Characterizing network traffic by means of the NetMine framework

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

Politecnico di Torino, Dipartimento di Automatica e Informatica, Corso Duca degli Abruzzi, 24, 10129 Torino, Italy

A R T I C L E   I N F O

Article history:
Available online 29 December 2008

Keywords:
Network traffic characterization
Network data analysis
Generalized association rules

A B S T R A C T

The NetMine framework allows the characterization of traffic data by means of data mining techniques. NetMine performs generalized association rule extraction to profile communications, detect anomalies, and identify recurrent patterns. Association rule extraction is a widely used exploratory technique to discover hidden correlations among data. However, it is usually driven by frequency constraints on the extracted correlations. Hence, it entails (i) generating a huge number of rules which are difficult to analyze, or (ii) pruning rare itemsets even if their hidden knowledge might be relevant. To overcome these issues, NetMine exploits a novel algorithm to efficiently extract generalized association rules, which provide a high level abstraction of the network traffic and allows the discovery of unexpected and more interesting traffic rules. The proposed technique exploits (user provided) taxonomies to drive the pruning phase of the extraction process. Extracted correlations are automatically aggregated in more general association rules according to a frequency threshold. Eventually, extracted rules are classified into groups according to their semantic meaning, thus allowing a domain expert to focus on the most relevant patterns. Experiments performed on different network dumps showed the efficiency and effectiveness of the NetMine framework to characterize traffic data.

1. 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 10 Gbps is received in less than 70 ns). (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 [9]. The application domains include studying correlations among data (e.g., association rule extraction for network traffic characterization [5,17] or for router misconfiguration detection [22], extracting information for prediction (e.g., multilevel traffic classification [19], Naive Bayes classification [27]), grouping network data with similar properties (e.g., clustering algorithms for intrusion detection [32], or for classification [12,13,36]), and context specific applications (e.g., multi-level association rules in spatial databases [23]). Data mining techniques play an important role in intrusion detection systems [34,30], where association rules are successfully exploited for anomaly identification [31].

While supervised 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 which may be exploited 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 [5]. To address this issue, a higher level abstraction than traditional association rules and a network summarized representation for traffic data are needed.

This work presents a framework, called NetMine, which performs network traffic analysis by means of data mining techniques to characterize traffic data and detect anomalies. NetMine performs (i) on-line stream analysis to aggregate and filter network traffic, (ii) refinement analysis to discover relationships among captured data, and (iii) rule classification into different semantic groups. NetMine allows on-line stream analysis concurrently with data capture by means of user-defined continuous queries. “Continuous queries” [4] is an efficient technique to perform real-time aggregation and filtering; thus it allows an effective reduction of the amount of network data to be analyzed. The result of this step is a meaningful summary of network traffic data appropriate for pattern discovery. NetMine performs a refinement analysis to discover traffic features from network data summaries by means of generalized association rule extraction. Differently from previous approaches [15], NetMine generalized rule extraction is support driven, i.e., rules are generalized only when lower levels in the taxonomy are below a frequency threshold. Thus, in NetMine generalization is automatically performed only when needed. Finally, extracted rules may be classified in several semantic categories, depending on the traffic features they observe. NetMine’s final output is a set of categorized and generalized rules [18] which are able to characterize network traffic and to show correlation and recurrence of patterns among data. Experiments performed on different network dumps showed the efficiency and effectiveness of the NetMine framework in characterizing traffic data and highlighting meaningful features.

The paper is organized as follows. Section 2 presents an overview of the NetMine framework and the main features of its building blocks. In Section 3 experiments to validate the proposed framework are reported. Section 4 discusses related works, while Section 5 draws conclusions.

1.1. Motivating example

A traffic capture process produces a trace which holds information on the stream of packets. A trace is composed of a set of records, each of which is a set of tagged items (also denoted as itemset in the following). While items are values captured from the net (e.g., the IP address 130.192.3.17), their tags are the description of the represented information (e.g., the label IP destination address).

Relevant patterns may be represented by means of association rules. An association rule is represented in the form \( X \rightarrow Y \), where \( X \) and \( Y \) are disjoint conjunctions of tagged items. Each rule is usually characterized by the support and confidence quality index [18]. The support is the prior probability of \( X \) and \( Y \) (i.e., its observed frequency in the data set). The confidence is the conditional probability of \( Y \) given \( X \) and characterizes the “strength” of a rule. The traditional association rule mining problem can be described as follows. Given a database of transactions (in our case a network trace), a minimum confidence threshold and a minimum support threshold, find all association rules whose confidence and support are above given thresholds. However, the following example highlights the need of a more powerful abstraction of association rules.

Consider a web server on port 80 having address 130.192.5.7. To describe the activity of a client connecting to this server, a rule in the form

\[
\{ \text{source address: } 142.134.4.5 \} \quad \downarrow \quad \{ \text{destination address: } 130.192.5.7, \text{destination port: } 80 \} \quad (s\%, c\%)
\]

may be extracted, where \((s\%, c\%)\) are support and confidence values. Since a single source address is a 1-itemset which is infrequent in a very large traffic network trace, extracting such rule would require enforcing a very low support threshold, which makes the task infeasible. However, a higher level view of the network may be provided by the following generalized association rule:

\[
\{ \text{source address: } 142.134.4.0/17 \} \quad \downarrow \quad \{ \text{destination address: } 130.192.5.7, \text{destination port: } 80 \} \quad (s\%, c\%)
\]

which shows a subnet generating most of the traffic. This generalized rule may provide valuable knowledge for network monitoring. The number of different tagged items in network traffic may be very large (e.g., different IP addresses) and association rules on single tagged items provide excessively detailed and hardly usable knowledge. Generalized association rules allow raising the abstraction level at which correlations are represented. Hence, they are a powerful tool to address this issue.

2. The NetMine framework

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

Fig. 1 shows NetMine main blocks: data stream processing, refinement analysis, and rule classification. Traffic packets, captured by means of available network capture tools [35,28], are the input data of the stream processing block, whose objectives are (i) summarizing the traffic while preserving structural similarities among temporally contiguous packets and (ii) discarding irrelevant traffic to reduce traffic volume. Data stream processing is performed concurrently with data capture by means of continuous queries, which perform aggregation (i.e., similar records can be summarized by a proper summary) and filtering (i.e., data irrelevant for the current analysis is discarded).
of network traffic. The output flows (i.e., filtered and aggregated packet summaries) may be saved in a permanent data store. Storage is required only when different refinement analysis sessions need to be performed.

