentially uncover new types of applications with the tuning of only a very limited number of threshold levels. In addition, an advantage of our approach should be to avoid the assumption that the traffic of one host should be mostly explained by a single application[11].

[11]To show the non-negligible amount of application mixture of a host, we quantify the degree of this for a host h as $p_{max}(h) = \max_a \frac{\#flow(h,a)}{\#flow(h)}$, where a is an application (except for UNKN),

Traffic characteristic evolution when increasing the number of analyzed packets. Sec. 2.6 showed that the method requires around 100 packets to classify hosts. This is larger than the findings of a few previous works. For instance, the work reported in [21] showed that major TCP flows can be identified on bi-directional links from their size and direction, by examining only the first four or five packets (after the handshake) in a connection. Other works [20, 17, 18] also claimed such an ability. The present work, however, deals with more general assumptions about traffic: uni-directional links, legitimate as well as anomalous and unknown traffic, a few protocols besides TCP, not certainty of observing the first packets of flows. In this context, the need to collect a larger amount of information to predict traffic characteristics does not come as a surprise. Moreover, our work is to profile hosts, not only identifying the application in a TCP connection.

Limitations. (a) The degree-based features used here do not include relations among non-neighboring columns such as A_1 and A_3. (b) In addition, real graphlets are not as clean as rewired ones, because they include packets unrelated to the main behaviors of the hosts. Features could be weighted to remove such noise, e.g., the width of edges and the radius of nodes could be set based on the number of packets. (c) In some cases, host behaviors may result from two dominant kinds of applications, e.g., a host serving both mail and DNS, or a NAT gateway with a web client and a P2P user. Such a host cannot easily be profiled. (d) In general, synoptic graphlets only provide shape information; Although such information provides meaningful insight into host behaviors as shown throughout this work, it is still difficult to identify the exact application names used by hosts. If we want to identify them, it would be helpful to put port numbers in the graphlet figure or to cross-compare with classifiers based on port numbers, payloads, IP addresses [9], packet sizes [21, 17], and so on.

Application to supervised approaches. A potential application is to create a reliable dataset of known flows, that then an approach like that of Iliofotou et al. [13] could use. This is because our experiment found graphlets corresponding to a single application (say C_{13} in table 2.3); Such graphlets could be known signatures for any supervised methods. This approach is better than just using a signature generator (say a payload-based classifier), as any classifier will have some misclassification; The clustering scheme presented here can group highly inter-related flows that can be characterized as learning data of enhanced reliability.

Application to unsupervised approaches. Another use case of our method is to help researchers (or network administrators) to interpret the results of unsupervised clustering over graphlets. In general, interpretation of resulting clusters

---

$\#flow(h)$ is the total number of h's flows identified as a certain application (except for UNKN) by the payload classifier, $\#flow(h, a)$ is the number of h's flows identified as application a by the classifier. In other words, $p_{max}(h)$ is the fraction of most dominant application in terms of #flows. For the result for $H = 20,000$ hosts in the 12 MAWI traces, we found that bottom 10% of hosts have $p_{max}(h) < 75\%$, bottom 20% have $p_{max}(h) < 95\%$, and bottom 25% have $p_{max}(h) < 99\%$ (i.e., remaining 75% of hosts are mostly characterized by a single application).

should require to examine a lot of numerical features ($\mathbf{x_h}$), as a prior work [24] does, which becomes significantly difficult as the dimension increases. On the other hand, the use of synoptic graphlet supports such interpretation by converting those features in a single intuitive figure. For example, if a synoptic graphlet for Cluster C contains only a single node in the column for srcPort and the node has several edges, one can easily interpret that C is mostly composed of server hosts (similarly, that for dstPort implies that C is related to client hosts). With such assistance in interpretation, operators will efficiently notice and understand the emergence of new types of application usages (e.g., malicious hosts, P2P software users, or rapid increase in web clients) appearing as new clusters in the monitored link.

2.8 Concluding remarks

The main issue of the present work was the trade-off in choosing between supervised and unsupervised approaches to end-host profiling. The former is comprehensive but is blind to undefined classes, while the latter can uncover unknown pattens of behavior at the sacrifice of interpretability. We aimed to bridge the gap between the two in the present work. The proposed method was designed to perform unsupervised clustering for finding undefined classes and to re-visualize the resulting clusters as synoptic graphlets for providing interpretability. We compared the method against a graphlet-based state-of-the-art classifier (BLINC) as well as against a classical port-based inspector and a payload-based one, by applying these methods to two sets of actual traffic traces measured at different locations. The proposed method spontaneously generated synoptic graphlets that are typical in their shape, which validates the graphlet models heuristically pre-defined in earlier works. Also, for methodological study of the improvements brought to host profiling, this work demonstrated how to extend beyond a simple classification to the production of an evolutionary tree by increasing the number of observed packets per host. The entire procedure requires only a few threshold to be tuned while the state-of-the-art method needs many. The new achievements in this contribution are as follows: (a) an unsupervised clustering applied to graphlet shape-based characteristics, which is further significantly extended to (b) a visualization-oriented auto-enumeration of typical host behaviors generated from actual data, successfully resulting in validating the relevance of past works, and (c) an analysis on evolutionary characteristics of the growth of host behaviors both in visual and quantitative manners, which is useful in understanding the evolutionary nature of host behaviors.

Chapter 3

Structural Configuration Pattern Analysis – Discovering Configuration Templates of Virtualized Tenant Networks in Multi-tenancy Datacenters via Graph-mining

Multi-tenant datacenter networking, with which multiple customer (tenant) networks are virtualized over a single shared physical infrastructure, is cost-effective but poses significant costs on manual configuration. Such tasks would be alleviated with configuration templates, whereas a crucial difficulty stems from creating appropriate (i.e., reusable) ones. In this work, we propose a graph-based method of mining configurations of existing tenants to extract their recurrent patterns that would be used as reusable templates for upcoming tenants. The effectiveness of the proposed method is demonstrated with actual configuration files obtained from a business datacenter network. [1]

3.1 Introduction

There has been a significant rise in the utilization of multi-tenancy in enterprise datacenter networking. Multi-tenancy is a form of deploying customer (tenant) networks, with which multiple customer networks are virtualized and consolidated in

[1]Contents of this chapter have been published as the following article: Yosuke Himura, and Yoshiko Yasuda, "Discovering Configuration Templates of Virtualized Tenant Networks in Multi-tenancy Datacenters via Graph-mining", ACM SIGCOMM Computer Communication Review, Volume 42, Issue 3, pp.13-20, July 2012. (DOI: 10.1145/2317307.2317310)

a single shared physical network and yet logically-independent each other [32, 33]. Leveraging multi-tenancy is hence cost-effective, being enabled by recent technologies of network virtualization; In addition to the conventional L2-level separation with virtual LAN (VLAN), L3-level separation is allowed by the use of virtual interface (VIF) and virtual router (VR). According to the recent increase in the processing performance of network devices, tens or even hundreds of tenants can be deployed in a single datacenter.

One of the essential tasks in multi-tenant datacenters is to create network configuration settings for deploying tenants, which are diversely composed according to various enterprise requirements (e.g., multi-tier architecture). Tenant construction in such datacenters still mostly relies on manual configuration because of the lack of tools that generally meet those various requirements. Manual configuration is substantially time-consuming and error-prone in general as claimed over at least a decade [34, 35, 36, 37, 38], and the use of multi-tenancy in datacenter context has unfortunately required more careful and sophisticated management processes than ever due to the increase in risks of misconfigurations in such shared environments.

A conventional approach for mitigating the burden of manual configuration is to leverage configuration templates. A configuration template is defined as a predefined composition of setting commands without specific assigned parameters (e.g., VLAN ID), which are automatically or manually assigned at the time of deployment. The use of templates is effective when similar form of configurations are repetitively created (i.e., templates are reusable). According to successful cases in other areas (e.g., ISP [35, 37, 38]), we expect that templates would also be effective in multi-tenancy datacenter networks.

A crucial problem in the use of templates, however, stems from the difficulty in creating appropriate (i.e., reusable) ones. Indeed, there is no generally-applicable templates, because the composition of tenant configurations varies depending on the form of physical networks (e.g., device types) and on specific domain-knowledge of network engineers. In addition, finding appropriate templates is uneasy as the frequent patterns are a priori unknown.

In this work, we aim to automatically generate appropriate configuration templates for deploying tenants in multi-tenant datacenters. The main idea towards this goal is derived from mining configurations of existing (i.e., already-deployed) tenants to find typical patterns of them, which would be represented as reusable templates for upcoming tenants. We propose a method based on (a) an abstraction of tenant configuration as attributed graphs, which enables to measure similarity among them, and on (b) an unsupervised clustering over those attributed graphs, which produces groups of similar (i.e., recurrent form of) tenants. The proposed method is evaluated with actual configuration files obtained from a multi-tenant production datacenter. This evaluation demonstrates that the proposed method correctly finds typical patterns of tenant configurations that can be effectively used as configuration templates for upcoming tenants. Originalities of the present work

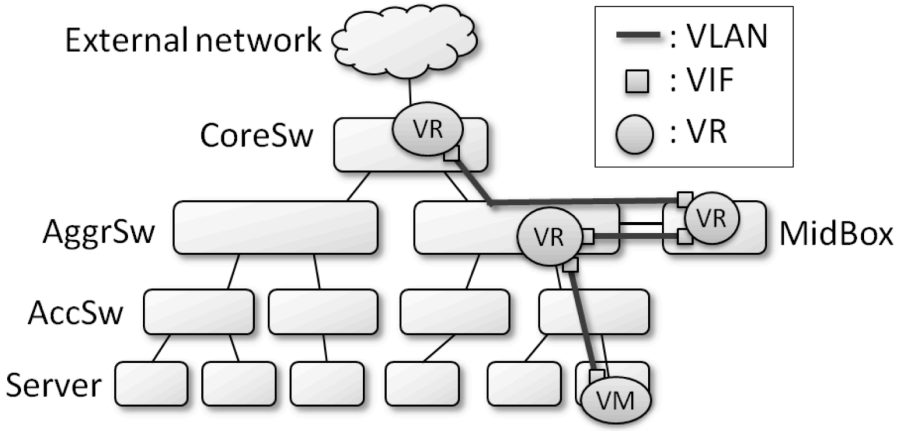

Figure 3.1: Schematic representation of a conventional physical network of datacenter deploying a virtual network for a tenant.

include (1) the method of automatically finding typical patterns of tenant configurations for creating appropriate templates, and (2) the case-study with an actual production datacenter.

3.2 Background and motivation

We first present the basic construction of the physical network in a datacenter (Sec. 3.2.1) and how network devices are configured in order to deploy virtualized networks for tenants over the datacenter (Sec. 3.2.2). Then, we explain the problem of difficulty in creating configurations (Sec. 3.2.3).

3.2.1 Multi-tenancy datacenter network

Figure 3.1 illustrates a conventional datacenter network. Typical datacenters comprise following devices.

- Access switches (AccSw) connect multiple servers basically with L2 functionalities.
- Aggregation switches (AggrSw) aggregate access switches and connect middleboxes, providing L3 connectivity.
- Middle-boxes (MidBox) provide network services such as firewall filtering, load balancing, NAT, VPN termination, or SSL acceleration.
- Core switches (CoreSw) further merge traffic from aggregation switches if necessary and connect to external networks (e.g., enterprise networks, or the Internet).

(a) Firewall

```
1  set vrouter "vrA"
2  set route 0.0.0.0/0 gatway  10.0.1.1
3  set route 0.0.3.0/24 gateway 10.0.2.2
4  exit
5  set zone "trust"  vrouter "vrA"
6  set zone "untrust" vrouter "vrA"
7  set int eth0/1.10 tag 10 zone "untrust"
8  set int eth0/1.10 ip 10.0.1.2
9  set int eth0/2.20 tag 20 zone "trust"
10 set int eth0/2.20 ip 10.0.2.1
```

(b) AggrSw

```
11 ip vrf v1
12  route  0.0.0.0/0  10.0.2.1
13 !
14 vlan 20
15 !
16 vlan 30
17 !
18 interface vlan 20
19  vrf forwarding v1
20  ip address 10.0.2.2/24
21 !
22 interface vlan 30
23  vrf forwarding v1
24  ip address 10.0.3.1/24
25 !
```

Figure 3.2: Configuration commands of a firewall MidBox (left) and an AggrSw (right) for the tenant network depicted in Figure 3.1.

We note that datacenter is usually constructed as redundancy structure to achieve high availability, although we omitted this from the figure for readability.

