Implementation and testing of the One-Way Active Measurement Protocol (OWAMP)

Student:	**Kenan Cenanovic**
Advisors at UPC:	**René Serral Gracià**
	Jordi Domingo Pascual
Advisors at KTH:	**Thomas Lindh**
Date:	**2/2/2004**

Abstract

The IETF IP Performance Metrics (IPPM) working group has proposed The One-Way Active Measurement Protocol (OWAMP) draft standard for measuring metrics like one-way packet delay and loss across Internet paths. The OWAMP protocol is used for active measurement of network performance by adding additional traffic to an IP-network. The task was to implement all parts of the protocol and verify that is works appropriately by performing tests on existing experimental network. The programming language that was used for implementing the OWAMP was C because it is fast and commonly used for developing of real time applications software.

Preface

This report is a part of the master thesis which is the concluding stage of 4,5 year long engineering program called Master of Computer Networks at the Royal Institute of Technology (KTH) in Stockholm. Thanks to the cooperation between KTH and Universitat Politècnica de Catalunya (UPC) in Barcelona, it was possible to perform the master thesis at UPC´s research center called Advanced Broadband Communications Center (CCABA). I would like to thank Josep Solé Pareta for giving me opportunity to do my master thesis at Universitat Politècnica de Catalunya (UPC) through Erasmus student exchange, and I would also like to thank my supervisors Jordi Domingo Pascual and René Serral Gracià for truly helping and supporting me during this whole project. Grateful thanks to Karin Linde at international KTH office for her engagement in making contacts with UPC.

Table of Contents

Abstract **5**
Preface **6**
Table of Contents **7**
1. Introduction **9**
1.1 Internet Protocol (IP) 9
1.2 TCP vs. UDP 10
1.2.1 Transmission Control Protocol (TCP) *10*
1.2.2 User Datagram Protocol (UDP) *12*
1.3 Passive Measurement 13
1.4 Active Measurement 14
1.5 The Network Time Protocol (NTP) 14
1.6 Linux vs. Windows 15
1.6.1 Linux in Prime Time *15*
1.6.2 Negative Things about Linux *16*
1.6.3 Positive Things about Linux *17*
1.7 OWAMP Overview 18
2 Background **20**
3 Method **21**
3.1 Goal Description 21
3.2 Implementation of OWAMP 22
3.3 OWAMP Control 24
3.3.1 Connection Setup *24*
3.3.2 OWAMP-Control Commands *24*
3.4 Unspecified Links 27
3.4.1 Server to Session-Receiver Link *28*
3.4.2 Client to Session-Sender Link *29*
3.5 Fetching the Results 31
3.6 OWAMP-Test 31
3.6.1 Test Sender Behaviour *31*
3.6.2 Test Receiver Behaviour *33*
4 Analysis **36**
4.1 Test scenario 37
4.1.1 Test with Big Packet Size *37*
4.1.2 Test with Small Packet Size *40*
4.2 Test Analysis 43
5 Economical Analyse **44**
5.1 Implementation cost 44
5.2 Cost for program testing 45
5.3 Cost for the equipment 45
5.4 Total cost 46
6 Conclusions **47**
6.1 Future Work 47
7 References **48**
8 Appendix **49**
8.1 Time Plan 49
8.2 Control-Client Configuration File 49
8.3 Fetch-Client Configuration File 51

8.4 The format of Request-Session message 52
8.5 The format of Result File 53

1. Introduction

Usually IP networks support only a best effort service, that limitation has not been a problem for traditional Internet applications like web and E-mail, but it does not satisfy the needs of many new applications like audio, video streaming, IP-Telephony or video-conference, which demand high data throughput capacity (bandwidth) and have low-latency requirements.

The performance of an IP network is of vital importance to both the service providers and the customers. Parameters such as bandwidth, delay, jitter and loss can be obtained when measuring the performance with two basic methods, actively by the addition of test traffic or passively by observing user generated traffic.

The introduction of this document explains general concept of IP networks including basic protocols like TCP and UDP that it uses, then two basic methods for measuring of IP performance are introduced and difference between two mostly used OS (Operative System) Windows and Linux. After that it explains basic concept and gives an overview of the OWAMP (One-Way Active Measurement Protocol).
More details about OWAMP are given in chapter Method which explains actual implementation of OWAMP. After implementation the program was tested on experimental IP network and results are shown in chapter Analyses.

1.1 Internet Protocol (IP)

The Internet Protocol (IP) is the method or protocol by which data is sent from one computer to another on the internet. Each computer (known as a host) on the Internet has at least one IP address that uniquely identifies it from all other computers on the Internet. When sending or receiving data (for example, an e-mail note or a Web page), the message gets divided into little chunks called packets. [2]

IP is a connectionless protocol, which means that there is no continuing connection between the end points that are communicating. Each packet that travels through the

Internet is treated as an independent unit of data without any relation to any other unit of data. (The reason the packets do get put in the right order is because of TCP, the connection-oriented protocol that keeps track of the packet sequence in a message. The most widely used version of IP today is Internet Protocol Version 4 (IPv4). However, IP Version 6 (IPv6) is also beginning to be supported. IPv6 provides for much longer addresses and therefore for the possibility of many more Internet users. IPv6 includes the capabilities of IPv4 and any server that can support IPv6 packets can also support IPv4 packets.

1.2 TCP vs. UDP

The Internet is a worldwide system of computer networks – a network of networks in which a user at one computer can get information from any other computer. Technically, what distinguishes the Internet is its use of a set of protocols called the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). TCP and the UDP are the basic communication languages for many applications and the Internet. There are a number of differences between the TCP and UDP, which are summarized below.

1.2.1 Transmission Control Protocol (TCP)

The following fields are defined for the TCP header (see figure nr.1), as described in [RFC793]:

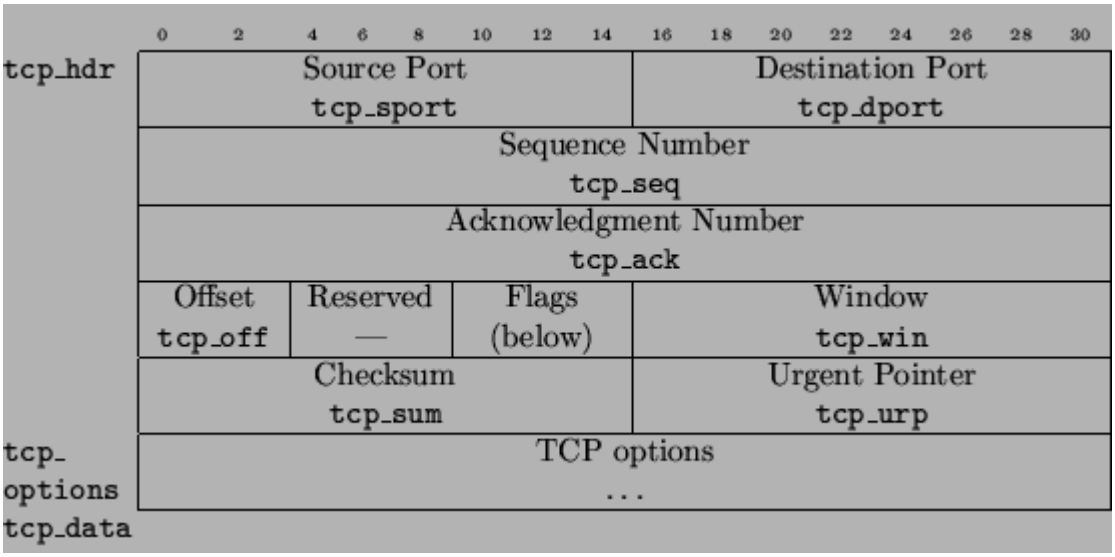


Figure nr. 1 showing TCP header

TCP Flags:

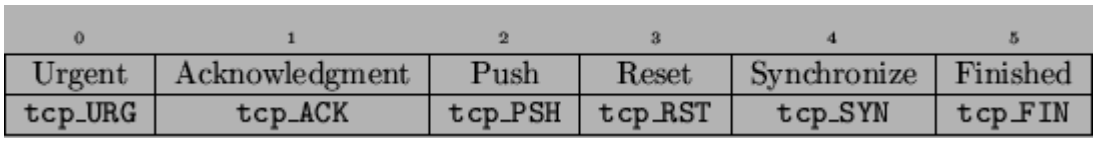


Figure nr. 2 showing TCP Flags

The TCP is the basic communication protocol for many applications and the Internet. The TCP relies upon the IP layer to send information from one place to another. The TCP layer manages the assembling of messages into smaller packets that are transmitted over the Internet and received by another TCP layer at the destination, which reassembles the packets into the original message. The IP layer handles the address part of each packet so that it gets to the right destination.

Although TCP and UDP use the same IP layer, TCP provides totally different service to the application layer. TCP provides a connection oriented reliable byte stream service. The term connection oriented means the two applications using TCP must establish a connection before they can exchange data with each other. This kind of handshake makes sure that both the client and server are prepared to accept the incoming data. The connection oriented TCP service provides the capability for reliable data transfer. TCP in addition provides flow control and congestion control.

In the event of a packet/acknowledgement, loss, TCP retransmits the packet to the receiver, adding to TCP's reliability. [3]

TCP protocol is used for OWAMP-Control setup but not during actual OWAMP-Test where UDP-Protocol is used for sending of Test-Packets in order to obtain proper test results, one example of that are possible packet losses that can't be detected if using TCP because of retransmission.

1.2.2 User Datagram Protocol (UDP)

The following fields are defined for the UDP header (see figure nr.3), as described in [RFC768]:

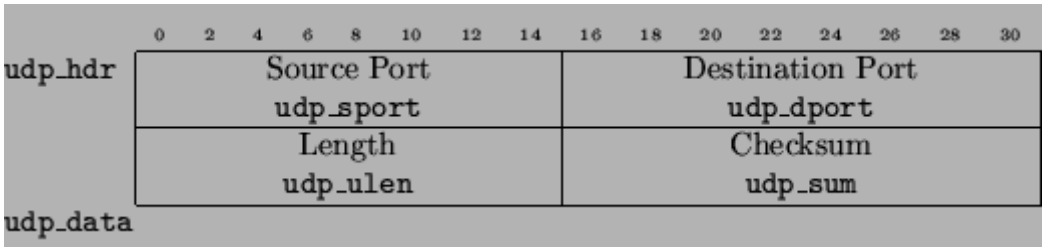


Figure nr. 3 showing UDP header

The UDP is a simple, datagram oriented transport layer protocol. Each output (write) operation by a process, causes exactly one UDP datagram to be sent. This is different from a stream-oriented protocol such as the TCP, where the amount of data sent has little relationship to what actually gets sent in a packet. Unlike TCP, UDP does not provide the service of dividing a message into packets and reassembling at the other end. UDP does not provide any reliability, UDP is not concerned with the guaranteed delivery of the packets to the receiver and any such reliability should be added on by the concerned application. UDP provides two services not provided by the IP layer. It provides various services through different port numbers and optionally a checksum capability to verify whether the data arrived intact. UDP does not provide flow control and congestion control.UDP (User Datagram Protocol) is anther commonly used protocol on the internet. UDP is never used to send data such as: webpages, database information, etc; UDP is used for audio and video. Streaming media such as Windows

Media Audio files, Real Player, and others use UDP because it offers speed! The reason UDP is faster than TCP is because there is no error correction. The data sent over the internet is affected by collisions, and errors will be present. This is one of the reasons why streaming media is not high quality, other reasons are delay and packet losses. It is also concerned with speed and use of bandwidth. [3]

1.3 Passive Measurement

Passive measurement is commonly used for monitoring the traffic without creating or modifying it in order to collect data about performance and behavior of packet streams. Passive measurement can be implemented by incorporating some additional intelligence into network devices to permit them to identify and record the characteristics and quantity of the packets that flow through them. Packet statistics can then be collected without the addition of any new traffic. The level of detail of the information collected, how the metrics are being processed and the volume of traffic passing through the monitor device is dependent on the network metrics of interest.

