Network Digest analysis by means of association rules

NEtwork Digest analysis by means of association rules

Daniele Apiletti, Elena Baralis, Tania Cerquitelli, Vincenzo D'Elia

Abstract— **The continuous growth in connection speed allows huge amounts of data to be transferred through a network. An important issue in this context is network traffic analysis to profile communications and detect security threats. Association rule extraction is a widely used exploratory technique which has been exploited in different contexts (e.g., network traffic characterization). However, to discover (potentially relevant) knowledge a very low support threshold needs to be enforced hence generating a large number of unmanageable rules. To address this issue in network traffic analysis, an efficient technique to reduce traffic volume is needed.**

This paper presents a NEtwork Digest framework, which performs network traffic analysis by means of data mining techniques to characterize traffic data and detect anomalies. NED exploits continuous queries to efficiently perform real-time aggregation of captured network data and supports filtering operations to further reduce traffic volume focusing on relevant data. Furthermore, NED provides an efficient algorithm to perform refinement analysis by means of association rules to discover traffic features. Extracted rules allow traffic data characterization in terms of correlation and recurrence of feature patterns. Preliminary experimental results performed on different network dumps showed the efficiency and effectiveness of the NED framework to characterize traffic data.

Index Terms— **Association rules, Continuous queries, Network traffic analysis, Stream analysis**

I. INTRODUCTION

Due to the continuous growth in network speed, terabytes of data may be transferred through a network every day. Thus, two major issues hamper network data capture and analysis: (i) A huge amount of data can be collected in a very short time (e.g., an Ethernet frame at 10Gbps is received in less than 70ns). (ii) It is hard to identify correlations and detect anomalies in real-time on such large network traffic traces. New efficient techniques able to deal with huge network traffic data need to be devised.

A significant effort has been devoted to the application of data mining techniques to network traffic analysis [6]. The application domains include studying correlations among data (e.g., association rule extraction for network traffic characterization [4], [11] or for router misconfiguration detection [14]), extracting information for prediction (e.g., multilevel traffic classification [13], Naive Bayes classification [16]), grouping network data with similar properties (e.g., clustering algorithms for intrusion detection [18], or for classification [7], [9], [21], [15]). While classification algorithms require previous knowledge of the application domain (e.g., a labeled traffic trace), association rule extraction does not. Hence, the latter is a widely used exploratory technique to highlight hidden knowledge in network flows. The extraction process is driven by enforcing a minimum frequency (i.e., support) constraint on the mined correlations. However to discover (potentially relevant) knowledge a very low support constraint has to be enforced hence generating a huge number of unmanageable rules [4]. To address this issue, a network digest representation for traffic data is needed. Continuous queries are an efficient technique to perform real-time aggregation and filtering, thus they can be exploited to effectively reduce traffic volume. A new approach jointly taking advantage of both continuous queries and association rules could efficiently perform network traffic analysis.

This leads to the proposed framework, called NED, which performs network traffic analysis by means of data mining techniques to characterize traffic data and detect anomalies. NED performs (i) On-line stream analysis to aggregate and filter network traffic, and (ii) refinement analysis to discover relationships among captured data. NED allows on-line stream analysis concurrently with data capture by means of user-defined continuous queries. This step reduces the amount of network data, thus obtaining meaningful network digests for pattern discovery. Furthermore, NED provides a refinement analysis to discover traffic features from network digests by means of association rule extraction. NED's final output is a set of association rules [12] which are able to characterize network traffic and to show correlation and recurrence of patterns among data. Preliminary experimental results performed on different network dumps showed the efficiency and effectiveness of the NED framework in characterizing traffic data and highlighting meaningful features.

The paper is organized as follows. Section II presents an overview of the NED framework. While Section III describes the network traffic stream analysis performed by means of continuous queries, Section IV presents the refinement analysis phase. In Section V experiments to validate the proposed framework are reported. Section VI draws conclusions and presents future developments of the proposed approach.

II. OVERVIEW

NED (NEtwork Digest) is a framework to efficiently perform network traffic analysis. NED addresses two main issues: (i) *Data stream processing* to reduce the amount of traffic data and allow a more effective use, both in time and space, of data analysis techniques, (ii) *Hidden knowledge extraction* from traffic data to characterize network traffic, detect anomalies, and identify recurrent patterns.

