Traffic Recognition in Cellular Networks

Alexandros Tsourtis
Abstract

Traffic Recognition in Cellular Networks

Alexandros Tsourtis

Traffic recognition is a powerful tool that could provide valuable information about
the network to the network operator. The association of additional information
 carried by control packets in the core cellular network would help identify the traffic
 that stem from each user and acquire statistics about the usage of the network
 resources and aid detecting problems that only one or a small group of users
 experience. The program used is called TAM and it operates only on Internet traffic.
 The enhancements of the program included the support for the Gn and Gi interfaces
 of the cellular network where the control traffic is transferred via the GTP and
 RADIUS protocols respectively. Furthermore, the program output is verified using
 two other tools that operate on the field with satisfactory results and weaknesses
 were detected on all tools studied. Finally, the results of TAM were demonstrated
 with conclusions being drawn about the statistics of the network. The thesis
 concludes with suggestions for improving the program in the future.
Acknowledgements

My thanks goes first to my supervisor in Ericsson, Tord Westholm for suggesting valuable comments, helping to structure the work and the presentation and allowing me to take decisions concerning the project direction. Mr. Jiasu Cao, also from Ericsson, helped me a lot during the project answering many questions that raised from the analysis of the trace files, suggested the packet payload calculation and the duplicate packet detection enhancements and has my sincere gratitudes.

I would like also to thank my reviewer, Ivan Christoff, for the support during the project.

A colleague working in the same program as I was working, Zhenfang Wei, contributed some of his work to my thesis, particularly the DHT application detection ability, and I would like to thank him.

Finally, I would like to thank everyone else that helped me on the thesis, particularly my brother that provided comments to an early version of the thesis report, and my family and friends that supported me during the project.

Remarks

The thesis work was performed in Ericsson, Kista. However, the views expressed below are the opinions of the writer and should not be considered as the company's perspective. The author is the one to be accounted for any mistakes in the study below.

The implementation of some parts of the GTP and the Radius protocol strictly targets only observation of traffic and shall not be seen as a way to implement the protocols to handle traffic in network elements.

The traces used in the thesis stem from real networks. However, the locations and operators of the networks shall not be disclosed and action is taken to remove operator related information from the results presented here.

To Sofia
# Table of Contents

Chapter 1 Introduction ................................................................. 9
  1.1 Background Information .................................................. 10
  1.2 Scope ........................................................................... 11
  1.3 Objectives .................................................................... 12
  1.4 Insights on TAM ............................................................. 12
  1.5 Related work ................................................................. 13
  1.6 Overview .................................................................... 15

Chapter 2 The cellular network ...................................................... 17
  2.1 The GPRS core network .................................................. 19
    2.1.1 Accessing the network – Gn interface ................................ 20
    2.1.2 Gi interface ................................................................ 21
    2.1.3 Network Elements ..................................................... 21
  2.2 GPRS tunneling protocol – GTP .......................................... 22
    2.2.1 GTP operation .......................................................... 24
  2.3 RADIUS Accounting protocol ........................................... 26

Chapter 3 Traffic Analysis Module ................................................. 29
  3.1 Enhancing TAM ............................................................... 30
    3.1.1 Common enhancements ............................................. 31
  3.2 Enhancements for the Gn interface ...................................... 38
  3.3 Enhancements for the Gi interface ...................................... 47
  3.4 A user filter ..................................................................... 51

Chapter 4 A real world trace study ............................................... 53
  4.1 Problems Encountered ...................................................... 55
    4.1.1 Trace file problems ................................................... 55
    4.1.2 IP fragmentation ......................................................... 56
    4.1.3 Duplicate linked list entries ....................................... 57
    4.1.4 C language ............................................................... 57
    4.1.5 TAM bugs ............................................................... 58
  4.2 Studying the Gn interface .................................................. 60
  4.3 Studying the Gi interface ................................................... 70
  4.4 Program output ............................................................... 73
  4.5 Verification – Comparison of programs ................................ 75
    4.5.1 CFlow ................................................................. 76
    4.5.2 MFlow ................................................................. 78
    4.5.3 CFlow vs MFlow ..................................................... 86
Figure 4: The PDP context creation.................................................................25
Figure 5: RADIUS Accounting procedure....................................................27
Figure 6: Sequence of messages in the Gn and Gi interfaces. Note that no RADIUS authentication messages are used in the termination of the user session..................................................................................50
Figure 7: Downlink and uplink traffic per APN..............................................92
Figure 8: Number of packets sent as downlink and uplink per APN............93
Figure 9: Amount of data transferred per transport layer protocol on identified flows..............................................................................................................94
Figure 10: Amount of data transferred per application.................................94
Figure 11: Data transferred per application on unidentified flows.................95
Figure 12: Data transferred per application in Apn1.....................................96
Figure 13: Data transferred per application in Apn2.....................................97
Figure 14: The number of users per APN.....................................................98
Figure 15: Duration of PDP contexts per APN. The mean value and standard deviation per APN are displayed here.................................................................99
Figure 16: Cumulative distribution function of the percentage of users per PDP context duration..................................................................................................................100
Figure 17: Cumulative distribution function of the percentage of users per data traffic volume..................................................................................................................101
Figure 18: Cumulative distribution function of the percentage of users per number of PDP contexts.............................................................................................................102
Figure 19: Downlink and uplink traffic per APN..............................................103
Figure 20: Number of packets sent as downlink and uplink per APN........104
Figure 21: Amount of data transferred per transport layer protocol..............104
Figure 22: Amount of data transferred per application.................................105
Figure 23: The number of users per APN.....................................................106
Figure 24: Duration of RADIUS sessions per APN. The mean value and standard deviation per APN are shown.................................................................107
Figure 25: Cumulative distribution function of the percentage of users per RADIUS session duration..................................................................................108
Figure 26: Cumulative distribution function of the percentage of users per data traffic volume.............................................................................................................109
Figure 27: Cumulative distribution function of the percentage of users per number of RADIUS sessions..................................................................................110
Figure 28: The GTP header.........................................................................125
Figure 29: Format of the GTP extension header..........................................126
Figure 30: The format of the GTP TypeValue IE.........................................128
Figure 31: The format of the GTP TypeLengthValue IE..............................129
Figure 32: Format of the RADIUS header..................................................133
Figure 33: Format of the RADIUS attributes..............................................134
Figure 34: The GRE protocol header...........................................................136
Figure 35: The L2TP protocol header...........................................................137
Figure 36: PPP header..................................................................................137
Chapter 1 Introduction

The amount of data being transferred in computer networks has increased significantly in the last decade[1**] and continues to increase due to new users, applications and needs. In order for Internet Service Providers (ISP) to be able to predict their future needs, they use network traffic analysis and recognition tools to monitor the utilization of the network. The statistics acquired help network operators to decide when, how and where they should upgrade their networks. Moreover, traffic analysis tools can be used in order to classify the traffic and prioritize certain types, like Voice over IP (VoIP), or drop others enforcing traffic shaping. Other uses include the detection of malfunctioning devices and protocols. Traffic analysis can assist security and traffic patterns which are used in Intrusion Detection Systems (IDS)[2**].

In the era of 3rd generation cellular systems users experience high speed data transfer on their mobile phones[3**]. Thus the core networks supporting the users are required to handle large amounts of traffic. It is interesting to study the applications that users use on the aspect of detecting what network resources are consumed by a user and for what purpose. That would aid charging policies and clever Quality of Service (QoS) mechanisms that would be able to adjust to users' needs so everybody would experience the best from the network.

Traffic measurement in cellular core networks is more challenging than in the Internet since the network protocols used are different. They are able to give more information about the users and sometimes not known as in some interfaces of the core network, where cellular operators have the flexibility to choose.

The study bellow tries to bind together the Internet traffic analysis with traffic analysis on the Third Generation Partnership Project (3GPP) core cellular network, utilizing the additional information provided by the core network protocols to get a better view of the traffic flows passing the network. Interesting questions to be answered are, for example, what is the popularity of an Access Point Name (APN), how many users use peer-to-peer P2P applications, how often particular users utilize the network and for how long and what volume of traffic is transferred per user (or per session or APN).

That is accomplished by studying two General Packet Radio Service (GPRS) core network interfaces (Gn and Gi) which aggregate traffic from many users.
1.1 Background Information

Network traffic analysis is the process of analyzing network traffic by studying its various properties like bit rate, packet rate or network and transport layer protocols used. A part of the analysis is traffic recognition and it is the process of recognizing the protocols being used in the application layer, the payload, of each packet.

Their goal is to acquire statistical knowledge on how the network resources are used and to provide insights on when, where and why the network infrastructure should be upgraded in order to increase the network capacity, performance and security. It is also a way to detect malfunctions on the equipment by evaluating the validity and correctness of the headers on the protocols used in the link of study. Moreover, traffic recognition can be used in security applications. The IDS, for example, enforce pattern recognition on the network traffic to detect anomalies.

The majority of networks for data transfer are packet switched, that is, using packets to carry data. Each packet carries only a small portion of the data being transferred and the source and destination addresses. Devices in the core network called routers study the addresses of each packet and forward it in the appropriate direction until the packet reaches the destination. Traffic recognition on packet switched networks is performed by filtering and capturing the packets that pass via a link in the network. Filtering is the process that selects for capture only the packets that satisfy the criteria given, such as “only ip-tcp traffic”, and capturing is the process of saving a copy of each packet for later processing by the traffic recognition software.

The recognition process analyzes the contents of each packet, being the protocols and the data contained in the packet. This information helps to build flows of traffic that are communication channels between a source and a destination and detect the application layer protocol used in each flow. Common application layer protocols are HTTP, DNS, FTP, POP3, IMAP, SMTP and P2P variances. Traffic recognition is used in the Internet to assist Internet providers and in IDS for pattern recognition[2**].

The data packets are captured by a network analysis tool and saved to files called network packet traces, or simply traces.

The cellular networks are wireless networks used in the past primarily for voice communication, like Global System for Mobile communication (GSM). Extensions enabled cellular networks to transfer packet switched data (GPRS). In the 3rd generation (3G) the maximum connection speed for data transfer has been increased and that enables the mobile phones¹ being used more and more to access data services. In order to assist operators on

¹ Actually, mobile phones are only a part of the devices used for data transfer. Other devices may be PDA, laptops etc.
network management decisions, traffic recognition may be applied in the cellular network core.

The cellular core is, however, different from the Internet in the aspect of protocols being used and the requirements for traffic recognition. For example, while in Internet the user identification is done via the unique IP address of the user, in cellular networks the primary user identification parameter is the subscriber identity, International Mobile Subscriber Identity (IMSI)[4**].

The cellular core contains many different interfaces that connect the elements of the network. Some interfaces are used for control information while others carry control and user information (data). The interfaces that carry the packet switched data inside the cellular core are the Gn and Gi. The Gn interface is used to connect the network elements of the cellular core while the Gi is a gateway interface to external networks.

On the Gn interface the GPRS Tunneling Protocol (GTP) handles the encapsulation of user data in order to keep the core independent of the protocols being used in the endpoints. The data are transferred via tunnels set up by the control stack of GTP.

The data are transferred to an external network along the Gi interface. Here encapsulation using Generic Routing Encapsulation (GRE) or Layer 2 Tunneling Protocol (L2TP) / Point-to-Point Protocol (PPP) is possible or the data may be transferred as plain IP packets. It is operator dependent whether authentication and/or accounting should be applied in the form of Remote Authentication Dial In User Service (RADIUS) or DIAMETER Authentication, Authorization and Accounting (AAA) servers.

1.2 Scope

The thesis is limited to study the Gn and Gi interfaces in the GRPS network. Control information on these interfaces should be extracted and bound to used data. This information consists of the IMSI the APN and the Mobile Subscriber ISDN Number (MSISDN) based on their appearance in control messages.

On the Gn interface the GTP version 1 protocol and on Gi the RADIUS accounting protocol should be analyzed.

A comparison of the tool developed to other two tools should be included for verification purposes.

Various statistics from the results of the traffic analysis are to be presented and explained.

Performance is not the primary objective of the thesis. However, since performance is of high importance in traffic recognition applications, it
would remain as a concern and drive decisions being made in the implementa-
tion phase.

Some parts of the GTP protocol are not implemented since they have not
been encountered in the trace files and testing was not possible, e.g. Multi-
media Broadcast Multicast Service (MBMS).

1.3 Objectives

The goal of the thesis is to enhance TAM, a traffic recognition software,
by developing a prototype tool to apply traffic recognition in cellular net-
works. The extra information carried in the core network protocols, like the
subscriber identity IMSI and the access point to an external network APN,
should be included in the construction of the traffic flows.

In order for the tool to be useful, its correctness should be considered.
That is done by analyzing data and arguing the validity of the results stem-
ing from the tool. Comparison of the TAM results to results from other
tools operating in the area is also performed.

Furthermore, the thesis should include and analyze statistics from the
study of a real network, acquired by the enhanced TAM version.

Last but not least, the performance of the tool should remain a concern in
the process.

1.4 Insights on TAM

The development of a traffic recognition tool that would support cellular
networks shares common parts with the traffic recognition for the Internet.
The way flows are constructed and the application layer protocol of a flow is
detected are similar in both networks. The difference is the requirement of
handling control traffic and extracting useful control information to accom-
pany the results. Thus, a traffic recognition tool for the Internet is to be en-
hanced to support the core cellular network.

The tool used is called Traffic Analysis Module (TAM) and supports the
aggregation of packets into flows of traffic by matching the IP addresses and
port numbers of the end points, source and destination. It also detects the ap-
plication layer protocol via regular expressions (similar regular expressions
can be found on the L7-Filter project in SourceForge [5**]) being executed
on the payload of each packet. This process is considered relatively slow,
since there exist many different application layer protocols and the software
tries to match each one to the payload of the packet being studied. The process resembles finding a password via a brute-force method.

The TAM output is the flows detected and statistics as the time a flow was active, the amount of bytes and packets transferred in a flow and the application used in the flow. The output is presented in a user friendly form to facilitate further analysis and acquisition of statistics.

1.5 Related work

Many traffic analysis and recognition tools exist. The most common is probably Wireshark[6**]. It is an open source program that is based on the libpcap[7**] library and offers an easy to use user interface to examine the contents of captured packets. It supports the analysis of many protocols, including the GTP and RADIUS protocols being studied here. However, Wireshark is not sufficient for the project since it has some fundamental drawbacks. First, it analyses the whole packet, a process that is slow and sometimes unnecessary as there could be protocols that contain no relevant information, like the Ethernet layer². Furthermore, Wireshark supports the recognition of application layer protocols via port matching. That is a simple way to detect the applications the user uses but it can be misleading since many applications do not follow this scheme[8**][9**]. It is common for many applications like p2p to use HTTP ports in order to avoid detection and bypass firewalls. For example, Skype makes use of port 80³. Finally, Wireshark is able to detect control information but it cannot assign that information to the relevant user packets. Thus, in order to detect the packets belonging to one user it is required to perform two queries (analyze the data twice), to locate the control information and then the user packets.

Other open source traffic analysis tools are e.g. CoralReef from CAIDA[10**] that supports the aggregation of data packets into flows of traffic. Argus[11**] is another tool for auditing network traffic and acquiring interesting statistics. However, none of these tools supports the GPRS protocols and also they do not offer user application recognition based on packet payload inspection.

Payload inspection in order to characterize traffic has been considered for example in [8**] and [12**]. The former paper focused on recognizing traffic for an online computer game while the latter focused on recognizing general Internet traffic. It also raised the issue of fragmented IP packets that may cause inaccurate results for some flows.

---

² In the network core, link layer protocols contain no user related information.
³ Port 80 is assigned by IANA to the HTTP application.
The types of payload inspection are mentioned in [13**] and briefly include:

**Packet Based No State (PBNS)** – classification via the TCP/UDP port numbers

**Packet Based per Flow State (PBFS)** – per packet application recognition, that is also the type used in TAM

**Message Based per Flow State (MBFS)** – processing of the network traffic as messages and not packets, reassembling IP fragments and TCP segments in order to perform recognition on the entire message transferred by the application.

**Message Based per Protocol State (MBPS)** – study of how the application layer protocols operate in order to decide the message application.

The paper also mentions that the accuracy of the payload-based application classification highly depends on the quality of the signatures (regular expressions) used.

Except for traffic recognition, payload inspection may also be used to detect and filter out malicious packets. However, a live filter would need to perform very fast to avoid being the bottleneck of the network. One solution as shown in [14**] is to use specialized hardware Field-Programmable Gate Array (FPGA) to scan the header and payload of each packet. Their device was able to process packets at a speed of 2.88 Gbit/sec. That approach could also be used to improve the performance of software tools like TAM, by implementing them in a dedicated hardware device. The procedure has drawbacks though, with the most important being the loss of flexibility (necessity to reprogram the device upon code change). However, it could produce an important performance gain.

To detect user applications, payload inspection is not the only solution. [15**] presents an alternative method to detect the user application by studying the transport layer semantics only, without the evaluation of the payload of the packet. However, their solution is limited to the detection of P2P traffic. The most important advantage of the technique is that it is capable of detecting P2P traffic of unknown protocols assuming the characteristics of the traffic are shared between the different protocols. As the paper suggests the P2P traffic is increasing and it should be expected that it would reach the mobile networks too. As the detectability of this traffic depends on the analysis of the P2P protocols, the work of [15**] may be used to signal the presence of new protocols in the area.

Actual traffic recognition in GPRS networks has been performed in the METAWIN project [16**][17**][18**][19**]. The project is a study of an operational GPRS network in Austria on its GPRS interfaces, including Gn and Gi. It covers the association of control information to user traffic, by examining the signaling information carried on the interfaces, like the GTP control messages. It continues by detecting congestion and bottlenecks on the network, undesired traffic on the data like worm infections and the aspect
of anomalous control traffic mostly caused by buggy terminals that it is easier to detect in interfaces closer to the terminal, like Gb.

The term border effect is used in [19**] in order to define the situations where the PDP context or Radius session started before the beginning of the packet capture. It is the most important reason of low success rate on associating control information to data traffic in TAM.

The authors of [19**] collected information for each PDP context and connection (similar to flow in TAM). They used a PDP context identifier in connections in order to associate them to PDP contexts. That inspired us of using a PDP ID in TAM as a flow aggregation parameter, in order to limit each flow to only one PDP context.

The association of control information to data traffic is also performed by other, mostly commercial, tools. For example RACOM[20**], provides many tools for network protocol analysis and monitoring that include the Gn and Gi interfaces. An example is the Cellular Expert. Another company specialized on the field is Tektronix[21**]. The concept is also applied to communication interception for law enforcement, as with STAR-GATE a product from Verint[22**]. All commercial solutions share a common line of not disclosing detailed information and although all support the association of data traffic to a particular user using control information, none mentions if e.g. application recognition is performed.

1.6 Overview

The rest of the thesis is organized as follows. Chapter 2 serves as an introduction to the core cellular network by analyzing relevant parts of the Gn and Gi interfaces that are required in order to comprehend the chapters that follow. However, some information about the protocols used in the study, like the format of the headers, is located in the Appendices. The reason is that the study is not a detailed analysis of the protocols, it is the analysis of data based on information extracted from the protocols.

The enhancements introduced on TAM are explained in Chapter 3. Chapter 4 contains the analysis and results of a traffic recognition study on real world data. The chapter supports Chapter 3 and explains what situations led to the implementation decisions being taken. Comparison of TAM to other tools for verification purposes is also present in Chapter 4.

In Chapter 5 the performance of the tool is examined and Chapter 6 concludes the master thesis with a summary, discussion, evaluation, suggestions and future work.
Chapter 2 The cellular network

The cellular system is standardized by 3GPP[23**], a collaboration of organizational partners such as the European Telecommunications Standards Institute (ETSI)[24**]. The 3GPP is divided into technical groups, each one with a specific purpose. The core network, is specified in the 3GPP Technical Specification Group - Core Network & Terminals.

The core cellular network has many interfaces that connect the different parts of the network. Some interfaces are used for control traffic while others transfer control traffic and user data. A study that includes all interfaces is out of the scope of the thesis. Instead, the aim is to study the interfaces that carry packet switched user data and perform traffic recognition. Thus, the focus is on the GPRS core network. For more information please refer to GPRS standardization or books related to the subject, e.g. [25**][26**]. Figure 1 shows the GPRS architecture where the interfaces of study, Gn and Gi, have been marked in blue(or light grey) lines.
These interfaces carry control information called signaling and data traffic. Control information carries interesting parameters to be associated with user data by TAM, including:

(note that some parameters are optional)

- **IMSI**
  IMSI is the International Mobile Subscriber Identity and it is used to uniquely identify a user in the cellular network.

- **APN – Access Point Name**
  The access point that the uses wishes to connect to. It identifies the Packet Data Network (PDN) that the user wishes to connect and its purpose is to allow many different networks like operator internal network, corporate Intranet, or Internet. It enables the use of different ISPs in the mobile network.

- **MSISDN – Mobile Subscriber ISDN Number**
  The phone number of the user.

- **EUA – End User Address**
  The IP address assigned to the user.

- **Charging related information**

- **QoS parameters**

- **The Radio Access Network (RAN)**
  The radio technology type used by the user, e.g. UTRAN or GERAN.

- **Location information**
  The routeing area of the user.

- **Other control information**
2.1 The GPRS core network

The GPRS core network connects the user equipment via the RAN to the gateway to a PDN. It consists of two primary nodes, the Serving GPRS Support Node (SGSN) and the Gateway GPRS Support Node (GGSN), and the user data are primarily transferred via two interfaces, namely the Gn and Gi. Figure 2 focuses into the GPRS architecture.

[Diagram]

Figure 2: The GPRS architecture, a closer look.

The Gn interface is located between the SGSN and GGSN network nodes. The Gi interface connects the GGSN node to an external PDN. As shown in Figure 2, there may exist many GPRS Support Nodes (GSN), as both the SGSN and GGSN are called, in a operator's network.

The process of sending packet switched data is in short as follows:

The Mobile Station (MS) that is GPRS capable should notify the network via a GPRS-attach procedure as is defined in [27]. When data are to be transmitted, a Packet Data Protocol (PDP) context[28] is required. The PDP contains the parameters of the connection like the user IMSI and the APN that the user would connect to among others. The MS initiates the PDP context via the PDP context activation procedure to an SGSN. The SGSN that handles the procedure sends a Create PDP Context Request to the GGSN that replies with a Create PDP Context Response. The GGSN may be required to authenticate the user and/or perform accounting to a RADIUS[29][30] server before replying to the SGSN. Assuming that
the process was successful, data may be now transmitted from the MS to the PDN and vice versa.

The interesting parts of the process are the ones performed in the Gn and Gi interfaces, namely the PDP context creation and the authentication/accounting actions. These parts are presented below in a greater detail.

2.1.1 Accessing the network – Gn interface

On the Gn interface the PDP context is used to transfer user data. It may be seen as two tunnels, one for control and one for data, between the SGSN and the GGSN carrying encapsulated user traffic via the GTP. The context is set up by the Create PDP Context Request message sent by a SGSN to the GGSN handling the APN that the user wishes to connect. The GGSN should then assign tunnel identifiers for the context, perform authentication, accounting and optionally allocate a network address to the user. It then replies to the SGSN with a Create PDP Context Response message containing i) information if the request was accepted or not and ii) the tunnel identifiers being set up in the GGSN. If the latter accepted the PDP context then it is now possible for the user to communicate with the APN she selected.

The activation of a PDP context may be optionally done by the network in cases there exist packet data to be delivered to the user. However, the procedure is still the same but initiated by a GGSN sending a PDU Notification Request message to the SGSN serving the user.

A PDP context contains QoS parameters and it is possible for a user to change these parameters via an Update PDP Context Request message or initiate another PDP context with different QoS (secondary PDP context) parameters. A user may also activate another PDP context with different APN (primary PDP context).

The termination of the PDP context is performed via the Delete PDP Context procedure. It is typically terminated when the user performs a GPRS-detach or stops communicating for an operator defined timeout time. However, it is possible for a PDP context to remain active for a long time to enable IP telephony or other network initiated services, typically via keep alive traffic.

When a PDP context is activated, the data transfer may start. The network address(es) assigned to the user may be permanently or temporarily associated with the user's IMSI or the user may have only a simple point to point connection on which a network address is not necessary. Typically though the IP protocol is used, and due to the shortage of IP version 4 addresses Network Address Translation (NAT) is used by operators in order to assign IP addresses from the private IP address space to their subscribers[6**].
2.1.2 Gi interface

The Gi interface is located between the GPRS core network GGSN node and an external packet network such as the Internet, an internal corporate network etc. Traffic recognition is more obscure here since it is operator dependent how the user data would access the PDN. The specification[31**] defines that operators may use normal IP or GRE[32**][33**] and L2TP[34**] tunnels to transport user data.

Two modes of network access are defined in the Gi specification document. One is named Non-Transparent Mode and involves the authentication of the user, typically via the RADIUS access messages, to the network. The second option, Transparent Mode, involves no authentication of the user to the network and thus no RADIUS authentication is performed. However, in both modes it is possible to use RADIUS for accounting purposes.

As a result, it may be required for authentication, authorization and accounting to be performed on the interface and that is typically being done via a RADIUS or DIAMETER\(^4\) server with the GGSN acting as the client Network Access Server (NAS).

The protocols used to communicate to the servers, Radius and Diameter respectively, are the ones that carry the control information on the Gi interface. If we ignore these protocols then the Gi interface resembles an Internet backbone connection link with the addition of tunneling.

However, it is common for Radius Accounting to be applied on Gi for charging, statistical or network monitoring purposes. As a result the protocol is analyzed here in order to extract the necessary control information being the MSISDN and APN.

Radius authentication may also be seen as a way to extract user information. However, the Radius Access messages do not carry the user IP address that is the parameter needed in order to associate information from control traffic to data traffic and that is the reason that they are not being used in order to identify data packets.

2.1.3 Network Elements

The two most important network elements of the GPRS network are presented below. Note though that it is possible to exist multiple times in the network and may be interconnected with routers. It is also possible the SGSN and GGSN be integrated into one network node.

\(^4\) The specification of the Gi interface defines also the use of the Diameter protocol, that is the new version of the Radius protocol. However, based on the scope of the thesis the Diameter protocol is not considered.
2.1.3.1 The Serving GPRS Support Node (SGSN)

The SGSN performs access control and security functions and keeps track of the location of the UE so to be able to route data traffic to it. It also controls the establishment of tunnels to the GGSN to transfer the user data. The SGSN is connected to the RAN via the Iu or Gb interface. There may exist more than one SGSN in a PLMN.

The SGSN is the node that sends the Create PDP Context Request message that establishes a PDP context between the SGSN and the GGSN, used to transfer user data. For efficiency, the SGSN may establish a direct link between the Radio Network Controller (RNC), the node that sends the user data to the SGSN, and the GGSN. The direct link is used only for user data and is called a direct tunnel in 3GPP terminology.

2.1.3.2 The Gateway GPRS Support Node (GGSN)

The GGSN connects, via the Gi interface, the Public Land Mobile Network (PLMN) to packet data networks such as corporate Intranet or the Internet. It is used to tunnel packets from a packet network to the location that the MS is attached (the SGSN) and from the MS (via the SGSN) to the packet network. The destination of the packets that stem from the MS is defined in the PDP address (typically an IP address). It may exist more than one GGSN in a PLMN.

The GGSN is the node that responds to the Create PDP Context Request message sent by an SGSN in the Gn interface and serve as the client to an AAA server if AAA is used in the Gi interface. On the latter interface it is connected to different networks, each classified via an APN.

2.2 GPRS tunneling protocol – GTP

On the Gn interface the control information mentioned in the beginning of the chapter is carried by the GTP[28]** protocol. Its main purpose is to tunnel data on the top of an IP network. That is in order to keep the core network independent from the protocols used between the MS and a packet switched network. For example on top of GTP may pass IP version 4, IP version 6, IPX, PPP or other network protocols without any change in the GPRS core network. However, the most common protocol used is IP. The tunneling
functionality is expected to increase the transfer time of a packet as it requires another level of encapsulation.

In order to extract the necessary information and perform traffic recognition, it is required to analyze how GTP works. However, its specification is about 150 pages, and to avoid replication, here only a small part is mentioned together with the header formats and some additional information in Appendix C.

The GTP protocol has three versions. GTP version 0 that has been defined before 1999, GTP version 1 defined by 3GPP in 1999 targeting the 3G networks and GTP version 2 has been specified in 2008 for the LTE/SAE network core. GTP version 0 is rarely used, it is not supported in GTP version 2 and the specification of the latter GTP version is not mature yet, as of February 2009, thus the thesis focuses on GTP version 1.