Examples of the types of information that can be obtained using passive monitoring are:

- Bit or packet rates
- Packet timing / inter-arrival timing
- Queue levels in buffers (which can be used as indicators of packet loss and delay)
- Traffic / protocol mixes
- Delay

Passive measurements are carried out by observing normal network traffic and as such do not disturb it.

1.4 Active Measurement

Active measurement is another way of measuring network performance and it involves the injection of some probe packets into the network from which the relevant metrics of that traffic can be measured. The sole purpose of probe packets is to provide some insight into the way real network traffic can be treated within the network.

The type of network metrics derived using active measuring, are:

- Delay (Time needed for a IP packet to travel from source to destination)
- Loss (Packets that are send but not received are considered to be lost)
- Delay variations (Even called Jitter is variation of packet delay times)

One of the advantages of active measurement is that it does not require full access to network resources. (e.g. routers), disadvantages of active measurement is that it may disturb the network by injecting artificial probe traffic into the network.

1.5 The Network Time Protocol (NTP)

The Network Time Protocol (NTP) is used to synchronize the time of a computer client or server to another server or reference time source, such as a radio or satellite receiver or modem. It provides accuracies typically within a millisecond on LANs and up to a few tens of milliseconds on WANs relative to Coordinated Universal Time (UTC) via a Global Positioning Service (GPS) receiver, for example. Typical NTP configurations utilize multiple redundant servers and diverse network paths in order to achieve high accuracy and reliability. [6]

With the increasingly wide availability of affordable global positioning system (GPS) and CDMA based time sources, hosts increasingly have available to them very accurate time sources either directly or through their proximity to NTP primary (stratum 1) time servers. By standardizing a technique for collecting IPPM one-way active measurements, it could be possible to create an environment where IPPM metrics may be collected across a far broader mesh of Internet paths than is currently

possible. One particularly compelling vision is of widespread deployment of open OWAMP servers that would make measurement of one-way delay as commonplace as measurement of round-trip time using an ICMP-based tool like ping.

The OWAMP uses Network Time Protocol (NTP) to synchronise system clocks for Test-Packet sender and receiver in order to obtain higher accuracy of Test-Packet delay times

1.6 Linux vs. Windows

The OS (Operative System) used when programming was Linux because of it's flexibility and stability when programming and makes it more suitable than Windows for software development .Main differences between those two main OS are explained below.

1.6.1 Linux in Prime Time

As Linux stands today, a majority of Windows users don't have a good reason to make a switch. For folks who aren't all that technically adept, or who have hardware that isn't supported in Linux, sticking with Windows and the applications they already have just makes more sense.

As far as Linux has come, it still has far to go to achieve universal appeal. Over the past two years, the companies that sell Linux distributions have improved setup, the user interface, and technical support. But many rough edges remain. Without more fixes to smooth the operation of apps and more support from the major hardware makers (many aren't writing Linux drivers for their products), Linux's future continues to be uncertain.

But for savvy users who want to try out its myriad options, Linux can provide a stable, secure, and inexpensive computing experience. With the parts it already has and more spit-and-polish, Linux could become a top-tier operating system.

1.6.2 Negative Things about Linux

- You must be an expert: Commercial distributors and GUI makers have simplified some tasks, but many procedures still require dropping to the command line, decoding cryptic system messages, or hand-editing what can be complex configuration files. If you fail to learn at least a smattering of Linux's intricacies, chances are good that you won't get much done.

- Lagging hardware support: Linux's altruistic band of designers does an admirable job of building in support for new types of hardware. Without a major push for Linux drivers from hardware manufacturers, however, your Linux distribution may never support some peripherals.

- Second-tier software: You can get every conceivable utility for Linux, most at no cost, but many can't match the best Windows or Mac apps. Premier Linux apps like StarOffice, Evolution, and The GIMP still provide only a subset of the features found in Microsoft Office, Microsoft Outlook, and Adobe Photoshop. What they do offer is more than adequate--unless you need one of the missing features.

- A confusion of distros: Since Linux is free, anybody can package the operating system and sell their own distribution. Once you decide to give Linux a shot, you still need to determine which distribution is a good fit: Mandrake, Red Hat, SuSE, or one of dozens of others. Hardware support can vary, and user friendliness can be nonexistent in some versions.

- Support at a price: People complain about the cost of Microsoft's $35-per-incident tech support. With few exceptions, however, Linux distributors aren't any cheaper; with SuSE, the free installation support is severely limited. If you're a Linux beginner planning on calling for help, consider installing Mandrake Linux, with its $15-per-call (or less, in quantity) support policy.

1.6.3 Positive Things about Linux

- It's free: Even though distributors can add value to Linux (by adding installers, providing tech support, and publishing multi-CD or multi-DVD installation sets), is also free to download it. From the Web site, you can download at no charge all of the software to install certain version of software. Download and give away as many copies as you want. Install it on as many PCs as you want. The only things that will cost you money are printed manuals, technical support over the phone, and a nice package with discs.

- Highly adaptable: Linux distributors can customize their version of Linux to target a specific type of user. Programmers can modify the source code to suit their needs, and redistribute the software for free. The result is a completely customizable OS, free from the constraints of having to do things in a particular way.

- Strong on security: Linux doesn't often fall victim to network security vulnerabilities. When it does, legions of Linux coders generally release patches that fix the problem within 24 hours--though users still need to download and apply these patches. Virus writers haven't made Linux a major target yet.

- Plentiful on line help: Can't find an answer to a Linux question in the included documentation? You'll find hundreds of FAQs, how-tos, and message boards on the Web. Start your search at The Linux Documentation Project.

- One OS fits all: Linux, in one form or another, will run on everything from a 486 doorstop with 8MB of RAM (try that with Windows XP) to clusters of high-speed servers. It won't be the same version of Linux running the same applications, but Linux is good at fitting in where Microsoft leaves machines behind with Windows' ever-increasing minimum system requirements. [4]

1.7 OWAMP Overview

OWAMP-Protocol is used for active performance measuring of IP-networks by inserting UDP-streams of test-packets in to IP-network in order to obtain the travel time from source to destination for each test packet (one way delay).

Growing availability of good time sources to network nodes, makes it increasingly possible to measure one-way IP performance metrics with high precision.
To do so in an interoperable manner, a standard protocol for such measurements is required. The One-Way Active Measurement Protocol (OWAMP) can measure one-way delay, as well as other unidirectional characteristics, such as one-way loss. This protocol is at the time of this writing is in draft status. Current draft is 7, is expected to become an RFC in few months.

The One-Way Active Measurement Protocol (OWAMP) consists of two inter-related protocols: OWAMP-Control and OWAMP-test. Where OWAMP-Control is used to initiate, start and stop test sessions and fetch their results, while OWAMP-Test is used to exchange test packets between two measurement nodes.[1]

OWAMP-Control is designed to support the negotiation of:

- Sender and receiver addresses
- Port numbers
- Session start time
- Session length
- Test packet size
- The mean Poisson sampling interval for the test stream
- Per-hop behaviour (PHB)

- Encryption

- Authentication for both test and control traffic

Defining roles of:

- Session-Sender: the sending endpoint of an OWAMP-Test session

- Session-Receive: the receiving endpoint of an OWAMP-Test session

- Server: an end system that manages OWAMP-Test sessions, is capable of configuring per-session state in session endpoints, and is capable of returning the results of a test session

- Control-Client: an end system that initiates requests for OWAMP-Test sessions, triggers the start of a set of sessions, and may trigger their termination.

- Fetch-Client: an end system that initiates requests to fetch the results of completed OWAMP-Test sessions

2 Background

The IETF IP Performance Metrics (IPPM) working group has proposed draft standard metrics for one-way packet delay [RFC2679] and loss [RFC 2680] across Internet paths. Although there are now several measurement platforms that implement collection of these metrics[SURVEYOR], [RIPE], there is not currently a standard that would permit initiation of test streams or exchange of packets in an interoperable manner. To do so a standard protocol for such measurements is required. The One-Way Active Measurement Protocol (OWAMP) can measure one-way delay, as well as other unidirectional characteristics, such as one-way loss.

In order to understand better the features of this report the reader should have basic knowledge about how IP-networks functions, about different QOS-parameters and tools that are used for measuring of network performance.

3 Method

3.1 Goal Description

The primary goal of this project was Implementing of the One-Way Active Measurement Protocol's (OWAMP) functions necessary for basic use of the protocol and test-performing, features like encryption, authentication and Poisson sampling were left as optional.

Planned goals (features):

- Implementing of OWAMP-Control
- Implementing of OWAMP-Test Unauthenticated Mode
- Variable packet length
- Test scheduler with pattern test support
- IPv4/IPv6 support
- Configuration file support

The protocol is implemented by using programming language C because it is fast and commonly used for developing of real time applications software. Because of several logically separated roles, the C-program was made of 5 different program-code files which follow the IPPM recommendation, allowing broad flexibility in use.

Given that this thesis is framed on a bigger research project, once implemented, the software will be used for analysing an existing network. This way is possible to verify

the good behaviour of the implemented protocol. Lately this program will be integrated in the existing CCABA software-platform for further tests.

3.2 Implementation of OWAMP

As shown in figures 4 and 5 different logical roles can be played by the same host allowing broad flexibility in use.

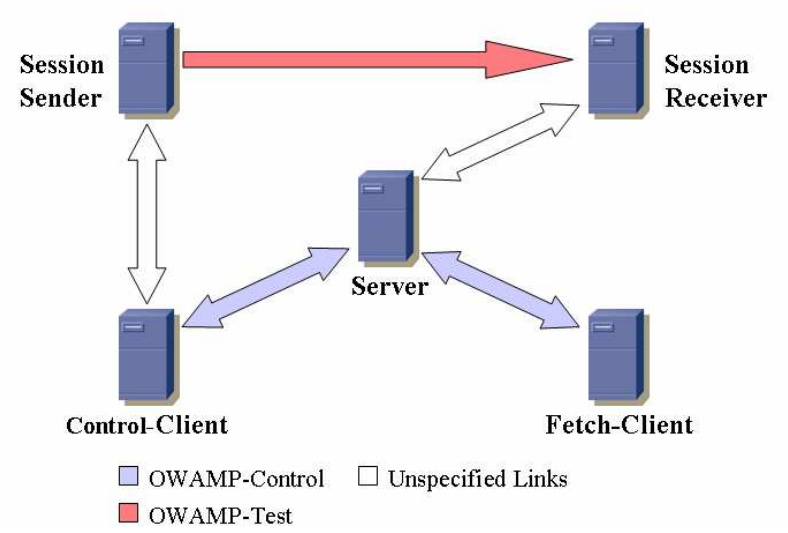


Figure nr. 4 showing relationship scenario between different logical roles

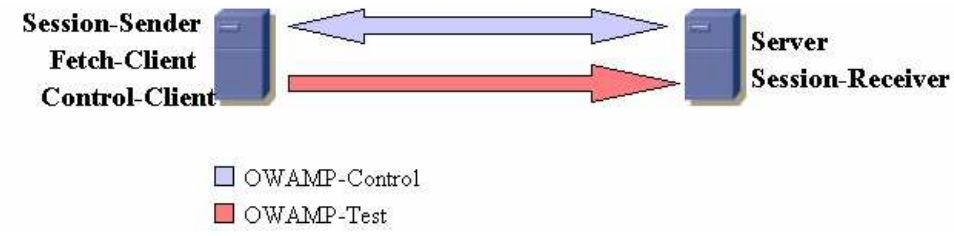


Figure nr. 5: Different logical roles can be played by the same machine

OWAMP consists of two inter-related protocols: OWAMP-Control and OWAMP-Test.