The aim of the refinement analysis is to discover interesting correlations, recurrent patterns and anomalies in traffic data. Currently, interesting patterns are extracted in the form of generalized association rules, i.e., rules which represent general correlations among network traffic data. But the framework allows different data mining techniques [18] to be easily integrated. 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) Generalized association rules mining is performed either on the data stream view, which contains the selected flows, or on all the flows in the permanent data store. Generalized association rule mining is implemented by means of the novel Genio algorithm [6]. The Genio algorithm is more effective and efficient than previous approaches [15] in automatically extracting interesting generalized rules from structured data.

The rule classification phase organizes the extracted rules according to their semantic interpretation in the network domain. Hence, it allows a network operator to focus on relevant features of the captured data without prior knowledge of the desired network patterns. All the building blocks of the NetMine framework are described in more detail in the following subsections.

2.1. Data stream processing

The data stream processing block of NetMine 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, where each captured packet is a record whose attributes (i.e., tags) are defined by network protocols. Each record is characterized by at most one value for each tag. More formal definitions follows.

Definition 1. Tagged item. Let $\mathcal{I} = \{t_1, t_2, \ldots, t_n\}$ be a set of tags which describe network protocol attributes. A tagged item $t_i = \text{item}$ assigns the value $\text{item}$ to the network protocol attribute $t_i$, where $\text{item}$ belongs to the domain of attribute $t_i$.

For example, in this context, relevant tags are source and destination addresses, source and destination ports, the level 3 and level 4 protocols (e.g., TCP, UDP) and the size of the packet.

Definition 2. Itemset. Let $\mathcal{I} = \{t_1 = \text{item}_1, t_2 = \text{item}_2, \ldots, t_n = \text{item}_n\}$ be the set enumerating all tagged items. An itemset $X \subseteq \mathcal{I}$ is a set of tagged items.

Definition 3. Network trace. Let $\mathcal{I} = \{t_1, t_2, \ldots, t_n\}$ be a set of tags which describe network protocol attributes and $\mathcal{I} = \{t_1 = \text{item}_1, t_2 = \text{item}_2, \ldots, t_n = \text{item}_n\}$ the corresponding set enumerating all tagged items. A network trace is a collection of records, where each record $r$ is an itemset $X \subseteq \mathcal{I}$. Each tag in $\mathcal{I}$ may occur at most once in any itemset $X$.

Since packets are captured as an unbounded stream, conventional query processing would never terminate. To overcome this issue, continuous queries [4], i.e., a set of instructions to process a subset of the stream, are exploited. Continuous query languages have been thoroughly addressed in [2], where CQL (Continuous Query Language [2]) is proposed to express continuous queries. Several approaches have been proposed to efficiently evaluate continuous queries over streaming data, among which NiagaraCQ [10] for internet databases, STREAM [4] for different streaming applications (e.g., network monitoring, telecommunications data management, publish-subscribe, web personalization), and Telegraph CACQ [24] for sensor databases. Data stream processing in NetMine is based on the continuous queries presented in [4].

Continuous 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 the SQL language instruction set, (ii) a sliding window, whose $\text{length}$ is expressed in seconds, which identifies the current set of data on which rules are applied, (iii) the step $\text{step} \leq \text{length}$, which defines how often the window moves and the output is produced. In NetMine a record produced as output by the continuous query is a flow, which summarizes a group of similar and temporally contiguous packets.

In NetMine, continuous queries have been decoupled in aggregation queries and filtering queries. Aggregation is performed on the incoming packet stream by the stream processing block. The same block also performs packet filtering, hence discarding uninteresting packets from the aggregation. Flow filtering is performed in the data stream view block and allows discarding undesired flows for the specific analysis purpose. A sample continuous query implemented in NetMine is the following:

Sample query. This query targets the extraction of the biggest IP traffic flows. Once packets have been aggregated...
by source and destination addresses, and source and destination ports, flows whose size 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. The query is expressed by means of CQL, whose most relevant clauses are introduced by examples referencing the language constructs in the query.

### Aggregate

| Select | source-address, destination-address, source-port, destination-port  
| Sum(size) as flow-size  
| Count( ) as packets  
| From | Packets [Range by 60 s]  
| Where | level3 = 'IP'  
| Group | source-address, destination-address, source-port, destination-port  
| Filter | source-address, destination-address, source-port, destination-port, flow-size  
| From | Aggregate  
| Where | flow-size > ratio *  

(Select Sum(flow-size) From Aggregate)

**Select clause.** It defines the names of the tags to be extracted from the incoming stream (e.g., source-address, destination-address, source-port, destination-port in the sample query) and the list of aggregate operations to be performed (e.g., Sum(size) returns the total size of selected flows, while Count( ) the number of packets in the sample query).

**From clause.** It introduces the name of the stream to be queried (e.g., the Packets network trace in the sample query). The stream name may be followed by the time period which defines how often the window moves and the output is produced, introduced by the Range by keywords (e.g., the time period is 60 s in the sample query).

**Where clause.** It expresses the conditions which should be satisfied by the incoming stream records to be selected. The conditions are expressed as a boolean combination of predicates of the form expression op expression, where op is one of the comparison operators {<, <=, >, >=, =}. An expression is a tag name, a constant, or an (arithmetic or string) expression (e.g., level3 = 'IP' in the sample query).

**Group by clause.** It defines the list of tag names with respect to which the aggregation operation needs to be performed (e.g., source-address, destination-address, source-port, and destination-port in the sample query). Records are grouped together when, for all the attributes in the list, they share the same value (e.g., a group might be formed by records sharing the values source-address = 130.192.1.1, destination-address = 130.192.254.2, source-port = 32,100, destination-port = 80). The aggregate functions specified in the select clause are computed separately for each group.

#### 2.2. Refinement analysis

Given a network trace stored in the data store, NetMine extracts association rules, which provide an abstract representation of interesting correlations and recurrent patterns in data.

**Definition 4.** Association rule. Let \( \mathcal{F} = \{ t_1, t_2, \ldots, t_n \} \) be a set of tagged items in a network trace. An association rule is represented in the form \( X \Rightarrow Y \), where \( X \) and \( Y \) (the body and the head of the rule, respectively) are disjoint conjunctions of tagged items \( (X \subseteq \mathcal{F}, Y \subseteq \mathcal{F}, X \cap Y = \emptyset) \).

A rule template defines the general structure of a set of association rules. All rules with the same template are characterized by the same tags (but not necessarily the same values) in the body and in the head of the rule.

**Definition 5.** Association rule template. Let \( \mathcal{T} = \{ t_1, t_2, \ldots, t_n \} \) be a set of tags in a network trace. An association rule template is represented in the form \( X \Rightarrow Y \), where \( X \) and \( Y \) (the body and the head of the rule template, respectively) are disjoint conjunctions of tags \( (X \subseteq \mathcal{T}, Y \subseteq \mathcal{T}, X \cap Y = \emptyset) \).

