Real Time Traffic Recognition

Hua Cao
Abstract

Real Time Traffic Recognition

Hua Cao

The rapid growth of Internet in size and complexity, and frequent emergence of new network applications have made it necessary to develop techniques that can monitor and control the traffic. Efficient and accurate recognition of traffic is the key to the management in real time. This thesis work accomplishes the performance evaluation and optimization of a traffic recognition tool called Traffic Analyzer Module (TAM) which implements a technique that is based on passively observing and identifying signature patterns of the packet payload at the application layer, i.e., signature-based payload recognition. This technique has two highlighted features. Firstly, in contrast to most of previous works which perform classification with offline trace files; this technique applies in online mode which can identify the traffic in real time. Secondly, instead of packet inspection, this technique adopts flow inspection, i.e., identifying traffic in terms of flows each of which consists of the well-known 5-tuple, which can produce more accurate and reliable results.

To demonstrate this technique, its throughput is evaluated in online mode within a high bandwidth network. Besides throughput measurement, optimizing the recognition algorithm in order to improve its performance is also a task of this thesis work. The results of performance measurement demonstrate the feasibility and reliability of this technique, as well as indicate some clues for future work.
Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor Tord Westholm at Ericsson Research for providing me the opportunity to conduct and accomplish my master thesis at Ericsson Research. I also appreciate all his guidance and suggestions for helping me fulfill this thesis work. I would like to thank Lars Westberg for all his instructions and advices around this thesis. I want to thank DJamel H. Sadok, Eduardo James Pereira Souto and Stenio Fernandes who helped me along this thesis with constructive information and comments. In the mean time, I want to thank my reviewer at Uppsala University, Ivan Christoff, for his comments. Last but not least, I would like to thank all the other people who helped me during the whole thesis work.
Contents

1 Introduction................................................................................................................. 5
2 Related Work ............................................................................................................. 7
3 Background............................................................................................................... 9
  3.1 Requirement........................................................................................................... 9
  3.2 Prototype............................................................................................................. 10
  3.2.1 Mechanism...................................................................................................... 10
  3.2.2 Architecture................................................................................................... 10
  3.2.3 Traffic Analyzer Module (TAM)...................................................................... 11
  3.2.4 Traffic Recognition....................................................................................... 12
4 Throughput Measurement and Optimization.................................................. 19
  4.1 Experimental Setup............................................................................................... 19
    4.1.1 Equipments..................................................................................................... 19
    4.1.2 Traffic............................................................................................................. 20
    4.1.3 Metrics............................................................................................................. 21
    4.1.4 Measurement Point....................................................................................... 25
    4.1.5 Configuration.................................................................................................. 25
  4.2 Results Overview................................................................................................... 27
  4.3 Experiments ........................................................................................................... 29
    4.3.1 Experiment 1: Getting Started ....................................................................... 29
    4.3.2 Experiment 2: At the Edge of the Network.................................................. 32
    4.3.3 Experiment 3: Heavy Traffic Load............................................................... 34
    4.3.4 Experiment 4: Kernel Configuration............................................................. 37
    4.3.5 Experiment 5: No Profiling............................................................................ 39
    4.3.6 Experiment 6: No Marshalling....................................................................... 40
    4.3.7 Experiment 7: Recognition Optimization................................................... 42
    4.3.8 Experiment 8: Gigabit Ethernet.................................................................... 46
    4.3.9 Experiment 9: Bandwidth Tuning................................................................. 49
    4.3.10 Experiment 10: Traffic Snapshot.................................................................. 51
5 Conclusion.................................................................................................................... 53
6 Discussion .................................................................................................................... 54
  6.1 Limitations.............................................................................................................. 54
  6.2 Future Work............................................................................................................ 54
7 References...................................................................................................................... 58
Appendices........................................................................................................................ 60
  A. Definitions................................................................................................................. 60
  B. Abbreviations.......................................................................................................... 61
1 Introduction

With the rapid development of computer networks, how to plan and design secure and robust networks as well as how to provide better services have become critical tasks for the network operators and the service providers. For this purpose, analysis of the network traffic has become one of the major interests for network operators and service providers. As network operators, it is important to know how the network resources are used and the status of the whole network in order to detect any abnormal situations, for instance exploit traffic, intrusion, malicious attacks or forbidden applications. The information and statistics can benefit the network operators in enforcing pre-defined security policies by controlling user’s traffic when needed for protecting security and improving robustness. From the perspective of service providers, it is profound to get an idea of how different services are used and the implicit demands from customers for further improvement of service quality, such as adding needed services and discarding less used services. In conclusion, monitoring the network traffic and recognizing the applications associated with traffic flows is an ideal approach. In this thesis, flow is defined as a uni-directional series of packets with the same 5-tuple identities, i.e. source IP address (SrcIP), destination IP address (DstIP), source port (SrcPrt), destination port (DstPrt) and transport layer protocol.

The purpose of this thesis work is to study a traffic recognition approach which has been proposed and implemented in a prototype called TAM by another internal project and then to evaluate and optimize its performance in order to make it become an online traffic recognition tool which can produce large throughput and high precision in a high speed link.

The developed tool is intended to run at the edge of the network, i.e. where the network connects to the Internet, and passively identify the applications of the traffic traverses the network. The target network is supposed to be high speed, such as Gigabit or 10 Gigabit Ethernet network.

The result of this thesis work can be applied to further detect any anomalous status of the target network, such as intrusion, malicious attacks or forbidden traffic, in order to maintain security and robustness of the network which means it can be part of a solution to build robust IP networks.

The contributions of this thesis work are summarized as follows:

- It describes the prior traffic classification approaches and addresses their merits and limitations.
- It illustrates the architecture of the prototype and the traffic recognition approach that is proposed in the prototype.
– It measures the online performance of the prototype and then points out the bottlenecks and means for overcoming them.

– It optimizes the implementation of the prototype to improve the throughput.

– It concludes the result and suggests future work for this topic.

The reminder of the paper is structured as follows. Chapter 2 discusses the related work of this field. Chapter 3 provides background information and illustrates the pre-developed prototype. The measurement of throughput including optimization process is stated in Chapter 4. Chapter 5 concludes the results. In Chapter 6, the limitations of this thesis work and future work are discussed.
2 Related Work

A lot of efforts are contributed to the traffic recognition and many different approaches have been proposed. In this chapter, some of these traffic classification methods are described.

- **Port-based classification**: The simplest approach to traffic classification is to associate the port numbers with applications as defined by IANA [1]. For instance, HTTP traffic takes port 80. It only needs to examine the port numbers in the TCP or UDP headers which are commonly available and match them in the pre-defined list so that it costs very little overhead. While the port-based approach was sufficient and effective in the early days, it is insufficient and currently misleading due to the emergence of P2P protocols and applications which use dynamic port negotiation instead of standard port numbers so that they have the ability to disguise their existence through the use of arbitrary ports [2]. Indeed, recently research has confirmed the failure of port-based method [3]. Therefore, it cannot be considered to be reliable.

- **Signature-based payload classification**: To make traffic recognition as accurate as possible, this approach accesses to the packet payload and attempts to match it with pre-defined payload signatures. This technique requires access to the payload and prior knowledge of the application signatures. Thus, it works well with well documented protocols and is able to identify specific applications. Due to its significant accuracy, sometimes it is selected as the validation benchmark of other methods. For example, [4] applies signature-based identification for evaluating its approach. However, its drawback is that this method costs a high storage and computational overhead to analyze every packet that traverses a link and cannot work on encrypted traffic.

- **Behavior-based classification**: It shifts the focus of the classification from the flow to the host. It associates hosts with applications by observing and identifying communication patterns of host behaviors [4], [5] instead of by studying flows individually. The communication patterns, i.e. the relationships among the source and destination IP addresses and ports, represent the flow characteristics corresponding to the applications. The advantage is that it does not need access to payload and can be accomplished in the transport layer which makes it be able to work on encrypted traffic. It is able to associate hosts with the services; however, it cannot classify a single flow, which is different from the goal of this thesis. Moreover, it is not efficient due to many flows are needed to go through a host for the accurate classification of the behavior patterns. Thus, it is not suitable for an online tool.
- **Information-theoretic-based classification**: The information theoretic approach is an aid in traffic classification by building a comprehensive view of the communication patterns of hosts and services. [6] introduces a methodology for profiling Internet backbone traffic that not only discovers significant behaviors of traffic, but also provides plausible means to aid in understanding and quickly recognizing anomalous traffic events. It extracts significant clusters from specific dimensions and then builds common behavior models for classifying end-host and service behaviors with similar communication patterns. It uses 5-tuple identity of the flows and only accesses to the packet headers. However, it classifies connections into types of applications instead of identifies the specific application itself, which is not the purpose of this thesis work. Meanwhile, both cluster extraction and classification phases can result in high degree of complexity so that this methodology produces low throughput. Another challenge is that this technique is sensitive to alternations of packet sizes and inter-arrival times. Consequently, it is not a good choice for online traffic recognition.

- **Statistics-based classification**: It extracts statistical features of the packet traces and treats them as discriminating criteria. These criteria are used to classify the traffic. The Bayesian analysis technique [7] exemplifies this classification idea. It requires manually classified traffic both for training and testing data-sets. It is able to classify the traces with only the access to the TCP headers. Instead, this technique classifies traffic flows until they are done so that it is not appropriate for early application recognition which limits its applicability for online usage.

- **First few packets-based classification**: [8], [9] illustrate a solution for early application identification at the beginning of a TCP connection. Based on an analysis of the packet traces, they demonstrate that it is possible to perform traffic classification from the observation of the size and direction of just the first few packets of a TCP connection. The size of the first few packets captures the applications’ negotiation phase, which is usually a pre-defined sequence of messages and distinct among applications. It works under two phases: an offline training phase and an online classification phase. The purpose is to provide an online mechanism that can identify traffic as early as possible. It limits the amount of memory required to store information associated with each flow. The limitation is that it needs each of the first few (e.g. four or five) packets of a TCP connection in the correct order and its accuracy depends on the training data. [10] proposes a method for encrypted traffic, which uses only the size of the first few packets of an SSL connection to recognize the application.
3 Background

The traffic recognition methods are difficult to execute correctly due to the knowledge commonly available to the network, i.e. packet headers in the transport layer, often does not contain sufficient information. While even with the access to the packet payload in the application layer, it is still not a trivial task. Thus, an accurate and efficient method which can identify the application associated with a traffic flow is indeed necessary to be developed.

The P2P traffic is one of the most challenging application types. It is not only because the usage of dynamic port negotiation for communication which leads the port-based method to be invalid, but also due to the large number of proprietary P2P protocols which increases the difficulty of the signature-based payload method. Thus, a lot of efforts have to be contributed for P2P traffic recognition. Moreover, some P2P traffic is relayed over HTTP, i.e. transmitting the first few packets with HTTP requests and then taking its own format in the following packets. Therefore, it is more reasonable and accurate to recognize the traffic based on flows (series of packets) than based on individual packets.

3.1 Requirement

As what is on demand is an online tool which is supposed to be capable of processing traffic recognition in a high speed link in real time, it is required to have a quite large throughput to make it practically efficient and feasible in use. This tool can be part of a solution to build robust IP networks. Figure 3.1 presents the architecture overview of the tool.

![Figure 3.1: Architecture overview.](image)
3.2 Prototype

There is an internal project which has implemented a prototype of a traffic recognition tool which introduces a novel architecture for robust IP network and proposes a signature-based payload identification method for traffic recognition as well as applies a behavior-based classification approach for anomalous traffic detection.

This thesis work is based on the prototype. Nevertheless, as this thesis work focuses on the traffic recognition, it only concerns the traffic recognition part of the prototype.

3.2.1 Mechanism

By capturing traffic and inspecting packet payload, this tool implements a signature-based approach to identify the underlying application and aggregate packets into flows. And then these flows are clustered in each dimension, i.e. SrcIP, DstIP, SrcPrt and DstPrt, respectively and the significant clusters are extracted according to their communication behavior patterns which are referred to as behavior classes (BC in short) in [6]. Consequently, the prototype is capable of discovering the behavior of the significant clusters and then executing the pre-defined policies on them.