OWAMP-Control is layered over TCP and is used to initiate and control measurement sessions and to fetch their results

OWAMP-Test is layered over UDP and is used to exchange test packets between two measurement nodes.

The initiator(client) of the measurement session establishes a TCP connection to a well-known port on the target point(server). An OWAMP server should listen to this well-known port.
OWAMP-Control messages are transmitted only before OWAMP-Test Sessions are actually started and after they complete (with the possible exception of an early Stop-Session message).

The OWAMP-Control and OWAMP-Test protocols support three modes of operation: unauthenticated, authenticated, and encrypted. The authenticated and encrypted modes are not implemented.

Performing of OWAMP-Test and Result-Fetching is made in five different steps as shown below in figure nr. 6.

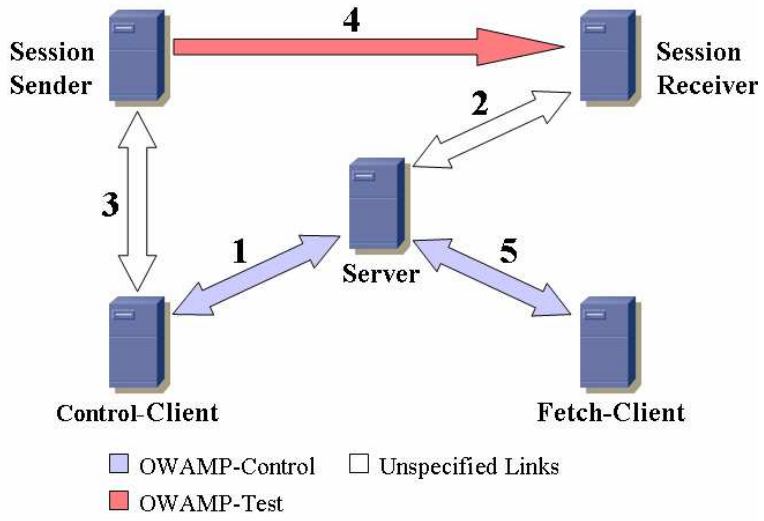


Figure nr. 6 showing order in which different computers connect each other during the OWAMP Test

3.3 OWAMP Control

3.3.1 Connection Setup

Before either a Control-Client or a Fetch-Client can issue commands of a Server, it must establish a connection to the server. Connection setup is a initiating stage for the client - server connection (step nr. 1, see figure 6), it is preformed directly after client opening a TCP connection to well-known port which server listens to and receiving new client connections at.

During this stage client and server agrees on which mode to use when exchanging different control messages between each other, the chosen mode is used during the whole connection time. There are three different modes: unauthenticated, authenticated, and encrypted. The authenticated or encrypted modes require endpoints to possess a shared secret.
Only unauthenticated mode is implemented.

3.3.2 OWAMP-Control Commands

The following commands are available for the client: Request-Session, Start-Sessions, Stop-Session, Fetch-Session. The command Stop-Session is available to both the client and the server.
While conducting active measurements, the only command available is Stop-Session.

The Request-Session command (see appendix 7.4) is issued by an OWAMP client to an OWAMP server (step nr. 1, see figure nr. 6), it contains all important test data needed for performing OWAMP test, that makes it the most important control command during Control setup. The OWAMP server must respond with an Accept-Session message, an Accept-Session message may refuse a request. Different fields of Request-Session Message are explained below.

- IPVN is the IP version numbers for Sender and Receiver. In the case of IP version number being 4, twelve octets follow the four-octet IPv4 address stored in Sender Address and Receiver address. These octets MUST be set to zero by the client and MUST be ignored by the server. Currently meaningful IPVN values are 4 and 6. OWAMP Control Protocol is implemented in the way that It can be used on either IPv4 or IPv6 networks, address resolution between hostname and host dot formed address is flexible making it unnecessary for user to have knowledge about what type of network involved hosts are located on

- Conf-Sender and Conf-Receiver is set to 0 or 1 by the client. The server must interpret any non-zero value as 1. If the value is 1, the server is being asked to configure the corresponding agent (sender or receiver). In this case, the server should disregard the corresponding Port value. At least one of Conf-Sender and Conf-Receiver must be 1. The whole program is based on scenario showed in figure nr. 4, that's why the server is only able to configure receiver, in this case configuration of sender is performed by the client.

- Number of Schedule Slots, as mentioned before, specifies the number of slot records that go between the two blocks of Integrity Zero. They are used for making the time schedule for sending of test packets, sender uses it to determine when to send test packets, and receiver for calculating send times for all test packets.

- Number of Packets is the number of active measurement packets to be sent during this OWAMP-Test session .

- If Conf-Sender is not set, Sender Port is the UDP port OWAMP-Test packets will be sent from. If Conf-Receiver is not set, Receiver Port is the UDP port OWAMP-Test packets are requested to be sent to.

- The Sender Address and Receiver Address fields contain respectively the sender and receiver addresses of the end points of the Internet path over which an OWAMP test session is requested.

- SID is the session identifier. It can be used in later sessions as an argument for Fetch-Session command. It is meaningful only if Conf-Receiver is 0. This way, the SID is always generated by the receiving side

- Padding length is the number of octets to be appended to normal OWAMP-Test packet.

- Start Time is the time when the session is to be started. This timestamp is in the same format as OWAMP-Test timestamps.

- Timeout (or a loss threshold) is an interval of time (expressed as a timestamp). A packet belonging to the test session that is being set up by the current Request-Session command will be considered lost if it is not received during Timeout seconds after it is sent.

- Type-P Descriptor covers specify the requested Per Hop Behaviour Identification Code (PHB ID)as defined in RFC 2836.If Conf-Sender is set, Type-P Descriptor is to be used to configure the sender to send packets according to its value. If Conf-Sender is not set, Type-P Descriptor is a declaration of how the sender will be configured. The value of all zeros specifies the default best-effort service.

- Integrity Zero Padding must be all zeros in this and all subsequent messages that use zero padding. The recipient of a message where zero padding is not zero must reject the message as it is an indication of tampering with the content of the message by an intermediary (or brokenness). This will ensure data integrity. In unauthenticated mode, Integrity Zero Padding is nothing more than a simple check.

One or more schedule slots immediately follow request-Session command:

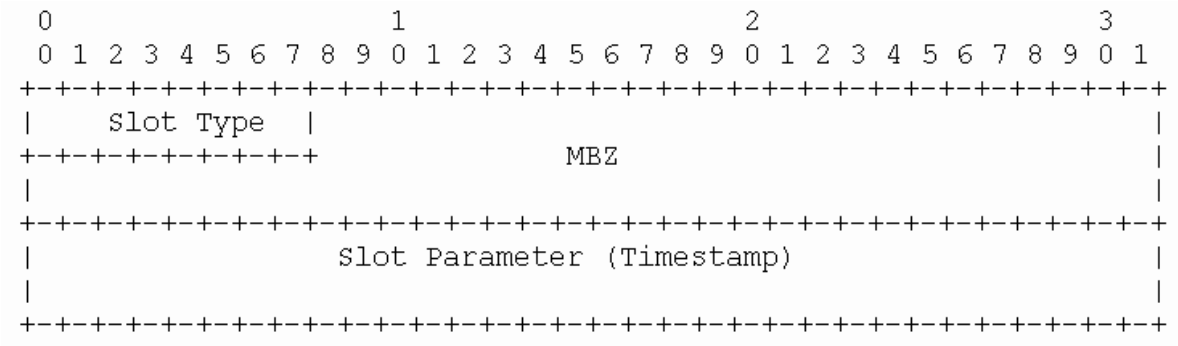


Figure nr. 7 showing the Format of Schedule Slot

The sender and the receiver need to both know the same send schedule. This way, when packets are lost, the receiver knows when they were sent.

To implement this, we have a schedule with a given number of `slots'. Each slot has a type and a parameter. Two types are supported: exponentially distributed pseudo-random quantity (denoted by a code of 0) and a fixed quantity (denoted by a code of 1). The parameter is expressed as a timestamp and specifies a time interval. For a type 1 slot, the parameter is the delay itself. The sender starts with the beginning of the schedule, and `executes' the instructions in the slots; for a slot of type 1, wait the specified time and send a test packet (and proceed to the next slot). The schedule is circular: when there are no more slots the sender returns to the first slot.[1]

The sender and the receiver must be able to reproducibly execute the entire schedule so if a packet is lost, the receiver can still attach a send timestamp to it.

3.4 Unspecified Links

As shown in figure nr. 4 unlabeled links that connects Control-Client to Session-Sender and Server to Session-Receiver are unspecified by OWAMP draft and are proprietary made protocols. In order to keep things simple and making it easier to understand the Control-messages that are used have the same format as specified by OWAMP draft, but only some of them is used like: Request-Session, Accept-Session, Start-Session and Stop-Session

3.4.1 Server to Session-Receiver Link

Server to Session-Receiver link (step nr.2 in figure nr. 6) is proprietary made protocol used for setting up an OWAMP-Test. After receiving Request-Session message (see appendix 7.4) from the client, server either accepts or rejects Request-Session, if rejected Accept-Session message (see figure nr. 9) is send back to client with value of 1 (reject) in accept field, otherwise if accepted the server connects Session-Receiver on a well known port, after establishing connection the same Request-Session message it is send to Session-Receiver. Then the message is examined by Session-Receiver and Control-Ack message (see figure nr.8) is send back to server, 0 in accept field means Request-Session accepted, 1 means that it is rejected. After receiving Control-Ack message the server examines value in accept field, if accepted the SID (Session Identity) number is made which could be later used for fetching results, SID value is copied in to SID field in Accept-Session message(see figure nr. 9) and accept field is set to 0 (accept) . Then Accept-Session message is send to the client.