The GTP protocol has three types:

- GTP-U that is used to transfer user data via tunnels in the GPRS network core. GTP-U encapsulates all user data that are transferred in the intra PLMN IP backbone network. The tunneling functionality is transparent to the MS.
- GTP-C that is used to transfer control information (signaling) between the GSNs (SGSN and GGSN). The signaling information is required to create, modify and delete tunnels that transfer user data.
- GTP' (prime). It is specified for charging purposes on the Ga interface and it is excluded from the study.

GTP runs over IP/UDP as shown in Figure 3. As a result, it adds 36 = 20 (IP) + 8 (UDP) + 8 (GTP) bytes of overhead to each data packet transferred via a GTP-U tunnel. The distinction of the two GTP types is done via the (source or destination) port used in the UDP protocol. If the port is

\[5\] For informational purposes only, GTPv2 defines the protocol for the control plane. For the user plane the GTPv1 protocol is still being used (GTPv1-U[38**]).
2123 then the GTP-C protocol follows UDP. If the port is 2152 then the GTP-U protocol follows UDP. After the GTP-U protocol usually follow the user data (IP/TCP-UDP-etc).

The way that user data are included in the GTP-U is via encapsulation and tunneling. The Tunnel Endpoint Identifier (TEID) field of the GTP header identifies the tunnel on the destination.

A user plane connection is used to transfer user data. A control plane connection is used to transfer control information and set up or terminate the user connection. For example, control information is a Create PDP Context Request message or an SGSN Context request.

2.2.1 GTP operation

The GTP header contains mandatory fields, optional fields and extension headers. The latter provide information that is not relevant to the study and need to be ignored while populating the packet contents in TAM. The optional fields carry the sequence number used in order to associate the control requests with the appropriate responses. In the main header lies the message type carried by the GTP packet and the tunnel identifier that message is destined to.

A GTP tunnel is identified via the tunnel identifiers TEID that are two numbers identifying the tunnel endpoints in both directions. A PDP context has two tunnels, one for control information and the other for data transfer. Thus, it requires four TEID numbers. The TEID are assigned locally by the receiving entity and control messages are used in order to inform the transmitting entity. In order to achieve that the first message between the source and the destination has TEID zero.

All GTP packets have a destination TEID in their GTP header that is used by the destination to select the appropriate tunnel for the incoming packet. The Create PDP Context Request message however has no control tunnel defined, thus its TEID is 0. The TEID is different in the tunnel endpoints, thus a tunnel endpoint is defined by the pair of the TEID and the GSN address.

The GTP protocol defines about 66 messages. Some are for GTP-C, others for GTP-U and other are used only in GTP'. Here only the relevant messages in order to associate the control information to user packets would be analyzed. For a complete list of the messages used please refer to [28**], Table 1. The control messages carry the control information and the TEID for a PDP context. The data messages carry the user data. The messages that concern the study are:

- Create PDP Context Request
• Create PDP Context Response
• Update PDP Context Request
• Update PDP Context Response
• Delete PDP Context Request
• Delete PDP Context Response
• SGSN Context Response
• SGSN Context Acknowledge
• G-PDU

A more detailed description of the messages is located in Appendix C.

The Create PDP Context messages define the control and data tunnel identifiers. The control TEID are used for the reception of control messages and the data TEID for the reception of data messages. These TEID numbers may be changed by update messages.

The control messages contain Information Elements (IE) that carry the information required for each message. For example, the Create PDP Context Response message contains an IE with the status of the PDP context, if it was accepted or not. Here the IE are treated as additional fields in the GTP header. Their appropriate format is included in Appendix C.

Figure 4 below illustrates a PDP context creation between the SGSN and the GGSN where the most important information carried by the control packets is presented.

Figure 4: The PDP context creation.
When the PDP context is established, two tunnels exist between the SGSN and the GGSN, one for control and the other for data traffic.

The data association to a PDP context is performed by checking the TEID contained in the data packet to the information being captured by the control packets.

2.3 RADIUS Accounting protocol

RADIUS accounting is defined in [29**] and uses the UDP destination port 1813. It consists of two entities, the NAS that acts as the RADIUS client and sends to the second entity, the RADIUS server, information about user activity via the RADIUS accounting messages.

The RADIUS client would send an Accounting start message to the RADIUS server when it starts serving a user. The client sends an Accounting stop message when it stops serving a user. The stop message may contain statistics like for how long the user session was open, the number of bytes or packets being transferred and other session related information.

RADIUS accounting consists of two packet types, Accounting-Request and Accounting-Response. The Accounting Request is sent by the client to the RADIUS server and the Accounting Response is used by the server to confirm the correct reception of the request to the client. The header format is illustrated in Appendix D. The protocol specifies attributes, similar to IE in GTP, that carry additional information.

The RADIUS Accounting Response does not need to carry any attributes and in most cases it does not. The Accounting Request carry many attributes defined by the RADIUS[30**] and the RADIUS Accounting[29**] documents and the ones important to the study are presented in Appendix D.

As it is mentioned before, the RADIUS authentication protocol is also present in the Gi interface. However, the authentication procedures are performed before a network address is assigned to the user and as a result the authentication RADIUS messages do not carry the necessary information to associate them to user traffic.

Figure 5 illustrates the RADIUS accounting procedure where the most important information carried by the packets is presented. The data identification is based on the Framed-IP-Address attribute of the RADIUS Accounting packet since the attribute carries the IP address assigned to the user.
Figure 5: RADIUS Accounting procedure.
Chapter 3 Traffic Analysis Module

The tool is a packet capture and analysis software that is based on the libpcap library\[7**\] in order to detect and capture packets in promiscuous mode. Promiscuous mode is a reception mode of the network interface that does not receive only the packets that are destined in the concerning host but rather all packets detected in the network interface. TAM consists of three primary functions:

- **Capture**
  - The function is used for online capture of network traffic packets on a network interface or offline read of packets stored in a capture file. The packets are filtered based on a filter expression, before the analysis phase.

- **Analysis**
  - The analysis phase extracts information from the IP header of the packets, categorizes them to ICMP/TCP/UDP\[6\] and aggregates the packets to flows based on source IP address, destination IP address, source port, destination port and the protocol field of the IP header (the protocol that follows IP).
  - Each flow contains the flow IP addresses and ports, the transport and application layer protocols, the time that the flow started and stopped and the number of bytes and packets transferred via the flow. The application layer protocol for each flow is detected via payload evaluation with the use of regular expressions\[5**\].
  - After being inactive for 64 seconds\[7\], the timeout value in which no new packet has been added to the flow, the flows are written to an output file in a binary form.

- **Print flows**
  - The function is used to convert the binary output file of the capture function to a user readable text file for easy interpretation.

The tool is written in the C programming language and its main futures are the aggregation of packets into flows, where the flows are stored in a hash table for fast access, and the detection of application layer protocols via regular expressions. The regular expressions are executed on the payload of

\[6\] It classifies all other protocols as Other.
\[7\] 64 seconds is a common flow timeout value as suggested in [15**].
every packet, a process that is considered relatively slow and perhaps limits the use of TAM for online capture at speeds of Gigabits per second.

The alternative is to use the port numbers used in the TCP and UDP transport protocols, in order to detect the application layer protocol. However, some applications do not follow this scheme and use port numbers that are registered for other applications (for example p2p is common to operate on TCP port 80 that is used by HTTP) or port numbers that are not registered to any application. As a result, port matching is not guaranteed to provide correct information[8**]. The advantage of using this alternative is its performance. Application detection via port matching involves only the mapping of the source and destination port numbers against a list of port numbers used per application. TAM sacrifices some performance by using pattern based application recognition for the increased accuracy of the results.

The number of recognized applications by TAM is 63. They include popular applications like HTTP and DNS, P2P applications like BitTorrent and other applications.

TAM supports only packets that have an Ethernet header followed by an IP version 4 header. It recognizes only the TCP, UDP and ICMP protocols that follow the IP protocol header, marking as Other anything else. Although the list of protocols supported by TAM seems limited, in the area of Internet traffic that is the aim of TAM, these protocols are the most popular ones[39**].

As a result, TAM does not support any of the cellular network protocols and the purpose of the study is the enhancement of TAM.

3.1 Enhancing TAM

The purpose of the new TAM features is:

• The ability to extract the IMSI and APN information (and if it does exist, the MSISDN too) from GTP control packets and associate it to the decapsulated user data from GTP user packets in the Gn interface.

• The ability to extract the MSISDN and APN information (and if it does exist, the IMSI too) from RADIUS accounting packets and associate it to user packets in the Gi interface.

Below common enhancements to both thesis goals are mentioned, where specific enhancements for the Gn and Gi interfaces are mentioned in the sections that follow.
3.1.1 Common enhancements

Initially, TAM supported only one protocol stack, being IP packets over Ethernet. However, in the cellular network it is possible to find the 802.1Q[40**] Virtual LAN (VLAN) protocol between the Ethernet and IP headers. TAM should be able to handle the protocol appropriately. A feature for this purpose has been added to the program in order to skip the 802.1Q header. It should be noted though, that by default TAM filters the incoming traffic, via the libpcap parameters used, on IP version 4. Thus in order TAM to be capable of analyzing the VLAN packets, we need to redefine the filter via the -e TAM startup parameter. The new value of the filter should be -e “vlan && ip”.

Furthermore, IP is not the only protocol following Ethernet or 802.1Q. Other protocols may, for example, be Address Resolution Protocol (ARP) or IP version 6. All packets that contain protocols other than IP version 4 should be excluded from further analysis since they are not supported by TAM. However, a future version of TAM should consider support IP version 6.

Moreover, it is common for IP packets to be fragmented. The fragmented IP packets, without considering the first fragment, carry user data but no transport layer headers, thus it is not possible for TAM to identify the flows. As a result, fragmented IP packets should not be allowed in the TAM analysis phase. The alternative is to reassemble the IP packets but it would require sufficient resources to keep track and temporary store the fragments. That is a complex procedure and would definitely decrease TAM performance in the online operation mode. However, IP fragment reassembly should be supported by a newer TAM version to provide more accurate results. Currently, TAM analyzes only the first fragment of a packet (the one containing the headers) and completely ignores the rest of the fragments (packets where their fragment offset field in the IP header is non zero). That gives results close to real life, mostly for UDP where the length of the packet is included in the UDP header. In TCP a mechanism for handling the sequence numbers has been added in order to capture the size of the fragments. Interestingly, in TCP some detected packets were fragmented in the middle of the TCP header, leaving half of the header fields (and particularly the flags field) in the second fragment. It is mentioned in [41**] that this type of fragmentation may be used in order to bypass firewalls. These fragmented packets cannot be analyzed without the correct format of the TCP header and are being skipped by TAM. Typically, at least in the Gn interface where the phenomena-

---

8 The version of TAM provided in the beginning of the thesis did not consider IP fragments. The outcome was that the program was fetching header information of the transport layer from the fragments. That is, the program was evaluating the payload as a TCP or UDP header, producing wrong results.
on occurs (in the Gi interface there is no such behavior), these user packets are encapsulated in GTP, thus only the inner(encapsulated) packet is being skipped if it is detected to carry a malformed header, and not the whole packet.

It is common for traces to contain truncated packets in order to save storage space. For example, packets larger than 200 bytes may be truncated to 200 bytes only. The libpcap library that is used to pass the captured packets to TAM, passes together with every packet the actual length of the packet in the wire and the length of the captured, perhaps truncated, packet. These values are very useful to limit the number of bytes being analyzed by TAM in every packet since the C programming language by definition does not have any pointer access constraints. The evaluation of the memory location that follows the packet length in memory space would result in reading arbitrary information and may also cause memory access errors. As a result evaluating the packet contents should guarantee that the packet length is not exceeded. Various statements have been introduced to TAM to ensure this. The phenomenon is more serious while manipulating the GTP or RADIUS control packets since a field read wrongly may affect the state of many flows and dramatically reduce accuracy.

Since the behavior of TAM should be different on the different interfaces, Internet, Gn and Gi a new startup option have been defined to select the interface. A FIFO Unix file (named pipe) has been used as input to TAM when it was necessary to run TAM over a list of files. The data provided to the pipe was the output of the mergecap program that is part of the Wireshark distribution and used to merge trace files.

The two interfaces included in the study, Gn and Gi, have the same output requirements. That is, in the Gn interface the IMSI and APN (and optionally the MSISDN) are binded to user flows where in the Gi interface the MSISDN and the APN are used (this time the optional field is the IMSI). Even the size of the data is closely related. The APN has the same size that can be up to 100 characters and the IMSI and MSISDN are of maximum 20 characters(digits) each. Thus a common flow structure and functionality can serve both interfaces. That is the approach being followed in TAM in order to avoid code repetition and complexity.

TAM uses a function that scans the application payload of each packet in order to detect the application protocols of every flow. The function does not consider port numbers in the application classification. However, that functionality is not the optimal in order to detect GTP or RADIUS packets for four reasons:

1. The recognition of the two protocols, GTP and RADIUS is based only on port numbers and port based recognition is fast and easy. It is difficult to define patterns in a binary header, and control information stemming from false positives may substantially bias the results.
2. The location of the TAM application level recognition functions is deep into the program code and it would be complex to decapsulate the GTP user packet, for example, and rerun the recognition phase on the encapsulated user packet. Assuming that the GRE or L2TP decapsulation in the Gi interface takes place before the recognition phase, it is easier to detect RADIUS during the recognition phase. The problem is that complexity increases as the new code spreads into the whole TAM structure. Thus it is considered incorrect to detect some of the protocols, GRE and L2TP, at one place and other protocols, GTP and RADIUS, in another place.

3. The Gn interface carries traffic of many users, all of which being encapsulated inside GTP and it is insufficient to execute the payload skimming functions that could take a considerable amount of time for each packet where a simple port match and verification of the GTP protocol upon packet reception is faster and simple.

4. The protocols detected are part of how the network operates and are not related to the user applications. Their detection is more similar to detect if 802.1Q follows Ethernet rather than detecting if the application layer protocol is HTTP or POP. Thus, the network protocols detection should take place before the payload evaluation.

TAM uses a flow structure that contains the IP addresses and ports of the flow together with the protocol used above the IP layer, the bytes and packets transferred on the flow and the user level application detected by TAM. In order to be able to store the IMSI and APN of the data packets aggregated to a flow a new flow structure is required. However, a new flow structure would cause the change of a large part of TAM code and, in general, how the program operates. The solution takes advantage of the C language capabilities and uses two flow structures, where the extended one is used only on the Gn or Gi interface. A startup option in TAM enables the use of this additional flow structure and store the IMSI and APN per flow. Since TAM has a function to store the binary flows to a file it should be noted here that the output of the program running on the Gn or Gi interface would be incompatible to how TAM previously stored the flows. That is expected since on a new interface that carries additional information about the flows this extra information is also required to be stored in the binary flow file. The print flow to string function has also being adapted to the new interface requirements maintaining a common output format for both interfaces. That would assist the data analysis process.

A system to report status information was also created in order to inform the user of TAM when something unexpected or interesting occurs. For ex-
ample, if a create response does not contain a cause IE, that should be indicated in the output. There are many status messages introduced to TAM. Some of them result to the whole packet being skipped, while others stop the analysis of some parts of the packet (e.g. the analysis of the GTP header, because a mandatory IE is missing) and some serve only an informational purpose (e.g. the number of delete requests detected in TAM). This status information covers also the number of packets processed by TAM, the number of identified data packets, information about the start and the stop time of the capture and more. For easy interpretation, when some status parameters are zero, they are not shown in the output. More information about the status of TAM is located in Appendix E.

Originally, TAM used the length field in the IP header in order to calculate the packet length. Although, in order to be more precise on how much data are transferred as payload, another field has been included in TAM output to show the payload of the transport layer protocol (the amount of data following the TCP header in the TCP case and the UDP header length field value in the UDP case).

Sometimes though, it is possible that the payload length measurements of the above technique being larger than the packet length calculated using the IP length when aggregated into flows. The reason is IP fragmentation of UDP packets, since the UDP length field, that carries the length of the initial packet, is used to calculate the payload. An interesting thing is that the payload approach detects also the length being skipped via fragmentation. That is, in the UDP case the length of the IP fragments being skipped by TAM, is measured via the UDP header length field. A similar solution in the TCP case is more complicated since in order to measure the correct payload of a fragmented TCP packet, it is required to study the sequence numbers of the TCP flow. Thus the payload value used on TCP flows is lacking accuracy.

In order to improve the payload length calculation for TCP flows a new field is introduced in TAM output. The field counts the payload of the TCP flow according to the sequence numbers used in the TCP session/sessions. The technique has the advantage of counting only once the bytes that are sent between the source and the destination, excluding the retransmitted bytes. That is important in order to measure the actual payload traffic of the user, ignoring any network related issues like retransmissions and header lengths.

The TCP payload approach was a challenging task, as it is difficult to analyze TCP traffic in only one flow direction. For example, it might be thought that by studying only the sequence numbers is enough in order to calculate the payload. That was not correct since TCP control information, mainly FIN and RST packets are required in order to keep track of TCP sessions\(^9\). SYN packets though can be ignored. Another problem is the possibility that the se-

\(^9\) More than one TCP sessions may be present on the same flow.
quence numbers may wrap around zero and TAM should be able to display the correct result.

The code to count the TCP payload via the TCP sequence numbers is as follows:

- For each flow the sequence number, flow_seqnum, the current counted TCP payload, flow_seqcount and the transmitted payload of previous TCP sessions on the flow, flow_tcpkeep, fields have been introduced.
- The sequence number and the payload length of the first packet of a flow is set in flow_seqnum and flow_seqcount only if the packet contains payload.
- If a subsequent flow packet contains the FIN or RST flag, the number of bytes transmitted in the flow, flow_seqcount are added to flow_tcpkeep. The other TCP fields are reset to zero.
- If a normal TCP packet with nonzero payload is encountered and the flow_seqnum is zero, the flow flow_seqnum and flow_seqcount fields are updated based on the packet information.
  - If flow_seqnum is not zero, the packet updates the flow fields only if the sequence of bytes in the packet is not already being counted in TAM. Thus, retransmitted bytes are not counted.
- If the difference of sequence numbers in the packet and the flow differ more than 100000000 bytes the numbers are considered to have wrapped around zero. In this case, the flow seq fields update the flow_tcpkeep field and the packet information is used to set these fields.

The TCP payload code verification was performed via the ratio of TCP payload to packet payload length in Trace3. Particularly, TCP / payload ratio was greater than 100 in only 5 flows. Examining the flow data showed that TCP payload was correct and the large variance was due to skipped IP fragments. Payload / TCP ratio was greater than 20 in only 6 flows and the effect is attributed to TCP retransmissions.

As a result the TAM output has been enhanced in order to show the payload calculated via the IP length field, the number of bytes following the transport layer header in the TCP header and the UDP length field based on the occasion and finally the payload of TCP packets calculated via the TCP sequence numbers. The last field should only be evaluated if the transport layer protocol of the packet is TCP and in any other case is being set to zero.

Another useful information that was included in TAM, primarily in order to facilitate comparison of TAM to other tools as is mentioned in Chapter 4, was a new output field containing the number of packets in the flow that carry payload data (payload length is greater than zero). In conjunction with the actual packet count in the flow, this field could be used in order to detect
situations where transport protocols send packets without data (mostly used by TCP to acknowledge received data).

Detecting the TCP payload length revealed a mistake in the calculation. That is because of Ethernet padding introduced in small packets. Particularly, packets with a smaller than 60 bytes Ethernet payload are being padded with zeros until they reach the minimum Ethernet payload length of 60 bytes. The reason of the mistake was that the TCP payload length calculation used the number of bytes that follow the TCP header in the packet, counting also the Ethernet padding bytes. Although the mistake seems serious it affects only TCP and only when the packet is very small. For example it is not applicable to TCP packets carried by GTP since the minimum packet size is 20 IP and 8 UDP and 8 GTP and 20 IP and 20 TCP, or 76 bytes. TCP packets carried by GRE are also not affected since the minimum packet size is 20 IP and 4 GRE and 20 IP and 20 TCP, or 64 bytes. That is also shown by the examination of the traces. In Gn Trace3 only 16 flows, all by one user, that do not use the GTP protocol experience the mistake. However in Gi, and mostly in Trace8 where most of the packets travel without any form of encapsulation, the padding downgrades the results. As a result new code was inserted in order to avoid the wrong TCP payload calculation.

Based on the point of capture in a network link, it is possible to capture packets twice, or perhaps several times too. The reason for this is that while capturing the ports of a router, it is possible to detect a packet both as inbound and outbound traffic of the router. This duplicate packet does not actually exist many times in the network but it is a creation of the capture mechanism. The analysis of duplicate packets is possible to affect the results of a network analysis program and it is required to exclude these packets from the study.

The detection and omission of duplicate packets in flows has been incorporated into TAM via the IP identification field. Based on the IP version 4 specification\[42**\], the Identification field of an IP header should be used in order to merge the fragments of an IP packet. The field should not be evaluated when the packet is not an IP fragment. However, many IP implementations set the Identification field in all the packet they transmit to the network, making it possible to detect duplicate packets.

It is not safe though to count only the IP Identification field in order to detect duplicate packets as it is possible for packets to have the same IP ID but different contents. For example, the ID value of zero is considered as a special value since some implementations use this value for all non fragmented IP packets they transmit.

As mostly tunneled traffic is carried in the Gn and Gi interfaces, the IP ID field of the encapsulated IP packet is also checked as it is possible for the encapsulated IP packet to have a different ID number and the outer IP packet to have the same ID in two packets belonging to the same flow.
The procedure of detecting and omitting IP duplicate packets in TAM is based on heuristics and is as follows:

- For each flow, the IP ID, IP len, outer IP ID and transport checksum fields have been introduced.
- The first packet on the flow updates the above fields. Where no outer IP header exists, the IP ID and outer IP ID fields contain the same value.
- Every subsequent packet for the flow is being checked along these fields and if all fields match, the difference of the packets in reception time is less than 1 second and none of the ID fields is zero, the packet is considered as duplicate and it is omitted, without affecting the statistics of the flow (packet count and byte count).
  - For a duplicate packet, if the two ID fields are different the packet is considered as a duplicate inner, or encapsulated, IP packet.
- If a packet is not duplicate it updates the flow fields. Thus only the last packet for each flow can be detected as duplicate, and not the packets previously aggregated in the flow.

The transport layer checksum, TCP or UDP header checksum field, is used since in the Gi interface packets were detected that were not duplicate as they carried different payload but still met all prerequisites to be flagged as duplicate packets. That is, these packets had the same nonzero IP ID, same length, and less than 1 second difference in time. The use of the checksum field helped to recognize these packets as non duplicate.

It should be noted that using the IP checksum field to indicate a duplicate packet is not correct since the packet probably traveled from one end of the router to the other before being captured the second time (as duplicate) and this traversing from a router reduces the TTL value of the packet by 1 that requires a new, different, IP checksum to be calculated for the packet.

The duplicate packet detection function works along with the flow aggregation function as it uses the flow structure in order to save the required state information. Since processing of control traffic (GTP and RADIUS) is performed before the flow aggregation function, the duplicate packets are processed as control traffic if they contain control information. For example a duplicate Create PDP Context Request packet would be processed by the GTP code but ignored since the original packet had already created the PDP context in TAM. Later when the duplicate packet is aggregated to a flow it can be detected as duplicate and skipped.

The above code exempts for handling the TCP payload via sequence numbers and the detection of duplicate packets are considered as heuristics without some specific definition. Thus, the outcome of this code is not considered fail-proof and may decrease the credibility of the results. Thus, results based on the above code should be studied for validity. Perhaps a better
approach to study TCP flows would be to incorporate tools that are designed for this purpose, like e.g. Tcptrace[43**]. IP duplicate packets could be dealt consistently as part of IP fragmentation reassembly in a future version of TAM. From this heuristic code, only the code about duplicate packets may degrade the results. The TCP payload calculation uses an extra field in TAM output and in case the calculation is wrong the field can always be skipped. As a result, in order to limit the potential damage of a malfunctioning heuristic, the omission of duplicate packets is controlled via an additional TAM startup option. When the option -d is defined in TAM input, the program does not skip duplicate packets.

Below the specific enhancements to the Gn and Gi interfaces are presented. They involve the processing of control information, GTP control or RADIUS packets, in order to associate the IMSI, APN and MSISDN information to user packets. However, note that while the control messages are correctly processed by TAM it is possible for the user to not communicate in the session. As a result the information stored by TAM, like the user IMSI, is not displayed in TAM output. The program identifies only the packets that carry user data. None of the detected control packets is being associated with a PDP context or a RADIUS session.

A requirement for comparing the TAM output to other tools as presented in paragraph 4.5 and having a standard flow output was to produce output that consists of bidirectional flows. Since TAM outputs unidirectional flows, and in order to maintain the TAM structure intact, the program could not be enhanced to produce bidirectional flows. Thus, a script has been used to process the TAM output and aggregate the unidirectional flows to bidirectional.

The script and the statistics collected require the direction of the flows to be known. Thus, a feature was added to TAM to assign the direction of the flow (being either uplink or downlink) to all identified data flows. The direction of unidentified flows could not be defined with certainty, thus they are being marked with unknown direction. The flow direction is being output by TAM and its used by the script producing the bidirectional flows in order to present a common flow output. For more information refer to section 4.4.

3.2 Enhancements for the Gn interface

On the Gn interface the GTP packets should be analyzed and state should be maintained in order to associate GTP control information to user data.

First, the GTP packets should be detected. That happens before the TAM analysis of the payload for performance reasons as a simple comparison of the UDP ports against the ports used by GTP for control and user communic-
Detection is faster than trying to match many regular expressions to the UDP payload.

**Detect a GTP packet**

- If `Packet.UDP_Src_Port equals 2123` or `Packet.UDP_Dst_Port equals 2123`
  - **Packet carries GTP-C**
- If `Packet.UDP_Src_Port equals 2152` or `Packet.UDP_Dst_Port equals 2152`
  - **Packet carries GTP-U**

As is illustrated above, the port based GTP classification has been used. If one of the UDP ports (source or destination) is 2123 then the GTP-C protocol is detected. If one of the UDP ports is 2152 then the GTP-U protocol is detected. In either case verification of the GTP version and type is being done, as is shown below, to ensure that the packet carries the GTP protocol.

**Verify the GTP Protocol**

- If `Packet.GTP_Version equals 1` and `Packet.GTP_Type equals 1`
  - **Packet is classified as GTP (either GTP-C or GTP-U)**

When the packet is verified as being GTP, the message field of the GTP header is evaluated in order to select the following actions.

The GTP packets may contain extension fields, that carry no information concerning the study. These fields should be skipped appropriately when the GTP header is evaluated.

The GTP control messages have their application level protocol being identified as PROTO_ID_GTP_C in TAM. It could be also possible that GTP user messages contain no user data (or their format is unrecognizable by TAM, e.g. PPP\(^\text{10}\)). In this case the packets are being identified as PROTO_ID_GTP_U. In the above two cases the GTP specific code have identified the application of the packet so there is no reason to try to match the application via regular expressions. Thus, these packets are excluded from the TAM payload evaluation function. When the GTP user messages contain user data, the first IP header of the packet together with the UDP and GTP headers are being skipped and the rest of the packet (the encapsulated IP header together with the user data) is sent to TAM to detect the application level protocol and aggregate the packets to flows.

All GTP control messages analyzed by TAM should have a valid sequence number (the S flag in the GTP header should be 1). In case a control message packet does not have a valid sequence number it is immediately discarded.

\(^{10}\) The PPP protocol is not supported on the Gn interface mainly due to the fact that very few packets were detected and none of them encapsulated IP traffic.
That is in accordance to the GTP specification of using the optional sequence number field in the GTP header in order to associate the request with the response control messages.

In the GTP protocol only the Create PDP Context Request/Response, Update PDP Context Request/Response and Delete PDP Context Request/Response control messages are used to maintain the state of the GTP protocol. These are the most common control messages in the GTP and the ones that carry the necessary information to identify data flows. The Create messages contain the IMSI and APN that should be binded to user flows together with the TEID to detect the flows. The Update messages may contain different TEID than the Create messages, thus an analysis of the updates is required to be able to follow the user flows consistently. Delete messages are used to delete expired state information, for performance and consistency reasons, in TAM. Delete messages control the size of the state information maintained in TAM by deleting items that are not used anymore and if present may cause biased results.

These six control messages have been increased by another two messages in order to improve TAM success of associating control information to user data. The two new messages are the SGSN Context Response and the SGSN Context Acknowledge. They are used by the GTP protocol when a handover procedure takes place for a user from one SGSN to another SGSN, and carry the PDP context information maintained in the old SGSN to the new one. An update procedure follows the SGSN context messages in order to inform the GGSN of the change. If the PDP context already exists in TAM, the SGSN procedure is not required. However, when TAM is unaware of the PDP context (TAM missed the Create procedure) the SGSN context messages may be used to construct the PDP context. That is because the PDP context information carried along an SGSN Context Response includes the IMSI, APN and tunnel identifiers required to build the PDP context in TAM.

