FaRNet: Fast Recognition of High Multi-Dimensional Network Traffic Patterns

Ignasi Paredes-Oliva
UPC BarcelonaTech
Barcelona, Spain
iparedes@ac.upc.edu

Pere Barlet-Ros
UPC BarcelonaTech
Barcelona, Spain
pbarlet@ac.upc.edu

Xenofontas Dimitropoulos
ETH Zurich
Zurich, Switzerland
fontas@tik.ee.ethz.ch

ABSTRACT

Extracting knowledge from big network traffic data is a matter of foremost importance for multiple purposes ranging from trend analysis or network troubleshooting to capacity planning or traffic classification. An extremely useful approach to profile traffic is to extract and display to a network administrator the multi-dimensional hierarchical heavy hitters (HHHs) of a dataset. However, existing schemes for computing HHHs have several limitations: 1) they require significant computational overhead; 2) they do not scale to high dimensional data; and 3) they are not easily extensible. In this paper, we introduce a fundamentally new approach for extracting HHHs based on generalized frequent item-set mining (FIM), which allows to process traffic data much more efficiently and scales to much higher dimensional data than present schemes. Based on generalized FIM, we build and evaluate a traffic profiling system we call *FaRNet*. Our comparison with AutoFocus, which is the most related tool of similar nature, shows that *FaRNet* is up to three orders of magnitude faster.

Categories and Subject Descriptors: C.2.6 [Computer - Communication Networks]: Internetworking

Keywords: Network Operation and Management; Traffic Profiling; Data Mining

1. INTRODUCTION

In recent years, the Internet traffic mix has changed dramatically. Mobile applications, social networking, peer-to-peer applications and streaming services are only a few examples of the ever-growing list of applications that mold Internet traffic today. Furthermore, existing applications continuously change their behavior, while new applications, services and cyber-threats are emerging. In this rapidly changing network environment, it is critical to build traffic profiling tools that efficiently process big traffic data to extract knowledge about what is happening in a network.

AutoFocus [2], the state-of-the-art traffic profiling tool based on hierarchical heavy hitters, has some important limitations. First, its computational overhead grows exponentially with the number of dimensions. Because of this, it is restricted to 5-dimensional HHs (where the five dimensions correspond to the well-known 5-tuple) and it is very hard to extend it with additional traffic features.

SIGMETRICS'13, June 17–21, 2013, Pittsburgh, PA, USA.
ACM 978-1-4503-1900-3/13/06.

We introduce a fundamentally new approach based on generalized frequent item-set mining (FIM) [3]. Generalized FIM scales much better to higher dimensional data than AutoFocus and supports attributes of hierarchical nature, like IP addresses or geolocation data. We exploit generalized FIM to design and implement a new system, called *FaRNet* (FAst Recognition of high multi-dimensional NETwork traffic patterns), for (near) real-time profiling of network traffic data. Our system is capable of analyzing multi-dimensional traffic records with both flat and hierarchical attributes.

2. FARNET

FaRNet receives three inputs: NetFlow data, *minimum support (s)* and data treatment type. s is the threshold that determines if the size of a set of flows is big enough to be considered a frequent item-set. The next parameter indicates the type of mining: flat or hierarchical. While for the flat case the input data is considered to be completely plain, in the hierarchical scenario certain dimensions have associated hierarchies. For example, IP addresses consist of prefixes from length 8 to 32. *FaRNet* has a single output: frequent item-sets.

By default, *FaRNet* takes 10 dimensions from each input flow and builds transactions. Each transaction consists of the source and destination IP addresses, the source and destination port numbers, the protocol number, the inferred application that generated the flow, the source and destination ASes, and the geolocation of the IP addresses (continent, country, region and city). Note that although our current implementation of *FaRNet* is based on these features, any other hierarchical element could be trivially addded as the system scales well with the number of dimensions.

Depending on the selected type of mining, *FaRNet* will take different paths. For flat treatment, a FIM algorithm for flat data will be used. For hierarchical treatment, this paper presents an optimized FIM algorithm extended to deal with hierarchical traffic attributes. In particular, we first adapt, extend and optimize the implementations of different FIM algorithms (Apriori, Eclat, FP-growth, RElim and SaM) and then select the one performing the best.

The straightforward solution for allowing FIM to deal with the hierarchical nature of network traffic is expanding each element of a flow with its corresponding ancestor/s. For example, for an IP this means replacing it with all its possible prefixes from length 8 to 32, i.e., 25 items. For all 10 dimensions this accounts for 67 items per transaction.

Nonetheless, in most cases, extending e.g., all prefixes of an IP is not necessary because a fully defined 32-bit IP ad-

dress is rarely frequent by itself. However, prefixes of inferior length have higher chances of being above s. For instance, a certain prefix $a.b.0.0/16$ might be frequent even though a more specific subnet (e.g., $a.b.c.0/24$) is not. Similarly, while a city is not often frequent by itself, its corresponding region, country or continent might be.

For simplicity, from here on, all the extensions and optimizations will refer only to IP addresses. However, note that all the proposals made are applicable to the other hierarchical features presented in this paper and, in general, to any other hierarchical element.

We first present *Progressive Expansion (PE)*, which will not always generate all 25 prefixes of an IP. A prefix of length k will only be explored if its corresponding $k-1$ prefix (parent prefix or ancestor) is frequent. Otherwise, the expansion for that IP will end at level $k-1$. This is because if a certain prefix is not frequent, all prefixes of superior length will not be frequent either (*downward-closure* property [3]). The frequency of a particular prefix is calculated by progressively counting the frequency of its shorter prefixes.