The format of Control-Ack message:

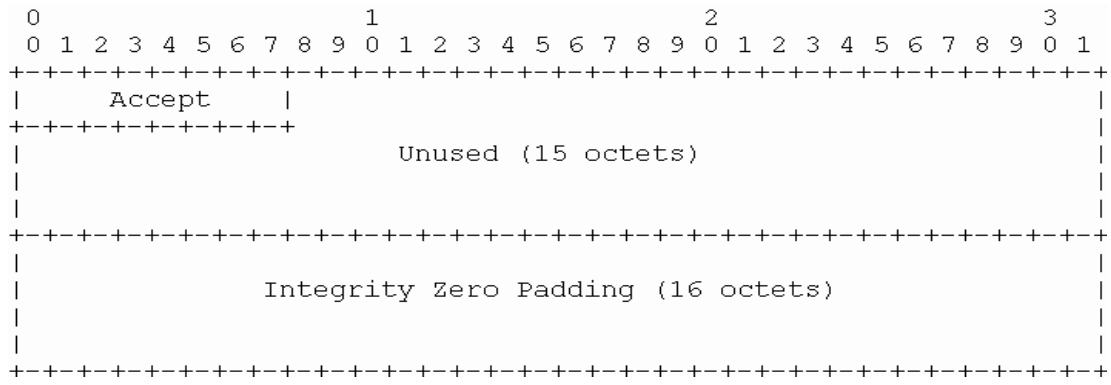


Figure nr. 8 showing the Format of Control-Ack Message

To each Request-Session message, an OWAMP server must respond with an Accept-Session message:

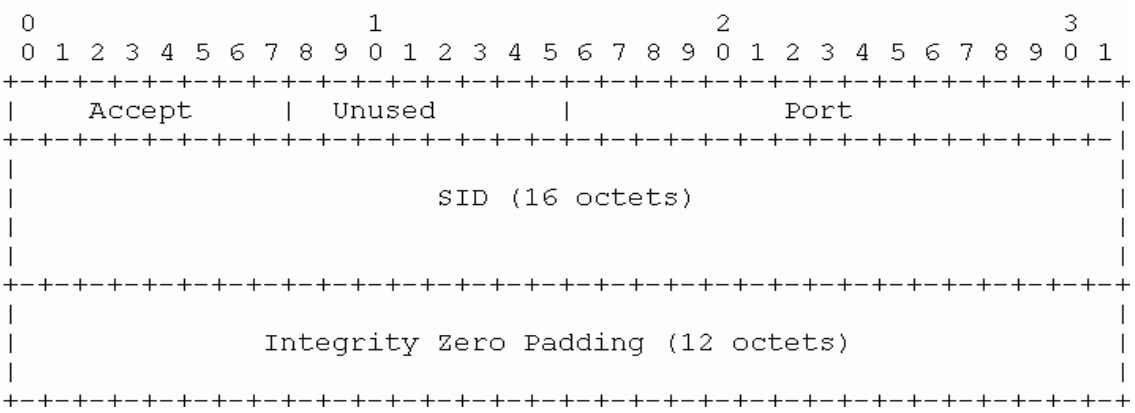


Figure nr. 9 showing the Format of Accept-Session Message

3.4.2 Client to Session-Sender Link

When Accept-Session is received by the client accept field is examined, if Request-Session is accepted, the client connects Session-Sender (step nr. 3 in figure nr. 6) on a well known port then client sends the same Request-Session message to the Session-Sender, which responds by sending Control-Ack message, if Request-Session is accepted by the Session-Sender then it is ready for performing an OWAMP-Test (step nr. 4 in figure nr. 6) and waits for the Start-Session message which is issued by the client.

The format of Start-Session message:

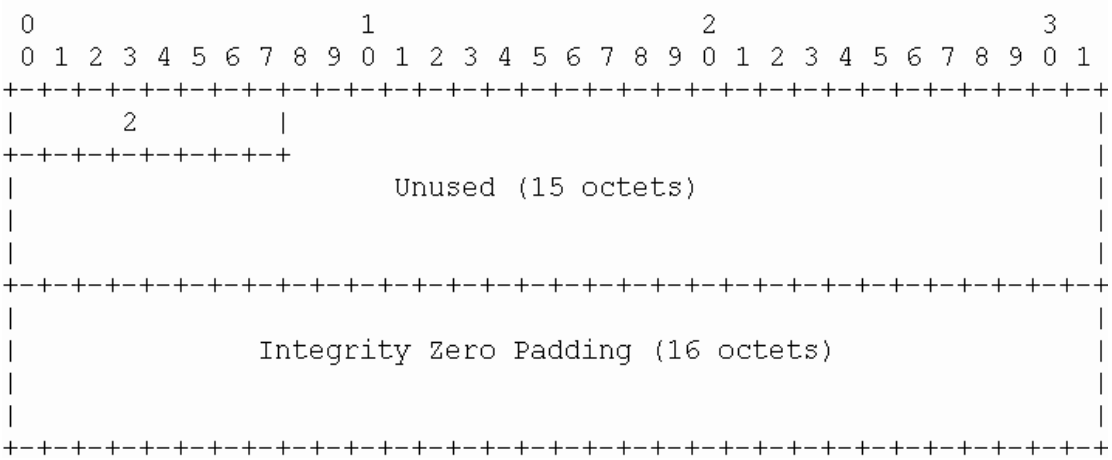


Figure nr. 10 showing the Format of Start-Session Message

Above, the first octet (2) indicates that this is the Start-Session command.

After receiving the Start-Session message, Session-Sender starts performing OWAMP-Test by sending UDP Test-Packets to Session-Receiver. It keeps sending until all Test-Packets are sent or until Stop-Session message is issued by the client.

The format of Stop-Session message:

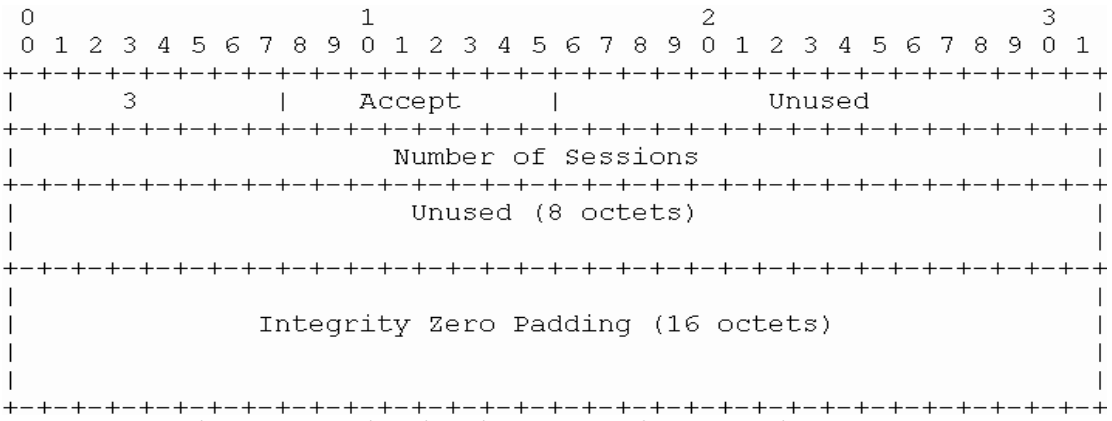


Figure nr. 11 showing the Format of Stop-Session message

This is immediately followed by 0 or more Session Packets sent descriptions (the number of session packets sent records is specified in the 'Number of Sessions' field above):

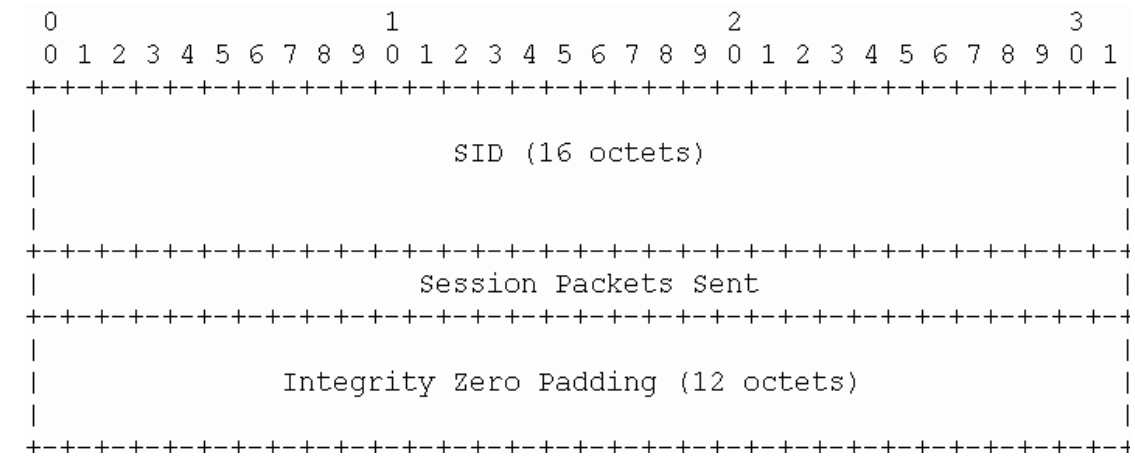


Figure nr. 12 showing the Format of Session Packets

All these messages comprise one logical message: the Stop-Session command indicated by first octet (3)

When Session-Sender receives Stop-Session command, it ends OWAMP- test by cancelling further sending of Test-Packets. After not receiving any of Test-Packets during the receive time-out period, Session-Receiver considers OWAMP- test being finished.

3.5 Fetching the Results

After performing of an OWAMP- Test and working on result data in order to detect and record possible packet losses, result data is sent to the Server.
Result data is implemented as structure called Packet-Records, each structure contains test data in form of: Send-time stamp, Receive-time stamp and sequence number for one Test-Packet.
Fetch-Client's task is to fetch result data from the Server (Step nr.5 in figure nr. 6). In order to perform fetching it must be configured. For that purpose a configuration file called fetchconfig.cfg is made (see appendix 7.3).
The configuration data that user must enter before fetching starts is: Server-name, Session Identification number (SID) of OWAMP-Test to fetch results from, the number of Packet-Records by specifying start-sequence and end-sequence number to fetch Packet-Records between and file name to so save Test-results to.

3.6 OWAMP-Test

OWAMP-Test protocol. It runs over UDP using sender and receiver IP and port numbers negotiated during Request-Session exchange.
As OWAMP-Control, OWAMP-Test has three modes: unauthenticated, authenticated, and encrypted. Like in OWAMP-Control the only implemented mode unauthenticated because all OWAMP-Test sessions spawned by an OWAMP-Control session inherit its mode.

3.6.1 Test Sender Behaviour

The sender sends the receiver a stream of packets with schedule as specified in the Request-Session command (Step nr. 4 in figure nr. 6). The format of the body of a

UDP test packet in the stream depends on the mode being used, because of implementing unauthenticated mode, only Test Packet used in that mode is shown below in figure nr. 13.

Test packet in unauthenticated mode:

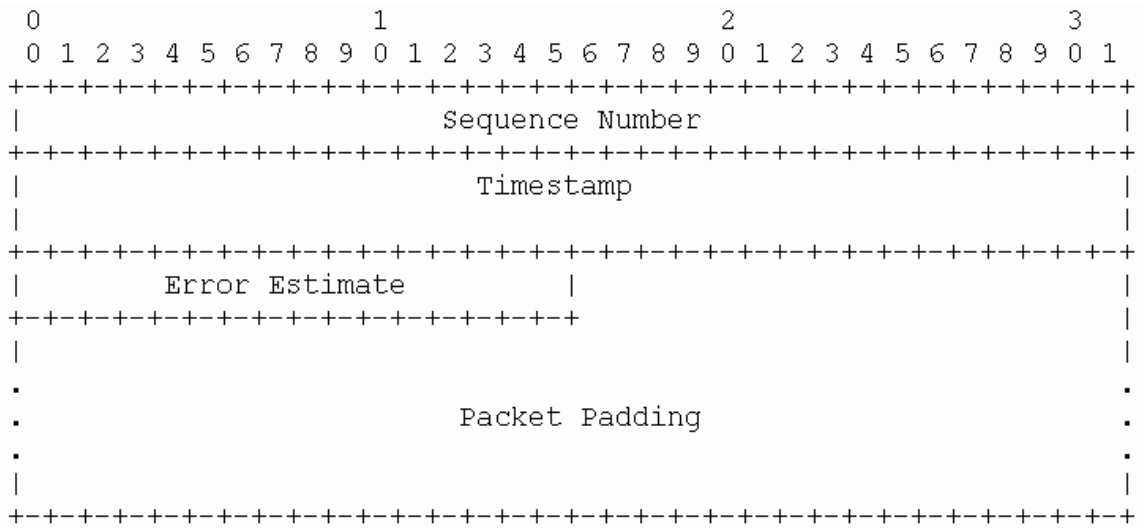


Figure nr. 13 showing the Format of Test Packets

- Sequence numbers starts with 0 and are incremented by for each subsequence test packet.

