Analysis of Communities of Interest in Data Networks

William Aiello[1], Charles Kalmanek[2], Patrick McDaniel[3],
Subhabrata Sen[2], Oliver Spatscheck[2], and Jacobus Van der Merwe[2⋆]

[1] Department of Computer Science, University of British Columbia,
Vancouver, B.C. V6T 1Z4, Canada
aiello@cs.ubc.ca
[2] AT&T Labs – Research,
Florham Park, NJ 07932, U.S.A.,
{crk,sen,spatsch,kobus}@research.att.com
[3] Department of Computer Science and Engineering, Penn State University,
University Park, PA 16802, U.S.A.
mcdaniel@cse.psu.edu

Abstract. *Communities of interest* (COI) have been applied in a variety of environments ranging from characterizing the online buying behavior of individuals to detecting fraud in telephone networks. The common thread among these applications is that the historical COI of an individual can be used to predict future behavior as well as the behavior of other members of the COI. It would clearly be beneficial if COIs can be used in the same manner to characterize and predict the behavior of hosts within a data network. In this paper, we introduce a methodology for evaluating various aspects of COIs of hosts within an IP network. In the context of this study, we broadly define a COI as a collection of interacting hosts. We apply our methodology using data collected from a large enterprise network over a eleven week period. First, we study the distributions and stability of the size of COIs. Second, we evaluate multiple heuristics to determine a stable core set of COIs and determine the stability of these sets over time. Third, we evaluate how much of the communication is not captured by these core COI sets.

1 Introduction

Data networks are growing in size and complexity. A myriad of new services, mobility, and wireless communication make managing, securing, or even understanding these networks significantly more difficult. Network management platforms and monitoring infrastructures often provide little relief in untangling the *Gordian knot* that many environments represent.

In this paper, we aim to understand how hosts communicate in data networks by studying host level *communities of interest* (COIs). A community of interest is a collection of entities that share a common goal or environment. In the context of this study, we broadly define a community of interest as a collection of interacting hosts. Using data collected from a large enterprise network, we construct community graphs representing the existence and density of host communications. Our hypothesis is that the

⋆ This research was conducted when the authors were with AT&T Labs – Research.

behavior of a collection of hosts has a great deal of regularity and structure. Once such structure is illuminated, it can be used to form parsimonious models that can become the basis of management policy. This study seeks to understand the structure and nature of communities of interest ultimately to determine if communities of interest are a good approximation of these models. If true, communities of interest will be useful for many purposes, including:

- *network management* - because of similar goals and behavior, communities will serve as natural aggregates for management
- *resource allocation* - allocating resources (e.g., printers, disk arrays, etc.) by community will increase availability and ensure inter-community fairness
- *traffic engineering* - profiles of communal behavior will aid capacity planning and inform prioritization of network resource use
- *security* - because communities behave in a consistent manner, departure from the norm may indicate malicious activity

Interactions between social communities and the Web have been widely studied [1, 2]. These works have shown that the web exhibits the *small world phenomena* [3, 4], i.e., any two points in the web are only separated by a few links. These results indicate that digital domains are often rationally structured and may be a reflection of the physical world. We hypothesize that host communication reflects similar structure and rationality, and hence can be used to inform host management. In their work in network management, Tan et. al. assumed that hosts with similar connection habits play similar roles within the network [5]. They focused on behavior within local networks by estimating host *roles*, and describe algorithms that segment a network into host role groups. The authors suggest that such groups are natural targets of aggregated management. However, these algorithms are targeted to partitioning hosts based on some *a priori* characteristic. This differs from the present work in that we seek to identify those characteristics that are relevant. Communities of interest can also expose aberrant behavior. Cortes et. al. illustrated this ability in a study of fraud in the telecommunications industry [6]. They found that people who re-subscribed under a different identity after defaulting on an account could be identified by looking at the similarity of the new account's community.