3.2.2 Architecture

The state-of-art architecture of the prototype which is built for a robust IP network is shown in Figure 3.2.

![Figure 3.2: Architecture of prototype](image)
It consists of three main architectural elements: Traffic Analyzer Module (TAM), Policy Decision Point (PDP) and Policy Enforcement Point (PEP). TAM is responsible for capturing traffic, recognizing the underlying application associated with flows and to communicate traffic behaviors to the PDP. The PDP is capable of making a decision based on notification from TAM and defined policies in the repository. A decision specifies whether the behavior of the end host is permitted or denied. The PEP is responsible for enforcing the policy decisions made by the PDP.

It is noticeable that TAM is the focus and task of this thesis work. Therefore, only TAM will be concerned in the following context.

3.2.3 Traffic Analyzer Module (TAM)

It is composed by three components: Recognition Component, Marshalling Component and Profiling Component. They are implemented as three separate programs respectively and run as distinct processes simultaneously. The structure of TAM and interactions among components are shown in Figure 3.3.

![Figure 3.3: Structure of TAM and the interactions among components.](image)

3.2.3.1 Recognition Component

This component is intended to deal with three tasks: traffic capture, payload recognition and flow aggregation.

As a requirement of the first task, this component is designed to work in two modes: online and offline. In the online mode, this component is responsible for capturing frames from network interface in real time. In the offline mode, this component reads frames from trace files in the packet data file format. To fulfill the demand of this thesis work, it should run in online mode. However, the offline mode could be utilized to evaluate the accuracy of the tool. The payload recognition part identifies the underlying applications by inspecting the payload data. The aggregation part aims to aggregate frames into flows.
The input for this component is the traffic coming into the network interface or offline files while the output is a binary file containing recognized flows with application identifiers. There is a specified interval among every output. The detailed illustration of this component is presented in Section 3.2.4.

3.2.3.2 Marshalling Component

This component is designed to transform the presentation of flows in the binary files produced by Recognition Component to a new presentation that can be read by Profiling Component.

The input for this component is the binary file produced by Recognition Component while the output is a text file that can be read by Profiling Component.

3.2.3.3 Profiling Component

The purpose of this component is to discover the anomalous behavior patterns of the flows recognized by Recognition Component, i.e. extract significant clusters of flows in four dimensions respectively, i.e. corresponding to 4-tuple (SrcIP, DstIP, SrcPort and DstPort) and classify the significant flows of each cluster into behavior classes with distinct patterns respectively. The significant clusters of flows are believed to represent potential anomalous traffic.

Every time the input for this component is the text file which contains the group of flows transformed by Marshalling Component while the output is a log file with profiling result and four text files containing flows with anomalous behaviors that are discovered in each dimension. This component may run offline due to it receives data input from Recognition Component and does profiling analysis.

As is mentioned earlier, TAM is the purpose of this thesis work, indeed only Recognition Component is the part of TAM concerns the purpose. Thus, in the following context, only the Recognition Component will be concerned.

3.2.4 Traffic Recognition

TAM recognizes the traffic by inspecting the packet payload in Recognition Component. The recognition is based on identifying characteristic bit strings in the application user data which potentially reflect the protocols that generate the traffic. In order to achieve a significantly accurate approach, TAM is able to capture specified length of packet payload, by default the full payload. To accomplish identification, it implements a signature-based approach which is able to identify the majority of current Internet traffic.
3.2.4.1 Traffic Capture

To monitor the traffic of the specific network, TAM is set up at the edge of the network for being able to monitor the traffic traverses monitored point(s) on the network, i.e. being able to receive all the replicated packets which are sent to or sent by the monitored point(s).

To capture traffic from the Ethernet network interface, the existing solution libpcap library [11], which provides a standard way (userspace programming interface) to access IP data and BPF (Berkeley Packet Filter) packet filtering devices as well as uses a packet data file format which has become an industry standard, is applied for collecting traffic and analyzing payload data. Figure 3.4 illustrates the architecture of TAM’s packet classification.

Figure 3.4: Architecture of TAM’s Packet Classification.

TAM is a userspace application. As a frame is received by the network interface it will be delivered to BPF Driver directly which is a very fast packet copy operation. Then the filter passes it to TAM which is running in the userspace only if it satisfies the user configurable filtering condition, otherwise the filter discards the frame. Thus, being transferred from the network interface to TAM, there is no analyzing on the frame except a basic filter process and context switch.
When the frame arrives at the userspace, TAM which is based on *libpcap* receives it as an Ethernet frame. Then by parsing Ethernet header and IP header of the packet, a new flow is initialized with the transport layer protocol, source and destination IP address. In the case of TCP or UDP, source and destination port are obtained as well and added to the new flow by parsing the header of TCP or UDP header. For ICMP traffic, both the ports are initialized as number 0. Then, a 5-tuple flow has been created. Meanwhile, the payload is obtained as well.

Since TAM is based on *libpcap*, BPF allows the user to only receive the interested packets so that it reduces the unnecessary data transfer from kernel to userspace. This thesis work only focuses on IP packets. Moreover, the maximum length of payload to be captured in each Ethernet frame can be specified, by default 1518 bytes.

### 3.2.4.2 Payload Recognition

However, even with the access to the payload, identification of traffic is not a piece of cake due to it requires a priori knowledge of the protocol patterns. In spite of many protocols are well-documented, others run on top of nonstandard, such as proprietary P2P protocols. Hence, to accomplish the purpose, this thesis work references protocol patterns not only from RFCs and public documents of the protocols but also from empirically derived bit strings from other works, for instance the list of bit strings presented in [12]. Table 3.1 lists a subset of the signature bit strings at the beginning of the payload of the protocols where “0x” implies hex characters.

<table>
<thead>
<tr>
<th>Application</th>
<th>String</th>
<th>Transport protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>BitTorrent</td>
<td>0x13”BitTorrent protocol”</td>
<td>TCP</td>
</tr>
<tr>
<td>SMTP</td>
<td>“220”</td>
<td>TCP/UDP</td>
</tr>
<tr>
<td>FTP</td>
<td>“pass”</td>
<td>TCP/UDP</td>
</tr>
<tr>
<td>SSH</td>
<td>“ssh”</td>
<td>TCP/UDP</td>
</tr>
<tr>
<td>IRC</td>
<td>“USER”</td>
<td>TCP/UDP</td>
</tr>
</tbody>
</table>

Table 3.1: A subset of the bit strings at the beginning of the payload.
The per packet recognition compares the payload of each packet against the array of signature bit strings and associates the corresponding flow with an application in case of a match. Due to one application may have multiple signatures, the prototype has developed a signature-matching identifier which contains a group of signature-based recognition methods; each of them performs identification of all the signatures of the corresponding application. Algorithm 3.1 presents the recognition approach.

Algorithm 3.1: Application-wise signature-based payload recognition (Exhaustive search)

1: Parameters: protocol, payload;
2: Return Value: app;
3: if payload is empty
4: app = "NONPAYLOAD";
5: end if
6: if protocol == TCP
7: if payload belongs to eDonkey
8: app = "eDonkey";
9: end if
10: if payload belongs to BitTorrent
11: app = "BitTorrent";
12: end if
13: // signature matching for other applications
14: app = "TCP"; // none of the applications matches
15: else if protocol == UDP
16: if payload belongs to eDonkey
17: app = "eDonkey";
18: end if
19: if payload belongs to BitTorrent
20: app = "BitTorrent";
21: end if
22: // signature matching for other applications
23: app = "UDP"; // none of the applications matches
24: else if protocol == ICMP
25: app = "ICMP";
26: else
27: app = "OTHER";
28: end if

It is dedicated to identify characteristic bit strings in the packet payload of the new flow by finding a match from one of the candidate recognition methods. If the payload is empty, such as TCP handshaking, it is tagged as "NONPAYLOAD". In case of a match between the payload and one of the methods, the flow is tagged as the corresponding application. If none of the methods matches, i.e. the flow is generated by the application that cannot be recognized so far, the flow is tagged as “TCP” or “UDP” according to its transport layer protocol. The exception is that for ICMP traffic, the prototype just identifies it as “ICMP” rather than inspects its payload. If the transport layer protocol is none of the three concerned protocols, the flow is tagged as “OTHER”. Table 3.2 lists the applications that can be recognized by TAM.
### Table 3.2: Applications that can be recognized by TAM.

<table>
<thead>
<tr>
<th>Category</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web</td>
<td>HTTP</td>
</tr>
<tr>
<td>P2P</td>
<td>BitTorrent, eDonkey, Kazaa, GNU, Goboogy, Soulseek, Ares, WinMX, Mute, Napster, XDCC, Direct Connect, Applejuice, Waste, Soribada</td>
</tr>
<tr>
<td>Management</td>
<td>DNS, NetBios, NBNS, NBDS, Bootstrap</td>
</tr>
<tr>
<td>Data</td>
<td>FTP, MySQL</td>
</tr>
<tr>
<td>Mail</td>
<td>SMTP, POP3</td>
</tr>
<tr>
<td>Chat</td>
<td>IRC, MSN Messenger, Yahoo Messenger, AIM, Skype</td>
</tr>
<tr>
<td>Streaming</td>
<td>RTSP, HTTP Quicktime, HTTP Video, YouTube, Google Video, Zippy Video, HTTP Audio, Veoh, VidiLife</td>
</tr>
<tr>
<td>Encryption</td>
<td>SSL, SSH</td>
</tr>
<tr>
<td>Game</td>
<td>HalfLife, Quake3</td>
</tr>
</tbody>
</table>

3.2.4.3 Flow Aggregation

According to the definition of the flow, the 5-tuple is used as the criterion of the flow aggregation. The aggregation allows the data of the distinct flows accumulated together, such as the number of packets and bytes as well as time. In case of recognition collision with previously recognized flows, for example streaming traffic is relayed over HTTP, a further examination makes them consistent. The aggregation of a flow also leads to the subsequent examination of its counterpart in the reverse direction, if it exists. In addition, a flow expires if it is idle for the specific period.

During running time, TAM maintains a table of flows each entry of which represents a distinct flow. After a flow's application has been identified, the flow needs to be aggregated into the flow table. The operation can be either inserting if it is new or updating if it already exists. It is basically composed of two phases: Firstly, it checks if this flow already exists in the flow table; secondly, it checks if the flow in the reverse direction already exists. Algorithm 3.2 depicts the procedure of the flow aggregation.
Algorithm 3.2: Flow aggregation

1: Parameters: flow, table;
2: Return Value: table;
3: flow_h = find(table, flow); // return index to the same flow in the table
4: rev_flow_h = find_rev(table, flow); // return index to the reversed flow in the table
5: if flow_h == NULL
6:   insert(table, flow);
7:   if rev_flow_h != NULL
8:     verifyApp(flow, rev_flow_h);
9:   end if
10: else if verifyTimeout(flow_h, flow) // flow_h has expired
11:   delete(table, flow_h);
12:   insert(table, flow);
13:   if rev_flow_h != NULL
14:     verifyApp(flow, rev_flow_h);
15:   end if
16: else
17:   update(flow_h, flow); // update data of flow_h (frames, bytes, time etc.)
18:     verifyApp(flow_h, flow);
19:   if rev_flow_h != NULL
20:     verifyApp(flow_h, rev_flow_h);
21:   end if
22: end if

In the first phase, if this flow does not exist in the flow table then it is added to the table and the procedure goes to the second phase; otherwise, TAM updates the data of its counterpart in the flow table and then checks if they share the same application identifier. Note that it is possible to get different application identifiers in the same flow based on signature matching approach, i.e. recognition collision. For instance, a flow is initially recognized as HTTP, afterwards it could indeed be a P2P application (it’s known that some P2P application payload is carried by HTTP requests). In that scenario, TAM executes a verification subroutine which implements a concept of super class to check whether or not a former flow’s identification identifier must be complied with the new one. The subroutine gets two identifiers and determines which one is more particular. That decision is based on basic rules which actually focus on P2P, HTTP and SSL traffic.