The update procedure is also important because if one GSN would like to configure the PDP context to change for example the QoS or the user address it is likely that the tunnel endpoints would also change. Another reason to change the tunnel endpoints is if direct tunneling is implemented. Direct tunneling is a tunnel for the user traffic only that connects directly the RNC with the GGSN without the SGSN intervening in the middle. However, the SGSN still maintains the management of the control traffic.

In the GPRS network, the GSNs can easily detect the PDP contexts on the received packets by using the TEID field in the packets' GTP header. A packet capture program that runs in the middle of SGSN and GGSN in the Gn interface, like TAM does, cannot use only the TEID fields to relate detected packets to PDP contexts because the TEID are not defined as unique in the GPRS core network. The TEID are unique in the GSN that assigns them. That GSN then uses control messages to inform other GSNs of the TEID values it has assigned to the PDP context. The other GSNs must use these
TEID when sending messages (control and data) to this GSN. As a result, TAM needs also to monitor the IP addresses for each PDP context and relate the captured packets to PDP contexts via their destination IP address and TEID. That is the reason why the data structure used in TAM for the storage of state information, as is shown in Table 1, contains so many fields.

In the beginning of the study a simple two-way linked list data structure was used for the storage of the state information of every PDP context and the association of that information to user flows. Every list item contained the IMSI and APN of the PDP context together with the TEID and GSN addresses for control and data. Additional information was also included in the list items to facilitate the correct function of the GTP protocol. The size of every list item was about 200 bytes. That, together with the fact that the searching functionality in a linked list requires checking all items from the beginning of the list, reduced the performance of TAM considerably. That is because a GTP trace file may contain thousands of PDP contexts, creating thousands of records in the linked list (one PDP context is one linked list item) and for every GTP packet it is required to locate the PDP context in the list in order to be able to extract the IMSI and APN. During the study a better way of handling the GTP state was introduced as is shown later.

A PDP context is being uniquely identified via its IMSI and Network Service Access Point Identifier (NSAPI) parameters. The NSAPI is a 4 bit number with valid values between 5 and 15 [44**]. Thus, only 11 PDP contexts may be active per user. A user may select to start one or several primary PDP contexts. The primary PDP contexts may differ in all parameters (APN, EUA, QoS) and only share the same user (IMSI). They may be used to access other networks than IP. For example, two PDP contexts may be active, one for an IP network and the other for X.25. Another choice for the user is to start a primary PDP context and a secondary PDP context sharing most of the properties with the primary PDP context, but with different QoS.

In order to detect the IE in a GTP control message a new function has been implemented. The function skips the unnecessary IE (the IE with not relevant information) and returns an error if it is unable to locate the IE of interest. The appearance of IE in a GTP header is defined in the specification, as their order of appearance. That order is based on the IE type field being sorted. That is, an IE with a small type value appears before an IE with a larger value. On the other hand the specification defines that some IE may exist multiple times inside a GTP header. The function returns the location of the first occurrence of the IE of interest in the GTP header. For performance reasons, an index is used after which the function should look into the packet header and a limit defines where the function should stop. For example, searching for the NSAPI that is a mandatory IE and comes 9th in a Create PDP Context Request message, requires to skip the previous eight IE so the

---

11 Assuming that all conditional and optional IE are included in the packet.
function starts execution in byte 30 and not byte 0 inside the part of the GTP header containing the IE. The length of the NSAPI is 2 bytes. In order to avoid checking the whole packet, the function is limited to byte 32 and if by passing the limit it has not detected the NSAPI it returns an error.

A linked list item contains the following fields:

Table 1: The linked list fields on the Gn interface.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMSI</td>
<td>the IMSI extracted from the Create PDP Context Request or the SGSN Context Response</td>
</tr>
<tr>
<td>APN</td>
<td>the APN extracted from the Create PDP Context Request or the SGSN Context Response</td>
</tr>
<tr>
<td>MSISDN</td>
<td>the user phone number. It may exist in the Create request.</td>
</tr>
<tr>
<td>End User Address</td>
<td>the address being assigned to the user. It is required in the delete process.</td>
</tr>
<tr>
<td>NSAPI</td>
<td>the NSAPI value for the PDP context.</td>
</tr>
<tr>
<td>TEID control S</td>
<td>the tunnel identifier for control messages that have as destination the SGSN.</td>
</tr>
<tr>
<td>IP control S</td>
<td>the IP address for control messages that have as destination the SGSN.</td>
</tr>
<tr>
<td>TEID control G</td>
<td>the tunnel identifier for control messages that have as destination the GGSN.</td>
</tr>
<tr>
<td>IP control G</td>
<td>the IP address for control messages that have as destination the GGSN.</td>
</tr>
<tr>
<td>TEID data S</td>
<td>the tunnel identifier for data messages that have as destination the SGSN.</td>
</tr>
<tr>
<td>IP data S</td>
<td>the IP address for data messages that have as destination the SGSN.</td>
</tr>
<tr>
<td>TEID data G</td>
<td>the tunnel identifier for data messages that have as destination the GGSN.</td>
</tr>
<tr>
<td>IP data G</td>
<td>the IP address for data messages that have as destination the GGSN.</td>
</tr>
<tr>
<td>SEQ number</td>
<td>the sequence number in the Create PDP Context Request. It is used to bind the create request to the create response message.</td>
</tr>
<tr>
<td>UPDATE TEID control S</td>
<td>the tunnel identifier for control messages that have as destination the SGSN. Used in the update procedure.</td>
</tr>
<tr>
<td>UPDATE IP control S</td>
<td>the IP address for control messages that have as destination the SGSN. Used in the update procedure.</td>
</tr>
<tr>
<td>UPDATE TEID data S</td>
<td>the tunnel identifier for data messages that have as destination the SGSN. Used in the update procedure.</td>
</tr>
<tr>
<td>UPDATE IP data S</td>
<td>the IP address for data messages that have as destination the SGSN. Used in the update procedure.</td>
</tr>
<tr>
<td>UPDATE SEQ number</td>
<td>the sequence number in the Update PDP Context Request. It is used to bind the update request to the update response message.</td>
</tr>
<tr>
<td>UPDATE End User Address</td>
<td>the address to be assigned to the user in the update process. It is required when the update is rejected.</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DELETE SEQ number</td>
<td>the sequence number in the <em>Delete PDP Context Request</em>. It is used to bind the delete request to the delete response message.</td>
</tr>
<tr>
<td>Teardown</td>
<td>A flag in the <em>Delete PDP Context Request</em>. It is required in the <em>Delete PDP Context Response</em> where the deletion of the PDP context takes place.</td>
</tr>
<tr>
<td>SGSN SEQ number</td>
<td>the sequence number in the <em>SGSN Context Response</em>. It is used to bind the <em>SGSN Context Response</em> to the <em>SGSN Context Acknowledge</em> message.</td>
</tr>
<tr>
<td>PDP context ID</td>
<td>a value assigned locally by TAM to uniquely identify a detected PDP context</td>
</tr>
</tbody>
</table>

The UPDATE fields are set by an update request message originating from the SGSN that has different fields from the ones set by the create message. They are used to store the values until an update response for the PDP context is received. Upon reception of a successful update response, these fields replace the normal ones. If the update response is rejected, the fields are being reset.

The GSN addresses in the GTP control messages are usually different from the addresses given by the IP layer. That is the reason why these fields are analyzed and stored in the linked list.

The use of the linked list and not any other data structure is because of the number of different queries performed on the list. These queries include:

- the TEID control S or D, IP control S or D and NSAPI
  - to find relative entries for the Delete Request message
- the TEID control S or D, IP control S or D and DELETE Seq number
  - to find relative entries for the Delete Response message
- the EUA and IMSI
  - to find relative entries for deletion on the Delete Response message
- the TEID control G, IP control G and NSAPI
  - to find relative entries for Update Request messages
  - to find relative primary entry for Secondary Create Request messages
- the TEID control S, IP control S and NSAPI
  - to find relative entries for Update Request messages
- the TEID control S, the TEID control SU, IP control S, IP control SU and UPDATE Seq number
  - to find relative entries for Update Response messages
- the TEID control D, IP control D and UPDATE Seq number
  - to find relative entries for Update Response messages
- the TEID control S, IP control S, the TEID data S, IP data S, NSAPI and IMSI
  - to detect duplicate linked list entries or Create Request messages
- the TEID control S, IP control S and CREATE Seq number
  - to find relative entries for Create Response messages
- the IMSI, NSAPI
  - to detect possible duplicate linked list entries on SGSN Response messages
- the TEID control S, IP control S and SGSN Seq number
  - to find relative entries and delete them for SGSN Acknowledge messages
- the TEID control S, IP control S, SGSN Seq number and NSAPI
  - to find relative entries for SGSN Acknowledge messages

Two more queries, used in order to associate the control traffic to data packets, are:
- the TEID data S, IP data S
- the TEID data G, IP data G

The last two queries are the ones occurring most frequently, once for every data packet, while the rest of the linked list queries occur only when the appropriate GTP control message is being detected. As is mentioned later, the performance of TAM is mainly reduced due to the last two queries shown above as the control messages are only a small portion of the GTP packets traveling the GPRS core network. These two queries have been removed and a hash table has been introduced to avoid the linked list's drawbacks.

When a data packet is received (a GTP G-PDU message), its internal encapsulated IP packet is being extracted and passed to TAM. Furthermore, the TEID of the GTP header together with the destination IP address in the external IP header are used to search in the linked list\textsuperscript{12}, using the two queries shown above, for the PDP context related to the packet. If the PDP context is located in the list the IMSI and APN are being copied from the list and associated to the packet. It is not uncommon that the PDP context does not exist in the linked list. That may happen, for example, when the Create PDP Context Request for the PDP context was not captured.

All Create Requests originate, by definition, from an SGSN and have as destination a GGSN. When a Create PDP Context Request is received, its TEID is checked. A value equal to zero means that the create request initiates a new PDP context. The IE of the create request are being analyzed and used to create a new list item. The create request contains the IMSI, the APN, the NSAPI, the control and data TEID and IP addresses in the SGSN side and possibly an EUA assigned to the user. When the TEID in the create request is

\textsuperscript{12} Here the linked list approach is mentioned. In the performance section is described how the hash approach for the last two queries of the linked list has been implemented.
not zero, the request is identified as a secondary PDP context request. This request is related to an already active PDP context and they share the same IMSI, APN and EUA. The secondary request differs only in the QoS, Protocol Configuration Options (PCO) and Traffic Flow Templates (TFT)[10] parameters. The latter are used in the GGSN to filter packets as defined in the GTP specification. The secondary request contains the NSAPI of the linked PDP context that is the already active PDP context being related to the secondary PDP context. TAM uses the Linked NSAPI (LNSAPI) and the TEID in the secondary context request to locate the related PDP context in the list. The IMSI, APN and End User Address (EUA) fields from the located list item are used to create a new list item for the secondary PDP context.

A Create PDP Context Request sets up only the SGSN side of the PDP context. To complete the picture, the Create PDP Context Response must also be received and analyzed. The TEID, destination IP address and Sequence number in the GTP header of the response are used to locate the PDP context in the list. The response contains the Cause IE that indicates if the request is being accepted by the GGSN or not. A non accepted request results in the deletion of item from the list. A response with a CAUSE equal to Request Accepted(128) updates the PDP context in the list to include the data and control TEID and IP addresses for the GGSN side of the PDP context. When the PDP context is being set up, the sequence number in the list item is being reset since it is not needed anymore.

The update request sets the UPDATE fields in the list item and the successful update response uses these fields to update the normal list item fields. A non successful update response may have as a result the deletion of the PDP context from the list. The GTP specification defines the deletion of the PDP context if the Update response was rejected when the Cause IE in the Update response is equal to Non_Existent PDP context or the Update request contained a new EUA for the user.

The delete requests are used to delete one or several PDP contexts from the list. An IE in the delete request, the Teardown Indication specifies whether all PDP contexts that share the same PDP address (end user address) as the PDP context in the delete request should be also deleted. That happens when the Teardown Indication is true. When the Teardown Indication is false only the PDP context referred to by the delete request is deleted. However, a simple search in the list for the EUA in order to delete these PDP contexts is not correct\(^\text{13}\) since in the GTP specification it is mentioned that the PDP contexts that have the same PDP address within the same MS should be deleted. TAM uses the IMSI together with the EUA to delete the PDP contexts when the Teardown Indication is true.

\(^{13}\) It is common for GGSN to run NAT and the author believes that is possible two different GGSN to assign the same EUA to their users.
The delete response is used as a reply to the delete request in the GTP specification. However, in the specification is defined that PDP context should be deleted when the delete request is detected. So the GSN that sends the delete request it should also delete the PDP context at that time. The GSN that receives the delete request should delete the PDP context and then reply with a delete response. As a result, TAM only needs to handle the delete request messages and ignore the delete response ones. A negative aspect is that the GSN that is the destination of the delete request, may send user packets in the PDP context in the mean time between the delete request has been send and the delete request has been received and processed. It is cleat that, since TAM lies in the middle of the two GSN, it would detect the delete request before the recipient. As a result TAM would delete the PDP context first. Packets traveling the PDP context after TAM deleting it would fail the association of IMSI and APN since the PDP context does not anymore exist in TAM. That is not wrong since the initiation of a delete request means that the user packets following it would probably not be forwarded to their destination (MS or external network) because the PDP context is being deleted. In the studied GTP traces the above situation happened in the last one to three user packets in some PDP contexts while they were being deleted. However, the latest version of TAM uses the Delete PDP Context Response to delete the PDP context in order to associate more packets with the PDP context as if the packets were skipped they could produce several new flows in the program's output. That is a consequence of the PDP ID used in TAM aggregation function.

The GTP IE that contains the IMSI in the Create PDP Context Request has it encoded in a TBCD form [28**]. The IMSI is converted to a character string before being stored in the list item. The same applies to the MSISDN IE, excluding the first data byte that is used only for indication purposes[45**]. The APN has its own structure in the APN IE[44**]. It is divided into a number of sections where each section starts with a byte containing the length of the section and the following bytes contain the section value in characters. The APN is also converted to a character string before being stored in the list item. The bytes that contain each section length(excluding the first one) are replaced by dot '.' characters. For example an APN may look like 0x03 W W W 0x08 E R I C S S O N 0x03 C O M and the output of TAM would be WWW.ERICSSON.COM

The IMSI and APN (and if exists, the MSISDN too) are stored in the flow structure and being printed for every flow in the TAM output.
Enhancements for the Gi interface

On the Gi interface TAM should detect the RADIUS messages that carry the APN and MSISDN (and if present the IMSI too) of the user and associate them to the user packets.

The Gi interface is not specified as well as the Gn. That is because the target of the Gn interface is to transparently transfer user data and do not interfere with the actual protocols used in the user data. The Gi interface does not carry the user data transparently, thus it may have many different protocols and networks to connect to. The use of protocols in the Gi interface is operator (carrier) dependent. However, the most common protocol used to carry the packet data is IP. And RADIUS is commonly used for authentication and accounting purposes.

Operators often use GRE on the Gi interface that is a tunneling protocol on top of IP. The format of the GRE header and additional information about the protocol is mentioned in Appendix D. TAM filters and drops the GRE protocol together with the outer IP header of the packets before processing takes place in the Gi interface. Another tunneling mechanism used is to tunnel PPP frames via L2TP[7]. Since the PPP frames may carry IP packets, they should be analyzed in TAM. L2TP tunnels network traffic via IP/UDP using UDP port 1701. TAM ignores the L2TP protocol and the outer IP header and processes only the encapsulated IP packets.

The procedure to be performed by TAM on the Gi interface is relatively simpler than analyzing the GTP protocol. Only one Radius message should be detected and analyzed, and that is the accounting request\textsuperscript{14}. The Accounting Request contains all fields necessary to bind the APN and MSISDN to user data. The binding is performed by using the IP address of the user carried in the RADIUS accounting messages and detecting the packets that use this IP address as a source or destination.

TAM detects the RADIUS Accounting protocol via the same mechanism used for the GTP case, port matching, as is shown below:

\textit{Detect a RADIUS packet}

\textbf{If} Packet.UDP\_Src\_Port equals 1813 \textbf{or} Packet.UDP\_Dst\_Port equals 1813  
\hspace{1cm} Packet carries RADIUS Accounting  
\hspace{1cm} Packet is classified as RADIUS Accounting

The RADIUS message identified by TAM is only the Accounting Request message since the response does not carry important information. The Accounting Response message does not carry important information, it is rather a response to the request. Furthermore, the Radius Access messages used for authentication purposes do not carry the IP address of the user and, thus, are not useful in the user profiling process.

\textsuperscript{14} The Accounting Response message does not carry important information, it is rather a response to the request. Furthermore, the Radius Access messages used for authentication purposes do not carry the IP address of the user and, thus, are not useful in the user profiling process.
counting Status Type attribute of the RADIUS Accounting Request message controls the creation and deletion of state information in TAM. A value of 1 (start) creates a new linked list entry in TAM and a value of 2 (stop) deletes the relative entry, if it exists in TAM.

The period between the Accounting Request start to the Accounting Request stop is known as an Accounting or RADIUS session, similarly to the PDP context in GTP. It is the period that the user data can be associated to the control information of the RADIUS messages.

Similarly to the Gn interface, a linked list in TAM maintains the state required for the association. An item on the list contains the following fields:

<table>
<thead>
<tr>
<th>Framed IP Address</th>
<th>the network address assigned to the user</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accounting session ID</td>
<td>The ID of the accounting session used to bind together the Accounting Request start and stop messages</td>
</tr>
<tr>
<td>MSISDN</td>
<td>the user phone number. Named as Calling Station ID in the RADIUS terminology</td>
</tr>
<tr>
<td>APN</td>
<td>the APN is called Called Station ID in RADIUS</td>
</tr>
<tr>
<td>IMSI</td>
<td>the IMSI of the user may be carried in the RADIUS messages as a 3GPP Vendor Specific attribute</td>
</tr>
<tr>
<td>RADIUS ID</td>
<td>a value assigned locally by TAM to uniquely identify every detected Accounting Session</td>
</tr>
</tbody>
</table>

The Accounting Request start message creates a new list item and fills the above fields. It then creates a new entry in the hash table to facilitate fast search for the MSISDN and APN in the user packets. The hash table operates similarly to the hash table on the Gn interface.

The association of the MSISDN, APN and the IMSI if exists is done via the matching of the user IP address in the data packets.

The Accounting Request stop message locates the list entry via the accounting session ID and the user address and deletes it.

The queries on the linked list on the Gi interface include:
- the Framed-IP-Address, Accounting Session ID, MSISDN and APN
  - in order to detect possible duplicate linked list entries on the Accounting Request Start messages
- the Framed-IP-Address and the Accounting Session ID
  - to find relative entries for the Accounting Request Stop messages

One more query, used for the association of control traffic to data packets, is:
- the Framed-IP-Address

The use of a linked list on the Gi interface rather than some other data structure, considering that the processing required is simpler without the many queries that GTP uses, is because: i) it keeps the same structure in the
two interfaces, ii) it is easy to extend the list in case more state information, like support for RADIUS authentication, needs to be maintained and iii) serves as a location to store the data and use short entries and the fast searching abilities of a hash table to link to them.

The RADIUS protocol includes also RADIUS authentication messages. However, these messages cannot help identify user flows as they do not carry the Framed IP Address attribute that contains the network address of the end user. That is supported by all Gi traces presented below.

Furthermore, according to the specification of the Gi interface the authentication messages are used in order to grant access for the user. When the user is authenticated then an IP address is assigned to the user. As a result it is highly unlikely the RADIUS authentication messages to contain the Framed IP Address attribute.

The RADIUS accounting messages are sent after access is granted for the user and the user IP address becomes known. As a result the accounting messages are the ones to be processed in order to associate control information to user traffic in the Gi interface.
Another reason to not analyze the RADIUS access messages is illustrated in Figure 6. As the figure shows, there is no RADIUS authentication message used to signal the end of the RADIUS session. As a result even when the session is terminated, the state information would remain in TAM assuming that it captures and analyzes the RADIUS access messages. The RADIUS accounting messages are used both in the beginning and the end of the RADIUS session, making them more appropriate for session tracking.
3.4 A user filter

A requirement introduced during the project was a filter being implemented in TAM, in order for the program to process the data that belonged to one or several users by using the IMSI and the MSISDN in the Gn and Gi interfaces respectively.

As a result a new option has been added in TAM (-u) that takes as argument the name of a file that contains the identification of the users that their data should be examined by TAM.

The file is loaded on startup by TAM, creating a new linked list with the appropriate values, IMSI on Gn interface and MSISDN on Gi interface. If the default TAM interface (Internet) is used, the option is ignored.

If the option is enabled, only the control information that concerns the user identification passed to TAM is processed by the program. Furthermore, only flows with data packets that can be identified are being output. That leaves out any control traffic.
Chapter 4 A real world trace study

On this chapter the experiences acquired by testing TAM on real world trace data are expressed. The purpose is twofold. To verify TAM functionality in the real world where verification and consistency are considered and to analyze the TAM output and draw conclusions.

The traces provided were all offline, acquired by undisclosed tools in undisclosed locations at undisclosed times, and their characteristics are as follows:

Table 3: The studied traces.

<table>
<thead>
<tr>
<th>Trace name</th>
<th># of pkts</th>
<th>Trace size (MB)</th>
<th>Duration (seconds)</th>
<th>Identified %</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace 1 Gd</td>
<td>30805433</td>
<td>4577</td>
<td>12988.9</td>
<td>33.5</td>
<td>VLAN is present GTP: Truncated Create Request messages Trace: Truncated packets to 200 bytes Longest duration</td>
</tr>
<tr>
<td>Trace 2 Gd</td>
<td>17051083</td>
<td>5475</td>
<td>1027.11</td>
<td>26.3</td>
<td>Trace: Some files were damaged</td>
</tr>
<tr>
<td>Trace 3 Gd</td>
<td>54981559</td>
<td>33318</td>
<td>2879.68</td>
<td>40.5</td>
<td>GTP: SGSN Response messages can help build a PDP Context Trace: Largest in size Very good success rate</td>
</tr>
<tr>
<td>Trace 4 Gi</td>
<td>16978482</td>
<td>5608</td>
<td>1026.89</td>
<td>21</td>
<td>Radius: Accounting only (with IMSI) GRE: Identified packets have GRE Non GRE packets cannot be identified Trace: Some files were damaged</td>
</tr>
<tr>
<td>Trace 5 Gi</td>
<td>57487</td>
<td>22</td>
<td>1453.19</td>
<td>64.5</td>
<td>Radius: Accounting only Tunnels: No GRE/L2TP messages Trace: Old trace, of long duration Great success rate</td>
</tr>
<tr>
<td>Trace 6 Gi</td>
<td>413193</td>
<td>199</td>
<td>5.55</td>
<td>0.3</td>
<td>Radius: Accounting only GRE is present L2TP/PPP is present Trace: Very short duration Only non GRE and non L2TP/PPP</td>
</tr>
</tbody>
</table>
As is pointed in the above table the most important characteristics of each trace are as follows:

Trace1: The truncation of packets to 200 bytes that limited the number of valid GTP control messages being examined because mandatory information was not captured.

Trace2 and Trace4: This is actually one trace that contained data from both interfaces. Although the packets were not truncated, some trace files were damaged limiting the quality of the trace.

Trace3: While the duration of the trace is not long, the trace is considered the high quality trace of the study for the Gn interface since it has no truncated packets and a very good success rate.

Trace5: An old Gi trace. Its duration is very long for such a small trace size pointing out that at that time user traffic was not dominant.

Trace6: The most important contribution of the trace was the existence of L2TP/PPP traffic (the only trace contained this type of traffic). However, this traffic could not be identified.

Trace7: The trace contained both RADIUS authentication and accounting packets. However, the packets were truncated to 200 bytes, prohibiting TAM to detect all necessary control information. Particularly, RADIUS Access messages did not contain the Framed IP Address attribute that is necessary in order to create a RADIUS session in TAM. The RADIUS Accounting messages were truncated so the MSISDN could not be fetched, causing TAM to ignore the messages. The program was unable to even create just one RADIUS session, explaining the zero success rate here.

Trace8: This is the second Gi trace containing both RADIUS authentication and accounting packets. Here, same as in Trace7, the RADIUS Access messages do not contain the Framed IP Address attribute. Since this trace has no truncated packets or damaged files it is considered as the high quality trace of the Gi interface. The success rate is low mainly because of the very short trace duration.

We have been provided with 3 traces for the Gn interface and 5 for the Gi interface. As it is described in [25], high quality traces are required in order to improve accuracy. These traces should capture all frames from the link of study and include all information of control traffic. From the above traces,
only Trace 3, 5, 6 and 8 meet the prerequisites of a high quality trace. Trace 5 is too old to be proven useful and as a result is not analyzed further. For the rest high quality traces their duration is also considered as a parameter since longer traces are better as they capture more network information. Only trace 3 and 8 are left as high quality and long duration traces and they would be the main source of results in the following sections. The rest of the traces either lack on capture problems, are too old or too sort in duration.

Concerning the number of identified data packets, it is clear that some PDP contexts or RADIUS sessions may have started before the capture takes place. An effect that is called border effect in [36]. These PDP contexts or RADIUS sessions may not be detected by TAM since simply their control messages are not captured and analyzed.

4.1 Problems Encountered

Below the most important problems encountered while studying the trace files are presented. The problems are categorized in 5 types. Trace file problems, IP fragmentation, duplicate list entries, C language problems and problems of the original TAM version.

4.1.1 Trace file problems

Trace 1 could not be examined by TAM because it contained 802.1Q VLAN protocol following the Ethernet header. That required the support of the protocol in TAM. Furthermore, TAM filtered out by default, using libpcap filters, any packet that contained an Ethernet protocol different than IP version 4 since only IPv4 packets are supported. Thus, in order to support VLAN the startup TAM parameters had to be changed.

It is common while capturing traffic to limit (truncate) the captured packet sizes to small values in order to save disk space. For example, all packets may be limited to 200 bytes. The capturing software should be able to detect this and limit its parsing capabilities based on the captured rather than the original packet size. If the capturing size is exceeded in the packet analysis it is possible to bias the results or prevent the normal execution of the program. That problem was present in Trace 1 where the capture packet size was 200 bytes and some control messages (GTP create requests) had size larger than that. By default TAM did not consider the truncation until it was processing the payload of the packets and it was required to have TAM use the captured
packet size variable provided by the libpcap library for every packet and adjust the program accordingly to be able to solve the problem.

It should be noted that in order to be safe the program should be positive that the data are present before it reads them. Thus a number of checks have been introduced to TAM to ensure this. Failure to verify that the whole data are present, like the TCP header being at least 20 bytes, has the result of omitting the packet and inform the status mechanism of TAM.

Finally, Trace2 and 4 had some files being partially damaged, probably during transfer. As a result data were missing from various parts of the traces, affecting the results.

4.1.2 IP fragmentation

IP fragmentation occurs when an IP packet is too big to be transferred by the lower layer, like more than 1500 bytes for Ethernet. As a result the IP packet needs to be divided into smaller data portions called fragments. An IP header is appended to each fragment with the information contained in the IP header of the original packet and additional information to inform the receiver how to reassemble the fragments. For a study on IP fragmentation refer to [52**]. The paper mentions that IP fragmentation reduces the performance of the core network as routers must capture and reassemble IP packets thus it should be avoided. That is the case on Internet where most of the times path MTU discovery is used. However IP fragmentation still exists, mainly because of tunneling protocols like GTP or GRE used to transport data.

The Gn interface always uses tunneling via the GTP user messages, thus it should be expected, and that is the case, to detect a considerable amount of fragmented IP packets. The Gi interface may use tunneling via L2TP/PPP or GRE and on these cases it is more probable to detect IP fragments on it.

The original version of TAM did not check for IP fragments, completely ignoring the phenomenon. That was a problem since TAM was evaluating the payload of the fragmented packets (fragments 2 to n) as header information of the transportation layer. For example, the second fragment of a packet carrying UDP contains only part of the UDP packet payload and not the UDP header. However, since TAM did not check for fragments, it used this packet payload to get the UDP ports. As a result, spurious flows have been produced.

There are two approaches in order to solve the problem. Either reassemble the fragments to the original IP packets, a complicated and memory consuming process, or just skip the fragments since they contain only payload information and no protocol headers. In TAM the second approach is used. The packets are skipped when the fragment offset field of the IP header is other than zero. The first packet in a series of packet fragments always has
fragment offset as zero. The rest of the packets update the fragment offset field accordingly. These latter packets are being skipped in TAM. Furthermore, it is possible to have IP fragments inside the encapsulated data. It is not known why IP fragments are present on encapsulated data but TAM also skips these packets.