This paper extends these and many other works in social and digital communities of interest by considering their application to data networks. We begin this investigation in the following section by outlining our methodology. We develop the meaning of communities of interest in data networks and then explain how our data was collected and pre-processed. While the data set that we analyze is limited to traffic from an enterprise network, we believe that the methodology is more broadly applicable to data networks in general. In Section 3 we present the results of our analysis and conclude the paper in Section 4 with a summary and indication of future work.

2 Methodology

In this section we consider the methodology we applied to the COI study. First we develop an understanding of what COI means in the context of a data network. Then we

explain how we collected the data from an enterprise network and what pre-processing we had to perform on the data before starting our analysis.

2.1 Communities of Interest

We have informally defined COI for a data network as a collection of interacting hosts. In the broadest sense this would imply that the COI of a particular host consists of *all* hosts that it interacts with. We call the host for which we are trying to find a COI the **target-host**. We begin our analysis by exploring this broad COI definition, by looking at the total *number* of hosts that target-hosts from our data set interact with. Thus in this first step we only look at the COI set size and its stability over time.

Considering all other hosts that a target-host ever communicates with to be part of its COI might be too inclusive. For example, this would include one-time-only exchanges which should arguably not be considered part of a host's COI. Intuitively we want to consider as part of the COI the set of hosts that a target-host interact with *on a regular basis*. We call this narrower COI definition the *core* COI.

In this work it is not our goal to come up with a single core COI definition. Instead, it is our expectation that depending on the intended *application* of COI, different definitions might be relevant. For example, in a resource allocation application the relevant COI might be centered around specific protocols or applications to ensure that the COI for those applications receive adequate resources. On the other hand an intrusion detection application might be concerned about deviations from some "normal" COI. However, in order to evaluate our methodology, we do suggest and apply to our data two example definitions of a core COI:

- **Popularity**: We determine the COI for a *group* of target-hosts by considering a host to be part of the COI if the percentage of target-hosts interacting with it exceeds a threshold T, over some time period of interest Y.
- **Frequency**: A host is considered to be part of the COI of a target-host, if the target-host interacts with it at least once *every* small time-period Z (the bin-size) within some larger time period of interest Y.

Intuitively these two definitions attempt to capture two different constituents of a core COI. The most obvious is the *Frequency* COI which captures any interaction that happens frequently, for example access to a Web site containing news that gets updated frequently. The *Popularity* COI attempts to capture interactions that might happen either frequently or infrequently but is performed by a large part of the user population. An example would be access to a time-reporting server or a Web site providing travel related services.

From the COI definitions it is clear that the *Popularity* COI becomes more inclusive in terms of allowing hosts into the COI as the threshold (T) decreases. Similarly the *Frequency* COI becomes more inclusive as the bin-size increase. For the *Popularity* case where the threshold is zero, *all* hosts active in the period-of-interest are considered to be part of the COI. Similarly, for the *Frequency* case where the bin-size is equal to the period-of interest, all hosts in that period are included in the COI. When the period-of-interest, Y, is the same for the two core COI definitions, these two special cases (i.e.,

$T = 0$ for the *Popularity* COI and $Z = Y$ for the *Frequency* COI), therefore produce the same COI set.

Notice that the *Popularity* COI defines a core COI set for a "group" of hosts, whereas the *Frequency* COI defines a per-host COI. We have made our core COI definitions in the most general way by applying it to "hosts", i.e., not considering whether the host was the initiator (or client) or responder (or server) in the interaction[4]. While these general definitions hold, in practice it might be useful to take directionality into account. For example, the major servers in a network can be identified by applying the *Popularity* definition to the percentage of clients initiating connections to servers. Similarly, the *Frequency* definition can be limited to clients connecting to servers at least once in every bin-size interval to establish a per-client COI.

In the second step of our analysis we drill deeper into the per-host interactions of hosts in our data set to determine the different core COI sets. Specifically, we determine the *Popular* COI and the *Frequency* COI from a client perspective and consider their stability over time.

Ultimately we hope to be able to predict future behavior of hosts based on their COIs. We perform an initial evaluation of how well core COIs capture the future behavior of hosts. Specifically, we combine all the per-host *Client-Frequency* COIs with the shared *Popularity* COI to create an **Overall** COI. We construct this COI using data from a part of our measurement period and then evaluate how well it captures host behavior for the remainder of our data by determining how many host interactions are *not* captured by the *Overall* COI.