This single infrastructure consolidates separated virtual networks for multiple customers, which we define as tenant networks. Figure 3.1 also illustrates a tenant network deployed over the above-mentioned physical devices. Engineers use following virtualization technologies to achieve logical independence among different tenant networks.

- Virtual Router (VR) generates logical instances for routing functionality inside a device. A representative of VR used in many networking fields is known as Virtual Routing and Forwarding (VRF).
- Virtual Interface (VIF) creates logical instances of network interface. Well-known realizations of VIF are VLAN interface and sub-interface.
- Virtual LAN (VLAN) (mainly IEEE 802.1Q) is a common technology for L2 separation, which virtually separates a physical link into logically independent links.

These virtualization technologies create logical instances of network components such as routers, interfaces, or links – we name them virtual resources, and hence a tenant network can be interpreted as a collection of virtual resources.

3.2.2 Tenant configuration

Configuration settings in the networking literature are generally stored as configuration files, each of which is a list of commands of networking functions. Figure 3.2 represents two examples of configuration files used to construct a tenant network.

Here, Figure 3.2(a) and (b) correspond to a list of configuration commands for a firewall as MidBox role and a switch as AggrSw role, respectively. Those configuration files are related to the tenant network depicted in Figure 3.1. We note that these configuration files are partially modified for brevity and not exactly same as those of existing products. These configuration files can be broken down in terms of virtual resources as follows.

- Allocating VR instances and specifying static routes (lines 1-4, 11-13): The virtual router named vrA is allocated in the firewall and that tagged as v1 is allocated in the switch.

- Creating VIF instances and attaching them to a VR instance (lines 5-10, 18-25): For instance, the firewall creates two VIFs named eth0/1.10 and eth0/2.20, and connect these interfaces with the virtual router vrA via the definition of zones, each of which is a logical grouping of interfaces. These lines include the assignment of IP address on the interfaces; IP addresses on VIFs are essential for routing among multiple VLANs.

- Defining VLAN instances and relating them to VIFs (lines 7, 9, 14-17, 18, 22): Some lines can be interpreted as creating VLAN instances from the viewpoint of virtual resources, although they are not explicitly declared. For example, lines 7 and 9 can be interpreted as creating VLANs 10 and 20 and attaching them to the physical interfaces eth0/1.10 and eth0/2.20.

Our empirical observation on real enterprise datacenters was that tens or even a hundred of tenants are contemporarily consolidated in a single shared infrastructure, resulting in thousands (or even tens of thousands) of lines of commands.

3.2.3 Problem in deploying tenants

Usual procedure of deploying a tenant has mostly been conducted in a manual fashion due to the lack of tools suitable for the variety of requirements for enterprise networks (e.g., multi-vendor infrastructure, multi-tier tenant topology). Manual configuration is notoriously time-consuming and error-prone as claimed over at least a decade.

The use of template would alleviate the burden of manual configuration in general. Templates are effective as long as there are recurrent patterns of provisioned networks because of their reusability. We empirically observed such recurrent patterns in multi-tenancy environments, and hence we consider that this general approach would be applicable as well.

A crucial problem in the use of templates, however, is the difficulty in creating appropriate set of configuration templates. We regard the appropriateness of templates as their reusability, and thus such templates should reflect the frequent patterns of tenant configuration; However, such frequent patterns are generally a priori unknown.

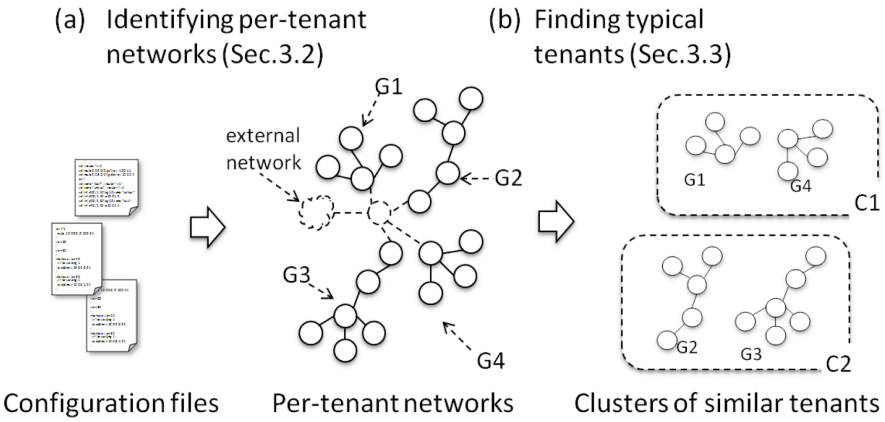

Figure 3.3: Overview of the proposed method.

3.3 Methodology

3.3.1 Overview

We aim to automatically generate appropriate (i.e., reusable) configuration templates for tenant deployment in multi-tenant datacenters. In the present work, we focus on the topologies of tenants as the first step toward this goal, because most of networking functions such as routing, filtering, and/or quality-assuring are realized after constructing network topology. The main idea for auto-finding topology templates is based on mining existing tenant network configurations to discover recurrent patterns of their topologies that can be repeatedly used as templates for upcoming tenants. We develop an automated method, which is represented as Figure 3.3, consisting of two major steps: (a) the identification of per-tenant networks by analyzing configuration files and abstraction of them as attributed graphs (Sec. 3.3.2), and (b) the discovery of recurrent patterns of tenant networks based on a graph mining technique performed over the abstracted tenant networks (Sec. 3.3.3).

3.3.2 Identifying tenant networks

The first task is to identify per-tenant network topology as there is no direct mapping between a tenant network and a device configuration. We first present the model of tenant network (Sec. 3.3.2.1) and the method of acquiring them from configuration files (Sec. 3.3.2.2).

3.3.2.1 Tenant network model

Since a tenant network consists of a set of virtual resources and connections between pairs of them as described in Sec. 3.2.2, any tenant network can be abstracted as

an undirected graph; A node (vertex) represents an instance of virtual resource (i.e., VR, VIF, VLAN), whereas a single edge stands for the referential relationship between a pair of virtual resources. This device-neutral abstraction allows us to leverage a variety of analytical capabilities for graphs.

We formally define a tenant topology as an attributed graph $G = (V, E, \Sigma)$, where $V = (v_1, \ldots, v_N)$ is a set of nodes, $E = (e_1, \ldots, e_M)$ is a set of edges (also noted as $e_i = (v_j, v_k)$), and $\Sigma : V \to L$ is a labeling function that maps a node to a label. A label $l \in L$ is composed of 4 attributes $l = (t, p, d, r)$, where t is the type of virtual resource (e.g., VLAN, VR), p is the parameter of virtual resource (e.g., 10, eth0/2.20, vrA), d is the ID of device, and r is the role of device (e.g., coreSw, aggrSw). For example, a virtual router vr3 in core switch CS1 is abstracted as a node v_i with the corresponding label $L(v_i) = (t, p, d, r) = (VR, vr3, CS1, coreSw)$. The first 3 attributes (t, p, d) are required for uniquely identifying the virtual resource among others in the network, whereas we use 4th one (r) for the configuration mining as shown in Sec. 3.3.3.

We note that the device id (d) and the device roll (r) of VLAN node should exceptionally be constant across all the devices (e.g., $d = null$), because we regard a VLAN node as an instance of an aggregated L2 network connecting multiple virtual interfaces across devices. This abstraction of an L2 network as a node hides the need for inputting physical topology to obtain correct topology, which can be conducted when networks are correctly configured; This would be true in the in-service datacenter, where deployed tenants are actually and correctly providing services.

We also note that the definition of the graph G does not contain virtual hosts (VMs) as nodes, but contain only virtual networking resources (e.g., VLAN, VR). The main motivation for this is that the number of VMs can easily be various according to the computational requirement from tenants, compared to that of networking nodes. Even if a pair of graphs have the identical network configuration, those graphs would produce high dissimilarity if they contained different number of VMs as nodes, impairing the discovery of typical networking configurations.

3.3.2.2 Acquiring tenants from configuration files

We select to reverse-engineer configuration files to obtain tenant networks. This choice is motivated by the fact that those files (a) are commonly available for engaged network operators, (b) store correct running status of networks, and (c) can be automatically analyzed thanks to their pre-defined syntaxes. Identification of tenant topologies is, however, not obvious due to the inexistence of direct mapping between a configuration file and a tenant; A single tenant network is composed of several virtual resources allocated by several devices, whereas a single device allocates lots of virtual resources for multiple tenant networks.

Identification of tenant topologies from configuration files can be achieved according to the following two steps. (1) According to the tenant network model (Sec. 3.3.2.1), we first generate the nodes from all the configuration files, and then

produce the edges connecting these nodes by re-scanning those files. (2) We use a conventional graph traversal method to obtain per-tenant networks appearing as 'connected components' inside the entire network reproduced by the above step (Figure 3.3(a)). This second step is plausible because any pair of tenant networks must not be inter-connected due to the principle of multi-tenant networking.

Indeed, there might be a case where tenant networks are exceptionally interconnected at a gateway router; We need to remove the corresponding node before performing the graph traversal in this case. Removal of the shared nodes can be both manual and automatic; Those nodes can manually be indicated by operators, or can automatically be identified based on heuristics (e.g., such shared node will have many edges than the others). In our case, we manually indicated the shared nodes beforehand, as they were a priori known.

3.3.3 Finding typical tenants

The second step after identifying tenant networks is to extract similar patterns of them. We breakdown this issue into defining similarity between a pair of topologies with graph edit distance customized for this specific issue (Sec. 3.3.3.1), and finding clusters of similar topologies with an unsupervised clustering (Sec. 3.3.3.2).

3.3.3.1 Computing tenant similarities

Discovering typical patterns can generally be achieved by unsupervised clustering, which groups similar objects on the basis of a quantified similarity measure. Conventional similarity is Euclidean distance between two vectors, but unfortunately this measure cannot be directly applied to structured graphs.

In contrast to the conventional approach, we leverage graph edit distance $D(G_1, G_2)$ to quantify the similarity between a pair of graphs G_1 and G_2. The graph edit distance is the minimum number of operations needed for transforming G_1 into an isomorphic of G_2 [39]. The transformational operations are addition/deletion of a single node/edge. Here, $G_1 = (V_1, E_1, \Sigma_1)$ and $G_2 = (V_2, E_2, \Sigma_2)$ are isomorphic if there is a bijective function $f : V_1 \to V_2$ satisfying both (a) $\Sigma_1(v) = \Sigma_2(f(v))$ for $\forall v \in V_1$, and (b) $(f(v_i), f(v_j)) \in E_2$ for $\forall (v_i, v_j) \in E_1$. The isomorphism between a pair of graphs means that the two graphs have the same set of labeled nodes representing the identical structure. Higher $D(G_1, G_2)$ indicates higher degree of structural difference between G_1 and G_2.

In the context of discovering similar patterns of tenant networks, we decided that two nodes can be mapped via f if their node types t and device roles r are same. In other words, this similarity does not consider the parameter of virtual resource p and the ID of physical device d. The basic structure of tenant networks can be represented with t and r, and thus assigned parameters p and actual used devices d should be determined at the moment of using a template (not the moment of creating templates). Hence instead of the labeling function Σ (Sec. 3.3.2.1), we

define an alternate labeling function $\Sigma' : V \rightarrow L'$, where $l' \in L'$ consists of the 2 attributes $l' = (t, r)$. With this similarity measure, we compute the $N \times N$ similarity matrix $A = (a_{i,j})$ for N graphs $G_1, \ldots, G_N$, where $a_{i,j} = D(G_i, G_j)$.

The main motivation to use the graph edit distance is that this metric directly reflects the engineering workload to create a tenant configuration settings. For example, adding a node to a graph is similar to the attaching a virtual resource (e.g., VR) with a tenant network. This engineering nature would not easily be represented by isomorphism or by statistical metrics such as eigenvalues, degree distributions, centrality measures (e.g., betweenness), and so on.

3.3.3.2 Clustering tenant networks

Unsupervised clustering produces clusters of graphs based on similarities between graphs. A collection of N graphs $G_1, \ldots, G_N$ is converted into a set of M clusters $C_1, \ldots, C_M$ (Figure 3.3(b)). A cluster C_i is a collection of graphs that are close in the distance space, i.e., similar in terms of topological shape.

