CHALMERS

Assessment of IxLoad in a GGSN environment

Master of Science Thesis in the Programme Networks and Distributed Systems

FREDRIK LARSSON
AZADEH HOSSEINZAD AMIRKHIZI

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, June 2010

Assessment of IxLoad in a GGSN environment

FREDRIK LARSSON
AZADEH HOSSEINZAD AMIRKHIZI

Examiner: ALI SAHLESON

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2010

Abstract

Today's telecommunication world is developing toward an increasing demand for accessing the Internet, multimedia applications and IP technologies. As radio access technology evolves, the complexity of the core network increases and the need for intelligent verification procedures rises. The Gateway GPRS Support Node is an essential component of GPRS packet core network which acts as the gateway between the packet core and the external network. Testing such a product has become a non-trivial task. With an efficient test process, it is possible to reduce deployment cost and thoroughly validate the GGSN functionality, thus providing the customers with a higher quality product. IxLoad is a full-featured layer 4-7 test system that provides real-world traffic simulation which can be used for testing Gateway GPRS Support Node. This thesis is a thorough assessment of IxLoad as an advanced test tool in a GGSN environment. Furthermore, this document will provide the reader with an overview of packet core and GGSN functionality. Several test scenarios with IxLoad in a GGSN environment were developed in which the quality of IxLoad could be determined. The outcome of this thesis are several suggested solutions meant to increase and expand IxLoad capabilities.

Preface

This thesis was written for Chalmers University of Technology, Göteborg, Sweden as part of a Master Programme, Networks and Distributed Systems. The master thesis was performed by Fredrik Larsson and Azadeh Hosseinzad Amirkhizi at Ericsson AB, Göteborg.

Examiner for this thesis is adjunct Ali S Sahleson of the Department of Computer Science and Engineering at Chalmers University of Technology. The Ericsson supervisor is Rohit Guliani at System Test GGSN-MPG department at Lindholmen, Göteborg.

Hereby we would like to thank all of the people who have been involved in the making of this thesis, especially Per Österström, Rohit Guliani, Ulf Damberg, Lars Jonasson, Magnus Carlhammar, Peter Lithner and Johan Köpman for their help and support during this project.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

3GPP	3rd Generation Partnership Project
AAL5	ATM Adaption Layer type 5
APN	Access point name
ATM	Asynchronous Transfer Mode
BD	Billing Domain
CCA	Credit Control Answer
CCR	Credit Control Request
CDR	Charging Data Record
CGF	Charging Gateway Function
CS	Circuit switched
DCCA	Diameter Credit-Control Application
DCP	Dynamic Control Plane
DHCP	Dynamic Host Configuration Protocol
DIB	Dynamic Interface Behaviour
DiffServ	Differentiated services
DNS	Domain Name Server
DPI	Deep Packet Inspection
DSCP	DiffServ Code Point
DT	Design Test
DTI	Direct Tunnel Interface
DUT	Device under Test
EPC	Evolved Packet Core
FR	Frame Relay
FT	Function Test
FTP	File Transfer Protocol
Ga	Interface between GPRS support node and charging gateway
Gc	Interface between CGF and Gateway GPRS Support Node
GGSN	Gateway GPRS support node
Gi	TCP/IP based interface from GGSN to external data networks
G-MSC	Gateway MSC
Gn	Interface between GPRS Support Nodes within a PLMN
Gp	Interface between GPRS Support Nodes in different PLMNs
GPRS	General Packet radio service
GRE	Generic routing encapsulation
GSM	Global System for Mobile Communications
GTP	GPRS tunnelling protocol
GTP'	Enhanced GPRS Tunnelling Protocol
GTP-C	GPRS Tunnelling Protocol Control plane
GTP-U	GPRS Tunnelling Protocol User plane
Gx	Interface between PCRF and Gateway GPRS Support Node
Gy	Interface between OCS and Gateway GPRS Support Node
HDLC	High-Level Data Link Control
H-DPI	Heuristic analysis Deep Packet Inspection
HLR	Home Location Register
HPLMN	Home Public Land Mobile Network
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force

IGMP	Internet Group Management Protocol
IMSI	International mobile Subscriber Identity
IP	Internet protocol
IPTV	Internet Protocol television
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
L2F	Layer 2 Forwarding
L2TP	Layer 2 Tunnelling Protocol
LAC	L2TP Access Concentrator
LDAP	Lightweight Directory Access Protocol
LNS	L2TP Network Server
LTE	Long Term Evolution
MGCP	Media Gateway Control Protocol
MIB	Management Information Base
MLD	Multicast Listener Discovery
MME	Mobility Management Entity
MMS	Multimedia Messaging Service
MOS	Mean Opinion Score
MPG	Mobile Packet Gateway
MS	Mobile Station
MSC	Mobile Switching Center
MSISDN	Mobile Subscriber International ISDN Number
MTU	Maximum Transmitted Unit
NFT	Network Failure Threshold
NIV	Network Integration Verification
OCS	Online Charging System
OSP	Octet Stream Protocol
P2P	Peer-to-peer
PCRF	Policy and Charging Rules Function
PDF	Probability Density Function
PDCP	Packet Data Convergence Protocol
PDN-GW	Packet Data Network Gateway
PDP	Packet Data Protocol
PDU	Protocol Data Unit
POP3	Post Office Protocol
PPP	Point-to-Point Protocol
PPTP	Point-to-Point Tunnelling Protocol
PS	Packet switched
PSTN	Public switched telephone network
QoE	Quality of Experience
QoS	Quality of Service
RADIUS	Remote Authentication Dial In User Service
RLC	Radio Link Control
RNC	Radio Network Controller
RSVP	Resource reservation protocol
RTP	Real-time Transport Protocol
RTSP	Real-Time Streaming Protocol
SGSN	Serving GPRS support node
SGW	Serving Gateway
SIP	Session Initiation Protocol

SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SNDCP	Sub-Network Data Convergence Protocol
SPI	Shallow Packet Inspection
ST	System Test
TCP	Transmission Control Protocol
TEID	Tunnelling Endpoint ID
TFT	Traffic Flow Template
TOS	Type of Service
UDP	User Datagram Protocol
UE	User Equipment
UMTS	Universal Mobile Telecommunications System
WAP	Wireless Application Protocol
VLAN	Virtual LAN
VoIP	Voice over IP
VPLMN	Visited Public Land Mobile Network

1 Introduction

Today's telecommunication world is developing towards an increasing demand for accessing the Internet, multimedia applications and IP technologies. Subscribers are demanding to browse the web, access their e-mails, downloading and uploading audio and video files through their mobile phones. UMTS, HSPA and HSPA+ technology are trying to satisfy these requirements by achieving convergence between mobile environment and Internet services, providing subscribers with higher quality of service and data rate. The mobile packet core needs to be able to handle the vast increase of data volumes carried over the new radio access technologies. It has become an essential part in shaping the end user's quality of experience, with important factors like speed, availability and accurate charging.

Ericsson is a world leading company in the telecommunication business and provides global telecom services, communication networks and multimedia solutions. This company has always been involved in the standardization work in the 3rd Generation Partnership Project (3GPP), which has given them the possibility to be first on the market with new radio access technology. Facing the challenge of subscriber demand, the Ericsson Packet Core solutions are providing the capacity, functionality and performance needed. Gateway GPRS Support Node (GGSN) is a part of the packet core network, which acts as an interface between Mobile Station (MS) and Packet Data Network (PDN), such as the Internet or a corporate network. As radio access technology evolves, the complexity of the core network increases and the need for intelligent testing of GGSN rises.

Ixia is one of the biggest test system suppliers for IP-based infrastructure and services. One of Ixias product is called IxLoad, which mainly focuses on testing converged IP networks. IxLoad is a full-featured layer 4-7 test system that provides real-world traffic simulation for testing of voice, video, and data network components. [1] IxLoad can be used to implement a scalable testing of Gateway GPRS Support Nodes (GGSNs) by simulating Serving GPRS Support Nodes (SGSNs) and the subscribers User Equipment (UE).

1.1 Objective

The main objective of this document is to make a thorough assessment of IxLoad as an advanced test tool in a GGSN environment and find ways of how to improve the test tool in order to increase the quality of GGSN both now and in the future.

Furthermore, this document will provide the reader with an overview of GGSN and it goes further in to depth in specific areas where a high-level tool is needed in order to make proper test verification. Moreover, several GGSN specific test scenarios will be proposed in which the quality of IxLoad can be proven. The expected outcome is to first identify and propose new functionality which should be added to IxLoad in order to meet the expectations of a proper test tool in a GGSN environment. Second, identify the areas in which IxLoad can be expanded.

This thesis is not meant to be similar to the IxLoad user guide supplied with the client program. The detailed information about how to configure the proposed test cases that can be found in the user guide are not provided. The presented information about the test cases aims to display important functionality that a test tool should be able to deliver.

1.2 Problem description

Gateway GPRS Support Node (GGSN) is an essential component of GPRS packet core network which acts as the gateway between the packet core and the external network. The GGSN is responsible for sending user packets to outside IP-based networks and routing packets back to the mobile user [2]. GGSN has several functions, including packet inspection for detecting different types of traffic, which can be used for shaping the traffic under different network load conditions, it may also be in charge of policy control and charging of user data flows. Testing such a product has become a non-trivial task and increasing the functionality requires further structured test scenarios. With an efficient test process, it is possible to reduce deployment cost and thoroughly validate the GGSN functionality, thus providing the customers with a higher quality product.

The verification process of a GGSN product has been divided into four primary phases which are Design test, Function test, System test and Network Integration Verification:

- **Design Test (DT)** is known as white box testing, meaning that the designer knows the underlying structure in depth. This verification phase aims for finding module level bugs. Each software module needs to be tested and verified individually.

- **Function Test (FT)** is known as grey box testing where the testers have knowledge about the product internal structure. The main goal of this phase is to disclose functional level bugs and to confirm that the feature implementation meets the functional requirements as well as, ensuring that the added functionality can comply with already implemented functionality.

- **System Test (ST)** is the phase which is seen as a black box testing without being concerned about the underlying implementation. The main task of this phase is to make sure that the product operates as expected and verify that new functionality is working together with generated background traffic. They accomplish this by treating the product as customers do.

- **Network Integration Verification (NIV)** This way of testing is meant to verify the GGSN behaviour in a complete non-simulated network environment. This might be verifying new functionality with some specially produced background traffic, verifying conformance with different mobile vendors.

Tests developed in system test are constructed on requirement specifications, operator use cases and standards, while functional tests have their foundations about functional specification and architecture design. Furthermore, there is a gap between Function test and System test regarding the number of users in the test, System test is closer to a real world scenario with thousands of subscribers whereas a Function test's focus is on one user. This leads to a huge difference in the levels of testing.

IxLoad is a GGSN test tool which is able to simulate end user behaviour by sending stateful and realistic multiplay traffic [1]. The aim of this test tool in a GGSN environment would be to fill some of those gaps and find new ways of increasing the quality of GGSN. To achieve this goal, an assessment of the test tool has been the main focus of this document.

1.3 Structure of the thesis

Before going in to depth about making a thorough assessment of the test tool, it is needed to make the reader acquainted with the GGSN environment as well as introducing the test tool itself. This is followed by a presentation of the implemented test cases and a discussion about how to expand the usability of IxLoad in a GGSN environment.

Chapter 2 outlines the Universal Mobile Telecommunications System packet core network and elements, focusing on the Gateway GPRS Support Node, its interfaces and main functionalities. Also the GPRS Tunnelling Protocol and an overview of the session management in the packet core network is introduced.

IxLoad is presented as a test tool in a GGSN environment in chapter 3, with a general overview of the test tool main functionalities. Some of the distinct capabilities of IxLoad such as Subscriber modelling, Impairment and QoE Detective are explained further. Additionally the GTP implementation and applications that IxLoad supports are presented.

In chapter 4, several implemented GGSN specific test scenarios are presented. The assessment of the test tool capabilities and shortcomings are based on the achieved experience in this chapter.

Chapter 5 discuss how to expand IxLoad usability as a GGSN test tool, in order to simulate proper user scenarios, which is vital in the packet core network. The information presented in this chapter originates from the development of test scenarios and identified requirements for a GGSN test tool.

The future work of this thesis together with the evolution of IxLoad are analyzed in chapter 6.

Chapter 7 provides a summary of this thesis and conclusion.

2 The GGSN and its surrounding environment

This chapter will summarise the Universal Mobile Telecommunications System (UMTS) core network architecture and elements, with focus on GGSN functionality and responsibilities. Furthermore, GTP protocol, which is the primary communication protocol used between the two main elements of the core network, Gateway GPRS Support Node (GGSN) and Serving GPRS Support Node (SGSN), will be presented.

2.1 Core network architecture

UMTS refers to a set of third-generation (3G) mobile phone technologies and services. The UMTS core network connects the existing cellular radio network with Packet Data Network (PDN), by equipping the Global System for Mobile Communications (GSM) infrastructure with IP based packet switching capability. With this upgrade, it provides data rates up to 144kbit/s in macrocellular environments, up to 384kbit/s in microcellular environments and up to 2 Mbit/s in indoor orpicocellular environments [3]. Furthermore, several users are able to share the same air interface resources and it gives possibility to allocate uplink and downlink channel independently, leading to more efficient use of the radio channel.

UMTS predecessor GPRS was developed on top of the existing GSM network together with some new network elements, interface and protocols required to handle packet based traffic. The most important elements are the SGSN and the GGSN forming the GPRS backbone, where the SGSN interfaces toward the radio access network, and GGSN interfaces to the PDN. UMTS inherited the same architectural structure from GPRS and it uses W-CDMA as the underlying air interface. Figure 1 shows the UMTS architecture, elements and interfaces.

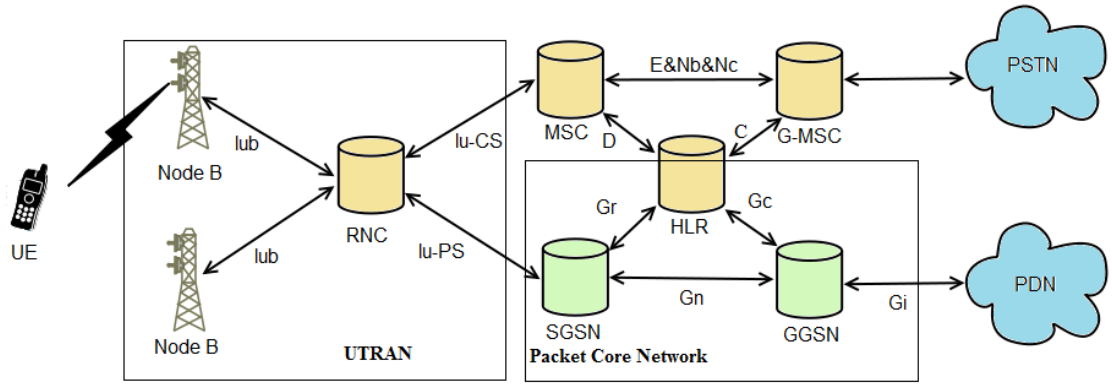

Figure 1: UMTS architecure [4]

The architecture is divided into two parts, the Circuit-Switched (CS) part and the Packet-Switched (PS) part, where the CS part connects phone calls toward the public switched telephone network (PSTN). The CS-part consists amongst other the Mobile Switching Center (MSC) and the Gateway-MSC (G-MSC). The PS most important nodes are SGSN and GGSN. The database Home Location Register (HLR) is a common database for both CS and PS. As it can be seen in Figure 1, the UTRAN connects to the MSC through the Iu_CS interface and to SGSN through Iu_PS interface.[4]

When a user activates its UMTS device, it will typically scan for an available channel and try to connect to the network. Through the various interfaces and nodes seen in Figure 1, the SGSN

will receive an “attach request” from the device and will try to authenticate the user by getting the user’s profile information from the HLR. From the profile information and the attach request message, the SGSN derives the Access Point Name (APN), which identifies the PDN and decides the correct GGSN to route towards. The SGSN will send a “create context” message towards the GGSN, which will associate the serving SGSN with the user. GGSN establishes the context and routes the user packets to the wanted PDN. When the return traffic arrives to the GGSN it will have the users IP address as destination, so the GGSN needs to associate the SGSN with the user and route the traffic accordingly. The SGSN forwards the traffic to the base station where mobile user is situated.

2.2 Serving GPRS Support Node

The SGSN acts as a mobile’s point of connection to the core network. Its main task is to make mobility of UMTS users possible. When a UE connects to the UMTS network a logical connection is made between the UE and its appropriate SGSN which allows UE to roam between different cells without losing its logical connection. When a UE receives packets from a PDN the SGSN keeps updated view of which Radio Network Controller (RNC) is usable. The behaviour is similar to regular IP router, but with some functionality added to handle radio network issues such as:

- Handling call control signalling with data services location registers.
- Interfaces to HLR, RNC and GGSN.
- Data encryption and compression.
- Radio resource management.
- User authentication and verification.
- Converting the actual user’s traffic between the core network and radio network.

A SGSN can only serve mobiles in a limited area which is referred to as the SGSN service area that can consist of one or many RNC areas. Inter SGSN delivery can be achieved when a UE moves from one SGSN service area to another. During the delivery the packets buffered in the old SGSN might be deleted and resent by using higher layers of the IP stack. When a UE is communicating with a server on the Internet, most of the data losses generally take place over the radio link, and taking care of the retransmission with higher layer protocol like TCP would be inefficient and slow. For that reason a quick retransmission protocol has been designed that only covers the wireless part and hides the loss from higher layer protocols. [5]

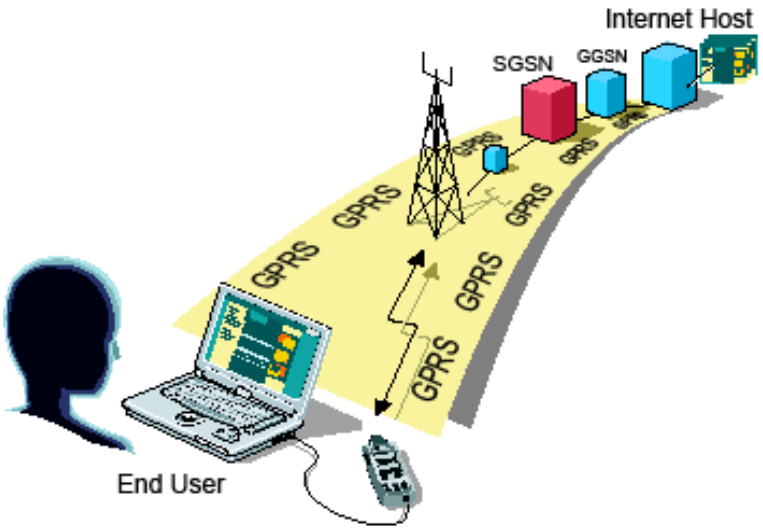

Figure 2: End-users road towards an Internet host

2.3 Gateway GPRS Support Node

The Gateway GPRS Support Node (GGSN) is a core network point of contact to the external PDN such as the Internet. GGSN is linked to the SGSN through an IP based backbone. It forwards packets from the PDN to the MS' current point of attachment which is SGSN that has routing information of the user. It also routes packets destined to the external PDN. From an external IP network, GGSN is seen as a common IP router which serves as the gateway between the wireless network and the data network.

The GGSN interfaces have been depicted in Figure 3. The Gi interface is the connecting point between GGSN and external PDN. Gn is an interface between GGSN and SGSN. Ga is an interface between GGSN and Charging Gateway Function (CGF). Gp is the interface between the GGSN of the Home Public Land Mobile Network (HPLMN) and the SGSN in the Visited Public Land Mobile Network (VPLMN). Gc is the interface between the HLR and the GGSN. Gx and Gy are interfaces between GGSN on one side and Policy and Charging Rules Function (PCRF) and Online Charging System (OCS) on the other side.

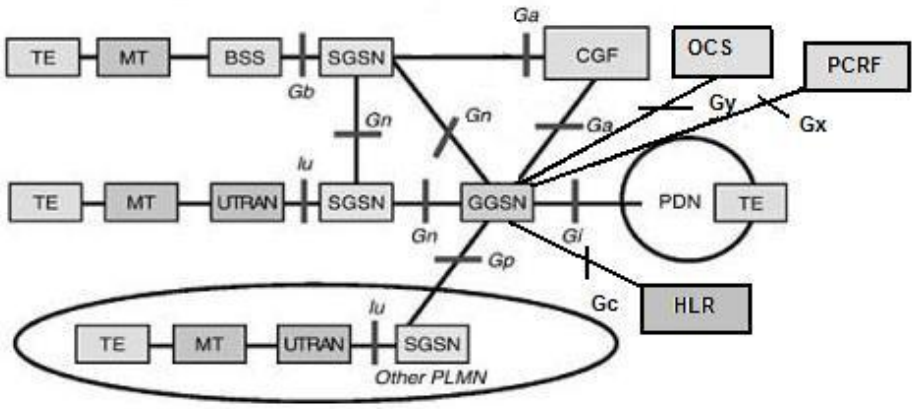

Figure 3: GGSN Interfaces [2]

Gi interface: The user data packets are delivered from the external PDN to the GGSN through this interface. The standard used for physical layer and data link layer can be determined by mutual agreement between the operator and the external ISP. Moreover, an extended set of Internet Engineering Task Force (IETF) standardised routing protocols are available in GGSN. Specific stateless filters and policies might be applied on Gi interface traffic. Both IPv6 and IPv4 are supported on the Gi interface. GGSN can perform encapsulation of user traffic on the Gi interface into various tunnelling and security protocols, such as Layer 2 Tunnelling Protocol (L2TP) and IPSec. Since GGSN has IP router functionality it is also able to perform various router functions, such as Ethernet Virtual Local Area Network, IPv6 tunnelling and IP-over-IP-tunnelling.

Gn interface: Control signalling for mobility and session management as well as tunnelling of user payload are performed through this interface between GSN nodes GGSN and SGSN. Before user data protocol can be tunnelled to SGSN through Gn interface, it should be encapsulated into the GPRS Tunnelling Protocol (GTP).

2.4 GPRS Tunneling Protocol

One of the main fundamentals of GGSN functionality is the support for GPRS Tunneling Protocol (GTP) which includes both GTP signalling (GTP-C) on the control plane and data transfer (GTP-U) to tunnel user data on the user plane. A deviation of GTP is the GTP' (*GTP prime*), which uses the same message format, but is used to carry charging data between GSNs and the charging gateway function (CGF). [6]

GTP is responsible for encapsulating and transmitting Protocol Data Units (PDUs) between SGSN and GGSN. To specify which tunnel a particuar PDU belongs to, GTP uses Tunneling Endpoint ID (TEID) in its header. This enables packets from the same UE to be multiplexed and demultiplexed by GTP between the two GPRS support nodes. [6]

The TEID value is assinged during Packet Data Protocol (PDP) context establishment which takes place on GTP control plane. PDP context activation creates a bijective GTP tunnel between the GSN pairs to transfer encapsulated user data packets between UE and external PDN. This process results in the establishment of a tunnel between the MS and SGSN, through the UTRAN, that is serving the UE by transferring the PDP data between the SGSN and MS. Figure 4 depicts the tunnelling procedures and protocol within UMTS backbone and radio network. [6]

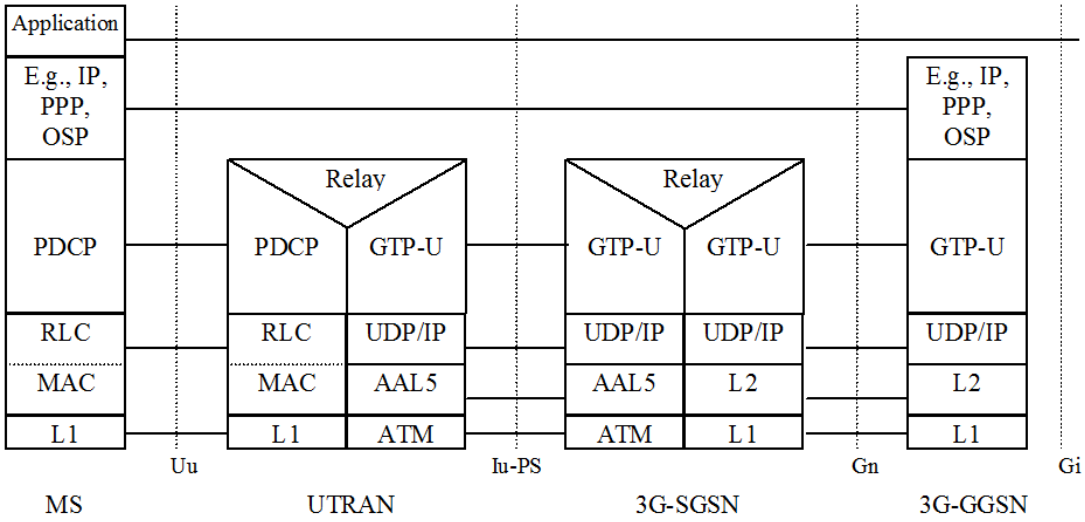

Figure 4: UMTS transport protocol layer

When the mobile user is communicating with a host on the PDN, the users IP traffic is tunnelled on top of the radio network through the UTRAN towards the SGSN, where the SGSN decapsulated the underlying tunnelling protocol and encapsulates the user plane packet in GTP messages on the Gn interface to be able to route multi-protocol packets towards the GGSN. GGSN decapsulates the user plane traffic and extracts routing information and based on the information, the GGSN will send it towards the wanted PDN. [6]

2.5 Session Management

When a MS wants to exchange data with an external PDN, it first needs to perform a successful GPRS attach, then it needs to apply for an IP address to use in the sought PDN. This address is referred to as the PDP address. A PDP context is created for each session initiated, which contains the desired characteristics of the specific session, including the PDP Type (IPv4 or IPv6), demanded Quality of Service (QoS), the assigned PDP address to the MS and the GGSN address serving as gateway towards the PDN. This context is saved in the GGSN, SGSN and MS. When a MS has an active PDP context it is available towards the external PDN and is able to send and receive data. The GGSN is able to transfer data between the PDN and MS, because of the mapping done between the PDP address and the International Mobile Subscriber Identity (IMSI) received from the HLR. The IMSI is a unique number associated with each GSM and UMTS mobile phone user, stored in the SIM card inside the phone. [2][7]

A mobile user can have multiple simultaneous PDP contexts active during the same time. The allocation of the PDP address can either be static or dynamic. For the first case, the network administrator of the HPLMN has permanently assigned a PDP address to the user. For dynamic PDP address allocation, a PDP address can be assigned to the user when “PDP context activation” is requested.

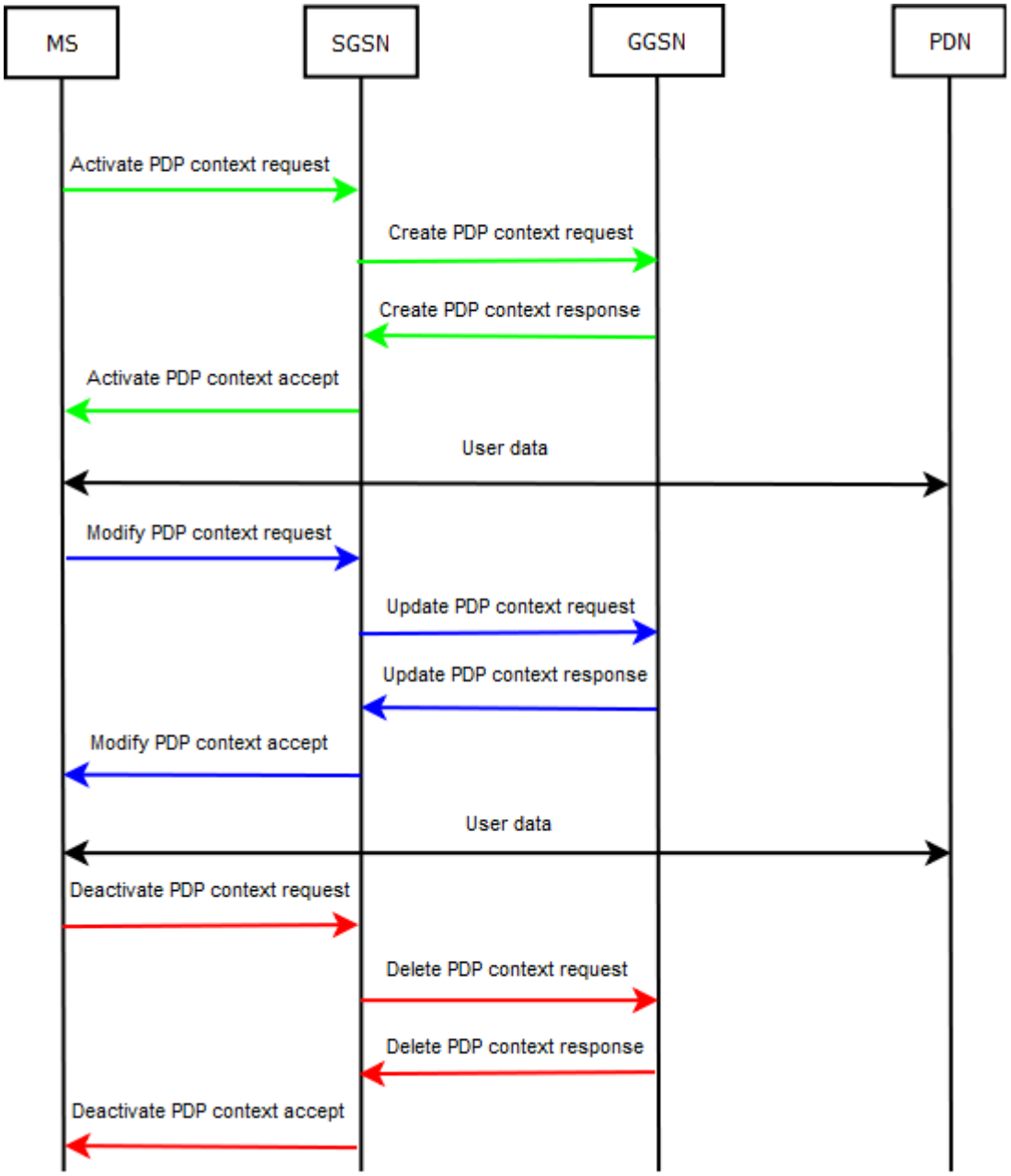

Figure 5: PDP context activation/modification/deactivation

Figure 5 depicts the different PDP context procedures. When the MS wants to connect to its desired PDN, it sends an "Activate PDP context request" to the SGSN specifying which APN to connect through. The PDP address field should be left empty if dynamic PDP address allocation is demanded. The SGSN then performs its regular tasks such as user authentication and authorization. If access is granted, the SGSN will send a "Create PDP context request" message to a specific GGSN where the desired APN is present. A new entry in the GGSN PDP mapping table will be created to make GGSN capable of routing packets between SGSN and PDN. After this step GGSN replies with "Create PDP context response" to SGSN including the PDP address if dynamic PDP address has been requested.

By sending an "Activate PDP context accept" message, SGSN updates its PDP context table and approves the activation of the new PDP context to MS. At this stage, the PDP context is created and user traffic may flow freely between the MS and PDN. A GGSN may do a Network-Requested PDP Context Activation in order to activate a PDP context. To support this procedure, the GGSN needs to have static PDP information about the PDP address, additional routing information requested towards the HLR.

Upon a “Create PDP context request” GTP-C message, the GGSN may respond either with a successful activation, including the cause code “Request Accept” and details of the PDP context or it will give a cause code, stating the reason of failure. Various cause codes exists, such as “No resources available”, which states, for example, that there is no memory available or that all dynamic PDP addresses are occupied, “Missing or unknown APN”, which states that the GGSN does not support the specified APN, and “User authentication failed” which indicates that the PDN has rejected the service requested by the user.