In the second phase, if the reverse flow exists, the same verification subroutine is applied to enforce both flows reversed with each other contain the same application identifier; otherwise, nothing needs to be done.

Both phases concentrate on the verification of the application identifiers. It is reasonable to do so since it keeps the application identifiers consistent in the same flow as well as in the reversed flow.
In the middle of the aggregation, if the flow already exists and it has been idle for longer than the specific timeout, its entry in the flow table is deleted and saved to the file on the disk, then the flow is inserted into the table. In this thesis, the commonly accepted 64-second flow timeout [13] is applied, i.e. if no frame arrives in a specific flow for 64 seconds the flow expires which means being flushed out of the table to the file. Moreover, TAM runs another single thread in parallel and periodically for verifying timeout of all the flows in the table.

In detail, for the sake of large numbers of inserting and deleting operations in the flow table, a hash table is used as the data structure of the flow table. It is a chaining hash table which has a linked list of entries at each slot which can be directly addressed by the searched flow and on which a flow can either be inserted or deleted efficiently.
4 Throughput Measurement and Optimization

It is made clear the importance of having an online tool with good accuracy and performance features. As processing at wire speed will be the main issue, the performance should be the main goal, followed by the accuracy which could be taken into account in future work. In this thesis work, the performance is the throughput which is defined as the amount of data per time unit that is processed by TAM.

In this chapter, firstly the correct measurement method is found by attempting different experimental setups, removing negligible components and tuning kernel configurations; and then the throughput is optimized in the algorithmic implementation; finally it is measured in a high speed network.

4.1 Experimental Setup

4.1.1 Equipments

- **PC**
  - Motherboard: *Asus P5GD2-X* (800MHz FSB / *Intel 915P* Chipset / Dual Channel DDR2 533 / PCI-E bus)
  - CPU: *Intel LGA775 Pentium 4* 3.40GHz (1MB L2 cache)
  - Memory: *Corsair* 2GB DDR2
  - Network Interface Card (NIC): *Marvell Yukon 88E8053 PCI-E Gigabit Ethernet Controller*
  - Operating System: *Ubuntu 7.04* (Linux kernel 2.6.20-15)

- **Cisco Catalyst 2900XL Fast Ethernet switch**
  - 10/100 Mbps (Megabit per second) Ethernet
  - Managed layer 2

- **D-Link DGS-3224TGR Gigabit Ethernet switch**
  - 10/100/1000 Mbps Ethernet
  - Managed layer 2

- **IXIA 400T Traffic Generator/Performance Analyzer Instrument**
  - 10/100/1000 Mbps Ethernet, 10 Gigabit Ethernet
  - Operating System: *Windows 2000 Pro*
4.1.2 Traffic

In this thesis, only IP traffic is concerned, thus the testing traffic is generated as Ethernet frames with IP protocols. In the following context, the term “frame” will be referred to Ethernet frame.

<table>
<thead>
<tr>
<th>Ethnet Frame</th>
<th>Raw Frame Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added by transmitter</td>
<td>Data Frame Size</td>
</tr>
<tr>
<td>Removed by receiver</td>
<td>User’s transmission</td>
</tr>
<tr>
<td>Preamble</td>
<td>MAC Header</td>
</tr>
<tr>
<td>8 bytes</td>
<td>14 bytes</td>
</tr>
</tbody>
</table>

**Figure 4.1: Structure of an Ethernet Frame.**

Figure 4.1 is depicts the structure of an Ethernet frame in accordance with the present standard. Referring to Figure 4.1, an Ethernet frame includes a “Preamble” field of 8 bytes used for frame synchronization or physical stabilization including a SFD (Start Frame Delimiter) of 1 byte, a “MAC Header” filed composed of the destination and source MAC addresses each with 6 bytes as well as a type indicator of 2 bytes presenting the protocol, a “Data” field, an optional “Pad” field used when frame size is smaller than the minimum frame size, a “CRC” (Cyclic Redundancy Check) field of 4 bytes and an “IFG” (Inter Frame Gap) field of more than 12 bytes. However, neither “Preamble” field nor “IFG” field accounts for the length of the frame. Hence, the term “Frame size” appears in this thesis is used to refer to “Data Frame Size” instead of “Raw Frame Size” illustrated in Figure 4.1.

Referring to the standard, the frame structure is compatible with both Fast Ethernet and Gigabit Ethernet. The difference between them is the minimum size of the frame which is 64 bytes for Fast Ethernet but 512 bytes for Gigabit Ethernet. If the frame size is smaller than the minimum size, extra bytes will be padded on it in the “Pad” field. As another part of the standard, there is a gap (IFG) which is no smaller than 12-byte slot-time between any two frames. In the following experiments, IFG is equal to 12-byte slot-time. Thus, transmitting a 64-byte frame in the Fast Ethernet produces a data rate of

\[
\text{DataRate} = \frac{\text{FrameSize} \times \text{Bandwidth}}{\text{RawFrameSize}} = \frac{64 \times 100}{8 + 64 + 12} = 76.19 \text{ Mbps} \quad (1)
\]
The "Data" field in Figure 4.1 consists of network layer header which is IP header with 20 bytes, transport layer header and payload. In this thesis, the transport layer protocol is one of TCP with a 20-byte header, UDP with an 8-byte header or ICMP with an 8-byte header. For instance, for a 64-byte TCP frame, the length of “Data” field is

\[
\text{DataLen} = \text{FrameSize} - \text{MacHeader} - \text{CRC} = 64 - 14 - 4 = 46 \text{ bytes}
\]

the length of payload is

\[
\text{PayloadLen} = \text{DataLen} - \text{IPHeader} - \text{TCPHeader} = 46 - 20 - 20 = 6 \text{ bytes}
\]

Figure 4.2 lists “Data” field for a 64-byte TCP packet.

<table>
<thead>
<tr>
<th>TCP Packet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>Data</td>
</tr>
<tr>
<td>IP Header</td>
</tr>
<tr>
<td>20 bytes</td>
</tr>
</tbody>
</table>

Figure 4.2: Structure of “Data” field for a 64-byte TCP Packet.

4.1.3 Metrics

4.1.3.1 Frame Rate

The main metric used to assess the throughput is frame rate, i.e. fps (frames per second), which indicates the capacity of TAM for processing frames per time unit. It is

\[
\text{FrameRate} = \frac{\text{NumberOfFrames}}{\text{Time}}
\]

In the experiments, frame rate will be measured by accumulating the number of frames in a certain period and then calculated by Equation (4).

In addition, frame rate can be understood like

\[
\text{FrameRate} = \frac{\text{ProcessingRate}}{\text{RawFrameSize}}
\]
Equation (5) can not be measured or calculated directly due to TAM’s processing speed is unknown. However, it indicates that frame rate is the combination of the processing rate of TAM and frame size. The processing rate is the recognition rate which is influenced by frame transmitting rate, bandwidth, network interferences like latency and frame loss rate, and packet loss rate by the kernel. Thus, all of these factors should be taken into account before drawing conclusions of TAM’s performance and will be described later.

4.1.3.2 Maximum Frame Rate

Maximum frame rate is the ideal peak frame rate with transmitting at full bandwidth and receiving without any interference, which means the processing rate is equal to the bandwidth. Then it can be written as

\[ \text{MaxFrameRate} = \frac{\text{Bandwidth}}{\text{RawFrameSize}} \]  \hspace{1cm} (6)

4.1.3.3 Bandwidth Utilization Rate

Bandwidth utilization rate is defined as the proportion of the bandwidth which is utilized by TAM. Formally, it is

\[ \text{BandwidthUtilRate} = \frac{\text{FrameRate}}{\text{MaxFrameRate}} \]  \hspace{1cm} (7)

Frame rate shows the capacity in terms of absolute values and is influenced by frame size. The larger the frame is, the lower frame rate is, and so is maximum frame rate. In contrast, bandwidth utilization rate states the capacity by percentages in order to reveal the comparative differences of throughputs among the frames with different sizes.

4.1.3.4 Frame Transmitting Rate

Frame transmitting rate is the frame rate that is transmitted by the traffic generator. It can be measured by dumping traffic at the end-system which transmits traffic by tcpdump [14] in certain period, and then calculating the rate. In addition, the IXIA instrument has the functionality displaying frame transmitting rate when transmitting frames.
4.1.3.5 Latency

In a packet-switched network, latency is a time delay between a packet is sent and received, which depends on factors like bandwidth and receiving and retransmitting delay in routers and switches. It indicates the network efficiency. Therefore, the latency may impact TAM's performance to some extent and should be measured at first.

Latency can be measured either one-way (the time from the source sending a packet to the destination receiving it), or round-trip (the one-way latency from source to destination plus the one-way latency from the destination back to the source). The well-known ping utility [15] can be facilitated to measure round-trip latency from a single point. Meanwhile, ping performs no packet processing; it merely sends a response back when it receives a packet, thus it is a relatively accurate way of measuring latency. In addition, the IXIA instrument has the functionality displaying latency when transmitting frames.

If a high latency is detected, then certain network element is interfering with TAM's performance. Then further improvement can be taken on the network element to gain better performance over TAM.

4.1.3.6 Frame Loss Rate

Frame loss rate is the proportion that the number of frames lost during the transmission before arriving the receiver's kernel accounts for the number of frames transmitted by the traffic generator. Formally, it is

\[
FrameLossRate = \frac{TransmittedFrames - ReceivedFrames}{TransmittedFrames}
\]  

(8)

It reflects the traffic loss on the wire, which may be caused by receiving and retransmitting delay in routers and switches as well as network interface driver. Thus it indicates the network efficiency as well and should be considered before the measurements.

tcpdump is able to show the number of packets received by BPF. As is shown in Figure 3.4, the frames are sent to BPF directly by a simple packet copy when received by the network interface so that the transfer overhead between them can be ignored. Thus frame loss rate can be measured by dumping traffic at both end-systems by tcpdump in a certain period, and then comparing their numbers of frames. In addition, the IXIA instrument has the functionality displaying the number of transmitted (received) frames when transmitting (receiving) frames.
4.1.3.7 Packet Loss Rate

Packet loss rate is the proportion of packets dropped by the receiver’s kernel (or not received by userspace) accounting for packets received by the receiver’s kernel. Formally, it is

\[
\text{PacketLossRate} = \frac{\text{DroppedFrames}}{\text{ReceivedFrames}}
\]  

(9)

Figure 3.4 shows after a frame is processed by the filter, it is delivered to the userspace. Whereas, there exists a costly context switch exchanging data from kernel to userspace which may cause packet loss. In other words, packet loss rate presents the proportion of packets lost between BPF and TAM. Consequently, the combination of frame loss rate and packet loss rate indicates the packet drop between the traffic generator and TAM which helps discover influences from external factors and then identify TAM’s inherent performance.

Packet loss happens when kernel packet queue (buffer) overflows due to userspace application cannot cope with kernel speed. It depends on some factors, such as filtering condition, hardware and kernel configuration. If it is assumed that all the packets received by the filter can be successfully filtered, packet loss rate reflects the performance of the end-system which runs TAM. Therefore, it is a very important factor to be taken into account before drawing conclusions of TAM’s performance.

Considering the definition of packet loss rate, a basic packet capture userspace application based on \textit{libpcap} such as \textit{tcpdump} that counts and discards, with no further analysis, the captured packets can be utilized to measure packet loss rate.

4.1.3.8 Capacity Utilization Rate

Since TAM’s performance is influenced by some external factors, in order to draw more reasonable conclusion, capacity utilization rate is introduced. It is defined as the proportion of the end-system’s userspace capacity that is utilized by TAM which reveals TAM’s real performance that is independent of these external factors. It complements the disadvantage of bandwidth utilization rate. The end-system’s userspace capacity is the userspace packet receiving rate, thus compared with Equation (7), capacity utilization rate it is

\[
\text{CapacityUtilRate} = \frac{\text{FrameRate}}{\text{FrameTxRate} \times (1 - \text{FrameLossRate}) \times (1 - \text{PacketLossRate})}
\]  

(10)

(\text{FrameTxRate} is frame transmitting rate)
It is noticeable that if frame transmitting rate is equal to maximum frame rate and both frame loss rate and packet loss rate are 0, then capacity utilization rate is equal to bandwidth utilization rate.