We use an aggromerative hierarchical clustering [40]. This technique starts with N clusters, each of which contains a single graph, then a pair of most similar clusters are merged as one. The operation of merging clusters is iterated until any pair of clusters does not yield distance higher than a threshold θ. Here, the distance of two clusters $D(C_i, C_j)$ is defined as $D(C_i, C_j) = E(C_i \cup C_j) - E(C_i) - E(C_j)$, where $E(C_i)$ is the Euclidean intra-cluster distance within C_i. The inter-cluster distance $D(C_i, C_j)$ can be computed from the similarity matrix A, following the Lance-Williams similarity update formula. Refer to Ref. [40] for details. Clusters obtained with $\theta = 0$ include only exactly identical structure of graphs, whereas those with $\theta > 0$ result from fuzzy matching among graphs.

3.3.3.3 Computational load

The computation of the graph edit distance is most expensive as it requires exponential time- and space-complexity. Computational costs we measured were less than 10 sec. with a 3.3GHz CPU and 1GB memory space for conventional graphs (around 10 through 30 nodes – most of the graphs examined in this work); Those for large graphs (around 100 nodes) were unfortunately intractable due to the large search space, but this is insignificant because such graphs are generally atypical in practice so that template finding does not require the exact computation of similarity regarding such highly-different graphs. One option for this issue is to use the isomorphism as an alternative similarity measure (e.g., 0 for isomorphic and 1 for non-isomorphic) instead of the graph edit distance for low computational cost at the sacrifice of fuzzy clustering. More specifically, the use of isomorphism corresponds to the $\theta = 0$ case of the use of graph edit distance, and hence can only obtain clusters with $\theta = 0$ as well as cannot provide interpretation of the degree of dissimilarity among different clusters.

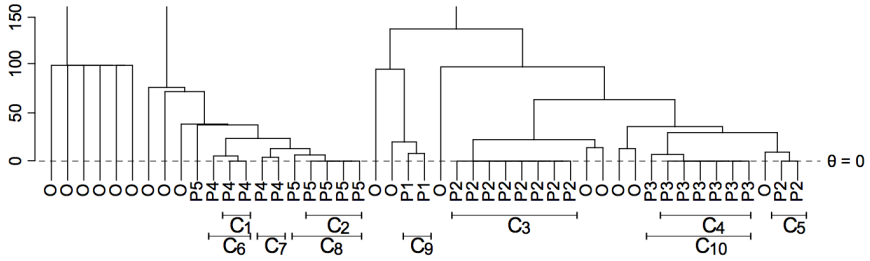

Figure 3.4: Clustering result represented as a hierarchical view with manually classified labels, showing the characteristics of the threshold θ.

On the other hand, the clustering method requires $O(N^2)$ with N objects for both time- and space-complexity, and we observed that 1,000 objects are clustered within a few minutes with that CPU, requiring 10 MB memory space.

Overall, the total processing time we measured with the dataset described later was less than 3 hours (computation of intractable cases were not completed but terminated during the processing when reaching to the limitation of memory). Hence, in practice, this processing can daily be performed to follow the longitudinal changes in typical pattern of tenant networks.

3.4 Validation

3.4.1 Dataset

The feasibility of the proposed method is demonstrated with a set of configuration files obtained from switches and firewalls in an actual business datacenter. This datacenter offers a hosting service that consolidates virtualized systems of various customers in a multi-tenancy fashion. More than 1,000 instances of virtual resources are combined to form a number of virtualized tenant networks. Those configuration files compose tens of thousands of lines in total.

In order to well interpret the results produced from the method, we create a referential information of tenant topologies by inspecting operational documents maintained by engineers dedicated to that datacenter. These documents consists of the drawing of topological structure of virtual resources allocated for each tenant. As a preliminary setup, we put effort on manually classifying around 50 tenants by examining their visual figures. This procedure identified 5 typical patterns based on their service usage. We annotate these typical patterns as P1 through P5 and atypical patterns as O (i.e., Other).

3.4.2 Relevance of method

Figure 3.4 displays the clustering result of tenant topologies. The horizontal line represents a series of the tenants (labeled by hand), whereas the vertical line measures the similarity between tenants (e.g., zero similarity means identical topology). We found that there were intractable similarity values for large graphs as discussed in Sec. 3.3.3.3. We decided to set the similarity 100 in that case as it is unimportant for template finding to exact computation of similarity regarding such highly-different graphs. Such topologies are located at the left area in the figure. Labeled clusters C_1 though C_5 are obtained with $\theta = 0$, whereas C_6 through C_{10} are produced with $\theta > 0$.

This figure leads to the following findings.

- The resulting clusters having the same form of tenants imply that the use of template would be effective (i.e., reusable) in the examined environment. For example, with respect to the threshold $\theta = 0$, there are 8 same topologies labeled as P2 in Cluster C_3, and 6 ones labeled as P3 in Cluster C_4. Approximately 50% of tenants are clustered into C_1 through C_5 in total. Regarding the remaining half of tenants, some are clustered with $\theta > 0$, but others (mainly labeled O – about 35%) are not clustered, as shown below.

- Also, this method successfully captured minor clusters containing small amount of tenants (e.g., C_1, C_5). Indeed, it should be easy to manually enumerate major clusters containing high number of tenants (e.g., C_3, C_4 – each of which contains over 10% of total tenants), because human can easily perceive frequently repetitive patterns from the entire set of tenants. On the other hand, however, manual enumeration will tend to overlook less frequent patterns; This difficulty will become polynomially significant according to the increase in the total number of tenants N, as finding typical tenants requires to compare each pair of two tenants, resulting in $N \times (N-1)/2$ comparisons.

- Regarding the fuzzy clustering with $\theta > 0$, we discovered several tenants that are similar yet not exactly identical (e.g., Clusters C_6 through C_{10}). We manually inspected their topologies and found that their difference mostly came from the number of network segments (VLANs) in a tenant. Such tenants would be constructed with small modification of configuration commands generated via templates, as the graph edit distance directly reflects the number of modification steps. For example, C_{10} tenants can be created with small modification from C_4 configuration commands since $C_4 \subset C_{10}$. Also, we consider that the slight differences among the tenants within a fuzzy cluster can possibly be further parameterized, and such parameterization can be further effective in creating configurations; However, automatic extraction of parameterization is a difficult problem, which should be explored in the future work.

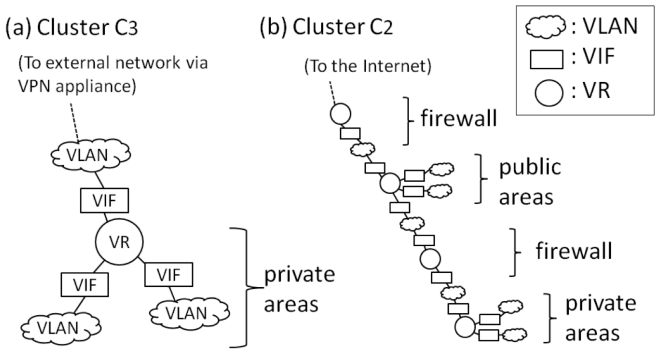

Figure 3.5: Examples of typical patterns found.

- On the other hand, there are a non-negligible amount of atypical tenants (labeled as O), which are not similar to any of typical ones; Such tenants account for about 35% of total tenants as shown in Figure 3.4. Manual inspection on the composition of those tenants revealed that some of them consisted of tens of even around a hundred of virtual resources; Those tenants cannot be effectively represented as templates and should be manually customized as value-added services.

Figure 3.5 exhibits two of the typical patterns found by the proposed method.

- Figure 3.5(a) shows the most frequent pattern of tenant topology (Cluster C_3), which is composed of a single VR and a few VLANs. Consulting the operators' documents, we found that one of the VLANs is attached to a VPN device, which is used in establishing private connections between this datacenter and customer sites via external networks.

- On the other hand, Figure 3.5(b) displays another recurring pattern (Cluster C_2), which consists of two types of areas, i.e., public areas and private ones; The public areas are connected to the external network via a virtual router in a firewall, and to the private networks through another virtual router of firewall as well. This is a common construction of multi-tier network architecture with different security levels across tiers.

To summarize, we found that the proposed method fruitfully captured common usages of virtualized networks in a multi-tenant datacenter.

3.4.3 Operational practicability

The automated discovery of typical tenant configuration allowed us to implement a prototype of multi-tenancy-aware networking management tool, which automatically generates a set of configuration files for a tenant based on the 5 configuration

Table 3.1: Operational cost with and without templates based on auto-discovered patterns.

	Operation	w/o template	w/ template
1	Topology design	20 min	20 min
2	Configuration creation	120 min	$\sim$ 0 min
3	Self-validation	60 min	60 min
4	Cross-validation	60 min	$\sim$ 0 min
	Total	240 min	80 min

templates corresponding to the discovered patterns C_1 through C_5 (can be obtained with $\theta = 0$). The parameters of various types of resources (e.g., VLAN ID, VRF ID) are automatically assigned by the tool so as not to cause misconnection to any of the existing tenants. The users of this tool only have to select an appropriate template if it matches with the requirement from the upcoming customer. As shown in Sec. 3.4.2, about 50% of tenant networks can be represented as one of the 5 templates, although this service still accepts customized tenant networks.

An engineer actually used this tool. Usual procedure to deploy a tenant was as follows (Table 3.1): (1) designing the topological structure of a tenant and drawing a document regarding that topology; (2) creating configuration commands for deploying that tenant; (3) validating the correctness of the commands (before inputting into devices); (4) cross-validating those commands (performed by another engineer to avoid human-errors).

The use of the tool mitigated the burden of this operational procedure as follows.

- The tool achieved in reducing the operational time of the per-tenant network construction/validation from 240 min. to 80 min. The breakdown of the operational costs is displayed in Table 3.1. Configuration creation is fully automated by the template tool, whereas topology design still requires operational cost because of the need for manually creating documents including the drawings of their topologies. On the other hand, despite auto-generated configurations are correct and not needed to be validated in essence, configuration files generated by the tool are still cross-validated by the engineer to assure the correctness of the prototype.

- In addition to the above achievement, the tool contributed to reducing the number of dedicated engineers from 2 to 1. The cross-validation requires another engineer in addition to the one in charge of creating configuration commands for the tolerance to human-error, needing at least 2 engineers in total. Since the template tool plays the role of configuration creator, there needs only 1 engineer for validation.

- The automation of deploying typical tenants enables engineers to concentrate

their effort on the atypical (value-added) deals. This means that atypical cases could indirectly be accelerated by the template tool by mitigating the total operational cost on engineers.

In summary, the templates based on the discovered configuration patterns contributed to reducing operational time, human costs, and the possibility of human-error although not perfect yet.

3.5 Related work

Configuration templates. A few works have developed configuration templates in different networking fields (e.g., ISP [35, 37, 38]), and the present work shares the spirit of the use of templates. In contrast to those work, an originality of this work is the focus on the fundamental and crucial difficulty in creating appropriate templates. Indeed, Refs. [41, 42] automatically find frequent device-level configuration patterns by means of a device-to-device copy-and-paste detection tool. However, this kind of tool cannot be directly applied to finding typical patterns of tenant-level network-wide configuration, because tenant networks are structured and not constructed over a single network device. We instead select the approach of finding typical network-wide configuration by customizing a graph-mining technique according to the context of this problem. To the best of our knowledge, this is the first detailed statement on appropriately finding and using configuration templates in multi-tenant datacenter networks.

Network configuration analysis. There have been various efforts on analyzing network configuration files. Configuration files have been used mostly in understanding network characteristics such as routing design [43], Class-of-Service [44], Route Redistribution [45], complexity of enterprise network [41], access control lists [46], co-occurrence of device updates [47], operational history [48], and longitudinal trends in ISP services [42] and campus networks [49]. Ref. [36] argues the possibility of the use of graph-mining on configuration files, but unfortunately does not provide details. This work newly develops the graph-mining-based method of analyzing configuration files.

Management in multi-tenant datacenters. In the context of device settings in multi-tenant datacenters, there are a few works on future datacenter architectures addressing management aspect as one of primary issues [50, 51]. On the other hand, we focus on current situation where engineers use existing products of hardware and software commercially available today, which new architectures of future paradigm cannot be directly applied. Existing tools (mostly commercial) provide intuitive user-friendly GUI interface for ease of settings, but such settings seem have to be made per instance of logical resource (e.g., VLAN, VRF), different from per-tenant deployment that we aim to achieve.

3.6 Discussion

Generality. We consider that the proposed method would be applicable to a great deal of other multi-tenant datacenter networks, only if tenant networks are logically independent at the level of virtual resources[2]. Device-specific languages are abstracted as device-neutral graph components (nodes/edges) for generality, and virtual resources defined in this work (VLAN, VIF, VR) are essential building-block of computer communications in principle. Configuration compositions of tenant networks are not necessarily specific to the analyzed datacenter, because a few design guides recommend similar (although not exactly same) patterns [32, 33]. The effectiveness of the method would be impaired when all tenant networks are totally different, but this should be unrealistic in actual environments.