To modify PDP context parameters for instance QoS, Radio priority or Packet Flow Id during an active PDP context, the MS is able to send “Modify PDP context request” message to the SGSN. Correspondingly, SGSN sends “Update PDP context request” message to GGSN and if both GGSN and SGSN complies with the requested update they will reply with “Update PDP context response” and “Modify PDP context accept” back.

Furthermore, an “Update PDP Context Request” shall be sent from SGSN to a GGSN as a part of the GPRS Inter-SGSN Routing Update procedure or to redistribute contexts due to load sharing. Also, it shall be mentioned that GGSN and SGSN are both able to initiate modification of parameters. A GGSN can ask for modification by sending an “Update PDP Context Request” to the SGSN and the SGSN can ask for it by sending a “Modify PDP Context Request” message to the UE. [2]

The PDP context can be deleted by sending “Deactivate PDP context request” message from the UE to SGSN. As a result, SGSN will send a “Delete PDP context request” message to GGSN and both GSN nodes will confirm the PDP context deletion [6]. Deactivation procedures can be initiated by the MS, SGSN and GGSN. [2]

To gain a different QoS from what is provided in the primary context it is also possible to activate a secondary PDP context. The same PDP address and APN as the primary PDP context will be used for the secondary context. The MS defines what services shall be carried over the secondary context by the use of a Traffic Flow Template (TFT). The TFT is used by the GGSN as a distinguisher amongst the shared PDP contexts. It is a list of packet filters unique for each PDP context including parameters such as QoS, PDP context and security. [7]

The PDP address can be assigned by the operator of the subscriber’s HPLMN or by the operator of the VPLMN. The other possible case is to assign a permanent or dynamic IP address to the MS by external PDN operator or administrator. The GGSN is responsible for the address allocation and the subsequent activation and deactivation of the PDP in the case of using dynamic addresses from VPLMN or HPLMN. [2]

During the external address allocation, the PLMN may gain a PDP address from the PDN and deliver it to the MS during PDP context activation. It is also possible that the MS exchange PDP address with PDN directly after PDP context activation. In the former case that PLMN is providing the MS with PDP addresses, GGSN and PDN is in charge of assigning and releasing

the dynamic address, by use of DHCP or RADIUS protocols as well as using local pool PDP address allocation. When using DHCP, the GGSN acts as a DHCP client, when using RADIUS the GGSN acts as an RADIUS client and when using local pool the GGSN acts as a DHCP server. In the latter case, when MS exchanges PDP address directly with the PDN, the MS and the PDN are in charge to assign and release the PDP address using DHCP or Mobile IP (MIP) protocols. When DHCP is in use, the GGSN acts as a DHCP Relay Agent and when MIP is in use GGSN acts as a Foreign Agent. [2]

IPv6 dynamic address allocation has a different procedure compared to IPv4 address allocation. In this procedure, there are two different ways of allocating addresses, stateless and stateful autoconfiguration. In the stateful autoconfiguration a DHCP server is required to assign the addresses for the IPv6 MS. The stateless procedure is slightly more complicated, because the MS is more active in the process. On the other hand, the advantage of using the stateless autoconfiguration is that there is no requirement for external entity such as DHCP server and RADIUS server.

2.6 GGSN functionality

In every business, there is always a need for income, in the GPRS world, this comes from charging and the GGSN ability to identify the traffic by performing packet inspection and service classification and report the results to the charging system. Users are offered an enormous amount of services in this area and it is up to the operators to come up with different ways of billing the users for consuming their services.

Some of GGSN main functionality can be divided into following dimensions:

- **Interconnection to external networks**
- **Charging**
- **QoS and QoE handling**
- **Traffic shaping**
- **Packet Inspection and Service Classification**
- **Direct Tunnel**
- **Layer 2 Tunnelling Protocol**

2.6.1 Interconnection to external networks

GGSN performs signalling for creating, modifying and deleting PDP contexts to the packet core network. It also executes signalling to configure the PDP context toward PDN for example to allocate the IP addresses for the UE. Normally this signalling occurs after PDP context creation.

2.6.2 Charging

As mentioned in the beginning of this chapter, GGSN needs to be able to analyse the user-traffic and report the measured usage to the charging system. For reporting, GGSN has support for both of the two standardized charging mechanisms, offline charging and online charging.

2.6.2.1 Offline charging

For mobile telephony charging information there is a basic unit called Charging Data Record (CDR). A CDR may include fields specifying the subscriber, and information about the network resources and services used during the subscriber's session, especially the amount of transferred downlink and uplink user traffic and duration of the PDP context.

With offline charging, the collection of charging information is done concurrently with the network resource usage. This information is then transferred into a CDR that is then transferred to the network operator's Billing Domain (BD). The BDs purpose is to handle the subscribers billing, some inter-operator accounting. It is also possible for the operators to collect statistics from the BD. The transport protocol used when sending CDRs is a derivation of GTP, called GTP' (*GTP prime*). It is a separate protocol, based on the GTP-C reusing the same message structure. However GTP' includes a synchronization protocol in order not to lose or duplicate CDRs. It is used on the Ga interface between the GSN and CGF. The term offline charging means that the charging information does not influence the user or its service until the bill arrives. This methodology is also called post-paid. [8]

2.6.2.2 Online charging

Pre-paid subscribers need another solution than offline charging, in which the charging information cannot influence the service rendered. The online charging is the key for solving that issue. In this solution an Online Charging System (OCS) is in charge of the credit control, it communicates with the GGSN on the Gy interface, through Diameter Credit-Control Application (DCCA) with the two Diameter messages, the CCR (Credit Control Request) and the CCA (Credit Control Answer). The credit metrics can be volume, time or a specially defined unit indicating the usage. For quota management the GGSN sends a CCR to the OCS requesting units for the subscriber. The server grants the units and charges the subscribers account. It is also possible to do a price inquiry, where the GGSN asks the OCS what price a specific unit costs. An example of a traditional credit control session is shown in Figure 6. [9]

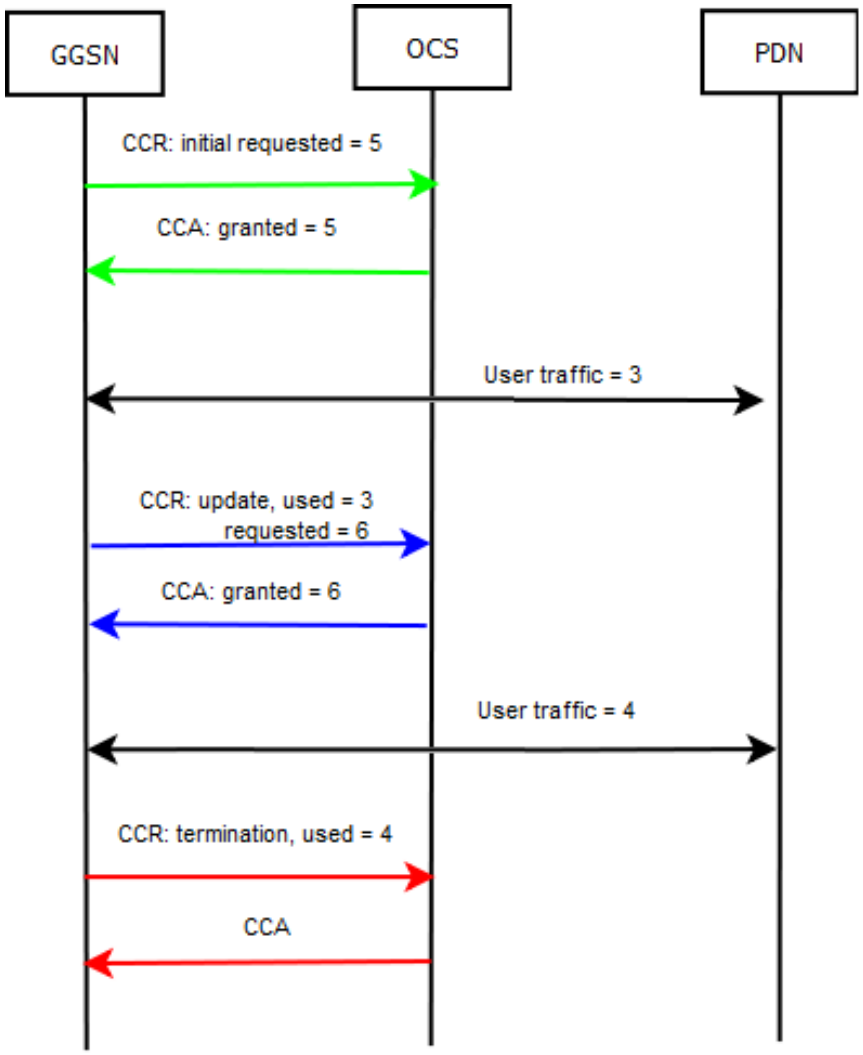

Figure 6: Credit control message overview

When a user has completed the PDP activation procedure and wishes to access a service, the GGSN shall request quota from the OCS before access is granted. The OCS controls the service delivery by reserving an amount of the user's available credit and granting the GGSN a quota of volume, time or units. If the user would have had insufficient balance in his account, the OCS may limit it or reject it completely. When a subscriber is about to use up all its granted units, The GGSN may send an update towards the OCS if for example the quota has been exhausted or reached a threshold level, the validity time has been expired, the PDP context has been updated etc. The OCS re-evaluates the user and if it has sufficient balance it grants the further requested quota. When the user is finished with its traffic it will send a PDP context delete request towards the GGSN, which initiates context deletion and sends a CCR termination towards the OSC reporting the last used units.

If a user does not have sufficient balance the OCS may request for redirection of the user traffic, for example if the user is using HTTP, the OCS may redirect the user towards the operator's site, where the subscriber may choose to refill its account. [9]

It is possible to accompany online charging with offline charging in order to generate CDRs for the online charged subscribers, giving a backup possibility if any of the systems fail. [8]

2.6.3 QoS, QoE and Traffic Shaping

As commonly known, traffic on the Internet is carried with best-effort, meaning that there is no guarantee that all the packets are going to be delivered through the network. Packet loss might occur for several reasons for example by network congestion or overloaded equipment. Increasing the bandwidth capacity might appear as a simple solution but the protocol that caused the congestion might eat up the extra bandwidth and the problem will appear again.

Different services and protocols have different requirement and features, FTP is one example which is not overly affected by jitter, packet loss or delay although bandwidth limitation might have impact on duration of downloading files but the final result will not be altered. On the other hand conversational real-time services, such as video and voice, are sensitive to the network behaviour. The conversations might sound choppy and changed when the voice packets have more than 150-200 ms delay.

It is also important to note that different users need different quality and thereby it is most likely that some customers are willing to pay more to get better services in order to get less failure in transmissions and fewer disconnections, while other customers prefer to pay less. [10] In such cases the QoS concept can be used to offer guaranteed services for mobile subscribers.

Differentiated services (DiffServ) are one of the better approaches for enabling the QoS concept in IP networks. DiffServ is a QoS mechanism relying on the static configuration of forwarding resources. [11] DiffServ networks categorizes packets into a small number of aggregated flows or “classes” based on the DiffServ Code Point (DSCP) in the IP header of the packets, this categorization is also known as behaviour aggregate classification. [12] Another solution is over-provisioning which is providing sufficient bandwidth to avoid congestion. In most cases the over-provisioning solution fulfils the QoS requirements for delay-tolerant traffic, such as file downloading and web browsing. However, more advanced solutions such as differentiated services are needed when the network load becomes higher or service assurances are offered by the operator. [13]

The QoS architecture is a composition of two approaches: The Internet Engineering Task Force (IETF) QoS development for fixed IP networks and the 3rd Generation Partnership Project (3GPP) QoS architecture designed for efficient use of scarce radio resources.

QoS classes, also referred to as traffic classes is defined in the 3GPP standard “Quality of service concepts and architecture” TS 23107 [14]. The four QoS classes are, in descending order of delay sensitivity:

- **Conversational**
  Intended for telephony speech, Voice over IP (VoIP), and video conference.
- **Streaming**
  Intended for real-time audio- and video-streaming applications
- **Interactive**
  Intended for interactive application such as interactive Email or interactive Web browsing, telnet, and those that require request responses.
- **Background**

Intended for Background traffic for example emails and file downloading, Short Message Service (SMS), file transfer, and such services that does not require delivery within a certain time

Where Conversational class is intended for traffic that is the most sensitive to delay and Background class is intended for traffic that has less sensitivity. The Interactive class and the Background class should give lower error rate by means of channel coding and retransmission in the radio channel compared to Conversational and streaming classes, due to the different delay tolerance requirements. Traffic in the Interactive class should have a higher priority in scheduling than Background class traffic which is important in wireless environments where the bandwidth is low. [14]

Integrating DiffServ in the GGSN is vital for it to be able to take care of IP packets with respect to their traffic class or during periods of congestion. The UEs requested QoS should be negotiated to assure the integrity of the GGSN and the adjacent nodes.

The GGSNs QoS procedures are triggered from signalling messages from the UE, the SGSN, or unsolicited provisioning from external policy control servers, such as the Policy and Charging Rules Function (PCRF). The specific QoS procedure depends on the GSNs capabilities as well as the UEs QoS capabilities. As it is stated in 3GPP standard TS 23207, a GGSN is required to support at least a DiffServ function. It may also support resource reservation protocol (RSVP) and policy enforcement functions. [15]

The model portrayed in Figure 7 shows an end-to-end perspective of the QoS offered by the network, including three different layers; network, terminal, and application layers.

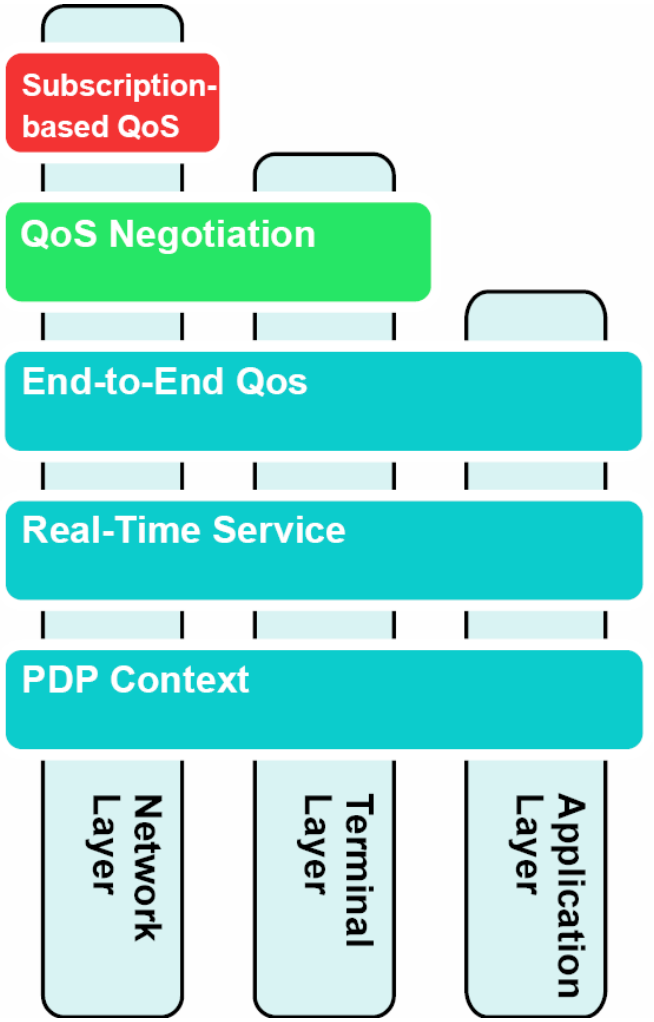

Figure 7: QoS Layer Model

The network layer handles subscription-based QoS, involving the radio and backbone networks, requiring that the network nodes, RNCs, NodeBs and the GSNs, negotiate QoS parameters with each other and forward packets accordingly. This subscription-based QoS can be controlled by the network operator.

The terminal layer involves the subscribers' MS in the QoS provisioning. The mobile stations that are supporting QoS can forward packets according to the negotiated QoS. Examples of this are packet marking and scheduling.

The application layer involves the application running on the MS which makes it possible for the application to actively take part in the QoS negotiation and provides the network with information about specific QoS requirements. The application is aware of what QoS received from the network and may attempt renegotiation. [16]

For an operator it is essential to have increased control over the QoS offered over the core and radio network resources. Having a proper quality for the end users without allocating unnecessary resources is essential for keeping the expenses at a lower bound. Through the Gx interface, operators' application servers are allowed to be more involved in the QoS negotiation

at PDP context activation and modification. This is enabled by the use of a PCRF providing service authorization and policy control for services accessed through the GGSN. The GGSN and the PCRF communicates in a client-server fashion over the Gx interface. The PCRF is supporting the GGSN with different charging rules that specifies which services that are authorized and which are not. The rules may explicitly tell the GGSN how to classify the users' service data flow. The PCRF can also additionally be responsible for assessing which charging rule to apply for the users' service based on the subscription type, user profile, if the MS is roaming or based on the time and date. Based on the configuration the GGSN may redirect the user depending on the response from the PCRF. [16]

Another essential issue that the operators need to focus on is the subscriber's service satisfaction. The cause of this satisfaction is not only the dealings with operator but also having good service experiences. Surfing the Internet, sending videos or photos and downloading ringtones are some examples of subscriber's service experiences. The measurement of how well the subscribers are satisfied with getting services is called "Quality of Experience" (QoE). [17]

In contrast with QoS which is a well-defined concept in software engineering, QoE is about human satisfaction and gratification when using the services. As described above QoS is an analysis of the network functionality conditions like jitter, packet loss or noise whereas QoE is an analysis of the end user satisfaction from current provided application and services. Although, QoS and QoE are two different concepts they are certainly related since the quality of service provided is one of the items affecting the subscriber satisfaction, but having a high QoS level does not necessarily leads to a high QoE level. For instance the operator might provide data services with high QoS but the subscriber is not satisfied with the content which leads to low QoE. [17]

Traffic shaping is a method to improve and guarantee performance and control the traffic on a network for example increasing usable bandwidth by delaying certain packets or controlling the volume being sent to the network or other actions on flows on the network. Traffic shaping is a way to guarantee that the network provides an expected level of QoE and user satisfaction. Today, there is an increasing demand for traffic shaping, because of the increasing data volumes transferred over the radio network and the operators' infrastructure may not be able to cope with it. This will force them to upgrade their capacity without having the corresponding revenue. In these situations certain traffic needs to be throttled in order to protect the infrastructure. If traffic shaping is successfully applied, it will achieve prioritization of important and low-tolerance traffic, management of high-consumption data services to ensure that certain users doesn't overload the system. Also it will make sure that the low-tolerance traffic will have decent traffic flow attributes such as good enough bandwidth, low jitter, delay and packet loss.

With the convergence to an All-IP-based solution, the variety of flow classification and resource reservation offered by DiffServ may not be enough. In order to meet the service delivery expectations of the subscribers, the core network needs to be able to distinguish between the types of services and also distinguish between different types of specific applications. In this Service-Based QoS, the bearer needs to interact further with the core network. The user profile is also becoming vital, since the services will be based on what subscription the user got and what privileged services the user has acquired. This will result in more PCRF-involvement, since the PCRF is responsible for the user profile enforcement, by informing the GGSN about the privileges of the users. [18]

2.6.4 Packet Inspection and Service Classification

Packet inspection and traffic analysis of the user plane traffic is one of the essential functionalities in GGSN. The user plane traffic normally includes Hypertext Transfer Protocol (HTTP), Wireless Application Protocol (WAP), Multimedia Messaging Service (MMS), Email and streaming voice or media with Session Initiation Protocol (SIP) and Real-Time Streaming Protocol (RTSP). The capacity to know what is running on the network is imperative for operators in their quest for offering new differentiated services to their customers as well as being able to guarantee their QoS.

Filtering and inspection of packets has been subjected to continuous development. Figure 8 depicts packet analysis which can be divided into 3 different levels, Shallow Packet Inspection (SPI), Deep Packet Inspection (DPI) and Heuristic analysis Deep Packet Inspection (H-DPI). SPI uses header information such as source and destination IP addresses, ports and the transport protocol for analysing and investigation of the packets, while DPI inspects packets headers as well as payload by using protocol based analysis on layer 7 such as finding out what sites the user is browsing by looking at the HTTP request URI in HTTP GET messages. [18] It provides application awareness, capability to analyse network usage and improving network performance. [19] As said by James Crashaw, research analyst, "DPI helps operations improve the performance of interactive applications, preventing certain application traffic from unduly hogging resources and contributing to congestion. It also mitigates the effects of network attacks, which helps reduce capex and opex while increasing subscriber satisfaction, leading to lower churn ". [19]

H-DPI, examines packet transaction based on the detected behavioural patterns. This makes it possible to even analyse traffic has been encrypted. Behavioural patterns are kept in a database which is used for matching packet flows and they contain observations of flow rate, typical length of packets, uplink/downlink packet rates, parallel connections etc. [18]

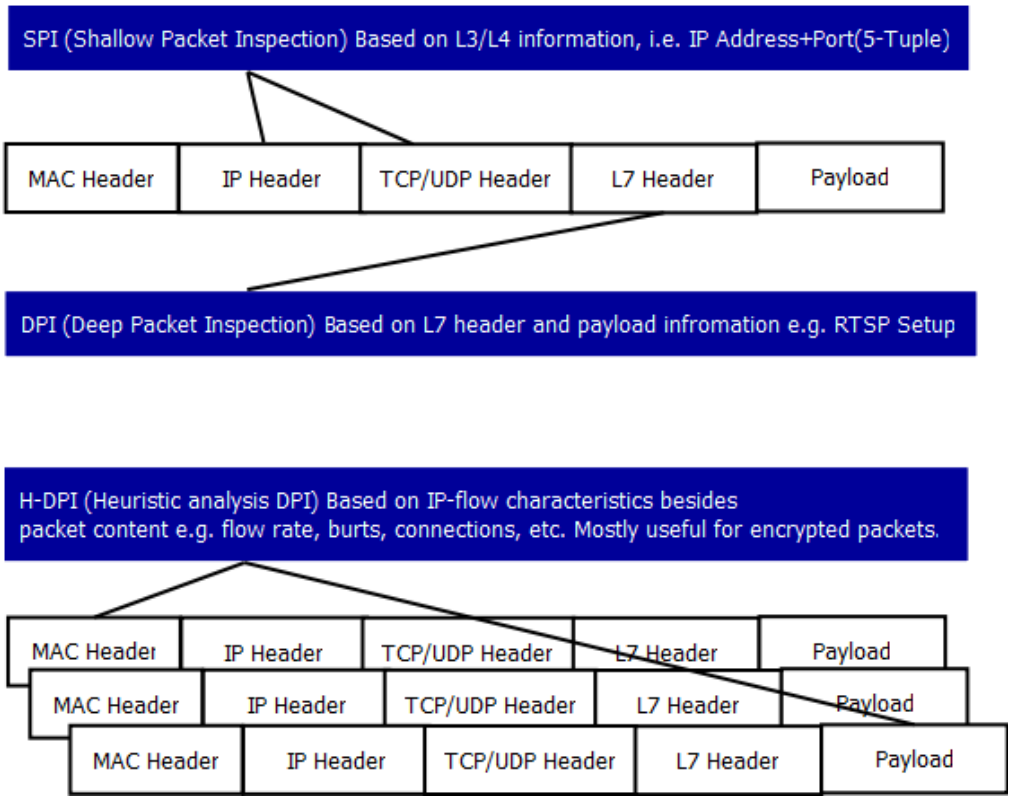

Figure 8: Packet Analysis Levels [17]

Managing and detecting peer-to-peer (P2P) traffic is more complicated since DPI is not quite adequate for protocols such as P2P as a result of some characteristics such as using dynamic port assignment, end to end encryption etc. Figure 9 shows comparison between P2P traffic and HTTP. [19]

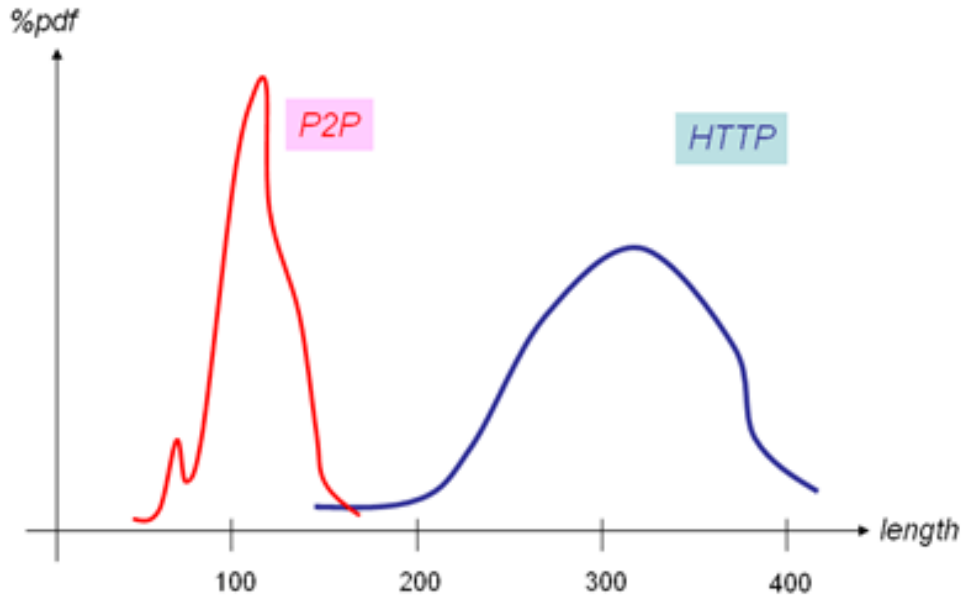

Figure 9: Patterns comparision between P2P and HTTP [18]

If only packet length histogram flow (PDF) is investigated, it is obvious that P2P control packets are likely to be shorter than HTTP packets particularly during setup time. If a heuristic analysis with the above knowledge is applied on an example where a P2P-application is using port 80, which is commonly associated with HTTP, it will be able to investigate the packet length histogram and determine if port 80 carries pure HTTP traffic or P2P related traffic. [18]

GGSN performs packet inspection and packet analysis in all 3 different levels that has been discussed. It can identify different types of traffic which is useable for traffic shaping purposes under different network load conditions. The GGSN DPI functionality starts investigating and analysing the traffic from network layer up to application layer. As the data being captured in real time all the content and packets in layer 3 are getting inspected by GGSN DPI rule base engine. In this way the operators have the ability to collect information about what the subscribers are experiencing about the provided services. Having such an ability improves the level of QoE and it leads to greater satisfaction as well as increased profitability. [20]

2.6.5 Direct Tunnel

In the 3G packet core architecture, the SGSN is responsible for tunnelling the signalling traffic and the user plane traffic from the radio network towards the GGSN. This traffic is first encapsulated between the Radio Network Controller (RNC) and the SGSN, which then the SGSN needs to terminate, extract the user plane traffic and put it into the tunnel towards the GGSN. This procedure requires both time and processing power. During most circumstances this is not really required, since both the GGSN and the RNC are regular IP routers, which are able to communicate directly to each other. One approach is the standardized Direct Tunnel Function, which enables the SGSN to establish a direct user plane tunnel between the RNC and the GGSN, thus eliminating itself from the user plane traffic. However the mobility management remains on the SGSN, meaning that it is still responsible to modify the tunnel if the UE is roaming to an area served by another RNC. [21] In this way, the 3G payload traffic is bypassing the SGSN, now only working as a signalling server, meaning that the dimensioning metric for a SGSN is the signalling load and not the user plane traffic. [22] There are some limitations for this feature, for example international roaming does not work if the SGSN needs to be involved in counting the traffic for inter-operator billing. An additional case is when the SGSN is in charge of counting the traffic flow, however in practice it is possible to work around such limitation with the use of OCS**.** [23] As can be seen in Figure 10, when initiating a DTI tunnel the MS sends its usual Activate PDP Context Request to the SGSN and a normal create PDP context procedure will occur. The SGSN then sends a RAB Assignment Request, ensuring that the RNC and MS is aware of that a tunnel is about to be created. After the RAB Assignment procedure the SGSN will inform the GGSN by sending an Update PDP Context request including DTI information informing which RNC that is going to send user plane traffic towards the GGSN. After the GGSN has responded to the SGSN, an Activate PDP Context Response is sent to the MS, thus enabling the MS to start sending the payload.