- Timestamp is same as in RFC 1305 with first 32 bits representing unsigned integer for number of seconds, and second 32 bits also unsigned integer for number of microseconds.

- The Error Estimate specifies the estimate of the error and synchronization.

- Packet Padding consists of all zeros, the size of it depends of value in Padding Length field in Request-Session command. Padding Length is same for all test packets and all of them have same packet size during the entire test . With Padding Length(packet size in configuration file) set to zero the packets are send without any padding which gives the minimum payload size of 40 Bytes for a Test Packet, the maximum payload size is chosen by the user depending on network type for test to be performed on.

During the implementation of Test Sender among the other built in C programming functions two of them were mostly used and are most important for sender's behaviour. The first one is signal(SIGALRM, send_packet_signal) function for handling of alarm signals received from the other function called setitimer(REAL, &new, NULL), parameter new contains struct timeval variable time expressed in seconds and microseconds to wait for before the next signal is send, it follows send times specified in Time Schedule in Request Session message , when signal is received by the signal(SIGALRM, send_packet_signal) function it calls for send_packet_signal function which is used for actual sending of Test Packet.
The sending process is repeated until sending the number of Test-Packets specified in Request-Session.

3.6.2 Test Receiver Behaviour

IPv4 specification makes no claims about the time it takes the packet to traverse the last link of the path. Receiver knows when the sender will send packets with the Timeout parameter defined in Request-Session. Packets that are delayed by more that Timeout are considered lost
The choice of a reasonable value of Timeout is a problem faced by a user of OWAMP protocol, not by an implementer

As packets are received:

- Timestamp the received packet.

- In authenticated or encrypted mode (not implemented) decrypt first block (16 octets) of packet body.

- Store the packet sequence number, send times, and receive times for the results to be transferred.

- Packets not received within the Timeout are considered lost. They are recorded with their sequence nr, presumed send time, and receive time consisting of a string of zero bits.

Packets that are actually received are recorded in the order of arrival. Lost packet records serve as indications of the send times of lost packets. They are placed at the very end of Packet-Record list.

Packets that have send time in the future are recorded normally, without changing their send timestamp, unless they have to be discarded.

Packets with a sequence number that was already observed (duplicate packets) are recorded normally, because IP networks sometimes introduce duplicate packets. The protocol is able to measure duplication.

If any of the following is true, packet must be discarded:

- Send timestamp is more than Timeout in the past or in the future.

- Send timestamp differs by more than Timeout from the time when the packet should have been sent according to its sequence nr.

- In authenticated or encrypted mode (not implemented), any of the bits of zero padding inside the first 16 octets of packet body is non-zero.

After receiving Request-Session which contain all data necessary for setting up and performing an OWAMP Test, receiver starts listening on UDP-port (Step nr. 4 in figure nr. 6), when UDP Test-Packets are received, receiver takes a receive-time stamp and stores that time value together with send-time stamp in the structure called Packet-Record. After that Packet-Record structure is stored in to Packet-Record linked-list, which is of dynamic size because of possible packet losses and duplicate packets.

Every time a packet is received receiver starts waiting (listening) for a next one to arrive, if not received during the time period of time-out seconds (see appendix 7.4) receiver stops receiving Test-Packets. For setting up time-out three functions are used signal (SIGALRM, stop_receive) which calls for stop_receive function when alarm signal is received, alarm signal is send by alarm function alarm(), it takes receive-time out value as argument.

Then receiver makes the other list containing calculated send-time stamps and sequence numbers(Test-Packet numbers) for all Test-Packets. Then that list is compared with Packet-Record linked-list in order to detect packet losses and duplicate packets. Lost and duplicate Test-Packets are recorded at end of Packet-Record linked-list. At the end Packet-Record linked-list is send to the server.

4 Analysis

After concluding implementing OWAMP draft, the program was tested in two different scenarios based on different packet size to insure that it functions properly and to analyse collected result data. Before running an OWAMP-Test a configuration file [config.txt] (see appendix 7.2) must be made for a Test Client, Values in rows that begin with a sign * are not to be changed, they are made for further implementing.
Result files that were fetched after four different OWAMP- Tests were analysed with an existing program called NetMeter [5]. In order to fetch results a configuration file must be made as showed in appendix 7.3.
Result file (see appendix 7.5) obtained after fetching contains send and receive times for every Test-Packet and should be processed by some program like NetMeter in order to obtain an OWAMP-Test result overview.

The test was performed using three different computers, where two of them were acting respectively as a sender and receiver and the third one was configured as a router between them (see figure nr. 14).

During the OWAMP test sender and receiver have also been generating background traffic using NetMeter.

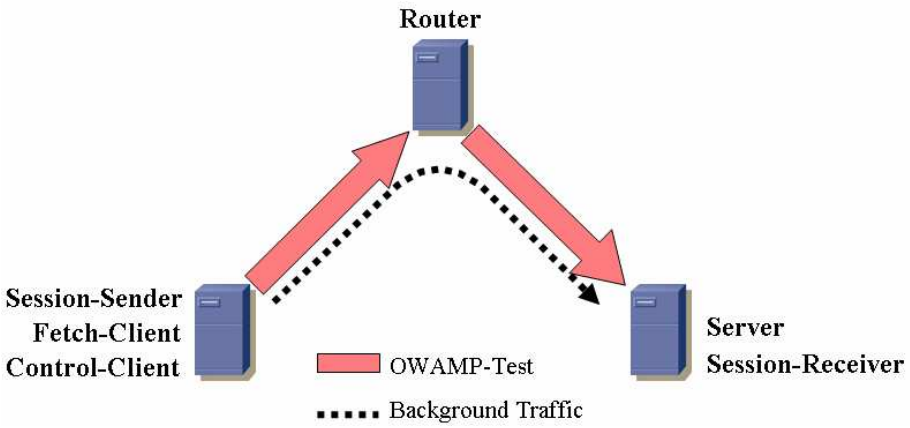


Figure nr. 14 showing Test Scenario

4.1 Test scenario

4.1.1 Test with Big Packet Size

The first test was performed using big packet size which is a maximum total IP packet size of 1500 Bytes that can be used in an Ethernet network, with a maximum payload of 1454 Bytes [1500 – 20 (IPv4 header) – 8 (UDP header) -18 (Ethernet header)].
Test is performed two times, first time with a background traffic utilized by NetMeter program along test path, and second time without any background traffic.
Result files are analysed with NetMeter program, results are shown below.

Input data in config.txt:

SERVER_ADDRESS = xarello.ccaba.upc.es #Server_Address
SENDER_ADDRESS = xarello.ccaba.upc.es #Test_Sender_Address
RECEIVER_ADDRESS = dell.ccaba.upc.es #Test_Receiver_Address
TEST_START_TIME = 10 #Nr_of_Sec_Before_Test_starts
SHEDULE_SLOTS_TIMES = 1 1 1 1 1 1 1 1 1 1#Send_schedule_times_in_microsec
Nr_OF_TEST_PACKETS = 30000 #Nr_of_Test[UDP]_Packets
IP_VERSION = 4 #Ip_Version
TEST_SENDER_PORT = 3594 #Sender_UDP_port
TEST_RECEIVER_PORT = 3595 #Receiver_UDP_port
TEST_PACKET_SIZE = 1454 #Test_Packet_Payload_Size[Bytes]
TEST_RECEIVE_PACKET_TIMEOUT =15 #WaitTimeout_for_UDP_Packets[Sec].
Background traffic generated by NetMeter was 4000 packets/sec with a packet size of 1400 Byte, makes it 4000 * 1400 = 5 600 000 Byte/sec which is around 44,8 Mbit/sec

Output results obtained after running the result file in NetMeter:

FLOW: 0001 (BACKGROUND TRAFFIC ≈ 45 Mbit/sec)

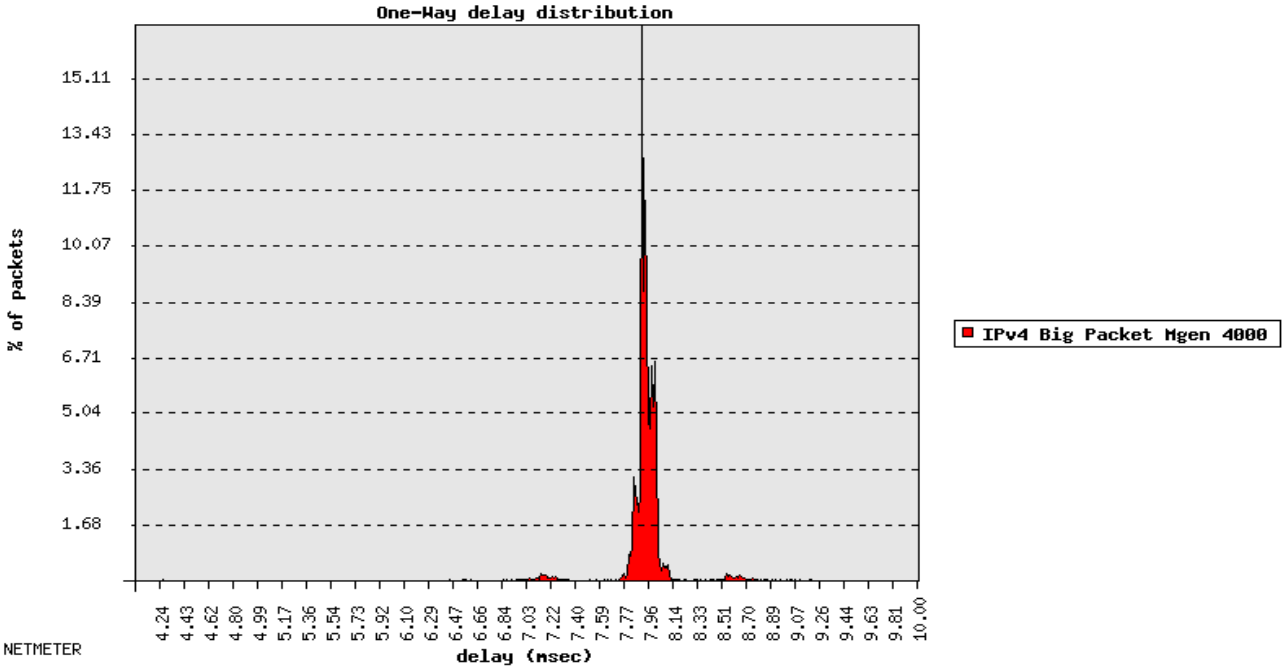


Figure nr. 15 showing Big packet Test result with Background traffic

SOURCE : 147.83.130.169:3594

DESTINATION : 192.168.11.4:3595

Num pkts recvd : 30000

Join delay : 115507.007812 sec

Recv pkt rate : 99.989 pkt/sec

Recv data rate : 1180.708 kbps

Pkts dropped : 0

Ave. Pkt Delay : 0.007954 sec

Max. Pkt Delay : 0.060314 sec

Min. Pkt Delay : 0.004259 sec

Delay variation : 0.056055 sec

FLOW: 0002 (WITHOUT BACKGROUND TRAFFIC)