Limitation. The present work mainly focused on topological aspects of tenant networks, and did not capture access control functions such as routing, filtering (ACL), quality assurance (QoS). The identification of typical topologies should be the first step to discuss the capability and effectiveness of finding template regarding those access control functions, because they are performed over deployed topologies. Although the use of ACL has become less significant in multi-tenant datacenters (because the inter-tenant isolation is achieved by the logical independence of virtual resources), those remaining tasks would be interesting future works.

3.7 Concluding remarks

We developed a configuration analysis method to automatically find appropriate (i.e., reusable) configuration templates in deploying tenant networks for virtualized multi-tenant datacenter networks. The main idea was based on mining existing tenant configurations to discover typical patterns that can be converted as reusable templates for upcoming tenants. The evaluation with actual configuration files obtained from a production datacenter network demonstrated the effectiveness of the proposed method. Future works include discovering domain-knowledge from configuration information in a systematic manner as well as quantifying the scalability characteristics of the method. Also, another interesting future work is to investigate the applicability of this method to multiple datacenters.

[2]The proposed method cannot be directly applied to the case where separation among tenants are realized at application-level; In this case, the graph model has to additionally include the application-level information of separation, and the mining method will need to be extended to handle them, although the fundamental idea of this paper (i.e., mining per-tenant information) would be applied.

Chapter 4

Discussion and Future Perspectives Towards Efficient Analysis for Management of Field Domains

Contrary to the previous chapters (Chapters 2 and 3) that dealt with traffic and configuration data in the computer networking domain, this chapter discusses the possibility of extending those analytical approaches to the management of field domains (e.g., building), which have become more important as more field devices become connected. We consider that the spirit of data analytics for the management of end-hosts (by traffic analysis) and virtual network configurations (by configuration analysis) should have commonality to some extent for the of filed domains such as to manage end-devices (e.g., sensors and actuators) and their inter-link configurations. However, analytics on data obtained in field domains is relatively difficult due to the differences in the characteristics of those domains (e.g., open systems for field domains, rather than closed systems in computer networking domain), particularly leading to the difficulty in interpretation of analytical results. In addition, characteristics of data in computing fields are rather aggregated view of discrete and relative information (i.e., aggregated cardinality), which would be suitable for the discrete structural model (e.g., graphlet), whereas field data additionally suggests the importance of individual point-by-point time-series view with non-discrete and absolute values. We discuss the need for trial-and-error approach of analyzing field data (e.g., per-room temperature in a building), interpretation of which is gradually complemented by introducing multiple types of contexts (e.g., switch states of air-conditioning appliances in rooms) and even other data sources (e.g., external temperature outside the building) that are considered as structurally related. This approach is demonstrated with an unsupervised analysis over an actual dataset about Electrical Heat Pump (EHP) equipped in a number of rooms in a building.

4.1 Introduction

There has been increasing expectations about digitalization in field domains, which is often termed as Internet of Things (IoT) in the recent literature. A representative example of such the digitalization includes building automation or smart building that deploys sensors and actuators inter-linked with computer networkings inside a building in order to measure digitally-processable information about the usage of resources (e.g., electrical power consumption) and the state of buildings (e.g., per-room light and air-conditioning settings, and temperature); Those data are used for holistic visualization, finding knowledge based on analytics (e.g., detecting unnecessary consumption of electrical power), and further controlling equipped appliances to the desired state at the fine-grained views. Such the expectation is supported with the recent technological advances in sensing, sensor networking, communication protocol and data format for field domains (e.g., IEEE 1888), and data analytics (e.g., Artificial Intelligence (AI) including Machine Learning (ML)).

The deployment of digitalizations will also suggest the importance of the management in field domains. Digitally-connected sensors/actuators deployed in fields will enable additional use cases about field management such as to detect and reduce unnecessary power consumption in building and/or to detecting faults of appliances at earlier stages; Also, such the digitalization with sensors and actuators instead requires additional management of those devices, which may become important because those devices can be essential components of field controlling (e.g., building automations); Errors and faults in those devices can possibly affect the entire behaviors of the controlled field domains.

We consider that such the management in the field domain can promisingly be supported with the extension of unsupervised analysis for network management described throughout this dissertation. This perspective is motivated from the similarity in the general notion of management tasks and observable data, based on the comparative discussion about the two domains (i.e., network management and field management). From the viewpoint of management tasks, field management also relies on data flow management (i.e., handling sensor-generated data flowing sensor networks in field) and configuration management (i.e., dealing with network-wide configuration of inter-sensor/actuator network), which is respectively comparable to traffic and configuration management in a broad sense by analogy. In addition, with regard to the observable data, there will be many similar end-devices deployed in a single field domain, resulting in the practical possibility of comparative study of data observed with those devices (i.e., unsupervised analysis) in order such as to finding misbehaviors of end-devices (e.g., faults and misconfiguration in end-devices) and that of intermediate networks (failures and misconfiguration in intermediate assets).

However, as stated throughout this dissertation, unsupervised analysis is in general faced with the difficulty in interpretation of analysis outputs, and this difficulty will become more significant in the field domain. This difficulty of interpretation

in field domains stems from their system characteristics, and accordingly data characteristics in the domain. In terms of system characteristics, different from computer networking domain, which is relatively closed system of interactions among end-hosts, field domains are rather open systems, where end-devices (i.e., sensor/actuator) are affected through external factors such as human intentions and outside climates. Accordingly, as data characteristics, there would be limited view of the entire system from the dataset observed only with the sensors of limited scope (i.e., only temperature sensor and/or only power consumption meter). Furthermore, different from natively-digital computer networking domain that are suitable for analysis of discrete and relative values (i.e., graph analysis), the field domain is originally analog and will not necessarily be directly suitable for that analysis, which indicates the difficulty in direct application of graph-based structural analysis to the field domain.

In this present work, we discuss the use of unsupervised analysis in field domains, particularly building domain. We focus on data flow management of end-devices (including misbehavior detection) due to the current availability of data, rather than configuration management as multi-tenant virtual sensor network is not currently widely-deployed. Instead of graph-based analysis, we take a trial-and-error approach that gradually introduces additional sources of information starting from simple single type of data (i.e., measured temperature with sensor attached with an appliance) to multiple types of data (i.e., internal setting information of the appliance) and to macroscopic external data (i.e., external global temperature) that are considered as structurally related; This enables to complement interpretation of analysis outputs. We demonstrate this approach with an actual dataset about Electrical Heat Pump (EHP) equipped in a number of rooms in a building.

Table 4.1: Comparison between field domain and computer networking domain in terms of management tasks and domain objects.

		management tasks and domain objects	
		management of data flow from end-devices (measured as time-series data)	management of network configurations (measured as spatial data)
domain	computer networking domain	managing traffic data generated from a number of end-hosts over shared network	managing configurations of a number of virtual networks over shared infrastructure (e.g., wide-area network, datacenter network)
	field domain	managing sensory data measured from a number of sensor devices (e.g., attached with individual appliances and spaces) within a single domain	managing configurations of a number of virtual sensor networks within a single domain (e.g., inter-sensor/actuator networks for individual use)

4.2 Preliminaries

4.2.1 Possibilities: extending unsupervised analyses of computer networking domain to field domains

Improving management tasks in field domains has recently been increasingly motivated from the following aspects.

- **Opportunity for data-driven management in field domain**. Recent increasing deployment of sensor devices (e.g., equipped with building appliances) opens the possibility of enabling to collect small-grained and diverse data that will be meaningfully analyzed for the use of improving management in that domain, which would have conventionally been conducted in an offline as well as manual manner based on experts experiences.

- **Need for managing newly-deployed sensor devices and networks**. Also, deploying sensor/actuator devices instead requires management of those devices (and their networks) themselves. Management of those devices will become important as management in that domain relies on automated sensor/actuator-based approach; Failures and errors can possibly impact the running state of the entire management domains.

Those emerging possibilities are derived from the fact that increasing amount and variety of data in field domains have been becoming available due to the recent advancement of sensor devices, that are networked especially through open standards

of communication protocol and data format (e.g., IEEE 1888), with which those variety of data can be collected and made accessible to many developers. An expectation is that developers (including third party) develop applications of management improvement by using wide variety of available data, which was conventionally stored in closed domain due to vendor specific protocols and management systems.

Similar to the data-driven management in the computer networking domain, one difficulty is to find out appropriate combination of data types and their analysis methods used (from the choices on a variety of available data and analytical tools), and we consider that there will be promising possibility of extending the notion of unsupervised analysis (for computer networking as presented in the previous chapters) to field domains. This consideration is motivated by our observation about (a) management tasks and (b) objects deployed over domain as follows (and summarized in Table 4.1).

(a) Management tasks. As the aspect of management tasks in field domain, the tasks will also be classified as follows (although this may not be widely-used).

- **Data flow management**. As previously mentioned, one important tasks in the computer networking domain is to manage traffic data that are generated from end-devices (i.e., end-hosts) connected to the network. Traffic data can be regarded as data flow in that specific domain, which we consider would be conceptually comparable to the field domains, i.e., managing sensory data flow generated from end-devices (i.e., sensor devices). Those data in the two domains tend to be sequentially observed as aggregated time-series data for individual end-devices.

- **Configuration management**. Also, as previously mentioned, another important task in the computer networking domain is to manage network-wide configurations originating from the settings of individual networking devices. Network configuration itself will be formed in sensor network that topologically connect sensor/actuator devices as well as servers (e.g., storage server that stores sensory data). Spatial configuration management is hence at least conceptually required also in the field domains.

(b) Domain objects. As the aspect of objects in domain, there also becomes increasing amount of similar objects within a single domain that can be comparable through unsupervised approach.

- **End-devices**. As previously-mentioned, there are a number of end-hosts sharing the same network. The behaviors of those end-hosts (observed as per-hosts traffic) can be analyzed without pre-defined knowledge by using unsupervised approaches. Also, there are similar objects in field domain. For example, a single building is composed of a number of rooms and appliances such as Electric Heat Pump (EHP), behaviors of which can be comparable by using an unsupervised analysis over corresponding data that can be obtained thanks

to environmental sensors and/or appliance sensors connected with a sensor network.

- **Virtual networks**. Also, as previously-mentioned, there has been a number of virtual networks over a single computing infrastructure (e.g., multi-tenant datacenter networking) that can be analyzed with an unsupervised analysis as well. Although not common in the current literature, we predict that a number of sensor networks will also be formed within a single field domain (e.g., isolated networks each of which connect specific different sensors/actuators within a building such as for flexible access controlling).

According to the above discussion, we predict that unsupervised analysis will be also effective in field domains. We hereafter deal with data flow of end-devices because virtual network in field domain is not widely common in the current practice.

Table 4.2: Comparison of characteristics of system behavior and data from the viewpoint of data analytics.

		characteristics from the viewpoint of data analysis	
		system behavior	data characteristics
domain	computer networking domain	closed system, and autonomous within subsystem (following standard communication protocols) → capability of interpretation with obtained data itself	traffic data: discrete (or digital), relative, and aggregated → suitability of visualizable graph-based representation and analysis
	field domain	open system, and interaction among subsystems → interpretation requires complementary information although analyses can be performed over obtained dataset (e.g., statistical misbehavior detection)	sensor data: continuous (or analog), absolute values, longitudinal characteristics (in addition to digital data) → difficulty in direct use of visualizable graph-based representation and analysis

4.2.2 Challenges: difference in characteristics of system behavior and data

In spite of the above-mentioned similarity in the concept of managements tasks and increase in virtual objects deployed over the shared infrastructure, direct application of unsupervised structural analysis to field domains is not effective due to the differences in system characteristics and correspondingly data characteristics as follows (also summarized in Table 4.2).

As the aspect of system characteristics, there are following differences in computer networking domain and field domains.

- **Computer networking domain** is relatively a closed system (i.e., traffic data is mostly explained with computer communications deriving from application programs). Also, there will be less interactions among different subsystem (in other words, the systems is more autonomous).

- **Field domain**, on the other hand, is rather an open system. For example, appliance states are affected through environmental factors (e.g., temperature) and human factors (e.g., intention of use), and resultingly attached sensors can only see the partial view of the domain. Also, field domain tends to be subject to interaction among subsystems, leading to complicated behaviors of system itself.