A solution to IP fragmentation would be to adapt the libnids library[53**] used mainly for network IDS to TAM as the library supports, among others, the reassembly of IP fragments.

4.1.3 Duplicate linked list entries

In packet switched networks it is possible for packets to be lost and retransmitted. As a result it is possible a packet that creates a linked list entry (PDP context or RADIUS session) to be received more than once. The reasons may be that the initial packet was dropped at some point between the capture and the destination and the sender retransmitted it, or it was duplicated by the capturing mechanism (a rare but not impossible case). TAM checks the list entries in order to guarantee that every message that affects the state information maintained in the program does not create a duplicate list entry. That applies to the process of creating an entry as the deletion of an entry is safe since there is no entry to be affected. The process of updating an entry, applying to the GTP protocol, is also considered safe since the Update request messages just add some extra fields to the entry and the first Update response resets the update sequence number field so no more Update responses could affect the entry.

4.1.4 C language

The language along with the compiler caused some problems when custom C structures were used to facilitate fetching header information from the packets. For example a C structure representing the GTP header has been created to enable easy access to the GTP header fields as the TEID or the message type. However, the C compiler by default tries to optimize the structures by introducing empty padding bytes along the structure fields. That is because it is faster to perform a read of a 4 byte value than a 1 byte value and align it accordingly in a 32-bit machine. The 1 byte value may require correct alignment before being placed to a 1 byte register. This compiler behavior is acceptable when just storing values to a C structure but causes problems when used in an array of bytes coming from the network.
card because that line does not contain the padding introduced by the C compiler. A read of a structure value in the above case may cause to read the wrong value. That problem was detected at an early stage in the debugging phase. The solution was to instruct the compiler to avoid padding in the structures that are used to read values from the studied packet by using the ____(packed)____ struct parameter.

Moreover, the C function used to copy string values mainly for the IMSI, MSISDN and APN fields, strcpy, produced wrong results when it was provided with null pointers. Thus, prior to use of strcpy the function arguments are being validated.

### 4.1.5 TAM bugs

The original TAM program contained some bugs of its own:

A TAM function used to calculate the payload size of the packet could only return the correct value when running on IP version 4. TAM did not check for the IP version though. Some times the encapsulated GTP user packets use other protocols than IP version 4, like PPP, IP version 6 or others and the program was treating these protocols as IPv4 producing wrong results. As a result a new condition has been introduced to TAM, to analyze only the packets that contain the IP version 4 network protocol.

When TAM was running on a trace with truncated packets, e.g. Trace1, it strangely output duplicate flows. The reason was that the program did not consider the truncated packets and while evaluating them it accessed the memory of other structures and produced these duplicate flows. In order for the program to display the correct results, the startup option -s 200 had to be set. The option limits the size of captured/analyzed packets and 200 is the maximum number of bytes to be considered for each packet (assuming the program was running on Trace1). That is inconvenient for offline analysis as the user is required to first examine the trace file in order to detect if the packets were truncated and then use the correct options in TAM. The version produced on the thesis detects the truncated packet size automatically without the need of additional options in TAM startup via a parameter provided by the libpcap library.

TAM uses a separate program in order to convert the binary output to human readable string output. That tool is based on the program code but it does not check if the input file exists, causing errors.

Originally TAM did not support IP fragments assuming that no IP fragments exist and produced wrong results\(^\text{15}\) when it encountered them. Cur-

\(^{15}\) The results contained spurious flows where no actual data existed. The reason was the use of the payload data in order to get the port numbers in fragmented packets.
rently, TAM process only the first of IP fragments, the one containing the
transport layer header, and ignores the rest of them.

The program uses a function in order to keep the same application in the
two flows that concern a user connection (downlink and uplink). This func-
tion checks the application assigned to each flow and returns a value to be
assigned to both flows. However, the function does not check if both applic-
ations are same, as on this situation it can return immediately.

TAM contains a function to print the value of the IP protocol field as a
string value, like TCP, ICMP etc. However, that function produced no output
and needed to be changed to support this. Currently, it supports TCP, UDP
and ICMP showing Other for any other protocol. Furthermore, the function
was enhanced to output also the number of the IP protocol field when it
showed Other. The new output is for example Other-45. That is in order to
distinguish the flows that contain different protocols following IP in the ag-
gregation function of unidirectional to bidirectional flows that takes place
after TAM produces the flow results.

The application recognition code of TAM uses some very generic regular
expressions for some protocols, particularly Skype. As a result many flows
may be identified as Skype but that does not mean that they carry Skype.
Zhenfang Wei, a colleague working also on TAM in his thesis project [54**],
contributed to the application recognition function of the program by intro-
ducing the detection of the Distributed Hash Table (DHT) application. That
reduced the number of flows identified as Skype.

Another problem with the application recognition was that some applica-
tions were used to classify flows but they were not shown in the output, pro-
ducing a significant amount of flows that have the Other application. The
problem was the function used in order to convert the application code to a
string value, e.g. BitTorrent, that was not up to date to include all applica-
tions recognized by the program.

Finally, a small bug in the flow aggregation function caused spurious ap-
lications to be assigned when the flows had the same IP source and destina-
tion address and the same source and destination port. The reason was that
the program uses the same entry in the hash table in the forward and reverse
flow directions as the key to the hash table, the 5 tuple, is the same. It is not-
able to mention that identified flows are not, in general, affected by this as
their key to the hash table contains also the direction and ID of the flow.
4.2 Studying the Gn interface

The GTP specification defines many functions of the protocol, like charging or QoS that are not related to the study. Thus, parts of the specification have been left out. It is worth mentioning that GTP defines about 50 control messages and TAM needs to analyze only 8 of them in order to associate the IMSI and APN to user packets. The number of IE being defined easily reaches 100 but the relevant ones are closer to 40.

In Trace1 the reason that success rate is low is that some GTP Create Request messages were not evaluated by TAM since they did not contain all necessary IE because they were truncated to 200 bytes. In these messages the PCO IE is very long, like 40 to 100 bytes. Because this IE comes before the two GSN address IEs it is likely that the truncated packet would not contain the GSN addresses, that are mandatory IEs in the Create Request. Thus if TAM cannot read the GSN addresses it cannot build a correct PDP context and as a result discards the message. The GSN addresses are needed in order to define the tunnel endpoints for control and data traffic.

The Create PDP Context Request is the most important GTP message since it is the one that carries the IMSI and APN. However, there are two types of this message. The primary Create PDP Context Request and the secondary Create PDP Context Request\(^\text{16}\). The purpose of the secondary context request is to assign another context to the user (same IMSI) that shares the same APN and PDP address but it may be different in the TFT and QoS properties. The secondary context request is related to the primary PDP context and it does not carry the IMSI and APN fields. It carries a field called LNSAPI that is used to locate the primary PDP context. The secondary context request carries another NSAPI to be assigned to the secondary context. TAM implements the secondary PDP context concept. However, in all trace files analyzed in the study no secondary PDP context request message has been detected.

The Update PDP Context Request messages may stem from either the GGSN or the SGSN. The code executed in each case is slightly different since the update request message sourcing from an SGSN may carry new TEID assigned to the PDP context and a GGSN sourcing request may carry a new EUA to be assigned to the user. Both cases have been implemented in TAM. However, the analysis of the trace data showed that all update request messages being detected in the traces stem from the SGSN.

A solution to improve the success of locating the IMSI in the Gn interface was to utilize the SGSN Context messages. These messages carry the PDP context information between SGSN when the user roams from a location that

\(^{16}\) Their detection is via the TEID value of the Create PDP Context message. The primary create context request has a value of zero while the secondary create context request has a non zero TEID value that is the the control TEID of the related primary PDP context in the GGSN.
is controlled by one SGSN to another location controlled by another SGSN. The PDP context carried in these messages contains useful information such as the IMSI, the APN and the TEID for the uplink direction (towards the GGSN)\(^{17}\). This information may be used together with the update messages that always follow the SGSN context messages to create an entry in TAM linked list. The problem with the SGSN responses is that sometimes they do not carry PDP contexts\(^{18}\), they carry only the Mobility Management (MM) context of a user. For example in Trace2 there were no PDP contexts in the SGSN context messages.

The SGSN context messages do not need to be analyzed by TAM if the PDP context already exist in the program's memory. Thus, a check is performed using the IMSI and NSAPI of the PDP context. If the entry exists in the linked list, the SGSN context message is skipped because the Update messages that always follow a SGSN handover procedure are used to change the TEID of the PDP context.

Three SGSN context messages are defined in GTP specification. The *SGSN Context Response* sent by the old SGSN to the new SGSN and the *SGSN Context Acknowledge* sent by the new SGSN to the old SGSN in order to verify correct reception of the information. The SGSN context request is sent by the new SGSN to the old SGSN, but since it carries no useful information it is not processed by TAM.

In order to verify the TAM output on the Gn interface many techniques have been used. Some methods follow:

Based on how the GTP protocol operates, the TEID being used in the data packets should be present (defined) in at least one control packet. In the first stages of verification this rule was used to check if it is possible to find any control messages for the data packets that were not associated with an IMSI and APN. The rule applied to Trace1 and the results confirmed the output of TAM. However, some TEID data from “unidentified” packets were contained in truncated create requests that were not processed because they were missing mandatory IEs or some TEID data existed in update messages but the create messages were not present in the trace.

Another technique was to construct a biased trace file with a predictable output, analyze it with TAM and compare the results. A small 297 packet (1 user) trace file was used and TAM correctly detected 34 flows and associated all data flows with the IMSI and APN of the Create Request message.

Another method used was to see how many of the packets associated with IMSI by TAM actually appeared in the trace. The TEID of all packets associated with IMSI were checked against all the packets of the trace that had the same TEID. The results were as follows: 24020 packets were detected by TAM and 24359 packets were present on the trace. The small difference is

\(^{17}\) The SGSN response messages do not carry the MSISDN of the user.

\(^{18}\) The SGSN response messages usually carry one on several PDP contexts concerning only one user.
mainly due to the delete function. Here, the PDP contexts were deleted when TAM detected a *Delete PDP Context Request* GTP message for the PDP context. However, sometimes it is possible that data packets in the opposite direction still exist in the network. Some trace investigation revealed that zero to three packets may be present in the network after the delete request. Another reason is that packets may have the same TEID but different GTP type (GTP control messages) or same TEID but different IP destination address, targeting a different network element. Since the TEID are assigned uniquely per network element it is possible for two network elements to use the same TEID.

The status information produced by the status system implemented in TAM revealed interesting results:

Trace1:

25929 *Delete PDP Context Request* messages (total 90236) could not be associated to a PDP context in TAM (the linked list entry did not exist) showing that many contexts were active when the capture started. From 89897 *Delete PDP Context Response* messages only 64242 could be associated with linked list entries in TAM. The number of deleted entries is 64246, showing that some users may have more than one PDP contexts active.

The number of skipped *Create PDP Context Request* messages was 968. Most of them were because the GSN Address IEs were not found in the packets. In 56 cases the packets were skipped because the MSISDN IE could not be evaluated although existed in the packet.

There were 3554 *Create PDP Context Response* messages with a cause other than accepted. Almost half of them due to exhaustion of dynamic PDP addresses to be assigned to the users and another half due to failed user authentication. Very few, 47, failed because of no resources available. The total number of creates responses were 76545 and in 974 of them TAM could not detect an entry in the linked list (to TAM knowledge the PDP contexts did not exist).

All the *Update PDP Context Request* messages originated from a SGSN.

In 3224 *Update PDP Context Request* and another 3224 *Update PDP Context Response* messages, from a total of 3819 messages the entry could be found in TAM linked list. That is another indication of the number of already active PDP contexts.

A small number, around 300, *SGSN Context Response* and *Acknowledge PDP* messages were present in the trace but they were affected considerably by the limited captured packet size. Only 91 *SGSN Context Response* messages were verified. However, the PDP contexts related to these messages already existed in TAM causing the contribution of the messages to the association of IMSI to data packets in the trace as zero.

The above errors appeared to be “normal” considering the truncated *Create PDP Context Request* messages and the border effect.

The data had also some interesting errors:
21 data packets were too small to carry an IP header. They were PPP fragments. Furthermore, in 23988 packets the inner protocol was not IP version 4. Possible alternatives are PPP and IPv6. The amount of fragmented packets in the inner IP header is small, 10098 packets.

The IP fragments of the outer IP header were 3003481, a very large number. Moreover, 2170 packets carried other than IP version 4 protocol above Ethernet. An example is ARP.

The trace contained 15211 duplicated packets. Only 6 of the packets contained encapsulated user data. Thus most of duplicate packets are GTP control traffic.

The number of GTP version 0 packets on the trace was 13232. The control packets were 305. Considering the very small number of packets the success rate would not have increased much even if TAM supported GTPv0.

The rest of error statistics collected, like if the IMSI was present in the Create PDP Context Request messages, showed no error.

Trace2:

21139 Delete PDP Context Request (of total 65249) messages had no entry in the list. That shows a considerable amount of PDP contexts existed before capture starts.

The number of Create PDP Context Request messages was 65749. In one of them the entry already existed in the linked list, thus TAM maintained state information for 65748 PDP contexts.

In only 5 Create PDP Context Response messages the cause is not accepted and in 956 the entry is not found in the list. The large number of non-detected response messages is mainly due to the damaged trace files that exaggerate the border effect.

4 Update PDP Context Request messages had TEID value zero that is used to change GTP version.

In 21047 Update PDP Context Request and 21153 Update PDP Context Response messages the entry is not found in TAM. The total number of messages was about 24000. And in 17 Update PDP Context Response messages the cause was not accepted, undoing the changes of the update request.

All Update request messages originated from an SGSN.

On this trace no SGSN Context Response and acknowledge messages were detected. Another zero contribution of their implementation into TAM.

The data had also some interesting errors:

188 packets have been detected as too small (they are encapsulated and less than 20 bytes that is the standard size of the IP header). They are PPP fragments. Furthermore, in 2643 packets the inner protocol was not IP version 4. Possible alternatives are PPP and IPv6. The amount of fragmented packets in the inner IP header is also considered high, 293440 packets.

The IP fragments of the outer IP header were 174550.
No duplicate packets were detected in the trace. Perhaps a different capturing mechanism was used.

The trace contained 19558 GTPv0 packets. Only 622 control packets were detected but no packet contained the Create PDP Context Request message indicating that even if GTPv0 was implemented in TAM, the program would face difficulties constructing the PDP contexts.

The rest of error statistics collected, showed no error.

Trace3:
The trace contained 733860 GTP control packets, almost the amount of control packets being present in both previously analyzed Gn traces.

19933 Delete PDP Context Request messages, of total 130380, could not be related to a linked list entry.

The 126882 Create PDP Context Request messages of the trace created 126821 linked list entries in TAM. The rest 61 messages were pointing to an already existing entry.

Only 7 Create PDP Context Response messages had no entry in the linked list showing the very good capture of create messages. However, 2099 Create PDP Context Response messages had cause value other than accepted. More than half of these non accepted cause values were pointing to the exhaustion of all dynamic PDP addresses.

In 22 Create PDP Context Response messages the EUA IE was empty. The users perhaps used a static network address, some other way to acquire one, like via the Dynamic Host Configuration Protocol (DHCP), or some parameter of the PCO IE.

In 8638 Update PDP Context Request and 8641 Response messages the entry was not found in TAM. Total messages were close to 20000.

In 3 Update PDP Context Request messages the TEID is 0, used to change the GTP version. Only 3414 messages (89 control messages) contained GTPv0 and, considering the large number of GTPv1 packets on the trace, that shows the evolution of the GPRS core towards GTP version 1.

This is the only GTP trace that contained SGSN context messages that contributed new linked list entries to TAM. 4542 new entries were created based on information carried on these messages, making their implementation in TAM very important to increase the success rate of identifying data packets. More than 6000000 packets were identified by SGSN messages.

In 38611 SGSN Context Response messages the entry was already present in TAM. Some further study of the SGSN messages showed that some users experienced many handovers between SGSNs. For example, one user experienced more than 5 handovers between two specific SGSNs. The reason is perhaps oscillation introduced in the network or the user was located in the border of two routing areas and handover between base stations introduced handover up to the SGSNs.

\[19\] In the GTP specification the SGSN handover is called Inter SGSN Routeing Area Update.
The data had also some interesting errors:

10635 packets contained a network protocol different than IP version 4 (mostly IP version 6 and some PPP).

833 encapsulated packets were so small that they could not carry an IP header (they were PPP).

The trace contained 49068 duplicate packets that perhaps were caused by the capturing mechanism. However, none of the packets contained user data, limiting the duplicate packets to mostly GTP control traffic.

A very interesting error was that 38318 encapsulated packets had TCP header size less than 20 bytes. As the packets contained an invalid TCP header they were skipped by TAM. Further study showed that the packets were actually IP fragments that were cut in the middle of the TCP header. That is, fragment 1\(^{20}\) carried only a portion of the TCP header and fragments 2-n carried the rest of the header together with the user data. It is mentioned in the literature that these packets are more likely to bypass firewalls\([41]\).

The rest of error statistics collected, showed no error.

The large number of update messages that could not be associated with a PDP context in TAM, reaching almost 85 % in traces 1 and 2 and 42 % in trace 3, seemed to be wrong and the behavior was further investigated using Tshark, a terminal version of Wireshark. The tool was used to search for these 'missed' TEID of the update messages in the whole data. The plan was to acquire all the TEID of the Update messages that were not found in the TAM linked list and try to find other GTP control messages that carry these TEID numbers (mostly create messages). Since the method was very slow when executed on the whole data trace, a new trace file had been created that included only the GTP control messages of Trace2. The results showed that some of the TEID could be detected in other control packets. These control packets were most of the times Create PDP Context Response messages that had no related Request messages. As a result the logic used by TAM to process the GTP protocol seemed to be correct and the problem was that perhaps some packets were not captured. The large number of update messages in all Gn traces that have no relative entry in the program may also be explained by the fact that the PDP contexts that stay active for a long period of time are more likely to produce more update messages (due to handover, change of settings etc), where the create messages exist only in the activation of the PDP context and might not exist in the (captured) trace.

The SGSN response messages caused 379 flows in Trace3 to be identified only in one direction. That is probably due to the fact that some update messages were missing. An example is when TAM creates a PDP context after an SGSN Context Response message but because of an unknown reason the update messages for that context come after the data. That cannot be seen as

\(^{20}\)Fragment 1 is considered the IP packet that has the More Fragments flag as 1 and the Fragment Offset field zero.
a TAM mistake since TAM just used the available information. The control process in the network did not perform so well or simply the Update messages were not captured.

Different methods have been tried in order to test the TAM output and improve its performance. For example, in the beginning the IMSI was copied to the flow structure for every packet arriving for that flow. Comparing the TAM output when the above case was used to the case where the IMSI was copied only in the creation of a flow revealed some problems in the TAM flow structure and the PDP context concept. The reason that the two cases were tested was that the SGSN response messages may be detected at any time by TAM and thus there should exist some mechanism to update the IMSI in flows that already exist.

Particularly, 130 flows were different\(^{21}\) out of more than 4 million flows. The most probable reason that the flows were different was that it is possible for one user to talk to another user inside the GPRS network. The packets though still require a PDP context for each user and must travel all the way to the GGSN. The packets that travel from one user to the other have, of course, the same IP addresses in the uplink\(^{22}\) and downlink\(^{23}\) direction. Thus, TAM detects the same packet twice with two different TEID. One TEID is when the packet towards the GGSN and the other TEID when the packet passes the GGSN and travels toward the SGSN of the second user. As a result, the IMSI inserted in the flow in the two TAM versions mentioned above was different.

Another similar case encountered later was that one user happened to talk to itself via the GGSN. That is, the user packets had the same IP source and destination address and traveled via the same PDP context to the GGSN and back to the user. It is a very strange case and probably expensive for the user since the GPRS data are usually charged per volume of traffic. Two reasons are considered that caused this. One is that the user terminal/application used the wrong interface for internal communication and the other that statistics about the RTT time of the network were collected.

An alternative reason is perhaps that using only the IP addresses and port numbers to build flows of traffic is not enough when running behind a NAT, as is mostly the case with cellular operators\(^{25**}\). There are situations, mostly when ICMP or other transport protocols are used, where TAM added packets from 2 different PDP contexts to the same flow. The packets of the first PDP context had a difference in time in the order of 20 seconds and always less than the TAM timeout parameter (64 seconds). What happened was that the first PDP context ended and probably the EUA address was im-

\(^{21}\) The effect is considered very difficult to detect in other ways since the number of packets prohibits it.

\(^{22}\) Uplink traffic is when the user sends packets to the network. It is also referred to as Upload.

\(^{23}\) Downlink traffic is when the network sends packets to the user. It is also referred to as Download.
mediately given to another user. It is possible for that to happen when the operator runs out of IP addresses to assign to the user. For this purpose a GTP cause message is defined to indicate that no more dynamic PDP addresses exist and the PDP context is rejected. 1288 out of 2099 Create PDP Context Response messages with a cause value other than request accepted, contained this cause in Trace3.

The reason that the above undesired results appear is that TAM processes only the encapsulated data and does not know if the data belong to the same or have different PDP contexts that use the same IP and port numbers. As a result TAM always aggregates the data to the same flow if they have the same IP and port numbers. That happens usually when a PDP context stops and a new PDP one starts using the same EUA before the flow timeout expires in TAM. If these two users use the same application (the effect is mostly present when the ports are known, like the Network Time Protocol (NTP)) then their data are aggregated to the same flow. Another case is when a user talks directly to another user or itself in the network. On this case the PDP context changes (the same packet is detected two times with 2 different PDP contexts) and thus the flow appears in the comparison having two different IMSI numbers. Another case is that a user starts and stops PDP contexts regularly, using the same PDP address. That explains why only some part of the flow is identified assuming that not all PDP context creation messages were captured.

Two solutions are possible in order to solve the above situation. The first is to also keep the data TEID numbers on the flows in TAM. This approach requires many changes in the TAM code. The most important is that the flows need to aggregate both flow directions as it is impossible to locate the reverse direction of a flow if its TEID is not known. In addition, on this case only the packets that have the same TEID would be aggregated to the same flow and packets with the same PDP context but a different TEID, that changed for example via an update procedure, would constitute new flows. Another problem is that the solution could only be applied in the GTP case as there are no TEID numbers in the Gi interface.

The second solution is to use an internal number in TAM to identify the PDP contexts, as it is very briefly mentioned in [19**]. The approach is more robust as it guarantees that the packets aggregated to a flow actually belong to the flow and the PDP context. When the PDP context is deleted, the information about the PDP ID is removed from the linked list in TAM and no more packets could be added to the flow. Using the PDP ID in the TAM output makes it also easier to collect statistics on the PDP context level. The negative effect of the approach is that it does not identify the packets that come before the control messages which create the PDP context.

---

24 Another option is to flush the flows when the PDP context ends but it does not perform well since the flow identification is the IMSI and in order to flush the flows, all flows maintained by TAM should be compared to the IMSI value of the deleted PDP context.
Thus, it is possible for a flow to be divided to two or several smaller flows in
order some of these smaller flows to be identified. That is the reason why we
selected to delete the PDP context from TAM when the Delete PDP Context
Response message is detected. If the PDP context was deleted on the Delete
PDP Context Request message, the packets of a flow being transmitted
between the request and the response (usually zero to three packets) would
not be identified as the PDP context does not exist and they would create a
new, very small, and perhaps spurious flow.

The PDP ID drawbacks being mentioned, its advantages are very import-
ant. It guarantees that all the packets of a flow belong to only one PDP con-
text. As the ID is included in TAM output is very easy to detect the flows
that are identified (they have no zero ID). Furthermore, in the case that two
users of the same GPRS network communicate to each other, the packets are
associated to the correct PDP context for each user. Moreover, as the IMSI,
APN and MSISDN parameters of a PDP context could not be changed in the
duration of the context, it is sufficient to copy them to the flow structure only
once, when the flow is being created. Finally, it is very easy to define also
the direction of the flow as uplink or downlink.

Unfortunately, the PDP ID by itself could not solve the problem when one
user is communicating to itself, simply because the PDP context is the same
in both directions. Thus the uplink and downlink data would create only one
flow that counts the data twice. The solution is to use the direction of the
flow together with the PDP ID in order to find the flow of interest. The dir-
ection of a flow may have three values: Unknown(0) when the flow is not
identified with an IMSI, Uplink(2) when the flow traffic travels toward the
GGSN and Downlink(1) when the flow traffic travels toward the SGSN and
finally to the user. This approach affects only the case when a user talks to it-
self by splitting the uplink and downlink data to different flows and improv-
ing TAM correctness.

The low success rate of identifying data packets in the Gn interface drew
our attention. Possible causes were that perhaps some additional control
messages should be analyzed by TAM in order to increase the success rate.
An example is messages for multicast and broadcast tunnels, known as
MBMS. The GTP specification defines these messages. However, none of
the Gn traces contained at least one MBMS message. That is the reason why
these messages have not been introduced in TAM, since it would be im-
possible to test the program and perhaps already proven correct results could
be damaged by a faulty implementation of MBMS. The implementation of
the MBMS messages is suggested as future work.

The main reason for low success rate is mainly the border effect men-
tioned above. The outcome of the effect is more imminent if the duration of
the PDP contexts is considered. As is documented in [46**] it is possible for
PDP contexts to be kept open for a rather long time increasing the border ef-
fect substantially and degrading TAM results. In another case, it is men-
tioned in [36] that the authors were able to identify the 98% of the traffic when capturing the whole day (24 hours). Unfortunately, none of the traces examined in our study were that long in duration, not even close, so it is difficult to accurately say that if TAM analyzes a 24 hours trace it would identify 98% of the traffic. We do believe that is possible though.

A less important reason reducing the success rate is the existence of the previous version of the GTP protocol, GTPv0, in the traces. GTPv0 has been standardized in 1996 and updated in 1999 to GTP version 1. GTPv0 is still supported by the GTPv1 specification, not broadly used though. The traces pointed out about 14000 packets on Trace1, about 20000 packets on Trace2 and about 3500 packets on Trace3 having GTPv0. It is a very small number of packets to consider reimplementing the protocol. GTPv0 has similarities to GTPv1 but the way that tunnel endpoints are handled is different. For example in GTPv0 the IMSI is used to construct the tunnel endpoints.

The most interesting statistics presented here are been summarized in Table 4 below for easy reference.

Finally, it is interesting to notice some unexpected situation. Logically the assigned EUA for a user should be unique for the user as long as the user PDP context is active. However, in Trace3 the same EUA was used at the same time by two different PDP contexts, having different users and even different IMSI. However, the case was more like that one of the entries still existed in TAM because the delete messages were not captured, rather that actually two users were sharing the same EUA.

**Table 4: Important status information of Gn traces**

<table>
<thead>
<tr>
<th></th>
<th>Trace1</th>
<th>Trace2</th>
<th>Trace3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Delete Request without entry / Total</strong></td>
<td>25929 / 90236</td>
<td>21139 / 65249</td>
<td>19933 / 130380</td>
</tr>
<tr>
<td><strong>Delete Response without entry / Total</strong></td>
<td>25655 / 89897</td>
<td>21936 / 65132</td>
<td>19776 / 130027</td>
</tr>
<tr>
<td><strong>Create Request not processed / Total</strong></td>
<td>968 / 76618</td>
<td>1 / 65749</td>
<td>61 / 126882</td>
</tr>
<tr>
<td><strong>Create Response with non accepted cause</strong></td>
<td>3554 (50% because of no more dynamic PDP addresses exist)</td>
<td>5</td>
<td>2099 (1288 because of no more dynamic PDP addresses exist)</td>
</tr>
<tr>
<td><strong>Create Response without entry / Total</strong></td>
<td>974 / 76545</td>
<td>956 / 65700</td>
<td>7 / 126811</td>
</tr>
<tr>
<td><strong>Update Request without entry / Total</strong></td>
<td>3224 / 3819</td>
<td>21047 / 24727</td>
<td>8638 / 20428</td>
</tr>
</tbody>
</table>
### 4.3 Studying the Gi interface

On the Gi interface the GRE protocol is commonly used for network layer tunneling. GRE is located over IP and it encapsulates the user IP packet. That packet needs to be extracted from GRE and passed into TAM. The GRE protocol itself does not carry any information needed by TAM.

L2TP/PPP may also be used to encapsulate user packets. L2TP is defined in RFC 2661[34**]. It is transported above IP/UDP using UDP port 1701. The version 2 of the L2TP protocol has a header resembling GRE. Although, L2TP defines both control and user traffic, TAM processes only the user traffic. L2TP is always followed by PPP [47**][48**].