Each rule is characterized by the support (Definition 7) and confidence (Definition 8) quality indices [18].

**Definition 6.** Itemset support. Let \( X \) be an itemset. The support of \( X \) is the prior probability of \( X \), i.e., the proportion of records including \( X \) to all the records in the network trace.

**Definition 7.** Association rule support. Let \( X \Rightarrow Y \) be an association rule. Its support \( s(X \Rightarrow Y) \) is the support of the itemset \( X \cup Y \).

**Definition 8.** Confidence. Let \( X \Rightarrow Y \) be an association rule. Its confidence \( c(X \Rightarrow Y) \) is the conditional probability of \( Y \) given \( X \), i.e., the proportion of records including both \( X \) and \( Y \) to all the records including \( X \).

The traditional association rule mining problem can be described as follows. Given a database of transactions (in our case a network trace), a minimum support threshold and a minimum confidence threshold, find all association rules whose confidence and support are above given thresholds. However, such thresholds in traditional association rules may prevent relevant knowledge from being highlighted, and enforcing very low thresholds may make the task infeasible. Hence, a more powerful abstraction is needed for association rules. Generalized association rules are a powerful tool to address this issue. They extend the notion of multi-level association rules [15,20], which provide a higher level domain abstraction. Generalized association rule extraction requires the definition of a taxonomy, which describes a hierarchy among tag items.

**Definition 9.** Taxonomy. Let \( t_i \) be a tag (i.e., a network protocol attribute) and \( \Omega_i \) its corresponding value domain. A taxonomy \( T_i \) is a pre-determined hierarchy of aggregations over values in \( \Omega_i \). \( T_i \) is a tree whose leaves are values in \( \Omega_i \). Each non-leaf node in the tree is an aggregation of its children, which may be further generalized by its father. Children nodes are mutually exclusive and collectively exhaustive.

Fig. 2a shows a taxonomy for the source-address, or destination-address tags. Source-addresses are aggregated with
respect to their subnet. Fig. 2b shows a taxonomy for the source-port or destination-port tags. source-ports may belong to the well known class if lower than 1024, registered if in the range 1024-49151, dynamic otherwise.

**Definition 10.** Generalized association rules. Let \( \mathcal{E} = \{ t_1 = \text{expression}_1, t_2 = \text{expression}_2, \ldots, t_n = \text{expression}_n \} \) be a set of tagged items. \( \text{expression}_i \) is either (a) a value \( v_i \) in the domain \( \Omega_i \) of network protocol tag \( t_i \), or (b) an aggregation value in the taxonomy \( T_i \) over \( t_i \), where at most one taxonomy may be defined over each tag \( t_i \). A generalized association rule is represented in the form \( X \Rightarrow Y \), where (i) \( X \) and \( Y \) are disjoint conjunctions of tagged items in \( \mathcal{E} \), \( X \neq \emptyset \), \( Y \neq \emptyset \), \( X \cap Y = \emptyset \), and (ii) each tag may be referenced at most once in \( X \cup Y \).

For example, the association rules

\[
\{\text{source-address} = 130.192.1.1\} \Rightarrow \{\text{destination-address} = 130.192.254.1\}
\]

\[
\{\text{source-address} = 130.192.1.2\} \Rightarrow \{\text{destination-address} = 130.192.254.1\}
\]

may not be frequent (i.e., they may have a lower support than the minimum support threshold), but the generalized rule, obtained by aggregation of the source-address tag,

\[
\{\text{source-address} = \text{Subnet1}\} \Rightarrow \{\text{destination-address} = 130.192.254.1\}
\]

might. Hence, generalized patterns derived from rare itemsets may be relevant for network monitoring (e.g., source subnets which contact a print server). Instead of extracting rules for all levels of the taxonomy and post-pruning them [15], NETMINE generalization over the taxonomy is support driven, i.e., if lower levels of the taxonomy are over the support threshold, they are not aggregated.

2.2.1. Generalized association rules mining

Generalized association rule mining is a two-step process: (i) frequent generalized itemset extraction and (ii) rule generation from frequent itemsets. Since itemset mining is considered the most computationally intensive knowledge extraction task [1], the novel contribution of the Genio algorithm focuses on this step, while the second step is based on Goethal’s implementation [7] of the Apriori algorithm [1].

Given a dataset (either a network trace or a data stream view), a set of taxonomies (at most one for each network protocol attribute), and a minimum support threshold \( s \), the Genio algorithm [6] extracts all generalized itemsets whose support is above support threshold \( s \). Instead of extracting itemsets for all levels of the taxonomy and post-pruning them [15], the Genio algorithm performs a support driven opportunistic aggregation of itemsets. More specifically, generalized itemsets are extracted only if items at a lower level in the taxonomy are below the support threshold.

Genio is a level-wise algorithm, which, at each iteration, generates only the frequent, possibly generalized, itemsets of a given length. At arbitrary iteration \( k \), three steps are performed: (i) candidate generation, in which all possible \( k \)-itemsets are generated from \( \binom{k}{C_0} \) itemsets, (ii) database scan to count the support of candidate itemsets and discard infrequent itemsets, and (iii) infrequent itemset management to extract hidden knowledge ignored by previous approaches. In the third step, a generalized version of infrequent itemsets is generated by means of taxonomies. Only generalized itemsets above the support threshold are kept.

Finally, after all generalized itemsets have been extracted, interesting generalized rules are generated by means of Goethal’s implementation of Apriori [7], possibly enforcing a confidence threshold.

The above generalized rule extraction process is not designed to be performed in real-time during the capturing phase. However, experiments on the current implementation of the framework show the feasibility of this approach for appropriate sliding window update frequencies (see Section 3.6 for more details).
2.2.2. Data stream view

The data stream view block allows the selection of a subset of the flows obtained as continuous query output. A data stream view is a query to reduce the amount of data by means of aggregation and filtering operations. Aggregation operations reduce the amount of data by collapsing many semantically connected records in a single group. Hence, aggregation provides a higher level domain abstraction. Filtering operations select a subset of records or groups by applying a boolean combination of conditions (expressed by means of comparison operators \{\textless , \leq , \geq , >, <\geq =\}) on records or groups.

The following example shows the usefulness of data stream views.

Example. 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 (e.g., packets differing in one of these features belong to different flows). For each flow the sum of the packet size is computed. This step is performed by running the following continuous query on the data stream.