### 4.1.4 Measurement Point

In order to get frame rate, instead of using other tools, TAM itself is modified to involve an internal measurement point in Recognition Component for outputting its throughput every constant period. This measurement point accumulates the number of frames every time TAM processes a frame. When a period elapses, the measurement point calculates the frame rate according to Equation (4) and outputs the frame rate to a file in the disk. Figure 4.3 illustrates the position of the measurement point placed in TAM.

![Figure 4.3: Position of Measurement Point in TAM.](image)

The mode of measurement point (on/off), the length of period and the location of the measurement output file can be specified as parameters by the users which will be described in Section 4.1.5.1.

### 4.1.5 Configuration

#### 4.1.5.1 TAM

Throughput has to be measured in online mode for the purpose of this thesis work. Thus, the executive parameter of TAM should be set to online mode.

Time out for expired flows to be flushed out of the memory is set to 30 seconds (time bin) and the number of time bins to generate a new output binary file name is set to 1. The binary files store the classified and expired flows. Both of them are set according to a lot of experimental results, which means they are believed to produce representative results. If they are not set, their default values are 300 and 0 (which means all day long).
In order to measure throughput, the mode of measurement point should be turned on otherwise throughput will not be output. The length of interval to output throughput is the length of the time out mentioned previously. The location of the output file can be decided by the user, by default it is the file “throughput” in the current directory.

By using libpcap library, BPF can be utilized by tcpdump expressions to only receive the interested traffic. The expression can be specified by the user. By default, the expression is “ip” which means TAM receives all IP traffic.

The snapshot length which determines how much of each frame should be captured can be specified. Due to analyzing traffic at the application level, TAM requires more payload of each packet to produce the more accurate result. The default snapshot length is set to 1518 bytes. However, the smaller the snapshot length, the less data per accepted packet needs to be copied to the user space by the packet filter, which aids in accelerating packet processing and avoiding loss. Hence, this parameter may be set to a threshold which is believed to be a tradeoff between accuracy and efficiency.

4.1.5.2 Instrument

The IXIA instrument contains two types of network interfaces (Fast Ethernet and Gigabit Ethernet) each of which has several ports and can simulate traffic transmission among them. Thus, it can play a role as both traffic transmitter and receiver.

Traffic transmitting rate can be tuned between 0 and bandwidth depending on the type of interface. Setting the rate as the peak value is supposed to reflect clearly how TAM performs under the full loading circumstance while tuning the rate at a certain values helps understand its performance in certain conditions.

In order to generate traffic similar to the reality more or less, in each experiment two Ethernet ports with same type are selected and are enforced to transmit IP traffic to each other. In addition, destination and source IP addresses of the packets are configured in IXIA’s user interface to enforce the number of packets are distributed equally in most of the flows only with a minority of the flows containing significantly more packets. Table 4.1 states the configuration which keeps constant in all the experiments and results in approximate 3990 flows and 30 source IP clusters, 1 source port cluster and 1 destination port cluster per time bin (30 seconds). In Table 4.1, “DstIP” (“SrcIP”) stands for initial destination IP (source IP), “Mode” is the increasing or decreasing trend like “Cont. Incr. Host” means continuously increment host, “Repeat” represents that how many times one address is repeated, “Class” is the indicator of IP address type and “Mask” is the network mask.
Table 4.1: Configuration of DstIP and SrcIP fields in IXIA.

<table>
<thead>
<tr>
<th>Port 1</th>
<th>DstIP Mode</th>
<th>Repeat</th>
<th>Class</th>
<th>Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.1.253</td>
<td>Decr. Host</td>
<td>30</td>
<td>C</td>
<td>255.255.255.0</td>
</tr>
<tr>
<td>Port 2</td>
<td>DstIP Mode</td>
<td>Repeat</td>
<td>Class</td>
<td>Mask</td>
</tr>
<tr>
<td>10.0.0.1</td>
<td>Incr. Host</td>
<td>50</td>
<td>C</td>
<td>255.255.255.0</td>
</tr>
</tbody>
</table>

In addition, frame size and transport layer protocol can be set in each experiment according to different purposes.

4.2 Results Overview

Table 4.2 provides an overview of the measurement, which illustrates the procedure of finding the correct setup as well as the effect of coding optimization. In Table 4.2, each row represents every single experiment, the first two columns states the sequential number of the experiments (#) and frame size (FS), the following six columns lists frame rate (FR) and capacity utilization rate (CUR) for three protocols respectively and the last column gives a brief description of each experiment. Figure 4.4 plots the trends of capacity utilization rate accompany with bandwidth tuning, which will be explained later in the context.

<table>
<thead>
<tr>
<th>#</th>
<th>FS</th>
<th>ICMP</th>
<th>TCP</th>
<th>UDP</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>FR</td>
<td>CUR</td>
<td>FR</td>
<td>CUR</td>
</tr>
<tr>
<td>1</td>
<td>64</td>
<td>8000</td>
<td>1</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
<td>8000</td>
<td>1</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>3</td>
<td>64</td>
<td>146700</td>
<td>0.99</td>
<td>28700</td>
<td>0.19</td>
</tr>
<tr>
<td>4</td>
<td>64</td>
<td>148809</td>
<td>1</td>
<td>35900</td>
<td>0.24</td>
</tr>
<tr>
<td>5</td>
<td>64</td>
<td>148809</td>
<td>1</td>
<td>42400</td>
<td>0.28</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>148809</td>
<td>1</td>
<td>48100</td>
<td>0.32</td>
</tr>
<tr>
<td>7</td>
<td>64 148809</td>
<td>1</td>
<td>95600</td>
<td>0.64</td>
<td>136500</td>
</tr>
<tr>
<td>300 39062</td>
<td>1</td>
<td>13600</td>
<td>0.35</td>
<td>27500</td>
<td>0.70</td>
</tr>
<tr>
<td>512 23496</td>
<td>1</td>
<td>7500</td>
<td>0.32</td>
<td>17500</td>
<td>0.74</td>
</tr>
<tr>
<td>700 17361</td>
<td>1</td>
<td>5500</td>
<td>0.33</td>
<td>12200</td>
<td>0.71</td>
</tr>
<tr>
<td>900 13587</td>
<td>1</td>
<td>4400</td>
<td>0.32</td>
<td>9800</td>
<td>0.72</td>
</tr>
<tr>
<td>1500 8224</td>
<td>1</td>
<td>2700</td>
<td>0.33</td>
<td>6000</td>
<td>0.73</td>
</tr>
</tbody>
</table>

| 8 | 512 195200 | 0.90 | 8100 | 0.03 | 16200 | 0.07 |
| 700 161400 | 0.93 | 6100 | 0.04 | 10900 | 0.06 |
| 900 135869 | 1 | 4100 | 0.03 | 7600 | 0.06 |
| 1500 82237 | 1 | 2600 | 0.03 | 6100 | 0.07 |

| 9 | 512 140977 | 1 | 8000 | 0.06 | 15400 | 0.11 |
| 700 104167 | 1 | 5700 | 0.05 | 12100 | 0.12 |
| 900 81521 | 1 | 4300 | 0.05 | 9500 | 0.12 |
| 1500 49342 | 1 | 2400 | 0.05 | 5400 | 0.11 |

| 10 | 512 70489 | 1 | 7800 | 0.11 | 15300 | 0.22 |
| 700 52084 | 1 | 5500 | 0.11 | 11000 | 0.21 |
| 900 40761 | 1 | 4400 | 0.11 | 9300 | 0.23 |
| 1500 24671 | 1 | 2500 | 0.10 | 5600 | 0.23 |

| 11 | 512 211600 | 0.90 | 17600 | 0.08 | 41800 | 0.18 |
| 700 165200 | 0.95 | 18400 | 0.11 | 40500 | 0.23 |
| 900 135869 | 1 | 21400 | 0.16 | 42700 | 0.31 |
| 1500 82237 | 1 | 20500 | 0.25 | 38500 | 0.47 |

Table 4.2: Throughput Measurement Overview.
4.3 Experiments

4.3.1 Experiment 1: Getting Started

4.3.1.1 Issue

To get started, traffic can be generated to traverse the network interface which TAM monitors, for example making a host send simple requests to TAM, to see the throughput. However, before that frame transmitting rate, latency, frame loss rate and packet loss rate need to be measured in order to check if the setup is interfering. ping can be used to measure latency, while tcpdump can be utilized to measure frame transmitting rate, frame loss rate and packet loss rate.

4.3.1.2 Experiment

In this experiment, two PCs are chosen, one of them acts as the traffic generator, i.e. transmitting traffic by ping to the other; the other PC is supposed to act as TAM and to receive ping requests. In addition, a Cisco Catalyst 2900XL Fast Ethernet switch is applied as the gateway for communication between the PCs. Figure 4.5 shows the setup of the experiment.

Figure 4.4: Bandwidth versus Capacity Utilization Rate.
The reason that throughput is measured by ping is that ping request is carried by ICMP protocol which is the simplest protocol to be identified by TAM. If it cannot reach peak throughput, there should be some bottlenecks in TAM, such as the kernel parameters or the hardware, so that it is unnecessary to test TCP or UDP protocol.

Firstly, the transmitting host and the receiving host are synchronized with the time server which can help identify. After that, the traffic transmitting end-system sends ping requests to the other one, and then the number of transmitted packets and the latency can be seen from the output of ping.

The end-system acting as TAM just simply conducts a basic packet capture tcpdump which just counts and discards the received packets so as to see frame loss rate and packet loss rate. Then the generating end-system generates traffic once again, and TAM is executed on the other end-system to check throughput. To get a representative result, throughput is the average value of 30 results from the measurement point.

ping command performs for 5 minutes. The size of payload is set to 18 bytes which translates into a 64-byte frame. The interval between sending each frame is set to 0 which results in transmitting frames as fast as they come back or one hundred times per second, whichever is more. Quiet output is used, i.e. nothing is displayed except the summary lines at startup time and when finished, in order to reduce the number of I/O operations.

4.3.1.3 Result

Table 4.3 lists the performance-dependent metrics and Table 4.4 displays the throughput measurement where “FS” stands for frame size, “FR” is frame rate, “BUR” accounts for bandwidth utilization rate and “CUR” represents capacity utilization rate. “FS” is in terms of bytes, “Maximum Frame Rate”, “Frame Transmitting Rate” and “FR” is in units of fps.
### Table 4.3: Performance-dependent Metrics in Experiment 1.

<table>
<thead>
<tr>
<th>FS</th>
<th>Latency</th>
<th>Maximum Frame Rate</th>
<th>Frame Transmitting Rate</th>
<th>Frame Loss Rate</th>
<th>Packet Loss Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>Low</td>
<td>148809</td>
<td>8000</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table 4.4: Throughput in Experiment 1.

<table>
<thead>
<tr>
<th>FS</th>
<th>ICMP</th>
<th>TCP</th>
<th>UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FR BUR CUR</td>
<td>FR BUR CUR</td>
<td>FR BUR CUR</td>
</tr>
<tr>
<td>64</td>
<td>8000 0.05 1</td>
<td>/ / /</td>
<td>/ / /</td>
</tr>
</tbody>
</table>

Output from the transmitting end-system:

```
root# ping 192.36.157.176 -w 300 -s 18 -i 0 -q
PING 192.36.157.176 (192.36.157.176) 18(46) bytes of data.
--- 192.36.157.176 ping statistics ---
1200297 packets transmitted, 1200297 received, 0% packet loss, time 300000ms
rtt min/avg/max/mdev = 0.174/0.235/3.915/0.005 ms, ipg/ewma 0.249/0.234 ms
```

Output from the receiving end-system:

```
root# tcpdump -i eth0 -w /dev/null –s 0
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
1200297 packets captured
1200297 packets received by filter
0 packets dropped by kernel
```

From the summary of ping, average round trip time is 0.235 ms, which indicates a very low latency. Meanwhile, the number of transmitted packets can be seen and then frame transmitting rate can be calculated. However, that is not the total number of packets traversing the network interface of TAM due to the property of ping. Whenever TAM receives an ECHO_REQUEST datagram, it has to reply an ECHO_RESPONSE datagram. And TAM monitors bi-directional traffic (transmitted and received) of the network interface. Hence,

\[
FrameTxRate = \frac{1200297 + 1200297}{300} = 8000 \text{ fps}
\]

On the other end, the result of tcpdump shows the number of received packets is equal to that transmitted by ping, which results in

\[
FrameLossRate = \frac{1200297 - 1200297}{1200297} = 0
\]
Then it can be concluded that the Fast Ethernet switch and the network interface provides sufficient network efficiency at the current rate which does not influence the experiment.