This comparison indicates that there are limited coverage of information viewable

from sensory data measured in field domains, compared to traffic data measured in computer networking domain, because simple set of sensory data will not enough to explain the actual condition of real fields, which triggers the increased difficulty in interpretation of analysis outputs with data in field domains.

As the aspect of the characteristics of data flows, there are followings differences in computer networking domain and field domains.

- **Traffic data** in computer networking domain would mainly be dealt with as (a) digital discrete values (e.g., TCP port numbers and flags) as well as (b) relative and aggregated view (e.g., the total number of TCP ports rather than the individual values of TCP port numbers).

- **Sensor data** in field domain would also put emphasis on (a) analog continuous values (e.g., temperature) as well as absolute and individual view (e.g., time-series behavior of absolute temperature value), although there is also digital data in field domains (e.g., on/off switch state of appliances).

In other words, from the viewpoint of the use of structural analysis, traffic data is suitable for the use of graph structure due to the commonality of discrete and relative nature (represented as nodes and edges), whereas sensory data however would not necessarily be directly suitable for that structure, which makes it difficult to introduce graph-based support of interpretation; We need expanded approach for the analysis of sensory data in field domains.

4.3 Approach and dataset

4.3.1 Approach

Based on the above-mentioned considerations (Table 4.2), we take a trial-and-error approach that performs unsupervised analysis by introducing complementary information obtained from different data sources (that are considered as structurally related) in a step-by-step manner. Starting point is the same as the analysis in the computer networking domain; We first investigate a dataset about data flow (i.e., sensory data) obtained from a single kind of objects that can be analyzed in an unsupervised inter-comparative manner. Its analysis results will provide suggestions about hypothesis about the requirements about lacking sort of information, and then we make an advanced analysis with additional dataset of different sort, which complements interpretation about the outputs.

An example of building domain dealt with in this study is as follows. We first take notice of that there are similar form of rooms in a certain kind of building (e.g., research building that accommodates a number of research groups for different rooms). Among the available dataset, the room state is characterized with sensory data measured from Electric Heat Pump (EHP), the data of which includes room temperature as well as the state of the appliances (e.g., setting temperature,

Table 4.3: Dataset used.

#	Type	Granularity	Symbol	Description	Possible values
1	EHP	Per-room	T_{set}	Target air-conditioning temperature set to EHP	Celsius degree
2	EHP	Per-room	$T_{measured}$	Temperature measured by EHP	Celsius degree
3	EHP	Per-room	S_{switch}	Switch state	ON/OFF
4	EHP	Per-room	S_{mode}	Air-conditioning mode	COOL, HEAT, FANONLY, AUTO
5	EHP	Per-room	S_{fan}	Fan volume	HIGH, MID, LOW, OFF
6	External	Geographical	$T_{external}$	External temperature	Celsius degree

on/off switch state, and air-conditioning mode). As an introductory step of analysis, we first compare the measured temperature data per rooms, and find that there are anomalies while facing the difficulty in finding keys to their reasons. We next introduce on/off switch state and setting temperature of EHP as supplementary information that will be a major factor to the measured temperatures, and find that there are anomalous events where measured temperature does not converge to the setting temperature despite the EHP is turned on. We then introduce the macroscopic external temperature as another major factor to the measured temperatures, with which we to some extent can add insight about the sources of anomalous events such as open/close states of doors/windows.

4.3.2 Field domain and dataset

In this study, we focus on the building domain. Building is a major place of productive activities of human and in other words is a major place of the use of energy resources (e.g., electricity) and appliances, which motivates the efficient management of buildings. For example, it is said that tens of percent of electricity is consumed in buildings in the U.S. case [52]. On the other hand, there has been emerging opportunities for building management with fine-grained and various sensory data through IT technologies; Sensor devices have been attached to a variety of appliances (such as ceiling lights and Electric Heat Pump (EHP)), the data of which include power consumption, running status, and setting information. Those data can be efficiently transferred over networks, stored in storages, and made available to developers through open standards of communication protocol and data format (e.g., IEEE 1888).

Table 4.3 shows the datasets used in this study. The dataset are primarily (a) EHP sensory data in a building and (b) external temperature as follows.

(a) EHP (#1 ... #5): We use data obtained from a building, which measures a variety of information about its states (e.g., power consumption of a number of appliances, setting and running states of EHP). From the dataset, we use a dataset of EHPs equipped with tens of individual rooms in the building. This dataset consists of a variety of information about EHP: setting temperature for air-conditioning

T_{set}, measured temperature $T_{measured}$, on/off state of switch S_{switch}, air-conditioning mode S_{mode}, and fan volume S_{fan}, example values of each of which is shown in Table 4.3. Measured data at time t of room R can be represented as $d(R,t) = (T_{set}(R,t), T_{measured}(R,t), S_{switch}(R,t), S_{mode}(R,t), S_{fan}(R,t))$. Those types of data are periodically measured and we analyze the data measured over the course of a one month from the entire dataset.

(b) JMA (#6): We also use data about the external environment as a source of complementary information. Specifically, we use the weather data published from Japan Meteorological Agency (JMA) [1], which includes global meteorological information such as temperature, humidity, wind, and rainfall observed with the measurement granularity of ten minutes. Among the meteorological data, we use the temperature information about the area where the building is located, as those temperature data can be considered relatively related to the EHP behaviors (e.g., EHP actively runs as cooling mode at the time of high external temperature). Measured data at time t can be represented as $T_{external}(t)$ for time point t.

4.4 Field study in building domain

4.4.1 Step 1: single-dimensional and manual anomaly detection

4.4.1.1 Data and method

As a first step of data analysis, we survey the available dataset containing diverse types of data such as electricity consumption measured at various observation points and temperature of rooms measured by EHPs. We notice that the dataset contains data for a number of EHP equipped in different rooms in the building. Those EHP data is characterized with common data types shown in Table 4.3, and can be analyzed through directly inter-compared manner due to the common data types.

By examining the types of the EHP data through visualization (e.g., Figure 4.2), we notice that the temperature data well represent the differences in the behaviors of rooms. The temperature data will be the staring point of unsupervised analysis, but that kind of data is generally continuous time-series data and we need extract related feature values in order to inter-compare the time-series data. Being inspired by the time-series analysis of network traffic in one of our previous studies [29], we analogically select to use the average and standard deviation of temperature values within a time-window; This enables to discuss the similarity and differences of room behaviors from a holistic view.

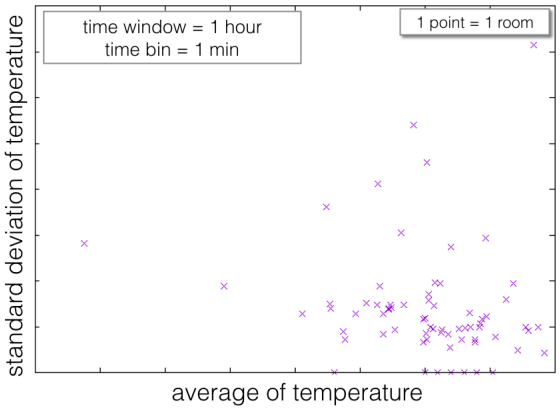

Figure 4.1: Basic scatter plot of per-room activities.

4.4.1.2 Result and considerations

Figure 4.1 shows the scatter plot of the behavior of individual rooms. The horizontal and vertical axes represent the average and the standard deviation of room temperature within the 1 hour time-window, respectively. One plotted data point corresponds to one room. This figure suggests that most of the rooms are concentrated on a certain area and whereas there are also outliers from this statistical viewpoint. The outliers include (a) low average temperature compared to the majorities as well as (b) high standard deviation of temperature compared to the majorities.

That naive scatter plot lacks the meaningful reason of the outlier behaviors. Although we can revisit the original time-series data having the outlier values of average and standard deviations, we can only discuss the characteristics of time-series behavior (e.g., high standard deviation drives from frequently fluctuating or only one large change point of baseline). We need additional information about the actual factors affecting the time-series temperature values, and among the dataset we introduce the other types of EHP-related data such as setting temperature values and on/off switch state that will affect the measured temperature values.

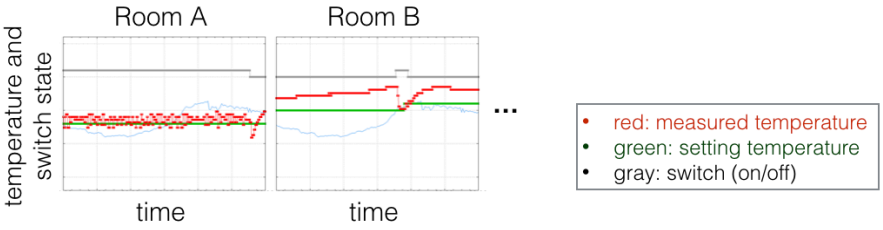

Figure 4.2: Basic plot of per-room EHP behavior.

4.4.2 Step 2: multi-dimensional and automated anomaly detection

4.4.2.1 Data

As stated above, measured temperature is dependent on other factors such as running state of EHP (i.e., switch on/off) and setting values of temperature to which the EHP controls air conditions. Based on this observation, we introduce those additional data for improving interpretation, and also we perform automated anomaly detection considering the difference between the setting temperature and measured one.

Figure 4.2 displays the measured temperature $T_{measured}$ as red lines, setting temperature T_{set} as green lines, and on/off switch states S_{switch} as gray lines (the switch state represents 'on' for upper side and 'off' for lower side of the plot). As a whole (although not accessible from this figure), there are similar values of temperature for rooms where S_{switch} is off, so that the measured temperature can be considered as dependent on the external temperature. On the other hand, measured temperature $T_{measured}$ for rooms where S_{switch} is on tends to be close to its setting values T_{set}, but there are cases where the measured temperature $T_{measured}$ does not converge to its setting temperature T_{set}. An example of sources of outlier behaviors we found through actual investigation is derived from the difference in the use of the room, i.e., there are additional heat sources (e.g., server computers).

This case study suggests the possibility of understanding room behaviors and detecting anomalies by comparing per-room activities. In particular, per-room EHP data in a single building will be effectively analyzed on the use of the difference between measured temperature $T_{measured}$ and setting temperature T_{set}. There are however still issue for improving interpretations as it requires investigations on actual locations.

[1]http://www.data.jma.go.jp/

4.4.2.2 Anomaly detection method

Based on the above survey across rooms, we further investigate the building dataset by focusing on anomaly events to make concrete discussions. Although there are choices in the definition of anomaly (e.g., changes in inter-data correlation [52] and deviation from the expected time-series behavior [53]), we focus on the temperature data as discussed above. The basic idea of our anomaly detection is simply based on the difference in the measured temperature $T_{measured}$ (i.e., actual temperature state) and setting temperature T_{set} (i.e., intended temperature state) but we note that this difference becomes informative when the switch state is on, which we hence have to deal with as well. Also, we have to be careful to the noisy behaviors such as short-term fluctuations in the switch state.

Based on the above consideration we use following algorithm of anomaly detection.

1 **Extracting EHP running event**. For each room R, we first extract the periods where EHP is activated, i.e., the periods correspond to the time spans where $S_{switch} = ON$. We ignore short-term fluctuation of S_{switch} values (considered as noisy behaviors) and we define the EHP running event as the period where the time point of $S_{switch} = ON$ (or OFF) continues more than T minutes (we set $T > 10$ minutes in this study). We note that during the $S_{switch} = ON$ period, we ignore $S_{switch} = OFF$ states continuing for $< T$ durations within the period (and we deal with the total duration as a single $S_{switch} = ON$ event), and also during the $S_{switch} = OFF$ period, we ignore $S_{switch} = ON$ states continuing for $< T$ durations within the period in the same manner. Finally, in order to remove insignificant events, we only extract events where $S_{switch} = ON$ states continues more than $\tau = 30$ minutes.

2 **Detecting EHP anomaly event**. We next identify the anomaly of the extracted EHP running events. We define the anomaly as the event where the difference between measured temperature $T_{measured}(R, t)$ and setting temperature $T_{set}(R, t)$ exceeding the threshold θ (i.e., $|T_{measured}(R, t) - T_{set}(R, t)| > \theta$) that continues more than $\tau_{anomaly} = 30$ minutes for room R ($\theta = 2.5$). We also set the timeout of anomaly as $\tau_{normal} = 60$ minutes, during which $|T_{measured}(R, t) - T_{set}(R, t)| < \theta$ continues (if we detect anomaly events after the timeout, then we deal with as different anomaly event). Those parameters (θ, T, τ, ...) can be tunable and in this study empirically determined.