<table>
<thead>
<tr>
<th>Aggregate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select source-address, source-port, destination-address, destination-port, Sum(size) as flow-size</td>
</tr>
<tr>
<td>From Packets [Range 60 s]</td>
</tr>
<tr>
<td>Where level4 = ‘TCP’</td>
</tr>
<tr>
<td>Group source-address,source-port,destination-address,destination-port</td>
</tr>
<tr>
<td>By address,destination-port</td>
</tr>
</tbody>
</table>

Since the complete dataset contains thousands of flows, the output of the previous continuous query may be further filtered. The following query may be exploited to create a data stream view based on a filtering operation. Hence, flows whose size is lower than a threshold \(x\) are the output of the data streaming processing phase.

<table>
<thead>
<tr>
<th>Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select source-address, source-port, destination-address, destination-port, flow-size</td>
</tr>
<tr>
<td>From Aggregate</td>
</tr>
<tr>
<td>Where flow-size &lt; (x)</td>
</tr>
</tbody>
</table>

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

Among the extracted information, rules with the template \{destination-address, destination-port\} \(\Rightarrow\) \{flow-size\}, may highlight attempts of SYN flooding attacks, where the left term is the victim fingerprint, while the right term is the size of the flow (which is supposed to be small, due to the filtering condition in the data stream view).

Recall that a SYN flooding attack occurs when a victim host receives an excessive number of incomplete connection requests that it cannot 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 [16] may be detected by applying the proposed framework in the described conditions.

Other continuous queries may be exploited to define different data stream views, which may lead to the same results. For example, the following query exploits the information on the number of packets in a flow.

<table>
<thead>
<tr>
<th>Aggregate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select source-address, source-port, destination-address, destination-port, count(+) as flow-packets</td>
</tr>
<tr>
<td>From Packets [Range 60 s]</td>
</tr>
<tr>
<td>Where level4 = ‘TCP’</td>
</tr>
<tr>
<td>Group source-address,source-port,destination-address,destination-port</td>
</tr>
<tr>
<td>By address,destination-port</td>
</tr>
</tbody>
</table>

The filtering phase changes accordingly as follows.

<table>
<thead>
<tr>
<th>Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select source-address, source-port, destination-address, destination-port, flow-packets</td>
</tr>
<tr>
<td>From Aggregate</td>
</tr>
<tr>
<td>Where flow-packets &lt; (y)</td>
</tr>
</tbody>
</table>

The query counts the number of packets in a flow and keeps flows with a number of packets higher than a threshold \(y\) (the filtering effect is similar to that of the size attribute in the previous query). Also in this case, extracted rules can reveal SYN flood attack attempts. In particular, the rule template \{destination-address, destination-port\} \(\Rightarrow\) \{flow-packets\} highlights them. Hence, the framework is able to support the detection of specific traffic issues even if the chosen queries are not intentionally designed to address them and are not based on previous knowledge of the desired results. \(\square\)

2.3. Rule classification

Association rules highlight correlations among attributes and recurrent patterns. However, analyzing the (usually) large number of extracted rules is not a trivial task. To address this issue, we propose a classification of the rules into groups according to their semantics in the network domain. The semantics of a rule is determined by its template, i.e., by the referenced tags (e.g., source and destination addresses, port numbers, etc.) and their position (i.e., in the body or head of the rule).

Although an exhaustive analysis of all the possible semantic classes is infeasible, due to the high number of different attributes and their combinations, some basic groups have been identified. Their purpose is to guide the analysis of the extracted rules in the network domain.

The basic classes are built upon the following attributes (i.e, tags) of network data:

- Source address (abbreviated \(sa\)).
- Destination address (\(da\)).
- Destination port (\(dp\)).
The source and destination addresses represent fundamental information for tracing network flows, and the destination port is typically associated with a specific service\(^1\) (e.g., well known ports). Given this information, three basic classes of rules may be defined to analyze the following features of network traffic.

Traffic flows (abbreviated TF) can be analyzed by rules involving source and destination addresses.

Provided services (PS) are shown by rules consisting of destination port (i.e., the service) and destination address (i.e., the service provider).

Service usage (SU) is reported by rules having destination port and source address (i.e., the service user).

To define the semantic classes that categorize the rules, in Table 1 we analyze all the possible length-2 rule templates (i.e., templates for rules involving 2 tagged items) built from the three chosen tags, while in Table 2 all the length-3 rule templates are reported.

Rules which are not classified by the above classes may fall into two categories. They may be a specialization of the basic classes, or they may reference different tags. Specializing rules add one or more attributes to the rules identified for the basic classes (see Tables 1 and 2). For example, consider also the attributes source port (sp) and size (sz). The specializing rule template \(\{da, dp\} \Rightarrow \{sa, sp, sz\}\) adds the sp and sz tags to the rule template \(\{da, dp\} \Rightarrow \{sa\}\). Hence, it specializes the traffic flow and service usage classes by considering also correlations with the source port and the size of exchanged data.

The rule template \(\{sp\} \Rightarrow \{sz\}\) does not involve any of the three tags on which classes are built. However, it can reveal unexpected correlations, extracted by an exploratory analysis because of a high support or a high confidence.

Example 1 – Support and aggregation. Consider the following generalized association rule with support \(s = 2\%\) and confidence \(c = 100\%\).

\[
\{sa = AdminSubnet, dp = SharedPrinter\} \Rightarrow \{da = PrintServer\}
\]

---

\(^1\) The association of the destination port to a specific service is restricted to the well-known and registered ports. However, throughout the paper we generally describe rule templates considering the included tags (e.g., destination port) and not their actual value (e.g., actual port number). Hence, if a rule includes the destination port tag, then it is classified as providing service information, independently of the port number.
It belongs to the classes TF (traffic flow between the Admin Subnet and the Print Server) and PS (it concerns a service provided by the Print Server). The rule highlights that the Admin subnet always (confidence is 100%) uses the Print Server when the Shared Printer service is concerned. Since an aggregation is performed for the source address (from single host to whole subnet), we can conclude that no host in the Admin subnet generates enough traffic to exceed the minimum support threshold. Note that if the proposed generalized rule approach had not been applied, neither the rule with the host, nor the rule with the subnet as source address would have been generated.

Example 2 – confidence. Consider the following generalized association rule with confidence 80%.

\[ \text{da} = \text{PrintServer} \]
\[ \Rightarrow \{ \text{dp} = \text{SharedPrinter}, \text{sa} = \text{AdminSubnet} \} \]

The rule confidence value highlights a usage profile where 80% of the traffic directed to the Print Server comes from hosts in the Admin Subnet which request the Shared Printer service. This strongly characterizes the service provided by the Print Server (class PS) correlated with a specific traffic flow (class TF).