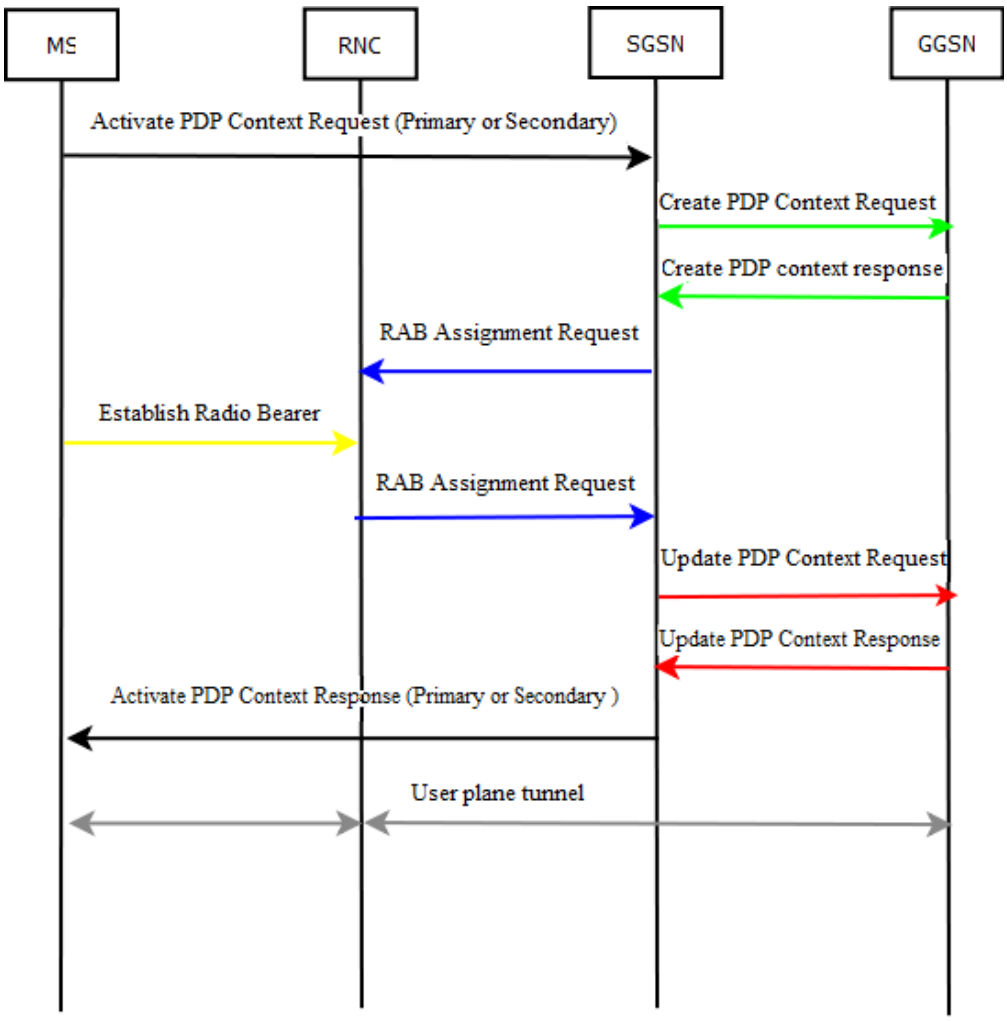

Figure 10: DTI PDP context activation procedure

2.6.6 Layer 2 Tunneling Protocol

Layer 2 Tunnelling Protocol (L2TP) is used to tunnel the user traffic through an insecure network, for example the Internet to the remote networks that can be a local area network or a corporate network. The main purpose for initiating L2TP tunnel in GGSN environment is to extend the point to point connection between UEs and LAC over an insecure network. Standardized L2TP (RFC 2661) is developed from different proprietary protocols such as layer 2 forwarding (L2F) and point-to-point tunnelling protocol (PPTP). [23] It supports PPP sessions, on top of UDP or on top of link layer protocols such as frame relay (FR) and ATM. L2TP operation is supported in GRPS and UMTS Core network by the Release'98 and Release '99 standards**.** Consequently, GGSN must support PPP-based PDP contexts. [24]

The L2TP tunnel is initiated between GGSN and another router, usually a corporate edge router as can be seen in Figure 11. During a L2TP scenario, mobile users are connecting to a specified APN for which the GGSN is creating PPP sessions. User packets are encapsulated and transmitted over these sessions. The GGSN will act as the L2TP Access Concentrator (LAC) as soon as the PDP context is activated. The GGSN will tear down the GTP tunnel and

use a L2TP tunnel to carry the PPP packets towards the corporate L2TP Network Server (LNS). The LNS performs authentication and will assign a corporate IP address to the mobile users [24]

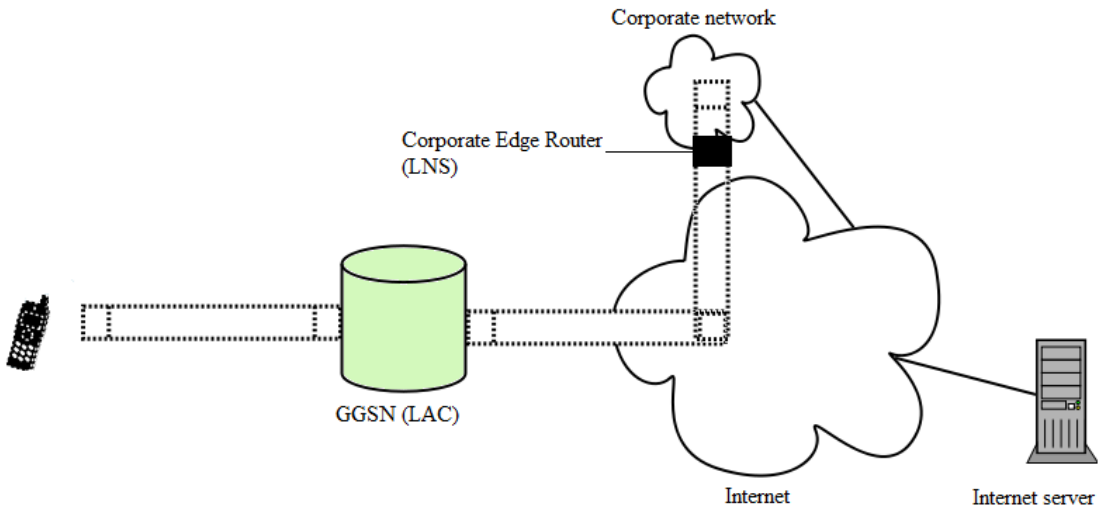

Figure 11: L2TP-functionality in a packet core network

One of the important advantages of using L2TP in UMTS/GPRS networks is that the network operators are able to transmit the UMTS/GPRS traffic through IP network to their customers in order to ensure that the mobile data is kept apart from the other traffic on the PDN. Another advantage by using L2TP is that the router on the other side of the tunnel which is in customer network handles the IP address allocation and user authentication for the mobile terminal instead of GGSN which is in the operators network. In this way, the external networks that want to be secure do not have to a trust third party for their users' authentications and IP address allocation, meaning that a mobile user belonging to the corporate organisation may get an IP address from the corporate network instead of an IP address provided by the GGSN. [25]

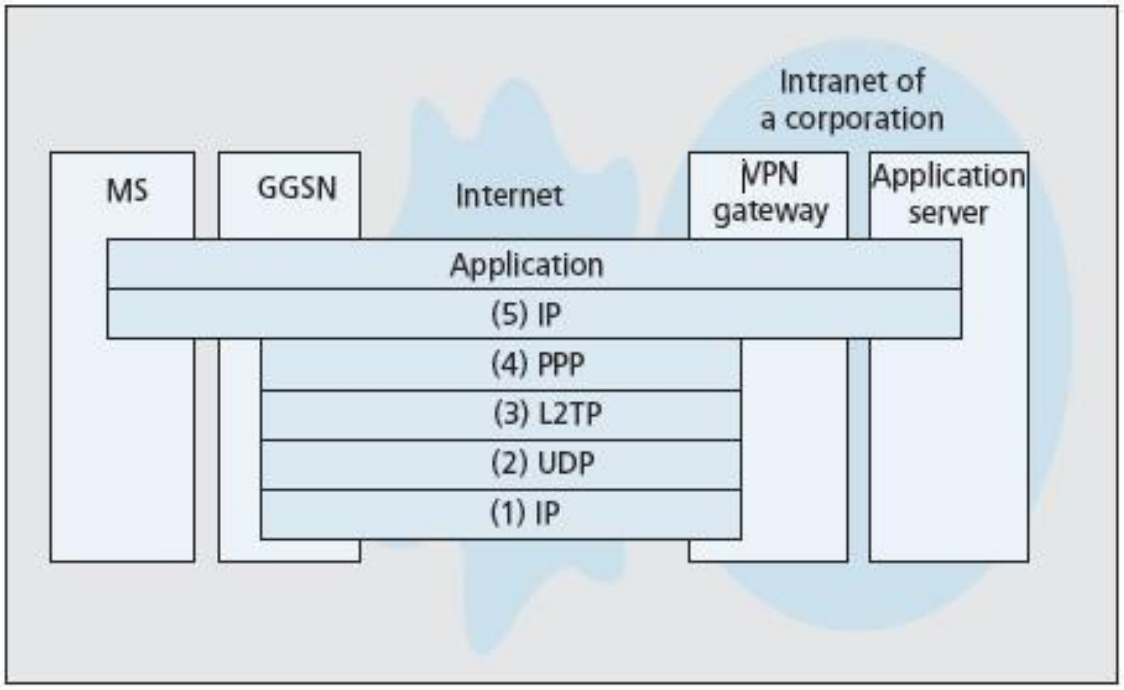

Figure 12: L2TP tunnelling [26]

Though, L2TP can offer secure CHAP-link authentication of L2TP control connection, it is not fully secure since tunnel hijacking is still possible. IPSec can be used on top of L2TP in order to provide a fully secure tunnel. [24]

L2TP is not the only tunnelling method, IP-in-IP tunnelling and Generic Routing Encapsulation (GRE) tunnelling are two methods that can be used for tunnelling in the UMTS. For IP-in-IP, the users' IP packets are encapsulates in tunnel IP packets and for GRE, tunnelling packets can carry the user packets with different protocols such as IP or PPP. Table 1 depicts features of the three possible tunnelling methods. All three approaches are usable for MS which supports both PPP and IP. Moreover, if the user application only supports PPP, GRE tunnelling is the best choice but if link layer protocol is FR or ATM, then L2TP must be selected. However, as a result of short protocol overhead IP-in-IP tunnelling has better efficiency. [26]

Tunnelling method	Overhead	Transport network protocol support	MS protocol support
IP-in-IP	Low	IP	IP
GRE	Medium	IP	PPP
L2TP	High	IP/UDP, FR, ATM	IP

Table 1: Tunneling methods

3 IxLoad

This chapter will introduce IxLoad as a GGSN test tool and give a short summary of its general capabilities. Also, short overview of the following subjects will be presented:

- Hardware structure of IxLoad
- Subscriber modelling
- Impairment
- Quality of Experience
- IxLoad Statistics
- IxLoad Analyzer
- Application supported over GTP

3.1 General overview

The Internet is about to take its second step, from just giving local and global connection that used regular address and port-based routing into complex application delivery where the infrastructure must identify, prioritize and handle traffic with different QoS demands. Such application-aware infrastructure is allowing the businesses to give the users a better QoE by providing higher application performance and vastly improved DPI capabilities. For such a solution to work, it is needed a comprehensive test procedure which validates the scalability, performance and capabilities. A solution to evaluate such application-aware infrastructure can be to generate stateful traffic for each application as well as being able to easily measure their performance.

IxLoad is a test tool with stateful protocol emulation for realistic multiplay services, including subscribers, clients and servers. [27] IxLoad can be used for testing converged multiplay services and application delivery platforms by creating real-world traffic scenarios at the TCP/UDP layer and Application layer.[1] The tool has subscriber modelling capabilities which can be used to validate subscriber QoE. IxLoad is designed to be able to test the delivery of application, voice, video and multiplay services envisioned in Figure 13.

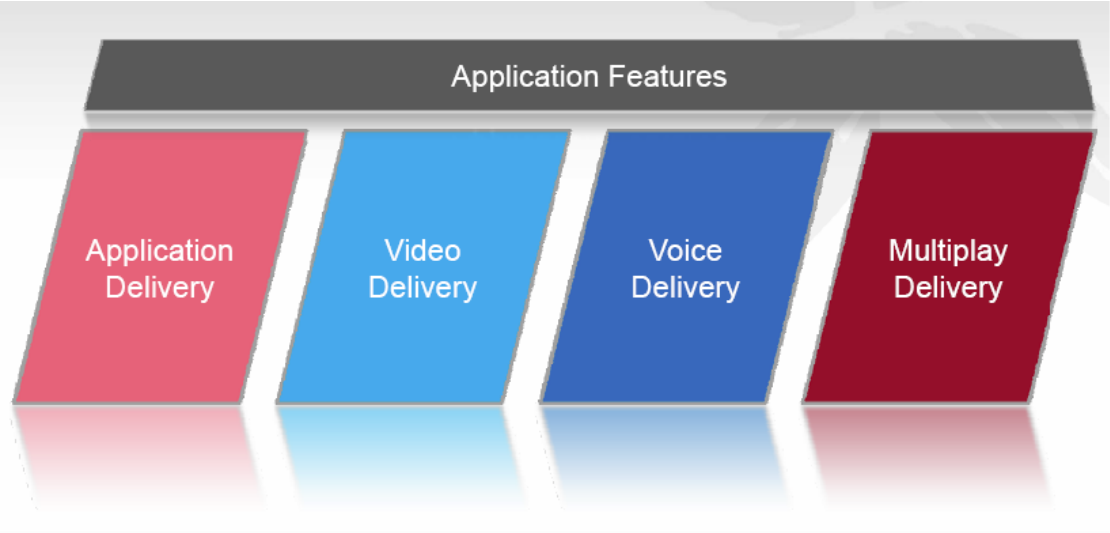

Figure 13: IxLoad application test platforrm

IxLoad can be used for several purposes such as testing multiplay networks, voice, IPTV, infrastructure, security and application-aware devices which performs DPI. For each of these purposes, IxLoad emulates the subscribers and the associated protocol in order to evaluate the performance and to ensure that each subscriber maintain the proper QoE. It supports a wide variety of protocols, including video protocols like RTSP, MLD, IGMP; voice protocols like SIP, MGCP and data protocols like HTTP, SMTP, POP3 and FTP. It has also support for testing the infrastructure of a network, by properly emulating DNS, LDAP, RADIUS and DHCP services. [28]

A device or subsystem connected to other network devices or servers generally communicates by requesting or delivering services. As can be seen in Figure 14, IxLoad is providing the device under test (GGSN in this case) with emulated service subscribers and protocol servers. If the device is a independent server, the IxLoad only needs to emulate the subscribers; however IxLoad is also capable of emulating supporting devices in order to properly evaluate the device under test, for example in a GGSN environment it is fundamental that the test tools should be able to emulate both mobile stations and SGSNs.

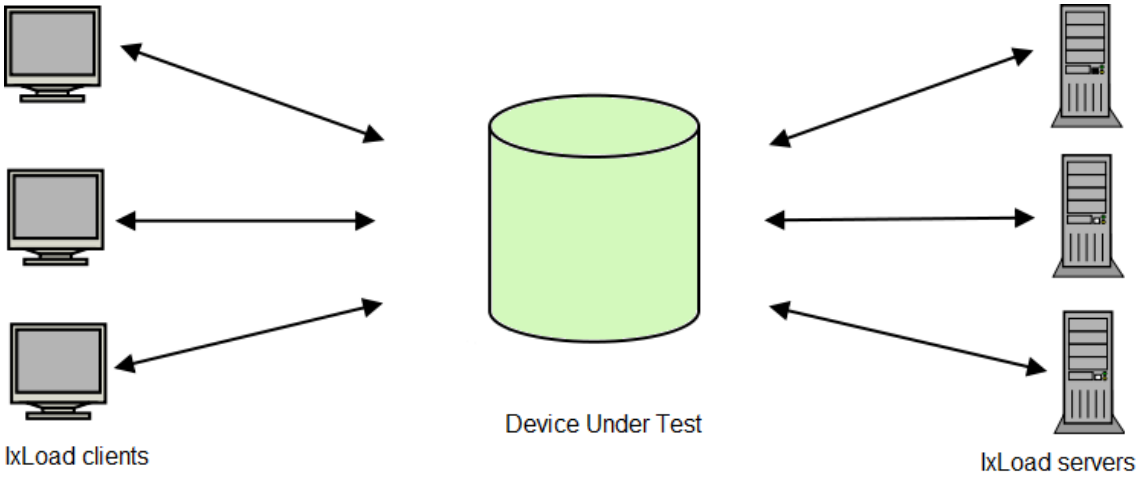

Figure 14: IxLoad emulated environment

The IxLoad chassis transmits and receives control and data traffic with the device under test and it is populated with hot-swappable test interface cards that implement a wide variety of Ethernet, POS and other technologies. These cards are referred to as load modules, where each load module provides a number of interface ports. Each port is equipped with an independent processor and memory. Is also has a specialized traffic generator and capturing hardware. The ports a running a version of the Linux Operating System and supports the Linux TCP/IP protocol stack. [27]

The IxLoad client program is a program which can be installed on any windows computer. It communicates over IP with the chassis. It is not needed to be connected to the server in order to design a test scenario. When the test scenario is ready, the user should connect to the Ixia chassis and select the ports the user wants to use. If the performance of one port is not enough, it is possible to add several ports to the network group. If the Device Under Test (DUT) has several interfaces they can be connected to IxLoad ports, otherwise a switch is needed in the middle.

For the ASM1000XMV12X load module, the ports can be used independently or in an aggregated mode. In this module there is 12 one-gigabit (1G) ports and 1 ten-gigabit (10G) port. There are two modes of aggregation, the 1G aggregated mode and the 10G aggregated mode. In the 1G aggregated mode the twelve 1G-ports combines their CPUs in order to generate complex traffic coming out from one physical port. The 10G aggregated mode makes use of the 10G aggregation port. In this mode, the twelve 1G-ports are combined in order to generate stateful traffic at 10 Gbps from the 10G-port. [29]

IxLoad is using a GUI application for creating test scenarios. When the ports have been assigned and the scenario is ready to be configured, the user can either press the “run” button or the “apply config” button. The different test options are showed in Figure 15.

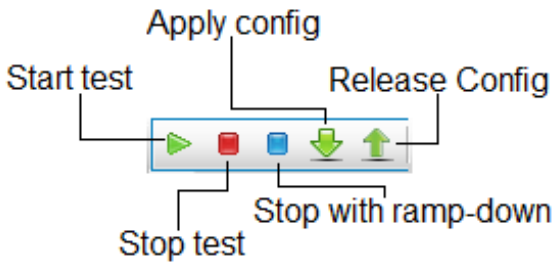

Figure 15: Test menu

The run button triggers the procedure to configure the ports and initiate the test. The apply button can be used to just load the configuration onto the ports, and prepare the interfaces, going into the start “configured”-mode, this means that if the user wishes to initiate his test, he can immediately start his test without any delay. When the test is finished it returns to the configured state, maintaining the lower layer mechanisms such as responding to ARP and ping from the DUT. If the user wishes to tune the test case, it is possible for IxLoad to perform a smart cleaning and only apply the tuning rather than reapplying the whole configuration. When a test is running, the user has two ways of stopping the test, one way is by pressing the red button that abruptly stops all on-going transactions or the blue button which initiates the ramp-down phase, in which the transactions are allowed to finish their on-going transaction in order to make a more gradual finish, instead of terminating all the users transactions and making the client and server statistics inconsistent and nonmatching. When the user deems that he is finished with a test, he presses the “release config” causing IxLoad to clean the ports from the configuration. After having executed a test case, it is possible to generate a report in PDF format which includes details about the test.

It is also possible to generate Tcl scripts after creating test scenario in IxLoad GUI application for test automation purpose. This is useful since no IxLoad specific skills are required to execute test scenarios developed by IxLoad. Also this Tcl scripts can be changed and adapted in different ways.

3.2 Subscriber Model

IxLoad provides modelling functionality by generating traffic that simulates the dynamic behaviour of subscriber. It put emphasis on achieving network realism, with multiple subnetworks, having unique MAC addresses for each client. Also, on each port it is possible to have multiple application protocol, having multiple VLAN, and emulated routers. It supports

different test objectives, such as simulated users, concurrent connections, connection rate, and throughput, which may include a constraint on the amount of underlying network-level interfaces. The objective is associated to a specific protocol activity, which makes it possible for different activities to have different objectives. Each of these activities has its own timeline, which includes ramp-up and ramp-down phases. To create scenarios closer to real-world traffic where users are connecting and disconnecting arbitrarily, IxLoad offers timeline-functionality, which allows the users to create different timeline segments with individual duration and other properties. Figure 16 shows an example of a timeline, where the Y axis shows the “Objective Values” based on chosen “Objective Type” and X axis shows the duration of the test case. Furthermore, it depicts a linear segment, a negative step segment, a Poisson segment, a burst segment with two repetitions, one pulse segment and later a negative linear segment which ends the test. It is also possible to repeat the timeline a number of times by increasing the iteration counter.

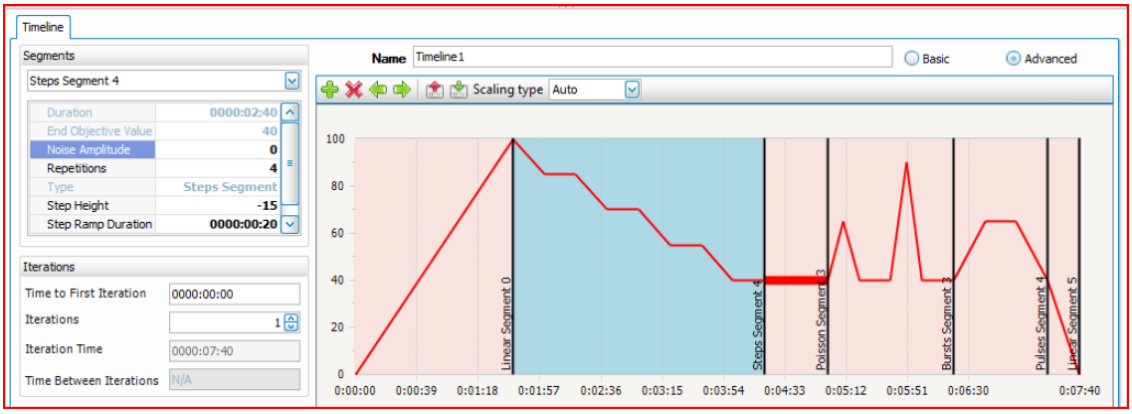

Figure 16: Timeline example

Dynamic Interface Behaviour (DIB) is a part of the subscriber modelling which gives further capability to mimic subscriber manners and network realism in order to test application-aware devices and networks. DIB is used to create dynamic interface setup and tear-down behaviour associated with subscriber behaviour as well as evaluating the network control plane performance correlation with the subscriber application-layer. It includes Dynamic Control Plane (DCP) and Network Failure Threshold (NFT) features. The DCP is creating the interfaces dynamically during the ramp-up phase and the application data starts to flow as soon as its corresponding interface is up. NFT is working in parallel with DCP and even if some interfaces fail during the test run it allows IxLoad to continue the test by controlling the number of allowed failures, beyond which the test is terminated. [30]

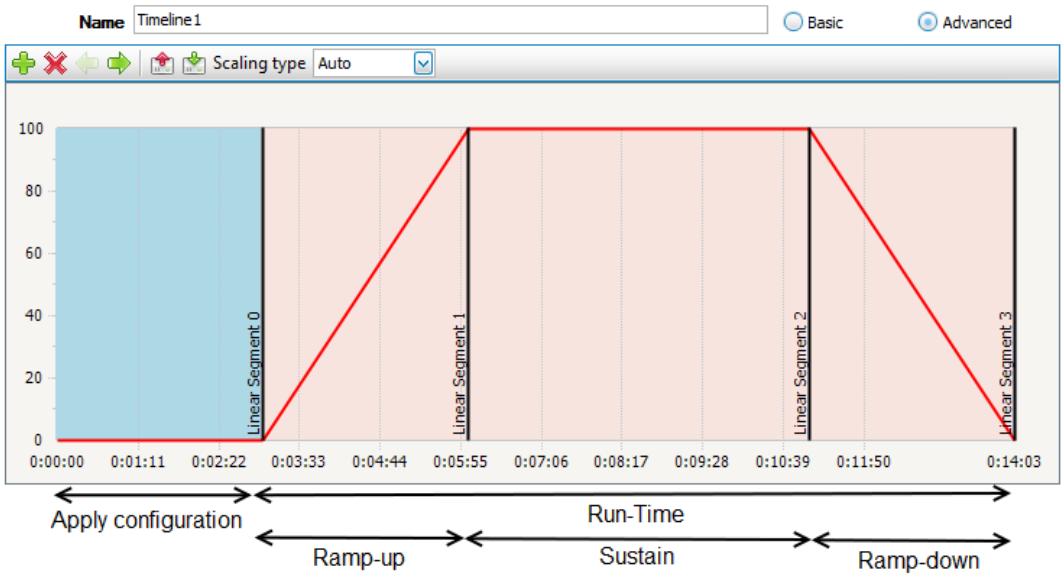

Figure 17: Dynamic Control Plane (DCP)

Figure 17 describes the dynamic interface behaviour where no interfaces are brought up during the Apply Configuration phase and interfaces comes up dynamically as users are spawned and traffic associated to those users begin to flow. Those interfaces are torn down dynamically in case that the users leave the test. [31]

The IxLoad subscriber modelling offers several functionalities: [27]

- It defines how subscribers join their service provider network for instance using PPP/L2TP, IP, VLAN, DHCP, IPSec or 802.1P.

- It can configure the upload/download bandwidth on a subscriber level, enforce a traffic mix and how the mix can interact with each other, it also offers a randomized think command which is intended to properly emulate the user behaviour, for instance when a user downloads a page and reads it before it continues browsing another page.

- It is able to measure QoE per application for example packet loss, jitter, response latency, MDI and MOS_V for video and MOS for VoIP.

In IxLoad, a Network is the list of addresses and other properties belonging to the lower-lever protocols, which the upper-layer protocol activity is running over. It is possible to create several networks to resemble a complex scenario, in which the subnetworks share certain characteristics. However, each of these networks is required to be associated to separate IxLoad ports. Each of these networks are either the type NetTraffic or of the type Subscriber. The NetTraffic type is the one that is normally used; it supports all the available protocols and objective types, but the network parameters are the same for all NetTraffics. The Subscriber type supports the possibility to control the interaction in a sequential manner between the configured protocols for each user, making it possible to accurately mimic real world user traffic patterns. However, the subscriber type only supports a subset of the protocol supported by NetTraffic and the only test objective available is Subscribers, which is similar to the objective type ‘simulated users’. Although for simulated users, all the protocols are running concurrently without any correlation.

Within each Network, it is the possible to define a set of network ranges that should belong to the different applications. Figure 18 is an example where different applications are associated with a set of the defined network ranges. In this example, the HTTP server is active for both IPv4 and IPv6 with IPv4 address 10.10.20.10 and within the IPv6 address range 2001:f040::1 to 2001:f040::15.

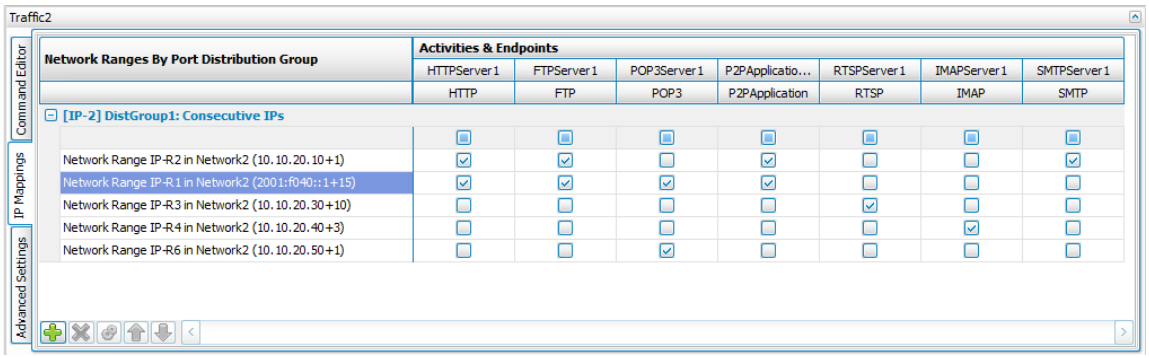

Figure 18: IP mapping on the server side

Figure 19 depicts an example of a subscriber model where subscribers are following a scenario. In this particular scenario the subscriber first checks his mail, in one of his mail he gets a web link to a streaming site, on the site the subscribers decides to stream a video, he likes the video so he downloads it via the sites' FTP server and he then shares it amongst his friend by using BitTorrent.

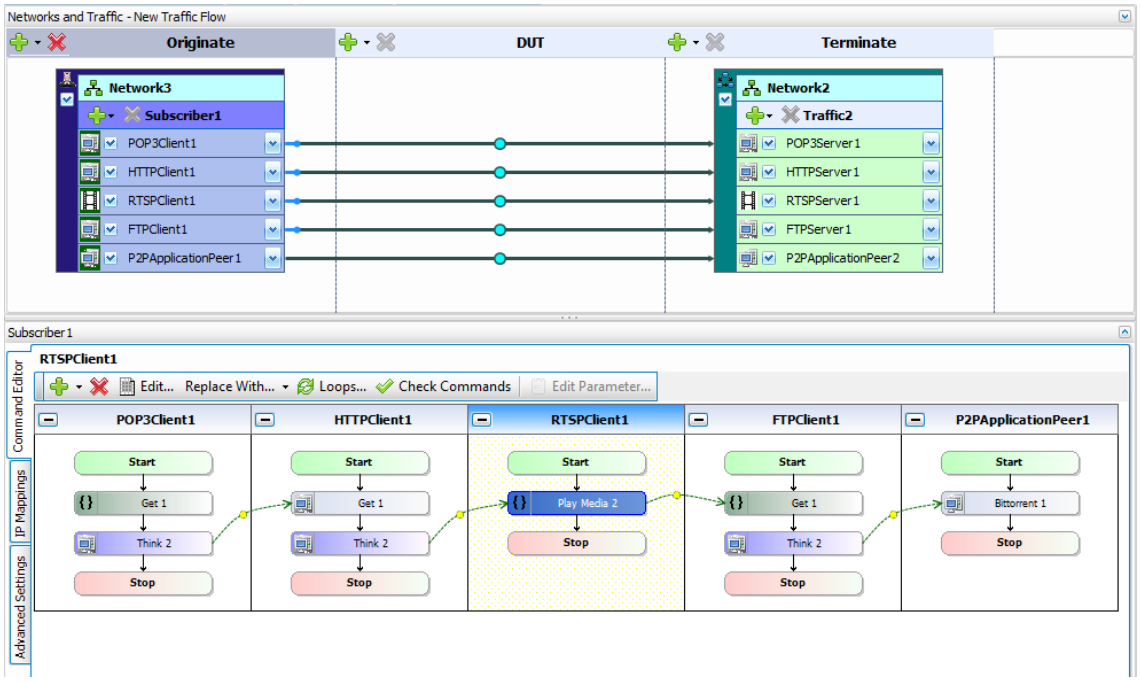

Figure 19: Subscriber modeling example

On the network level it is possible to configure different Global services which includes: [31]

- **DNS Global Service:** This gives the possibility to use symbolic host names simplifying the configuration for some DUTs in order to achieve a more natural work flow. An example could be to configure the clients to know the relation between a URL and the IP address, in this way the DUT can be configured to use the URL instead of using an IP number. There is also the possibility to configure IxLoad to query name servers available to the IxLoad port.

- **TCP Global Service:** For a fully stateful emulator, like IxLoad, it is easy to configure various settings of the TCP transmission, for instance the buffer sizes, retransmission timeouts and the retries of various TCP types. Additional RFC-documented functionality such as “Explicit Congestion Notification”, “TCP timestamps” or “TCP Selective Acknowledgements” are available to give the most suitable test scenario to test the DUTs’ performance.

- **Filter Global Service:** In this setting it is possible for the ports to filter the network traffic received on the port in order to exclude unrelated traffic and only allow the appropriate test traffic. This can be done by configuring a layered custom filter, including filters for specific packets on MAC, IP, ICMP, UDP and TCP values. These settings can be automatically configured by IxLoad to match the network traffic included in the test scenario.

- **GratARP Global Service:** This service gives the possibility to broadcast or listen to gratuitous ARPs on all connected interfaces before starting the test. This ensures that both the DUT and the IxLoad ports have a valid ARP cache, so that they have the correct IP matched to the link-level hardware address.

- **Settings Global Service:** This service handles the interface behaviour. If it is going to be static behaviour that creates the interface at the start of the test case or if the interface shall be created dynamically. The dynamic control plane mode offers two different settings, the first option enables the arbitrary DCP in which the interface is created together with the simulated user, there is also a sub-option to this where the interface can be torn down with the user during the ramp-down phase. The second option creates the interface at the start and tears it down at the end of the command list. Notice that this is done on a global level so that all the configured activities on this network will have the same behaviour. This means that if multiple users share the same interface, the interface will be created together with the first user and torn down with the last user. The same behaviour occurs when multiple activities are sharing the same interface.

3.3 Impairment

One distinct feature of IxLoad is its ability to simulate an unreliable network with packet loss, jitter, delay, etc. This feature is called Impairment and offers the tester the possibility to introduce traffic flow anomalies and errors.

The Impairment functionality is defined as a plug-in and can be applied on top of IP, DHCP Client, PPPoX and GTP, which will impair the user traffic carried on top of the tunnelling protocol. This functionality is configurable through the use of Impair profiles. It is possible to

define several profiles and apply them to any traffic stream by specifying a filter, which can be based on source IP address, destination IP address, source port, destination port, Type of Service (TOS) fields, TCP flags, and protocol type. These parameters can be used separately or in a combination which gives the union of the involved parameters.

To influence traffic streams, IxLoad offers several impairment attributes, such as: [31]

- **Delay**: Inserts latency errors into a packet stream, including a static delay and random jitter.

- **Drop**: Emulates random packet loss from a packet stream by specifying the desired percentage.

- **Drop Sequence**: Emulates packet loss using a specified drop sequence. This setting has two parameters, the dropped packets (d) and the skipped packets (s), meaning that a sequence of d packets is dropped after each transmitted sequence of s packets.