Expected examples of anomalies, from the data perspective, include the case where the room temperature does not converge to the intended one within the $\pm\theta$ band within $\tau_{anomaly}$ minutes after turning on the switch (or during the period of switch on for any time). These anomalies will be related to inappropriate settings, overcapacity of rooms, measurement errors, and other unknown misbehaviors in the practical context.

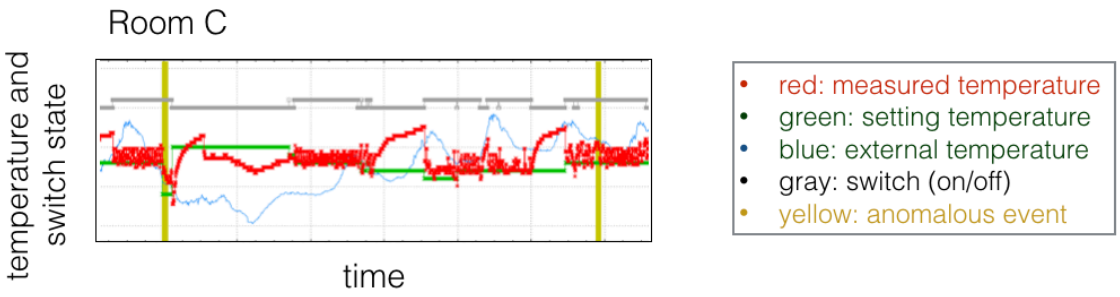

Figure 4.3: Example of anomalous events (highlighted with filled yellow area).

4.4.2.3 Result and considerations

Figure 4.3 shows part of the result of the above-mentioned anomaly detection. Anomaly events are highlighted as yellow areas in that figure, and we in total extracted more than 100 anomaly events. According to the definition of anomaly events, the behavior of detected events is more interpretable than only viewing the time-series plot of measured temperature (as performed in Sec. 4.4.1). On the other hand, we still need to improve the interpretation about the sources (or causes) of the anomaly behaviors in order to take appropriate actions. Even so, this analysis provides considerations about the candidate sources of anomalies such as inappropriate settings, overcapacity of rooms (e.g., there are significant heat sources), and inappropriate setting of rooms (e.g., doors are opens), which can be more discussed with additional dataset as conducted in the following section.

4.4.3 Step 3: complementary interpretation with external data

4.4.3.1 Data and method

We introduce additional perspectives (from available dataset) to interpret the anomaly events. Table 4.4 shows the perspectives additionally introduced for the complementary interpretation of anomaly events. The introduced perspectives are fourfold – f_1: the average of difference between measured temperature and setting one ($T_{measured} - T_{set}$) during the period of the anomaly event, f_2: the average of difference between measured temperature and external one ($T_{measured} - T_{external}$) during the period of the anomaly event, f_3: statistical mode (i.e., most frequent value) of air-conditioning mode S_{mode} during the period of the anomaly event, and f_4: statistical mode (i.e., most frequent value) of fan volume settings S_{fan} during the period of the anomaly event. The two perspectives f_1 and f_2 represent numerical values and not directly comparable to f_3 andf_4 that represent categorical values. We aggregate f_1 and f_2 values into categorical representation as '+' ($f \geq \lambda$),'-' ($f \leq -\lambda$), '~ 0'

Table 4.4: Perspectives for classifying anomalous events.

#	Perspectives	Example
f_1	$\mu(T_{measured} - T_{set})$	+
		-
		~ 0
f_2	$\mu(T_{measured} - T_{external})$	+
		-
		~ 0
f_3	$Mo(S_{mode})$	COOL
		HEAT
		FANONLY
		AUTO
f_4	$Mo(S_{fan})$	HIGH
		MID
		LOW
		OFF

$\mu(\cdot)$: average, $Mo(\cdot)$: mode (most frequent value)

Table 4.5: Classification of anomalous events found.

#	f_1	f_2	f_3	f_4	# events	label
1	-	+	COOL	HIGH	26	C_1
2	+	+	COOL	HIGH	20	U
3	+	+	AUTO	MID	17	?
4	-	+	COOL	MID	12	C_1
5	+	~ 0	COOL	HIGH	11	U
6	+	-	COOL	HIGH	11	U
7	+	+	COOL	OFF	7	I
8	-	~ 0	COOL	HIGH	6	C_3
9	-	-	COOL	HIGH	6	C_2
10	-	+	FANONLY	LOW	6	I
11	-	+	COOL	OFF	5	I
12	+	+	FANONLY	HIGH	5	I
13	-	-	COOL	LOW	3	C_2
14	-	+	FANONLY	MID	3	I
15	+	+	HEAT	HIGH	3	H
...	...	...	...	...	...	...

$(-\lambda < f < \lambda)$, where $\lambda = 1.0$ in this study. Introducing these four perspectives are motivated by the expectation of complementing the lacking information from only using EHP temperature data; For instance, lacking information (discussed in the previous step) includes open/close states of door/windows (as $T_{measured}$ will converges to $T_{external}$ in case of open states), EHP setting errors (as will be discussed with S_{mode} and S_{fan}), and heat sources in rooms (as $T_{measured}$ will not converge to T_{set} even in the closed state)

The method of anomaly analysis is based on simple grouping with the four perspectives; Anomaly events having the same values of $f_1, \ldots, f_4$ are grouped into the same cluster.

4.4.3.2 Result and considerations

Table 4.5 represents the results of grouping (or clustering) the anomaly events based on the additionally-introduced perspectives. This table shows the groups according to the cluster sizes. We manually put labels to the clusters based on our additional interpretation with the outputs. Considerations about clusters with the individual labels are as follows.

- I: The EHP can be considered as not working as temperature management. This inference is based on that the air-conditioning mode (f_3) is FANONLY, or the fan volume (f_4) is OFF. Major reason for this will be inappropriate settings of EHP, or the users of the EHP do not intend to control the temperature.
- C: The room can be considered as overcooled. This inference is based on that the air-conditioning mode (f_3) is cooling (COOL) and yet $f_1 < 0$ (i.e., measured temperature is lower than setting temperature). For the case C_1, we infer that users of the room actually intended to set cooling mode as $f_2 > 0$ (i.e., measured temperature is higher than external temperature); For the case C_2, whereas, it is uneasy to discuss because $f_2 < 0$ but one possible interpretation is that this events are triggered from inappropriate settings. The case C_3 results in $f_2 \sim 0$ (i.e., measurement temperature is similar to external one), possibly implying that the door/windows state can be open.
- H: The room can be considered as overheated. This inference is based on that the air-conditioning mode (f_3) is heating (HEAT) and yet $f_1 > 0$ (i.e., measured temperature is higher than setting temperature). There is only one case in the table, and in this case $f_2 > 0$ (i.e., measured temperature is higher than external one), possibly implying that (a) simply over-heating, or (b) inappropriately setting heating mode while intending cooling mode.
- U: The EHP can be considered as under-capacity. This inference is based on that the air-conditioning mode (f_3) is cooling (COOL) with strong fan value ($f_4 =$ HGIH) and yet $f_1 > 0$ (i.e., measured temperature is higher than setting one). Possible factors of under-capacity include the large size of room and/or the existence of other heat sources, but this is not confirmed from the current dataset (also, another possible factors is open door state in case of $f_2 \sim 0$).
- **?**: Difficult to discuss. This is because of the air-conditioning mode f_3 is automatically determined (AUTO), with which it is difficult to interpret the intention of the settings (e.g., increase the room temperature, or decrease).

4.5 Discussion

Possible use cases. Although we need further investigation, this field study will suggest that the anomaly detection of room temperatures (using data measured

by EHP) can be applied to reducing power consumption (by detecting unnecessary EHP usage), increasing maintenance efficiency (by proactively detecting abnormal behavior of EHPs), and to sophisticating asset planning (by detecting overcapacity usage of EHPs). In particular, such the use cases will be efficient for tenant-rending buildings where building managers cannot directly control the equipment of appliances (that are brought by room users) in individual rooms. Also, another possible use case of unsupervised clustering of field data is to automatically generate if-else rules for maintenance (e.g., by decision tree methods) that are conventionally enumerated by hand of special-skilled engineers. In addition, from the viewpoint of control engineering, one possible use case of field data analysis in a broader sense will be to estimate internal models of appliances, and to automatically determine parameters (e.g., thresholds) from the measured data.

Handling anomaly events. Taking appropriate actions against detected anomalies in practice require to understand the causes, for instance, by investigating the actual location where the anomalies are found. Investigators will survey open/close states of doors and windows, existence of heat sources (e.g., servers), and intentions of use of appliances (e.g., through interviewing to the users). It is, however, not easy to obtain those information although to some extent can be estimated from the available dataset as shown in this study. Also, it is not realistic to deal with all the anomaly events due to the limited human resource against the number of anomalies, and engineers need to put priorities on the events. One of the future works is to quantify the importance of anomalies, for instance, from the viewpoint of the duration of event and the amount of estimated power consumptions.

Improving interpretability. As a first step of data analytics, this field study selected to use categorical data $f_1 \dots f_4$ due to the easiness of event grouping (i.e., a form of clustering) and the ease of interpretations. Ideally, we should deal with absolute values of temperature, time of day, and duration of events, each of which can affect the characteristics of events. For the case of the dataset investigated in this study, we can further make use of power consumption of rooms light that will further complement the interpretations (e.g., to complement the information about the presence of human in night). Another future work would be to further examine the relationship between the type of additional data we complement and the type of information we can uncover.

Improving the reliability of measured data and interpretations. We also further discuss the trade-off between the reliability of data and interpretation and the cost of measurement. The temperature data we used in this study are measured from internal sensors equipped within EHP, and hence the accuracy of measured sensory data can be affected by the surrounding environment (e.g., fan volume of EHP); There might be difference between the measured temperature and the representative one of the room. Also, different from meteorological data, the sensory data tend not to be measured through a standard measurement method (e.g., standardized box for outdoor meteorological instruments). On the other hand, introducing additional

sensors to measure accurate representative data will cause additional device costs and management costs. Also, interpretation of analysis results will also rely on surrounding states such as the existence of heat sources and open/close state of rooms, measurement of which can possibly be achieved through the per-outlet power consumption (for the existence of heat sources) and the use of magnets (for the state of door and windows); This requires additional instruments, and we need to discuss the practicability of this from the viewpoint of device cost, management cost, and practical returns.

4.6 Related work

Anomaly detection in building sensor data. Anomaly detection is one of the important use cases in analyzing sensor data obtained in buildings. Ref. [54] proposes a conventional method to detecting and filtering anomalies with pre-defined if-else rules about chiller, boiler, and air handling unit (state of on/off switch, setting values, measured temperature, existence of alert), which is used as a Fault Detection and Diagnostic (FDD) tool for HVAC. Ref. [53] develops modern statistical method based on General Additive Model (GAM) applied to estimating baseline explained with external temperature and day of the week, anomalies of which are investigated as deviations from its Auto Regressive Moving Average (ARMA) model. Ref. [52] studies an unsupervised method named Strip Bind Search (SBS) that extracts anomalies of sensor data as the deviations in ordinal in-concert correlations among different sensors (e.g., correlation in power consumption of various appliances such as light and EHP), applications of which include detecting unnecessary power consumptions (e.g., leaving ceiling light turned on without the use of its rooms). Ref. [55] devises an anomaly detection method based on unsupervised clustering over feature vectors of frequency spectrum obtained from the Fourier transform on time-series data of electrical power consumptions. Our research rather focuses on dealing with multiple types of data types (not single type of data such as only examining power consumption data) to detecting anomalies as well as complementing interpretation of the outputs.

Analysis of multiple types of building sensor data. The use of multiple types of data has been studied, for instance, in above-mentioned fault detection in appliances with if-else rules [54]. Another well-studied use case is occupancy estimation, which is to estimate the number of persons within a certain area in a building. Ref. [56] develops a system of controlling HVAC appliances thorough the estimation of places of persons based on WiFi access points and authentication information about smartphones. Ref. [57] develops an occupancy estimation method that detects the direction of persons movement based on camera movie data and infrared sensor data about entrance/exit of persons. Ref. [58] proposes an occupancy estimation method based on the combination of door open/close events (measured by using magnetics) and motion events (measured with infrared sensor inside room). Ref. [59]

devises an method of estimating the number of persons in a room by analyzing CO2 density information. In addition to occupancy estimation, our research addresses the possibility of additional use cases such as unsupervised anomaly detection in sensor data.