Moreover, the output of tcpdump shows no packets are dropped by the kernel.

\[ \text{PacketLossRate} = 0 \]

And then the output of throughput shows that

\[ \text{FrameRate} = 8000 \text{ fps} \]

According to Equation (6),

\[ \text{MaxFrameRate} = \frac{\text{Bandwidth}}{\text{RawFrameSize}} = \frac{100 \times 10^6}{(8 + 64 + 12) \times 8} = 148809 \text{ fps} \]

According to Equation (7),

\[ \text{BandwidthUtilRate} = \frac{\text{FrameRate}}{\text{MaxFrameRate}} = \frac{8000}{148809} = 0.05 \]

According to Equation (10),

\[ \text{CapacityUtilRate} = \frac{\text{FrameRate}}{\text{FrameTxRate} \times (1 - \text{FrameLossRate}) \times (1 - \text{PacketLossRate})} = \frac{8000}{8000 \times (1 - 0) \times (1 - 0)} = 1 \]

The low bandwidth utilization rate is caused by low traffic generating rate, however capacity utilization rate indicates that TAM utilizes the full capacity.

### 4.3.2 Experiment 2: At the Edge of the Network

#### 4.3.2.1 Issue

In Experiment 1, TAM only monitors one PC’s traffic and in a P2P way. However, the purpose of TAM is to monitor the traffic in one or more network segments instead of only one user. As is illustrated in Figure 3.1, TAM is supposed to be put at the edge of the network to probe all the traffic exchanged with Internet and within the local network. Hence, a similar topology needs to be simulated.
4.3.2.2 Experiment

Another PC which represents another user is added to receive ping requests and to form a LAN. The new setup is presented in Figure 4.6, where TAM is responsible recognizing the traffic traversing the switch (gateway).

For making TAM probing these two users’ traffic like a sniffer, the switch is configured to mirror the traffic received by the ports connected to the two users to the port connected to TAM. This configuration is accomplished by the SPAN (The Switched Port Analyzer) feature [16] of Cisco Catalyst 2900XL Fast Ethernet switch. Its principle is propagating every data delivery traversing the monitored ports to the monitoring port.

![Figure 4.6: Setup of Experiment 2. Traffic traversing two hosts is mirrored to TAM.](image)

Firstly, all the three hosts synchronized with the time server. Then the generator sends ping requests to the receiver, while the end-system acting as TAM runs tcpdump to check frame loss rate and packet loss rate. Finally, the generator sends ping requests again, and TAM runs to see throughput. The setting of parameters for ping is the same as Experiment 1.

4.3.2.3 Result

Table 4.5 lists the performance-dependent metrics and Table 4.6 displays the throughput measurement.

<table>
<thead>
<tr>
<th>FS</th>
<th>Latency</th>
<th>Maximum Frame Rate</th>
<th>Frame Transmitting Rate</th>
<th>Frame Loss Rate</th>
<th>Packet Loss Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>Low</td>
<td>148809</td>
<td>8000</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.5: Performance-dependent Metrics in Experiment 2.
<table>
<thead>
<tr>
<th>FS</th>
<th>ICMP</th>
<th>TCP</th>
<th>UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FR</td>
<td>BUR</td>
<td>CUR</td>
</tr>
<tr>
<td>64</td>
<td>8000</td>
<td>0.05</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.6: Throughput in Experiment 2.

From the summary of ping, average round trip time is still 0.235 ms, which means a low latency. Meanwhile, the number of transmitted packets can be seen and then frame transmitting rate can be calculated as 8000 fps. The output of tcpdump shows the number of received packets is equal to that transmitted by ping so that frame loss rate is 0 and latency for mirroring traffic is very low. Thus, the Fast Ethernet switch is not interfering. In the mean time, tcpdump displays no packets are dropped. And then frame rate is measured as 8000 fps.

Therefore,

$$\text{BandwidthUtilRate} = \frac{8000}{148809} = 0.05$$

$$\text{CapacityUtilRate} = \frac{8000}{8000 \times (1-0) \times (1-0)} = 1$$

The low bandwidth utilization rate and full capacity utilization rate are the same as Experiment 1. The purpose of startup is accomplished by constructing a simulated experimental environment.

### 4.3.3 Experiment 3: Heavy Traffic Load

#### 4.3.3.1 Issue

Obviously ping cannot generate very frequent traffic by which throughput should be measured. Since ping is not capable of generating traffic frequently, another effective method should be applied to replace ping.
4.3.3.2 Experiment

A new hardware instrument *IXIA 400T Traffic Generator/Performance Analyzer* which is able to generate traffic as fast as 1 Gbps (Gigabit per second) is introduced. In this experiment, it is just enforced to generate 100 Mbps traffic to check throughput in low bandwidth so that two of its Fast Ethernet ports are connected to Fast Ethernet switch. Each of the ports transmits traffic to the other with 5% speed (50 Mbps) and then the real traffic bit rate traversing the network interface of TAM is bandwidth (100 Mbps). TAM is able to probe the traffic generated by both ports of the instrument. Figure 4.7 illustrates the setup. In the configuration of the instrument, ICMP protocol is selected and frame size is set to 64 bytes. If TAM produces a high frame rate, then TCP and UDP protocol needs to be measured as well.

![Traffic generator instrument](image1)

**Figure 4.7: Setup of Experiment 3. Traffic traversing instrument is mirrored to TAM.**

Firstly, the time of TAM and instrument are synchronized and latency between these two *IXIA*’s ports is measured by *IXIA*. Then both ports generate traffic for 3 minutes while the end-system acting as TAM runs *tcpdump* in order to check frame loss rate and packet loss rate, in the mean time, frame transmitting rate is calculated based on the number of transmitted frames from each port which is shown in its user interface. After that these two ports generate continuous traffic, then TAM’s throughput is measured.

4.3.3.3 Result

Table 4.7 lists the performance-dependent metrics and Table 4.8 displays the throughput measurement.

<table>
<thead>
<tr>
<th>FS</th>
<th>Latency</th>
<th>Maximum Frame Rate</th>
<th>Frame Transmitting Rate</th>
<th>Frame Loss Rate</th>
<th>Packet Loss Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>Low</td>
<td>148809</td>
<td>148809</td>
<td>0</td>
<td>0.0027</td>
</tr>
</tbody>
</table>

*Table 4.7: Performance-dependent Metrics in Experiment 3.*
Table 4.8: Throughput in Experiment 3.

<table>
<thead>
<tr>
<th>FS</th>
<th>ICMP</th>
<th>TCP</th>
<th>UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FR</td>
<td>BUR</td>
<td>CUR</td>
</tr>
<tr>
<td>64</td>
<td>146700</td>
<td>0.99</td>
<td>0.99</td>
</tr>
</tbody>
</table>

IXIA shows that latency between two ports is very low. From TAM, tcpdump outputs:

```
root# tcpdump -i eth0 -w /dev/null -s 0
tcpdump: listening on eth0, link-type EN10MB (Ethernet),
capture size 65535 bytes
26712930 packets captured
26785620 packets received by filter
72690 packets dropped by kernel
```

As is shown in each port’s user interface, the number of transmitted frames from port 1 is equal to the number of received frames from port 2, and vice versa. Then the total number of transmitted frames can be calculated as 26785620 which is equal to the number of packets received by filter so that frame loss rate is 0 and it can be concluded that the Fast Ethernet switch does not impact the experiment under bandwidth of 100 Mbps. Meanwhile, tcpdump displays 72690 packets are dropped. And then frame rate is measured as 146700 fps. Hence,

\[
PacketLossRate = \frac{72690}{26785620} = 0.0027
\]

\[
FrameTxRate = \frac{26785620}{180} = 148809 \text{ fps}
\]

\[
BandwidthUtilRate = \frac{146700}{148809} = 0.99
\]

\[
CapacityUtilRate = \frac{146700}{148809 \times (1 - 0) \times (1 - 0.0027)} = 0.99
\]

The result shows TAM produces a high frame rate and utilizes almost full bandwidth and capacity when processing ICMP protocol in Fast Ethernet network. Bandwidth utilization rate is approximately equal to capacity utilization rate because frame transmitting rate is equal to maximum frame rate, frame loss rate is 0 and packet loss rate is very close to 0 (comparing Equation (7) and (10)). Compared with the previous measurements, the improvement in figures comes from the heavier traffic load generated by the instrument than by ping. As a result, it becomes necessary to measure throughputs of TCP and UDP protocol.
Since either latency, frame loss rate, packet loss rate or frame transmitting rate is independent with the transport protocol so that they keep constant when the protocol is changed to TCP or UDP. Then there is no need measuring them again. Measurements of frame rate for TCP and UDP are 28700 fps and 33100 fps respectively, and by calculation, both of them account for around 20% performance, which are stated in Table 4.8.

One fact is that TAM processes ICMP traffic much efficiently than TCP or UDP traffic. The reason is that the traffic recognition procedure is only applied for TCP and UDP frame, whereas, for ICMP frames, TAM omits the recognition. Another aspect is that TAM processes UDP traffic more efficiently than TCP traffic. It happens due to the inherent recognition procedure for TCP traffic is more complex than that of UDP traffic, in details, it contains a larger number of signature candidates and they are more complicated.

In this experiment, there is some packet loss between TAM's kernel and userspace which impacts TAM's throughput.

4.3.4 Experiment 4: Kernel Configuration

4.3.4.1 Issue

As is described in Section 4.1.3.7, packets are dropped due to kernel buffer overflow depending on some factors, such as filtering condition, hardware and kernel configuration. Since in the measurements, the filter is always set as default which receives all IP traffic so that no packet can be dropped by filtering condition. Moreover, between hardware and kernel configuration, the latter is easy to achieve so that the kernel parameters can be configured at first before concerning other issues.

4.3.4.2 Experiment

The hardware setup is the same as Experiment 3 which is shown in Figure 4.7. The instrument still generates 64-byte ICMP, TCP and UDP frames with the same transmitting rate respectively.

The kernel parameters of the end-system which runs TAM are set to the appropriate values. Then packet loss rate, frame transmitting rate and throughput are measured like the way taken in Experiment 3.

4.3.4.3 Result

Table 4.9 lists the performance-dependent metrics and Table 4.10 displays the throughput measurement.
Table 4.9: Performance-dependent Metrics in Experiment 4, 5 and 6.

<table>
<thead>
<tr>
<th>FS</th>
<th>Latency</th>
<th>Maximum Frame Rate</th>
<th>Frame Transmitting Rate</th>
<th>Frame Loss Rate</th>
<th>Packet Loss Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>Low</td>
<td>148809</td>
<td>148809</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.10: Throughput in Experiment 4.

<table>
<thead>
<tr>
<th>FS</th>
<th>ICMP</th>
<th>TCP</th>
<th>UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FR</td>
<td>BUR</td>
<td>CUR</td>
</tr>
<tr>
<td>64</td>
<td>148809</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Because neither network nor instrument setups has been changed, latency, frame loss rate and frame transmitting rate are the same as Experiment 3.