In a different network context, we could have obtained the following couple of rules.

\[ \text{da} = \text{PrintServer} \]
\[ \Rightarrow \{ \text{dp} = \text{SharedPrinter}, \text{sa} = \text{AdminSubnet} \} \text{ c = 40\%} \]
\[ \text{da} = \text{PrintServer} \]
\[ \Rightarrow \{ \text{dp} = \text{SharedPrinter}, \text{sa} = \text{ResearchSubnet} \} \text{ c = 30\%} \]

In this case, of all the traffic toward the Print Server, 40% is from the Admin Subnet, 30% is from the Research Subnet, and both subnets request the Shared Printer service. Thus, we also expect a new rule stating that at least 70% of the traffic directed to the Print Server requests the Shared Printer service. Such rule has the following form (class PS).

\[ \text{da} = \text{PrintServer} \Rightarrow \{ \text{dp} = \text{SharedPrinter} \} \text{ c = 70\%} \]

The former example shows an important property of association rules. Lowering the minimum confidence threshold and increasing the minimum support drives the analysis toward more frequent patterns with weaker correlation. In the network domain, this approach leads to profiling the traffic of the top users/providers of the network.

Vice versa, lowering the minimum support threshold and keeping the confidence high leads to the identification of hosts with particular traffic patterns, e.g., using or offering very specific services and/or contacting only specific subnets. In the following, examples of this behavior are given.

Example 3 – specializations. Consider the following rule.

\[ \{ \text{sa} = \text{internal-host}, \text{da} = \text{external-host}, \text{dp} = 443 \} \Rightarrow \{ \text{sz} = 254\text{Kbytes} \} \text{ c = 100\%} \]

It expresses information on a traffic flow for a specific service (https). An interesting (and unexpected) correlation links such conversation with a fixed size of the flows. This may disclose a property of the specific protocol version in use by those hosts or, if such hypothesis is unacceptable (based on previous knowledge of the protocol itself), it may be a symptom of improper use of a well known port for a different service, which is worth to be further investigated. Examples of such rules find evidence in Example 3 of the experimental results (Section 3.5).

Example 4 – specializations and aggregations. Consider the following rule.

\[ \{ \text{sa} = \text{external-network}, \text{da} = \text{host}, \text{dp} = 1000 \} \Rightarrow \{ \text{sz} = \text{small} \} \text{ c = 100\%} \]

The rule highlights that flows from the external network to a specific host on port 1000 always have a small size, where small is defined as an aggregation of the size attribute, as the external-network is an aggregation of the external hosts. This is a generalization over the size and source address attributes of a rule similar to Example 3. Aggregation performed on the sa attribute indicates that all the external hosts behave in the same way as identified by the rule. Aggregation on the sz attribute shows that packet sizes may vary, but small packets are always exchanged. Similar patterns are also reported in Example 3 of the experimental results (Section 3.5).

3. Experimental validation

A set of experiments has been performed by applying the proposed techniques on network traffic datasets. We analyzed the effect of the support and confidence thresholds on various classes of generalized rules (Section 3.1), the impact of the window size parameter (Section 3.2) and the effectiveness of our generalization approach in extracting hidden knowledge (Section 3.3). The selection of interesting rules by means of the lift quality index is discussed in Section 3.4. Section 3.5 presents the analysis of a subset of interesting rules in the network context and some interesting analysis scenarios. Finally, the feasibility of the online rule extraction process is discussed in Section 3.6, in which generalized association rules are mined from traces of different duration.

Two network traffic datasets have been obtained by performing two capture sessions by means of the Analyzer tool [28] on a backbone link of our campus network. In Table 3, the number of packets, the trace file sizes (including packet headers only), and the capture durations are reported for each dataset. For each flow, the network attributes considered in the analysis are source address, destination address, and destination port, together with source port and flow size (i.e., the size in byte of the traffic flow).

<table>
<thead>
<tr>
<th>ID</th>
<th># of packets</th>
<th>Trace file [MB]</th>
<th>Capture duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>25969389</td>
<td>2621</td>
<td>41'15s</td>
</tr>
<tr>
<td>B</td>
<td>26023835</td>
<td>2625</td>
<td>41'56s</td>
</tr>
</tbody>
</table>

Example 3 – specializations. Consider the following rule.

\[ \{ \text{sa} = \text{internal-host}, \text{da} = \text{external-host}, \text{dp} = 443 \} \Rightarrow \{ \text{sz} = 254\text{Kbytes} \} \text{ c = 100\%} \]
Experiments addressed the execution of the sample query presented in Section 2.1. 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

\[
\text{link speed \times window size \over \text{minimum packet size \times number of flows}}.
\]

Experiments have been performed considering a window size of 60 s and a link speed of 100 Mbps. The value of the window step \( \text{step} \) has been set to \( \text{window size \over 2} \). The ratio parameter in the sample query (Section 2.1) has been set to 0.1.

Generalized itemset mining is performed by means of the Genio algorithm [6], which is able to generalize and aggregate infrequent items according to given taxonomies. The taxonomies used in this set of experiments aggregate infrequent items according to the following strategies. TCP ports are aggregated into three ranges: well known ports (between 1 and 1023), registered ports (between 1023 and 49,151) and dynamic ports (otherwise). IP addresses which are local to the campus network are aggregated by subnet. IP addresses which do not belong to the campus network are aggregated in a general external address group. Other aggregation strategies, for example driven by geographical location, can be considered as well. The flow size attribute is aggregated over a taxonomy of 4.
The current implementation of the NETMINE framework on a 2.6 GHz Pentium IV PC with 2 Gb main memory running Linux (kernel 2.6.81). The current implementation of the NETMINE framework allows real-time execution of the stream processing at a network speed of 100 Mbit/s, with realistic packet sizes and network loads. Experiments have been performed on both datasets A and B. However, for some specific experiments we only report results for dataset A, since they are analogous for dataset B.

### 3.1. Effect of the support and confidence thresholds

Different minimum support and confidence thresholds significantly affect the cardinality of the extracted rule set and the nature of the rules. In Figs. 3–6 we report, for the two datasets, the number of extracted rules for different combinations of the threshold values. Rules are divided in three categories: basic rules, i.e., rules belonging to the six classes described in Section 2.3 (see Tables 1 and 2), specializing rules, which add one or more tags to the former rule templates (see Section 2.3), and other rules. The absolute number of extracted rules is different in the two datasets, because of the different number of flows (see Table 4) and the different sparsity of original data. However, the global trend is rather similar.

As expected, the number of rules decreases when both the minimum support and the minimum confidence thresholds are increased. The support threshold is rather selective. For high support values, only a small number of rules is extracted (see Figs. 3 and 4). However, it is important to note that frequent is not necessarily a synonym for interesting. A rather high number of strong correlations is instead extracted also for high confidence values (see Fig. 5).