4.7 Concluding remarks

In this chapter, we presented comparative discussions about extending the spirit of structural data analytics for field domains, especially, the building domain. Contrary to the previous chapters (Chapters 2 and 3) that dealt with traffic and configuration data in the computer networking domain, this chapter discusses the possibility of extending those analytical approaches to the field domains (e.g., building), which have become more important as more field devices become connected. We consider that the spirit of using data analytics for the management of end-hosts (by traffic analysis) and virtual network configurations (by configuration analysis) should have commonality to some extent for the management of filed domains such as to manage end-devices (e.g., sensors and actuators) and their inter-link configurations. However, analytics on data obtained in field domains is relatively difficult due to the differences in the characteristics of those domains (e.g., open systems for field domains, rather than closed systems in computer networking domain), particularly leading to the difficulty in interpretation of analytical results. In addition, characteristics of data in computing fields are rather aggregated view of discrete and relative information (i.e., aggregated cardinality), which would be suitable for the discrete structural model (e.g., graphlet), whereas field data additionally suggests the importance of individual point-by-point time-series view with non-discrete and absolute values. We discuss the need for trial-and-error approach of analyzing field data (e.g., per-room temperature in a building), interpretation of which is gradually complemented by introducing multiple types of contexts (e.g., switch states of air-conditioning appliances in rooms) and even other data sources (e.g., external temperature outside the building) that are considered as structurally related. This approach is demonstrated with an unsupervised analysis over an actual dataset about Electrical Heat Pump (EHP) equipped in a number of rooms in a building.

Chapter 5

Summary

Computer networking, particularly the Internet, has become an essential platform for human life and many aspects of industries. Such the successful growth of the computer networking, instead, insists upon the importance of its efficient management such as traffic management and configuration management. On the other hand, one of recent expectations towards efficient network management stems from the recent fashion of statistical data analytics such as Artificial Intelligence (AI), whereas one difficulty in applying statistical data analyses is derived from its low-level numerical nature differing from actual domain-specific data set. An insight is that the widespread of computer networking has been contributing to data of inter-similar objects (e.g., a number of hosts and/or a number of multiple virtual networks over the shared infrastructure), which can be compared each other to uncover knowledge (e.g., typical patterns and atypical anomalous patterns); This indicates the potential usefulness of unsupervised analyses (e.g., cluster analysis), which extract similar patterns inside data itself without established knowledge database pre-defined by human experts. However, orthodox unsupervised approaches based on extracting and clustering numerical values (i.e., feature vectors) face the difficulty in interpretation of resulting outputs, which is important in management domain in order to take appropriate actions against the outputs. In this research, we study unsupervised approaches with structural patterns (e.g., graph structure), which is more interpretable (e.g., by visualizations) than only using numerical features.

In Chapter 2, we presented structural pattern analysis on network traffic data. This analysis was demonstrated with the traffic data obtained in a measurement point in the Internet. We particularly focused on end-host profiling by analyzing network traffic, which is a major stake in traffic engineering. The use of *graphlet* for end-host traffic analysis is efficient for interpreting host behaviors, which essentially consists of a visual representation as a graph. However, graphlet analyses face the issues of choosing between supervised and unsupervised approaches. The former can analyze a priori defined behaviors but is blind to undefined classes, while the latter can discover new behavioral patterns at the cost of difficult a posteriori interpretation. This work aimed at bridging the gap between the two. First, to handle

unknown classes, unsupervised clustering was originally revisited by extracting a set of graphlet-inspired attributes for each host. Second, to recover interpretability for each resulting cluster, a *synoptic graphlet*, defined as a visual graphlet obtained by mapping from a cluster, was newly developed. Comparisons against supervised graphlet-based, port-based, and payload-based classifiers with two datasets demonstrated the effectiveness of the unsupervised clustering of graphlets and the relevance of the a posteriori interpretation through synoptic graphlets. This development was further complemented by studying *evolutionary tree* of synoptic graphlets, which quantifies the growth of graphlets when increasing the number of inspected packets per host.

In Chapter 3, we presented structural pattern analysis on network configuration data. This analysis was demonstrated with the configuration data in a multi-tenant datacenter network, where multiple customer (tenant) networks are virtualized over a single shared physical infrastructure. The use of multi-tenancy is cost-effective but poses significant costs on manual configuration. Such tasks would be alleviated with configuration templates, whereas a crucial difficulty stems from creating appropriate (i.e., reusable) ones. In this work, we proposed a graph-based method of mining configurations of existing tenants to extract their recurrent patterns that would be used as reusable templates for upcoming tenants. The effectiveness of the proposed method was demonstrated with actual configuration files obtained from a business datacenter network.

In Chapter 4, we also presented future perspectives for structural pattern analysis for the management of actual field domain. Contrary to the previous chapters (Chapters 2 and 3) that dealt with traffic and configuration data in the computer networking domain, this chapter discussed the possibility of extending those analytical approaches to the field domains (e.g., building), which have become more important as more field devices become connected. We consider that the spirit of using data analytics for the management of end-hosts (by traffic analysis) and virtual network configurations (by configuration analysis) should have commonality to some extent for the management of field domains such as to manage end-devices (e.g., sensors and actuators) and their inter-link configurations. However, analytics on data obtained in field domains is relatively difficult due to the differences in the characteristics of those domains (e.g., open systems for field domains, rather than closed systems in computer networking domain), particularly leading to the difficulty in interpretation of analytical results. In addition, characteristics of data in computer networking fields are rather aggregated view of discrete and relative information (i.e., aggregated cardinality), which would be suitable for the discrete structural model (e.g., graphlet), whereas field data additionally suggests the importance of individual point-by-point time-series view with non-discrete and absolute values. We discussed the need for trial-and-error approach of analyzing field data (e.g., per-room temperature in a building), interpretation of which is gradually complemented by introducing multiple types of contexts (e.g., switch states of

air-conditioning appliances in rooms) and even other data sources (e.g., external temperature outside the building) that are considered as structurally related. This approach was demonstrated with an unsupervised pattern analysis over an actual dataset about Electrical Heat Pump (EHP) equipped in a number of rooms in a building.

Future works include (a) investigating configuration management in field domain (e.g., building) thorugh unsupervised cluster analysis with acutual dataset, (b) establishing an efficient process (or workflow) of data-driven analysis using recent available technologies such as analysis and visualization tools, and utilimately (c) finding a key to systematic understanding about the selection of data types and analysis methods, and corresponding achievements.

Chapter 6

Publications

Publications included in this dissertation

Journal articles

- Yosuke Himura, Kensuke Fukuda, Kenjiro Cho, Pierre Borgnat, Patrice Abry, and Hiroshi Esaki, "Synoptic Graphlet: Bridging the Gap between Supervised and Unsupervised Profiling of Host-level Network Traffic," IEEE/ACM Transaction on Networking, Volume 21, Issue 4, pp.1284-1297, August 2013.

- Yosuke Himura, and Yoshiko Yasuda, "Discovering Configuration Templates of Virtualized Tenant Networks in Multi-tenancy Datacenters via Graph-mining," ACM SIGCOMM Computer Communication Review, Vol.42, No.3, pp.13-20, July 2012.

Other publications (as the first author)

Journal articles

- Yosuke Himura, and Yoshiko Yasuda , "Bridging the Gap between Tenant CMDB and Device Status in Multi-Tenant Datacenter Networking," IEICE Transactions on Communications, Vol.E98-B, No.11, pp.2132-2140, November 2015.

- Yosuke Himura, Kensuke Fukuda, Patrice Abry, Kenjiro Cho, and Hiroshi Esaki, "Characterization of Host-level Application Traffic with Multi-scale Gamma Model," IEICE Transaction on Communications, Vol.E93-B, No.11, pp.3048-3057, November 2010.

- Yosuke Himura, Kensuke Fukuda, Kenjiro Cho, and Hiroshi Esaki, "An evaluation of automatic parameter tuning of a statistics-based anomaly detection algorithm," International Journal on Network Management, Special Issue of

on "Traffic Monitoring and Network Measurements: from Theory to Practice," Volume 20, Issue 5, Wiley, pp. 295-316, August 2010.

International conferences/workshops

- Yosuke Himura, and Yoshiko Yasuda, "Static Validation of Network Device Configurations in Virtualized Multi-tenant Datacenters," IFIP/IEEE IM 2013, pp.160-168, May 2013.
- Yosuke Himura, Kensuke Fukuda, Kenjiro Cho, and Hiroshi Esaki, "An Automatic and Dynamic Parameter Tuning of a Statistics-based Anomaly Detection Algorithm," IEEE ICC 2009 CISS, pp.1003-1008, June 2009.
- Yosuke Himura, Kensuke Fukuda, Kenjiro Cho, and Hiroshi Esaki, "Quantifying Host-based Application Trafc with Multi-Scale Gamma Model," PAM2009 Student Workshop, p.2, April 2009.

Domestic conferences/workshops

- 肥村 洋輔, 福田 健介, 長 健二朗, 江崎 浩, "コネクションパターンに基づくトラフィック可視化ツール," 電子情報通信学会 技術研究報告書 IA2009-27, Vol.109, No.137, pp.33-38, 2009 年 7 月.
- 肥村 洋輔, 福田 健介, 長 健二朗, 江崎 浩, "統計的異常トラフィック検出手法の動的パラメータ最適化に関する研究," 電子情報通信学会 技術研究報告書 IN2008-106, Vol.108, No.342, pp.121-126, 2008 年 12 月.
- 肥村 洋輔, 福田 健介, 長 健二朗, 江崎 浩, "スケッチおよびガンマ関数モデルを用いたインターネットトラフィック異常検出アルゴリズムの性能評価と最適化," JSSST WIT 2008, 2008 年 6 月.
- 肥村 洋輔, 福田 健介, 長 健二朗, 江崎 浩, "スケッチおよびガンマ関数モデルを用いたインターネットトラヒック異常検出アルゴリズムの性能評価," 電子情報通信学会 総合大会, B-16-14, 2008 年 3 月.

Other publications (as a co-author)

Journal articles

- Yoji Ozawa, Yoshiko Yasuda, and Yosuke Himura, "A Tenant Network Provisioning Platform with Provisioning Template for Multi-Tenancy Data Centers," IEICE Transactions on Communications, Vol.E97-B, No.12, pp.2658-2667, December 2014.

- Guillaume Dewaele, Yosuke Himura, Pierre Borgnat, Kensuke Fukuda, Patrice Abry, Olivier Michel, Romain Fontugne, Kenjiro Cho, and Hiroshi Esaki, "Unsupervised host behavior classification from connection patterns," International Journal on Network Management, Special Issue on "Traffic Monitoring and Network Measurements: from Theory to Practice," Volume 20, Issue 5, Wiley, pp.317-337, August 2010.

- Romain Fontugne, Yosuke Himura, and Kensuke Fukuda, "Evaluation of Anomaly Detection Method based on Pattern Recognition," IEICE Transactions on Communications, Volume E93-B, Number 2, pp.328-335, February 2010.

International conferences/workshops

- Yoji Ozawa, Yoshiko Yasuda, and Yosuke Himura, "A Platform for Tenant Network Provisioning with Provisioning Template," IFIP/IEEE ManFI 2013 (short paper), pp.1167-1170, Ghent, Belgium, May 2013.

- Romain Fontugne, Yosuke Himura, and Kensuke Fukuda, "Anomaly Detection Method based on Pattern Recognition," PAM2009 Student Workshop, p.2, April 2009.

Acknowledgement

I would like to thank all the persons and organizations who supported as well as inspired my research activities. I am very fortunate to have the opportunity to doing this interesting and fruitful research activities in my life. I am deeply grateful to Dr. Hiroshi Esaki for welcoming me to his laboratory, providing the significant opportunities for the research activities in WIDE Project, and always giving invaluable advices on any aspect ranging from research details to my personal things. I would also like to sincerely thank Dr. Kensuke Fukuda for always giving insightful discussion on my researches as well as detailed instructions of research methodologies of any type ranging from data analysis and computer networking to paper reading, technical writings, international presentations, etc. I am also very thankful to Dr. Kenjiro Cho for providing the opportunities for the interesting theme of traffic analysis including the collaborative research activities with CRNS through the MAWI working group in WIDE Project. I gratefully owe all the members of the CNRS-WIDE collaboration, particularly, Dr. Partice Abry, Dr. Pierrer Borgnat, and Dr. Guillaume Dewaele, for welcoming me to the research activities of network traffic analysis including the stays in Lyon, France (starting from the time when I was the very novice at any skills). I would like to express my gratitude to all the members of the Esaki laboratory (current and alumni members), particularly Dr. Seiichi Yamamoto, Dr. Manabu Tsukada, Dr. Hideya Ochiai, and Dr. Hirochika Asai, for giving a lot of feedbacks on my research as well as teaching a lot of knowledge and skills such as computer science in general and networking managements in practice. I also would like to express my gratitude to all the members of the Fukuda laboratory, particularly Dr. Romain Fontugne, for giving specific feedbacks and discussions a lot of times. I am very grateful to the professors in charge of evaluating my Ph.D dissertation and advisory professors in my Ph.D course for providing the chance of having sophisticated discussions on my research. Also, I am significantly indebted to all the colleagues of my current and past affiliations. In particular, I am very grateful to Dr. Yoshiko Yasuda for a lot of detailed instructions and supportive feedbacks on my research activities. Finally, I would like to express my best gratitude to my family, particularly my parents, for supporting any aspect of all my activities in my life.