- **Reorder**: This attribute is used to create reordered packets which arrive out of order in the packet stream. A real world example might be that the packets were delayed during transmission.

- **Reorder Sequence**: It performs the reordering by skipping and delaying packets in a sequential manner.

- **Duplicate**: Emulates the appearance of duplicate packets in a packet stream by specifying the wanted percentage.

- **Fragment**: Imitates the presence of fragmented packets, by specifying the desired percentage and the Maximum Transmitted Unit (MTU). In this setting it is possible to select if the fragments should be sent in reverse order allowing testing the worst case scenario for reassembling packets, to only send the first fragment in order to test the mechanism for reassembly timeouts and an option to trigger overlapping fragments.

- **Fragment Sequence**: This option performs the same fragmentation, but based on a defined packet sequence.

- **Outbound Rate**: Limits egress traffic speed in order to simulate a lower bandwidth network by specifying the desired rate.

- **Inbound Rate**: Limits ingress traffic speed in the same way as outbound rate.

3.4 Statistics

It is essential in each test scenario to be able to easily verify that the device under test behaved correctly. Collected statistics in IxLoad are presented by the StatViewer application which can be displayed in a separate window or as an IxLoad component. Each statistical display is called a view and it is a combination of different statistics for a single purpose, as the view for “HTTP Client- TCP connections” for example includes the SYN packets sent/received and the FIN packets sent/received. For each test case, views can be created or the test case can be associated to a predefine view, which can be customized.

Below is the general list of statistics provided by IxLoad: [31]

- Layer 2-3 Statistics for Client Ports
- Layer 2-3 Statistics for Server Ports
- Port Statistics
- SNMP Statistics
- Protocol Specific Statistics

The L2-3 statistics presents the link data statistics, for example the “Frame Sent Rate”, “Bytes Received” and “Transmit Arp Requests”. The port statistics include information about disk utilization of the port and the average CPU load. The SNMP statistics gives the user the possibility to request information from the device under test. The Simple Network Management Protocol (SNMP) is a standardized device management protocol. SNMP uses the Management Information Base (MIB) which is a hierarchical collection of information, where each object is a variable updated and maintained by the responsible device. The protocol specific statistics gives information about the different protocols that is included in the test, if HTTP is used it will present layer 4 and layer 7 information both from the client and from the server. An example is shown in Figure 20.

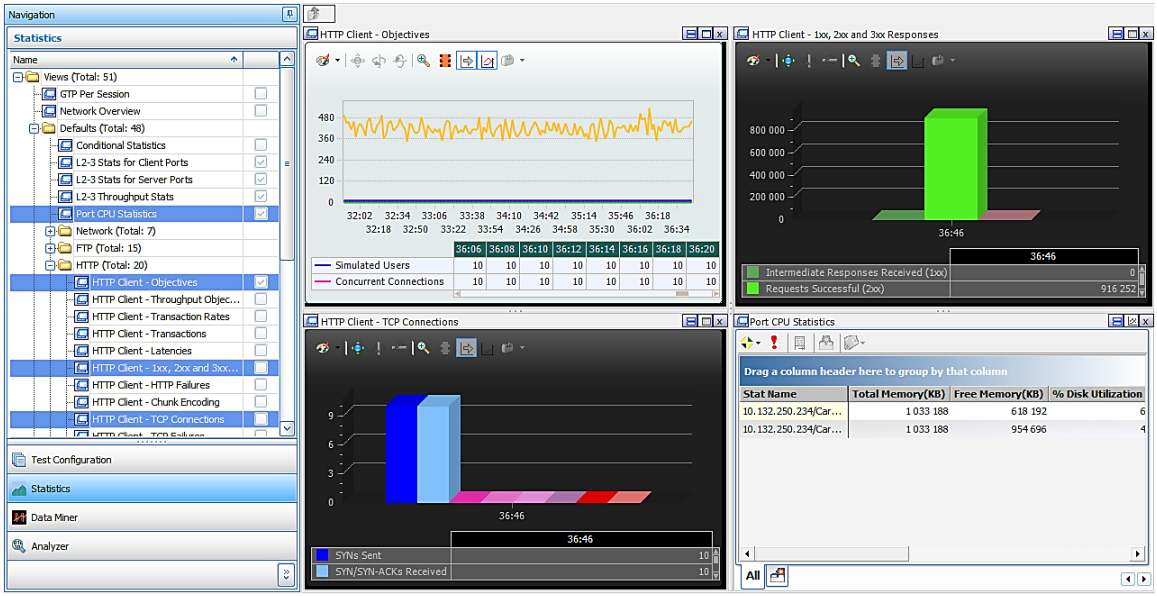

Figure 20: Example of HTTP statistics

The StatViewer is continuously saving the collected statistics into .csv files on the computer that is running the IxLoad client. When a test is finished, a report can be produced; this report displays the collected information either in a pdf or in html format. The content in the report is configurable to include just a summary, the test configuration or a fully detailed report including all the information collected during the test.

As stated previously, IxLoad is saving the collected statistics locally on the computer, it stores them together with the test configuration file (.rxf). The Data Miner is an integrated application within IxLoad which provides a complete view of the executed test. The Data Miner gives the users the possibility to review the gather statistics from a previously run test by viewing the .csv-files, if a report was generated it will also be visible here or it is also possible to generate a

report from the gathered statistics. From this view, it is possible to select an old test case and either reload the configuration or rerun the test.

3.5 QoE Detective

QoE is defined as a user´s gratification with their experienced service. These services can be a composite mix of several services such as video, voice over IP, streaming media and gaming. By measuring the QoE, the operator gains an insight of how a customer may perceive the performance of their network therefore, realistic measurement of QoE is vital. Consequently it is necessary that QoE issues in a scaled test environment should isolated to the emulated subscribers and what services they have experienced. IxLoad QoE Detective offers the following features to isolate error and failures of simulated subscribers: [32]

- **Conditional Statistics:** This feature gives the possibility to monitor statistics by the user ID, IP address, or VLAN ID in addition to the aggregated statistics. Also, statistics can be sort or filtered by specific condition which helps isolating subscriber problems that might appear during the test.

- **Network Diagnostics:** This feature make it possible to view a summary of the entire number of interfaces requested, initiated and failed. It can aid to identify the problem caused by misconfiguration in IxLoad or the DUT by providing statistics about details per session or per interface. It is also possible to sort and filter the Network Diagnostics based on the special condition.

- **Drill-down Statistics:** This feature provides statistics per NetTraffic, per Activity and per Port which helps to go through details of each individual network port, traffic or even activity statistics. This can be used for example to see the exact error cause for each session.

- **IP Assignments View**: This feature shows the mapping between the physical IxLoad ports and the assigned addresses of the activities running on that port. This can help to distinguish configuration problems and interface problems. It provides information for both static and dynamic interfaces such as PPP and DHCP along with real time information like IP addresses, or VLANs. This feature is not supported over GTP stack.

3.6 Analyzer

Analyzer is another application within IxLoad for troubleshooting, which captures the traffic on the IxLoad ports and shows the captured packets, network and application specific statistics such as SIP/MGCP call diagrams or MPEG video streams. It allows the user to inspect each individual packet from layer 7 to layer 2 by decoding traffic streams. [31] [33]

Analyzer capture options such as the buffer size for storing the capture, can be configured before starting the test through the Capture Settings window which is shown in Figure 21. The capturing of each assigned port can be filtered based on protocol or through filtering using filter string expressions from the tcpdump application. Packet capturing can be started or stopped when the test is running if you have already checked Traffic 1 and 2 boxes.

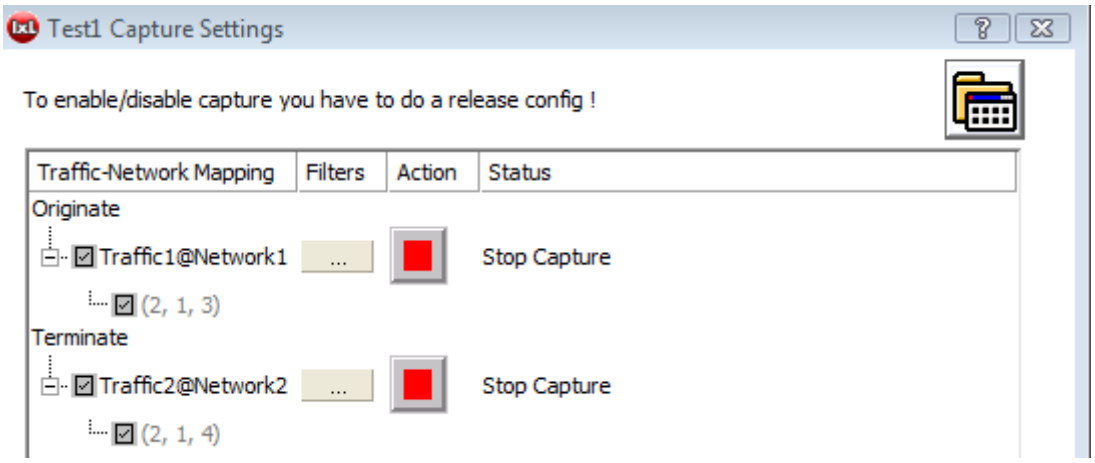

Figure 21: Analyzer Capture Settings window

Figure 22 depicts the Analyzer main window which consists of Navigation toolbox, Common view, Packet tree view and application view panels. The Application view contains packet details for the specified port. Packet number, time, packet length, source MAC, destination MAC, source IP, destination IP and protocol type can easily be viewed in this window. Also packets can be sorted or filtered based on each of those items. The Navigation Toolbox is a direct way to Application, Network and Physical layer categories which gives the possibility to see associated packets in different layers. The Common view consists of two different tabs: Ladder diagram and Flow summary. The Ladder diagram displays the selected flow between source and destination, making it easy to identify specific transactions. The flow summary view provides brief information about the current flow. The Packet tree view displays the selected packet in hexadecimal form and it also contains packet details information for each layer.

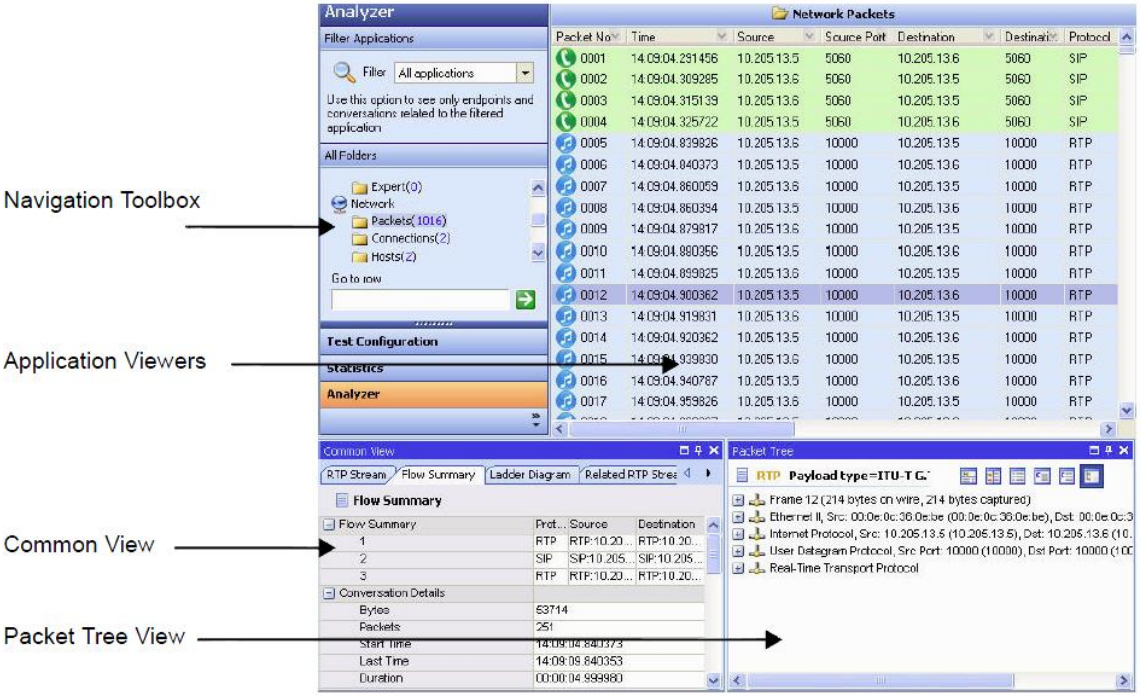

Figure 22: Analyzer main window

3.7 IxLoad GTP overview

GTP is one of the several protocol supported by IxLoad which gives the possibility to emulate a packet core environment such as SGSN and external PDN. In addition to this, IxLoad is also able to concurrently emulate multiple fully stateful layer 7 protocol activities, making it a suitable tool for scalable GGSN testing. Also, all the supported layer 7 protocols can cooperate with real world servers and peers.

IxLoad is used through a graphical user interface which simplifies the creation of test cases with a higher-layer perspective. The GTP-C signalling procedures, are handled by underlying mechanisms of IxLoad based on the user applied settings. The IxLoad implementation of the standard of GTP-C includes: [34]

- UE/SGSN initiated PDP creation, modification and deletion and network initiated modification and deletion.
- Full QoS configuration together with configurable interworking between the 3GPP QoS settings and the IETF settings.
- Support for TFT and secondary contexts together with preconfigured TFT per layer 7 activity.
- Configurable APNs per subscriber range.
- Dynamic IP address allocation during session establishment.
- Protocol Configuration Options for MS-GGSN communication.
- Configuration of IMSI, MSISDN.
- Support for user mobility between SGSNs on the same port with configurable event intervals and mobility paths, including mobility concurrently with DTI.

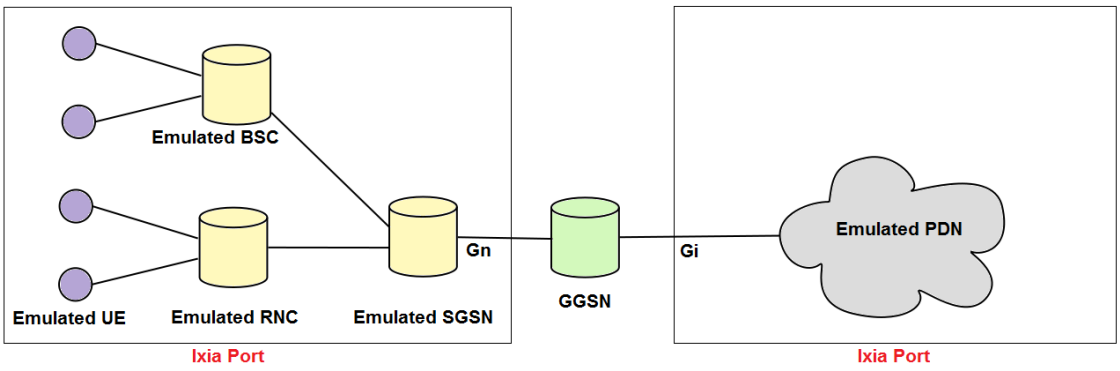

Figure 23: IxLoad emulated environment for GGSN testing

Figure 23 shows the primary IxLoad emulation for testing GGSN functionalities and performance. With this setting at least one IxLoad port is needed to emulate the Gi interface with emulated PDN devices, and at least one port is needed to manage the Gn interface with its SGSNs RNCs and UEs.

One GTP session can support up to 11 IxLoad traffic activities. When handling GTP in IxLoad, the activities are defined for a range of GTP users. The settings and options for all the users are equal in terms of the specified APN, responsible SGSN and mobility path. The PDP context created for each activity has its own traffic setting which consists of specified APN, Session time settings, PCO, QoS IE settings, TFT and secondary context settings. [35]

The usual test configuration consists of creating different traffic activities and their command lists. However, IxLoad offers a session duration configuration timer if a time based test scenario with dynamic control plane is required. If the command list is finished without session duration enabled then the interfaces will be torn down, but the session duration is enabled, the interface stays up until the session timer expires even if the command list is already finished. Likewise, the interface would also be torn down if the session duration is enabled and the timer expires even if the command list is not finished. Consequently, statistics measurements will be erroneous reflecting the fact that the command list was abruptly stopped. To avoid this, it is suggested to use a long think command in the end of the command list in order to provide enough time for all the transaction to be finished when the timer expires. [35]

The GTP Plug-in Statistics in divided in to two major subgroups which are Global statistics and Conditional statistics. All ranges, network groups and protocols which are defined for the test scenario can be represented in Global statistics. Conditional statistics can be generated either per session or per range. The conditional statistics per session shows each PDP and its associated GTP statistics. The conditional statistics per range aggregates all the per session statistics, and represents the statistics of the entire range. [35]

Since IxLoad is designed to a stateful simulator, all the possible settings are provided to enable thorough and comprehensive testing of any sort. IxLoad has support for a variety of protocols, but they are not all supported over GTP or applicable in GGSN testing. All GTP supported protocols are divided into different blocks and shown below: [36]

- **HTTP, SSL and FTP**: IxLoad supports HTTP version 1.0 and 1.1 and TLS version 1.0 according to RFC 1945, RFC 2616 and RFC 2246 [36]. In essence, IxLoad supports cookies, redirection, decompression on HTTP clients, Content-MD5 integrity check, Chunked Encoding processing on HTTP clients and the option to use multiple TCP connections per user together with pipelining. Apart from all the RFC related functionality, IxLoad offers a sequence generator that create large numbers of user sessions with unique credentials, definable pages on HTTP servers in terms of randomized or fixed page size, configurable HTTP request headers and customizable response codes. The HTTP server can be configured to handle multiple TCP ports and includes randomized response delays between a minimum and maximum. For HTTPS IxLoad gives SSL session reuse, configurable certificates for client and server side, Diffie-Hellman ephemeral key exchange and selectable ciphers for SSL handshake including AES, RSA, 3DES, etc. Furthermore, IxLoad supports FTP protocol based on RFC 959. [34] It gives the possibility to configure the FTP server to listen on multiple TCP ports and user-definable files on FTP server. [37]

- **Mail (SMTP, POP3 and IMAP):** IxLoad supports SMTP according to RFC 2821, POP3 according to RFC 1939 and IMAP according to RFC 3501. [36] It is able to emulate several email clients and servers and related protocols. Therefore, it can be used for email delivery testing. It gives possibility to detect and troubleshoot email delivery by measuring the load influence on the mail gateway handling rate, testing the capacity of mail server and evaluating the content filtering effect on the mail delivery system.
  Moreover, for all Mail protocols that IxLoad supports it provides features such as the capability to create sequence of related commands, configuring email headers, mail body and attachments, ability to insert custom mail header or randomizing size of mail body and attachment. It also gives the possibility to initiate thousands of user sessions

and configuring mail server to listen to multiple TCP ports for requests. Also for POP3 it supports authentication based on passwords or APOP shared secrets, which can authenticate the client. [38]

- **Streaming (RTSP, RTP):** Media streams and related devices for audio and video delivery over the Internet can be tested by IxLoad.it has support for RTP based on RFCs 3550, 3551, 2833 and 3389. [36] IxLoad is able to evaluate the streaming media servers and media caches behaviour together with measuring the network effects on the quality of delivered media to the users. In addition, IxLoad offers functionality to detect RTP lost or out of order packets as well as RTP jitter and latency measurements for RTSP commands. It is also able to emulate media players using customizable client side RTSP headers and it supports RTSP redirection and proxy server commands. [39]

- **Session Initiation Protocol (SIP):** IxLoad supports SIP based on RFCs 3261, 2327, 2976, 3262, 3264, 3265, 3311, 3515, 3428, 3966. [36] SIP is basically used as an application-layer control protocol to create, modify and terminate Internet sessions with one or more members. These sessions can be used for several purposes such as Internet telephone calls, multimedia conferences, and multimedia distribution. However SIP is not a self-contained protocol, it relies on other protocols to perform the specific tasks in a multimedia session, such as RTP for transporting real-time data and QoS feedback and RTSP for streaming. SIP is in charge of handling and establishing the initiation of a session. IxLoad has the capability to emulate voice calls with video with calculating the Mean Opinion Score (MOS) reported on a per call basis. The SIP module has support for the sequence generator mentioned above in the HTTP section, in order to create unique users, it has customizable SIP settings including what type of TOS/DSCP, the SIP or the resulting RTP protocol should have. This gives the possibility to examine the performance and capacity of the network by mixing low priority traffic with delay and jitter sensitive SIP traffic or testing the performance and capacity of a DUT doing SIP registration, handling call setup, teardown and redirections.

- **Peer to Peer (P2P):** IxLoad is able to generate a traffic-mix composed of P2P, voice, video and data traffic which gives the ability to recognize and prioritize P2P traffic during different test scenarios. Therefore, it can be used for evaluating the performance of DPI capable devices such as GGSN. BitTorrent and eDonkey are the two P2P protocols supported by IxLoad. Each P2P peer can be configured as an initiator, a responder, or both. It is just necessary to configure the initiator since the responder learns the configuration from the initiator. The flow between two peers is configurable in the manner of the amount of data transferred, together with if the flow should consist of download, upload, bidirectional traffic or an intermixing of two out of three. Connection rate, Concurrent, Connections, Throughput, Initiator Peer Count and Transaction Rate are the selectable test objectives.[40]

- **Application Replay:** IxLoad is capable to statefully replay selected application flows from imported capture files. The Application replay feature is able to replay bi-directional traffic concurrently. For this matter, IxLoad chooses to regard the application replay agents as being peers in a P2P model, so that the agents can be an initiator, a responder or even both. The application replay was implemented to consider those protocols that are not originally implemented in IxLoad, for instance Skype, Google Talk, Gnutella or other P2P traffic patterns. In this way, IxLoad can replicate user-

specified traffic alongside IxLoad-emulated traffic. This functionality combined with the IxLoad infrastructure gives the possibility to test and benchmark a content-aware device both on a scalability level and on a performance level. The special thing about the IxLoad Application Replay functionality is that it builds up the message handshakes needed to properly recreate the recorded session. It is possible to preserve the packet boundaries and the TCP/IP headers or to change the IP addresses and port numbers in the replay. By changing the IP addresses and port numbers, the TCP/UDP checksum will be recalculated to match the new values. IxLoad offers the possibility to filter the capture file based on which IP and port the initiator and responders should have as starting points. With the specific nature of a capture file, it is also possible for IxLoad to verify the content and length of the payload compare to the expected payload. An App-replay activity can be configured into three types; custom flow-IP, custom flow-TCP or custom flow-UDP, where the last two includes improvements and enhancements in each area. The custom flow- IP is however the most complete replay activity which can replay all kind of IP-based captures including a mix of UDP and TCP packets which some P2P protocol can consist of. The IP custom flow is also able to maintain the inter packet time interval based on the information in the capture file. The only test objective that custom flow-IP has, is simulated users, while custom flow-UDP and custom flow-TCP flow have Connection Rate, Concurrent Connections, Throughput, Initiator Peer Count (equivalent to Simulated Users) and Transaction Rate. [31]

IxLoad is currently not supporting the following protocols and modules over GTP:

- Application Test
- DDoS
- DHCP
- DNS
- IPTV/Video
- LDAP
- MGCP
- Packet monitor
- QHTTP
- QTCP
- RADIUS
- SSH
- TELNET
- TFTP
- TraceFile Replay
- Vulnerability
- WAP
- Stateless peer
- Voip

4 Test scenarios

In this chapter different GGSN specific test scenarios will be proposed in which the quality of IxLoad can be proven. Each test case will include the following:

- **Purpose:** Defines the purpose of the test case in the whole environment.
- **Configuration:** Consists of two parts, the general GGSN configuration and the IxLoad main configuration.
- **Verification:** How to analyse each part. Which items have to be checked in order to verify the result of the test.

For all the proposed test cases, a GTP stack is set for the IxLoad client network, since IxLoad has an IP stack by default for both client and server side. Moreover, all the test cases have been configured in dynamic mode on the client side and static mode on server side to properly simulate real world scenario when users are connecting and leaving the network.

Due to confidentiality reasons, the full GGSN configuration will not be shown here. Some parts of the IxLoad configuration that involves Ericsson specific test solutions will also be kept confidential due to the risk of information leakage. The following test cases has been implemented with IxLoad software version 4.30.119.79, version 5.0.117.28 EA and version 5.10.137.20.

4.1 HTTP Dynamic Redirect

An example of this functionality is when a user is trying to access a website without having enough credit and instead of accessing the wanted website the user is redirect to the operator's webpage where he can refill his credit.

Purpose: The dynamic nature of IxLoad has been evaluated during this test case, in which the emulated UE should adhere to a HTTP response 302. This response includes a URI location that the UE shall be redirected to. In this test case the GGSN handles the user redirection by sending HTTP 302 message to the user. As stated in sections 2.6.2.2 and 2.6.3, this redirect may be sent for many reasons, for example that the user doesn't have enough credit and shall be redirected to the operators' site in order to refill his card.

Configuration: The GGSN needs to be configured to perform packet inspection and identify the traffic that shall be redirected, GGSN is able to redirect the traffic based on the header of the packets or identifying which URI is requested and performing URI based redirection. The aim of this test case is to implement URI-based redirection. The scenario is shown in Figure 24:

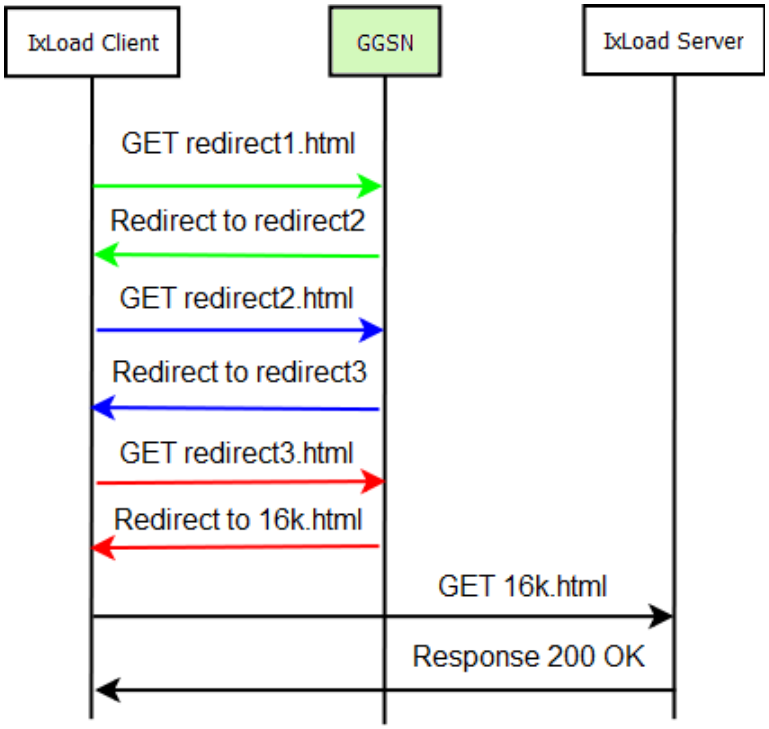

Figure 24: The HTTP redirect scenario

In this scenario, the IxLoad client was configured to get the page “redirect1.html” from the IxLoad server envisioned in Figure 24. The GGSN is configured to deny that request and redirect the user to another page since in this scenario the user is not allowed to roam to that site. When the client has received the redirect response, it follows the redirect which guides him to “redirect2.html”. The GGSN is configured to disallow access, because the demanded QoS cannot be provided for the subscriber, so the client is again redirected to a new page “redirect3.html”. When the client is requesting that site, the GGSN determines that the client is blacklisted and it is prohibited to access the site so it will be redirect to the page, “16k.html”, which is a default page in the IxLoad server. The GGSN allows the client to access that page, so the client is finally able to successfully acquire a page. The reason to create such complicated test case with three redirects for different reasons is to test the dynamic behaviour of IxLoad and its reactions.

The essential IxLoad client configuration is shown in Figure 25 where the HTTPClient1 has been created and configured to request the page “redirect1.html” from the server “Traffic2_HTTPServer1” on port 80. To get a distinct overview about what IxLoad is doing, the selected value for objective type “Simulated user” is equal to one meaning that there is just one subscriber executing the command list once. The designed scenario is easy to achieve with a short and basic timeline. It is easy to scale the users and the timeline from this scenario to meet the wanted objective, also infinite loop can be added on the command list to extend the user activity. The network interface behaviour was set to create the interface with the user and tear it down with the user in order to scale well with an increasing objective value.

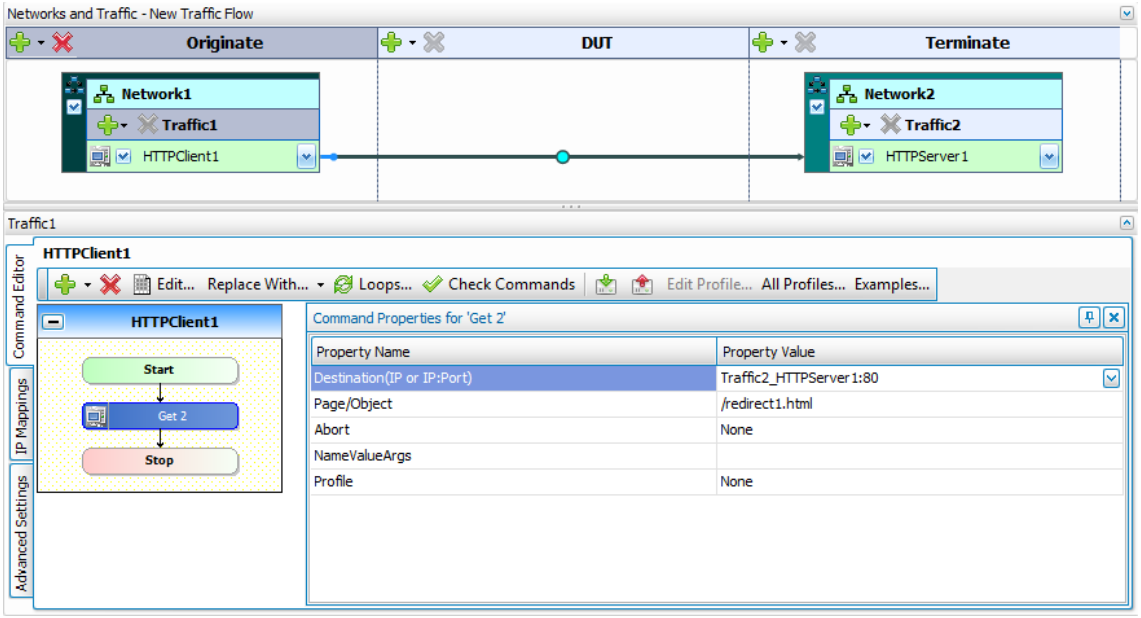

Figure 25: Redirect IxLoad client configuration

Verification: To verify this test case, it is needed to look at the differences between the HTTP client and the HTTP server, since the HTTP client should receive three HTTP 302 messages and do four HTTP GET requests. The server on the other hand, should only receive one GET request and respond to it. This process can be verified through the ladder diagram in Analyzer. However, this requires a human to be involved to inspect the captured packets. Another way is to look at the gathered statistics, which includes “HTTP Server Response Sent”, “HTTP Client Throughput”, “GTP General” and “HTTP Client Responses”. An example is shown in Figure 26. These have been selected from a vast amount of statistics, since they make verification more clear. The GTP general statistics was selected in order to see if the underlying PDP context was successfully initiated. The HTTP client and HTTP server statistics was selected to clearly depict the behaviour of a GGSN redirect, where the client has received three 302 and one 200 messages while the server has only sent one 200 response.