The cardinality of the set of specializing rules is much higher than other categories. In particular, the number of specializing rules with low support is rather high (see Figs. 3 and 4). This effect may be due to the loose constraints on their structure. These rules have not a fixed length and they may include many different combinations of attributes. Thus, for low support thresholds, a large variety of combinations may satisfy the support constraint. These rules are suitable for capturing unexpected peculiar knowledge. Furthermore, their meaning may be more easily interpretable by exploiting their specialization of the basic classes.

Many examples of specializing rules highlight correlations relating basic attributes (da, sa, dp) with the size attribute. This effect is referable to the particular taxonomy of the size tag. Its values are discretized into four bins only, leading to a very dense aggregation. Hence, each single aggregation value becomes rather frequent. Diverse discretization techniques or intervals may lead to a different behavior. A similar behavior is shown by the source port attribute, which is often aggregated as registered or dynamically-assigned. This reveals the allocation policy for the client source port, which is typically dynamically assigned on the client host, always excluding the well-known ports.

The support and confidence thresholds also affect the kind of extracted rules. By setting high support thresholds, only very frequent patterns are extracted. However, their interest may be marginal. For example, when the support threshold is set to 10% and the confidence to 20%, the basic category contains four rules. One of the extracted rules is \( \{ \text{sa} = \text{external-address} \} \Rightarrow \{ \text{dp} = \text{registered} - \text{port} \} \)

which is characterized by a support of 57% and a confidence of 87%. To satisfy the high selectivity of the minimum support threshold, the generalization process has led to a rule which is too general to provide interesting knowledge. Instead, the use of a low support threshold coupled with different quality indices (e.g., confidence, lift) leads to the extraction of a higher number of rules where peculiar patterns arise. This issue is further discussed in Section 3.4, while Section 3.5 provides some interesting examples of rules which are extracted by lowering the support threshold.

Focusing on basic rules, in Fig. 7 their distribution among the six classes is reported for dataset A. For rules of length 2 (i.e., rules belonging to TF, PS, and SU classes) the TF class is by far the most frequent. This is an expected behavior because the cartesian product of the domains of the tags source-address and destination-address has a much higher cardinality than that of the domains of the other tags. Furthermore, their aggregations over the taxonomy have a finer granularity (i.e., there are more subnets than families of ports), thus further increasing the domain cardinality. On the contrary, rules of length 3 are equally distributed among the three classes (TF + PS, TF + SU, PS + SU).

### 3.2. Effect of the window size

The window size directly affects the aggregation of the traffic data performed by the continuous query. The sam-

---

**Table 4**

Number of flows yielded by the sample continuous query on each dataset.

<table>
<thead>
<tr>
<th>Dataset ID</th>
<th># of selected flows</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>18,051</td>
</tr>
<tr>
<td>B</td>
<td>16,783</td>
</tr>
</tbody>
</table>

Bins, whose intervals are \([1,1000), [1000, 2000), [2000, 3000)\] and \(\geq 3000\) bytes.
ple continuous query (see Section 2.1) is characterized by a ratio parameter whose absolute value varies for different window sizes. This parameter is expressed as a percentage of the whole flow size in a window. Hence, changing the window size in the data stream block strongly affects the input of the refinement analysis phase, because the obtained flows are characterized by a different data distribution. It becomes thus very difficult to detect a correlation between the refinement analysis results and the variation of the window size.

In Table 5, the number of flows obtained for different values of the window size is reported. The results confirm the expected behavior on both datasets, with a decreasing flow number for increasing window size values.

Fig. 8 shows, for dataset B, the number of extracted rules with different settings of the window size. Changing the window size from 30 s to 60 s does not affect the number of extracted rules, which varies only slightly. On the contrary, when the window size is set to 10 s, the difference becomes significant, because very few rules are extracted.

Unfortunately, a direct impact of the window size on the final rule set could not be identified, because of the joint effects of this parameter. Furthermore, the number of extracted rules for different window size values is modified by the effect of the window size on the ratio parameter. Thus, general conclusions cannot be drawn.

3.3. Impact of generalization on the rule extraction process

Since the number of different values assigned to each attribute is huge, a network trace is characterized by a very sparse data distribution. In similar conditions, the number of items which are frequent in the dataset is usually small. This problem usually affects the effectiveness of traditional association rule extraction.

Figs. 9 and 10 show, for different settings of minimum support, the percentage of generalized association rules and the percentage of specific rules, which are built using non-aggregated items. This experiment gives a measure of the number of rules which would have been discarded if a traditional approach had been used with the same support threshold. Other values of minimum confidence yield analogous results, as rules with high confidence are rather uniformly distributed over a wide support range.

Since in the NETMINE framework infrequent items are aggregated during the extraction process, the percentage of generalized rules increases when the support threshold is increased. Especially for high support thresholds the extraction of generalized rules allows highlighting correlations that would otherwise remain hidden, because of their low support.

The main purpose of generalized association rule extraction is providing an extended set of association rules, including rules which do not separately meet the support threshold but may contribute to a generalized rule, thus not being discarded by the frequency threshold. A quality index may be then applied for selecting high quality rules from this extended pool. We discuss in Section 3.4 how to exploit the lift quality index to rank the set of extracted rules and select the most interesting ones. Some of the se-

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Window size (s)</th>
<th># of selected flows</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td>93,648</td>
</tr>
<tr>
<td>A</td>
<td>30</td>
<td>36,205</td>
</tr>
<tr>
<td>A</td>
<td>60</td>
<td>18,051</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>89,931</td>
</tr>
<tr>
<td>B</td>
<td>30</td>
<td>33,610</td>
</tr>
<tr>
<td>B</td>
<td>60</td>
<td>16,783</td>
</tr>
</tbody>
</table>

Fig. 8. Dataset B: Number of rules extracted with minimum confidence = 20% and minimum support = 1%, by executing the query with different values of window size.

Fig. 9. Dataset A: Percentage of generalized and specific rules extracted for different values of minimum support with minimum confidence = 20%.
lected rules would not have been extracted with traditional approaches (i.e., without generalization).