Bibliography

[1] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multilevel Traffic Classification in the Dark. *ACM SIGCOMM'05*, pages 229–240, 2005.

[2] M. Roesch. Snort - Lightweight Intrusion Detection for Networks. *USENIX LISA'99*, pages 229–238, 1999.

[3] l7-filter. http://l7-filter.sourceforge.net.

[4] OpenDPI. http://opendpi.org/.

[5] CAIDA The CoralReef Project. http://www.caida.org/tools/measurement/coralreef/.

[6] G. Tan, M. Poletto, J. Guttag, and F. Kaashoek. Role Classification of Hosts within Enterprise Networks Based on Connection Patterns. *2003 USENIX Annual Technical Conference*, pages 15–28, 2003.

[7] A. Lakhina, M. Crovella, and C. Diot. Mining Anomalies Using Traffic Feature Distributions. *ACM SIGCOMM'05*, pages 217–228, 2005.

[8] J. McHugh, R. McLeod, and V. Nagaonkar. Passive network forensics: behavioural classification of network hosts based on connection patterns. *ACM SIGOPS Operating Systems Review, Vol.42, No.3*, pages 99–111, 2008.

[9] I. Trestian, S. Ranjan, A. Kuzmanovic, and A. Nucci. Unconstrained Endpoint Profiling (Googling the Internet). *ACM SIGCOMM'08*, pages 279–290, 2008.

[10] K. Xu, Z. L. Zhang, and S. Bhattacharyya. Profiling Internet Backbone Traffic: Behavior Models and Applications. *ACM SIGCOMM'05*, pages 169–180, 2005.

[11] Y. Jin, E. Sharafuddin, and Z. L. Zhang. Unveiling Core Network-Wide Communication Patterns through Application Traffic Activity Graph Decomposition. *ACM SIGMETRICS'09*, pages 49–60, 2009.

[12] M. Iliofotou, M. Faloutsos, and M. Mitzenmacher. Exploiting Dynamicity in Graph-based Traffic Analysis: Techniques and Applications. *ACM CoNEXT 2009*, pages 241–252, 2009.

[13] M. Iliofotou, B. Gallagher, T. E.-Rad, G. Xie, and M. V. Faloutsos. Profiling-by-Association: A Resilient Traffic Profiling Solution for the Internet Backbone. *ACM CoNEXT 2010*, page 12, 2010.

[14] T. Karagiannis, K. Papagianakki, N. Taft, and M. Faloutsos. Profiling the End Host. *PAM 2007*, pages 186–196, 2007.

[15] S. Lee, H. Kim, D. Barman, S. Lee, C. Kim, and T. T. Kwon. NeTraMark: A Network Traffic Classification Benchmark. *ACM SIGCOMM CCR Vol.41 No.1*, pages 23–30, 2010.

[16] H. Kim, kc claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Y. Lee. Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices. *ACM CoNEXT 2008*, page 12, 2008.

[17] V. C. Espanol, P. B. Ros, M. S. Simó, A. Dainotti, W. D. Donato, and A. Pescapé. K-dimensional trees for continuous traffic classification. *TMA 2010*, page 14, 2010.

[18] Y. Lim, H. Kim, J. Jeong, C. Kim, T. T. Kwon, and Y. Choi. Internet Traffic Classification Demystified: On the Sources of the Discriminative Power. *ACM CoNEXT 2010*, page 12, 2010.

[19] A. W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. *ACM SIGMETRICS'05*, pages 50–60, 2005.

[20] M. Pietrzyk, J. L. Costeux, G. Urvoy-Keller, and T. En-Najjary. Challenging statistical classification for operational usage: the ADSL case. *ACM IMC'09*, pages 122–135, 2009.

[21] L. Bernaille, R. Teixeira, and K. Salamatian. Early Application Identification. *ACM CoNEXT 2006*, page 12, 2006.

[22] J. Erman, M. Arlitt, and A. Mahanti. Traffic Classification Using Clustering Algorithms. *ACM SIGCOMM'06 MINENET*, pages 281–286, 2006.

[23] Kuai Xu, Feng Wang, and Lin Gu. Network-Aware Behavior Clustering of Internet End Hosts. *IEEE INFOCOM 2011*, pages 2078–2086, 2011.

[24] Guillaume Dewaele, Yosuke Himura, Pierre Borgnat, Kensuke Fukuda, Patrice Abry, Olivier Michel, Romain Fontugne, Kenjiro Cho, and Hiroshi Esaki. Unsupervised host behavior classification from connection patterns. *International Journal of Network Management, Vol.10 Issue 5*, pages 317–337, 2010.

[25] MAWI Working Group Traffic Archive. http://mawi.wide.ad.jp/mawi/.

[26] K. Cho, K. Mitsuya, and A. Kato. Traffic Data Reposity at the WIDE Project. *USENIX 2000 FREENIX Track*, page 8, 2000.

[27] G. Dewaele, K. Fukuda, P. Borgnat, P. Abry, and K. Cho. Extracting Hidden Anomalies using Sketch and Non Gaussian Multiresolution Statistical Detection Procedure. *ACM SIGCOMM LSAD'07*, pages 145–152, 2007.

[28] P. Borgnat, G. Dewaele, K. Fukuda, P. Abry, and K. Cho. Seven Years and One Day: Sketching the Evolution of Internet Traffic. *IEEE INFOCOM 2009*, pages 711–719, 2009.

[29] Y. Himura, K. Fukuda, P. Abry, K. Cho, and H. Esaki. Characterization of Host-Level Application Traffic with Multi-Scale Gamma Model. *IEICE Transactions on Communications*, E93-B(11):3048–3057, 2010.

[30] L. Yu and H. Liu. Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution. *International Conference on Machine Leaning (ICML-03)*, pages 856–863, 2003.

[31] M. Iliofotou, H. C. Kim, M. Faloutsos, M. Mitzenmacher, P. Pappu, and G. Varghese. Graph-based P2P Traffic Classification at the Internet Backbone. *IEEE Global Internet Symposium 2009*, page 6, 2009.

[32] Cisco Systems Inc. Cisco Virtualized Multi-Tenant Data Center, Version 2.1 Implementation Guide.

[33] Juuniper Networks Inc. Cloud Ready Data Center Network Design Guide.

[34] A. Feldmann and J. Rexford. IP Network Configuration for Intradomain Traffic Engineering. *IEEE Network, Vol.15*, pages 46–57, September-October 2001.

[35] J. Gottlieb, A. Greenberg, J. Rexford, and J. Wang. Automated Pprovisioning of BGP customer. *IEEE Network, Vol.17*, pages 44–55, November-December 2003.

[36] D. Caldwell, A. Gilbert, J. Gottlieb, A. Greenberg, G. Hjalmtysson, and J. Rexford. The Cutting EDGE of IP Router Configuration. *ACM SIGCOMM CCR, Vol.34*, pages 21–26, January 2004.

[37] W. Enck, P. McDaniel, S. Sen, P. Sebos, S. Spoerel, A. Greenberg nad S. Rao, and W. Aiello. Configuration Management at Massive Scale: System Design and Experience. *USENIX ATC'07*, pages 73–86, June 2007.

[38] L. Vanbever, G. Pardoen, and O. Bonaventure. Towards validated network configurations with NCGuard. *IEEE INM'08*, page 6, October 2008.

[39] X. Gao, B. Xiao, D. Tao, and X. Li. A survey of graph edit distance. *Springer Pattern Analysis & Applications, Vol.13, No.1*, pages 113–129, February 2010.

[40] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys, Vol.31, No.3*, pages 264–323, September 1999.

[41] T. Benson, A. Akella, and D. Maltz. Unraveling the Complexity of Network Management. *USENIX NSDI'09*, pages 335–348, April 2009.

[42] T. Benson, A. Akella, and A. Shaikh. Demystifying Configuration Challenges and Trade-Offs in Network-based ISP Services. *ACM SIGCOMM'11*, pages 302–313, August 2011.

[43] D. Maltz, G. Xie, J. Zhan, H. Zhang, G. Hjalmtysson, and A. Greenberg. Routing Design in Operational Networks: A Look from the Inside. *ACM SIGCOMM'04*, pages 27–40, August-September 2004.

[44] Y.-W. E. Sung, C. Lund, M. Lyn, S. Rao, and S. Sen. Modeling and Understanding End-to-End Class of Service Policies in Operational Networks. *ACM SIGCOMM'09*, pages 219–230, August 2009.

[45] F. Le, G. G. Xie, D. Pei, J. Wang, and H. Zhang. Shedding Light on the Glue Logic of the Internet Routing Architecture. *ACM SIGCOMM'08*, pages 39–50, August 2008.

[46] T. Benson, A. Akella, and D. Maltz. Mining Policies From Enterprise Network Configuration. *ACM IMC'09*, pages 136–142, 2009.

[47] Y.-W. E. Sung, S. Rao, S. Sen, and S. Leggett. Extracting Network-Wide Correlated Changes from Longitudinal Configuration Data. *PAM'09*, April 2009.

[48] D. Plonka and A. J. Tack. An Analysis of Network Configuration Artifacts. *USENIX LISA'09*, page 13, November 2009.

[49] H. Kim, T. Benson, A. Akella, and N. Feamster. The Evolution of Network Configuration: A Tale of Two Compuses. *ACM IMC'11*, pages 499–512, November 2011.

[50] A. Edwards, A. Fischer, and A. Lain. Diverter: A New Approach to Networking Within Virtualized Infrastructures. *ACM WREN'09*, pages 103–110, August 2009.

[51] J. Mudigonda, P. Yalagandula, J. Mogul, B. Stiekes, and Y. Pouffary. NetLord: A Scalable Multi-Tenant Network Architecture for Virtualized Datacenters. *ACM SIGCOMM'11*, pages 62–73, August 2011.

[52] R. Fontugne, J. Ortiz, N. Tremblay, P. Borgnat, P. Flandrin, K. Fukuda, D. Culler, and H. Esaki. Strip, Bind, and Search: A Method for Identifying Abnormal Energy Consumption in Buildings. *ACM/IEEE IPSN'13*, pages 129–140, April 2013.

[53] J. Ploennigs, B. Chen, A. Schumann, and N. Brady. Exploiting Generalized Additive Models for Diagnosing Abnormal Energy Use in Buildings. *ACM BuildSys'13*, page 8, November 2013.

[54] J. Schein and S. Bushby. A hierarchical rule-based fault detection and diagnostic method for hvac systems. *HVAC&R Research*, 12(1):111–125, 2006.

[55] G. Bellala, M. Marwah, M. Arlitt, G. Lyon, and C. E. Bash. Towards an Understanding of Campus-Scale Power Consumption. *ACM BuildSys'11*, page 6, November 2011.

[56] B. Balaji, J. Xu, A. Nwokafor, R. Gupta, and Y. Agarwal. Sentinel: Occupancy Based HVAC Actuation using Existing WiFi Infrastructure within Commercial Buildings. *ACM SenSys'13*, page 14, November 2013.

[57] V. L. Erickson, S. Achleitner, and A. E. Cerpa. POEM: Power-efficient Occupancy-based Energy Management System. *ACM/IEEE IPSN'13*, pages 203–216, April 2013.

[58] Y. Agarwal, B. Balaji, S. Dutta, R. K. Gupta, and T. Weng. Duty-cycling buildings aggressively: The next frontier in HVAC control. *ACM/IEEE IPSN'11*, pages 246–257, April 2011.

[59] W. J. Fisk, D. Faulkner, and D. P. Sullivan. A Pilot study of the accuracy of CO2 sensors in commercial buildings. *IAQ 2007 Healthy and Sustainable Buildings*, page 12, October 2007.