By examining the protocol field of PPP, the protocol that follows PPP could be detected. PPP may contain the address and code fields before the protocol field. Their values are 0xff and 0x03 respectively. Furthermore, the protocol field may be compressed occupying 1 byte or uncompressed 2 bytes long. Based in IANA standardization, the IP protocol following PPP has a value of 0x0021.

The only L2TP data example was found in Trace6. The trace contained also GRE version 1[32] used to carry only PPP. Thus the protocol was implemented in TAM. The order that the protocols are examined is L2TP/PPP is analyzed and the outer header is skipped if it exists. Then GRE is analyzed and the outer header is skipped too if it exists. That is based on the observation that is possible for L2TP to carry IP packets that contain the GRE pro-

<table>
<thead>
<tr>
<th></th>
<th>Update Response without entry / Total</th>
<th>PDP contexts created via SGSN Response</th>
<th>SGSN Response messages not processed / Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3224 / 3819</td>
<td>21153 / 24720</td>
<td>8641 / 20425</td>
</tr>
<tr>
<td>Packets identified via the SGSN Response</td>
<td>0</td>
<td>0</td>
<td>about 6000000</td>
</tr>
<tr>
<td>PDP contexts created via SGSN Response</td>
<td>0</td>
<td>0</td>
<td>4542</td>
</tr>
<tr>
<td>SGSN Response messages not processed / Total</td>
<td>299 / 299</td>
<td>0</td>
<td>38611 / 55854</td>
</tr>
<tr>
<td>IP fragments in encapsulated / outer IP header</td>
<td>10098 / 3003481</td>
<td>293440 / 174550</td>
<td>168299 / 275530</td>
</tr>
<tr>
<td>Skipped duplicate packets</td>
<td>15211</td>
<td>0</td>
<td>49068</td>
</tr>
<tr>
<td>GTPv0 packets</td>
<td>13232</td>
<td>19558</td>
<td>3414</td>
</tr>
<tr>
<td>Packets skipped because of malformed TCP header</td>
<td>0</td>
<td>0</td>
<td>38318</td>
</tr>
</tbody>
</table>
The status information produced by the status system implemented in TAM revealed interesting results:

Trace4:
The trace contained only RADIUS accounting and no RADIUS authentication packets. The packets involved 47338 accounting start and 46777 accounting stop messages. Most of the messages contained the optional 3GPP IMSI field.

The most important thing though, is that for 12293 messages no entry could be found in TAM. That is because of the border effect and the presence of damaged files in the trace.

All RADIUS messages and identified data packets were transported over GRE tunnels explaining the large number of IP fragments (297206).

However, there existed other data packets outside the GRE tunneling functionality. A test was performed in order to determine if these packets could be identified by the information carried in RADIUS messages. The results were that none of the packets could be identified. Their presence remains a mystery. If we exclude the non GRE data the success rate climbs to 30%. Considering the border effect, the damaged trace files and the short trace duration the success rate is satisfactory.

In only 34 Accounting Request messages TAM could not extract the user information. The reason was that these messages had accounting type other than start and stop (the two RADIUS accounting types analyzed by TAM).

Trace5:
The trace contained only RADIUS accounting messages and no tunneling protocols. However, the trace is dominated by the Wireless Transaction Protocol that is a layer of the Wireless Application Protocol (WAP). Although the success rate of the trace is high, WAP is not recognized by TAM.

Trace6:
The trace contained very few Accounting Request messages, 1108, transferred as normal IP packets. All of them contained the user IMSI. However, they were able to identify only 1295 data packets.

The main reason was that most of the packets were encapsulated on L2TP/PPP and/or GRE tunnels. None of these packets could be initially recognized by TAM. They consisted more than half of the total data packets on the trace.

Trace6 was the most important reason that L2TP was implemented in TAM as is the only trace that carries the protocol. However, testing the results of Trace6 after the L2TP implementation, showed zero improvement in the success rate. The RADIUS packets could only identify packets that were not carried via L2TP or GRE.
Finally, the trace was the only Gi trace that contained Diameter packets. The number of the packets was very low, only 20 packets, and carried no user identification information.

Trace7:
RADIUS access messages first appeared on this trace. They did not contain the user address (Framed-IP-Address) though.
Furthermore, from the 2212 Accounting Request messages, none could be processed correctly by TAM since they were truncated to 200 bytes and important RADIUS elements were missing, e.g. the MSISDN of the user.
The success rate of the trace was as a result zero.

Trace8:
The trace contained both RADIUS access and accounting messages. However, only the latter contained the user address.
The total number of Accounting Request start messages was 3849. The stop messages were similar in number but in 2434 of them the entry did not exist in TAM showing partially the amount of active RADIUS sessions
The accounting messages contained 3GPP attributes but not the IMSI.
As the trace contained no tunneling data the number of IP fragments is very low, less than 0.3 %.
The trace success rate is low mainly because of the very short trace duration.
A general comment about the Gi traces and the RADIUS messages is that based on the studied traces RADIUS 3GPP attributes, like the IMSI, are included only in accounting messages. That is another reason why the analysis of RADIUS access messages is not performed in TAM.

---

25 The partial number of active RADIUS sessions could be observed if we consider the sessions that started, stopped or started and stopped during the capture. For the rest of the sessions that have started sometime before the capture and ended sometime after the capture no direct method exist in order to quantify them.
4.4 Program output

Originally, TAM had the following flow output:
- Start time (seconds)
- Start time (milliseconds)
- Stop time (seconds)
- Stop time (milliseconds)
- Source IP address
- Source Port
- Destination IP address
- Destination Port
- Number of bytes transferred based on the IP packet length
- Number of packets transferred
- IP protocol
- Recognized application for the flow.

After our enhancements in TAM, its output has been appended with the following fields:
- Length of the payload of the packet
- Number of packets in the flow that carry a non zero payload
- Payload of the packet calculated using TCP sequence numbers that is valid only for TCP flows and in other cases zero
- Direction of the flow that is valid only in identified data flows and in other cases zero
- Flow ID that is either the PDP context ID or the Radius ID depending on the interface.

Moreover, in identified flows only, the following three fields are appended to the output:
- IMSI
- APN
- MSISDN

Since the new TAM output requires a new flow structure in TAM that is used only in the Gn and Gi interfaces, the output is displayed only when these interfaces are used.

In order to process the TAM output Perl or Shell scripts were considered. Perl dominated in performance and capabilities, thus it was selected. Gnuplot is used to produce the graphs demonstrated in paragraph 4.6.

Since it is required the output of the program to consist of one flow for both directions, the output of TAM is processed in order to aggregate the two directions to one. The Perl script used to aggregate the unidirectional TAM flows, utilizes the flow ID and direction of each flow in order to aggregate the flows consistently. However, in unidentified flows the direction and flow ID cannot be used. In this case, the flows still consist of bidirectional traffic but the direction of the traffic is not known.
The script first sorts the TAM output and operates similarly to TAM keeping the flows to a hash table and flushing the expired flows to a file. In order to detect the expired flows the timeout value used in TAM is required as a script parameter. The script output is also being sorted. Moreover, it is possible for the script to aggregate flows using a larger timeout value from the one used in TAM. That is a quick way to see how the output is affected by the timeout value. This ability of the script was checked by comparing a TAM output of timeout 64 seconds, manually adjusted to 120 seconds by the script to an TAM output of timeout 120 seconds. The results were similar, same number of flows, with some small differences:

- the script cannot examine the TCP sequence numbers so in TCP flows it is expected to miss retransmitted TCP traffic
- internally TAM uses a function in order to keep the application of the two unidirectional flows consistent, while the script does not consider this and as a result the application of some flows may be different^26

The script outputs the bidirectional flows in a consistent format, as follows:

(note that the script output is slightly different for unidentified flows)

- Start time (seconds)
- Start time (milliseconds)
- Stop time (seconds)
- Stop time (milliseconds)

in identified flows
- user IP address
- user Port
- server IP address
- server Port
- Downlink payload length calculated via the IP length
- Downlink number of packets
- Downlink payload length calculated via the length, of the payload of each packet
- Downlink number of packets that have non zero payload
- Downlink payload length calculated using TCP sequence numbers
- Uplink payload length calculated via the IP length
- Uplink number of packets
- Uplink payload length calculated via the length, of the payload of each packet
- Uplink number of packets that have non zero payload

^26 The application field may also be different when using the same timeout value as TAM. The reason is that a bidirectional flow may consist of more than one unidirectional flows (the flow expired only in one direction in TAM) and one of them may have a different application, e.g. because it was too sort.
- Uplink payload length calculated using TCP sequence numbers in unidentified flows
  - source IP address of packet that started the flow
  - source Port of packet that started the flow
  - destination IP address of packet that started the flow
  - destination Port of packet that started the flow
  - Src to Dest payload length calculated via the IP length
  - Src to Dest number of packets
  - Src to Dest payload length calculated via the length, of the payload of each packet
  - Src to Dest number of packets that have non zero payload
  - Src to Dest payload length calculated using TCP sequence numbers
  - Dest to Src payload length calculated via the IP length
  - Dest to Src number of packets
  - Dest to Src payload length calculated via the length, of the payload of each packet
  - Dest to Src number of packets that have non zero payload
  - Dest to Src payload length calculated using TCP sequence numbers
- IP protocol
- Recognized application for the flow
- Flow ID
- IMSI
- APN
- MSISDN
  (in unidentified flows the latter three fields are empty)

The flow ID is necessary when processing the output of TAM in order to find the traffic transferred per PDP context or RADIUS session.

### 4.5 Verification – Comparison of programs

Every software tool should be correct in order to be useful. The verification that the software works correctly is a challenging task. Moreover, in network environments where packet capture takes place the verification procedure faces problems as it is needed to predict all possible packet errors, including the state errors caused by damaged packets.

The verification phase was performed with the aid of other programs operating on the same field of Gn and Gi interface analysis by comparing their
results to TAM. The conclusions drawn from the comparison might help improve all programs.

In the study two other programs were used in order to verify TAM results. However, only the results of the programs on the same trace files were provided (Trace3 and 8) and not the code or additional information about the two programs. For easy reference the programs are named CFlow and MFlow below. Note that the tools are not publicly available.

Although the comparison is vital in order to find weaknesses in each program a thorough approach would consume large amounts of time and still without the code of the two tools, or at least their executable\(^\text{27}\), it is impossible to make concrete statements. As a result the comparison is not considered exhaustive. The statements below are heading towards the right direction but may also express invalid information.

The CFlow program output was compared to TAM output on Trace3. The MFlow program output was compared to TAM output for both Trace3 and Trace8.

Note that none of the programs seems to support GTP version 0. That was easy to detect by checking if the GTPv0 packets in Trace3 were shown in the output of the programs. As there was no match of an IP address none of TAM, CFlow and MFlow support GTPv0.

### 4.5.1 CFlow

CFlow was a experimental tool operating on the Gn interface that aggregated packets to flows, supported the reassembly of IP packets, analyzed the GTP protocol and associated the IMSI to the data flows. Control packets were not included in the program output. The program detected the user application via a payload evaluation approach.

Comparing the results of TAM to CFlow in Trace3 showed that the default timeout setting of 64 seconds in TAM was a main reason of difference. CFlow had a larger timeout value and, as a result, showed less flows than TAM, with a higher per flow duration. Later we were informed of CFlow timeout value, that is 2 minutes. The comparison results below are based on TAM timeout value of 120 seconds.

The program performed similarly to TAM but it had some fundamental differences:

- only counted the packet payload by using the IP length field.
- only counted the packet count by counting all packets seen.
- CFlow reassembled IP fragments while TAM skipped them.

\(^{27}\) It would be very helpful to compare how the programs react on a special case of some hundred packets, focusing in just one difference.
• supported only TCP and UDP (no ICMP or Other protocols following IP).
• skipped control traffic.
• did not show the APN and MSISDN fields.
• had many differences in flow application, mainly due to different application names (e.g. TAM: http, CFlow: web). It is not easy to judge which program was correct here. For more information see Table 5. However, probably CFlow uses port based application recognition together with payload inspection as a flow carrying no payload was classified as email by the program (the flow was on TCP port 993 that is assigned to IMAP). On this case TAM classified the flow as NonPayload.
• was sometimes different in the millisecond time fields for the start and stop times of a flow. Perhaps because the program counted also the time of IP fragments that are skipped by TAM.
• had difference on the payload bytes, probably because TAM skips fragments. In one flow though, CFlow showed 1 packet less than TAM. It is considered as a case of unsuccessful IP fragment reassembly due to missing fragments.
• was missing some flows that were present in TAM. Unknown why.
• it identified some flows that had traffic before the control packets could be detected (flows identified from SGSN response messages) so it perhaps maintained additional state information and completed many passes of the data.
• completely missed the identification of a flow. TAM split the flow into two flows to associate the latter flow with IMSI.
• it is not exactly clear how the program displayed the flow information, particularly the order of the IP addresses and ports and the traffic direction. TAM is strict on how it displays the IP numbers. For unidentified flows the IP that started the transmission of data is kept first, while for identified data the first IP is always the user IP. For traffic statistics, TAM shows the traffic based on which IP appears first for unidentified data and for identified data TAM always shows first the amount of downlink traffic. CFLOW is not clear on this and flows differ on this aspect. That is, CFLOW does not keep a constant presentation of IP addresses (sometimes user addresses appear first, while sometimes they appear second applying to both identified and not identified flows). As a result it is perhaps difficult to calculate the downlink and uplink traffic of the flow if the two directions are not known.
• keeps the data to one flow even if the flow contains multiple PDP contexts. TAM splits that flow to flows per PDP context.
• TAM has 10000 more identified flows. However, the above case could be the reason why. The difference on identified flows is, considering only data messages with TCP or UDP without the application and the two millisecond fields, only 40000 flows. The total number of identified flows is about 700000.

• TAM finds 1300 IMSI that are not output in CFlow and the latter finds 140 IMSI that are not output in TAM. In one occasion studied, the programs identify a flow with different IMSI. Examining the actual data pointed out that TAM was correct.

• Sometimes CFlow identifies a flow with 2 IMSI. That occurs when two users communicate to each other. TAM on this case splits the flow to two flows each one dedicated for the traffic of one user.

• There are occasions where the IMSI identification between the two programs is not the same. At least 4 cases were detected. Examination of the actual data showed that TAM had selected the correct IMSI in order to classify the flow. It is unknown why CFlow displayed another (wrong) IMSI on the flows.

• In one occasion that a user talks to itself (unknown why but it happened), CFlow shows 1 flow where TAM splits the flow in two flows.

• A positive difference for CFlow is that it detects the WAP protocol, something that is not done by TAM because WAP is found mostly on GPRS data and TAM targets application recognition for the Internet.

4.5.2 MFlow

The second program, MFlow, was similar to an official product and its output was very close to TAM. The main differences are being separated in a per trace fashion:

Trace3:

• The program did not output the APN in flows detected via the SGSN Context Response messages. That is the main feedback contribution of the study towards the authors of MFlow.

• The timeout value was 600 seconds.

• MFlow parsed the data more than once. That was easy to detect since in some cases the data were associated with an IMSI but in the trace the control messages carrying the IMSI followed the data in time. TAM could not identify these flows as it parses the data only once.

78
MFlow had 4 output files. One for PDP contexts only, and one for each of IP, TCP and UDP flows. Thus TAM output had to be adjusted accordingly.

- **PDP contexts**, differences:
  - MFlow considered the duration of a PDP context based on the time it detected the messages that create and delete the PDP contexts. While it is very easy in TAM to extract the PDP contexts' flow data (utilize the flow ID field), their start and stop time are the times when data were transmitted. As a result no PDP context matched due to time differences.
  - MFlow also displayed the PDP contexts that transferred no data.
  - different on APN in some flows (identified via SGSN Context Response messages) that MFlow output as NULL.
  - many differences in data being transferred (bytes and packets). MFlow or TAM may show more packets in the PDP contexts. The reason may be due to IP fragments, duplicate packets or other packets counted in the PDP contexts by MFlow that are missed in TAM or that MFlow stops the PDP contexts upon receiving a delete request message (TAM stops the PDP context when it detects the delete response message since there still may be packets on the network).
  - if only the PDP contexts with network traffic are considered, excluding the time, traffic and APN fields only 200 PDP contexts differ in each program from a total of about 80000 PDP contexts. A reason is that TAM sometimes keeps a wrong EUA (not actually wrong, it exists in data but it is not the one assigned to the user). The user on these cases uses more than one addresses, with usually the first address as local, e.g. 192.168.1.102, and without downlink traffic indicating the wrong address and after a while the user switches to another address. Since the script used to aggregate the flows to PDP contexts uses the first address it sees as user address, the outcome is that the PDP contexts differ on the PDP addresses. In the future, TAM could also output the address assigned to the user per PDP context as the address is being captured in GTP control messages but not displayed.
  - Checking the number of IMSI displayed by each program showed that MFlow had 3409 more unique IMSI than TAM where TAM had only 9 unique IMSI more. The difference is
attributed to MFlow output that includes also the PDP contexts without traffic.

- In one case a flow was identified by TAM via a SGSN response but the following Update messages were missing. The flow was not identified in MFlow.

- IP flows, differences:
  - MFlow does not show GTP control traffic.
  - In IP flows MFlow shows only the protocol that follows, e.g. TCP, UDP and not the application. Sometimes it shows NULL, probably for flows that are not TCP, UDP or ICMP.
  - It also shows some time differences in microseconds.
  - MFlow does not output the APN for some flows (those detected via SGSN response messages). It also mostly displays identified flows but sometimes shows the IMSI as NULL. And it sometimes identifies a flow before it detects the control packets. That indicates that the program performs at least two passes over the data externally or internally.
  - Sometimes either TAM or MFlow have more packets or bytes on a flow without clear indication why (e.g. time difference). If TAM has less packets that it may be probably to omitted fragments. If TAM has more packets then probably MFlow failed to reconstruct some fragments. The malformed TCP packets are the number one reason why TAM shows less packets transferred.
  - Sometimes also the flows are same but MFlow shows more seconds in the flow stop time (e.g. 2 more seconds). Unknown why.
  - Keeping only the time fields for seconds, the flows that have either TCP, UDP or ICMP without NULL IMSI and no APN at all, excluding also the traffic fields, shows about 8000 flows to differ in each program from a total of about 710000 flows.
  - The above differences may also be caused by TAM failing to identify flows that have control traffic following the data. MFlow can probably detect that.

- UDP flows, differences:
  - Only the identified flows from TAM output are considered since MFlow shows only identified flows.
  - MFlow outputs only the UDP payload in this case and not the number of packets transferred.
  - In this case MFlow displays the application it detects in the flows. However, it seems that it detects the application based on the port numbers. We were able to detect that by checking the applications against the ports. As a result there
were many differences in the applications because e.g. TAM missed to detect the application. (TAM misses some flows probably carrying DNS traffic).

- Similar to the IP case, MFlow shows the APN as NULL in some flows. Some of these flows have also the IMSI as NULL, but they are too few to be considered as unidentified traffic.
- MFlow mostly shows less bytes being transferred in the downlink traffic.
- Some time differences exist on the flows. E.g. MFlow shows a 5 second difference on the end of the flow. The rest of the fields are same.
- Excluding the NULL IMSI, NULL APN, application, bytes transferred and time fields shows that TAM has 1375 more flows than MFlow, where the latter program has 790 more flows that are different. The total number of flows for each program is around 207500.
- In one case MFlow failed to assign a packet to an identified flow because the packet contained a different TEID number. An update procedure took place between the transmission of previous packets and this unidentified packet shows that probably sometimes MFlow does not execute the update procedure correctly. In another case, MFlow loses all downlink traffic for the same reason.

TCP flows, differences:

- In the TCP case, MFlow output consists of 510850 flows, in 10501 of which the IMSI is NULL. TAM identifies 500341 TCP flows.
- The output of MFlow consists of the TCP payload, probably calculated via TCP sequence numbers and the number of packets carrying payload.
- Still there exist flows with NULL APN in MFlow.
- MFlow perhaps does check the payload of packets as some flows are identified as SSL by TAM where MFlow outputs NULL. The port number of the flows is 993 that is registered for encrypted email traffic. Checking the applications in the MFlow flows against the port numbers though, showed that most MFlow applications are classified via the study of ports. However, it exists one application, FTP_DATA that is classified in MFlow although using private port numbers both for the source and the destination. The reason is probably the analysis of FTP control information.
TAM on the same case shows many applications per port, e.g. exist at least 16 different applications in TCP port 80. Some of them may be valid but the payload inspection approach could also produce false positives (use the wrong application to classify a flow).

- MFlow have identified with IMSI more flows than TAM. Further investigation showed that, actually, in one case the context was built via an SGSN Response message. However, many more SGSN Response messages followed\textsuperscript{28}, all for the same PDP context. An update procedure (transmission of the Update Request and Update Response messages) was performed for every second SGSN Response message detected. That is not normal and perhaps indicated the omission of some packets from being captured.

At some point in time, two SGSN Response messages without an update message between them had different fields on the PDP Context IE. As the IE contains the settings for the GGSN, it is impossible an SGSN Response message to change these settings. In order for the settings to change, an update message need to be sent to the GGSN and based on the GGSN decision, the update response message may contain different values on these settings. That kind of update message was not captured.

TAM always relies to the information being carried in update messages as soon as it creates a PDP context via a Create Request or SGSN Response message and skips all subsequent create or SGSN messages if they refer to the same PDP context as duplicates. As a result, in the above situation, TAM would never update its entries for the PDP context since no update message existed / was captured. Missed control information for PDP contexts may cause TAM perform unexpectedly and it should be avoided.

MFlow on the above situation, detects the changes in the SGSN Response message and that explains the identification of more flows in its output. The same applies to CFlow. Perhaps, that is a bug in TAM but the other two programs do not seem to follow the GTP protocol.

However, it does exist a case too where MFlow fails to identify flows that are identified by TAM. MFlow detects the PDP context parameters from the SGSN response messages but only enforces them after the update messages are

\textsuperscript{28} The SGSN Response messages were transmitted in a oscillation like fashion between two specific SGSNs.
detected. In one case these update messages were missing so MFlow failed to identify the flow, where TAM based on the information collected from the SGSN Response message was able to identify it.

To alleviate the above situations it is highly recommended to capture all possible interfaces, at least the ones carrying control traffic.

- Sometimes TAM displays more traffic (both bytes and packets) than MFlow. There are also cases that MFlow shows no packets/bytes. In the former case the reasons may be some bug in the TAM TCP sequence number calculation code or the packets may be fragmented. In the latter case the reasons why this is happening are not known.

- Skipping the null IMSI and APN, application and traffic fields reduces the number of differences at about 3800 flows, from a total of 500000 TCP flows on each program.

- Concerning the value of MFlow timeout, the value seems to vary depending on the situation or something else affects the flows and makes them split to smaller parts. E.g. in one situation a flow was divided in to 2 flows with an estimated timeout value of 60 seconds (if timeout was the reason for the division). In another situation the timeout was about 500 seconds and in another there was no division and the largest 'silence' space between the data packets was about 270 seconds.

- Although MFlow detects the TCP payload via TCP sequence numbers, it probably fails to do so when exist more than one TCP sessions on one flow. On this case MFlow detects only the payload of one TCP session. TAM is able to display the correct payload.

  - IP – TCP differences

- Sometimes MFlow showed payload packets and bytes in flows when checking the IP output but in the TCP output considering the same flow it showed zero traffic in both uplink and downlink. A possible explanation is that perhaps the packets did not carry any payload. However, checking TAM output and the actual packets showed otherwise.

Trace8:
  - The timeout value was 600 seconds.
  - MFlow showed only identified data traffic in the output (flows that the MSISDN is known).
• The program had 4 output files similarly to the Gn interface. One for PDP contexts only, and one for each of IP, TCP and UDP flows. Thus TAM output had to be adjusted accordingly.

• MFlow showed NULL IMSI in the flows. That is acceptable since no RADIUS message carried the IMSI of the user. TAM also showed no IMSI.
  
  ° RADIUS sessions, differences:
    ▪ MFlow outputs the RADIUS session duration, based on the time that RADIUS access or accounting messages were captured. It also shows the APN and MSISDN of the flow. However, it shows no traffic fields on the flows and considering that TAM does not study the time that control packets were captured makes the results incomparable. That is because MFlow shows all RADIUS sessions detected and not only the ones that have actual traffic as TAM does.
    ▪ That is the reason why TAM shows only 3510 RADIUS sessions and MFlow shows 6290.
  
  ° IP, differences:
    ▪ MFlow prints only the identified flows, 24597 in total. Same number as in TAM.
    ▪ Very few differences shown, mostly because TAM outputs Other for any traffic other than TCP, UDP, ICMP where MFlow outputs NULL.
    ▪ There exist some time differences, mostly in the end time of each flow where MFlow shows additional seconds. The reason why that happened is not known.
    ▪ MFlow shows the APN as lower case letters. TAM uses the exact value extracted from the RADIUS control messages.
    ▪ The program shows more packets in some flows due to the TAM function that skips duplicate packets. In one test case the packets were indeed duplicate.
    ▪ Skipping the time fields and converting to lowercase the APN field in TAM output showed that only 3 flows were different, and that was because of the omission of IP fragments by TAM. The size of the fragments is calculated by MFlow and included in the flow traffic.

Based on the IP case we were able to detect that MFlow does not skip duplicate IP packets. In order to assist the comparison by reducing differences, the UDP and TCP case are compared against a TAM output that includes the duplicate packets. Perhaps, someone would consider that the same should also apply to the comparison in Trace3. However, in that trace the duplicate packets were only control traffic and not encapsulated data traffic so the inclusion of the duplicate packets in the study would not change the results as long as the control information is not considered.
- **UDP, differences:**
  - The APN is converted to lowercase from the beginning as we know that it is different.
  - Here MFlow shows only the payload of the UDP packets and not the number of packets per flow.
  - The most common difference is the application of the flows, similarly to the Gn case.
  - There still exist some time differences, mostly in the end time where MFlow shows additional seconds.
  - Excluding the time and application fields, there are no flows different in the two programs. Note that the 3 flows that were different in the IP case were all UDP flows. However, in the IP case the payload of the packets was calculated via the IP length field in the IP header and here in the UDP case the payload of the packets is calculated via the UDP length field in the UDP header. As a result, TAM can also calculate the payload of the fragmented packets and that is the reason why there are no flows different in the UDP case.

- **TCP, differences:**
  - Once again, lowercase APN is used.
  - On the TCP case MFlow shows the payload of the packets calculated using the TCP sequence numbers and the number of packets containing payload.
  - As in the UDP case, the application is the most important difference.
  - There are 524 flows (from a total of 13552) that differ in the number of packets and bytes per flow. Here that can not be attributed to IP fragmentation as based on the comparison of the IP case, only 3 flows seem to be affected by IP fragmentation and all of them are UDP. In one case the flow is classified as POP3 in MFlow and the traffic difference in the downlink is about 12 times more traffic being displayed in TAM. The number of packets are also about 3 times more in TAM. Perhaps MFlow calculates the payload according to the application layer protocol, in this case POP3, and that is the reason why the difference is so significant. However, further investigation showed that this could not be the case as MFlow also outputs different payload values on flows that are classified as NULL. One case may be that TAM counts also the TCP RST packets if they carry a status code. That though would cause only very small changes in the payload values, and in no occasion a 12 times larger value.
In cases where only the payload differs and the number of packets is the same, it is possible that the heuristics used in TAM in order to calculate the TCP payload produce wrong values.

However, considering a similar case in Trace3, TCP in MFlow was not calculating cases where many TCP sessions are included in the flow so perhaps MFlow also calculates wrong values.

- Skipping the application and traffic fields (both for payload bytes and payload packets) reduces the differences to only 7. All these 7 flows belong to different users, have the same application (RTSP) and differ on the end time. The end time that they all display is very close to the capture end time. The difference is up to 100 milliseconds.

An explanation is the study of application layer protocols by MFlow and the use of the duration of the session on the application layer as the flow duration. TAM does not continue the analysis to protocols above transport layer.

4.5.3 CFlow vs MFlow

It would also be interesting to see how the two programs compare to each other. This could only be seen in Trace3 and by using only the IP case in MFlow as that is the output being used by CFlow. Since on the MFlow IP case transport layer information is not included in the output, the comparison does not consider the port numbers of the flows on CFlow. Furthermore, the MFlow IP case also does not show the application of the flow. Thus also this information of CFlow is excluded from the comparison below.

The comparison is not as thorough as comparing the tools to TAM for two reasons. First, it is interesting to know the differences of the tools but it is not the goal of the thesis to find where these two tools differ and second, each program follows its own output format and without access to the code of the programs it is impossible to fine tune them for easy comparison as was the case in TAM where in order to compare the UDP and TCP cases with MFlow new fields had to be included in the output.