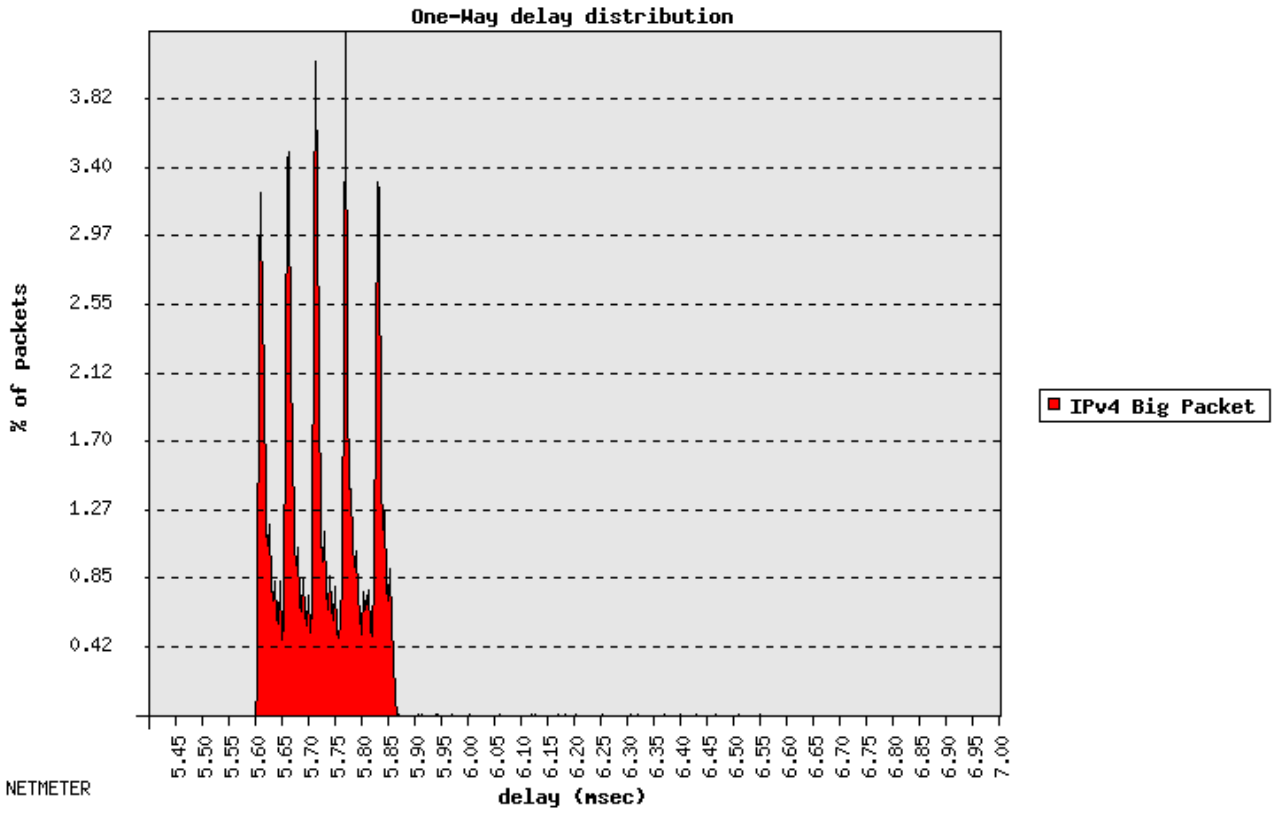


Figure nr. 16 showing Big packet Test result without Background traffic

SOURCE : 147.83.130.169:3594
DESTINATION : 192.168.11.4:3595

Num pkts recvd : 30000
Join delay : 113484.007812 sec
Recv pkt rate : 99.989 pkt/sec
Recv data rate : 1180.708 kbps
Pkts dropped : 0
Ave. Pkt Delay : 0.005727 sec
Max. Pkt Delay : 0.006549 sec
Min. Pkt Delay : 0.005604 sec
Delay variation : 0.000945 sec

4.1.2 Test with Small Packet Size

The second test is almost the same as the first one, only difference is test packet size, this time the minimum packet size is used, where the minimum total size is 86 Byte [40 (payload) + 20 (IPv4 header) + 8 (UDP header) + 18 (Ethernet header)]. Test results are shown below.

Changed values in configuration file (config.txt):

TEST_PACKET_SIZE = 40 #Test_Packet_Payload_Size[Bytes]

Background traffic ≈ 45 Mbit/sec

Output results obtained after running the result file in Netmeter:

FLOW: 0003 (BACKGROUND TRAFFIC ≈ 45 Mbit/sec)

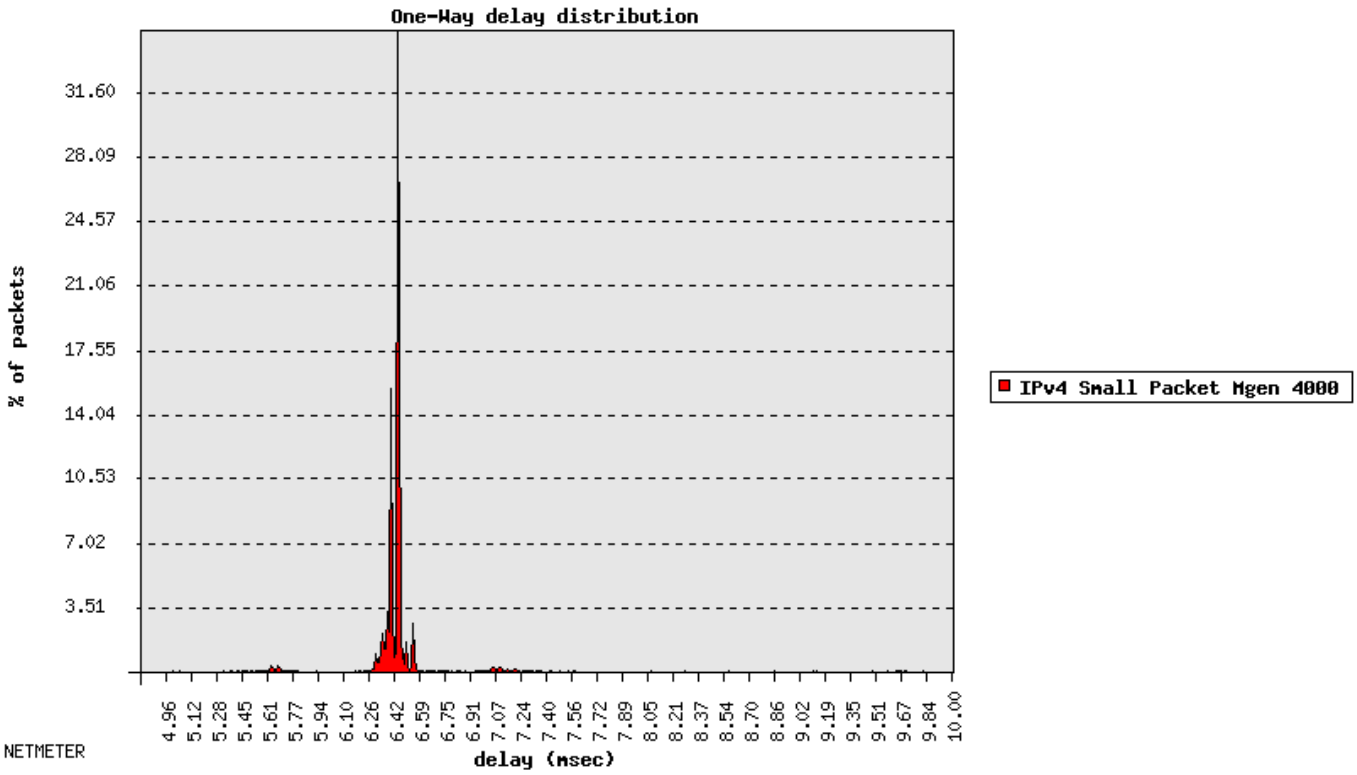


Figure nr. 17 showing Small packet Test result with Background traffic

SOURCE : 147.83.130.169:3594

DESTINATION : 192.168.11.4:3595

Num pkts recvd : 30000

Join delay : 120563.007812 sec

Recv pkt rate : 99.991 pkt/sec

Recv data rate : 31.998 kbps

Pkts dropped : 0

Ave. Pkt Delay : 0.006498 sec

Max. Pkt Delay : 0.058723 sec

Min. Pkt Delay : 0.004997 sec

Delay variation : 0.053726 sec

FLOW: 0004 (WITHOUT BACKGROUND TRAFFIC)

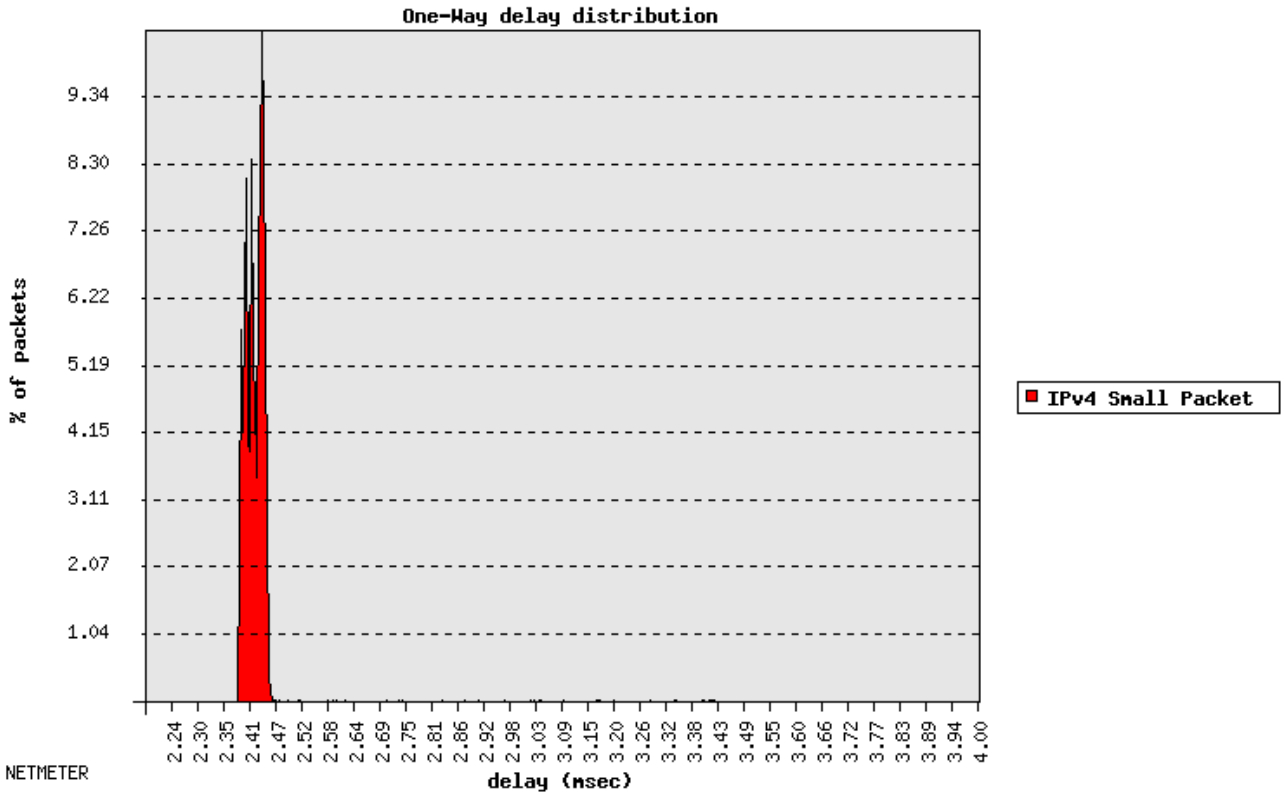


Figure nr. 18 showing Small packet Test result without Background traffic

SOURCE : 147.83.130.169:3594

DESTINATION : 192.168.11.4:3595

Num pkts recvd : 30000

Join delay : 119581.007812 sec

Recv pkt rate : 99.989 pkt/sec

Recv data rate : 31.998 kbps

Pkts dropped : 0

Ave. Pkt Delay : 0.002499 sec

Max. Pkt Delay : 0.052467 sec

Min. Pkt Delay : 0.002382 sec

Delay variation : 0.050085 sec

4.2 Test Analysis

Generally small packets travels faster through a network than big packets because of smaller amount of memory proceeded from program to kernel and less switching time when travelling, but on the other hand risk for packet losses is much higher for small packets. Packet delay times and losses are proportional to utilization of IP network recourses because different data flows interfere each other and have to share same network resources. In overall difference between packet delay times between big and small packets is small, less packet losses and bigger amount of payload data makes big packet size much better to use when sending data.