Data stream processing is performed concurrently with

D. Apiletti, E. Baralis, T. Cerquitelli, V. D'Elia are with the Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy {daniele.apiletti, elena.baralis, tania.cerquitelli, vincenzo.delia}@polito.it

data capture by means of *continuous queries*, whereas hidden knowledge is extracted from the stored continuous query results in a refinement analysis step, which currently implements an efficient *association rule mining* algorithm. Other data mining techniques [12] may be easily integrated in this step.

Continuous queries perform *aggregation* (i.e., similar records can be summarized by a proper digest) and *filtering* (i.e., meaningless data for the current analysis is discarded) of network traffic. Their results can be stored on disk for several refinement analysis sessions. Eventually, meaningful knowledge is extracted in the refinement analysis session in the form of association rules, e.g. rules which represent correlations and implications among network traffic data.

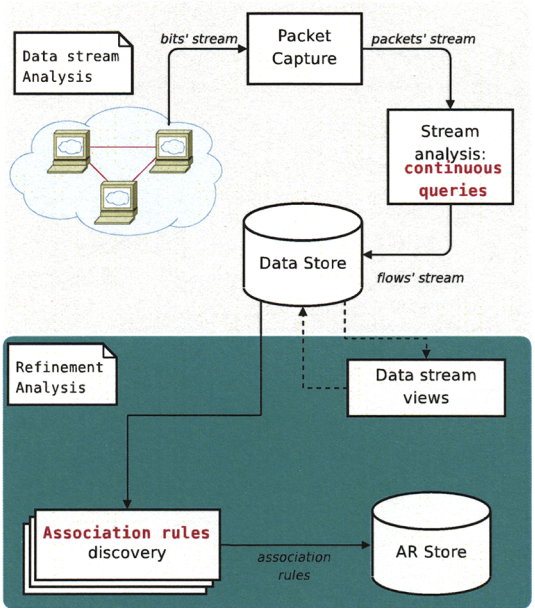


Fig. 1. NED's architecture

Figure 1 shows NED's main blocks: Data stream processing and refinement analysis. Captured traffic packets are the input stream of the continuous query block, whose objectives are (i) summarizing the traffic while preserving structural similarities among temporally contiguous packets and (ii) discarding meaningless traffic to reduce traffic volume. The output *flows* (i.e., filtered and aggregated packet digests) can be saved into a permanent data store. The storage is required only when different refinement analysis sessions need to be performed. The refinement analysis is a two step process: (i) An optional data stream view block selects a suitable user-defined subset of flows to focus the following analysis on. (ii) Association rule extraction is performed either on the data stream view, which contains the selected flows, or on all the flows in the permanent data store. The aim of the refinement analysis is to discover interesting correlations, recurrent patterns and anomalies among traffic data. Association rule analysis is currently implemented, but the framework allows different analysis techniques to be easily integrated.

To describe NED we will use a running example, which will be validated on real datasets in Section V.

III. Data Stream Processing

The data stream processing block of NED reduces the volume of traffic data by grouping similar packets and discarding irrelevant ones. Network traffic can be considered as a stream of structured data. Each packet is a record whose attributes are defined by network protocols. In our running example, the available attributes are source and destination IP addresses, source and destination TCP ports, the level 4 protocol (e.g., TCP, UDP) and the size of the packet. Since packets are captured as an unbounded stream, a conventional aggregation process would never terminate. To overcome this issue, continuous queries [3] are exploited and CQL (Continuous Query Language [2]) is used. Queries are issued once and then logically run continuously over a sliding window of the original stream. Hence, the following parameters need to be defined: (i) Aggregation and filtering rules expressed in a subset of SQL instructions, (ii) a sliding window, whose l_{ength} is expressed in seconds, which identifies the current set of data on which rules are applied, (iii) the step $s_{tep} \leq l_{ength}$, which defines how often the window moves and the output is produced. In NED a record produced as output by the continuous query is a flow, which summarizes a group of similar and temporally contiguous packets, as shown in the following examples.

Example 1: Figure 2(a) reports a toy packet capture to describe how the sample query works. The length of window l_{ength} is 6 UOT (Units Of Time) and the output is produced every 2 UOT ($s_{tep} = 2$ UOT). Figure 2(b) shows the output produced by the continuous query and how the window evolves. Some steps are not reported due to lack of space. ∎

Three types of continuous queries are implemented in NED (see Section III-A, III-B, and III-C for more details).