2.2 Data Collection and Pre-processing

To perform the analysis presented in this paper we collected eleven weeks worth of flow records from a single site in a large enterprise environment consisting of more than 400 distributed sites connected by a private IP backbone and serving a total user population in excess of 50000 users. The flow records were collected from a number of LAN switches using the Gigascope network monitor [7]. The LAN switches and Gigascope were configured to monitor *all* traffic for more than 300 hosts which included desktop machines, notebooks and lab servers. This set of monitored hosts for which we captured traffic in both directions are referred to as the **local hosts** and form the focal point of our analysis. In addition to some communication amongst themselves, the local hosts mostly communicated with other hosts in the enterprise network (referred to as **internal hosts**) as well as with hosts outside the enterprise environment (i.e., **external hosts**). We exclude communication with external hosts from our analysis as our initial focus is on intra-enterprise traffic. During the eleven week period we collected flow records corresponding to more than 4.5 TByte of network traffic. In our traces we only found TCP, UDP and ICMP traffic except for some small amount of RSVP traffic between two test machines which we ignored. For this initial analysis we also removed weekend data from our data set, thus ensuring a more consistent per-day traffic mix. Similarly, we also excluded from the analysis any hosts that were not active at least once a week during the measurement period.

[4] We provide an exact definition of client and server in the next section.

Our measurement infrastructure generated unidirectional flow-records for monitored traffic in 5 minute intervals or bins. A flow is defined using the normal 5-tuple of IP protocol type, source/destination addresses and source/destination port numbers. We record the number of bytes and number of packets for each flow. In addition, each flow record contains the start time of the 5 minute bin and timestamps for the first packet and last packet of the flow within the bin interval. The collected "raw" flow-records need to be processed in a number of ways before being used for our analysis:

Dealing with DHCP: First, because of the use of Dynamic Host Configuration Protocol (DHCP), not all IP addresses seen in our raw data are unique host identifiers. We use IP address to MAC address mappings from DHCP logs to ensure that all the flow records of each unique host are labeled with a unique identifier.

Flow-record processing: The second pre-processing step involves combining flows in different 5 minute intervals *that belong together from an application point of view*. For example, consider a File Transfer Protocol (FTP) application which transfers a very large file between two hosts. If the transfer span several 5 minute intervals then the flow records in each interval corresponding to this transfer should clearly be combined to represent the application level interaction. However, even for this simple well-known application, correctly representing the *application* semantics would in fact involve associating the FTP-control connection with the FTP-data connection, the latter of which is typically initiated from the FTP-server back to the FTP-client.

Applying such application specific knowledge to our flow-records is not feasible in general because of the sheer number of applications involved and the often undocumented nature of their interactions. We therefore make the following simplifying definition in order to turn our flows records into a data set that captures some application specific semantics. We define a **server** as any host that listens on a socket for the purpose of other hosts talking to it. Further, we define a **client** as any host that *initiates* a connection to such a server port. Clearly this definition does not perfectly capture application level semantics. For example, applying this definition to our FTP interaction, only the control connection would be correctly identified in terms of application level semantics. This client/server definition does however provide us with a very general mechanism that can correctly classify all transport level semantics while capturing some of the application level semantics.

To summarize then, during the second pre-processing step we combine or splice flow-records in two ways: First, flow-records for the same interaction that span multiple 5 minute intervals should be combined. Second, we combine two uni-directional flow-records into a single record representing client-server interaction.

To splice flow-records that span multiple 5-minute intervals, we use the 5-tuple of protocol and source/destination addresses and ports. We deal with the potential of long time intervals between matching flows by defining an *aggregation time* such that if the time gap between two flow records using the same 5-tuple exceed the aggregation time, the new flow-record is considered the start of a new interaction. If the aggregation time is too short, later flow-records between these hosts will be incorrectly classified as a new interaction. Making the aggregation time too long can introduce erroneous classification for short lived interactions. We experimented with different values of aggregation time

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.