The differences are as follows:

- As MFlow contains only identified flows, CFlow output was filtered to also include only the identified flows. CFlow had 725486 flows in total where MFlow had 732364 (712737 with not NULL IMSI).
- As pointed out when CFlow was compared to TAM, the tool does not have a strict way of presenting the IP addresses and the traffic direction. E.g. while MFlow always shows first the IP address of the user, in CFlow the first address seems to depend on the source ad-
dress of the first packet on the flow. That makes the comparison really difficult as that affects the flow direction that affects almost all fields in the output.

- About 85000 flows seem to differ though.
- CFlow has 31851 IMSI and MFlow has 32951. However, they differ in about 3000 IMSI that are included in one program and not the other. That is because both programs have flows that are identified in only one program.
- The timeout value used in the two programs was different, 120 and 600 seconds respectively. And about 20000 flows differ only in the time fields, probably affected by the timeout. The timeout value also affects the number of detected flows.
- CFlow uses 1 flow when one user talks to itself where MFlow outputs two flows, similarly to TAM.

4.5.4 Differences on the application

The tables below show the difference in applications for the tools based on the number of flows per application. As the number of flows is directly relevant with the timeout value used for each program, we cannot compare directly CFlow with MFlow. The results stem from Trace3.

For example, in Table 5 we see that TAM classifies 2037 flows as BitTorrent where CFlow classifies 12939 flows. TAM, however, classifies 28248 flows as DHT that is not shown in CFlow.

Table 5: Number of flows recognized per application on TAM and CFlow.

<table>
<thead>
<tr>
<th>TAM with timeout 120 seconds</th>
<th>CFlow with timeout 120 seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIM</td>
<td>chat</td>
</tr>
<tr>
<td>Ares</td>
<td>email</td>
</tr>
<tr>
<td>BitTorrent</td>
<td>filexfer</td>
</tr>
<tr>
<td>Bootstrap</td>
<td>gaming</td>
</tr>
<tr>
<td>DHT</td>
<td>login</td>
</tr>
<tr>
<td>DNS</td>
<td>login.SSH</td>
</tr>
<tr>
<td>EarthS5</td>
<td>na</td>
</tr>
<tr>
<td>eDonkey</td>
<td>p2p.Ares</td>
</tr>
<tr>
<td>FTP</td>
<td>p2p.BitTorrent</td>
</tr>
<tr>
<td>GNU</td>
<td>p2p.DIRECTConnect</td>
</tr>
<tr>
<td>HTTP</td>
<td>p2p.eDonkey</td>
</tr>
<tr>
<td>HTTPAudio</td>
<td>p2p.Gnutella</td>
</tr>
<tr>
<td>HTTPQuicktime</td>
<td>query</td>
</tr>
<tr>
<td>HTTPVideo</td>
<td>streaming</td>
</tr>
<tr>
<td>IRC</td>
<td>streaming.HTTP</td>
</tr>
<tr>
<td>Kazaa</td>
<td>system</td>
</tr>
<tr>
<td>MSNMessenger</td>
<td>system.DNS</td>
</tr>
<tr>
<td>53</td>
<td>2140</td>
</tr>
<tr>
<td>52</td>
<td>87642</td>
</tr>
<tr>
<td>2037</td>
<td>331</td>
</tr>
<tr>
<td>959</td>
<td>1672</td>
</tr>
<tr>
<td>28248</td>
<td>4821</td>
</tr>
<tr>
<td>335751</td>
<td>14</td>
</tr>
<tr>
<td>19</td>
<td>768964</td>
</tr>
<tr>
<td>113</td>
<td>119</td>
</tr>
<tr>
<td>952</td>
<td>12939</td>
</tr>
<tr>
<td>2857</td>
<td>77</td>
</tr>
<tr>
<td>558601</td>
<td>99</td>
</tr>
<tr>
<td>625</td>
<td>1535</td>
</tr>
<tr>
<td>159</td>
<td>146</td>
</tr>
<tr>
<td>16596</td>
<td>1120</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5028</td>
</tr>
<tr>
<td>1092</td>
<td>377665</td>
</tr>
<tr>
<td>TAM with timeout 120 seconds</td>
<td>CFLOW with timeout 120 seconds</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>NBDS</td>
<td>tunnel</td>
</tr>
<tr>
<td>NBNS</td>
<td>voip</td>
</tr>
<tr>
<td>NetBIos</td>
<td>wap</td>
</tr>
<tr>
<td>NonPayload</td>
<td>web.HTTP</td>
</tr>
<tr>
<td>OtherChat</td>
<td>web.HTTP_P</td>
</tr>
<tr>
<td>OtherDNS</td>
<td>web.OperaMini</td>
</tr>
<tr>
<td>OtherHTTP</td>
<td></td>
</tr>
<tr>
<td>OtherMail</td>
<td></td>
</tr>
<tr>
<td>OtherVideo</td>
<td></td>
</tr>
<tr>
<td>POP3</td>
<td></td>
</tr>
<tr>
<td>SkypeOut</td>
<td></td>
</tr>
<tr>
<td>SkypeToSkype</td>
<td></td>
</tr>
<tr>
<td>SMTP</td>
<td></td>
</tr>
<tr>
<td>Soulseek</td>
<td></td>
</tr>
<tr>
<td>SSH</td>
<td></td>
</tr>
<tr>
<td>SSL</td>
<td></td>
</tr>
<tr>
<td>TCP</td>
<td></td>
</tr>
<tr>
<td>UDP</td>
<td></td>
</tr>
<tr>
<td>VALIDCERTSSL</td>
<td></td>
</tr>
<tr>
<td>WinMX</td>
<td></td>
</tr>
<tr>
<td>YahooMessenger</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>1986026</td>
</tr>
</tbody>
</table>

Table 6: Number of flows recognized per application on TAM and MFlow.

<table>
<thead>
<tr>
<th>TAM with only identified flows and timeout 600 seconds</th>
<th>MFlow with not NULL IMSI and timeout 600 seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIM</td>
<td>12</td>
</tr>
<tr>
<td>Ares</td>
<td>16</td>
</tr>
<tr>
<td>BitTorrent</td>
<td>172</td>
</tr>
<tr>
<td>Bootstrap</td>
<td>169</td>
</tr>
<tr>
<td>DHT</td>
<td>898</td>
</tr>
<tr>
<td>DNS</td>
<td>150050</td>
</tr>
<tr>
<td>EarthS5</td>
<td>11</td>
</tr>
<tr>
<td>eDonkey</td>
<td>55</td>
</tr>
<tr>
<td>FTP</td>
<td>375</td>
</tr>
<tr>
<td>GNU</td>
<td>244</td>
</tr>
<tr>
<td>HTTP</td>
<td>319712</td>
</tr>
<tr>
<td>HTTPAudio</td>
<td>391</td>
</tr>
<tr>
<td>HTTPQuicktime</td>
<td>118</td>
</tr>
<tr>
<td>HTTPVideo</td>
<td>8298</td>
</tr>
<tr>
<td>Kazaa</td>
<td>1</td>
</tr>
<tr>
<td>MSNMessenger</td>
<td>516</td>
</tr>
<tr>
<td>NBNS</td>
<td>662</td>
</tr>
<tr>
<td>NetBIos</td>
<td>1</td>
</tr>
<tr>
<td>NonPayload</td>
<td>97146</td>
</tr>
<tr>
<td>OtherChat</td>
<td>120</td>
</tr>
<tr>
<td>OtherDNS</td>
<td>11839</td>
</tr>
<tr>
<td>OtherHTTP</td>
<td>145</td>
</tr>
<tr>
<td>OtherMail</td>
<td>6469</td>
</tr>
<tr>
<td>Total</td>
<td>161876</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

88
The conclusions drawn from the above tables point out that a comparison on the application layer is not easy. Furthermore, TAM always shows more applications than the other programs. MFlow shows very few applications and perhaps its weakness on the application classification procedure. CFlow seems to divide applications in categories and without the actual application recognized, it may be difficult to interpret the results. E.g. it is unclear what the entry named 'system' represents.

As TAM analyzes only the payload of the packets it is expected to include false positives in the output. For example, many flows are classified as Skype but the regular expression used in order to detect the protocol is too generic and may also become true to flows that do not transfer Skype traffic.

It is clear that each program uses its own traffic recognition method and in order to correctly judge which program performs better, analysis of previously classified traffic by all programs and comparison of their results is necessary. If no correct judgment is to be made then it would be better to have all traffic recognition methods in one program and show three fields for the application in the output in order to increase flexibility.

### 4.5.5 Concluding the verification

Final conclusion about the choice of the best tool cannot be drawn from the simple comparison performed. Furthermore, TAM improved during the comparison in order to assist the comparison procedure and eliminate faulty code. Also additional information about the other tools is not known. For example the other tools may have more capabilities that are unknown to us. As a result, the conclusion below, performed to the best of our knowledge, should be interpreted with all these issues in mind and it is also possible to

29 The previously classified traffic could consist of known traffic of many applications, captured separately and merged to one trace.
be misleading. The table below serves as a reference point of the differences and properties of the three tools.

Table 7: Comparison of TAM - CFLOW - MFLOW

<table>
<thead>
<tr>
<th>Issues</th>
<th>Program</th>
<th>TAM</th>
<th>CFLOW</th>
<th>MFLOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv6 support</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GTPv0 support</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IP fragment reassembly</td>
<td>-</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Output of control traffic as flows</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
| User Identification Fields  | IMEI - APN - MS- ISDN | IMSI      |          | G: IMSI - APN<sup>30</sup>  
  Gi: MSISDN - APN          |
| Timeout value (seconds)     | 64               | 120       | 600      |          |
| IP protocols shown          | TCP - UDP - ICMP - Other | TCP - UDP | TCP - UDP - ICMP – NULL(other) |
| Skip duplicate packets      | Y                | ?         | -        |          |
| Calculation of packet payload | IP payload - UDP payload - use of TCP sequence numbers for TCP payload | IP payload | IP payload - UDP payload - use of TCP sequence numbers for TCP payload |
| Calculation of number of packets in a flow | All packets - packets that carry payload | All packets | All packets - packets that carry payload |
| GTP Protocol evaluation     | One parse of data Strict rules, data must come after control messages | Sometimes wrong IMSI Many parses of data | Sometimes loses or gets identified flows Many parses of data |
| Application recognition     | Payload Inspection | Payload Inspection and port based | Port based and application layer protocol evaluation |
| Additional Information      | Requires post processing for construction of bidirectional flows | Support for WAP protocol | Many output files, containing only identified flows  
  Support for WAP protocol |

All programs had their advantages and disadvantages. The future evolution of TAM might give it more advantages against the other tools and make it the number one choice. Main steps are the support for IP fragment reas-

<sup>30</sup>The APN is not displayed in PDP information created via SGSN Context Response messages.
sembly, verified application recognition and optimization for performance. IP version 6 support is another big step ahead for TAM.

Using the current version of TAM in the race of tools would place it between MFlow and CFlow. CFlow goes last as it lacks many features like inclusion of the APN in the output, functionality in the Gi interface\(^{31}\), flow direction and better handling of GTP. MFlow is the winner only if identified flows were provided to us. If the program omits the unidentified flows, it simply loses its position to TAM. If TAM supported fragment reassembly then the two programs would perform similarly, and the winning one would be the one that handles better the GTP protocol.

4.6 Results

The results presented below are based on Trace3 in the Gn interface and Trace8 in the Gi interface. While other traces could also be used here, the focus is on high quality traces. The graphs shown below share the same structure in both interfaces and that might lead to directly compare the two cases. That is not correct though, as the duration and of the traces is different.

Based on the signed disclosure agreement, information about the operators should not be disclosed in any of the traces. The APN value could provide clues about the operator and as a result all APN values have been converted to simple values like Apn1, Apn2 etc. In any graphs that Apn1 appears though, it is the same APN, e.g. Blackberry.net.

4.6.1 Gn interface

The output of the flow ID is very helpful when acquiring per PDP context statistics. Particularly, only 81464 PDP contexts carry data information, a considerably lower number from the 131363 PDP contents maintained in TAM. An explanation is that either the users use some protocol not supported in TAM, e.g. PPP or they open the PDP context for incoming downlink traffic, when it arrives. The border effect in the end of the trace is another reason why so many PDP contexts do not carry user traffic.

The number of users that transfer data are 33011. Other users could also start PDP contexts without transferring data as is shown above. However, since these users do not contribute traffic to the network, they cannot be observed in TAM output since the program outputs only traffic flows.

\(^{31}\) To the best of our knowledge the program does not support the Gi interface.
The number of APN used is very large, 119 APN. Since statistics per APN would be very difficult to comprehend as very high resolution is required and most interest is on the APN with the more users or traffic, only the 15 APN that have the largest per case measurements are presented while the rest are grouped into the Other measurement. An extra field, NO-APN, is used for non identified traffic, in order to be able to compare the effect of that traffic to the rest of the results.

The PDP context duration and the user duration shown do not concern the duration of the PDP context but the duration that traffic was communicated to the user. In the future more statistics about control traffic could be extracted from TAM, perhaps to an additional output file. This information would help to determine the number of all users that created PDP contexts, the duration of these contexts and other session related information. The changes to the program are small, only some additional fields in the linked list, like the context start and stop time, and a new function to print interesting fields of a linked list entry upon entry deletion.

The scripts used to produce the graphs are also part of the project and operate in Perl where the graphs are produced using the Gnuplot program.

A very interesting graph is the traffic as expressed for downlink and uplink for each APN. It is shown in Figure 7.

![Figure 7: Downlink and uplink traffic per APN.](image-url)
Note that most identified traffic\textsuperscript{32} concerns the downlink. That is expected as most users use small terminals that do not have the required resources to run servers or other programs that transfer large amounts of data in the uplink direction. Figure 8 shows the number of packets being transferred in each direction. It reveals that although most traffic is transferred as downlink, the number of packets sent in each direction is roughly the same, indicating that the uplink packets are usually of small length, carrying e.g. only commands like HTTP GET or TCP acknowledges since as shown in Figure 9, TCP is the most commonly used transport protocol.

\textsuperscript{32}The unidentified traffic concerns about 16.7GB. Since it is not possible to determine the direction, APN or users of the traffic, it is not presented here.
The amount of data transferred per application, classified by the TAM payload inspection function, is shown in Figure 10. The figure concerns all 29 GB of the trace.

Figure 9: Amount of data transferred per transport layer protocol on identified flows.

Figure 10: Amount of data transferred per application.
Most of the traffic concerns HTTP indicating that users are mostly interested about web surfing. The amount of traffic transferred for not classified applications is also significant because it is spread to two columns TCP and UDP in TAM. It is suspected that either users use other applications that TAM is unable to classify or the measurements represent false positives produced by the recognition algorithm. The SkypeOut application uses a very generic regular expression that operates on UDP and may also classify other protocols. There are many UDP flows concerning the ports 19760 and 19761 that cannot be recognized and they involve more than 1 GB of data. The existence of a significant number of GTP-U data is mostly due to fragmented traffic on the inner IP header that causes the omission of the inner packet since the outer IP header is still passed to TAM.

![Graph showing data transferred per application in APN NO-APN](image)

Figure 11: Data transferred per application on unidentified flows.

Focusing on the amount of data transferred per application for each APN separately would produce 120 graphs, including an additional APN for the unidentified traffic. However, the significance of the information would be low as the real name of the APN is not mentioned here. So only the two APN that carry the most traffic are shown below, together with the application traffic of the unidentified flows.

The graph of unidentified flows shown in Figure 11 resembles a scaled version of all traffic shown in Figure 10 while Figure 12 shows a high preference on TCP, mostly web, traffic for Apn1 and Figure 13 shows also that
most traffic is HTTP on Apn2. However, Apn2 contains a broader range of applications like eDonkey, a file sharing application.

Figure 12: Data transferred per application in Apn1.
Apart from the results concerning the users traffic analysis, other results concerning the network are also important. Figure 14 shows the number of users connected to an APN. The total number of users shown below is 33502, that is more than the 33011 users that exist on the trace since some users may connect to more than one APN either simultaneously or at different points in time.

Figure 13: Data transferred per application in Apn2.
While Apn2 comes second in the amount of traffic transferred as is shown in Figure 7 above, it comes sixth in the number of users. It is probable that the users of Apn2 are more active concerning transferring data than users of the other APNs.

The duration that users spend per APN, calculated based on the duration of their PDP context, is shown in Figure 15 where the mean duration and standard deviation per APN is displayed. Most users are active for less than 15 minutes. However, in APN Apn22 the mean duration value of the 48 users that are present in the APN is about 1300 seconds. The traffic of the APN is about 1.5 MB which indicates that mostly its users are sending very few data that could be keep alive messages, for example. APN Apn21 shows a very low variance as most of its 52 users have PDP context duration of about 2 seconds.

Figure 14: The number of users per APN.

![Number of users per APN](image-url)
Note that the PDP context duration value is not the time between the transmission of the Create PDP Context Request to the transmission of the Delete PDP Context Response since these numbers are not shown in TAM output. The PDP context duration is the time passed from the transmission of the first packet in a PDP context to the transmission of the last packet. As a result it is expected the actual duration of a PDP context to be larger that the values shown here. Furthermore, the border effect is expected to reduce the end time of many PDP contexts considering that the trace duration is less than an hour. Figure 16 below shows the percentage of users as a function of the PDP context duration. About 50% of the users have PDP context duration of less than 5 minutes while 10% of the users have PDP context duration of more than 25 minutes. That shows the high variation in the time that users spend on the network that is related to different user needs but can also be affected by other reasons, like charging policy.

Figure 15: Duration of PDP contexts per APN. The mean value and standard deviation per APN are displayed here.
Figure 16: Cumulative distribution function of the percentage of users per PDP context duration.
Other interesting measurements are the amount of data each user transfers in the whole duration of the trace. The relative graph is shown in Figure 17. More than 93 % of the users transfer less than 1 MB of data in the duration of the trace where 13 users transfer more than 60 MB of data. One particular user transferred 185 MB. The users that transfer large amounts data are expected to use powerful devices, e.g. laptops, in order to access the network where devices with limited resources like mobile phones are mainly used for web surfing, transferring very small quantities of data.

The number of PDP contexts a user starts during the trace, as shown in Figure 18, is also an interesting measurement as every new PDP context requires processing time in the GPRS network. It is interesting to notice that one user started 69 PDP contexts in the duration of 1 hour, while most users are satisfied using only one PDP context. The x axis in the figure below uses logarithmic scale.

Figure 17: Cumulative distribution function of the percentage of users per data traffic volume.
Additional statistics could also be collected from the program output. For example, how many control packets are transmitted per minute, more fine-graded results about users, the servers that the user contacted and other information. However, the results are affected directly by the small trace duration and the low success rate on identifying user flows. A longer trace and additional information about the network is required in order to draw conclusions about the users. Moreover, it could help detect bottlenecks on the network similarly to the study presented in [16**]. The authors used a trace with duration of 22 days, a high difference considering that the results presented here come from a trace with duration 1 hour.

4.6.2 Gi interface

On the Gi interface Trace8 is used in order to collect results. Since the duration of the trace is very small, about 2 minutes, and the volume of traffic consists of 3 GB, the results shown below may carry no significant information in order to draw conclusions.

The number of APN in the trace was only 15. As both Trace3 and Trace8 considered the same network, the APN names were kept the same in the res-
ults of the two traces, meaning that Apn1 in Trace3 results is the same APN to Apn1 in Trace8 results.

![Data transferred per APN](image)

Figure 19: Downlink and uplink traffic per APN.

The volume of data transfer per APN, as is illustrated in Figure 19, is considered small mainly due to the small duration of the trace. Similarly to the Gn interface, the amount of data transferred in the downlink is considerably higher than the uplink. However, the amount of packets transferred in each direction, as shown in Figure 20, is roughly the same. Since most traffic consists of TCP, based on Figure 21 below, it is safe to estimate that the uplink traffic consists mostly of TCP acknowledge packets.
Figure 20: Number of packets sent as downlink and uplink per APN.

Figure 21: Amount of data transferred per transport layer protocol.
Figure 22 shows the amount of traffic per application, for the whole 3GB trace. Based on the graph, HTTP is the number one application used by the GPRS users.

![Data transferred per Application](image)

Figure 22: Amount of data transferred per application.

The number of users detected per APN, as is illustrated in Figure 23, is low but that is a direct effect of the very low success rate on identifying flows, that is about 10%.
Figure 24 shows how the duration of the RADIUS sessions for all APN. The high variance in the first 9 APN is caused by the large amount of users on these APN and the effect of the small trace duration that limits the RADIUS session duration for all users. The last 6 APN have very few users and that is the reason why the variance is low.

Figure 23: The number of users per APN.
The RADIUS session duration per user is shown in Figure 25. Note that the duration reaches its maximum at 120 seconds which is the duration of the trace.
The volume of data transferred per user is shown in Figure 26. About 58% of the users have transferred less than 10 KB of data.
The number of RADIUS sessions per user, as is shown in Figure 27 shows an interesting point. One user opened 8 RADIUS sessions in 2 minutes. Further investigation showed that the sessions were not parallel but the user was opening and closing RADIUS sessions every few seconds. It is not known why the user opened so many sessions in so short notice. On each session the user was assigned a different IP address.

![CDF Percentage of users per transferred data volume](image)

Figure 26: Cumulative distribution function of the percentage of users per data traffic volume.

The number of RADIUS sessions per user, as is shown in Figure 27 shows an interesting point. One user opened 8 RADIUS sessions in 2 minutes. Further investigation showed that the sessions were not parallel but the user was opening and closing RADIUS sessions every few seconds. It is not known why the user opened so many sessions in so short notice. On each session the user was assigned a different IP address.
Graphs as the above could show potential DoS attacks on the core network, like trying to connect to the network many times per minute. The users that do not follow the behavior of normal users could be spotted easily as they would be located in another location in the graph.

Figure 27: Cumulative distribution function of the percentage of users per number of RADIUS sessions.
Chapter 5 Performance issues

Data networks are becoming faster and faster, thus a challenge for traffic recognition is to be done as fast as possible. The increase of the amount of traffic being transferred in a link per second by a factor of two, doubles the size of data for process by the traffic recognition software. The traffic recognition process may be split into two phases. Capture the data on storage and process the data. However, the storage required for Gbps speeds may be very large, particularly in traces of long duration. Furthermore, the analysis phase requires a considerable amount of time to process the data. The optimal solution would be to perform traffic recognition on the fly, by executing both phases at once and saving storage. In this case the performance of the analysis phase is of great importance. If it cannot keep up on the traffic speed the result would be that some packets would not be included in the analysis.

A packet analysis tool that analyzes the payload of the packets is known to perform slow so every optimization for performance is critical. The main purpose of the thesis, though, is not performance optimization. The comments mentioned below serve as experiences, observations and suggestions acquired by working with TAM and could be a guideline to improve the program's performance in the future.

The code for detecting the GTP and RADIUS protocols is located before the program most heavy function, the payload inspection, in order to save time by detecting network protocols in a prior state. Furthermore, control traffic like GTP-C or RADIUS messages is classified before the actual classification of the payload inspection function in order to improve the program's performance.

The program uses a linked list in order to store state information about GTP and RADIUS. The linked list was initially also used in order to get identification information for data packets, but performed poorly since it is required for every packet to locate the appropriate entry in the list, searching thousands of the list's items one by one. Thus a better way to operate the linked list has been used. The solution is based on the fact that the majority of GTP packets are G-PDU packets or just normal packets carrying user data. There are few control packets and only a part of them is being analyzed by TAM. In the GTP case only eight messages are analyzed, the Create request/response, Update request/response, Delete request/response and SGSN response/ acknowledge. In the RADIUS case only one message is analyzed,
the RADIUS Accounting Request. Thus, if the data packets stop using the linked list, TAM performance would increase significantly. Another fact is that the data packets do not require all information stored in the linked list. In the GTP case the destination IP address and TEID are enough in order to detect the PDP context. In the RADIUS case only the user IP address needs to be checked on the list. The solution is as follows. Use a hash table in the search for IMSI/MSISDN and APN. Keep the linked list for control traffic and use a hash table to find the IMSI/MSISDN and APN for the user data. The hash table does not need to store the actual values of the fields, just to point on the linked list item where the actual IMSI/MSISDN and APN values are stored. The performance gain is at least 50% faster execution time.

In the beginning the IMSI/MSISDN and APN fields were copied to temporary flows created by TAM for every packet but these flows are discarded when the packets are aggregated into flows. A better technique is to pass the pointer of the IMSI/MSISDN and APN fields to the flow aggregation function and copy the actual values of the IMSI/MSISDN and APN only when a new flow is to be created. That is because a flow should have the IMSI/MSISDN and APN fields in the beginning and their values should not be changed in the duration of the flow. That is accomplished via the use of the flow ID.

The program was also checked with a different hash table size in TAM from default 1000 entries to 100000 entries. The performance of the program did not increase. The results remained the same but the order of the flows in the output was changed because the hash table size is used in order to construct the key for each hash entry and a different key means a different location in the hash. When the flow hash table is being checked sequentially for expired flows, their order differs on the two cases producing a different order of flows in the program output.

The thesis concerns a single threaded version of TAM. There is also a multi threaded version of the program. However, in order to avoid additional complexity and because the single threaded version was the most recently updated, the single threaded version of the program was selected for the project. Transferring the current implementation to a multi-threading solution would require some changes in the code. Particularly, new locks are required for synchronization of the linked list. It would be better though to have separate threads handling the control and data traffic. As it is expected to be some delay in the network before the reception of the first data packet in a PDP context or RADIUS session, the method should not cause synchronization problems. Other data structures, like the one used to store the status information also require protection from multiple access.

The storage of the IMSI, APN and MSISDN values for each flow requires a considerable amount of memory and decreases the performance of the aggregation function. It is suggested to store the three additional fields for each flow to a dedicated data structure. However, doing this would damage the
binary flow storage function since then these fields would not be part of the binary flow file. An alternative solution is to completely remove the binary flow output and bind together the traffic analysis with the output of flow as strings. That is supported by the fact that the enhanced flow structure used consumes about 200 bytes of memory and some of the fields like the IP ID or the TCP sequence number need not be shown in the output so there is no reason to save these fields to a file. The size of the binary file produced in Trace3 was about 900 MB, where the alphanumeric file was half the size of it.

The part of TAM code that puts a limit in TAM performance is mostly the flow aggregation and the application recognition function. For example, running TAM with the -u option (output only the traffic for a particular user) for only 1 IMSI with very low traffic on Trace3, took only 12 minutes to complete while the execution time without the -u option is about 2.5³³ hours. Since the trace consists of 24 files, about 1.5 GB each, the processing of each file took about 30 seconds, a duration that is mostly attributed to disk activity rather than TAM performance leakage. Another test of the -u option was based on collecting all IMSI values from a previous program execution and use them as input to the user option. That considers 33011 IMSI values that are stored in a linked list. However, the program processing time was only 88 minutes. The actual process time on the trace without the user option enabled is about 150 minutes. The difference is attributed to the fact that the less flows processed by TAM reflect considerably faster program execution. Note that the output file was exactly the same as a previous TAM output with the expected difference of not including the unidentified flows.

The execution of TAM on Trace3 with a timeout parameter as 600 seconds instead of the default value of 64 seconds showed a small reduction in performance mainly because of the increased number of flows required to maintained into memory because the expiration time is larger. The output experienced a reduction of 16% to the number of unidirectional flows because of the larger timeout value. In addition, on Trace3 TAM required about 15 minutes in order to process the control messages only. The input file for TAM was only the control packets that were included in Trace3, about 700000 packets with total size 90 MB. That indicates a performance reduction mainly due to the use of a linked list implementation. However, as it is mentioned above, the linked list was used since there are many queries to be performed on the list. In the future, an alternative method in order to access the control state information maintained in TAM should be used to boost TAM performance. A potential solution is a large number of hash tables.

³³ The machine used throughout the study is a Pentium 4 CPU 3.40 GHz with 1GB RAM and OS Ubuntu Intrepid (kernel 2.6.27-11).
Considering the aggregation function, running TAM with input only one file from Trace3 but selecting the Internet interface\(^3\) that is treating all GTP packets as normal packets showed an execution time of about 30 seconds and very few flows. Running the program on the same trace file selecting this time the Gn interface showed an execution time of about 6.5 minutes. The big time difference is due to the number of flows required in the second case that is very large as now the user packets carried in GTP are being analyzed. As more and more flows are added, if we consider the whole Trace3 length, the program consumes more and more memory in order to keep all non expiring flows to memory. For example, the program uses about 120 MB of memory when it approaches the end of processing in Trace3.