To improve CQL query readability, aggregation queries and filtering queries are decoupled (see Figure 3). Aggregation is performed concurrently with data capturing, while filtering can be executed both on line and off line. The packet filtering is performed in the stream analysis block and discards meaningless packets from the aggregation, whereas flow filtering is performed in the data stream view block and discards undesired flows for the specific analysis purpose.

A. Query 1

The purpose of Query 1 is to reduce the volume of traffic data while preserving information about TCP flows, their participants, their size and their fragmentation.

Timestamp	Source	Size
0	A	$size_{A,0}$
1	A	$size_{A,1}$
2	B	$size_{B,2}$
3	A	$size_{A,3}$
4	B	$size_{B,4}$
5	B	$size_{B,5}$
6	A	$size_{A,6}$
7	B	$size_{B,7}$
8	C	$size_{C,8}$

(a) A toy packet capture

Time	Current flows Source	Size	Output
0	A	$size_{A,0}$	-
1	A	$\sum_{i \in \{0,1\}} size_{A,i}$	-
2	A	$\sum_{i \in \{0,1\}} size_{A,i}$	-
	B	$size_{B,2}$	
.	.	.	.
6	A	$\sum_{i \in \{0,1,3,6\}} size_{A,i}$	$\{A, B\}$
	B	$\sum_{i \in \{2,4,5\}} size_{B,i}$	
7	A	$\sum_{i \in \{1,3,6\}} size_{A,i}$	-
	B	$\sum_{i \in \{2,4,5,7\}} size_{B,i}$	
8	A	$\sum_{i \in \{3,6\}} size_{A,i}$	$\{A, B, C\}$
	B	$\sum_{i \in \{2,4,5,7\}} size_{B,i}$	
	C	$size_{C,8}$	

(b) Flows in window

Fig. 2. Packet aggregation: $l_{ength} = 6$, $s_{tep} = 2$

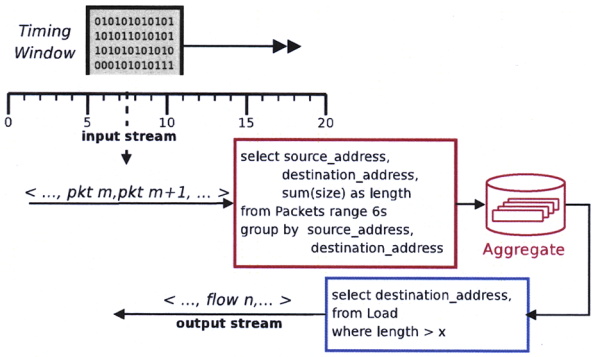


Fig. 3. Pipeline of continuous queries

Aggregate

Select	source-IP,source-Port,destination-IP, destination-Port, Sum(size) as flow-size Count(*) as packets
From	Packets [Range 6 seconds]
Where	level4 = 'TCP'
Group By	source-IP,source-Port,destination-IP, destination-Port

Filter

Select	source-IP, source-Port,destination-IP, destination-Port, flow-size
From	Aggregate

Since this query targets all TCP flows in network traffic, it does not perform any data filtering, but simply aggregates by IP source address, TCP source port, IP destination address, and TCP destination port. It also computes the total size and the number of packets of the flow.

B. Query 2

This query targets the extraction of the longest IP traffic flows. Once packets have been aggregated by source address and destination address, flows whose length is lower than a given threshold are discarded. The threshold is expressed as a percentage of the total traffic of the current window. Both filtering and aggregation considerably reduce the dataset size.

Aggregate

Select	source-IP,destination-IP, Sum(size) as flow-size Count(*) as packets
From	Packets [Range 6 seconds]
Where	level3 = 'IP'
Group By	source-IP,destination-IP

Filter

Select	source-IP, destination-IP, flow-size
From	Aggregate
Where	flow-size $\geq$ ratio * (Select Sum(flow-size) From Aggregate)

C. Query 3

This query targets the recognition of unconventional TCP traffic, which is usually exchanged on ports different from the well-known ones (i.e., port number > 1024).