Firstly, checking TAM’s default and maximum kernel receiving socket buffer sizes:

root# cat /proc/sys/net/core/rmem_max
110592
root# cat /proc/sys/net/core/rmem_default
110592

After that, enlarging them to 320 MB:

root# echo 335544320 > /proc/sys/net/core/rmem_max
root# echo 335544320 > /proc/sys/net/core/rmem_default

Then packet loss rate is measured by tcpdump:

root# tcpdump -i eth0 -w /dev/dulll –s 0
tcpdump: listening on eth0, link-type EN10MB (Ethernet),
capture size 65535 bytes
26785620 packets captured
26785620 packets received by filter
0 packets dropped by kernel

It indicates packet loss rate is 0. Therefore, the resizing of buffer size does help capture high speed traffic. And then frame rate, bandwidth utilization rate and capacity utilization rate for three protocols is measured and calculated separately which are listed in Table 4.10.

The result shows TAM’s throughput benefits from the extension of the socket buffer. TAM produces a full throughput for ICMP and 5% larger throughput for both TCP and UDP compared with default buffer size.

The configured buffer size 320 MB is appropriate value selected from several different attempts. Meanwhile, another parameter, kernel receiver backlog, which is located in /proc/sys/net/core/netdev_max_backlog, seems to be related. However, there is no obvious effect by tuning it.
4.3.5  Experiment 5: No Profiling

4.3.5.1  Issue

As the correct network, instrument and kernel configurations have been found, it is time to look into TAM itself. As is illustrated in Section 3.2.3, TAM consists of three components which run in three distinct processes and two of them plays main roles and communicate with each other through the third component. Meanwhile, one of the main components, Profiling Component, is not the purpose of this thesis. On the other hand, it accepts input from local files which indicates it runs in offline mode, thus it is unnecessary to enclose it in the online throughput measurement. As a result, Profiling Component is deselected from TAM and only Recognition Component and Marshalling Component will be involved in the following measurements.

4.3.5.2  Experiment

The experimental setup is shown in Figure 4.8. The network, instrument and kernel configurations are the same as Experiment 4 while the internal implementation of TAM is modified by removing Profiling Component.

![Figure 4.8: Setup of Experiment 5. Traffic traversing instrument is mirrored to TAM.](image)

4.3.5.3  Result

<table>
<thead>
<tr>
<th>FS</th>
<th>ICMP</th>
<th>TCP</th>
<th>UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FR</td>
<td>BUR</td>
<td>CUR</td>
</tr>
<tr>
<td>64</td>
<td>148809</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.11: Throughput in Experiment 5.
Because none of network, instrument or kernel configurations has been changed, all performance-dependent metrics are the same as Experiment 4 which is shown in Table 4.9.

Table 4.11 lists throughput measurements. The throughput of ICMP is still full while each of TCP and UDP increases about 6500 fps which accounts for 4% of the total capacity. The improvement comes from the release of the overhead and resources used to be occupied by Profiling Component whose complexity is \(O(n_1 + n_2 + n_3 + n_4)\) where \(n_1\), \(n_2\), \(n_3\) and \(n_4\) are the number of clusters for source IP, destination IP, source port and destination port. Thus Profiling Component’s overhead depends on the distribution of the 4-tuple among the traffic. In this thesis work, the testing traffic produces 30 source IP clusters, 1 source port cluster and 1 destination port cluster per time bin (30 seconds).

### 4.3.6 Experiment 6: No Marshalling

**4.3.6.1 Issue**

As described in Section 3.2.3.2, Marshalling Component is intended to transform flow representations between other two components, i.e. getting input from files produced by Recognition Component and producing output as files for Profiling Component. Thus, Marshalling Component runs in offline mode which can be got rid of in the online throughput measurement as well consequently reducing some amount of I/O disk routine overhead. Thus, for the sake of online throughput measurement, only Recognition Component will be focused in the following phases.

**4.3.6.2 Experiment**

The setup is shown in Figure 4.9. The network, instrument and kernel configurations are the same as Experiment 4 while the internal implementation of TAM is modified by removing Marshalling Component.
Figure 4.9: Setup of Experiment 6 and 7. Traffic traversing instrument is mirrored to TAM.

4.3.6.3 Result

<table>
<thead>
<tr>
<th>FS</th>
<th>ICMP</th>
<th>TCP</th>
<th>UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FR</td>
<td>BUR</td>
<td>CUR</td>
</tr>
<tr>
<td>64</td>
<td>148809</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.12: Throughput in Experiment 6.

Because none of network, instrument or kernel configurations has been changed, all performance-dependent metrics are the same as Experiment 4 which is shown in Table 4.9.

Table 4.12 lists throughput measurements. TAM’s performance for processing TCP and UDP frames improves another 4% related to the full capacity respectively, which becomes 32% and 35%.

The modification benefits by saving the overhead cost by Marshalling Component whose complexity is $O(f)$ where $f$ is the number of flows since it converts all the flows in the binary file one by one to the readable format. Hence, Marshalling Component’s performance depends on the distribution of the 5-tuple among the traffic. In this thesis work, the testing traffic generates approximate 3990 flows per time bin (30 seconds).

However, throughputs of TCP and UDP are still unsatisfying. Therefore, there must be certain bottleneck in Recognition Component. As described in section 3.2.3.1, Recognition Component is intended to deal with traffic capture, packet payload recognition and flow aggregation. The capture and aggregation phases are not complex; whereas, the recognition phase dominates and determines the throughput to a great extent due to inspection of the payload of each packet. Moreover, for the sake of accuracy, usually the full payload of each packet is captured and inspected which increases the complexity even more. It is also the reason why the throughput of ICMP is the best and able to reach peak bandwidth since the payload recognition phase is only applied for TCP and UDP. Thus, the bottleneck of this component is located in the payload recognition phase. In other words, improving the performance of the payload recognition phase which impacts the performance of the whole component mostly will enhance the throughput greatly.
Algorithm 3.1, which is described in Section 3.2.4.2, presents the recognition approach of the prototype. This algorithm is an exhaustive search without any heuristics in which TAM systematically enumerates all possible signature candidates and tries to find the first candidate that the payload matches with. In other words, it is an application-wise searching. For example, as a payload is ready to be inspected, TAM checks if it belongs to eDonkey, if it does then assigns “eDonkey” to the application identifier of packet otherwise it checks if it implies BitTorrent if it does then assigns “BitTorrent” otherwise it checks the next candidate. This procedure continues until the payload matches with one candidate or it matches with none. In the best case, the payload matches with the first candidate, whereas the worst case is that it matches with the last candidate or none. Hence, its average complexity is proportional to the number of candidates, i.e. \( O(n) \), where \( n \) is the number of application signature candidates.

Therefore, this approach is suitable when the search space is limited. However, it obviously becomes very inefficient when the number of candidates is prohibitively large. As is shown in Table 3.2, at the moment TAM is able to identify 43 applications \((n = 43)\). The number of applications is not small, thus this approach influences the performance badly. Further, it is supposed to extend the range of identification by adding new application signature strings. Thus, the weakness of this method is too vulnerable to be accepted.

4.3.7 Experiment 7: Recognition Optimization

4.3.7.1 Issue

To optimize the poor performing exhaustive signature searching procedure, a more efficient algorithm is implemented to replace the application-wise algorithm. Algorithm 4.1 illustrates the new bit-wise payload recognition algorithm which is based on the well-known decision tree. The new approach identifies the payload by checking its bytes one by one, i.e. as a payload is ready to be inspected, firstly TAM inspects the first byte then redirect to a probably successful branch where it goes to a subsequent branch according to the second byte for matching with the following bytes. This procedure is repeated on each branch until a leaf node is reached. In the leaf node, the payload is attempted to be matched with one or more signature candidates. If the payload matches with one, then that is the application generating the packet; otherwise, the payload can not be recognized by the bit-wise algorithm and it will be checked by the following methods.
Although most of the recognition procedure has been converted to bit-wise, there are still a few applications’ signature matching methods in the tail of the procedure. The reason is that these signatures are in the form of regular expressions which may appear in arbitrary positions of the payload which means they are compared with every equal-length sub string of the payload. These methods are called when the previous bit-wise approach can not recognize the payload.

Algorithm 4.1: Bit-wise signature-based payload recognition (Decision tree search)

```plaintext
1: Parameters: protocol, payload;
2: Return Value: app;
3: if payload is empty
4:   app = "NONPAYLOAD";
5: end if
6: if protocol == TCP
7:   switch payload.byte[1]
8:     case 0x00:
9:       if payload belongs to BitTorrent
10:         app = "BitTorrent";
11:     end if
12:     case 0x53:
13:       switch payload.byte[2]
14:         case 0x49:
15:           if payload belongs to MP2P
16:             app = "MP2P";
17:         end if
18:         case 0x65:
19:           if payload belongs to Gnuu
20:             app = "Gnuu";
21:         end if
22:     ... // other cases
23:   end switch
24:   ... // other cases
25: end switch
26: if payload belongs to MSN
27:   app = "MSN";
28: end if
29: if payload belongs to HTTP
30:   app = "HTTP";
31: end if
32: ... // signature matching for other applications
33:   app = "TCP"; // none of the applications matches
34: if protocol == UDP
35:   switch payload.byte[1]
36:     case 0x24:
37:       if payload belongs to DirectConnect
38:         app = "DirectConnect";
39:     end if
40:     case 0x16:
41:       switch payload.byte[2]
42:         case 0x02:
43:           if payload belongs to NBDS
44:             app = "NBDS";
45:         end if
46:         case 0x03:
47:           if payload belongs to SSL
48:             app = "SSL";
49:         end if
50:     ... // other cases
51:   end switch
52:   ... // other cases
53: end switch
54: if payload belongs to MSN
55:   app = "MSN";
```

43
Experiment

The network, instrument and kernel configurations are almost the same as Experiment 6 except that the instrument generates frames with various frame sizes separately. The main reason is that in many cases one 64-byte frame is too short to supply sufficient information for recognition, for instance one 64-byte TCP frame only contains 6-byte payload (see Equation(3)) but many application signatures contain characteristic bytes in the payload field beyond 6 bytes. Another factor is that one frame has the minimum size of 64 bytes in Fast Ethernet and commonly it is larger than 64 bytes. On the other hand and most importantly, internal implementation of TAM is modified by optimizing Recognition Component.

Result

<table>
<thead>
<tr>
<th>FS</th>
<th>Latency</th>
<th>Maximum Frame Rate</th>
<th>Frame Transmitting Rate</th>
<th>Frame Loss Rate</th>
<th>Packet Loss Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>Low</td>
<td>148809</td>
<td>148809</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>300</td>
<td>Low</td>
<td>39062</td>
<td>39062</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>512</td>
<td>Low</td>
<td>23496</td>
<td>23496</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>700</td>
<td>Low</td>
<td>17361</td>
<td>17361</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>900</td>
<td>Low</td>
<td>13587</td>
<td>13587</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1500</td>
<td>Low</td>
<td>8224</td>
<td>8224</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.13: Performance-dependent Metrics in Experiment 7.
Table 4.14: Throughput in Experiment 7.

Firstly, for each frame size all the performance-dependent metrics are measured by the method used in Experiment 3. The statistics are listed in Table 4.13 where each row corresponds to each frame size. All frame transmitting rates are full and there is no frame loss or packet loss at all. Hence, the capacity is equal to the bandwidth and the Fast Ethernet switch is not interfering.

Table 4.14 lists throughput measurements where each row corresponds to each frame size. Compared with results from Experiment 6, for 64-byte frames, TAM’s still produces full rate for ICMP frames, while throughputs for TCP traffic surges to 95600 fps which occupies 64% of the capacity and is double as the performance before optimization, and frame rate for UDP traffic also soars which becomes 116500 fps accounting 78% of the capacity and is more than twice efficient than the old algorithm.

One conclusion can be drawn from Table 4.14 is that frame rate declines as frame size grows. This can be explained by Equation (11). For the same recognition procedure, the larger the payload is the larger the number of comparisons is. Another reason which is presented in Equation (6) is that the larger the frame is the lower the maximum frame rate is, or says the slighter the traffic load is.