If the payload inspection function in TAM could perform faster, the program performance would improve significantly. A suggestion in order to accomplish this is to use the port numbers contained in the transport layer protocols as a guide into which regular expressions should be executed, reducing the calculations per packet and improving the overall performance. For example, if a UDP port is 53 (assigned to DNS) then the DNS regular expression could execute first and if that fails other regular expressions could be executed against the packet payload in a specific predefined order, e.g. since the packet is supposed to be DNS perhaps a peer-to-peer application is abusing the port in order to bypass security mechanisms of the network so the peer-to-peer regular expressions could execute after DNS on this case.

The performance of the scripts used in order to extract statistics information from the TAM output was also considered. While most scripts executed fast, the script used in order to aggregate unidirectional to bidirectional flows required about 12 minutes to finish. The reason is the sort function that executed in the input file in order to arrange the flows in a chronological order, a prerequisite of the script in order to work correctly. Moreover, the script also executes the sort function in the output file in order to have the results ordered in time.

3\(^3\) The Internet interface is mostly the original TAM program that does not include the analysis of the GTP and RADIUS protocols.
6.1 Conclusion

The main purpose of the thesis project was the enhancement of TAM, a network traffic recognition tool using packet payload inspection, in order to associate information found on control messages of the cellular network to data traffic. Two interfaces on the core cellular network, the Gn and Gi interface were considered. The control information analyzed by TAM carried by the GTP and RADIUS protocols on the Gn and Gi interfaces respectively. The information consists of the user IMSI and MSISDN numbers and the APN that the user connects to.

The additional features required additional fields in the program output. However, the original program functionality remained intact as the support of the core cellular network interfaces is controlled via newly introduced startup options. One option operates TAM as a filter for the traffic of one or several users.

The success rate measured as the amount of user packets that can be associated with a user IMSI/MSISDN was not as satisfactory as we expected it to be since in some cases the trace files experienced problems or in other cases the duration of the traces was small and the border effect appeared to dramatically affect their results.

Verification of the new code started as soon as trace files were provided to the author and many tests were performed in order to guarantee the validity of the TAM output. That included the manual study of packet contents, mainly for control traffic. The verification continued with the comparison of TAM output to CFlow and MFlow. That showed some weaknesses in the program, like the support for bidirectional flows. Some weaknesses were solved via new features in TAM, other required post processing of the output of the program and the rest, referring mostly to IP fragment reassembly, are suggested as future work. The results of the comparison could help improve all programs as no program performed flawlessly but still TAM performed similarly to the other programs in the majority of the tests performed, indic-
ating the robustness of the program. However, it should be noted that nor TAM, neither CFlow or MFlow, are publicly available.

The post processing of TAM output included also the extraction of statistics. A limited number of statistics are presented in paragraph 4.6, mainly in order to show the information that could be extracted from the program. Results acquired by analysis of larger traces together with additional information on the networks of study are expected to demonstrate valuable information about the users and the networks.

The performance of the program remained as a concern throughout the project although not a high priority one. Based on measurements performed, the function limiting the program's performance was the flow aggregation that includes the payload inspection of every examined packet. The new features consumed up to 15 minutes from the total TAM execution time of 2.5 hours in Trace3.

6.2 Discussion

What is the main contribution of the report is the indication that network analysis via packet capture requires the correct handling of many parameters and the prediction of all situations that something may go wrong. The verification of a tool bycompanying it to similar tools could help alleviate possible mistakes in the program but it may also raise questions as of which program output is the “correct” one, leaving the final decision to the examination to the actual data that is a labor intensive job and not always sufficient if the knowledge of how the tools operate is not available. Anyhow, the program is expected to work correctly as it has been tested against many trace data and other programs. However, there is always room for correction and as the product of the thesis is considered a prototype, its results should be validated first and not be blindly interpreted.

A suggestion to other people working on the area, or any area concerning both control and data traffic is to extract the control traffic of a trace and study it separately as this could save valuable time. When the comprehension of control traffic is correct then the data can be examined.
6.2.1 Limitations

Because of the limited time of the project, limits that were put by the program itself and features of the studied protocols that did not occur in the studied trace files, some limitations apply to the program as is shown below:

- The only link layer protocol supported is Ethernet, with the support of VLAN.
- Only the IP version 4 and GTP version 1 protocols are supported.
- The IP fragments and the GTP MBMS messages are not analyzed.
- Only RADIUS Accounting protocol is used for user profiling on the Gi interface. Moreover, the Framed IP Address is used in order to determine the user network address but the Framed IP Netmask is not analyzed. The later is used in order to assign a network subnet to the user. However, no RADIUS packet carrying this field was detected in the study and it is not supported in TAM.
- The maximum IMSI or MSISDN field length is 20 digits and the maximum APN field length is 100 characters.
- The TEID value zero and Sequence Number zero in the GTP header are considered as special numbers and are not expected to be encountered in control packets.
- Finally, the correct program execution depends on the correct structure of the protocol headers. Although TAM detects many errors that stem from malformed packet headers is not expected to cover every possible case. In addition, correct application recognition depends on the number of application that can be recognized by TAM, how well structured are their detection rules (e.g. regular expressions) and the presence of tunneling and encryption on the packet payload.

6.2.3 Future work

TAM is a prototype tool and as a result lacks some features that could improve the program's accuracy, efficiency and performance. Here features that TAM should support and features that the author would like TAM to support are summarized in a prioritized order according to the author's beliefs.

1. Use the port fields of a flow as a storage place for the type and code of ICMP packets. Alternatively, use additional flow fields for that information.
   The study of ICMP messages could help draw additional conclusions about the network.

Excluding the *Create PDP Context Request* TEID.
2. Handling of IP version 4 fragments. Better handling of duplicate packets produced by the capturing procedure. In the core cellular network, IP fragmentation occurs regularly and the omission of the information carried in fragmented packets reduces the accuracy of the program. Moreover, an IP fragment reassembly solution could also help ignore the duplicate IP packets produced by the capturing mechanism as these packets should be excluded from analysis as soon as possible.

3. Robust measurement of the TCP behavior
TCP semantics are very useful in order to draw conclusions about the performance of a network and the effects of the wireless link to the user. Properties like the TCP sequence, the TCP flags seen and what information is acknowledged should be maintained per flow. That would probably result on the replacement of unidirectional to bidirectional flow structures in TAM. We consider this replacement as a must for efficient memory consumption of the program.

4. Storage of the information about the user of a flow to a separate data structure in order to keep the flow hash table as small and efficient as possible. The linked list used in the code also requires to be changed to a faster data structure that supports the same functionality.

5. GTP version 2 support
The protocol specification is almost complete\textsuperscript{36} and deployment is expected to start in a few years period. A traffic recognition tool like TAM could help detect possible problems in the first deployment of the new technology.

6. IP version 6 support
Although the new IP version is not broadly used it could help solve problems, as NAT used in the core cellular network, in the near future and TAM should be ready to adapt to the new network protocol. Furthermore, support for ICMP version 6 should also be included.

7. Diameter protocol support on the Gi interface
Although not encountered in the study the DIAMETER protocol is the evolution of RADIUS and it is expected to replace RADIUS in the Gi interface.

8. Support for additional GTP version 1 messages, particularly MBMS
The traces studied did not contain MBMS messages. As in the future this could change, their implementation should be included in TAM.

9. Acquisition of additional information from the network, like the QoS value used in PDP contexts, and more statistics about the SGSN and GGSN like the traffic that each network element experiences.

\textsuperscript{36} There still exist some issues about GTPv2 in the 3GPP mailing list. Last checked, February 2009.
10. Verification of the packet inspection function of the program in order to reduce false positives. Moreover, detection capability of more applications with a preference to applications running on the GPRS network, e.g. WAP.

11. Parsing of the data two or several times may improve the results by identifying more packets belonging to PDP contexts. As a result it should be checked if the feature is able to improve the program's accuracy. An alternative is the study of traces with large duration.

12. Finally, depending on the requirement of executing the program in online mode studying live traffic, the performance of TAM should be considered by enforcing multi-threading processing of packets.

Furthermore, it is worth investigating the use of TAM output in order to detect bottlenecks in the GPRS network, similarly to work done on the METAWIN project[23].

If the above suggested factors in order to improve performance prove unsatisfactory, the utilization of port numbers could be used in order to define an order of regular expressions to be executed in the payload of the packet. For example, a packet with TCP port 80 should be checked first for the presence of the HTTP protocol and if that fails all other regular expressions could be executed. The last resort is the fact that the performance of software tools is considered limited against pure hardware implementations. Thus, perhaps a hardware implementation is required in order to keep up operating at network speeds.
Bibliography

8. TS Choi, CH Kim, SH Yoon. JS Park, BJLee, HH Kim, HS Chung, TS Jeong, Content aware Internet application traffic measurement and analysis, Network Operations and Management Symposium, 2004.
19. Philipp Svoboda, Fabio Ricciato, Eduard Hasenleithner, and Rene Pilz, Composition of GPRS UMTS traffic snapshots from a live network, 4th International
22 VERINT, accessed on March 2009, verint.com
31 3GPP, Interworking between the Public Land Mobile Network (PLMN) supporting packet based services and Packet Data Networks (PDN), TS 29.061 version 7.7.0 Release 7, 2008.
33 G. Dommety, Key and Sequence Number Extensions to GRE, RFC 2890, September 2000.
36 3GPP, Evolved General Packet Radio Service (GPRS) Tunneling Protocol for Control plane (GTPv2-C), TS 29.274 version 8.0.0 Release 8, 2008.
37 3GPP, Telecommunication management Charging management Charging Data Record (CDR) transfer, 3GPP TS 32.295 version 7.0.0 Release 7, 2007.
38 3GPP, LTE GPRS Tunneling Protocol for User Plane (GTPv1-U), TS 29.281 version 8.0.0 Release 8, 2009.
44 3GPP, Mobile radio interface Layer 3 specification Core network protocols, TS 24.008 version 7.13.0 Release 7, 2008.
Appendices

Additional abbreviations and definitions about the cellular network can be found in [49**].

Appendix A  Definitions

Flow classification  The process of assigning an application to the flow.
Flow identification  The process of assigning a user to the flow.
Identified flows  The flows that are known to belong to a particular user. The user of the flows is known via its IMSI or MS-ISDN. The APN of the flow is also known.

Appendix B  Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>3GPP</td>
<td>3rd Generation Partnership Project</td>
</tr>
<tr>
<td>AAA Server</td>
<td>Authentication, Authorization and Accounting Server</td>
</tr>
<tr>
<td>APN</td>
<td>Access Point Name</td>
</tr>
<tr>
<td>ARP</td>
<td>Address Resolution Protocol</td>
</tr>
<tr>
<td>DHCP</td>
<td>Dynamic Host Configuration Protocol</td>
</tr>
<tr>
<td>DHT</td>
<td>Distributed Hash Table</td>
</tr>
<tr>
<td>EDGE</td>
<td>Enhanced Data Rates for GSM Evolution</td>
</tr>
<tr>
<td>ETSI</td>
<td>European Telecommunications Standards Institute</td>
</tr>
<tr>
<td>EUA</td>
<td>End User Address</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field-Programmable Gate Array</td>
</tr>
<tr>
<td>GERAN</td>
<td>GSM EDGE RAN</td>
</tr>
<tr>
<td>GGSN</td>
<td>Gateway GPRS Support Node</td>
</tr>
<tr>
<td>GPRS</td>
<td>General Packet Radio Service</td>
</tr>
<tr>
<td>GRE</td>
<td>Generic Routing Encapsulation</td>
</tr>
<tr>
<td>GSM</td>
<td>Global System for Mobile communication</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>GSN</td>
<td>GPRS Support Node</td>
</tr>
<tr>
<td>GTP</td>
<td>GPRS Tunneling Protocol</td>
</tr>
<tr>
<td>IANA</td>
<td>Internet Assigned Numbers Authority</td>
</tr>
<tr>
<td>IDS</td>
<td>Intrusion Detection System</td>
</tr>
<tr>
<td>IE</td>
<td>Information Element</td>
</tr>
<tr>
<td>IMSI</td>
<td>International Mobile Subscriber Identity</td>
</tr>
<tr>
<td>ISP</td>
<td>Internet Service Provider</td>
</tr>
<tr>
<td>L2TP</td>
<td>Layer 2 Tunneling Protocol</td>
</tr>
<tr>
<td>LNSAPI</td>
<td>Linked NSAPI</td>
</tr>
<tr>
<td>MBMS</td>
<td>Multimedia Broadcast Multicast Service</td>
</tr>
<tr>
<td>MM</td>
<td>Mobility Management</td>
</tr>
<tr>
<td>MS</td>
<td>Mobile Station</td>
</tr>
<tr>
<td>MSISDN</td>
<td>Mobile Subscriber ISDN Number</td>
</tr>
<tr>
<td>NAS</td>
<td>Network Access Server</td>
</tr>
<tr>
<td>NAT</td>
<td>Network Address Translation</td>
</tr>
<tr>
<td>NSAPI</td>
<td>Network Service Access Point Identifier</td>
</tr>
<tr>
<td>NTP</td>
<td>Network Time Protocol</td>
</tr>
<tr>
<td>P2P</td>
<td>Peer-to-peer</td>
</tr>
<tr>
<td>PCO IE</td>
<td>Protocol Configuration Option IE</td>
</tr>
<tr>
<td>PDN</td>
<td>Packet Data Network</td>
</tr>
<tr>
<td>PDP</td>
<td>Packet Data Protocol</td>
</tr>
<tr>
<td>PDU</td>
<td>Protocol Data Unit</td>
</tr>
<tr>
<td>PLMN</td>
<td>Public Land Mobile Network</td>
</tr>
<tr>
<td>PPP</td>
<td>Point-to-Point Protocol</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RADIUS</td>
<td>Remote Authentication Dial In User Service</td>
</tr>
<tr>
<td>RAN</td>
<td>Radio Access Network</td>
</tr>
<tr>
<td>RNC</td>
<td>Radio Network Controller</td>
</tr>
<tr>
<td>SGSN</td>
<td>Serving GPRS Support Node</td>
</tr>
<tr>
<td>TAM</td>
<td>Traffic Analysis Module</td>
</tr>
<tr>
<td>TEID</td>
<td>Tunnel Endpoint Identifier</td>
</tr>
<tr>
<td>TFT</td>
<td>Traffic Flow Template</td>
</tr>
<tr>
<td>UMTS</td>
<td>Universal Mobile Telecommunications System</td>
</tr>
<tr>
<td>UTRAN</td>
<td>UMTS Terrestrial RAN</td>
</tr>
<tr>
<td>VLAN</td>
<td>Virtual LAN</td>
</tr>
<tr>
<td>VoIP</td>
<td>Voice over IP</td>
</tr>
<tr>
<td>WAP</td>
<td>Wireless Application Protocol</td>
</tr>
</tbody>
</table>
Appendix C  GTP Protocol

The GTP protocol header is shared between the two GTP types, GTP-C and GTP-U. It has a minimum length of 8 bytes and is shown in Figure 28.

Three flags control the appearance of optional fields. If any one of the flags is set then all three optional fields are present in the GTP header and its size becomes 12 bytes. Only the fields that their flag is set are being evaluated by the receiver. The GTP header fields are shown in Table 8.

Table 8: GTP header fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version (3 bits)</td>
<td>Version of the GTP protocol, set to 1</td>
</tr>
<tr>
<td>Protocol type (1 bit)</td>
<td>Discriminates between GTP (1) and GTP'(0)</td>
</tr>
<tr>
<td>Extension header flag E (1 bit)</td>
<td>If set then the Next Extension Header field is present in the GTP header and should be interpreted by the receiver. If the flag is set, all three options are present.</td>
</tr>
<tr>
<td>Sequence number flag S (1 bit)</td>
<td>If set then the Sequence Number field is present in the GTP header and should be interpreted by the receiver. If the flag is set, all three options are present.</td>
</tr>
<tr>
<td>N-PDU Number flag PN (1 bit)</td>
<td>If set then the N-PDU Number field is present in the GTP header and should be interpreted by the receiver. If the flag is set, all three options are present.</td>
</tr>
<tr>
<td>Message type (8 bits)</td>
<td>Indicates the type of the GTP message</td>
</tr>
<tr>
<td>Length (16 bits)</td>
<td>The length of the payload in bytes (the number of bytes that follow the mandatory 8 bytes of the GTP header)</td>
</tr>
<tr>
<td>Tunnel Endpoint Identifier (TEID) (32 bit)</td>
<td>This field unambiguously identifies a tunnel endpoint in the receiving GTP-U or GTP-C node. The receiving node of a GTP tunnel locally assigns the TEID value the transmitting node has to use. The TEID values are exchanged between tunnel endpoints using GTP-C messages.</td>
</tr>
</tbody>
</table>
Sequence Number (16 bits)  Optional field that appears depending on the value of the S flag. For the GTP-C case it is used to bind together requests with responses (they have the same sequence number). For the GTP-U case is used to indicate the order that packets have been sent when their transmission order should be preserved.

N-PDU Number (8 bits)  The field is used in handover procedures to assist the MS and the SGSN operating in the acknowledged mode.

Next Extension Header Type (8 bits)  If the field's value is other than 0, this field indicates that an extension header follows. It is used to include extensions to the GTP protocol without the need of a new protocol version.

The extension headers that are used by GTP have a common format as shown in Figure 29 and Table 9.

<table>
<thead>
<tr>
<th>Octet 1</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 29: Format of the GTP extension header.

Table 9: GTP extension header fields

| Extension Header Length (8 bits) | The length of the extension header in multiples of 4 bytes. All extension headers must have size multiple to 4 bytes. |
| Extension Header Content (variable length) | The content (value) of the extension header |
| Next Extension Header (8 bits) | If the field's value is other than 0, this field indicates that another extension header follows. |

There are currently defined five extension headers. However, none of these is related to our study. The specification defines also many messages for the GTP protocol. Their purpose is to carry control information (via the information elements) or user data. Some of the messages operate on GTP and are not mentioned here. The messages are split into three categories, the ones that operate on GTP control, the messages that operate on GTP user and some messages like Echo Request/Echo Response that operate on both GTP types.

A complete analysis of the GTP messages can be found in [28**]. Here only the relevant to the study GTP messages are analyzed and explained.

GTP control messages:

- Create PDP Context Request
  Message from the SGSN to GGSN that is used to create a new PDP context. The message contains the IMSI of the user, the APN and defines the control and data tunnel endpoints used for the PDP context in the SGSN
Create PDP Context Response
Message from the GGSN to the SGSN that is used as a response to a create request. Contains the status for the context (if it was accepted and if it was not, what was the problem), the tunnel endpoint used for the PDP context in the GGSN and may carry a network address to be assigned to the user.

Update PDP Context Request
Message sent by the GGSN or SGSN to update the context properties. It may update the QoS properties, the user network address and the tunnel endpoints. The IMSI and APN are defined in the creation of the PDP context and cannot be changed via an update procedure.

Update PDP Context Response
Message sent as a response to the update request. It contains the status if the update was successful or not. It may also update the tunnel endpoints on this side of the connection.

Delete PDP Context Request
Message sent by the GGSN or SGSN and used to delete the PDP context. Upon reception, the PDP context, if exists, is being deleted.

Delete PDP Context Response
Message sent as a response to the delete request. It carries the status of the delete request. However, the specification defines that the PDP context is deleted when the request is sent/received and not when the response is received.

SGSN Context Response
Message sent as a response to a request for MM and PDP contexts of a user by a SGSN. It carries the MM and PDP context information of a user from the old SGSN to the new SGSN during a SGSN handover procedure. In TAM the message can be used to construct PDP context information when the Create PDP Context message was not captured.

SGSN Context Acknowledge
Message sent as a response to the SGSN Context Response message. It is sent by the new SGSN to the old SGSN to confirm the correct context handover procedure.

GTP user messages:

G-PDU
Message sent by the GGSN or SGSN. Is the most common message type in GTP since it is used to carry user data as payload. It uses the GSN addresses and the tunnel identifiers specified by the control messages to send the user data to the appropriate GSN and tunnel. There are no IE used in message.

The Create PDP Context Request message has two types. Either a primary PDP context request or a secondary PDP context request. The secondary PDP context creates a new PDP context related to the already existing primary PDP context for a user that can have different QoS and TFT. The primary and secondary PDP contexts share the same IMSI, APN and EUA.
Every PDP context has an NSAPI assigned to it. The IMSI together with the NSAPI uniquely identify a PDP context. Every PDP context request contains the NSAPI assigned to the PDP context. The secondary PDP context request contains also the LNSAPI that is the NSAPI of the related primary PDP context request.

The Create PDP Context Request message contains the TEID used for control and data for the PDP context and the SGSN addresses used for control and data that the TEID should be destined to. The Create PDP Context Response message contains the TEID and GGSN addresses used for control and data messages. The response should have as destination the SGSN address for control messages defined in the request message and as TEID the TEID for control messages defined in the request message.

The update and delete messages use the appropriate control GSN addresses and TEID. The update messages change properties of the PDP context like the QoS, but may also change the control and data TEID.

The data messages use the data GSN addresses and TEID.

The GTP protocol defines structures that carry control information as a payload to messages. These structures are called Information Elements (IE). There are two types of IE used, TLV (Type, Length, Value) or TV (Type, Value). The order of the IE in the GTP header is the ascending order of their type field.

The presence of an IE in a GTP packet is defined by the GTP message and their appearance may be

- mandatory the IE should be present in the message at all times
- conditional the IE should be present in the message based on the specific situation
- optional the IE may be present in the message or not, it usually refers to service options

The format of the IE depends on the IE type as is shown in Figures 30 and 31. Their description is in Tables 10 and 11.

![Figure 30: The format of the GTP TypeValue IE.](image)

Table 10: Fields of a Type Value IE

<table>
<thead>
<tr>
<th>Octet 1</th>
<th>Bit 8 7 6 5 4 3 2 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Length is based on the IE Type</td>
</tr>
<tr>
<td>Length is based on the IE Type</td>
<td>IE TV Type</td>
</tr>
<tr>
<td></td>
<td>Value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type (8 bits)</th>
<th>Value (length is based on the IE type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>the type of the IE. For TV IE the first bit is always 0</td>
<td>the value that is carried by the IE. Its size is defined by the IE type</td>
</tr>
</tbody>
</table>
Table 11: Fields of a Type Length Value IE.

<table>
<thead>
<tr>
<th>Type (8 bits)</th>
<th>the type of the IE. For TLV IE the first bit is always 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length (16 bits)</td>
<td>the length in bytes of the IE value, excluding the type and length fields</td>
</tr>
<tr>
<td>Value</td>
<td>the value that is carried by the IE. Its size is defined by the length field</td>
</tr>
</tbody>
</table>

Since only a subset of the GTP messages is used in the study, only a subset of the IE need to be analyzed. However, the use of a predefined size in some of the IE require that the size of these IE be included in the code even if these IE do not carry any information related to the study.

As mentioned above, the order of IE in the messages is specific. It is based on the ascending order of their type field. Thus, IE with a small type number appear first in the message.

There are some IE that are used by the GTP' protocol and are defined in [37**]. As GTP' messages are not considered in the study, these IE are also not considered. In any case it is not expected that a GTP' IE would be present in a GTP header.

The format of the GTP header is as follows:
- first is the mandatory part of the header that is present in all GTP packets
- the mandatory header may be followed by zero or more extension headers
- based on the message type in the mandatory header, next follow IE or a Protocol Data Unit (PDU)[50**].

Below follow in pseudo-code the operations executed by TAM upon reception of each GTP message. For simplicity, failsafe code is not included.

**Delete PDP Context Request**

**Search** Entry fields TEID_C_G, IP_C_G, TEID_C_S, IP_C_S using Packet.TEID and Packet.IP_dst

**If** entry is found
- Entry.Tearown = Packet.Tearown
- Entry.Del_Seq_Num = Packet.Seq_Num
**Delete PDP Context Response**


If Entry is found
  If Entry.Teardown equals 1
    If Entry.EUA is valid
      Delete* all entries with same Entry.IMSI and Entry.EUA
    Else
      Delete* Entry
  Else
    Delete* Entry

* The Delete function also removes the hash entries

**Create PDP Context Request**

If Packet.TEID equals 0
  // primary context request

  If Entry is found
    // duplicate entry
    Return
  Else
    Add Entry
    Entry.Seq_Num = Packet.Seq_Num
    Add Hash_Entry_S
  Else
    // secondary context request

    If Entry is found
      Fetch Entry properties

      If Entry is found
        // duplicate entry
        Return
      Add New_Entry
      New_Entry.Seq_Num = Packet.Seq_Num
      Add Hash_Entry_S
Create PDP Context Response

Search Entry fields TEID_C_S, IP_C_S, Seq_Num using Packet.TEID, Packet.IP_dst, Packet.Seq_Num
If Entry is found
   If Packet.Cause equals Accepted
      Update Entry properties
      Entry.Seq_Num = 0
      Add Hash_Entry_G
   Else
      Return

Update PDP Context Request
If Packet.TEID equals 0
   // the message is used in order to change GTP version
   Return
Else
   If exists Packet.TEID_D
      // update source SGSN
      Search Entry fields TEID_C_D, IP_C_D, NSAPI using Packet.TEID, Packet.IP_dst, Packet.NSAPI
      If Entry is found
         Update Entry properties
         Entry.Upd_Seq_Num = Packet.Seq_Num
      Else
         // update source GGSN
         Search Entry fields TEID_C_S, IP_C_S, NSAPI using Packet.TEID, Packet.IP_dst, Packet.NSAPI
         If Entry is found
            Update Entry properties
            Entry.Upd_Seq_Num = Packet.Seq_Num
         Else
            Return
         End
      End
   Else
      Return
   End
Update PDP Context Response


If Entry is found
  // update source SGSN
  If Packet.Cause does not equal Accepted
    If Entry.Upd_EUA does not equal 0
      Delete Entry
  Return
  Else
    Update Entry properties
    Entry.Upd_Seq_Num = 0
    Update Hash_Entry_S

Else
  Search Entry fields TEID_C_S, TEID_C_SU, IP_C_S, IP_C_SU, Upd_Seq_Num using Packet.TEID, Packet.IP_dst, Packet.Seq_Num

If Entry is found
  // update source GGSN
  If Packet.Cause does not equal Accepted
    If Packet.Cause equals Non-Existent
      Delete Entry
  Return
  Else
    Update Entry properties
    Entry.Upd_Seq_Num = 0
    Update Hash_Entry_S
    Update Hash_Entry_G

SGSN Context Response

If Packet.Cause equals Accepted
  Iterate all PDP contexts of the packet
  Search Entry fields IMSI, NSAPI using Packet.IMSI, Packet.NSAPI

If Entry is not found
  Add Entry
  Entry.SGSN_Seq_Num = Packet.Seq_Num
  Add Hash_Entry_G
Appendix D  Protocols of the Gi Interface

D.1 RADIUS Accounting Protocol

The format of the RADIUS header is shown in Figure 32 and commented in Table 12, below:

<table>
<thead>
<tr>
<th>Octets</th>
<th>Octet</th>
<th>Code</th>
<th>Identifier</th>
<th>Length</th>
<th>Request/Response Authenticator</th>
<th>RADIUS Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable length</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 32: Format of the RADIUS header.
Table 12: RADIUS Accounting header fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code (8 bits)</td>
<td>defines the type of the RADIUS packet and its value is 4 for RADIUS Accounting Request and 5 for the Accounting Response</td>
</tr>
<tr>
<td>Identifier (8 bits)</td>
<td>is used to match an Accounting Request to the related Accounting Response, that is, both the request and the response share the same identifier</td>
</tr>
<tr>
<td>Length (16 bits)</td>
<td>the length of the RADIUS packet in bytes</td>
</tr>
<tr>
<td>Request/Response Authenticator (128 bits)</td>
<td>is a 16 octets value and is used to authenticate the messages sent between the RADIUS client and server via a shared secret</td>
</tr>
<tr>
<td>Attributes (variable length)</td>
<td>carry information related to the specific type of RADIUS packet. A RADIUS packet may contain zero or more attributes.</td>
</tr>
</tbody>
</table>

The RADIUS attributes may be located in any order inside the RADIUS packet. However, the order of the attributes that have same type in a packet should not be changed. The format of the attributes is shown in Figure 33 and short comments follow in Table 13.

![Figure 33: Format of the RADIUS attributes.](image)

Table 13: RADIUS Accounting attribute format.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type (8 bit)</td>
<td>defines the type of the RADIUS attribute</td>
</tr>
<tr>
<td>Length (8 bit)</td>
<td>the length of the attribute in bytes</td>
</tr>
<tr>
<td>Value (arbitrary length)</td>
<td>the value of the attribute. Its format is specified by the type field.</td>
</tr>
</tbody>
</table>

The Radius attributes that concern the study are shown in Table 14.