Aggregate

Select	source-IP,source-Port,destination-IP, destination-Port, Sum(size) as flow-size Count(*) as packets
From	Packets [Range 6 seconds]
Where	level4 = 'TCP'
Group By	source-IP,source-Port,destination-IP, destination-Port

Port-Filtering

Select	*
From	Aggregate
Where	source-Port > 1024 And destination-Port > 1024

Size-Filtering

Select	*
From	Port-Filtering
Where	flow-size $\geq$ ratio * (Select Sum(flow-size) From Port-Filtering)

Query 3 has two filtering stages. Firstly, only flows which do not have well-known ports as source and destination are kept. Secondly, the longest flows are selected. If these two filtering stages are both performed in the continuous query, the output flows are significantly reduced, but different analysis types become unfeasible. To avoid this limitation, filters may be applied in the data stream view block.

IV. Refinement Analysis

NED discovers interesting correlations and recurrent patterns in network traffic data by means of association rule mining, which is performed in the refinement analysis phase.

Let T_{raffic} be a network traffic dataset whose generic record F_{low} is a set of F_{eature}. Each F_{eature}, also called item, is a couple (attribute, value). An attribute models a characteristic of the flow (e.g, source address, destination port). Such a T_{raffic} dataset is available in the NED data store, i.e., the input of the refinement analysis block.

Association rules identify collections of itemsets (i.e., sets of F_{eature}) that are statistically related (i.e., frequent) in

the underlying dataset. An association rule is represented in the form $X \Rightarrow Y$ where X and Y are disjoint conjunctions of F_{eature}. Rule quality is usually measured by support and confidence. Support is the percentage of items containing both X and Y. It describes the statistical relevance of a rule. Confidence is the conditional probability of finding Y given X. It describes the strength of the implication. Association rule mining is a two-step process: (i) Frequent itemset extraction and (ii) association rule generation from frequent itemsets.

Given a support threshold $s\%$, an itemset (i.e., a set of F_{eature}) is said frequent if it appears in at least $s\%$ of flows.

Example 2: Consider the toy dataset in Fig. 2(a) for the itemset mining process. With a support threshold greater than 25%, the 2-itemsets

$$\begin{array}{ll} \{\langle dest\,addr : DA1\rangle, \langle dest\,p : DP1\rangle\}, & s_{\{DA1,DP1\}} = 50\% \\ \{\langle dest\,addr : DA2\rangle, \langle dest\,p : DP1\rangle\}, & s_{\{DA2,DP1\}} = 50\% \end{array} \quad (1)$$

are frequent. Hence, the flows directed to DA2 or DA1 at port DP1 are frequent.

Once mined frequent itemset, association rules [1], [12] are used to analyze their correlations. Given the itemsets in (1), the following association rule

$$\{\langle dest\,addr : DA1\rangle\} \Rightarrow \{\langle dest\,p : DP1\rangle\} \quad \begin{cases} s\% & \text{support} \\ c\% & \text{confidence} \end{cases}$$

states that $\langle dest\,p : DP1\rangle$ appears in $c\%$ of F_{low} which contains also $\langle dest\,addr : DA1\rangle$. ∎

A. Data stream view

The data stream view block allows to select a subset of the flows obtained as continuous query output. The following example, focusing on the SYN flooding attack, shows its usefulness.

Example 3: The SYN flooding attack occurs when a victim host receives more incomplete connection requests that it can handle. To make this attack more difficult to detect, the source host randomizes the source IP address of the packets used in the attack. An attempt of SYN flooding [10] can be recognized by mining rules in the form

$$\{\text{ victim-IP, victim-port}\} \Rightarrow \text{size} \quad \begin{cases} s\% & \text{support} \\ c\% & \text{confidence} \end{cases}$$

where the left term is the victim fingerprint, while the right part is the size of the flow which is supposed to be very small. The frequency of this pattern is measured by means of support s and confidence c.

Suppose that, to reduce the amount of stored data, the network traffic has been aggregated with respect to address and port of both source and destination. For each flow the size is computed (e.g. packets differing in one of these features belong to different flows). This step is performed by running the following continuous query on the data stream.