Another observation is that though frame rate changes corresponding to different frame sizes, both bandwidth and capacity utilization rates almost keep constant which are around 100%, 33% and 72% for ICMP, TCP and UDP respectively. Bandwidth utilization rate is the proportion of maximum frame rate that is occupied by the frame rate, which is stated in Equation (7). Capacity utilization rate is the proportion of TAM’s userspace packet receiving rate that frame rate accounts for, which is stated in Equation (10). As can be concluded from Table 4.13, maximum frame rate is equal to userspace packet receiving rate. Then these two utilization rates fluctuate so slightly seemly because frame rate and maximum frame rate decrease at the same rate when frame size grows. The only exception happens to 64-byte frames. Since 64-byte frames contain very short payload which extremely simplifies the payload recognition procedure, these statistics may be interfered and can be treated as outliers. Thereby, the observations of larger frames are more representative and reliable.
The improvement comes from the algorithm optimization. The merit of the new bit-wise methodology is the reduction in the number of matching attempts. It only needs to search for one or several potentially true candidates. It is accomplished by integration of most of the single applications’ signature matching sub procedures into a whole bit-wise procedure. Meanwhile, practically the number of branching steps is quite small. In most cases, it is sufficient to narrow the number of potential candidates in no more than three branching, i.e. by only checking first few bytes. Therefore, this approach is also flexible for further extension.

Nevertheless, as is mentioned earlier, even with the new byte-wise algorithm, there are still a few applications’ signatures in the form of regular expressions which can occur in arbitrary positions of the payload. Since they are compared with every equal-length sub string of the payload, then the new bit-wise algorithm cannot get any improvement from them. If it is assumed that the number of applications’ signatures in the form of regular expressions is $n$, then the total number of comparisons for them is

$$\text{NumOfComp} = \sum_{i=1}^{n} (\text{PayloadLen} - \text{SigLen}(i) + 1) \ (11)$$

where $\text{SigLen}(i)$ is the length of the $i$th application’s regular expression signature.

Clearly these regular expression signatures impact throughput to some extent if the packets can not be recognized by the bit-wise procedure. And for the same length of payload, the larger $n$ is, the larger the number of comparisons. In other words, the larger number of these signatures causes poorer throughput. Hence, frame rate of UDP traffic improves better than that of TCP traffic as there are more application signatures in the form of regular expressions in TCP recognition procedure than those in the counterpart. This issue may be improved by developing a more efficient string matching algorithm for the regular expressions than the C library functions, `regcomp()` and `regexec()`, which are using.

4.3.8 Experiment 8: Gigabit Ethernet

4.3.8.1 Issue

As is mentioned earlier, TAM is supposed to be applied in a fast speed link, like Gigabit Ethernet network. Thereby, TAM’s performance needs to be measured in the Gigabit Ethernet network.
4.3.8.2 Experiment

Profiting from the compatibility between Fast Ethernet and Gigabit Ethernet, most of the setup need not to be changed except one *D-Link DGS-3224TGR* Gigabit Ethernet switch is applied to replace the Fast Ethernet switch. Figure 4.10 shows the new setup. Two of the instrument’s Gigabit Ethernet ports are connected to the Gigabit Ethernet switch. Each of the ports transmits traffic to the other with half speed (0.5 Gbps). *D-Link DGS-3224TGR* Gigabit Ethernet switch also has traffic mirror feature which enables TAM to monitor the traffic generated by both instrument’s ports. Thus, the real traffic bit rate traversing the network interface of TAM is the bandwidth (1 Gbps). In the configuration of the instrument, all the three protocols and various frame sizes are selected respectively. In addition, since the network setup has been modified, all the performance-dependent metrics have to be measured at first by the method used in Experiment 3.

![Figure 4.10: Setup of Experiment 8. Traffic traversing instrument is mirrored to TAM.](image)

4.3.8.3 Result

<table>
<thead>
<tr>
<th>FS</th>
<th>Latency</th>
<th>Maximum Frame Rate</th>
<th>Frame Transmitting Rate</th>
<th>Frame Loss Rate</th>
<th>Packet Loss Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>Low</td>
<td>234962</td>
<td>234962</td>
<td>0</td>
<td>0.07</td>
</tr>
<tr>
<td>700</td>
<td>Low</td>
<td>173611</td>
<td>173611</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>900</td>
<td>Low</td>
<td>135869</td>
<td>135869</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1500</td>
<td>Low</td>
<td>82237</td>
<td>82237</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.15: Performance-dependent Metrics in Experiment 8.
Table 4.16: Throughput in Experiment 8.

Firstly, 64-byte frame is selected to measure the performance-dependent metrics. IXIA shows latency between two ports is very low. However, comparing the number of transmitted frames shown in IXIA’s user interface and the number of received frames shown in tcpdump output from TAM, it is strange to find frame loss rate is 50%.

It seems the network efficiency is only half. Nevertheless, the real reason is the inherent property of Gigabit Ethernet. It works with different principles physically and logically towards Fast Ethernet; meanwhile operating systems treat these two types of frames in different ways. The key point is that in Fast Ethernet, the maximum cable length is 100 meters with the minimum frame size and slot time left intact. Since Gigabit Ethernet is 10 times faster than Fast Ethernet, for maintaining the same slot size, the maximum cable length would have to be reduced to about 10 meters, which is impractical. To maintain compatibility with Ethernet, the minimum frame size is not increased. Instead, Gigabit Ethernet uses a technique called “Carrier Extension” to create a longer slot-time. With this, the frame size is not changed but the time used on the wire is extended to guarantee at least a 512-byte slot-time. Thus, if the frame is shorter than 512 bytes it is padded with extension symbols which cannot occur in the data stream. Consequently, the transmitting rate on the wire of a 64-byte frame is the same as that of a 512-byte frame. It can be concluded the number of transmitted frames shown in IXIA’s user interface is not real, if re-calculated as 512-byte frames the number of transmitted frames is the same as the number of received frames which proves the observation. As a result, measurements in Gigabit Ethernet should only involve frames equal to or larger than 512 bytes.

All the performance-dependent metrics are listed in Table 4.15 where each row corresponds to each frame size. All frame transmitting rates are full and there is no frame loss. Hence, the Gigabit Ethernet switch is not interfering. Except 512-byte frame, all the others’ packet loss rates are 0.
Table 4.16 lists throughput measurements where each row corresponds to each frame size. Observing across each frame size, capacity utilization rate is still stable and insensitive to frame size while bandwidth utilization rate is influenced by external factors. Therefore, it can be concluded that TAM’s performance is stable and reliable under the same external conditions.

However, compared with the results from Experiment 7, TAM still utilizes the full capacity when processing ICMP frames; however its performance degrades sharply for TCP and UDP frames, which becomes only around 1/10. It indicates that TAM’s real performance is influenced by the external factors seriously, except for ICMP traffic which does not cost recognition overhead. Since neither network setup nor end-system configuration has modified, also frame size has been proved not to impact performance, the main factor should be the bandwidth.

4.3.9 Experiment 9: Bandwidth Tuning

4.3.9.1 Issue

In order to see the relation between TAM’s performance and the bandwidth, the bandwidth is adjusted with all the other factors constant in this experiment.

4.3.9.2 Experiment

The setup is the same as Experiment 8 which is shown in Figure 4.10. The transmitting speed of each port reduces to 300/150 Mbps so that the network interface of TAM receives traffic with the rate of 600/300 Mbps. In addition, since the network bandwidth has been changed, all the performance-dependent metrics have to be measured.

4.3.9.3 Result

<table>
<thead>
<tr>
<th>FS</th>
<th>Latency</th>
<th>Maximum Frame Rate</th>
<th>Frame Transmitting Rate</th>
<th>Frame Loss Rate</th>
<th>Packet Loss Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>Low</td>
<td>140977</td>
<td>140977</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>700</td>
<td>Low</td>
<td>104167</td>
<td>104167</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>900</td>
<td>Low</td>
<td>81521</td>
<td>81521</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1500</td>
<td>Low</td>
<td>49342</td>
<td>49342</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(1) 600 Mbps
<table>
<thead>
<tr>
<th>FS</th>
<th>Latency</th>
<th>Maximum Frame Rate</th>
<th>Frame Transmitting Rate</th>
<th>Frame Loss Rate</th>
<th>Packet Loss Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>Low</td>
<td>70489</td>
<td>70489</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>700</td>
<td>Low</td>
<td>52084</td>
<td>52084</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>900</td>
<td>Low</td>
<td>40761</td>
<td>40761</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1500</td>
<td>Low</td>
<td>24671</td>
<td>24671</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(2) 300 Mbps

Table 4.17: Performance-dependent Metrics in Experiment 9.

<table>
<thead>
<tr>
<th>FS</th>
<th>ICMP FR</th>
<th>BUR</th>
<th>CUR</th>
<th>TCP FR</th>
<th>BUR</th>
<th>CUR</th>
<th>UDP FR</th>
<th>BUR</th>
<th>CUR</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>140977</td>
<td>1</td>
<td>1</td>
<td>8000</td>
<td>0.06</td>
<td>0.06</td>
<td>15400</td>
<td>0.11</td>
<td>0.11</td>
</tr>
<tr>
<td>700</td>
<td>104167</td>
<td>1</td>
<td>1</td>
<td>5700</td>
<td>0.05</td>
<td>0.05</td>
<td>12100</td>
<td>0.12</td>
<td>0.12</td>
</tr>
<tr>
<td>900</td>
<td>81521</td>
<td>1</td>
<td>1</td>
<td>4300</td>
<td>0.05</td>
<td>0.05</td>
<td>9500</td>
<td>0.12</td>
<td>0.12</td>
</tr>
<tr>
<td>1500</td>
<td>49342</td>
<td>1</td>
<td>1</td>
<td>2400</td>
<td>0.05</td>
<td>0.05</td>
<td>5400</td>
<td>0.11</td>
<td>0.11</td>
</tr>
</tbody>
</table>

(1) 600 Mbps

<table>
<thead>
<tr>
<th>FS</th>
<th>ICMP FR</th>
<th>BUR</th>
<th>CUR</th>
<th>TCP FR</th>
<th>BUR</th>
<th>CUR</th>
<th>UDP FR</th>
<th>BUR</th>
<th>CUR</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>70489</td>
<td>1</td>
<td>1</td>
<td>7800</td>
<td>0.11</td>
<td>0.11</td>
<td>15300</td>
<td>0.22</td>
<td>0.22</td>
</tr>
<tr>
<td>700</td>
<td>52084</td>
<td>1</td>
<td>1</td>
<td>5500</td>
<td>0.11</td>
<td>0.11</td>
<td>11000</td>
<td>0.21</td>
<td>0.21</td>
</tr>
<tr>
<td>900</td>
<td>40761</td>
<td>1</td>
<td>1</td>
<td>4400</td>
<td>0.11</td>
<td>0.11</td>
<td>9300</td>
<td>0.23</td>
<td>0.23</td>
</tr>
<tr>
<td>1500</td>
<td>24671</td>
<td>1</td>
<td>1</td>
<td>2500</td>
<td>0.10</td>
<td>0.10</td>
<td>5600</td>
<td>0.23</td>
<td>0.23</td>
</tr>
</tbody>
</table>

(2) 300 Mbps

Table 4.18: Throughput in Experiment 9.
Table 4.17 lists the performance-dependent metrics and Table 4.18 shows throughput measurements where each table corresponds to each bandwidth and each row represents each frame size. As can be seen from Table 4.18, capacity utilization rate is still almost insensitive to frame size which demonstrates the conclusion which is drawn in Experiment 8. On the other hand, for the purpose of this experiment, comparing with the statistics of Experiment 6, 7 and 8, a valuable observation is discovered, i.e. capacity utilization rate which is TAM’s real performance is nearly inverse proportional to the bandwidth, as is shown in Figure 4.4.

4.3.10 Experiment 10: Traffic Snapshot

4.3.10.1 Issue

As is mentioned earlier, TAM facilitates the payload-based recognition technique which captures and inspects the whole payload by default in order to achieve extremely accurate results. On the other hand, the snapped length of payload impacts TAM’s throughput due to the number of computation as is stated in Equation (11) and the amount of data copied from kernel to userspace.