Table 14: Important Radius Attributes.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acct-Status-Type</td>
<td>specifies the beginning (start, 1) or end (stop, 2) of the user service</td>
</tr>
<tr>
<td>Acct-Session-ID</td>
<td>is used to facilitate matching the start and stop messages for a user session</td>
</tr>
<tr>
<td>Framed-IP-Address</td>
<td>the IP address of the user being accounted</td>
</tr>
</tbody>
</table>
The operations executed by TAM upon reception of a RADIUS Accounting packet are shown below:

**RADIUS Accounting Request**

If Packet.Code equals Accounting Request

  If Packet.Acct-Status-Type equals Start

    Search Entry fields IP_Address, Session_ID, MSISDN, APN

    using Packet.Framed_IP_Address, Packet.Acct_Session_ID,

    Packet.Calling_Station_ID, Packet.Called_Station_ID

  If Entry is found

    // duplicate packet

    Return

  Else

    Add Entry

    Add Hash_Entry

  Else If Packet.Acct-Status-Type equals Stop

    Search Entry fields IP_Address, Session_ID

    using Packet.Framed_IP_Address, Packet.Acct_Session_ID

  If Entry is found

    Delete* Entry

* The Delete function also removes the hash entries

---

**Data packet**

Search Hash_Entry fields IP_address using Packet.IP_src

If Hash_Entry is found

  Classify Packet with

    Direction

    RADIUS_ID

    MSISDN

    APN

    IMSI

Else

Search Hash_Entry fields IP_address using Packet.IP_dst

If Hash_Entry is found

  Classify Packet with

    Direction

    RADIUS_ID

    MSISDN

    APN

    IMSI
D.2 GRE

The GRE[33**] protocol encapsulates packets via a GRE header that follows the IP protocol. The structure of the resulting packet is:

<IP>-<GRE>-<encapsulated packet>

The packet being encapsulated via GRE may in theory be any protocol defined by the GRE protocol type field. In the study, the most common protocol encountered following GRE was IP version 4. PPP may also be following GRE though.

The GRE header defined in [33**] with the additional fields of [34**] is shown in Figure 34. The analysis of the header fields is outside the scope of the study.

<table>
<thead>
<tr>
<th>Bytes/Bits</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C</td>
<td>R</td>
<td>K</td>
<td>S</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7-8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9-12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13-16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 34: The GRE protocol header.

GRE version 1[51**] has a similar header as GRE version 0 described above but it is only used in order to carry PPP packets.

D.3 L2TP - PPP

The L2TP[34**] protocol encapsulates PPP[48**] packets above the UDP protocol using UDP port 1701. The packet structure is:

<IP>-<UDP>-<L2TP>-<PPP>-<encapsulated packet>

The PPP packet could also encapsulate an IP packet, or some other protocol carried by PPP.

The L2TP header is shown in Figure 35. The analysis of the header fields is outside the purpose of the study. However, the specification defines that flag T is used in order to determine the packet type, control or data packets. TAM only analyzes data packets and skips the control packets.
The PPP header, according to [47**], is shown below in Figure 36. Note the difficulty on determining the protocol.

<table>
<thead>
<tr>
<th>Bytes/Bits</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>L</td>
<td>R</td>
<td>R</td>
<td>S</td>
<td>R</td>
<td>O</td>
<td>P</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7-8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9-10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11-12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13-14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15-16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Length (optional via L flag)*

**Tunnel ID**

**Session ID**

*Ns (optional via S flag)*

*Nr (optional via S flag)*

*Offset Size (optional via O flag)*

*Offset Padding (optional via O flag)*

Figure 35: The L2TP protocol header.

The PPP header, according to [47**], is shown below in Figure 36. Note the difficulty on determining the protocol.

<table>
<thead>
<tr>
<th>Bytes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Address (Value 0xFF) *</td>
</tr>
<tr>
<td>2</td>
<td>Control (Value 0x03) *</td>
</tr>
<tr>
<td>3-4</td>
<td>Protocol **</td>
</tr>
</tbody>
</table>

* The fields may be compressed (skipped)
** The field may be compressed (occupy 1 byte)

Figure 36: PPP header.

### Appendix E  Program status information

The status information produced by TAM, is divided into several categories as presented below. Note that the statistics are displayed only if their value is non zero.

#### Category 1 - Skipped packets:

displays the number of completely skipped packets from TAM analysis.

- Ethernet: too small length to contain Ethernet + VLAN + IP headers
  - Number of packets that were too small to contain an IP header.
- Ethernet: protocol is IPv6
  - Number of packets containing IPv6 as Ethernet protocol.
- Ethernet: protocol is Unknown
  - Number of packets that do not contain IPv4 as Ethernet protocol.
- IP: header length is smaller than normal(20)
  - The IP header length is less than 20 bytes that indicates malformed packets or packets that do not carry IPv4.
- IP: fragments, except fragment 0
  - Number of packets that are IP fragments. Fragment 0 is not skipped though.
- UDP: too small length to contain UDP header
  - Number of IP packets that were too small to contain an UDP header although they carried the UDP protocol.
- TCP: too small length to contain TCP header
  - Number of IP packets that were too small to contain a TCP header although they carried the TCP protocol.
- Duplicate
  - Number of packets found as duplicate according to the algorithm presented in Chapter 3.

Category 1.1 - Skipped packets on Gi interface:
displays the number of completely skipped packets from TAM analysis only on Gi interface.
- L2TP: too small length to contain UDP + L2TP + IP headers
  - Number of packets that were too small to contain an IP header.
- PPP: protocol is not IPv4
  - Number of packets that do not contain IPv4 as PPP protocol.
- GRE: too small length to contain GRE + IP headers
  - Number of packets that were too small to contain an IP header.
- GRE: version is not 0 or 1
  - Number of GRE packets that contained GRE version other than 0 or 1. Only the previous versions GRE versions are supported.
- GRE: protocol is not IPv4 or PPP
  - Number of packets that do not contain IPv4 as GRE or PPP protocol.

Category 2 - Protocol information:
displays information about the encountered protocols
- ICMP: number of packets
  - Number of packets carrying ICMP.
- Other IP protocol: number of packets
  - Number of packets carrying other than TCP, UDP, ICMP protocol.
- Skipped encapsulated pkts (IP: version is not 4)
  - Number of packets that were not decapsulated because the IP version was not 4.
- Skipped encapsulated pkts (IP: header length is smaller than normal(20))
  - Number of packets that were not decapsulated because the IP header length was less than 20 bytes that indicates malformed packets or packets that do not carry IPv4.
- Skipped encapsulated pkts (IP: fragments, except fragment 0)
• Number of packets that were not decapsulated because they were IP fragments. Fragment 0 is decapsulated.

• Skipped encapsulated pkts (duplicate)
  ◦ Number of packets that were duplicate and carried encapsulated data. The value serves only as information. The packets are completely skipped as duplicates.

Category 2.1 - Protocol information on Gi interface:
  displays information about the encountered protocols on the Gi interface

  • L2TP: too small length to contain L2TP header
    ◦ Number of packets that were not decapsulated because they were too small to contain a L2TP header.

  • L2TP: number of packets
    ◦ Number of packets that contain L2TP.

  • L2TP: packet does not contain L2TP version 2, ignored
    ◦ Number of packets that were not analyzed for L2TP because the version of the protocol was not 2.

  • L2TP control: number of packets
    ◦ Number of packets that contain L2TP control. They do not carry encapsulated data.

  • L2TP data: number of packets
    ◦ Number of packets that contain L2TP data. They carry encapsulated data.

  • PPP: number of packets
    ◦ Number of packets that contain PPP.

  • RADIUS Accounting: number of packets
    ◦ Number of packets that contain RADIUS Accounting.

  • RADIUS Accounting Requests: number of packets
    ◦ Number of packets that contain RADIUS Accounting Requests

  • RADIUS: wrong length of attribute Acct_Status_Type(6)
    ◦ Number of RADIUS Accounting Request packets that were not analyzed because of wrong length of attribute Acct_Status_Type(6).

  • RADIUS: wrong length of attribute Framed_IP_Address(6)
    ◦ Number of RADIUS Accounting Request packets that were not analyzed because of wrong length of attribute Framed_IP_Address(6).

  • RADIUS: wrong length of attribute Acct_Session_ID(>=3 && <=30)
    ◦ Number of RADIUS Accounting Request packets that were not analyzed because of wrong length of attribute Acct_Session_ID(>=3 && <=30).

  • RADIUS: wrong length of parameter Calling_Station_ID(>=3 && <=20)
    ◦ Number of RADIUS Accounting Request packets that were not analyzed because of wrong length of attribute Calling_Station_ID(>=3 && <=20).
• RADIUS: wrong length of attribute Called_Station_ID(>=3 && <=100)
  ◦ Number of RADIUS Accounting Request packets that were not analyzed because of wrong length of attribute Called_Station_ID(>=3 && <=100).
• RADIUS: wrong length of attribute Vendor_Specific(>=7)
  ◦ Number of RADIUS Accounting Request packets that were not analyzed because of wrong length of attribute Vendor_Specific(>=7).
• RADIUS: number of Vendor_Specific 3GPP attributes detected
  ◦ Number of Vendor_Specific 3GPP attributes detected in the analyzed RADIUS packets.
• RADIUS: number of Vendor_Specific 3GPP IMSI attributes detected
  ◦ Number of Vendor_Specific 3GPP IMSI attributes detected in the analyzed RADIUS packets.
• RADIUS: fail to detect all necessary attributes (acct_status_type, framed_ip_address, acct_session_id, calling_station_id, called_station_id)
  ◦ Number of RADIUS packets that did not contain all the required fields in order to create the RADIUS session. It happens also when the Acct_Status_Type attribute has other value than 1 (start) and 2 (stop).
• RADIUS Accounting Req Start: number of packets
  ◦ Number of RADIUS Accounting Request packets having Acct_Status_Type 1 (start).
• RADIUS Accounting Req Start: add entry
  ◦ Number of entries created in TAM in order to store session information.
• RADIUS Accounting Req Start: entry already exists, ignore
  ◦ Number of times that the new entry were not inserted in TAM because it already existed.
• RADIUS Accounting Req Stop
  ◦ Number of RADIUS Accounting Request packets having Acct_Status_Type 2 (stop).
• RADIUS Accounting Req Stop: remove entry
  ◦ Number of entries deleted in TAM because the sessions stopped.
• RADIUS Accounting Req Stop: entry does not exist
  ◦ Number of times that the entry could not be found in TAM. TAM does not contain information about the RADIUS session.
• RADIUS Accounting: number of identified packets
  ◦ Number of data packets that were identified on the Gi interface.

Category 2.2 - Protocol information on Gn interface:
displays information about the encountered protocols on the Gn interface
• UDP: too small length to contain UDP header
- Number of not decapsulated packets because they were too small to contain an UDP header although they carried the UDP protocol.
  - TCP: too small length to contain TCP header
    - Number of not decapsulated packets because they were too small to contain a TCP header although they carried the TCP protocol.
  - GTP: too small length to contain GTP header
    - Number of packets that were not analyzed because they were too small to contain a GTP header.
  - GTP control: number of packets
    - Number of packets that carry GTP-C traffic.
  - GTP user: number of packets
    - Number of packets that carry GTP-U traffic.
  - GTP: error getting GTP optional and extension headers size (packet truncated?)
    - Number of GTP packets that were skipped because they were too small to be analyzed correctly.
  - GTP user: number of packets that carry user data
    - Number of GTP packets that carry user traffic.
  - GTP user: number of packets that carry no data PDU
    - Number of GTP packets that carry GTP-U but no user traffic.
  - GTP user: number of packets that experienced error (in GTP or IP headers)
    - Number of GTP-U packets that were not decapsuated because they contain some error in the GTP or IP header. E.g. they were IP fragments.
  - GTP user: number of packets that carry user data analyzed by TAM
    - Number of GTP-U packets that were decapsuated and analyzed by TAM.
  - Skipped encapsulated pkts (GTP: too small length to contain IP header)
    - Number of GTP packets that were not decapsulated because they were too small to contain an IP header.
  - GTP user: number of identified packets
    - Number of data packets that were identified on the Gn interface.
  - GTP control: error locating IE
    - Number of GTP packets that TAM could not locate the IE.
  - GTP control: error packet does not contain Sequence Number
    - Number of GTP packets that were not analyzed because they did not contain Sequence Number. Based on the specification all control packets should carry this field.
  - GTP control: packet contains message that TAM does not analyze
    - Number of GTP packets that were not analyzed because the program by definition does not analyze them. E.g. Notification Request.
• GTP control: packet is too small to contain IE
  ◦ Number of GTP packets that were not analyzed because they were too small to contain IE.
• GTP control: end of packet reached and IE not found
  ◦ Number of GTP packets that were searched to the end of the packet but the requesting IE was not found.
• GTP Delete Request: number of packets
  ◦ Number of Delete Request GTP packets
• GTP Delete Request: packet contains Teardown IE
  ◦ Number of Delete Request GTP packets that contain the Tear-down IE used to delete all active PDP contexts of a user.
• GTP Delete Request: error packet does not contain NSAPI IE
  ◦ Number of Delete Request GTP packets that were not analyzed because they did not contain the NSAPI IE. The IE is mandatory.
• GTP Delete Request: entry does not exist
  ◦ Number of Delete Request GTP packets that refer to an entry that does not exist in TAM.
• GTP Delete Response: number of packets
  ◦ Number of Delete Response GTP packets
• GTP Delete Response: error packet does not contain CAUSE IE
  ◦ Number of Delete Response GTP packets that do not contain the CAUSE IE. The IE is mandatory.
• GTP Delete Response: cause is not accepted
  ◦ Number of Delete Response GTP packets where the CAUSE IE value is other than Accepted.
• GTP Delete Response: deleting entry
  ◦ Number TAM entries that were deleted via Delete Response GTP packets.
• GTP Delete Response: entry does not exist
  ◦ Number of Delete Response GTP packets that refer to an entry that does not exist in TAM.
• GTP Create Request: number of packets
  ◦ Number of Create Request GTP packets
• GTP Create Request(P): error packet does not contain IMSI IE
  ◦ Number of Primary Create Request GTP packets that were not analyzed because they did not contain the IMSI IE. The IE is mandatory.
• GTP Create Request(P): error packet does not contain TEIDATA IE
  ◦ Number of Primary Create Request GTP packets that were not analyzed because they did not contain the TEIDATA IE. The IE is mandatory.
• GTP Create Request(P): error packet does not contain TEICON- TROL IE
- Number of Primary Create Request GTP packets that were not analyzed because they did not contain the TEICONTROL IE. The IE is mandatory.
  - GTP Create Request(P): error packet does not contain NSAPI IE
    - Number of Primary Create Request GTP packets that were not analyzed because they did not contain the NSAPI IE. The IE is mandatory.
  - GTP Create Request(P): error packet does not contain EUA IE
    - Number of Primary Create Request GTP packets that were not analyzed because they did not contain the EUA IE. The IE is mandatory.
  - GTP Create Request(P): error EUA IE too small
    - Number of Primary Create Request GTP packets that were not analyzed because they were too small to contain the EUA IE. The IE is mandatory.
  - GTP Create Request(P): info EUA IE is empty
    - Number of Primary Create Request GTP packets where the EUA IE is empty. It is expected to be empty when the user is to be assigned a dynamic network address.
  - GTP Create Request(P): info EUA IE is not IPv4
    - Number of Primary Create Request GTP packets where the EUA IE is not an IPv4 address.
  - GTP Create Request(P): error packet does not contain APN IE
    - Number of Primary Create Request GTP packets that were not analyzed because they did not contain the APN IE. The IE is mandatory.
  - GTP Create Request(P): error APN IE too small
    - Number of Primary Create Request GTP packets that were not analyzed because they were too small to contain the APN IE. The IE is mandatory.
  - GTP Create Request(P): error packet does not contain GSN Address IE
    - Number of Primary Create Request GTP packets that were not analyzed because they did not contain the GSN Address IE. The IE is mandatory and it exists 2 times in the header.
  - GTP Create Request(P): error GSN Address IE too small
    - Number of Primary Create Request GTP packets that were not analyzed because they were too small to contain the GSN Address IE. The IE is mandatory.
  - GTP Create Request(P): error GSN Address IE is not IPv4
    - Number of Primary Create Request GTP packets that were not analyzed because the GSN Address IE was not an IPv4 address.
  - GTP Create Request(P): error MSISDN IE too small
    - Number of Primary Create Request GTP packets that were not analyzed because they were too small to contain the MSISDN
IE. The IE is not mandatory but if exists in the packet TAM analyzes it. If some error is encountered the packet is not analyzed.

- **GTP Create Request(P): adding entry**
  - Number of times a new entry is added in TAM, based on information carried in Primary Create Request GTP packets.

- **GTP Create Request(P): entry already exists**
  - Number of times the Primary Create Request entry was not inserted in TAM, because it already existed.

- **GTP Create Request(S): number of packets**
  - Number of Secondary Create Request GTP packets.

- **GTP Create Request(S): error packet does not contain TEIDATA IE**
  - Number of Secondary Create Request GTP packets that were not analyzed because they did not contain the TEIDATA IE. The IE is mandatory.

- **GTP Create Request(S): error packet does not contain NSAPI IE**
  - Number of Secondary Create Request GTP packets that were not analyzed because they did not contain the NSAPI IE. The IE is mandatory.

- **GTP Create Request(S): error packet does not contain LNSAPI IE**
  - Number of Secondary Create Request GTP packets that were not analyzed because they did not contain the LNSAPI IE. The IE is mandatory.

- **GTP Create Request(S): error packet does not contain GSN Address IE**
  - Number of Secondary Create Request GTP packets that were not analyzed because they did not contain the GSN Address IE. The IE is mandatory and it exists 2 times in the header.

- **GTP Create Request(S): error GSN Address IE too small**
  - Number of Secondary Create Request GTP packets that were not analyzed because they were too small to contain the GSN Address IE. The IE is mandatory.

- **GTP Create Request(S): error GSN Address IE is not IPv4**
  - Number of Secondary Create Request GTP packets that were not analyzed because the GSN Address IE was not an IPv4 address.

- **GTP Create Request(S): adding entry:**
  - Number of times a new entry is added in TAM, based on information carried in Secondary Create Request GTP packets.

- **GTP Create Request(S): entry already exists**
  - Number of times the Secondary Create Request entry was not inserted in TAM, because it already existed.

- **GTP Create Request(S): relative error entry does not exist**
  - Number of Secondary Create Request GTP packets that refer to an entry that does not exist in TAM.

- **GTP Create Response: number of packets**
  - Number of Create Response GTP packets.

- **GTP Create Response: error Seq Num is zero**
- Number of Create Response GTP packets that were not analyzed because the Seq Num was zero. The value should not be encountered according to TAM limitations.
- GTP Create Response: error packet does not contain CAUSE IE
  - Number of Create Response GTP packets that were not analyzed because they did not contain the CAUSE IE. The IE is mandatory.
- GTP Create Response: cause is not accepted
  - Number of Create Response GTP packets that were not analyzed because the CAUSE IE value was other than Accepted.
- GTP Create Response: error packet does not contain TEIDATA IE
  - Number of Create Response GTP packets that were not analyzed because they did not contain the TEIDATA IE. The IE is mandatory.
- GTP Create Response: info packet does not contain TEICONTROL IE
  - Number of Create Response GTP packets that did not contain the TEICONTROL IE.
- GTP Create Response: info packet does not contain EUA IE
  - Number of Create Response GTP packets that did not contain the EUA IE.
- GTP Create Response: error EUA IE too small
  - Number of Create Response GTP packets that were not analyzed because they were too small to contain the EUA IE. The IE is mandatory.
- GTP Create Response: info EUA IE is empty
  - Number of Create Response GTP packets where the EUA IE is empty.
- GTP Create Response: info EUA IE is not IPv4
  - Number of Create Response GTP packets where the EUA IE is not an IPv4 address.
- GTP Create Response: error packet does not contain GSN Address IE
  - Number of Create Response GTP packets that were not analyzed because they did not contain the GSN Address IE. The IE is mandatory and it exists 2 times in the header.
- GTP Create Response: error GSN Address IE too small
  - Number of Create Response GTP packets that were not analyzed because they were too small to contain the GSN Address IE. The IE is mandatory.
- GTP Create Response: error GSN Address IE is not IPv4
  - Number of Create Response GTP packets that were not analyzed because the GSN Address IE was not an IPv4 address.
- GTP Create Response: error relative entry does not exist
  - Number of Create Response GTP packets that refer to an entry that does not exist in TAM.
• GTP Update Request: number of packets
  ◦ Number of Update Request GTP packets
• GTP Update Request: error TEID is zero
  ◦ Number of Update Request GTP packets that were not analyzed because the TEID value was zero. The TEID value zero is used in order to change GTP version.
• GTP Update Request: source SGSN
  ◦ Number of Update Request GTP packets that come from an SGSN
• GTP Update Request: source GGSN
  ◦ Number of Update Request GTP packets that come from an GGSN
• GTP Update Request: error packet does not contain NSAPI IE
  ◦ Number of Update Request GTP packets that were not analyzed because they did not contain the NSAPI IE. The IE is mandatory.
• GTP Update Request: info packet does not contain EUA IE
  ◦ Number of Update Request GTP packets that did not contain the EUA IE.
• GTP Update Request: error EUA IE too small
  ◦ Number of Update Request GTP packets that were not analyzed because they were too small to contain the EUA IE.
• GTP Update Request: info EUA IE is empty
  ◦ Number of Update Request GTP packets where the EUA IE is empty.
• GTP Update Request: info EUA IE is not IPv4
  ◦ Number of Update Request GTP packets where the EUA IE is not an IPv4 address.
• GTP Update Request(source SGSN): error relative entry does not exist
  ◦ Number of Update Request GTP packets coming from an SGSN that refer to an entry that does not exist in TAM.
• GTP Update Request: error packet does not contain GSN Address IE
  ◦ Number of Update Request GTP packets that were not analyzed because they did not contain the GSN Address IE. The IE is mandatory and it exists 2 times in the header.
• GTP Update Request: error GSN Address IE too small
  ◦ Number of Update Request GTP packets that were not analyzed because they were too small to contain the GSN Address IE. The IE is mandatory.
• GTP Update Request: error GSN Address IE is not IPv4
  ◦ Number of Update Request GTP packets that were not analyzed because the GSN Address IE was not an IPv4 address.
• GTP Update Request(source GGSN): error relative entry does not exist
- Number of Update Request GTP packets coming from an GGSN that refer to an entry that does not exist in TAM.
- **GTP Update Response: number of packets**
  - Number of Update Response GTP packets.
- **GTP Update Response: error sequence number is zero**
  - Number of Update Response GTP packets that were not analyzed because the Seq Num was zero. The value should not be encountered according to TAM limitations.
- **GTP Update Response: source GGSN**
  - Number of Update Response GTP packets that come from an GGSN.
- **GTP Update Response: source SGSN**
  - Number of Update Response GTP packets that come from an SGSN.
- **GTP Update Response: error relative entry does not exist**
  - Number of Update Response GTP packets that refer to an entry that does not exist in TAM.
- **GTP Update Response: error packet does not contain CAUSE IE**
  - Number of Update Response GTP packets that were not analyzed because they did not contain the CAUSE IE. The IE is mandatory.
- **GTP Update Response: cause is not accepted**
  - Number of Update Response GTP packets that were not analyzed because the CAUSE IE value was other than Accepted.
- **GTP Update Response: deleting entry**
  - Number TAM entries that were deleted via Update Response GTP packets. It happens only when the PDP context is not existent in the other end or the Update set a new user address but it was not accepted.
- **GTP Update Response: error packet does not contain TEIDATA IE**
  - Number of Update Response GTP packets that were not analyzed because they did not contain the TEIDATA IE. The IE is mandatory.
- **GTP Update Response: error packet does not contain GSN Address IE**
  - Number of Update Response GTP packets that were not analyzed because they did not contain the GSN Address IE.
- **GTP Update Response: error GSN Address IE too small**
  - Number of Update Response GTP packets that were not analyzed because they were too small to contain the GSN Address IE. The IE is mandatory.
- **GTP Update Response: error GSN Address IE is not IPv4**
  - Number of Update Response GTP packets that were not analyzed because the GSN Address IE was not an IPv4 address.
- **GTP SGSN Context Response: number of packets**
  - Number of SGSN Context Response GTP packets.
- **GTP SGSN Context Response: error packet does not contain CAUSE IE**
  - Number of SGSN Context Response GTP packets that were not analyzed because they did not contain the CAUSE IE. The IE is mandatory.
- **GTP SGSN Context Response: cause is not accepted**
  - Number of SGSN Context Response GTP packets that were not analyzed because the CAUSE IE value was other than Accepted.
- **GTP SGSN Context Response: error packet does not contain IMSI IE**
  - Number of SGSN Context Response GTP packets that were not analyzed because they did not contain the IMSI IE.
- **GTP SGSN Context Response: error packet does not contain TEICONTROL IE:**
  - Number of SGSN Context Response GTP packets that were not analyzed because they did not contain the TEICONTROL IE.
- **GTP SGSN Context Response: error packet does not contain GSN Address IE:**
  - Number of SGSN Context Response GTP packets that were not analyzed because they did not contain the GSN Address IE.
- **GTP SGSN Context Response: error GSN Address IE too small**
  - Number of SGSN Context Response GTP packets that were not analyzed because they were too small to contain the GSN Address IE. The IE is mandatory.
- **GTP SGSN Context Response: error GSN Address IE is not IPv4**
  - Number of SGSN Context Response GTP packets that were not analyzed because the GSN Address IE was not an IPv4 address.
- **GTP SGSN Context Response: error PDP Context IE too small**
  - Number of SGSN Context Response GTP packets that were not analyzed because they were too small to contain the PDP Context IE.
- **GTP SGSN Context Response: info EUA IE is not IPv4**
  - Number of SGSN Context Response GTP packets where the EUA IE is not an IPv4 address.
- **GTP SGSN Context Response: adding entry**
  - Number of times a new entry is added in TAM, based on information carried in SGSN Context Response GTP packets.
- **GTP SGSN Context Response: entry already exists**
  - Number of times the SGSN Context Response entry was not inserted in TAM, because it already existed.
- **GTP SGSN Context Acknowledge: number of packets**
  - Number of SGSN Context Acknowledge GTP packets.
- **GTP SGSN Context Acknowledge: error sequence number is zero**
  - Number of SGSN Context Acknowledge GTP packets that were not analyzed because the Seq Num was zero. The value should not be encountered according to TAM limitations.
• GTP SGSN Context Acknowledge: error packet does not contain CAUSE IE:
  ◦ Number of SGSN Context Acknowledge GTP packets that were not analyzed because they did not contain the CAUSE IE. The IE is mandatory.
• GTP SGSN Context Acknowledge: cause is not accepted
  ◦ Number of SGSN Context Acknowledge GTP packets that were not analyzed because the CAUSE IE value was other than Accepted.
• GTP SGSN Context Acknowledge: deleting entry
  ◦ Number TAM entries that were deleted via SGSN Context Acknowledge GTP packets.
• GTP SGSN Context Acknowledge: error packet does not contain GSN Address IE
  ◦ Number of SGSN Context Acknowledge GTP packets that were not analyzed because they did not contain the GSN Address IE.
• GTP SGSN Context Acknowledge: error GSN Address IE too small
  ◦ Number of SGSN Context Acknowledge GTP packets that were not analyzed because they were too small to contain the GSN Address IE.
• GTP SGSN Context Acknowledge: error GSN Address IE is not IPv4
  ◦ Number of SGSN Context Acknowledge GTP packets that were not analyzed because the GSN Address IE was not an IPv4 address.
• GTP SGSN Context Acknowledge: error packet does not contain TEIDATA_II IE
  ◦ Number of SGSN Context Acknowledge GTP packets that were not analyzed because they did not contain the TEIDATA_II IE.
• GTP SGSN Context Acknowledge: error relative entry does not exist
  ◦ Number of SGSN Context Acknowledge GTP packets that refer to an entry that does not exist in TAM.

Category 3 - General information:
• The number of packets sent to TAM by libpcap
  ◦ Number of packets that TAM received for process.
• The number of packets processed by TAM
  ◦ Number of packets that TAM processed (excluding fragments etc).
• Time (seconds) of first packet processed
  ◦ Timestamp (seconds) of the first processed packet.
• Time (microseconds) of first packet processed
  ◦ Timestamp (microseconds) of the first processed packet.
• Time (seconds) of last packet processed
  ◦ Timestamp (seconds) of the last processed packet.
• Time (microseconds) of last packet processed
  ◦ Timestamp (microseconds) of the last processed packet.