3.4. Selection of the most relevant rules

By setting the minimum confidence threshold to 40% and the minimum support threshold to 0.05%, a total of 9810 rules have been extracted from dataset A. A simple technique to focus user analysis is the selection of rules containing specific combinations of attributes, or specific values. However, this technique does not consider the overall quality of rules. Hence, to support the user in the selection of the most interesting rules, many quality measures of the goodness of a rule may be exploited [18]. In this experiment rules have been automatically sorted by means of the lift quality index, which is a measure of how much a rule is likely to be interesting. For an association rule in the form

\[ A \Rightarrow B \]

the lift measure [18], is defined as

\[ \text{lift}(A, B) = \frac{s(A \Rightarrow B)}{s(A)s(B)} \]

i.e., the ratio of the support of the rule to the product of the support of \( A \) and the support of \( B \). If the resulting value is 1, then the itemsets \( A \) and \( B \) are not correlated, i.e., they are statistically independent. Instead, if the resulting value is less than 1, the itemset \( A \) and \( B \) are negatively correlated. Otherwise, they are positively correlated, meaning that the occurrence of one implies the occurrence of the other. This index can be used to rank rules. Exploiting the lift quality index, the top 10 rules in the previous set of 9810 are reported in Table 7.

Some interesting application examples are among these top rules and will be discussed in the next section. The last three rules in this restricted high quality subset are generalized rules, which would not have been extracted with traditional techniques. Aggregation allowed the generalized rules to meet the minimum support and confidence thresholds, while the specific rules contributing to them would not have been individually extracted.

3.5. Application examples

We discuss in the following some interesting analysis scenarios taken from the results presented in Section 3.3 and from the same experiment replicated on dataset B.

**Example.** Rule number 1 in Table 6 is analyzed.

\[ \{sa = xxx.x.119\} \Rightarrow \{da = 130.192.y.y\} \]

\( s = 4\%, c = 100\% \)

It belongs to class TF. In particular, it identifies a traffic flow from a specific external IP address to a single internal host. Since the rule confidence is 100%, all the traffic coming from this external IP goes to the specific internal host. The following rule of class TF + SU (also extracted) provides further insight on this traffic flow.

\[ \{sa = xx.x.xx.xx\} \Rightarrow \{da = 130.192.y.y.yy, dp = \text{registered}\} \]

\( s = 4\%, c = 100\% \)

In particular, this generalized association rule shows that the internal host is contacted on different port numbers but they all belong to the registered ones. We can con-
clude that the specific external IP is contacting exclusively the internal host, which we have found to be the campus VPN concentrator.

A domain expert may be interested in knowing whether the identified external IP is among the top users of the VPN concentrator. For this purpose, extracted rules of class TF + PS like the following should be considered:

\( \{ da = 130.192.y.y \} \Rightarrow \{ sa = x.x.x.119 \} \)  
\( \{ da = 130.192.y.y, sp = 8080 \} \Rightarrow \{ sa = x.x.x.70 \} \)

This generalized rule indicates that 62% of the traffic directed to the VPN concentrator comes from the external IP address previously identified. Hence, this address is actually the top user of such service. The same conclusions might be reached by directly querying the network traffic dataset, but the formulation of the appropriate set of queries would require previous knowledge of the specific profiles to be analyzed. On the contrary, the combination of generalized association rules and their categorization is able to assist the network analyst in quickly determining which patterns are worth investigating. □

**Example.** Rule number 8 in Table 6 is analyzed.

\( \{ da = 130.192.zz.zz \} \Rightarrow \{ dp = 57403, sa = External \text{–} address \} \)

\( s = 1.4\%, c = 86\% \)

It belongs to class TF + PS. This generalized association rule highlights an unconventional high-volume traffic towards a specific host of the campus network. It targets a single internal IP address and a single port that is not associated to any well known service. Furthermore, the rule is characterized by high confidence and support despite no aggregations over the destination port and address. Two situations may describe the identified behavior: (i) the host may be offering a kind of unconventional service, because most of its traffic is actually directed to a single port and comes from different addresses of the external network, as shown by the generalization on the source address, or (ii) the host is downloading data from different locations outside the campus network with the peculiar characteristic of using always the same port. □

**Example.** From dataset B with minimum support 0.05% and minimum confidence 40%, the following specializing rules have been extracted.

\( \{ sa = x.x.x.119 \} \Rightarrow \{ da = 130.192.y.y \} , \text{Lift} = 179 \)

\( \{ da = 130.192.y.y, sp = 8080 \} \Rightarrow \{ sa = x.x.x.70 \} , \text{Lift} = 147 \)

\( \{ sa = x.x.x.70 \} \Rightarrow \{ sp = 8080 \} , \text{Lift} = 110 \)

\( \{ sa = x.x.x.70, da = 130.192.y.y \} \Rightarrow \{ sp = 8080 \} , \text{Lift} = 110 \)

\( \{ sp = 8090 \} \Rightarrow \{ sa = 130.192.y.y \} , \text{Lift} = 90.0 \)

\( \{ da = 130.192.y.y \} \Rightarrow \{ sp = 8080 \} , \text{Lift} = 79.6 \)

\( \{ da = 130.192.y.y \} \Rightarrow \{ dp = 57403, sa = external \text{–} address \} , \text{Lift} = 70.5 \)

\( \{ dp = 57403 \} \Rightarrow \{ da = 130.192.y.y, sp = registered \} , \text{Lift} = 70.5 \)

\( \{ sa = external \text{–} address, dp = well \text{–} known \} \Rightarrow \{ da = 130.192.y.y, sp = registered \} , \text{Lift} = 46.8 \)

### Table 6

**Dataset A: Top 10 rules according to the lift quality index.**

<table>
<thead>
<tr>
<th>#</th>
<th>Rule</th>
<th>Lift</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{sa = x.x.x.119} ⇒ {da = 130.192.y.y}</td>
<td>179</td>
</tr>
<tr>
<td>2</td>
<td>{da = 130.192.y.y, sp = 8080} ⇒ {sa = x.x.x.70}</td>
<td>147</td>
</tr>
<tr>
<td>3</td>
<td>{sa = x.x.x.70} ⇒ {sp = 8080}</td>
<td>110</td>
</tr>
<tr>
<td>4</td>
<td>{sa = x.x.x.70, da = 130.192.y.y} ⇒ {sp = 8080}</td>
<td>110</td>
</tr>
<tr>
<td>5</td>
<td>{sp = 8090} ⇒ {sa = 130.192.y.y}</td>
<td>90.0</td>
</tr>
<tr>
<td>6</td>
<td>{da = 130.192.y.y} ⇒ {sp = 8080}</td>
<td>79.6</td>
</tr>
<tr>
<td>7</td>
<td>{da = 130.192.y.y} ⇒ {dp = 57403, sa = external – address}</td>
<td>70.5</td>
</tr>
<tr>
<td>8</td>
<td>{dp = 57403} ⇒ {da = 130.192.y.y, sp = registered}</td>
<td>70.5</td>
</tr>
<tr>
<td>9</td>
<td>{sa = external – address, dp = well – known} ⇒ {da = 130.192.y.y, sp = registered}</td>
<td>46.8</td>
</tr>
</tbody>
</table>

3.6. **Feasibility of the online rule extraction**

In the NetMine framework, packets are processed by the continuous query block concurrently with data capture. The flows which are produced by this block are then processed off-line for the extraction of generalized association rules. To assess the latency between the data capture and the extraction of the generalized association rules, we performed an experiment measuring the time requested for the latter phase. In this experiment, we considered a link capacity of 100 Mbps operating at the maximum theoretical speed (i.e., packets in the trace are assumed to arrive with the minimum inter-packet gap). The sample query has been executed on dataset A, with a window size of 60s and the ratio parameter set to 0.1. The extraction process has been performed with minimum support threshold = 1% and minimum confidence = 20%.