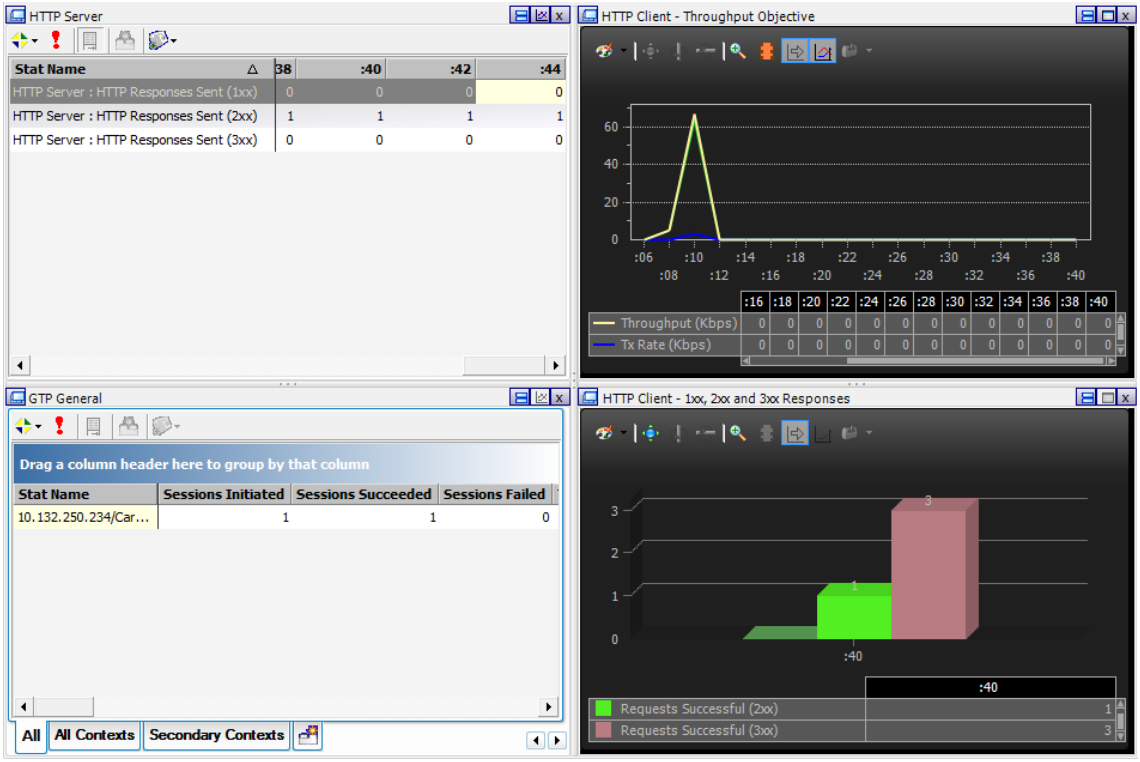

Figure 26: Redirect test statistics

4.2 Multi-application for DPI

Purpose: GGSN has evolved to be content aware and it is supporting DPI for many applications. Packet inspection is one of the core features in order for GGSN to do traffic shaping, service-aware credit control and dynamic redirect. These features are demanding to know what kind of traffic is passing through the GGSN and identify the specific packets. For this purpose a full blend of users performing different tasks is required to be simulated. Both header-based inspection and deep packet inspection are performed in this test case.

Configuration: The GGSN is configured to identify multiple protocols, based on the header level and on DPI level. In IxLoad, multiple client activities with a standard command list and infinite loop are created under NetTraffic1 and default servers are implemented on NetTraffic2. When the generated traffic hits the GGSN specific rule, the GGSN will classify it and show the accumulated statistics. Figure 27 shows the IxLoad setup, where each client activity does its default command followed by a 1 second think command. All these activities are configured with one subscriber and under the same APN with same UE range. The timeline needs to be long enough for packet inspection to be thoroughly performed and evaluated. If it is wanted to measure the DPI performance in terms of CPU load or throughput, it can be easily done by scaling the amount of users and the timeline. However, in this case the objective was to use one subscriber and see that all the traffic was classified to the correct rule. It is nonetheless important to note that IxLoad is able to generate traffic at near line-rate and if the GGSN is equipped with low-performance cards, it will not be able to inspect and keep track of all the flows.

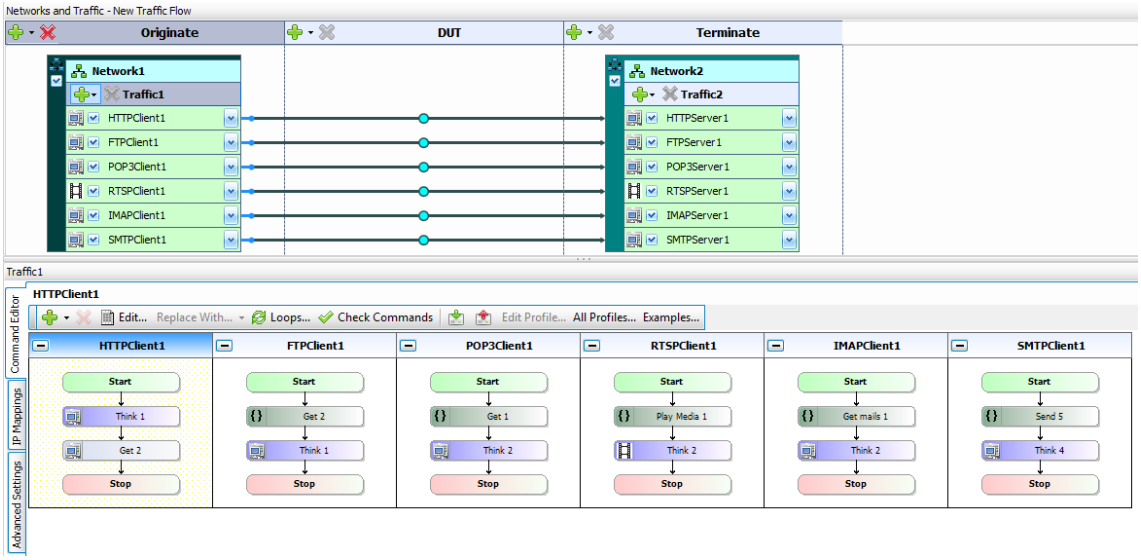

Figure 27: Multi-Application settings

Verification: For a full extensive functional test verification of the DPI, it requires an extensive amount of work. It is needed to capture the traffic and see exactly when the DPI rules catches the ongoing transactions, since in general the DPI rules are specified to look for a particular operation or a specific value in the application data. These values might come in the third packet in a transaction, meaning that the first two messeages will not match the DPI rule and be classified either to a default classification or a header rule that catches the traffic. This means that the tester needs to have a clear view of the traffic flow of the current application in order to know which rule the traffic should hit. With this clear in mind, the validation is to view each simulated application statistics and compare with what the GGSN is able to detect and inspect. For this test case, it was chosen to view the bytes sent and received for each of the simulated applications, as can be seen in Figure 28.

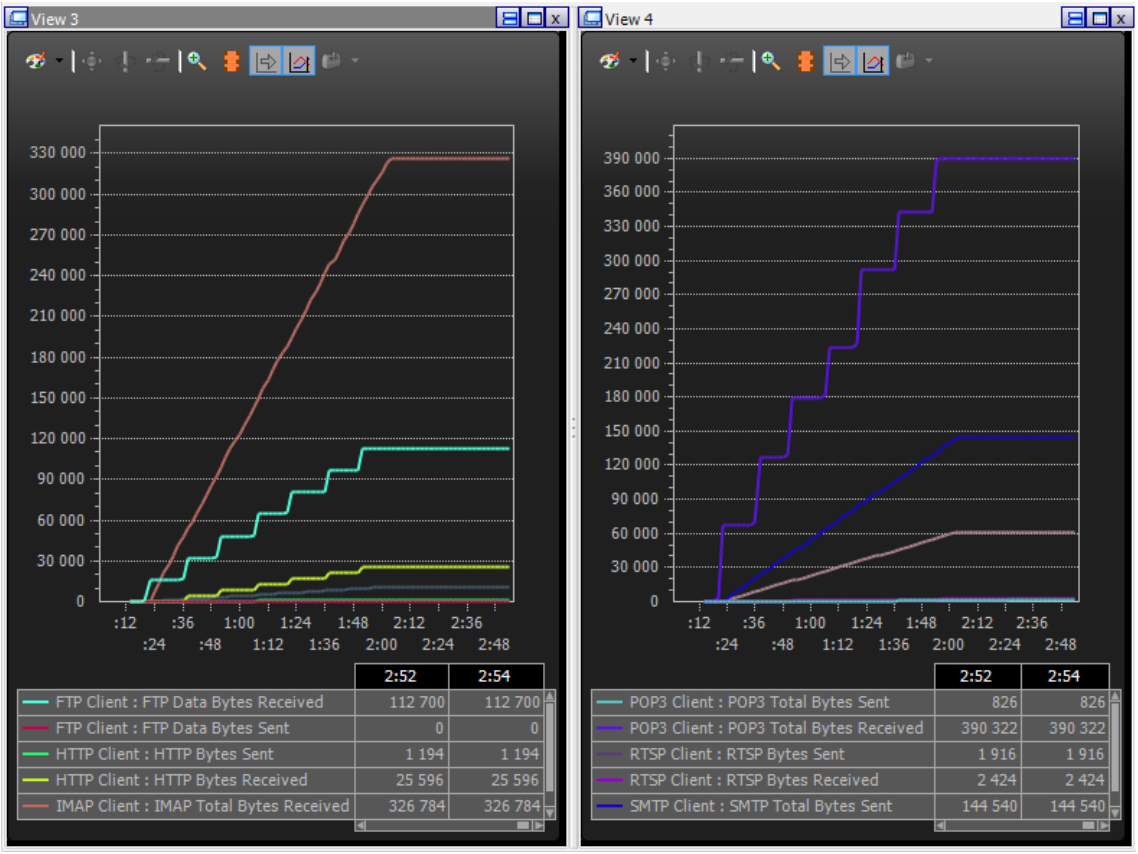

Figure 28: POP3, FTP, HTTP, SMTP, RTSP and IMAP Throughput

4.3 Peer-to-Peer

Purpose: It is essential to test the GGSNs' ability to catch and detect peer-to-peer traffic. There are existing a wide variety of P2P protocols and many of them differs a lot from each other in terms of connection patterns and what transport layer protocol they use. This issue makes it very inefficient to use DPI to identify this kind of traffic, therefore a heuristics based packet inspection with an extensive library of pattern recognition of different P2P protocols is needed to be used. The IxLoad P2P protocol simulation can be used for this purpose. IxLoad has support to replay both BitTorrent and E-donkey, with a library of predefined traffic flows.

Configuration: For this test case, BitTorrent was chosen since it is one of the most used P2P protocols and a large amount of Internet traffic is estimated to be BitTorrent traffic. IxLoad offers a variety of settings of how the data flow should look like. As can be seen in Figure 29, this test case includes one initiator on the Gn interface, one responder on the Gi interface and the dataflow "bidirectional to upload" was selected since it reflects how a user generally interacts with the BitTorrent protocol. The test case is implemented with one subscriber performing the defined command list once. This test case can be easily scaled, by adjusting the number of users and selecting to have an infinite command list.

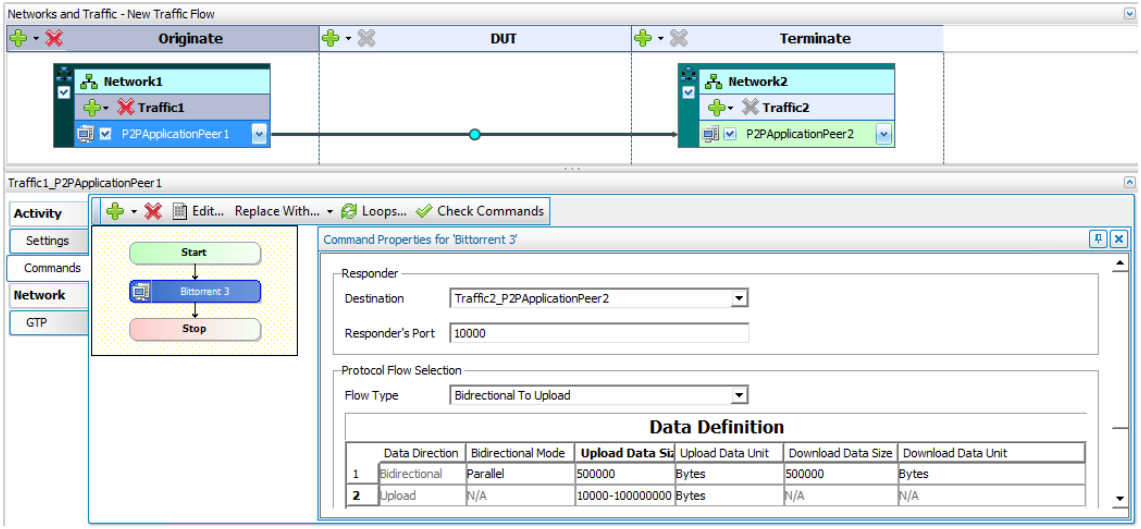

Figure 29: P2P IxLoad configuration

In the scope of this test case, the GGSN is needed to be configured to detect and catch the involved protocol. For this matter, a heuristic traffic inspection was chosen in order to test the ability of IxLoad to mimic and replay a correct BitTorrent procedure.

Verification: To validate that the test is successful, it is important to note that IxLoad only shows the Layer 7 data in the application statistics. It does not take into concern how the data was transported for example over several TCP packets. Therefore, if the GGSN rules counts the involved bearer levels of the inspected packets, the values will not match the values proposed in the IxLoad application statistics. Commonly, the initiator throughput and total bytes sent and received should be viewed in order to get an overview about the P2P flow. These statistics plus the responder throughput are shown in Figure 30.

Figure 30: P2P statistics in IxLoad

4.4 Application Replay

Purpose: The application replay is meant to incorporate non-native protocols into the IxLoad infrastructure. This gives the possibility to easily recreate customer scenarios and to test DPI and heuristic rules of the GGSN for different protocols not present in IxLoad. For instance protocols built on decentralized systems, such as different P2P protocols and chat applications.

Configuration: This test case focuses on simulating one peer on the Gn interface and one peer on the Gi interface and validating that the replayed traffic was correctly identified and classified by the GGSN. For this matter, a traditional HTTP capture was selected as an initial step. This capture includes a full HTTP transaction including the TCP process with SYN and FIN. The capture is first executed with the Application replay – TCP flow and then with the Application replay- IP flow, to see that they could statefully replay the capture. In a case that the capture file includes multiple flows, IxLoad has a filter mechanism to select which flows that should be replayed.

Verification: The application replay module offers the statistics in a very generic way, in terms of flows, connections, initiator/responder throughput and transmitted/received segments. As with the DPI test case, it is also here important that the tester knows what is inside the capture in order to set up the GGSN with coherent rules and draw the correct conclusions about the

amount of traffic that should be classified. The most straight-forward way of verifying an app-replay test would be to capture the traffic in Analyzer and compare it with that capture to see that the transaction match. As can be seen in Figure 31, the application replay statistics can be used to get an overview of the replaying mechanism in order to see if there was a problem with replaying the capture.

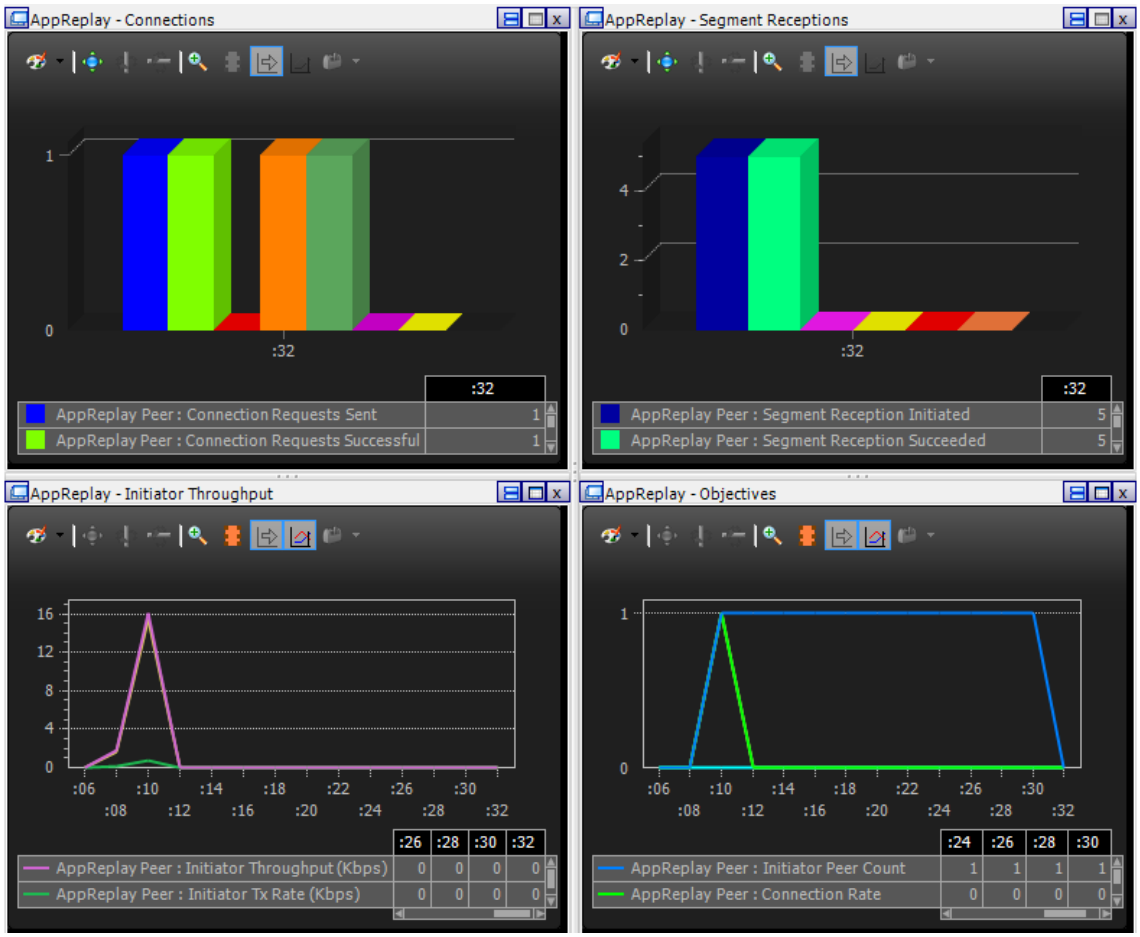

Figure 31: Application replay statistics

4.5 IPv6

Purpose: The current version of Internet protocol, IPv4, has been established in the early 1980s. However, the current size of the Internet and complexity reveals IPv4 inadequacies and imperfections. Consequently, IPv6 has been developed to address these issues by providing more address space, better QoS support, better security, etc. It is generally known that IPv6 is the future for IP networks. IxLoad supports IPv6 as well as IPv4. This test case is created to test the IPv6 functionality of all IPv6-supported protocols over GTP which consists of IMAP, SMTP, POP3, P2P, HTTP and FTP.

Configuration: The GGSN should be configured to handle the IPv6 on the APN and In IxLoad, after creating all related clients under GTP stack, the PDP type should change from IPv4 to IPv6 for specific UE range that is created. In this scenario, only one user for each protocol is simulated with one UE range. Figure 32 shows part of configuration for this scenario.

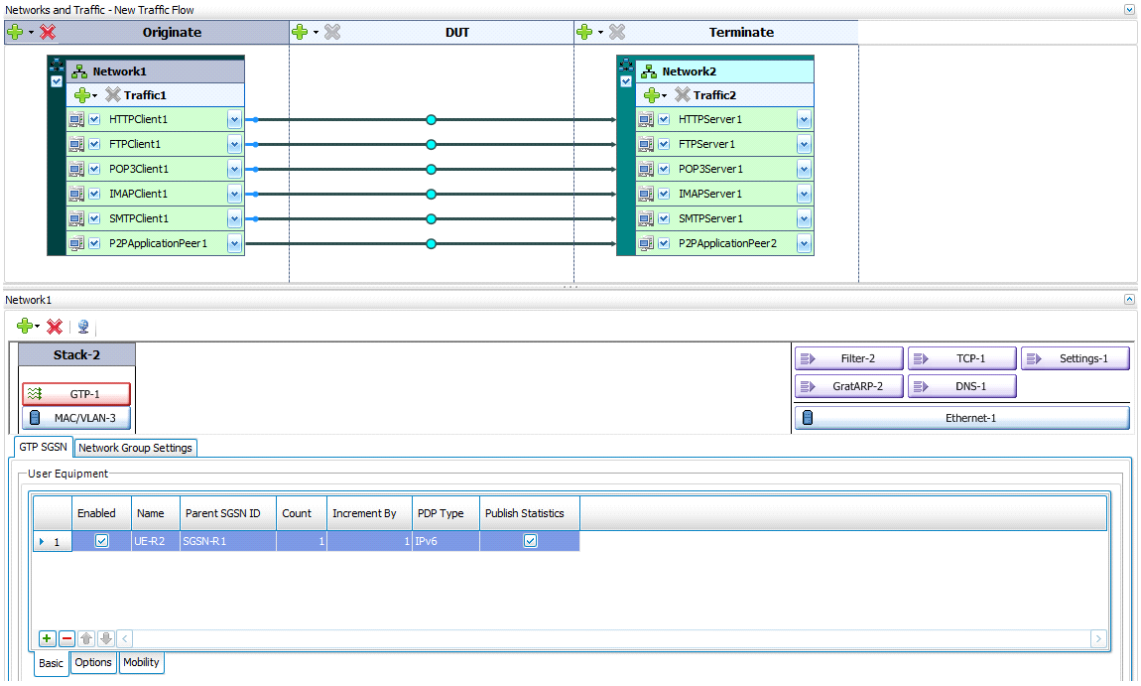

Figure 32: IPv6 UE range setting

Verification: The "GTP General Statistics" provides all information about succeeded and failed sessions. It is also possible to use the drill down feature and view the IPv6 addresses under "IP-GTP_CS" in "GTP per Session statistic". Also, "Throughput Objective" statistic shows the flows for each specified protocol. Moreover, all simulated packets and IPv6 addresses can be viewed through the Analyzer capture files on both Gn and Gi interfaces.

4.6 DTI with user mobility

Purpose: DTI is one of the important GGSN features which significantly improve operators' network capacity. It initiates a GTP tunnel and transfers the user plane traffic directly between RNC and GGSN bypassing the SGSN. Although the user plane traffic does not pass by the SGSN, it is still responsible for user mobility management by handling the control plane traffic. In this scenario, IxLoad simulates users, RNCs and different SGSNs to supports subscriber mobility. The purpose is to evaluate IxLoad capability to maintain user plane traffic as well as keeping track of user mobility.

Configuration: In this scenario, there is no specific configuration for GGSN other than enabling the DTI functionality. Figure 33 shows part of IxLoad DTI configuration where 3 different SGSNs and RNCs are enabled. Also the mobility path should be defined for the current UE range. IxLoad is designed in such a way that the user should start and finish the defined path with the parent SGSN which in this scenario is SGSN-R1. In this defined path, the user first moves from SGSN-R1 to SGSN-R5 and then it comes back again to SGSN-R1 then it moves to SGSN-R7 and finally will come back to SGSN-R1 again. IxLoad has support for four SGSNs per port and only allows UE mobility between SGSNs running on the same port. In this scenario, the number of simulated user is just one and the timeline should be long enough to allow the user to complete the defined mobility path. As it was mentioned before, the number of simulated users can be increased and infinite loop can be enabled to scale the test case.

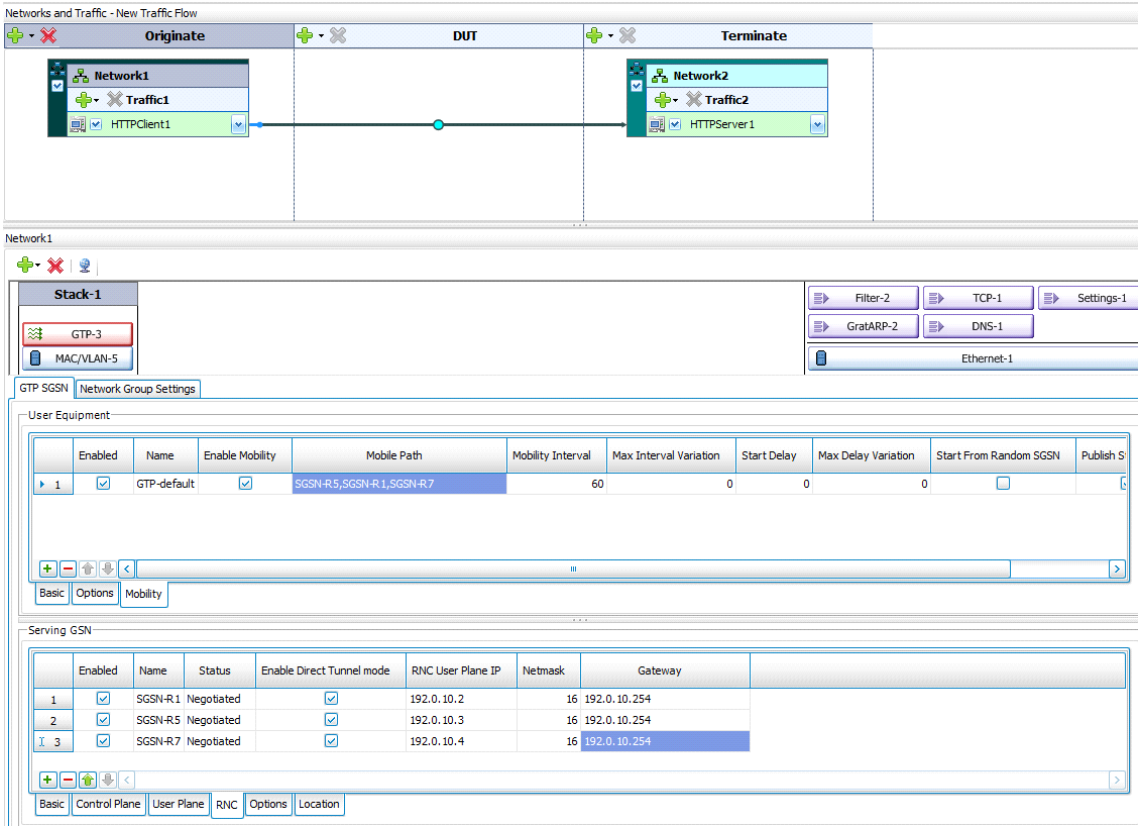

Figure 33: IxLoad DTI setting

Verification: To verify a DTI test scenario, the GTP general statistics should be checked where IxLoad shows the exact number of mobility event together with “DTI Update Initiated” and “DTI Update Complete” values. Figure 34 shows the selected DTI values from all GTP general values. The number of mobility events is equal to five and the “DTI Update Initiated” value is equal to six which consists of 5 mobility event and 1 session initiation.

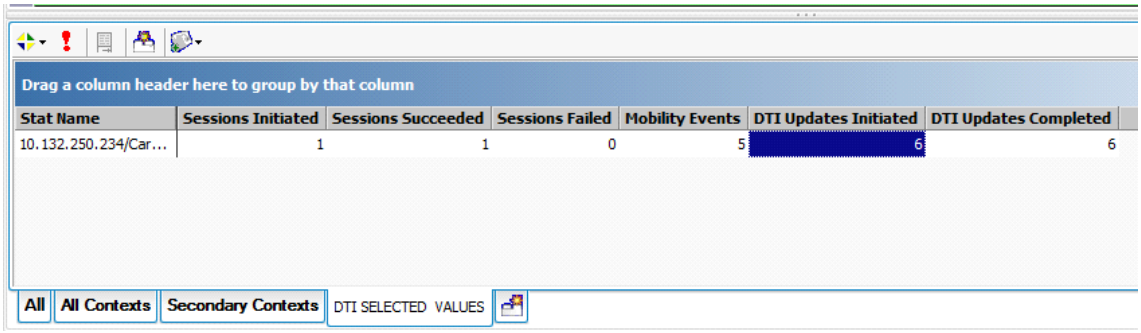

Figure 34: DTI GTP General statistic

Another way to verify the test case is through the Analyzer. The Host section should selected to get an overview of the session management, where the GGSN session controllers IP address can be found. Through the ladder diagram, IxLoad depicts the whole user mobility scenario with SGSNs IP addresses, as can be seen in Figure 35.

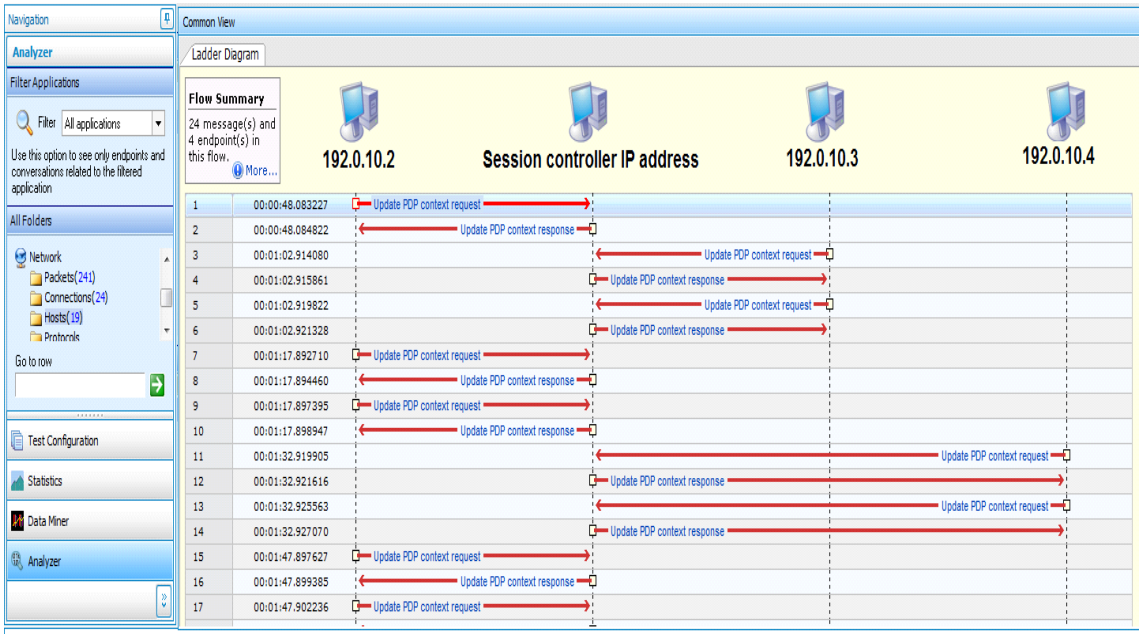

Figure 35: IxLoad DTI Analyzer view

4.7 Pipelining

Purpose: HTTP pipelining means sending several HTTP request without waiting for the responses. The HTTP server will respond to pipelined requests in same order as they are received. HTTP request can be pipelined within a single TCP packet, which results in reducing the network overhead and improving the users loading time, especially over high-latency connections. In this scenario, HTTP pipelined traffic is generated by IxLoad, and GGSN is configured to perform DPI and classify the HTTP pipelined requests.