Results obtained after Performing of OWAMP-Tests shows that implementation of OWAMP draft was successful. In this case results show logical difference in travel time needed for packets with different sizes (smaller packets travels faster) to travel from one point to another in IP network. As expected utilization of background traffic has resulted in higher average One-Way packet delay time regardless of test packet size. (See figure nr. 19)

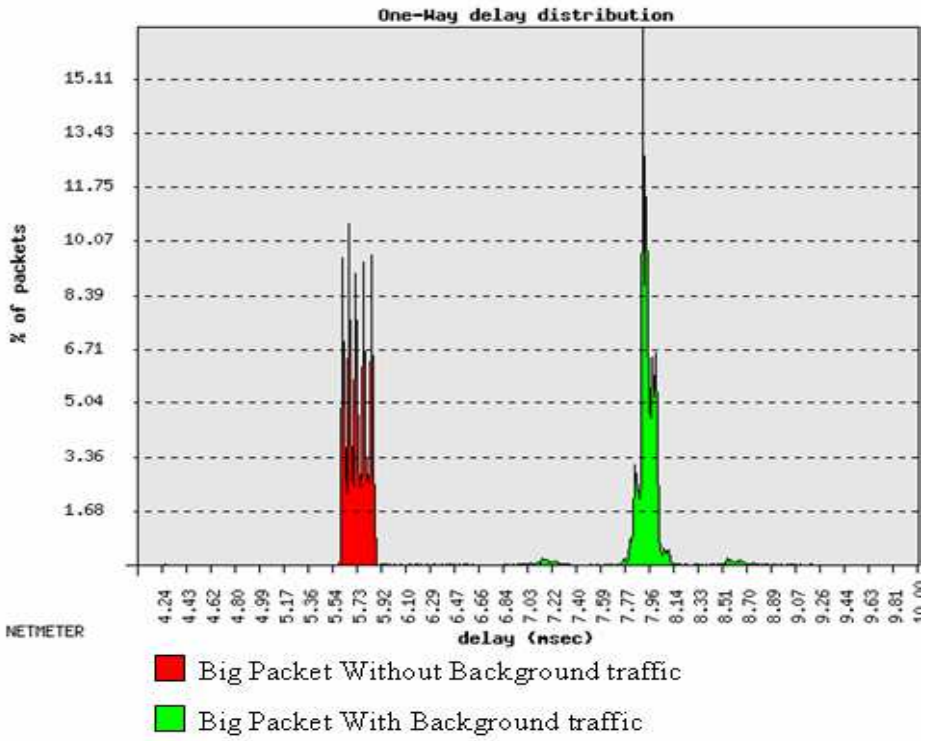


Figure nr. 19 comparing Delay times for Big packets with and without Background traffic without

5 Economical Analysis

This chapter gives overview about proximate total economical cost for whole project including specified amount of working hours spent working with various tasks, the other part that was included in analyses is cost for equipment that was used during the project

5.1 Implementation cost

Implementation of the program is made in several different stages (see figure nr. 20). Contend of each one of the stages and proximate number of working hours needed for accomplishing them is shown below. During the project, help of an analyst was required, which was also added to the total cost.

- **Phase 1:** Analysing different issues and features of Linux(Red Hat) OS-environment
- **Phase 2:** Studding OWAMP - document in order to get the complete overview on protocol and planning the implementation.
- **Phase 3:** Implementing of different logical parts which are included in the OWAMP-Protocol.
- **Phase 4:** Evaluating the complete program and running the test using the actual lab equipment.
- **Phase 5:** Working with documentation and writing the report.

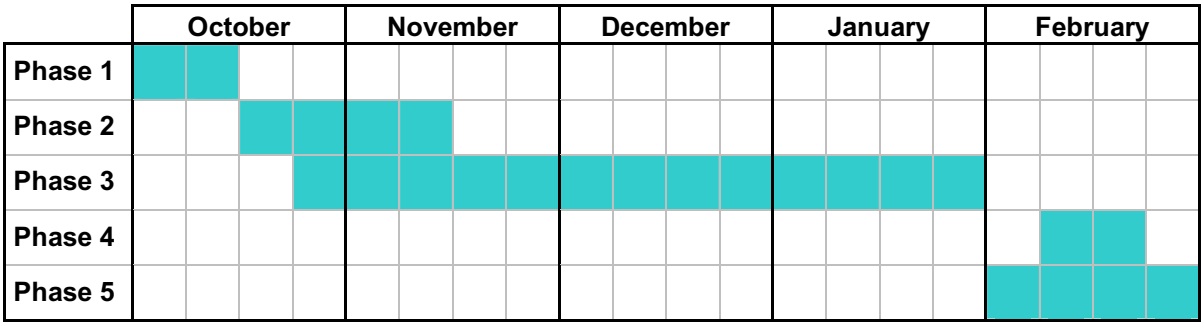


Figure nr. 20

People with different professions were engaged in all phases. Programmer and analyst were working during phases 1, 2 and 3 cost for that is shown below in figure nr. 21.

Profession	*Hours*	*Price per Hour*	Total
Programmer	400	35 €	14000 €
Analyst	60	50 €	3000 €
		TOTAL	**17 000 €**

Figure nr. 21 showing implementation cost

5.2 Cost for program testing

After implementing the program, it was tested during phases 4 and 5 in order to insure that implementation succeeded, cost for that is shown below in figure nr. 22.

Profession	*Hours*	*Price per Hour*	*Total*
Technical support	40	45 €	1800 €
Programmer	160	35 €	5600 €
Analyst	20	50 €	1000 €
		TOTAL	**8400 €**

Figure nr. 22 showing cost for program testing

5.3 Cost for the equipment

To implement and test program required totally three PCs. Implementation was made using only one PC while testing required use of all three PCs.(see figure nr. 23).

Equipment	*Units*	*Price per unit*	*Total price*	*Time of use*	*Cost*
PC	1	600	600	5 month	83
Test PCs	2	600	1200	1 month	33
				TOTAL	**116**

Figure nr. 23 showing cost for the equipment

5.4 Total cost

Total cost for human resources and IT equipment is shown below in figure nr. 24

Element	Cost
Human resources	25400 €
IT equipment	116 €
TOTAL	**25516 €**

Figure nr. 24 showing Total cost

6 Conclusions

Implementing of OWAMP draft with all of its parts went well; all planned goals that were set up at beginning of the project are achieved. OWAMP testes made on existing experimental network were successful where obtained results showed good program behaviour during all of test period.

Features like encryption, authentication and Poisson sampling were left for future work because they are not necessary for basic use of the protocol and test-performing. Even if they were implemented and used when performing OWAMP-Tests the obtained test results would be less accurate, because in encrypted mode both the sequence number and the time stamp the OWAMP-Test packet are encrypted while in authenticated mode the sequence number is encrypted and the time stamp is sent in clear text.

That is why the sender has to fetch the time stamp, encrypt it, and send it. In authenticated mode, the middle step is removed improving accuracy.

Even if implementation works and program is working properly, there are some could have been done differently in order to achieve higher performance of the program. In this case UDP Test-packets sending rate could be higher if Raw-socket were used instead of a normal UDP socket in Session-Sender because the UDP header in Raw-socket is hard coded while in a normal socket new UDP header is made every time a new packet is sent which takes little processing time (delay).

6.1 Future Work

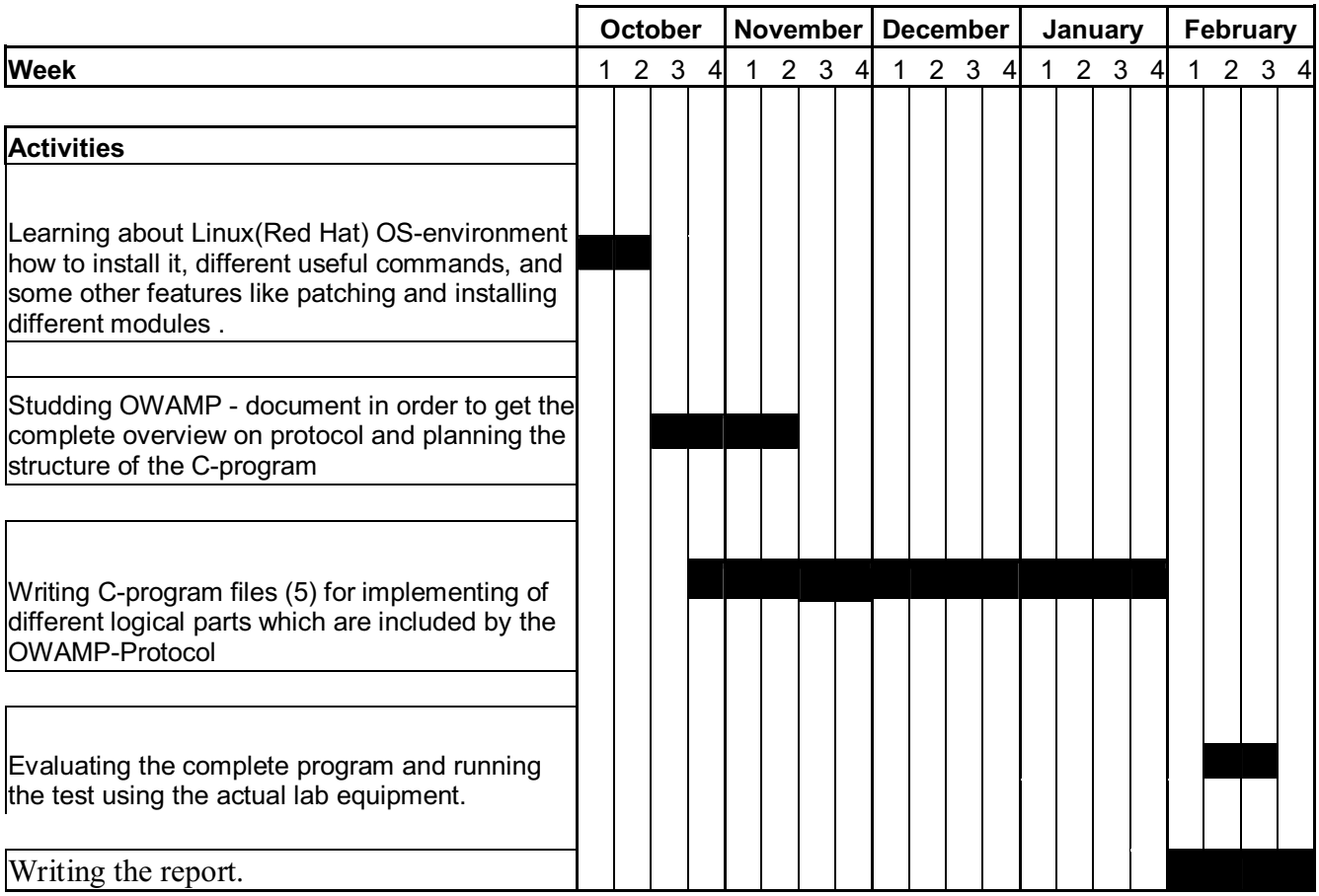
As mentioned before there are some OWAMP features that are left undone like encryption, authentication and Poisson sampling that should be implemented, but maybe there are more important things that could be done before that, like optimizing of the existing program. One way of doing it is making it much faster with higher UDP Test-Packet sending rate by using a Raw-socket. Other important work is adapting OWAMP code in to NetMeter making it possible to configure and perform OWAMP tests using the NetMeter, having OWAMP test option as an extra feature of the NetMeter.