Aggregate

Select	source-IP,source-Port,destination-IP, destination-Port, Sum(size) as flow-size
From	Packets [Range 6 seconds]
Where	level4 = 'TCP'
Group By	source-IP,source-Port,destination-IP, destination-Port

Since the complete dataset contains hundreds of flows, the support of the SYN-flooding rule may be too low to be relevant. To overcome this issue, the output of the previous continuous query may be appropriately filtered. Since we are interested in flows whose size is lower than a threshold x expressed in bytes, the following query may be exploited to create a data stream view.

Filter

Select	source-IP, source-Port,destination-IP, destination-Port, flow-size
From	Aggregate
Where	flow-size < x

The refinement analysis, performed on the results of the described data stream view, extracts a small number of association rules characterized by high support. These rules highlight more effectively any specific traffic behavior. ∎

V. Experimental Validation

A set of preliminary experiments have been performed by analyzing NED behavior on real datasets. We assessed (i) the number of extracted association rules and (ii) the interest of mined rules.

TABLE I
NETWORK TRAFFIC DATASETS

ID	Number of Packets	Size [Mbyte]
A	25969389	2621
B	24763699	2500
C	26023835	2625

Three real datasets have been obtained by performing different capture stages using the Analyzer traffic tool [17] on a backbone link of our campus network. We will refer to each dataset using the ID shown in Table I, where the number of packets and the size of each dataset is also reported.

Experiments target the execution of the queries discussed in Section III. They have been performed by considering three window lengths (60 s, 120 s, and 180 s) and two link speeds (10 Mbps and 100 Mbps). The value of the window step s_{tep} has been set to $\frac{\text{window length}}{2}$. Due to lack of space, reported results refer to experiments performed with 100 Mbps link speed and 60 s window length. The ratio parameter of Queries 2 and 3 has been set to 0.1.

To avoid discarding packets, a proper buffer size has to be determined. The buffer must be able to store all possible flows in a time window, whose worst case value is the maximum number of captured packets (i.e., each packet belongs to a different flow). Thus, the buffer size has been set to the following number of flows.

$$size = \frac{\text{link speed} * \text{window length}}{\text{minimum size of a packet}}$$

Frequent itemset extraction is based on the LCM v.2 algorithm [20] (FIMI'04 best implementation algorithm), while association rule generation is performed using Goethal's implementation of the Apriori algorithm [5].

Experiments have been performed on a 2800 Mhz Pentium IV PC with 2 Gb main memory running Linux (kernel 2.7.81). All reported execution times are real times, including both system and user times. They have been obtained using the Linux *time* command as in [8].

A. Query 1

Query 1 (Section III-A) aggregates packets with respect to source address, source port, destination address and destination port. Thus, it significantly reduces the data cardinality, while preserving general traffic features.

Figure 4(a) reports the number of extracted rules for each dataset considering different support and confidence thresholds. Since the three datasets have similar behavior, we focus on dataset A, where we observe that some 1-itemsets are highly frequent, such as $\langle source-address : 130.192.a.b \rangle$ and $\langle destination-address : 130.192.a.b \rangle$ [1] To further investigate the meaning of the rules, we consider the following examples.

Example 4: Considering minimum support $s \geq 0.1\%$ and minimum confidence $c \geq 50\%$ leads to the extraction of a large amount of rules in the following form.

$$\{\langle \text{*-port} : x \rangle\}$$
$$\Downarrow$$
$$\{\langle \text{*-address} : 130.192.a.b \rangle\}$$

Since port x is frequent (regardless of the port number), these rules state that the address 130.192.a.b (i) generates remarkable traffic both as receiver and as transmitter, (ii) it is likely to be a server which provides many services, because it uses a wide range of ports. We can conclude that 130.192.a.b is probably the public IP address of a router implementing NAT. An inspection of the network topology confirms such result. ∎

Figure 4(a) reports the number of extracted rules when varying support and confidence thresholds. For Query 1, as reported in Figure 4(a), by enforcing high support and confidence thresholds, the number of extracted patterns decreases. The decreasing trend is particularly evident for high support thresholds, whereas most of the rules have high confidence values for any support threshold. Thus, in this scenario, support is more selective in filtering patterns.