The main drawback of *PE* is that it needs to go through all transactions 25 times, which is very costly in terms of runtime. Note that the number of passes is due to the depth of the IP address hierarchy and, therefore, it would change depending on the hierarchical element we are dealing with (e.g., 4 for the geolocation). In order to improve this, we propose *Progressive Expansion k-by-k (PEK)*, which seeks to reduce this part of the process while avoiding the generation of useless prefixes. This is achieved by expanding k bits at each step instead of going one by one (*PE* is a particular case of *PEK* with $k = 1$). When using *PEK*, all transactions will be read $1 + 24/k$ times instead of 25.

First, *PEK* generates all prefixes of length $l = 8$ for all IPs of all the transactions and, uniquely for these that are frequent (i.e., these that at least have s flows), a binary tree is created (only the root node). Afterwards, for each prefix of length $l + k$ with a frequent ancestor (prefix of length l, tree level $l - 8$), its corresponding tree is expanded up to level $l + k - 8$. After going through all possible values of l ($8 \leq l \leq 32$), all frequencies in intermediate nodes (nodes between explored levels, i.e., among $l - 8$ and $l + k - 8$) are recursively computed. Finally, transactions are expanded only with those prefixes that are known to be frequent by going through the corresponding tree from the root to the leaves following a depth-first approach.

The following example illustrates how *PEK* would work for IP 192.168.10.5 and $k = 2$. The first step consists of generating the binary tree for its prefix of length 8, i.e., 192/8. Afterwards, if the root node is frequent, prefixes of length 10 are generated (2-bit expansion). Therefore, frequencies for prefixes 192.192/10, 192.128/10, 192.64/10 and 192.0/10 are calculated. Then, the computation for intermediate nodes (prefixes of length 9) is calculated by moving backwards in the binary tree. In this case, prefixes 192.192/10 and 192.128/10 have a common ancestor, i.e., 192.128/9. Thus, the frequency of the intermediate node 192.128/9 is the sum of frequencies of its two descendants, 192.192/10 and 192.128/10. Likewise, the frequency for 192.0/9 comes from prefixes 192.64/10 and 192.0/10.

3. PERFORMANCE EVALUATION

For the evaluation we used NetFlow traffic from 2011 from the European backbone network of GÉANT. The dataset used is 15 minutes long and has 0.51×10^6 flows, 5.46×10^6 packets and 5.55×10^9 bytes. Figure 1 shows how *FaRNet* and AutoFocus (AF) perform for different values of s. Note that only the first 10000 flows of the dataset are used for this comparison. This is because the available implementation of AF [1] is not dimensioned to handle more flows (when it receives more than that amount, it does not count them accurately due to collisions). *FaRNet* uses *PEK* with k=12 (*PEK12*) on RElim algorithm, which holds the best tradeoff between memory consumption and execution time. In terms of runtime (right plot), *FaRNet* is clearly faster regardless of s. Moreover, as s decreases, AF's execution time increases exponentially, while *FaRNet* is able to handle it smoothly. Although for the highest s AF's runtime ($1s$) is relatively close to *FaRNet*'s ($0.10s$), for $s = 0.1\%$ AF is approximately three orders of magnitude slower ($223s$ vs $0.48s$).

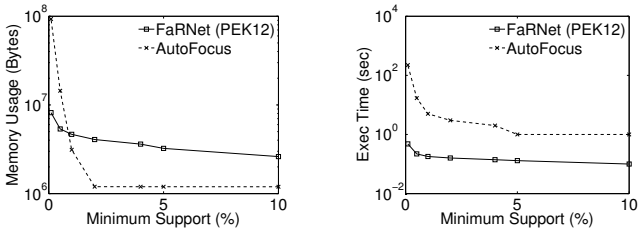

Figure 1: Memory usage (left) and execution time (right) comparison between *FaRNet* (*PEK12*) and AutoFocus for the first 10K flows of the dataset.

As regards the memory consumption (left plot), AF is better than *FaRNet* for $s \geq 1\%$. However, for lower values of s, AF consumption rises rapidly and ends up consuming far more memory than *FaRNet* (88.77 MB vs 7.79 MB for $s = 0.1\%$). All in all, *FaRNet* shows to be quicker and more resilient to low s than AF, although it uses more memory for $s = 1\%$ and above. Nonetheless, note that the memory consumption of *FaRNet* is reasonably low in the worst case (below 10 MB).

4. CONCLUSIONS

We built *FaRNet*, a network traffic profiling system that offers better performance and flexibility and also scales to a much higher number of dimensions than AutoFocus. In order to validate the correctness of *FaRNet*, we compare it with AutoFocus by using a limited version of our system configured to produce the same output. Using traffic data from a large backbone network, we show that when mining only the 5-tuple, *FaRNet* is up to three orders of magnitude quicker than AutoFocus. As a consequence, *FaRNet* is able to process high volumes of multi-dimensional traffic data in (near) real-time, while AutoFocus was designed for offline analysis of a pre-defined set of 5 dimensions.

5. REFERENCES

[1] AutoFocus implementation. http://www.caida.org/tools/.

[2] C. Estan, S. Savage, and G. Varghese. Automatically inferring patterns of resource consumption in network traffic. In *Proc. of ACM SIGCOMM*, Aug. 2003.

[3] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: current status and future directions. *Data Min. Knowl. Discov.*, 15(1):55–86, 2007.