Configuration: Since HTTP pipelining is supported in HTTP 1.1, IxLoad shall be configured to this version. The aim of this test case it to pipeline 3 GET requests, hence the IxLoad 'Max Pipeline' field value is equal to three. The HTTP command list is composed of three GET command without having infinite loop enabled. It should be noted that 'Transactions per TCP Connection' field should be set to 'Maximum Possible' otherwise it may override the value for the 'Max Pipeline' field. [31] Also, “concurrent TCP connection per user” is set to three. In this way there will be three pipelined requests on each TCP connection. Figure 36 shows part of configuration for this test case. Moreover, it is necessary to enable capture file to be able to keep track of the packets.

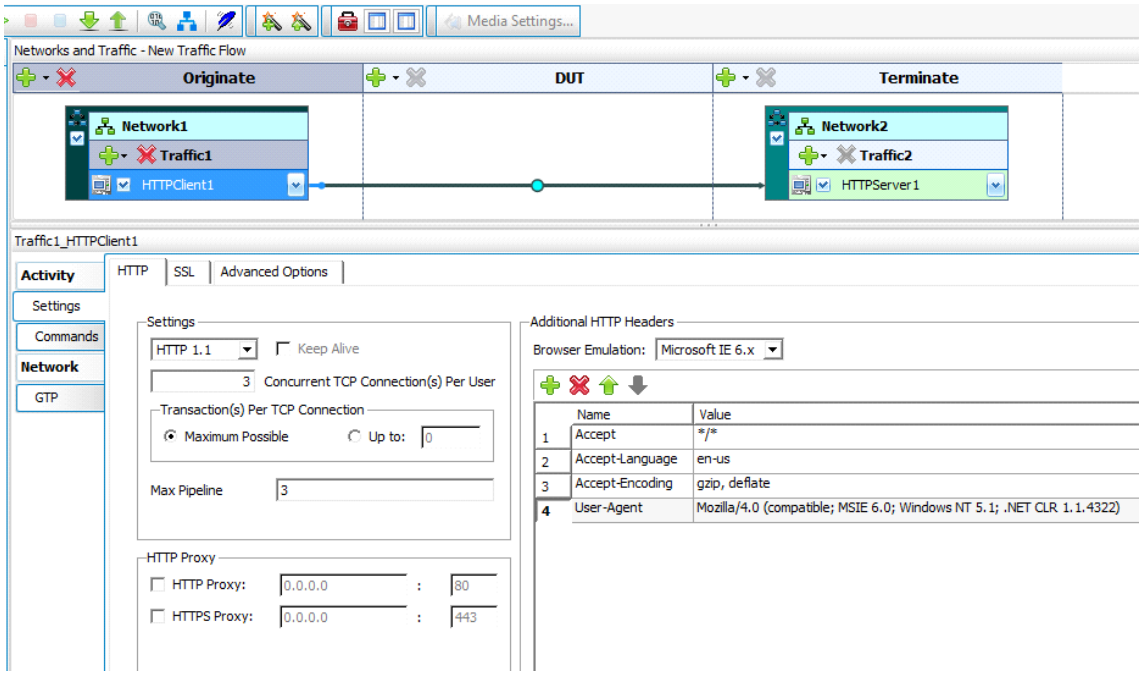

Figure 36: HTTP pipelining settings

Verification: To easily verify the test case, the Analyzer capture files should be filtered to only display the HTTP traffic. By selecting each packet it is possible to view all three HTTP pipeline GET requests and responses through the Analyzer “packet tree”. Since the Gn capture file shows both IP and GTP packets it is recommended to check Gi capture file. Figure 37 depicts client requests and server responses through the Analyzer. Moreover, the GGSN statistics shows the classification of each of the requests.

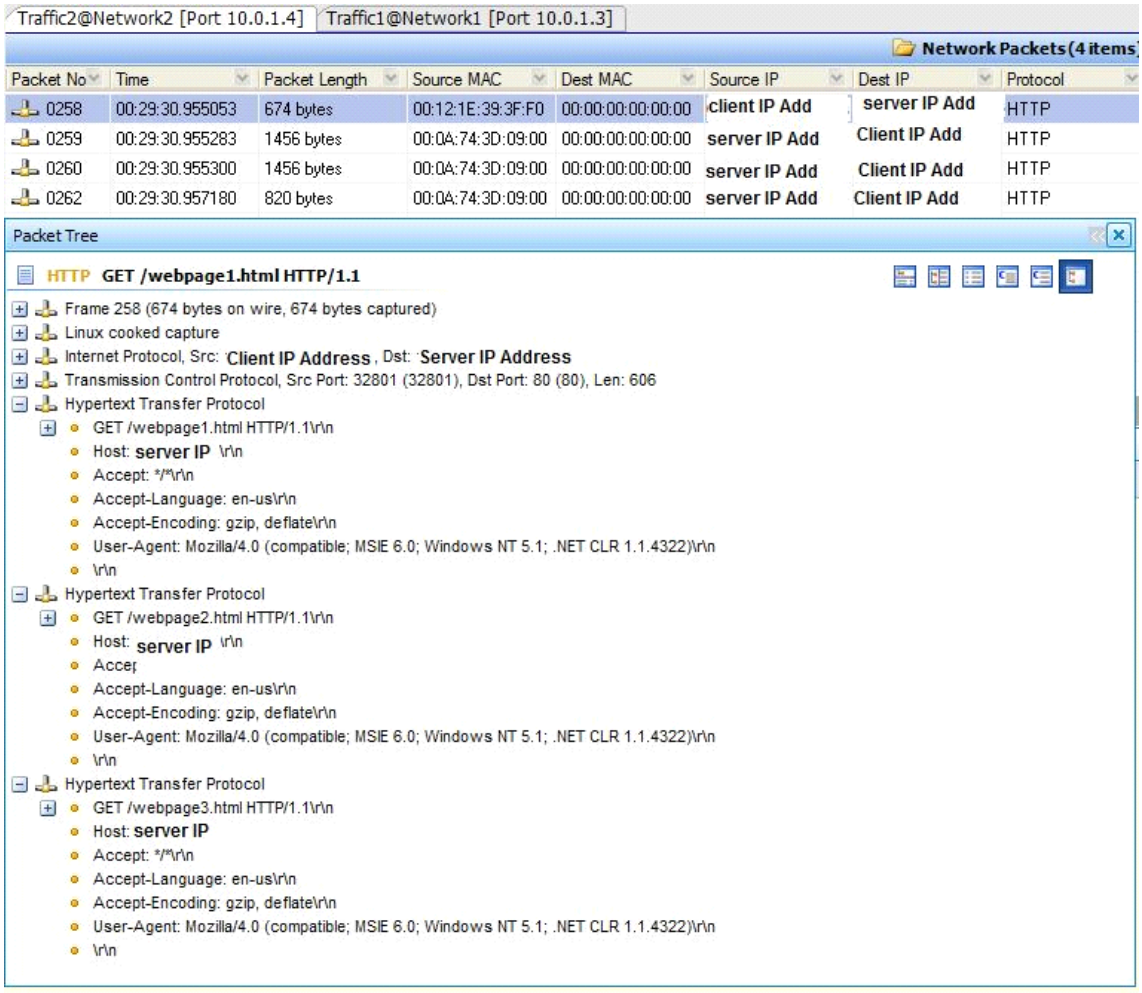

Figure 37: HTTP pipelining verification

4.8 L2TP

Purpose: As it is stated in section 2.6.6, the L2TP tunnelling is one the GGSN functionalities that has several benefits for the operators. IxLoad supports L2TP according to the RFC 2661 [41]. It is possible to implement several different L2TP test cases with IxLoad for different purposes, but the aim of this test case is to implement and evaluate IxLoad L2TP functionalities in a GGSN environment.

Configuration: The GGSN should be configured to act as a LAC and to set up the tunnel to the simulated LNS. IxLoad shall be configured to simulate the SGSN on the Gn side and LNS together with the corporate network on the Gi side. Therefore, GTP stack shall be created on the client side and L2TP stack shall be created on top of the IP stack on the server side. The IP stack shall be configured to match the Gi interface of the GGSN. Moreover, in the L2TP stack the GGSN name shall be specified in the Host field under the L2TP authentication tab. The Role field under Network Group setting shall be change to L2TP Network Server. Also, session number shall be matched to the number of UEs'. Figure 38 and Figure 39 shows some part of L2TP scenario configuration on IxLoad. The aim of the test is to define a trivial L2TP-tunnel in co-ordinance with a simple HTTP test case in order to see that the expected outcome matches the GGSN way of working. As an addition to the regular L2TP tunnelling, IxLoad has support for user authentication with both PAP and CHAP with name and secret available for configuration.

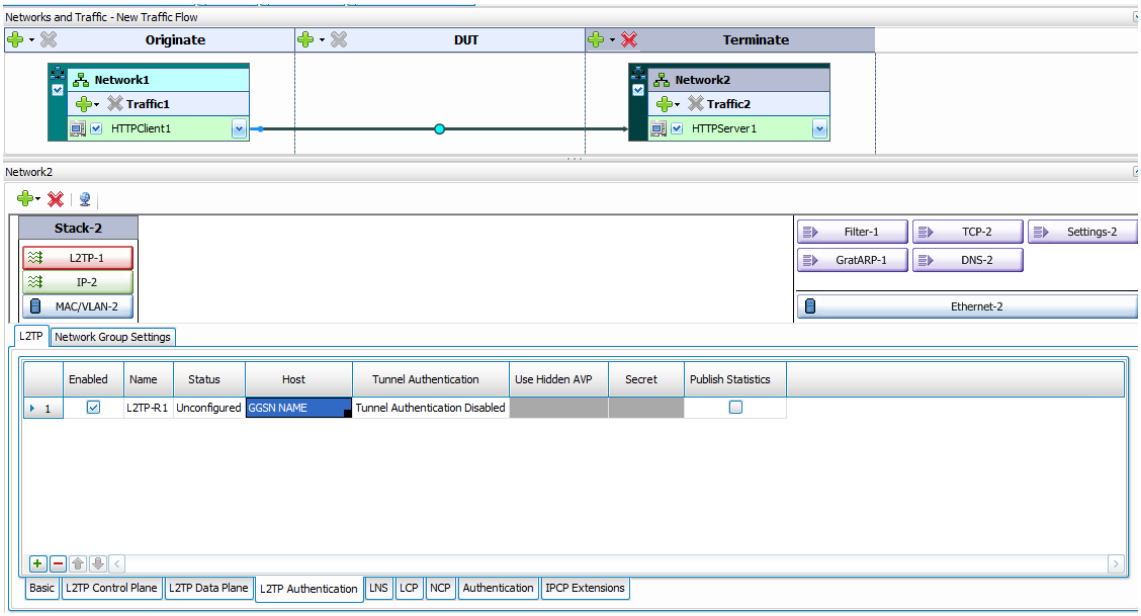

Figure 38: L2TP Stack Configuration in IxLoad

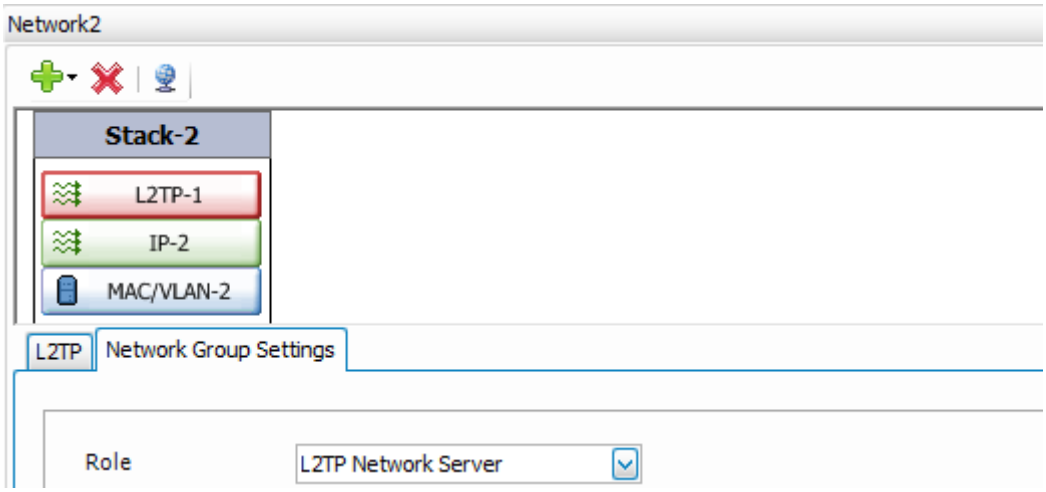

Figure 39: Network Group Setting for L2TP scenario

Verification: When verifying an L2TP test case, the quickest way of seeing a successful flow is to view the HTTP throughput, however more in-depth statistics are needed in more advanced test cases. IxLoad offers a various amount of statistics for L2TP testing, as can be seen in Figure 40. In this context, it was chosen to display the L2TP setup statistics and the L2TP Tunnel statistics which shows the L2TP tunnel activity and L2TP messages transmitted and received.

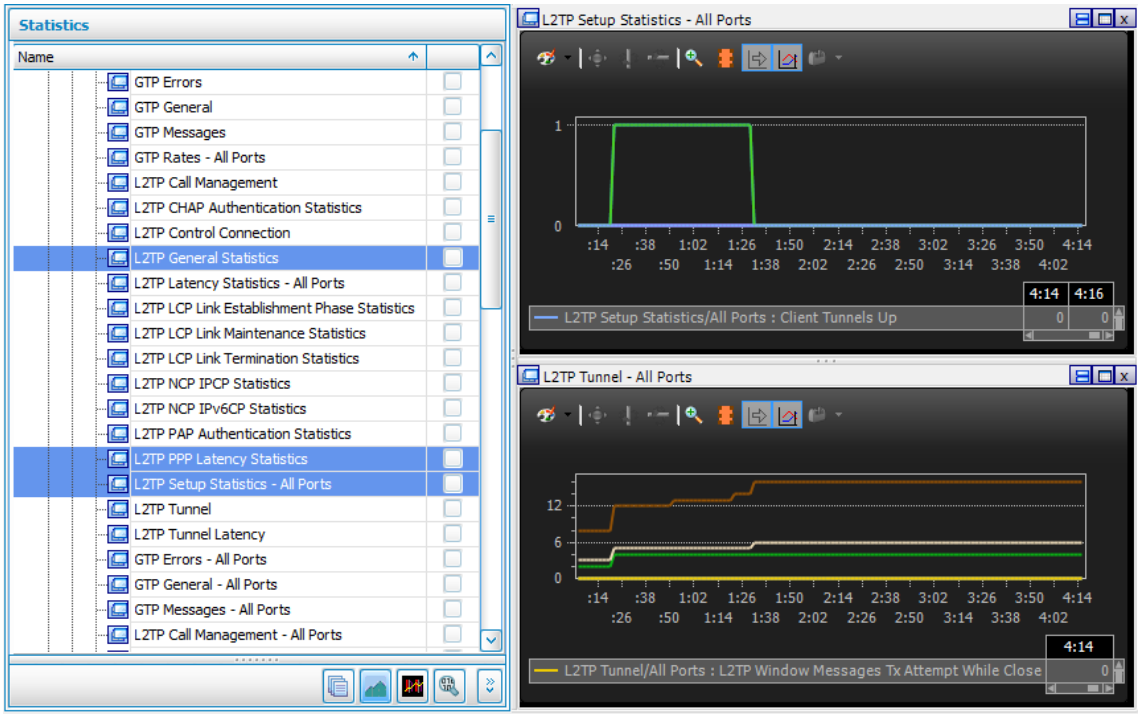

Figure 40: L2TP Statistics

4.9 Different GGSN restart scenarios

Purpose: This section consists of several different test cases. The main purpose to implement GGSN restart scenarios is to investigate the dynamic behaviour of IxLoad during different GGSN restart procedures. It is essential for a good test tool to have the capability to understand and cope with various GGSN restart situations. Through the following test cases the IxLoad dynamic session management towards the GGSN is evaluated.

Configuration: The GGSN node that is used for these test cases has one Node Controller, three Session Controllers, one user data traffic forwarder (U-pic) and one spare PIC in standby mode. For most of the test cases, IxLoad has been configured to create 5000 PDPs with a simple HTTP scenario with infinite loop. The following are the different GGSN restart test scenarios:

1. **Offline the Node Controller on the GGSN**: During this scenario the GGSN Node Controller is taken offline while IxLoad was in the sustain time phase. This means that IxLoad should create 5000 PDP contexts during the Ramp up phase and during the sustain time, the HTTP application is executing steadily.

2. **Restart the only Session Controller on the GGSN**: Before the test, two of the Session Controllers and the spare are offlined. During the sustain time, the remaining GGSN Session Controller is offlined and after 5 minutes it is brought online again. It is expected that IxLoad detects the Session controller restart and acts accordingly.

3. **Restarting the GTPC[1] process on the Session Controller:** In this scenario only one

[1] GTPC is the process which handles the PDP sessions in the Session Controller

Session Controller is used. During the sustain time, the GTPC process should be restarted several times, in order to verify the session management of IxLoad.

4. **Offline one Session Controller:** In the beginning of this test case, all the session controllers are online. During the sustain time phase, one session controller is offlined, while the other two are kept online, in order to test the dynamic session reestablishment of IxLoad, because part of the GGSN is still working.

5. **Restart GGSN FPCs:** This test is to verify the ability of IxLoad to understand that the GGSN is restarted and still being able to resume traffic when the GGSN is ready. This test case was executed with an objective value of 35000 PDP contexts. When the restart-command is executed, the IxLoad emulated SGSN should internally delete all the active PDPs and continuously send “create PDP context request”–messages towards the GGSN to see if it comes up again. When the GGSN is restarted and is online again, it will start to reply back. At that instance, IxLoad will resume the applied traffic model and reinitiates PDP context up to the specified objective value. One important observation is that even though the GGSN is online again, it might not be ready to handle PDP contexts, so it might respond with “No resources available”. If this occurs, the IxLoad emulated SGSN will not retry the session initiation process for these contexts.

6. **Restart GGSN by using request system restart:** This test case is aimed to verify the same behaviour as the previous test case, but with 10-15 minutes restart procedure. This test case targets the IxLoad solution for detecting that the GGSN has returned online after a long time.

7. **Restart GGSN by using request services GGSN restart node:** This restart procedure is only 10 seconds long and the GGSN will send “delete PDP context request” for all the active PDPs before being restarted. This test case stresses IxLoads' ability to cope with a lot of signalling traffic and its reaction time until it understands that GGSN has restarted.

Verification:

1. **Offline the Node Controller on GGSN:** When the Node Controller goes offline, the Spare should take over the Node Controller role. Since the Node Controller is not involved in the active PDP sessions, the PDPs should be still active and the IxLoad simulated SGSN should not terminate the active sessions. To verify this scenario the number of active PDPs in both IxLoad and GGSN shall be checked and compared. IxLoad active PDP numbers can be found under GTP General Statistic. Moreover, the graphs for HTTP throughput and GTP rate shows that IxLoad is keeping track of GGSN conditions.

2. **Offline the only Session Controller on GGSN:** In this scenario, IxLoad should terminate all the sessions as soon as it detects that the only Session Controller is offline. Therefore, the IxLoad warnings should be checked to verify that IxLoad understands that the GGSN has stopped responding, as shown in Figure 41. At this point, IxLoad tries to reinitiate the sessions, but as it is shown in GTP General, all the initiated sessions are failed. The reason for this is the GGSN is responding with the error “No Resources Available” and the IxLoad emulated SGSN is only reinitiating

sessions that failed with the error cause “Context Activation Failure timeout”. The conclusion is as long as the session are failed with a cause, other than timeout, the IxLoad emulate SGSN will not retry them.

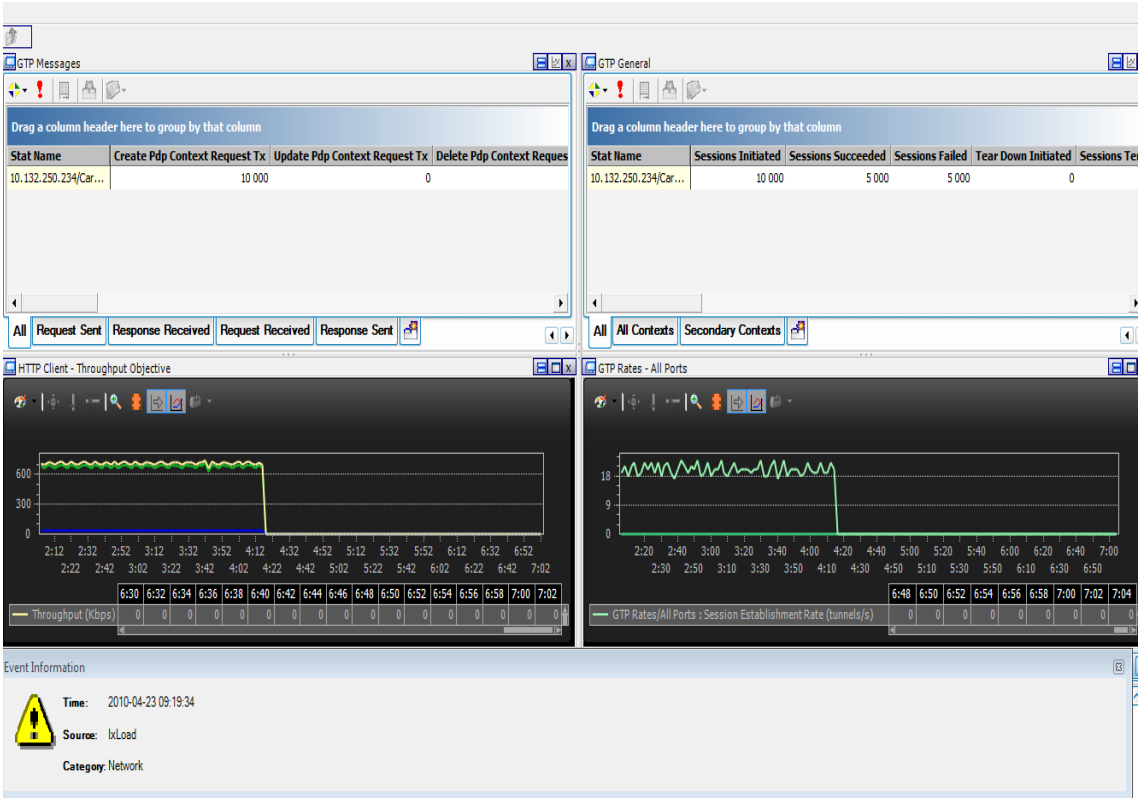

Figure 41: IxLoad statistic after taking the only Session Controller offline

3. **Restarting the GTPC process on the Session Controller:** After restarting the GTPC process on the GGSN, it will take down all the active PDPs and send “Delete Pdp Context Request”- messages towards IxLoad. IxLoad will detect that the GGSN has restarted and will display the warning “GGSN restarts”. As it is shown in Figure 42 both HTTP throughput and GTP Rates graph are affected by the GTPC restarts. When the restart occurs, all the 5000 IxLoad initiated sessions are terminated and reinitiated again and depending on the number of times the GTPC is restarted, the IxLoad session initiated statistics will be increased.

Figure 42: IxLoad GTPCD restart statistics

4. **Offline one Session Controller:** During this test case, the Node Controller on the GGSN will divide the 5000 active PDPs between the three Session Controllers. When one of them is offlined, the expected behaviour from the simulated SGSN in IxLoad is to detect it, terminate the associated sessions and then reinitiate them towards the GGSN Node Controller. After that the Node Controller will divide the reinitiated PDPs between the two remaining Session Controllers.

Figure 43: IxLoad statistics after offlining one of the three Session Controller

As shown in Figure 43, 2044 sessions, which were assigned to the offlined Session Controller, are terminated. Also the sessions initiated are incremented with the equal amount, since IxLoad will reinitiate the terminated session with the remaining Session

Controllers. The red peak in GTP Rates graph shows the rate of the create PDP request sent from IxLoad simulated SGSN towards the GGSN.

5. **Restart GGSN FPCs:** For this scenario, the verification is to observe the procedure described in the configuration part of this test case and validate that the corresponding statistics are updated accordingly. Due to the reasons mentioned in the configuration paragraph, the statistics for “Create PDP Context Request”, “Active Sessions”, Context Activation Failure (CAF), and “Deleted PDP (Peer Dead)” should be kept under surveillance, as can be seen in Figure 44. The “CAF Timeout” statistics are incremented due to the reason that the IxLoad emulated SGSN is continuously sending “Create PDP Context Request”-messages when it has identified that the GGSN is not responding. The “CAF No resources Available” is incremented as a result of that the GGSN is online but is not able to handle payload yet.

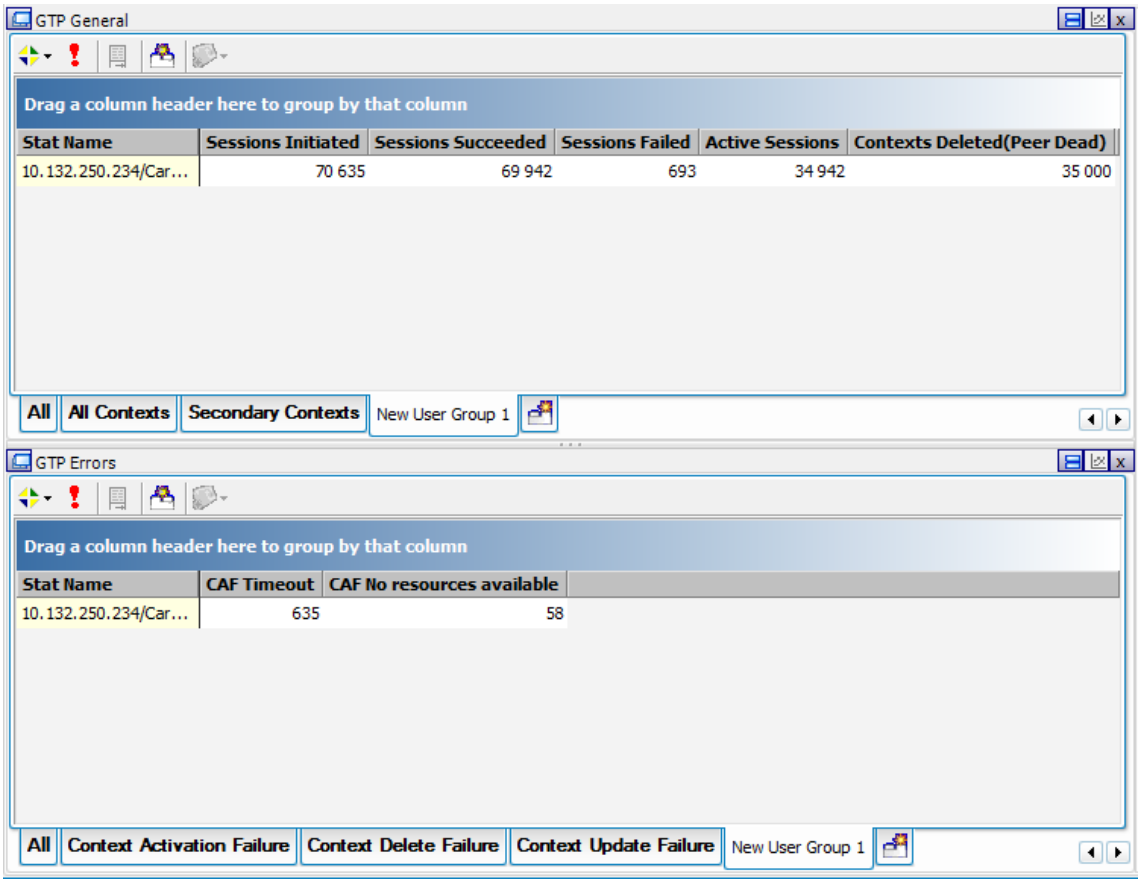

Figure 44: IxLoad statistics after FPC restart

6. **Restart GGSN by using request system restart:** Validating this test case corresponds to validating the “Restart GGSN FPCs” test case. However an observation is made, when the GGSN is restarted, there is a long interval between the state that the GGSN is starting up and the state when it is able to process PDP contexts. This results in that the IxLoad emulated SGSN receives “No Resources Available” for the 5000 PDPs, for this reason, the simulated user objective was increased to 25000 PDPs and re-executed, producing Figure 45.

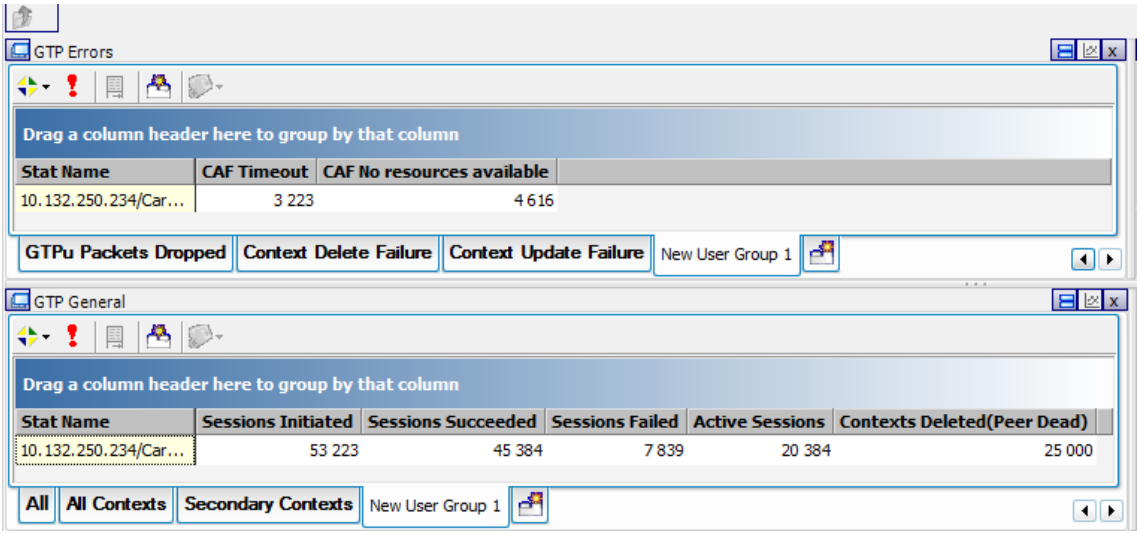

Figure 45: IxLoad statistics after a restart

7. **Restart GGSN by using request services GGSN restart node:** The validation of this test case is to observe that IxLoad receives “Delete PCP context request” to all the involved PDPs and that it will try to re-establish them. However, the GGSN will not respond immediately since it is in the process of restarting, so IxLoad will continuously send “Create PDP Contexts Request”-messages until GGSN is responding. When the GGSN is online again, the emulated SGSN will look for the restart counter and see that it has been increased, thus deleting all its on-going PDPs and reinitiate them to the newly restarted GGSN. The expected behaviour can be seen in Figure 46 where the dark blue line depicts the establishment of GTP tunnel and the light blue shows the termination of GTP tunnels.

Figure 46: IxLoad statistics for GGSN fast restart

4.10 QoS control

Purpose: IxLoad supports QoS in GTP specified in the 3GPP standards, TS 23.107 and TS 24.008.V7.6.0 [34], meaning that both the emulated UE and SGSN should adhere and respect the negotiated QoS parameters. The purpose of this test case is to evaluate IxLoad QoS functionality over GTP stack. Different QoS parameters exist, such as delay class, reliability class and Maximum Bit Rate (MBR) for both uplink and downlink. Currently, IxLoad has the possibility to create QoS parameters and send them to the GGSN in Create PDP Context requests. It can also learn the QoS parameters from both Create PDP Context Responses and Update PDP Context Requests and dynamically adjust the rate of data sent based on Maximum Bit Rate for Uplink received.