Example 5: By setting the minimum support to 0.3% and the minimum confidence to 0.5%, some interesting patterns are extracted. NAT rules are still present, and other rules become more evident. For example

$$\{\langle \text{source-address} : 130.192.c.d \rangle\}$$
$$\Downarrow$$
$$\{\langle \text{source-port} : 443 \rangle\}$$
$$\begin{cases} s = 0.3\% \\ c = 99\% \end{cases}$$

identifies 130.192.c.d as an `https` server. It was confirmed to be the student webmail server.

$$\{\langle \text{destination-port} : 6101 \rangle \langle \text{flow-size} : 96 \rangle\}$$
$$\Downarrow$$
$$\{\langle \text{source-address} : x.y.z.w \rangle\}$$
$$\begin{cases} s = 0.3\% \\ c = 98\% \end{cases}$$

highlights that Synchronet-rtc service is frequently and mostly used by $x.y.z.w$. ∎

Analyses performed on rules extracted from datasets B and C confirm the results obtained on dataset A. The traffic network features, inferred from the rules, highlight the same NAT routing and servers.

To identify patterns arising from long flows, another step of filtering is required. This issue, addressed by the second query, has been analyzed in the next section.

B. Query 2

The second query (Section III-B) selects the flows which generate an amount of traffic greater than a certain percentage of the total traffic in a window. The aim is to describe more accurately rules extracted by Query 1.

Figure 4(b) shows the number of association rules extracted from the results of Query 2 applied to the datasets A, B, and C. Rules discovered in this case predominantly have the following form.

$$\{\langle source\,address : SA \rangle, \langle source\,port : SP \rangle\}$$
$$\Downarrow$$
$$\{\langle destination\,address : DA \rangle, \langle destination\,port : DP \rangle\}$$

Many extracted rules describe general network features such as NAT routing or main servers. Furthermore, this analysis assesses the pervasiveness of different services. Mined rules highlight the importance of several protocols like `netrjs`, `systat` and `QMTP` in the examined network. Some rules are worth further investigation. Many flows have source and destination ports > 1024. This fact may highlight unconventional traffic, such as *peer-to-peer* communications. Another filtering step is necessary to clearly identify involved hosts. This issue has been addressed by Query 3.

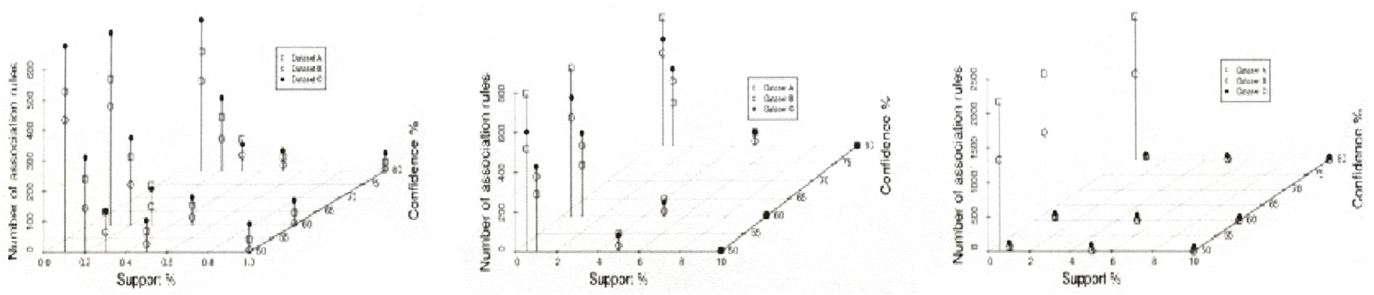
C. Query 3

The third query (Section III-C) extracts long flows whose source and destination ports are beyond 1024.

Figure 4(c) shows the number of association rules extracted from the result of Query 3 applied to datasets A, B, and C. Because of the additional filtering step, the number of rules is significantly lower than the ones extracted by Query 1 and Query 2. Furthermore, these rules are even more specific than previous ones, as shown by the following example.

Example 6: Consider the following rule.

[1] The actual IP addresses have been masked for privacy reasons.