Since TAM is based on libpcap so that it provides a user configurable parameter, i.e. snapshot length, which can be a tradeoff between efficiency and precision. In this experiment, the snapshot length is set to an appropriate value which is expected to be an indicator for finding threshold.

4.3.10.2 Experiment

The setup is the same as Experiment 8 that is shown in Figure 4.10. The transmitting speed of each port is 0.5 Gbps so that the network interface of TAM receives traffic with the rate of 1 Gbps. The snapshot length is set to 200 bytes which means TAM only captures and analyzes the first 200 bytes of each frame.

4.3.10.3 Result

<table>
<thead>
<tr>
<th>FS</th>
<th>Latency</th>
<th>Maximum Frame Rate</th>
<th>Frame Transmitting Rate</th>
<th>Frame Loss Rate</th>
<th>Packet Loss Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>Low</td>
<td>234962</td>
<td>234962</td>
<td>0</td>
<td>0.03</td>
</tr>
<tr>
<td>700</td>
<td>Low</td>
<td>173611</td>
<td>173611</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>900</td>
<td>Low</td>
<td>135869</td>
<td>135869</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1500</td>
<td>Low</td>
<td>82237</td>
<td>82237</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.19: Performance-dependent Metrics in Experiment 10.
Table 4.20: Throughput in Experiment 10.

Table 4.19 lists the performance-dependent metrics and Table 4.20 states throughput measurements where each row represents each frame size.

In Table 4.20, it can be discovered that TAM's performance for TCP and UDP varies as frame size changes and the larger each frame is the more capacity TAM utilizes. Compared with the results from Experiment 8 where the full payload is captured, TAM's performance for TCP and UDP improves significantly, especially for large frames. As is plotted in Figure 4.11 where "(s)" indicates 200-byte snapshot, when analyzing the full payload TAM's performance for TCP and UDP is insensitive to frame size change, however when only inspecting part of the payload the performance goes up gently as frame size increases. The result convinces that the overhead of data transfer and signature comparisons affects the performance seriously, and traffic snapshot is able to enhance the efficiency.

<table>
<thead>
<tr>
<th>FS</th>
<th>ICMP</th>
<th>TCP</th>
<th>UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FR</td>
<td>BUR</td>
<td>CUR</td>
</tr>
<tr>
<td>512</td>
<td>211600</td>
<td>0.90</td>
<td>0.90</td>
</tr>
<tr>
<td>700</td>
<td>165200</td>
<td>0.95</td>
<td>0.95</td>
</tr>
<tr>
<td>900</td>
<td>135869</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1500</td>
<td>82237</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 4.11: Full versus Snapshot Payload Inspection.
5 Conclusion

In this thesis work, the performance of a real time traffic recognition tool, called TAM which is based on passively packet payload inspection, is evaluated and optimized. For the purpose of evaluation, a passive measurement point is set up inside TAM. The evaluation shows that some the external factors, such as hardware equipments and kernel configurations, do impact TAM’s performance by interfering high speed traffic (1 Gbps) input for TAM. Before drawing conclusions about TAM’s performance, all the external factors need to be considered. By constructing a simulated experimental environment, selecting the correct equipments and setting the appropriate kernel configurations, TAM is finally able to receive full speed traffic which helps identify TAM’s inherent performance.

After optimizing TAM, with the bandwidth of 1 Gbps, the number of identified packets is almost constant among various frame sizes for each protocol which is 96%, 3% and %7 for ICMP, TCP and UDP traffic respectively. TAM’s ability for identifying nearly all ICMP packets proves that TAM is capable of capture full speed traffic. TAM performs ICMP traffic much better than TCP and UDP traffic due to the payload recognition procedure is not applied for ICMP packets. The recognition speed is the bottleneck which limits the performance for TCP and UDP traffic since the userspace recognition speed cannot cope with the kernel packet processing speed. Consequently, it results in overflow of the kernel packet queue that causes packet loss. In the mean time, TAM processes UDP traffic more efficiently than TCP traffic. The reason is that the recognition procedure for TCP traffic is more complex than that of UDP traffic, in details, it contains a larger number of signature candidates and they are more complicated.

By tuning bandwidth, an observation is discovered, i.e. TAM’s performance is nearly inverse proportional to the bandwidth which indicates the recognition speed is almost steady for each protocol in spite of different bandwidths. Thus, TAM can be regarded as a reliable tool.

Using snapshot of the traffic is able to enhance TAM’s performance because of less overhead of data transfer from kernel to userspace and less number of signature comparisons, however may lower accuracy on the other side. A separate work is deserved to find a balancing threshold between efficiency and accuracy.
6 Discussion

As is mentioned earlier, TAM is part of a prototype and TAM contains a key position in determining the performance of the whole architecture. Thus, the evaluation of TAM’s performance is a representative result of the whole tool’s performance.

6.1 Limitations

Recognizing traffic by inspecting the packet payload has several limitations. Some of them are specific to the approach of this thesis; however, the others are inherent of the problem.

- **Privacy issues**: As the increasing concern over privacy among users and ISPs, payload inspection is not allowed in some cases. Thus, this methodology will not be applicable.

- **Encryption**: A few protocols rely on encryption and SSL to transmit packets. However, payload string matching depends on the content of packet payload. Consequently, this technique cannot work on encrypted packets. However, this is probably true for most recognition techniques.

- **Unknown P2P protocols**: As P2P applications are used worldwide and more and more new protocols emerge, a broad variety of P2P protocols came into existence. Though some of them are well-known, the others are unknown or proprietary. However, application identification based on payload requires knowing the protocol patterns in advance. Therefore, this thesis work cannot guarantee identification of all P2P applications. However, the prototype has been evaluated that the P2P protocols which it can identify represent the majority of the current P2P traffic.

- **High storage and computational cost**: The methodology captures the specified part of payload of each packet that traverses the link and search for specific signatures. As a tradeoff to the extremely high precision, it consumes a lot of resource.

- **Only IP traffic**: Due to this thesis work is part of a solution to build robust IP networks, it only concerns IP traffic, in particularly TCP, UDP and ICMP protocol associated with the transport layer.

6.2 Future Work

With the work has been done so far, TAM’s efficiency in the Gigabit Ethernet network has been evaluated and optimized. The result shows there is room for further improvement and becomes an indicator for future work.
- **Platform:** TAM is based on *libpcap* library and is evaluated on a Linux platform. Despite *libpcap* offers the very same programming interface across different operation systems, the library performance are very different depending on the platform being used. TAM could be transplanted to other platforms in order to evaluate the overall performance gain on a non-Linux platform.

- **Kernel packet classification:** The context switches overhead between kernel and userspace impacts TAM’s performance. As the overhead of transferring packets from kernel to userspace via system calls is costly, severe packet loss happens if userspace applications cannot cope with kernel speed. Basically, it can be improved by reducing the number of data exchange. One approach to achieve it is to handle packets only inside the kernel, i.e. the packets are not passed to userspace applications. Instead only the flows are passed (the number of flows is much fewer than that of packets) to userspace applications [17]. Thus, at least the capture and recognition part of TAM need to be implemented as a kernel module. Figure 6.1 plots the architecture of kernel packet classification.

![Figure 6.1: Architecture of Kernel Packet Classification.](image-url)
- **Recognition approach:** Though the proposed byte-wise (decision-tree-like) algorithm improves TAM’s performance a lot, there is still some bottleneck inside it. Since there are a few applications’ signatures in the form of regular expressions which can occur in arbitrary positions of the payload and cost one-to-one search. Consequently, they cannot be integrated into the bitwise algorithm and affect the performance to some extent. One way to improve this is to assign different threads to perform bitwise and application-wise search separately. Before leading to the new multithreading software architecture, some effort deserves for finding out how much of regular traffic could be identified by the bit-wise approach.

- **Capability extension:** In order to be a more powerful recognition tool which can handle as many as possible type of current traffic, TAM’s capability should be extended to be able to recognize more applications. Benefited from the bit-wise recognition approach, TAM is easily extensible to involve recognition methods for new applications. Another extension could be considering flows both uni-directional and bi-directional. This can be achieved by modifying the flow aggregation procedure.

- **Accuracy evaluation:** Prior to this thesis work, original TAM’s accuracy has been measured which is for sure at the same level of some commercial products. Since TAM’s signature-based recognition process has been modified and improved, though some simple tests have been taken after modification, it is recommended to evaluate its precision by real traffic or trace files.

- **Tradeoff between efficiency and accuracy:** The snapshot length of payload to be captured and inspected impacts both TAM’s efficiency and accuracy. The larger snapshot length leads to lower performance but higher precision, vice versa. Finding the threshold for balancing these two factors deserves a separate study itself. For example, considering all applications recognized by TAM, what should be the snapshot length? What is the “performance” curve that correlates such length and the proportion of recognized applications?

- **10 Gigabit Ethernet:** For the higher demand of a traffic recognition tool, TAM’s performance needs be measured in the 10 Gigabit Ethernet.
- **Recognition collision:** In the signature matching procedure, matching collisions may happen. In other words, there are flows that could be labeled in more than one class. It is related to priorities and exclusive identity assignment for flows. Thus, a good classification algorithm should cope with this, if the analyst wishes to have this feature. For example, when video is streaming over HTTP, it could be regarded as “HTTP”, “HTTP Video” or both. It should be user configurable and TAM needs to be ready to take this into account.

- **Data structure:** TAM uses a hash table for storing the flow table which is efficient for searching, inserting and deleting. However, a more appropriate hash function for generating hash key and a more proper size of hash table could be selected with care in order to decrease the possibility of hash collision which may impact the time complexity of searching, inserting and deleting.
References


## Appendices

### A. Definitions

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>The amount of data per time unit that is processed by TAM.</td>
</tr>
<tr>
<td>Frame Rate</td>
<td>The amount of frame per second that is processed by TAM.</td>
</tr>
<tr>
<td>Maximum Frame Rate</td>
<td>The ideal peak frame rate with transmitting at full bandwidth and receiving without any interference.</td>
</tr>
<tr>
<td>Bandwidth Utilization Rate</td>
<td>The proportion of the bandwidth that is utilized by TAM.</td>
</tr>
<tr>
<td>Frame Transmitting Rate</td>
<td>The frame rate that is transmitted by the traffic generator.</td>
</tr>
<tr>
<td>Latency</td>
<td>The time delay between a packet is sent and received.</td>
</tr>
<tr>
<td>Frame Loss Rate</td>
<td>The proportion that the number of lost frames during the transmission before arriving the receiver’s kernel accounts for the number of frames transmitted by the traffic generator.</td>
</tr>
<tr>
<td>Packet Loss Rate</td>
<td>The proportion of packets dropped by the kernel accounting for packets received by the kernel.</td>
</tr>
<tr>
<td>Capacity Utilization Rate</td>
<td>The proportion of the end-system’s userspace capacity that is utilized by TAM.</td>
</tr>
</tbody>
</table>
### B. Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAM</td>
<td>Traffic Analyzer Module</td>
</tr>
<tr>
<td>SrcIP</td>
<td>Source IP</td>
</tr>
<tr>
<td>DstIP</td>
<td>Destination IP</td>
</tr>
<tr>
<td>SrcPrt</td>
<td>Source Port</td>
</tr>
<tr>
<td>DstPrt</td>
<td>Destination Port</td>
</tr>
<tr>
<td>BPF</td>
<td>Berkley Packet Filter</td>
</tr>
<tr>
<td>Gbps</td>
<td>Gigabit per second</td>
</tr>
<tr>
<td>Mbps</td>
<td>Megabit per second</td>
</tr>
<tr>
<td>FS</td>
<td>Frame Size</td>
</tr>
<tr>
<td>FR</td>
<td>Frame Rate</td>
</tr>
<tr>
<td>BUR</td>
<td>Bandwidth Utilization Rate</td>
</tr>
<tr>
<td>CUR</td>
<td>Capacity Utilization Rate</td>
</tr>
</tbody>
</table>