The results of the experiment are reported in Fig. 11, where the capture time is the duration of the capture with

### Table 7

**Dataset A: Execution times of the extraction process with respect to the capture times of the processed traffic data.**

<table>
<thead>
<tr>
<th>Capture time (s)</th>
<th>Execution time (s)</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>245</td>
<td>77</td>
<td>31%</td>
</tr>
<tr>
<td>520</td>
<td>138</td>
<td>27%</td>
</tr>
<tr>
<td>780</td>
<td>254</td>
<td>33%</td>
</tr>
<tr>
<td>1047</td>
<td>342</td>
<td>33%</td>
</tr>
</tbody>
</table>
the assumption of maximum theoretical speed and the execution time is the time required to perform the rule extraction on captured data. The execution time is (almost) linear with respect to the capture time. The time required for the extraction is always less than the time required for the data capture. More specifically, the execution time is always roughly 30% of the capture time, as reported in Table 7. Hence the refinement analysis could be performed concurrently with the traffic capture in the following conditions: (i) a sliding window selects a subset of the flows on which the extraction process is executed, (ii) the sliding window is updated at an interval less than a third of its size. In this way the results of the extraction process are always ready before updating the sliding window with the new captured data.

4. Related work

A significant effort has been devoted to the application of data mining techniques to the problem of network traffic analysis. The work described in [9] presents some theoretical considerations on the application of data mining techniques to network monitoring. Since the network traffic analysis domain is rather broad, research activities have addressed many different application areas, e.g., web log analysis [37], enterprise-wide management [21], and traffic classification [14,29,33].

Traffic data categorization, addressed by means of classification techniques, is an effective tool to support network management [29]. In general, classification techniques can be divided in supervised and unsupervised. While the first group requires previous knowledge of the application domain, i.e., new unlabeled traffic flows are assigned a class label by exploiting a model built from traffic network data with known class labels, the second does not. Furthermore, network traffic classification can be performed by analyzing different features: (i) packet payloads, (ii) traffic metrics, and (iii) statistical features computed on traffic flows.

Traditional traffic classification techniques perform a deeper inspection of packet payloads [14,33] to identify application signatures. To apply these approaches, the payload must be visible and readable. Both assumptions limit the feasibility of these approaches [29]. First of all, payloads could be encrypted, making the deep packet inspection impossible. Furthermore, the classifier has to know the syntax of each application payload, to be able to interpret it. The approach in NETMINE does not require packet payload inspection. It analyzes instead a set of packet header fields and some additional features of flows (e.g., sum of packet sizes, number of packets).

Other classification approaches which target traffic data categorization only consider a set of statistical features for each flow of packets (e.g., packet inter-arrival time, number of new sessions). These analysis techniques automatically infer categorization of traffic data according to the application level protocols, or extract patterns of normal and abnormal traffic. Both unsupervised [25,38,8] and supervised [3,26] classification techniques have been exploited.

In [25], several statistics are collected (e.g., packet length, inter-arrival time, connection duration) for each flow. Flows characterized by similar statistics are grouped together by means of the EM clustering algorithm [18]. Thus, according to the clustering structure, different types of traffic data are identified (e.g., HTTP, FTP, SMTP). The approach proposed in [38] also exploits the EM algorithm to identify the best cluster set from the training data. Given the clustering result, a Bayesian classifier is exploited to classify new incoming data. Instead the work in [8] exploits the K-Means algorithm to group together flows which belong to the same application level protocol.

In the context of supervised learning algorithms, several techniques (e.g., Naive Bayesian classifiers [26], Bayesian neural networks [3]) have been exploited to classify traffic data. These approaches perform the analysis by considering a very large set of statistics (e.g., 248 statistics [26]). However, these techniques require previous knowledge of the application domain (i.e., a set of labeled data).

Differently from the above approaches, the NETMINE framework provides an effective and compact explorative approach to support network traffic analysis and network management. Traffic analysis discovers a set of rules which describe the network behavior and supports the network administrator in efficiently identifying relevant and unexpected correlations among traffic data.

Continuous queries have been applied in the context of network traffic management to real-time monitoring of network behavior. In particular, they have been exploited to detect congestions and their causes [4] and to support load balancing [11]. In these works, network data analysis is directly performed by means of continuous queries, without data materialization and further data exploration by means of data mining techniques.

5. Conclusions

NETMINE is a framework to efficiently perform network traffic analysis. It combines two phases: data stream analysis and refinement analysis. The former reduces the amount of traffic data while the latter extracts relevant and unexpected correlations and recurrence of patterns among data. The proposed technique exploits a taxonomy to drive the extraction process by automatically aggregating extracted information according to the support threshold. Experiments proved the capability and the effectiveness of NETMINE to highlight meaningful characteristics of network traffic.

Acknowledgements

We are grateful to Fulvio Risso for providing the real traffic datasets captured from the campus network, and to Claudio Testa and Felice Iasevoli for developing parts of the NETMINE framework.
References

Tania Cerquitelli is a post-doctoral researcher in Computer Engineering at the Politecnico di Torino since January 2007. She holds a Ph.D. degree and a Master degree in Computer Engineering, both from the Politecnico di Torino. She also obtained a Master degree in Computer Science from Universidad De Las Américas Puebla. Her research activities are focused on the integration of data mining algorithms into database system kernels and on the exploitation of data mining techniques to analyze streaming data collected by sensor networks. She served in the program committee of DS2ME’08 and IADIS’06.

Vincenzo D’Elia is a Ph.D. student of the Database and Data Mining Group at the Dipartimento di Automatica e Informatica of the Politecnico di Torino since January 2007. He obtained a Master degree in Computer Engineering from Politecnico di Torino in November 2006. He is currently working in the field of data mining, and time series analysis. In particular, his activity is focused on the analysis of sensor network data to support effective data collection by means of data mining techniques. His research activities are also devoted to analyze physiological signals to study athlete conditions while they perform their activities.