Configuration: This test case is more clearly described if it is divided into two parts:

QoS enforcement: IxLoad SGSN simulator will learn QoS parameters from Create PDP Context Response message and will adjust the transmit rate based on “Maximum Bit Rate for Uplink”

Dynamic QoS: IxLoad SGSN simulator will respond to Update PDP Context Request and dynamically adjust rate of data sent based on “Maximum Bit Rate for Uplink” received.

Two different approaches were chosen to verify these test cases, the first one is implemented only with the involvement of a GGSN, and the second is making use of a GGSN and a PCRF.

With QoS enforcement, the emulated SGSN is initiating a PDP context with QoS parameters, requesting Interactive as the traffic class and 8640 kbps as the MBR for uplink, while the GGSN is configured to grant a MBR of 2048 kbps. The test case is based on a simple FTP test case with a FTP GET request in an infinite loop in order to make the data transactions stable without a lot of jitter.

As stated above, the Dynamic QoS test case is involving a PCRF, which is in charge of the policy control. An overview of the test scenario is shown in Figure 47.

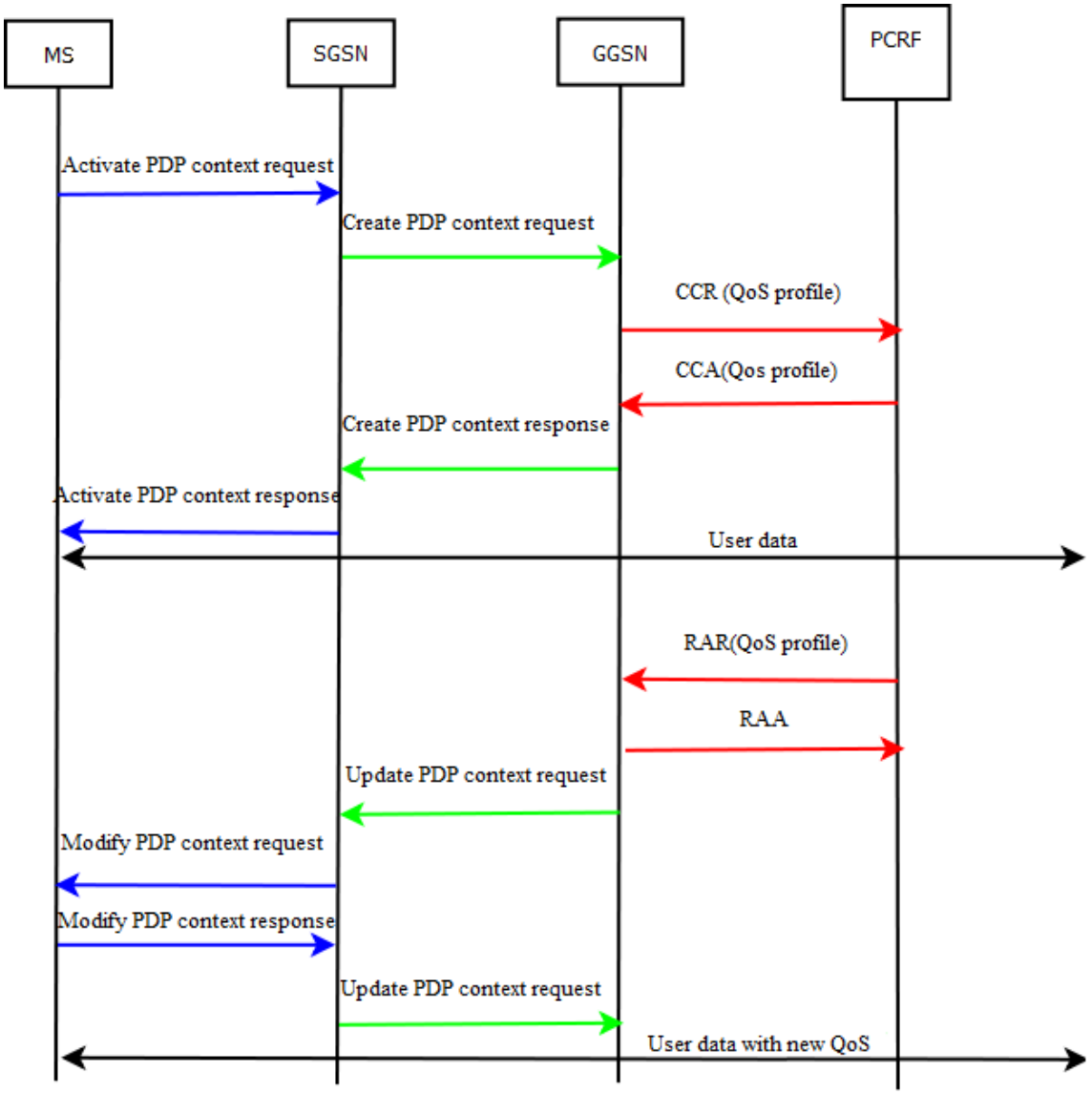

Figure 47: Dynamic QoS test scenario

The scenario is slightly more complicated than the regular session management shown in Figure 5 described in section 2.5. The GGSN has been configured to send the QoS profile to the PCRF and the PCRF should check that this matches its configured values for that particular user. Either the PCRF accepts the requested profile or it attaches the QoS profile in the Credit-Control-Answer. The PCRF has been further configured to send a RAR (Re-Authentication Request) with a changed QoS profile to the GGSN after a certain time has passed. GGSN should adhere to the PCRF and send an update request towards the SGSN, with the new QoS profile, meaning that an update PDP context request procedure has been initiated. When the update procedure is finished, both the SGSN and the MS are informed about the new QoS and can start to send user data that respects the new QoS profile. For this particular test case the SGSN is configured to request a MBR of 8192 kbps and the PCRF states an MBR of 2048 kbps in the initial request. The PCRF is configured to send a RAR-message, including a QoS profile stating a MBR of 1024 kbps, 60 seconds after the initial request.

Verification: For the QoS enforcement, the verification can be done in IxLoad or on the GGSN, by monitoring the throughput and see that these average values match the desired configuration. An example is shown in Figure 48, where the GGSN is configured to grant an uplink MBR of 2048 kbps.

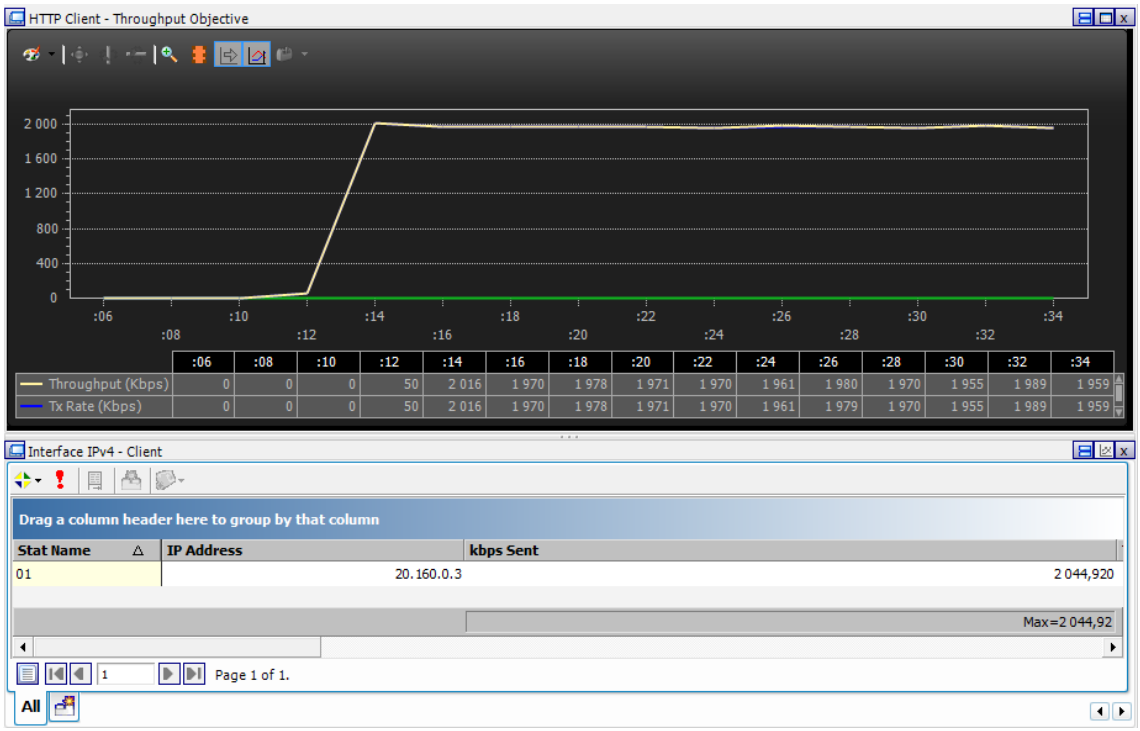

Figure 48: QoS enforcement statistics

For the dynamic QoS, the validation procedure is almost the same. Before the PCRF initiated QoS modification, the throughput should match the negotiated MBR during the create PDP procedure and after the update procedure, the throughput should match the updated MBR. An example of the throughput graph can be seen in Figure 49.

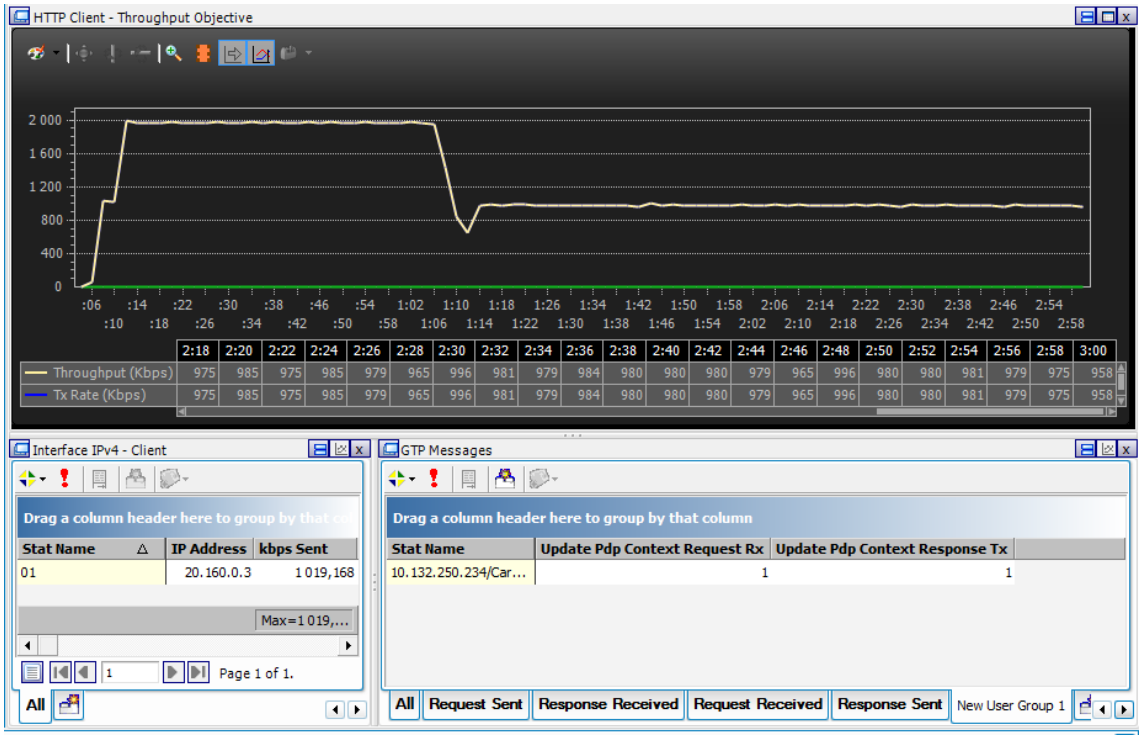

Figure 49: Dynamic QoS including a QoS update

4.11 Multi-purpose test

Purpose: This test case is developed to pressure IxLoad ability to handle multiple functionalities within the same test. It is aimed to verify that the essential core structure of IxLoad is working whenever Ericsson receives a new software version.

Configuration: This test case includes:

- Multi-application users for testing the deep packet inspection as well as P2P-detection by heuristic method.
- A HTTP user working over DTI followed by a mobility procedure towards other SGSNs.
- A HTTP client configured to pipeline its GET requests.
- A FTP client working over IPv6.
- A HTTP user responding to dynamic HTTP requests coming from the GGSN and actively responding and pursuing the new locations.

Each of the above entities is separated into different UE ranges to easily differentiate the variety of the simulated users. Furthermore, the test case should be run with capture enabled.

Verification: This test case is different from the previous test cases, since it requires the tester to be more involved in the verification procedure. The following is the set of actions that the tester needs to perform:

- Verify the GGSN DPI statistics corresponds to what IxLoad is sending.
- Perform the verification procedure described in Figure 35: IxLoad DTI Analyzer view in which the tester can verify the DTI and mobility path of the HTTP users.
- For verifying the HTTP pipelining, the user should check that the HTTP GET requests, in

the Analyzer, has been sent correctly, receiving the corresponding responses and making sure that the GGSN statistics are incremented accordingly as it is stated in section 4.7.

- Ensure that the IPv6 user is successful by following the recommended verification process in section 4.5.
- To verify HTTP dynamic redirect functionality, one way is to verify this statistically by looking at the client statistics and compare it to server statistics, especially the HTTP responses sent and received. Another way is to identify the flow of the capture file in analyzer. Both are described thoroughly in section 4.1.

4.12 IxLoad performance benchmarking

The IxLoad GTP performance in GGSN environment is evaluated by the following test cases. All the test cases have been conducted on a XM2 chassis with two ASM1000XMV12 load modules. Table 2 depicts the IxLoad ASM1000XMV12 back to back performance numbers for some protocols, per port and per load module, which is provided by the IXIA support team. The term back to back is used when two IxLoad ports are inter-connected without having the device under test in between. For back-to-back GTP performance testing, the SGSN and subscribers are emulated on port 1 and port 2 emulates both a GGSN and IP server.

The following performance test cases have been chosen to verify the IxLoad performance numbers but having GGSN as the DUT:

- Maximum PDP contexts for HTTP and FTP per port.
- Maximum throughput – FTP with 1000, 7000 and 25000 PDP context per port.
- Maximum throughput – HTTP with 1000, 7000 and 25000 PDP context per port.

All the test cases have been configured in dynamic mode on client and static mode on the server side. For benchmarking IxLoad, a high performance GGSN is used for all the test cases.

	Per Port CPU ASM1000XMV12	**Per Load Module ASM1000XMV12**
Max GTP-c transaction rate	175 transactions/second	2100 transactions/second
Max PDP Contexts - ftp	25530	306360
Max PDP Contexts – http	50000	600000
Max PDP Contexts – Video on Demand (VOD)	2000	24000
Max PDP Contexts – POP3	32000	384000
Max PDP Contexts – SMTP	40100	481200
Max PDP Contexts – IMAP	35200	422400
Max throughput – ftp (1K PDP contexts per port)	580 Mbps	6.9 Gbps
Max throughput – ftp (7K PDP contexts per port)	450 Mbps	5.4 Gbps
Max throughput – ftp (25K PDP contexts per port)	240 Mbps	2.88Gbps
Max throughput – http (1K PDP contexts per port)	600 Mbps	7.2 Gbps
Max throughput – http (7K PDP contexts per port)	500 Mbps	6 Gbps
Max throughput – http (25K PDP contexts per port)	380 Mbps	4.5 Gbps

Table 2: IxLoad back to back performance for ASM 1000XMV12 provided by IXIA

1. **Maximum PDP contexts for HTTP:** This test verified the maximum number of PDP context that IxLoad is able to maintain for a simple HTTP scenario. The test has been executed several times with 20 minutes sustain time. The activity consists of one HTTP 1.1 GET request, asking for a 1 kilobyte object followed by a long think command.

Results: As stated in the Table 2, the back to back value for this test case is 50000 PDP contexts. IxLoad was able to execute the test with the mentioned value, but when executing the test with a higher objective value the IxLoad failed to configure the test and was not able to execute it.

2. **Maximum PDP contexts for FTP:** This test verified the maximum number of PDP context that IxLoad is able to maintain for a simple FTP scenario. The test has been executed several times with 20 minutes sustain time. The activity consists of one FTP GET request, asking for 1 kilobyte object, which is followed by a long think command.

   Results: IxLoad was able to execute the test with the value stated in Table 2, which is equal to 25530 PDP contexts. However, when the objective value was increased, IxLoad failed to configure the test and was not able to execute the test.

3. **Max throughput for FTP** This test verified the maximum throughput for different amount of PDP context for FTP protocol in a GGSN environment. A simple FTP scenario was used and the objective type that has been chosen for all maximum FTP throughput test cases is “concurrent sessions”. This is because of the nature of the protocol, in which most of the payload is being transferred in the FTP-data sessions. The idea of testing with different amount of active PDP contexts is to test IxLoad internal structures of handling the separation of the user data. Each contexts’ internal TCP buffer size needs to be inversely scaled to the increased PDP contexts in order to achieve the maximum throughput. Additionally, the GET request object size needs to be scaled. This is because of the limited memory space on the IxLoad ports, since each PDP contexts requires having an optimal TCP buffer. If the amount of PDPs is increased the TCP buffer is needed to be decreased, in order to achieve an optimal overall throughput. The test case is a simple FTP scenario with one FTP GET request as the activity together with loop enabled and 20 minutes sustain. The test cases are separated by the following differences:

   - **FTP with 1000 PDP contexts:** 1048576 byte object size with client TCP transmit buffer and server TCP receive buffer equal to 25000 bytes.
   - **FTP with 7000 PDP contexts**: 1048576 byte object size with client TCP transmit buffer and server TCP receive buffer equal to 16000 bytes.
   - **FTP with 25000 PDP contexts:** 65536 byte object size with client TCP transmit buffer and server TCP receive buffer equal to 4096 bytes.

   Results: The outcome of this test was 530 Mbps for 1000 PDP contexts, 500 Mbps for 7000 PDP contexts and 250 Mbps for 25000 PDP contexts. The results and its correlation to the values in Table 2 are shown in Figure 50. These performance numbers is the average of several executions.

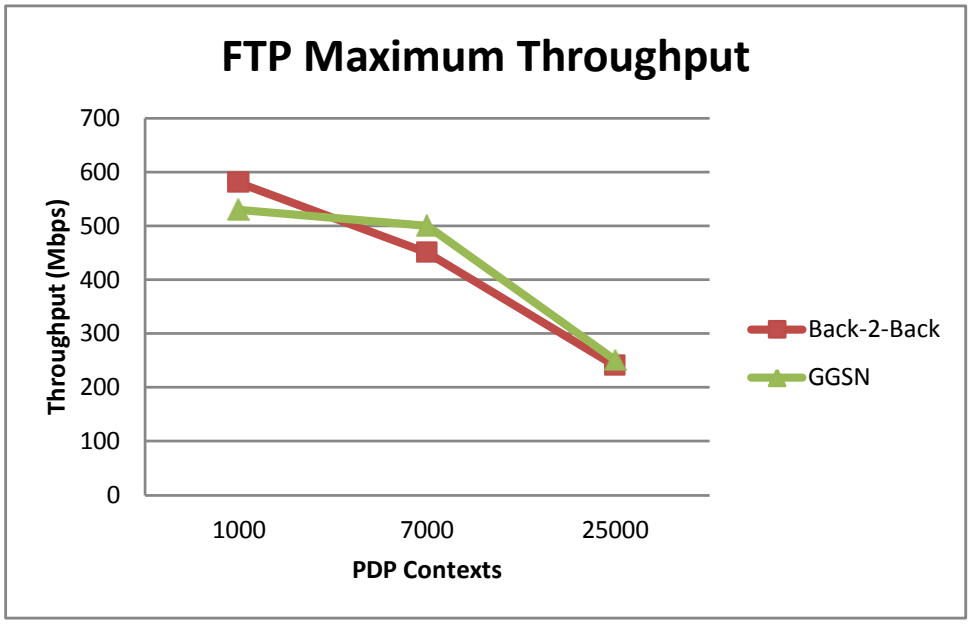

Figure 50: FTP Maximum Throughput per port

4. **Maximum throughput for HTTP:** This test verified the maximum throughput in a GGSN environment for different amount of PDP context in a simple HTTP scenario. The HTTP version 1.1 is used, one HTTP GET request with loop enabled and 20 minutes sustain time. The objective type that has been chosen for all the HTTP maximum throughput test cases is “concurrent connection”, since the maximum amount of concurrent TCP connections will achieve the highest performance. Each test has been executed several times with the following differences of parameters :

    - **HTTP with 1000 PDP contexts:** 1024 kilobyte object size with client TCP transmit buffer and server TCP receive buffer equal to 32768 bytes.
    - **HTTP with 7000 PDP contexts:** 1024 kilobyte object size with client TCP transmit buffer and server TCP receive buffer equal to 16384 bytes.
    - **HTTP with 25000 PDP contexts:** 128 kilobyte object size with client TCP transmit buffer and server TCP receive buffer equal to 8192 bytes.

Results: The result of the test cases is 700 Mbps for 1000 PDP contexts, 600 Mbps for 7000 PDP contexts and 400 Mbps for 25000 PDP contexts. The results and its correlation to the values in Table 2 are shown in Figure 51. These performance numbers is the average of several executions.

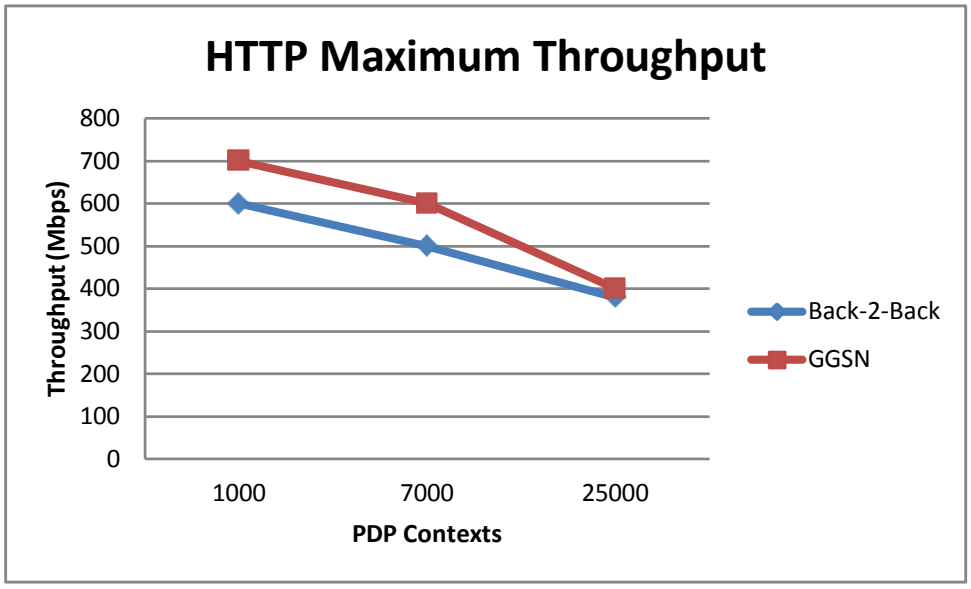

Figure 51: HTTP Maximum Throughput per port

As the figures 50 and 51 shows, the IxLoad performance is not decreased significantly in a GGSN environment for both FTP and HTTP. IxLoad achieved higher performance numbers with both 7000 and 25000 PDP contexts compared to the back to back testing for FTP. Also, all the HTTP values are higher than the back to back results. The reason can be the limitations of the simulated GGSN used by IXIA. Because during their benchmarking procedures, they let IxLoad simulate the GGSN and this might lead to a decrease of the performance of IxLoad.

5 Suggested improvements of IxLoad usability

In this chapter, several suggested solutions meant to increase and expand IxLoad capabilities are going to be presented. The first section will propose new functionality to be added to IxLoad in order to meet the expectations of a proper test tool in a GGSN environment. In the second section, IxLoad graphical user interface concerns are discussed. The last section is about areas that IxLoad needs to supply the users with up-to-date information in order to get better understanding of the test tool capabilities and features. The need for enhancements of existing functionalities arose throughout the implementation of different test cases in chapter 0. Moreover, the different subjects in this chapter were identified during the assessment phase of this thesis. Finally, each of the different paragraphs aims to explain one enhancement or issue.

5.1 Additional functionality

IxLoad should support WAP over GTP to be fully usable in the mobile packet core. This might be considered an old and dying technology in the modern world, but it is still widely used in developing countries. As stated in a survey report done by China Internet Network Information Center (CNNIC) there was 39 million WAP users in China as of March 2007. [42] In 2008, BBC news statistics showed that WAP is one of the top five platforms that African countries are using, by that time the mobile market had over 48 million subscribers. [43] By the early 2009, Rick Joubert, head of mobile advertising at Vodacom estimated that there are more than 10 million mobile Web users using WAP in South Africa. [44] New functionality to the packet core should still adhere to this technology and a proper tool for a GGSN must support WAP to create realistic test cases and to recreate certain user specific problems. For IXIA, it should be fairly convenient to at least implement WAP2 since it is based on HTTP.

When WAP is introduced, the next step is to add a MMS-module. This protocol is used world-wide and allows the users to send multimedia content to and from the mobiles. With WAP and MMS included, IxLoad will be more complete and usable in a packet core environment and it will become more competitive towards other packet core test tools that support these protocols.

IxLoad have a vulnerability module and a DDOS module that can be used for testing the security of a DUT. However, they are not supported over GTP. They are only usable on the Gi interface of the GGSN, but cannot be used to simulate the mobile users who are initiating attacks towards the GGSN or to the adjacent servers on the Gi interface. Implementing these modules over GTP will enable IxLoad to be used as an attack tool towards GGSN along with regular traffic on the network. This would increase IxLoad usability and makes it more applicable in a more realistic scenario where attacks are present. For example, it can be used for testing what happens with the GGSN when it receives a TCP SYN flood or a TCP RST flood from malicious users.

The IxLoad homepage states that IxLoad supports both IPv4 and IPv6. RTSP and Application Replay are two important modules which are not supported over IPv6. [36] RTSP is meant to be used for streaming applications which will become more vital in the future of the mobile evolution. Application Replay is meant to replay capture files, which are taken to mimic customer related issues or scenarios, and if a client would be using IPv6, IxLoad cannot be used to recreate that scenario. Therefore, both mentioned modules should be further expanded to incorporate IPv6. The IPv6 is the next generation of IP networks and as its deployment is spread, more people are going to use it. The GGSN customers will expect to have the same functionality available for IPv6 as for IPv4. Thus, for all applications, IPv6 should be present in a

GGSN test tool to ensure that GGSN can provide operators with the expected services over IPv6.

The impairment module of IxLoad could be further expanded. An interesting impairment that can be added is user plane packet corruption, which can be either payload corruption or checksum corruption. This could be used to send erroneous packets in order to test the DUT behavior upon such events. Currently, the impairment module only handles packets as an atom and does not go further inside the packets and alters them. This can be an interesting functionality that can be used to recreate certain user scenarios where a faulty mobile is sending malformed packets.

In this sense, it would be also convenient to have retransmission as an impairment-module, in order to investigate cases where faulty packet flows are central to verify the correct GGSN behavior. This is essential since an inspection engine shall be able to cope with situations where same packets are passing through more than once. Thus, having a test tool with such a capability, will give the verification engineer the power to verify a correct packet inspection engine such as the GGSN.

Another improvement proposal for the impairment module would be to do impairment on the control plane level instead of on the user plane traffic which has already been implemented. This can help, for example to simulate a scenario where one SGSN is not working correctly and it is dropping or duplicating control plane packets. This would result in the ability to create scenarios in IxLoad that can be used to examine and verify the GGSN behavior in the presence of a faulty SGSN.

As stated in section 3.3, the impairment module is altering packet flows with the use of impair profiles. These impair profiles are not flexible enough to filter out a specific packet that is not including a set of TCP flags. Suppose that the Impairment module is expanded to be applicable on an application level instead on a global network level. This would enable the ability to individually select a specific command in an applications' command list that should be impaired. By implementing this, it will change the orientation of IxLoad's impairment into application traffic flows instead of the network flow orientation which it currently has. Also, it would suit the whole nature of IxLoad as an application-based traffic simulator.

For a GGSN, as well as for any kind of packet inspection engine, it would have been interesting to have a test tool that allow the sending of TCP messages separate from the specific application protocol flow. A clear example is for the HTTP application, when a HTTP request is sent, the client will first request to setup a TCP connection, by doing a 3-way handshake with SYN, SYN|ACK and ACK messages. After this, the client can for instance; send an HTTP GET request using negotiated parameters in the handshake. After the transaction is finished, the client or server will send a FIN message which the other part responds back, with a FIN ACK in order to close the TCP Connection. After this, no more TCP messages should come on this connection. However if it does, the GGSN sometimes needs to treat them using special methods. Therefore, it would be good if IxLoad could force the client or server to send TCP packets after a TCP connection has been closed. This could be done by adding a kind of impairment which generates out of sequence TCP messages or resending previously transmitted packets.

The GGSN has support for gathering detailed packet statistics with the use of packet inspection. These statistics can be reported on bearer level (L3-L7) or application level. On the other hand, IxLoad is only displaying its application statistics on L4-L7. However, that is only supported for TCP. For UDP, only L7 statistics is supported. The bytes sent and received for TCP can be seen in the TCP statistics. Also it can display the bearer level statistics but that is only on a per-interface level, meaning that it aggregates all the different applications statistics and doesn't distinguish among them. The problem is when verifying the packet detection mechanism of a GGSN, where it is important to see the exact match between what IxLoad sends and what the GGSN is able to detect. For this reason, it is necessary that IxLoad includes the bearer level statistics per application, meaning that the statistics should include all the L3 packets included in the flow. This would result in an easier verification procedure, which a good test tool should provide.

In the GTP module of IxLoad, it is needed to specify more information elements regarding the Create PDP Context Request and Update PDP Context Request. For instance, it should be possible to specify the charging characteristics value, in order to simulate a proper SGSN which are getting user information from the HLR. Table 3 depicts the Information Elements in a Create PDP Context Request from the 3GPP standard TS 29.060 release 7 which IXIA states that IxLoad supports. [34]

Information element	Presence requirement
IMSI	Conditional
Routeing Area Identity (RAI)	Optional
Recovery	Optional
Selection mode	Conditional
Tunnel Endpoint Identifier Data I	Mandatory
Tunnel Endpoint Identifier Control Plane	Conditional
NSAPI	Mandatory
Linked NSAPI	Conditional
Charging Characteristics	Conditional
Trace Reference	Optional
Trace Type	Optional
End User Address	Conditional
Access Point Name	Conditional
Protocol Configuration Options	Optional
SGSN Address for signalling	Mandatory
SGSN Address for user traffic	Mandatory
MSISDN	Conditional
Quality of Service Profile	Mandatory
TFT	Conditional
Trigger Id	Optional
OMC Identity	Optional
Common Flags	Optional
APN Restriction	Optional
RAT Type	Optional
User Location Information	Optional
MS Time Zone	Optional
IMEI(SV)	Optional
CAMEL Charging Information Container	Optional
Additional Trace Info	Optional
Correlation-ID	Optional
Private Extension	Optional

Table 3: Information Elements in a Create PDP Context Request

Further investigation shows that the content of Table 4 is not present in IxLoad:

Information element	Presence requirement
Charging Characteristics	Conditional
Trace Reference	Optional
Trace Type	Optional
Protocol Configuration Options	Optional
Trigger Id	Optional
OMC Identity	Optional
Common Flags	Optional
Additional Trace Info	Optional
Correlation-ID	Optional
Private Extension	Optional

Table 4: Information Element not present in the GTP module

Out of the 10 elements shown in Table 4, the Charging Characteristics is the one that is most useful to have. It is also specified as conditional while the others in the table are stated to be optional. IxLoad have support for Protocol Configuration Options protocol for CHAP or PAP, but a more versatile PCO field is wanted that is not bound to CHAP or PAP. The field “Private Extension” is also requested to improve the versatility of the GTP module of IxLoad, since this can include proprietary extensions that are specific for an Ericsson packet core network.