(a) Query 1 (b) Query 2 (c) Query 3

Fig. 4. Number of extracted association rules

$$\{\langle \text{source-address} : 130.192.e.f \rangle\}$$
$$\Downarrow$$
$$\{\langle \text{destination-port} : 4662 \rangle\}$$
$$\begin{cases} s = 1.98\% \\ c = 77\% \end{cases}$$

The address 130.192.e.f is identified as having a remarkable amount of traffic toward remote hosts on port 4662. Since this is the default port for eDonkey2000 servers [19] of the ED2K peer to peer network, we can conclude that (i) the source host is exchanging data with the ED2K servers, and (ii) its amount of traffic on not well-known ports is mainly related to such peer to peer network. ∎

VI. Conclusions and Future Work

NeD is a framework to efficiently perform network traffic analysis. NeD provides techniques to perform data stream analysis and refinement analysis. The former reduces the amount of traffic data while the latter automatically extracts correlation and recurrence of patterns among traffic data.

Association rule extraction algorithms do not easily detect patterns nested in rare flows, even if they might be relevant. Hence future developments of the NeD framework will explore generalized association rules. Generalized association rules could represent network traffic at a higher level of abstraction and could become a more powerful tool to efficiently extract hidden knowledge not captured by traditional approaches.

VII. Acknowledgment

We are grateful to Fulvio Risso for providing the real traffic datasets captured from the campus network.

References

[1] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, 1994.

[2] A. Arasu, S. Babu, and J. Widom. The CQL continuous query language: semantic foundations and query execution. *The VLDB Journal The International Journal on Very Large Data Bases*, 15(2):121–142, 2006.

[3] S. Babu and J. Widom. Continuous queries over data streams. *ACM SIGMOD Record*, 30(3):109–120, 2001.

[4] M. Baldi, E. Baralis, F. Risso, and D. e Inf. Data mining techniques for effective and scalable traffic analysis. *Integrated Network Management, 2005. IM 2005. 2005 9th IFIP/IEEE International Symposium on*, pages 105–118, 2005.

[5] Bart Goethals. Frequent Pattern Mining Implementations. Available at http://www.adrem.ua.ac.be/~goethals/software.

[6] K. Burn-Thornton, J. Garibaldi, and A. Mahdi. Pro-active network management using data mining. *Global Telecommunications Conference, 1998. GLOBECOM 98.*, 2, 1998.

[7] J. Erman, M. Arlitt, and A. Mahanti. Traffic classification using clustering algorithms. In *MineNet '06*, pages 281–286, New York, NY, USA, 2006. ACM Press.

[8] FIMI. http://fimi.cs.helsinki.fi/.

[9] Y. Guan, A. Ghorbani, and N. Belacel. Y-Means: A clustering method for intrusion detection. *Proceedings of Canadian Conference on Electrical and Computer Engineering*, pages 4–7, 2003.

[10] B. Harris and R. Hunt. TCP/IP security threats and attack methods. *Computer Communications*, 22(10):885–897, 1999.

[11] M. Hossain, S. Bridges, and R. Vaughn Jr. Adaptive intrusion detection with data mining. *IEEE Internation Conference on Systems, Man and Cybernetics*, 4, 2003.

[12] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Series Editor Morgan Kaufmann Publishers. The Morgan Kaufmann Series in Data Management Systems, Jim Gray, August 2000.

[13] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. Blinc: multilevel traffic classification in the dark. In *SIGCOMM*, pages 229–240, 2005.

[14] F. Le, S. Lee, T. Wong, H. S. Kim, and D. Newcomb. Minerals: using data mining to detect router misconfigurations. In *MineNet '06*, pages 293–298, New York, NY, USA, 2006. ACM Press.

[15] W. Lee and S. Stolfo. A framework for constructin features and models for intrusion detection systems. *ACM Transactions on Information and System Security (TISSEC)*, 3(4):227–261, 2000.

[16] A. W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In *SIGMETRICS '05*, pages 50–60, New York, NY, USA, 2005. ACM Press.

[17] NetGroup, Politecnico di Torino. Analyzer 3.0. Available at http://analyzer.polito.it/30alpha/.

[18] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion detection with unlabeled data using clustering. *Proceedings of ACM CSS Workshop on Data Mining Applied to Security, PA,, November*, 2001.

[19] The SANS Institute. Port 4662 details. Available at http://isc.sans.org/port.html?port=4662.

[20] T. Uno, M. Kiyomi, and H. Arimura. LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In *FIMI*, 2004.

[21] Q. Wang and V. Megalooikonomu. A clustering algorithm for intrusion detection. *Proc. SPIE*, 5812:31–38, 2005.