7 References

[1] http://www.ietf.org/internet-drafts/draft-ietf-ippm-owdp-07.txt
[2] http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci214031,00.html//IP
[3] http://www.cs.cornell.edu/boom/2003sp/ProjectArch/PCATTCPC/about1.html//T
[4] http://www.pcworld.com/reviews/article/0,aid,104693,pg,10,00.asp
[5] http://www.ccaba.upc.es/netmeter
[6] http://www.lnf.infn.it/computing/Unix/ntp/

8 Appendix

8.1 Time Plan



8.2 Control-Client Configuration File

File name: Config.cfg

*UNAUTHENTICATED_MODE = 1

Only Unauthenticated Mode Supported[1]

SERVER_ADDRESS = raim.ccaba.upc.es

Server Address

SENDER_ADDRESS = xarello.ccaba.upc.es

Test Sender Address

RECEIVER_ADDRESS = raim.ccaba.upc.es

Test Receiver Address

*CONFIG_SENDER = 0

Sender(send UDPport)NOT Configured By Server

*CONFIG_RECEIVER = 1

Receiver(receiver UDPport) Conf By Server[1]

TEST_START_TIME = 9

Nr of Sec Before Test starts

SHEDULE_SLOTS_TIMES = 1 1 1 1 1 1 1 1 1 1

Send schedule times in microseconds

Nr_OF_TEST_PACKETS = 2000

Nr of Test[UDP] Packets

*Nr_OF_SHEDULE_SLOTS = 10

Nr of Shedule Slots[10]

*ENCRYPTED_MODE = 0

Not Supported[0]

*TYPE_OF_SLOTS = 1

Pseudo Random[0][Not Supported]

AND Fixed Quantity[1]

DIFF_SERVICES_CODEPOINT = 0

Only Best Effort Supported[0]

IP_VERSION = 4

IP Version

*AUTHENTIC_MODE = 0

Not Supported[0]

TEST_SENDER_PORT = 3594

Sender UDP port

TEST_RECEIVER_PORT = 3595

Receiver UDP port

TEST_PACKET_SIZE = 1400

Test Packet Payload Size[Bytes]

TEST_RECEIVE_PACKET_TIMEOUT = 8

Wait Timeout for UDP Packets[Sec]

8.3 Fetch-Client Configuration File

File name: Fetchconfig.cfg

SERVER_ADDRESS = dell.ccaba.upc.es

Server to fetch result from

SEQUENCE_START = 0

Start packet to fetch/0=Fetch All

SEQUENCE_END = 0

End packet to fetch/0=Fetch All

SID = 1

Session Identify Number

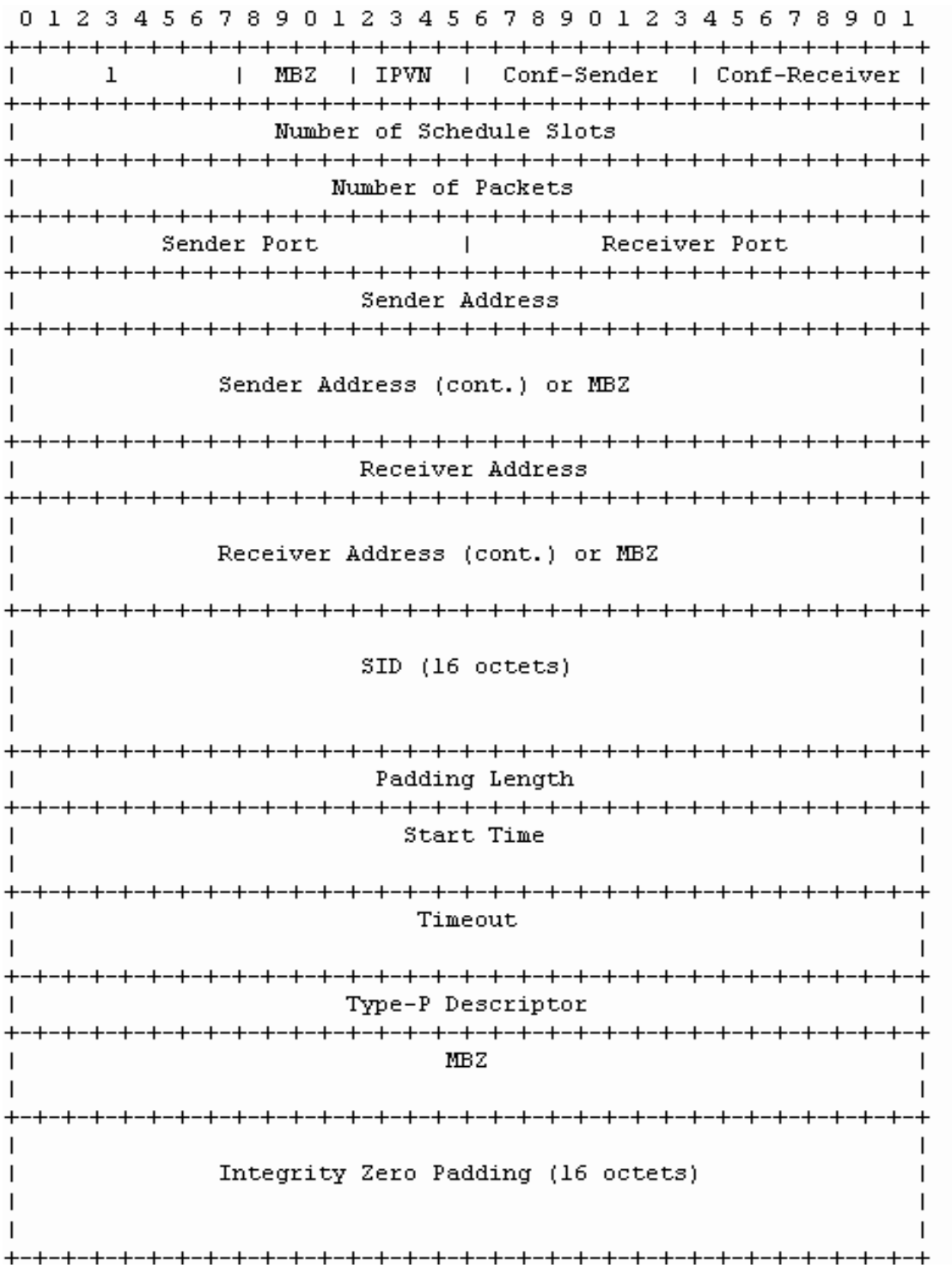
RESULT_FILE_NAME = ipv4-minsize_Netmeter4000_t3.dly

File name for when saving results

NO_PACKET_LOSS = 1

0=Record packet losses/1=NO Record

8.4 The format of Request-Session message



8.5 The format of Result File

```
Flow>0001 Seq>000001 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.003314 RxTime>32:05:07.010483 Size>1476
Flow>0001 Seq>000002 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.013612 RxTime>32:05:07.021797 Size>1476
Flow>0001 Seq>000003 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.023483 RxTime>32:05:07.031393 Size>1476
Flow>0001 Seq>000004 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.033503 RxTime>32:05:07.041515 Size>1476
Flow>0001 Seq>000005 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.043506 RxTime>32:05:07.051509 Size>1476
Flow>0001 Seq>000006 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.053503 RxTime>32:05:07.061509 Size>1476
Flow>0001 Seq>000007 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.063502 RxTime>32:05:07.071509 Size>1476
Flow>0001 Seq>000008 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.073508 RxTime>32:05:07.081507 Size>1476
Flow>0001 Seq>000009 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.083509 RxTime>32:05:07.091510 Size>1476
Flow>0001 Seq>000010 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.093507 RxTime>32:05:07.101515 Size>1476
Flow>0001 Seq>000011 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.103511 RxTime>32:05:07.111513 Size>1476
Flow>0001 Seq>000012 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.113539 RxTime>32:05:07.121513 Size>1476
Flow>0001 Seq>000013 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.123512 RxTime>32:05:07.131516 Size>1476
Flow>0001 Seq>000014 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.133511 RxTime>32:05:07.141515 Size>1476
Flow>0001 Seq>000015 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.143512 RxTime>32:05:07.151518 Size>1476
Flow>0001 Seq>000016 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.153510 RxTime>32:05:07.161518 Size>1476
Flow>0001 Seq>000017 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.163520 RxTime>32:05:07.171519 Size>1476
Flow>0001 Seq>000018 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.173514 RxTime>32:05:07.181520 Size>1476
Flow>0001 Seq>000019 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.183512 RxTime>32:05:07.191522 Size>1476
Flow>0001 Seq>000020 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.193515 RxTime>32:05:07.201522 Size>1476
Flow>0001 Seq>000021 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.203518 RxTime>32:05:07.211521 Size>1476
Flow>0001 Seq>000022 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.213514 RxTime>32:05:07.221518 Size>1476
Flow>0001 Seq>000023 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.223521 RxTime>32:05:07.231524 Size>1476
Flow>0001 Seq>000024 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.233517 RxTime>32:05:07.241525 Size>1476
Flow>0001 Seq>000025 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.243529 RxTime>32:05:07.251526 Size>1476
Flow>0001 Seq>000026 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.253521 RxTime>32:05:07.261523 Size>1476
Flow>0001 Seq>000027 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.263567 RxTime>32:05:07.271572 Size>1476
Flow>0001 Seq>000028 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.273524 RxTime>32:05:07.281530 Size>1476
Flow>0001 Seq>000029 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.283530 RxTime>32:05:07.291531 Size>1476
Flow>0001 Seq>000030 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.293550 RxTime>32:05:07.301553 Size>1476
Flow>0001 Seq>000031 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.303534 RxTime>32:05:07.311539 Size>1476
Flow>0001 Seq>000032 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.313528 RxTime>32:05:07.321537 Size>1476
Flow>0001 Seq>000033 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.323531 RxTime>32:05:07.331539 Size>1476
Flow>0001 Seq>000034 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.333528 RxTime>32:05:07.341536 Size>1476
Flow>0001 Seq>000035 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.343529 RxTime>32:05:07.351539 Size>1476
Flow>0001 Seq>000036 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.353530 RxTime>32:05:07.361539 Size>1476
Flow>0001 Seq>000037 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.363532 RxTime>32:05:07.371540 Size>1476
Flow>0001 Seq>000038 Src>147.83.130.169/3594 Dest>192.168.11.4/3595 TxTime>32:05:07.373534 RxTime>32:05:07.381541 Size>1476
```