Currently IxLoad is not supporting redirect request for HTTP PUT requests. IxLoad states that they are compliant to RFC 2616 concerning HTTP 1.1. [36] The RFC states “...user agent MAY then make its own decision regarding whether or not to redirect the request”. [45] Although this feature is not mandatory, it can be useful to have when evaluating GGSN redirect functionality, because as stated in section 4.1, the GGSN is capable of sending redirect requests to the user with both GET and PUT requests.

The configuration of a GGSN is very adaptive to how it is used and how the operators use it. This will result in many different customer specific configurations and if a tester has to adapt his/her test environment to each customer, it will take a very long time. These configurations might for example, include several inspection rules which classify traffic towards different webpages. It would be advantageous to have a test tool which could create traffic towards these URIs in a clear and convenient way. IxLoad includes a web server wizard, which could be expanded to incorporate such a solution, in a way that different URIs with different frequency weights are requested. The tester should be able to control these frequencies and assign them to the URIs. He/she should also be able to specify which of these addresses should be successful, which addresses that would generate a page not found or which addresses that would trigger a redirect. If this functionality was included, it would prove that IxLoad is not only dynamic and versatile in the protocol stacks but also incorporate a dynamic and versatile solution for specifying an advanced test configuration.

It seems that there is no interconnection in IxLoad, between the different modules, not only between application modules but also between the application module and the underlying

network modules. A clear example is when limiting the outbound rate with the impairment module, which creates a buffer that the application traffic generator stores its output. The impairment module then transmits these packets with the specified rate. A problem arises when the outbound rate is much smaller than the rate produced by the application layer. It appears to be no way for the lower layer module to inform the higher layer modules that it is being flooded by messages. This will result in that the buffer will increase in size until the port will be out of memory and crash. By fixing this issue and making the network modules dependant of the application modules, IxLoad could be used more efficiently during high performance testing, where the application modules could produce relevant traffic instead of producing traffic that is going to be stored in buffers.

Today, the mobile network is facing a new challenge with data-hungry smartphones for example the Apple iPhone. There is no current standard defining that how the mobile phones shall behave when they are trying to attach to the GPRS/UMTS networks. These smartphone have great potential to access the web and bring information to the users by regularly creating PDP contexts towards the packet core to access the PDN. They constantly initiate sessions and even if they receive a response with an error cause, they will still try to initiate sessions. It is essential to test packet core main elements like GGSN with different scenarios and conditions that smartphones can cause. Therefore, to create more realistic scenarios with IxLoad, it should support a feature to give the possibility to simulate subscribers that are acting as smartphones, meaning that even if the simulated subscriber cannot access the PDN for any reason they should retry continuously. With current features, as described in section 4.9 GGSN restart scenario, if created sessions fail for any reason except time out, IxLoad does not retry to create new sessions.

IxLoad is very horizontal in its design, with only incorporating testing of the Gn and the Gi interface of a GGSN. For a more complex test scenario, for example section 4.10 QoS control test scenarios, it is usually needed to include other test tools to simulate surrounding elements, such as PCRF, OCS, RADIUS and CDR handler. A test tool that is able to properly test and verify the GGSN's communication with all its surrounding elements is more usable and beneficial. A test process including different test tools is cumbersome for the tester and gives rise to differences between the involved test tools in layout, design and terminology. Having the possibility to simulate the major environmental elements other than SGSN and PDN gives IxLoad the ability to be independent of other test tools and give the tester a comfortable position where all the statistics are viewable from the same location. It will increase the tester's efficiency and decrease the configuration faults, thus decreasing the verification time and effort.

Currently, IxLoad is a very expensive tool to use for throughput testing. The biggest IxLoad chassis includes 12 slots for 12 load modules. The most powerful module for GTP, Acceleron-NP, consists of twelve 1G-port and one 10G port. To get maximum performance, it is possible to aggregate all twelve ports in order to enable the 10G port. Table 5 demonstrates the Acceleron-NPs' maximum performance numbers for different protocols. Based on the provided information in Table 5, for example to test 100 Gigabit HTTP throughput, at least 14 Acceleron-NP modules is needed since each module can at most provide 7.2 Gbps. Moreover, IXIAs' proposed solution for throughput testing is to have the equal amount of load modules on both sides of the GGSN. Thus, theoretically to test 100 Gigabit at least 28 Acceleron-NP load modules are needed, since 14 load modules are needed to emulate the client side and 14 load modules to emulate the server side. Three chassis are needed to reach this objective, which is expensive in terms of licenses and hardware. Therefore, If IxLoad wants to be ahead of its competitors in the market, IXIA should provide a solution for testing high performance devices in an efficient way.

	Per Port CPU Acceleron-NP	Per Load Module Acceleron-NP
Max GTP-c transaction rate	200 transactions/second	240 transactions/second
Max PDP Contexts - ftp	53.5 K	642 K
Max PDP Contexts – http	100 K	1.2 M
Max PDP Contexts – Video on Demand (VOD)	2 K	24 K
Max PDP Contexts – POP3	67.5 K	810 K
Max PDP Contexts – SMTP	84.4 k	1 M
Max PDP Contexts – IMAP	74 K	888 k
Max throughput – ftp (25K PDP contexts per port)	520 Mbps	6.2Gbps
Max throughput – ftp (50K PDP contexts per port)	270 Mbps	3.2 Mbps
Max throughput – http (25K PDP contexts per port)	600 Mbps	7.2 Gbps
Max throughput – http (50K PDP contexts per port)	520 Mbps	6.2 Gbps
Max throughput – Application Replay (25K PDP contexts per port)	310 Mbps	3.7 Gbps
Max throughput – Application Replay (50K PDP contexts per port)	380 Mbps	4.5 Gbps
Max throughput – P2P (25K PDP contexts per port)	500 Mbps	6 Gbps
Max throughput – P2P (50K PDP contexts per port)	330 Mbps	3.9 Gbps

Table 5: IxLoad performance number for Acceleron-NP

Another suggestion is to improve the test verification of IxLoad. This will be achieved by addressing interoperability issues with a high quality GGSN before delivering new features to its customers. During this thesis, several faults were found and fixed when the IXIA support team was demonstrating added functionality. It was observed that although some features were tested before, they were not working as expected with a proper GGSN. This resulted in unexpected resource and time consumption. However, If IxLoad would have been verified by a proper GGSN in advance, there would be no need for costly traveling expenses and allocating time and resources to address the issues on the customer side.

The Objective type Throughput is measured with bits per seconds, however there are companies that would like to specify the throughput values as packet per second. Sometimes packets per seconds are a more concise type, since some specific functionality is better visualized as packets per seconds than bits per second.

5.2 User interface related concerns

Improvement of the IxLoad user interface is important since it will lead to fewer support cases opened due to shortcomings of IxLoad. As the users understanding of IxLoad capabilities increases, there will be a reduction in the amount of time and resources needed for IXIAs' support.

An issue that has been observed several times is when there is an invalid configuration in IxLoad test scenario or in the device under test. The errors that are shown in IxLoad logs are not clear and they do not explicitly explain the reason that causes the problem. An example is when configuring a GTP test in static mode and the PDP is not initiated correctly, IxLoad shows

an error, stating that the network configuration timed out, although it was due to the GGSN responded with “No resources available”. Clearer errors with better clarification about what caused the problem are needed for better troubleshooting, which will result in less support involvement.

When creating a test using the objective type ‘Simulated User’, the underlying network-level GTP UE basic parameter ‘Count’ is meant to be specified accordingly in order to create a sufficient amount of PDP tunnels. The issue is when specifying a higher value for Simulated Users than what is specified in ‘Count’, the test will not meet the objectives and will not give any error at run-time or warning during configuration that the objective value is set too high.

IxLoad is always maintaining a log-file, where it displays current status and it includes the perceived warnings and errors. Additionally, when IxLoad experiences an error, a pop-up window is shown for the user. However, the displayed error is not usually the first error that IxLoad detects. It shows one error out of the set of errors that occurred. An example is when erroneously configuring the IxLoad SIP client and executing a test. The displayed error is "Failed to configure the Net-Traffic.", but looking in the log-file, it shows that the first error that appears is “Traffic1@Network1 error: Source IPs (10) must equal Destination IPs (1) for Traffic Map mode IP Pairs”. From a troubleshooting and test development perspective, it is important to see the first error that is detected by the test tool.

When the ports are being configured, the GUI is locked and cannot be subjected to any changes until the test is running, which is not satisfactory and pleasant for the tester. An example is when the tester opens a saved scenario and wants to execute it on another set of ports. The user might only change the ports and then executes the test without updating the capture setting, in such case user would not receiving a capture from the test. The capture settings is changed to having capturing enabled on the traffic-label and raises the warning “Capture enabled on Traffic, but no ports were selected”. During testing, it is observed that the capture-enabling on ports couldn’t be set when the ports are in the configure-phase, although it can be enabled when the test is running. This is not sufficient, since the capture will not record the packets that have already been transmitted. One solution would be to enable the tester to specify the port capturing settings and then append it to the configuration that is being sent to the ports. Another way that would solve this issue can be to copy the previously configured capture settings to the new set of port pairs. Improvements in this area would be less time-consuming since the tester doesn’t have to run a test several times, for just small changes. Consequently it would increase the usability of IxLoad.

Another related issue is that some parts of the test configuration are not accessible when a test is running. An example is when a test is being executing, the “save as”-functionality is not available, although the regular save is available. Another example which is shown in Figure 52 is where the scrollbar is locked and some information cannot be viewed. Figure 53 depicts the same window when IxLoad is in unconfigured mode. During a long test such an issue can be problematic, especially if the tester does not know the scenario by heart.

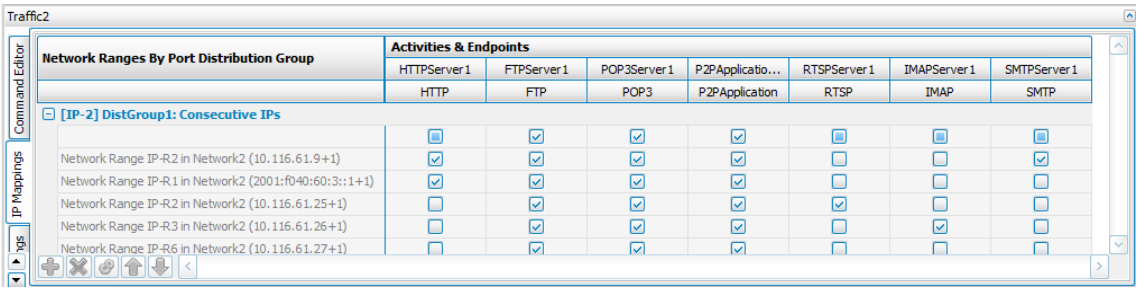

Figure 52: Locked scrollbar during exectution of test

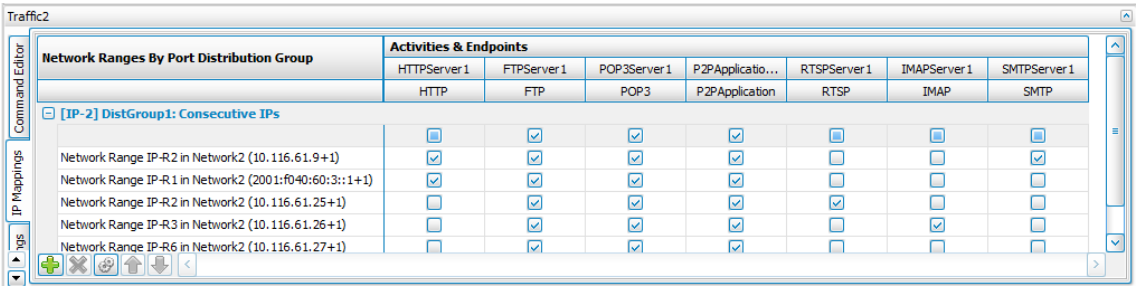

Figure 53: Unlocked scrollbar in unconfigured mode

Backwards compatibility is one of the issues that every company need to be aware of. With the introduction of version 5.0, IxLoad changed the structure of the .RXF-files, which prohibits older versions of IxLoad to use RXF-files created by newer version. IxLoad has sometimes problem with using old RXF-files that was created with an older version. Sometimes the RXF file has to be reconstructed due to changed behavior of IxLoad, such as increased memory constraints or changed predefined value unit for example Kbps to Mbps. With a proper test tool, testers should be able to easily execute already created test scenarios with new software versions, in order to maintain and upgrade old script. This will decrease the time and resource consumption for the responsible tester.

The IxLoad GUI should have the possibility to undo wrong changes in configuration. When creating a scenario it would be very convenient and comfortable for the tester to be able to undo previous changes. For example when mistakenly deleting an application or network traffic, the entire underlying configuration will be lost and the tester has to recreate the deleted material which is time-consuming and troublesome.

The objective type labelled as ‘Throughput’ is not capable to reach the maximum throughput, which was mentioned in Table 2: IxLoad back to back performance for ASM 1000XMV12. The objective type ‘Throughput’ is used to transmit and receive TCP payloads at a certain rate specified in the Objective Value field. Although, it is able to meet the objective for average to small values, it is not capable to reach the objective when specifying higher values. The end user does not care about how IxLoad internally handles the different objective types, the expectation of the test tool is to meet the specified objective by selecting the proper underlying structures to ensure that the desired outcome is achieved.

Configuring an IxLoad test case can be a delicate issue, this is due to IxLoads’ inherent dependencies to global and local settings. There are situations where a global setting overrides a local setting. An example is when selecting to publish statistics on the GTP client, but not having the global setting “Enable Network Dependencies” set, this result in no statistics being

shown and no warning is shown that the configuration is invalid. At this point, the tester needs to be informed about the current issue with a relevant warning for a quicker troubleshooting.

When configuring a L2TP test case without having the GGSN configuration for handling L2TP, IxLoad does not raise an error for it. It goes from configuring mode to clean-up mode without any errors and finishes the test. A good test tool should indicate and make the user aware that something is wrong, especially when a network problem occur that will prevent the test from being executed.

5.3 Information provisioning for the IxLoad user

The information presented on the IXIA homepage is not updated often. As stated on IxLoad webpages, most of the tables and IxLoad data sheets were updated last time November 2008/2009. [36] Although new software version was released, the information and performance numbers on the website was not updated and the new functionality was not presented.

Another information provisioning issue with IxLoad is that the user guide is not sufficiently updated with each released version. An example is the “Network Stack / Application Traffic Compatibility Matrix” where it for example states that IxLoad is supporting WAP over GTP, which is not correct. Also, more concise and more thorough information about the released software versions should be published to clearly state the differences between the versions, especially in terms of increased CPU and memory consumption.

IxLoad is providing a set of pre-defined templates, which are meant to make the learning process easier for the user. Investigating these templates shows that there are no GTP-templates provided. These predefined small test scenarios are important, as examples, for new beginners to grasp how to use IxLoad in GGSN environment. It would be convenient if IXIA could include such templates in future releases.

Another set of templates are available for the HTTP response codes. These are HTTP 200_OK and HTTP 404 PAGE_NOT_FOUND. There are possibilities to configure other response codes, but since this can be tedious and require some job, it would be convenient if IXIA could provide templates for at least the redirect responses 301 Moved permanently and 302 Found. [45]

After an IxLoad test is finished there is a possibility to generate a report, with all the statistics details, in a pdf format. However, the GTP statistics are not included in the generated report, which are of utmost importance when verifying GGSN test scenarios to visualize various GGSN functionalities.

6 Future work

In this chapter, the future work of IxLoad together with some suggestions about further assessment of IxLoad will be discussed.

LTE is the overall target for every company involved in the world of telecommunication. IXIA are planning to include support for LTE in IxLoad. Their aim is to support MME, eNode B, PDN, PCRF and Diameter servers to provide a full testing of S-GW and PDN-GW, as can be seen in Figure 54.

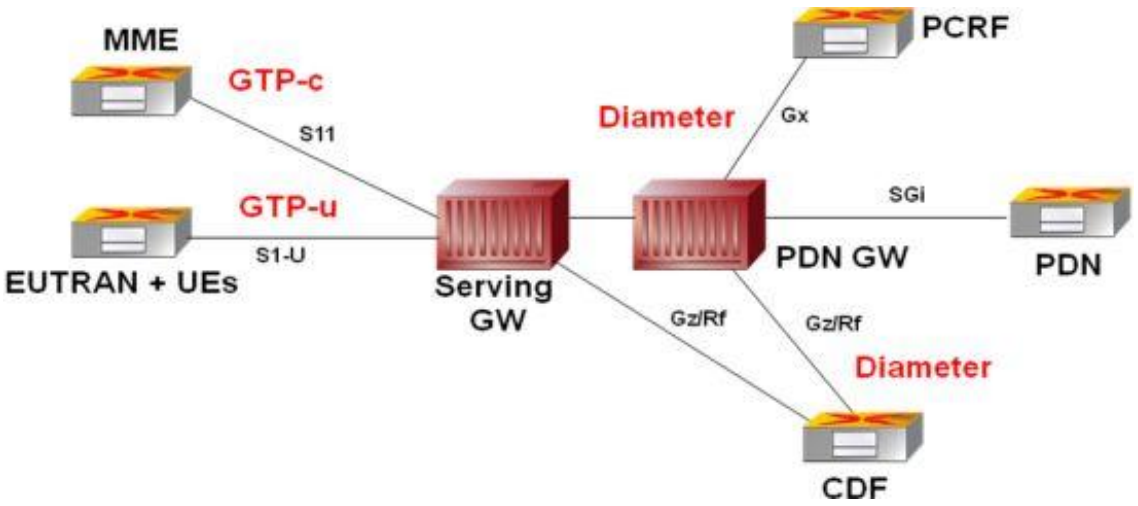

Figure 54: Enhanced Packet Core Network Testing in IxLoad

With the introduction of LTE, the packet core network will become transformed to the Evolved Packet Core (EPC), an all IP network without any special network for handling voice and SMS, where voice calls should be SIP-based calls ran over IMS. The packet core are standing in front of a huge change where the end-to-end QoS is crucial, especially if voice calls are meant to be transported concurrently with heavy data services such as streaming and P2P. LTE can offer a downlink data rate of 300 Mbit/s per sector. With several of these involved, an eNode B will have data rates from 1 to 3 Gbit/s. The EPC will be responsible over a set of eNode B with the duty to respect the QoS requirements and provide a low latency to each user together with having an immense amount of throughput. IXIA have stated in a white paper, that 99% of all network failures only occur under peak usage. With this in mind, in order to fully verify the throughput capacity of the EPC, it will require a high performance test tool. As discussed in section 5.1, IxLoad is currently an expensive test tool for throughput testing. IXIA has estimated to maintain their performance numbers for LTE as they had for UMTS using their most powerful load module, as can be seen in Table 5. Considering these numbers, testing S-GW and PDN GW throughput with IxLoad becomes a very costly situation and almost implausible task. IxLoad need to increase its throughput capacity to decrease the amount of load modules needed, consequently decreasing the costs. [46]

IxLoad is a test tool that has several different functionalities, where audio and video are two large cornerstones. An extensive evaluation of these modules was not part of the assessment, due to the reason that not all the audio and video protocols are supported over GTP and this evaluation is based on what IxLoad is capable of doing over GTP. Furthermore, due to the lack of Acceleron-NP load modules, it was not possible to perform a high throughput test scenario, in order to stress IxLoad and verify the performance that IXIA claims,

The suggested future work, in the context of this thesis, is to perform a further assessment when the proposed functionalities are implemented. As discussed above, another important future task is to assess IxLoad and its performance in an EPC network. This should consist of a functional evaluation and performing several interoperability test scenarios, where the quality of IxLoad is validated.

7 Conclusion

This thesis was aimed to assess the test tool, IxLoad, in a GGSN environment. For this purpose, GGSN is introduced as a main element in the UMTS core network, followed by a description of IxLoad and its capabilities. Additionally, several test cases are proposed based on the functionality of a GGSN, for which an advanced test tool is needed. This thesis points out several areas in which IxLoad is needed to be expanded and improved to become an ideal test tool for a GGSN and in the future for S-GW and PDN GW. By improving the tool in the suggested areas, it can fill out some of the gaps between different GGSN verification phases, which will result in an increase of the quality of the GGSN. Furthermore, increasing the usability of IxLoad has several business benefits for its company. It will result in a higher customer satisfaction, reduce the development time and maintenance cost in terms of lower support expenses and decreased amount of time spent on education. Currently, IxLoad is not capable enough to validate all the functionalities of a GGSN, but with the suggested improvements, it would have great potential to be unique test tool in a GGSN environment.

References

1. IXIA. 2010. IxLoad: Overview, IXIA company, viewed March 2010< http://www.ixiacom.com/products/ixload/index.php>

2. European Telecommunications Standards Institute, 2006, *Digital cellular telecommunications system (Phase 2+) Universal Mobile Telecommunications System (UMTS) General Packet Radio Service (GPRS) Service description Stage 2 ,* 3GPP TS 23.060 version 6.15.0 Release 6 , European Telecommunications Standards Institute.

3. Richardson K. W, 2000, UMTS Overview, *ELECTRONICS & COMMUNICATION ENGINEERING JOURNAL,* JUNE 2000, viewed 28 March 2010 < http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.114.1393&rep=rep1&type=pdf>.

4. Dahlman, E, Parkvall, S, Skold, J, Beming, P, October 3, 2008, *3G Evolution, Second Edition: HSPA and LTE for Mobile Broadband,* Academic Press.

5. Rahnema, M, 2008, *UMTS network planning, optimization, and inter-operation with GSM,* John Wiley and sons, Singapore.

6. 3rd Generation Partnership Project, 2009, *3rd Generation Partnership Project Technical Specification Group Core Network and Terminals General Packet Radio Service (GPRS) GPRS Tunnelling Protocol (GTP) across the Gn and Gp interface (Release 6),* 3GPP TS 29.060 version 6.21.0, 3GPP.

7. 3rd Generation Partnership Project, 2008, *3rd Generation Partnership Project Technical Specification Group Core Network and Terminals Mobile radio interface Layer 3 specification Core network protocols; Stage 3 (Release 6),* 3GPP TS 24.008 V6.19.0, 3GPP.

8. 3rd Generation Partnership Project, 2009, *3rd Generation Partnership Project Technical Specification Group Services and System Aspects Telecommunication management Charging management Charging architecture and principles (Release 9),* 3GPP TS 32.240 V9.0.0, 3GPP.

9. 3rd Generation Partnership Project, 2009, 3rd Generation Partnership Project Technical Specification Group Service and System Aspects Telecommunication management Charging management Diameter charging application (Release 9), 3GPP TS 32.299 V9.2.0, 3GPP.

10. Oodan A, Ward K, Savolaine C, Daneshmand M, Hoath, 2003, *Telecommunication: Quality of Service Management: from legacy to emerging services.* The institution of Electrical Engineers, London.

11. Nichols, et. Al, Differentiated Services Field, RFC 2474, December 1998 <http://www.ietf.org/rfc/rfc2474.txt>

12. Blake, et. Al, Architecture for Differentiated Services, RFC 2475, December 1998 <http://www.ietf.org/rfc/rfc2475.txt>

13. Alfonso, A, Vasilis, F, Aghvami, H, 10 December 2002, 'Differentiated Services versus Over-provisioned Best-effort for pure-IP Mobile Networks', Mobile and Wireless Communications Network, 2002. 4th International Workshop, Page(s): 450 – 457, viewed March 2010, < http://ieeexplore.ieee.org/Xplore/login.jsp?reload=true&url=http://ieeexplore.ieee.org/stamp/stamp.jsp%3Ftp%3D%26arnumber%3D1045806&authDecision=-201 >.

14. 3rd Generation Partnership Project, 2009*, 3rd Generation Partnership Project Technical Specification Group Services and System Aspects Quality of Service (QoS) concept and architecture (Release 9)*, 3GPP TS 23.107 V9.0.0, 3GPP.

15. 3rd Generation Partnership Project, 2009, *3rd Generation Partnership Project Technical Specification Group Services and System Aspects End-to-end Quality of Service (QoS) concept and architecture (Release 9)*, 3GPP TS 23.207 V9.0.0, 3GPP.

16. 3rd Generation Partnership Project, 2010, *3rd Generation Partnership Project Technical Specification Group Services and System Aspects Policy and charging control architecture (Release 9)*, 3GPP TS 23.203 V9.4.0, 3GPP.

17. Soldani, D, Li, M, Cuny, R, 2006, *QoS and QoE Management in UMTS Cellular Systems,* John Wiley and sons ltd, England.

18. Copeland, R, 21 December 2009, 'Network Intelligence – Facilitate Operators Win in Mobile Broadband Era'*, Intelligence in Next Generation Networks, 2009. ICIN 2009. 13th International Conference,* Page(s): 1 – 6, viewed March 2010 < http://ieeexplore.ieee.org/Xplore/login.jsp?url=http://ieeexplore.ieee.org/stamp/stamp.jsp%3Ftp%3D%26arnumber%3D5357062&authDecision=-201>.

19. Allot Communications, 20th December 2007, *Digging Deeper Into Deep Packet Inspection (DPI),* Allot Communications, viewed March 2010, < https://www.dpacket.org/articles/digging-deeper-deep-packet-inspection-dpi>.

20. Ericsson, 2010, *Core Networks /Packet Core /GGSN-MPG, Ericsson* Company, viewed March 2010, < http://www.ericsson.com/ourportfolio/products/ggsn-mpg?nav=fgb_101_214|fgb_101_256>.

21. 3rd Generation Partnership Project, 2009-12, *3rd Generation Partnership Project Technical Specification Group Services and System Aspects Direct tunnel deployment guideline (*Release 9*),* 3GPP TR 23.911 V9.0.0, 3GPP.

22. Ericsson, 2010, Ericsson world-first in delivering innovative 3G technology to Telstra, Ericsson Company, viewed March 2010 ,<http://www.ericsson.com/au/ericsson/press/2008/20080610_ericsson_worldfirst.shtml >.

23. 3rd Generation Partnership Project*,* 2007-03*, 3rd Generation Partnership Project Technical Specification Group Services and System Aspects General Packet Radio Service (GPRS) Service description Stage 2* (Release 7), 3GPP TS 23.060 version 7.4.0.

24. Shneyderman, A, Bagasrawala, A, Casati, A, 2001, Lucent Technologies Inc., '*Mobile VPNs for Next Generation GPRS and UMTS Networks'*, Lucent Technologies Inc., viewed March 2010<http://esoumoy.free.fr/telecom/tutorial/3G-VPN.pdf>.

25. Hoffman, J, 2002, *GPRS demystified*, First edition, McGraw-Hill Professional, United States of America.

26. Yuan-Kai, C, Yi-bing, L, 14 mars 2005, '*IP connectivity for gateway GPRS support node*', IEEE Wireless Communications, Volume: 12, Issue: 1 on page(s): 37 – 46, viewed March 2010, < http://ieeexplore.ieee.org/Xplore/login.jsp?url=http://ieeexplore.ieee.org/stamp/stamp.jsp%3Ftp%3D%26arnumber%3D1404571&authDecision=-201>.

27. IXIA, December 2008, '*IxLoad Testing the Application-Aware Delivery Network'*, IXIA Company, viewed March 2010, <http://www.ixiacom.com/pdfs/library/brochures/ixload_brochure.pdf>.

28. IXIA, April 2008, '*Testing Multiplay Networks',* IXIA Company, viewed March 2010, < http://www.ixiacom.com/solutions/testing_multiplay/ >.

29. IXIA, December 2008, '*Gigabit and 10 Gigabit Ethernet XMV12X Application & Streams Module',* IXIA Company, viewed March 2010, < http://www.ixiacom.com/products/interfaces/display%20?skey=in_xmv12x>.

30. IXIA, November 2009, '*Dynamic Interface Behavior (DIB)'*, IXIA Company, viewed March 2010, < http://www.ixiacom.com/products/display?skey=aptixia_ixload_dynamic_interface_behavior>.

31. IXIA, 2009, '*IxLoad User guide'*, IXIA Company, viewed May 2010.

32. IXIA, November 2009*, 'IxLoad: QoE Detective',* IXIA Company, viewed March 2010*,* < *http://www.ixiacom.com/products/display?skey=aptixia_ixload_qoe_detective*>.

33. IXIA*,* August 2007*, 'Analyzer user guide',* Release 1.10*, IXIA Company.*

34. IXIA, November 2009*, 'IxLoad: GGSN Testing, IXIA Company,* viewed March 2010*,* < *http://www.ixiacom.com/products/display?skey=aptixia_ixload_ggsn_testing*>.

35. IXIA*,* May 2009*, 'Aptixia IxLoad GTP Plug-in User Guide'* Version 4.20*,* IXIA Company.

36. IXIA*,* November 2009, *'IxLoad: IxLoad Data Sheet', IXIA Company*, viewed March 2010, < http://www.ixiacom.com/products/ixload/ixload_datasheets/index.php>.

37. IXIA, November 2009*, 'IxLoad: Data - HTTP, SSL and FTP',* IXIA Company, viewed March *2010,* < *http://www.ixiacom.com/products/display?skey=aptixia_ixload_http_ssl_ftp*>.

38. IXIA, November 2009*, 'IxLoad: Data - Mail (SMTP, POP3, IMAP)',* IXIA Company, viewed March 2010, <

http://www.ixiacom.com/products/display?skey=aptixia_ixload_smtp_pop3_imap>.

39. IXIA, November 2009*, 'IxLoad: Data - Streaming (RTSP, RTP)',* IXIA Company, viewed March 2010*,* < http://www.ixiacom.com/products/display?skey=aptixia_ixload_rtsp_rtp>*.*

40. IXIA, November 2009*, 'IxLoad: Peer to peer',* IXIA Company, viewed March 2010*,* < http://www.ixiacom.com/products/display?skey=aptixia_ixload_peer_to_peer>*.*

41. IXIA, November 2009, *'IxLoad: Network Protocol - DHCP, PPPoX'*, IXIA Company, viewed March 2010*,* <http://www.ixiacom.com/products/display?skey=aptixia_ixload_network_protocols >.

42. China Web Radar, 2007, *Stats: 39 Million WAP Users in China*, Dreamhost, viewed May 2010, < http://www.chinawebradar.com/608/stats-39-million-wap-users-in-china.html>.

43. Nigeria Mobile Marketing, 2008, *Opportunities and Challenges of WAP media in Nigeria,* CBS Interactive, viewed May 2010, <http://www.zdnet.co.uk/blogs/nigeria-mobile-marketing-10008775/opportunities-and-challenges-of-wap-media-in-nigeria-10008787/>.

44. Marion Walton, 2009, *How many mobile Internet users in SA?,* Wordpress.com, viewed May 2010, < http://marionwalton.wordpress.com/2009/08/29/how-many-mobile-internet-users/>.

45. Fielding et al, Hypertext Transfer Protocol -- HTTP/1.1, RFC 2616, June 1999 < http://www.ietf.org/rfc/rfc2616.txt>

46. IXIA, 2009, *IxLoad: Enhanced Packet Core (EPC) Network Testing,* IXIA, viewed May, 2010, < http://www.ixiacom.com/products/display?skey=aptixia_ixload_epc_network_testing >.