Measurement-Based Analysis of the Performance of several Wireless Technologies

Rares Ioan COSMA

Albert Cabellos-Aparicio
Jordi Domingo-Pascual

Table of Contents

MEASUREMENT-BASED ANALYSIS OF THE PERFORMANCE OF SEVERAL WIRELESS TECHNOLOGIES

Rares Ioan COSMA, E-mail: rares.cosma@gmail.com

ABSTRACT

Wireless technologies have rapidly evolved and are becoming ubiquitous. An increasing number of users attach to the Internet using these technologies; hence the performance of these wireless access links is a key point when considering the performance of the whole Internet. In this paper we present a measurement-based analysis of the performance of an IEEE 802.16 (WiMAX) client and an UMTS client. The measurements were carried out in a controlled laboratory. The wireless access links were loaded with traffic from a multi-point video-conferencing application and we measured three layer-3 metrics (One-Way-Delay, IP-Delay-Variation and Packet Loss Ratio). Additionally we estimate the performance of a WiFi and Ethernet client as a reference. Our results show that Ethernet and WiFi have comparable performances. Both the WiMAX and the UMTS links exhibited an asymmetric behavior, with the uplink showing an inferior performance. We also assessed the causes of the discretization which appears in the jitter distributions of these links.

1. INTRODUCTION

WIRELESS technologies have rapidly evolved in recent years. Nowadays, IEEE 802.11 (WiFi) [1] is one of the most used wireless technologies and it provides up to 54Mbps of bandwidth in an easy an affordable way. This technology has been deployed worldwide, from campus to commercial networks and it is becoming ubiquitous.

Recently new wireless technologies have been researched, and deployed. Currently UMTS [2] and IEEE 802.16 [3] (WiMAX) are among the new wireless technologies that stand out. These technologies provide more bandwidth, greater range and are intended to be used as last-mile access links. Additionally these new technologies incorporate mobility. With mobility, a user can connect to the Internet, and move within a limited geographical area without breaking its IP communications.

Currently an increasing number of users are attached to the Internet using one of these technologies. The end-to-end paths used by the user's connections include one, or even two of these wireless access links. That's why we believe that the performance of these technologies is a key point when considering the performance of the whole Internet.

The research community has focused on measuring the performance of these technologies. Initially A. Mishra presented in [4] a measurement-based analysis of the performance of different IEEE 802.11 network interfaces cards. The authors specifically focused on the handover process of WiFi-based networks. Later many other authors provided different analysis, at different levels, of the performance of WiFi [5,6] (and the references therein).

Regarding UMTS, some authors have also measured the performance of these links. An in-depth packet delay analysis and a model for the UMTS packet delay were proposed in [7] and an extensive measurement study on several commercial networks was conducted in [8].

Finally regarding WiMAX, the research community has started to provide empirical measurement-based analysis of its performance. In [9] the authors measured propagation loss at a specific frequency. A model for link throughput based on received signal strength was proposed in [10] and in [11] an experimental WiMAX link was subjected to different load conditions and tested from a transport layer point of view. Mobility performance with the link subjected to various physical phenomena (multi-path, Doppler shift, etc.) was carried out in [12].

In this paper we present a measurement-based analysis of the above-mentioned wireless access technologies. We determine the performance, at the IP layer, of an IEEE 802.11 (WiFi), a commercial UMTS (using HSPA) and an experimental IEEE 802.16 (WiMAX) access link. Additionally we measure the performance of an Ethernet client as a reference. We base our evaluation on the following metrics: One-Way-Delay (OWD), Packet Loss Ratio (PLR) and Inter-Packet Delay Variation (IPDV). These metrics have been defined in RFC 2679, RFC 3393 and RFC 2680 and are considered as standard metrics when measuring the performance of Internet data delivery services.

Our tests were carried out in a distributed laboratory across three Spanish universities (UPC, UPV and UPM) interconnected by the RedIris academic network. Instead of using synthetic traffic to load the different access links we used a video-conferencing application: Isabel [14]. Isabel is intended for large multipoint configurations and it uses an overlay star topology: all the flows are forwarded through a central server (flow-server). During a experiment we run Isabel for 900s with all the clients connected simultaneously. We capture all the packets sent/received at the clients and at the flow-server. With these captures, we are able to compute the above-mentioned metrics using the passive measurement methodology.

Our results show that the performance of the WiFi client is comparable to that of Ethernet. We have also found that the WiMAX client experiences an asymmetric behavior and high packet loss. The UMTS client outperforms the WiMAX client in terms of delay but its low throughput imposes certain limitations. The nature of jitter on both links is discrete due to the MAC Layer retransmission schemes employed.

2. SOFTWARE TOOLS

During the study we make use of several Software tools for different purposes. For injecting traffic into the network we used a videoconferencing application called Isabel [17]. To obtain our four data files, from the flow-server and each client, for the analysis we needed a packet capturing utility. For this purpose we used Wireshark, a PCAP-based packet sniffer. Then we need a way to perform packet matching between the captured files and to extract timestamping information for the selected flows / packets. We wrote a custom Perl script to achieve this. The output of the script was then fed to custom MATLAB scripts for graphical representation and analysis. In this section we propose an overview of these four software tools that we deployed / developed to achieve the goals of our study.

2.1. Isabel

Isabel is a multipoint group collaboration tool for performing distributed congresses, classrooms or meetings over the Internet. The application is highly modular and makes use of different components, services and configuration options to deliver a high Quality of Experience.

A component is every one of the multimedia modules that Isabel uses to perform distributed meetings. Some examples of Isabel components are: The video component, which is able to display local and/or remote video streams in separated windows distributed over the desktop, and, the audio component, which selectively plays the audio signal coming from local or remote sites in a terminal.

Combinations of components are preconfigured to build interaction modes. The available interaction modes for a terminal may vary depending on the service which is being used and the role of that terminal in the session.

As components can configure modes to behave in a certain way, modes are also grouped into **services**. A **service** is a set of specific interaction modes and a set of interaction permissions for the terminals in the session (this is, the capability of changing the active mode or changing some components configuration).

The three main **services** that Isabel supports are Telemeeting, Teleclass and Teleconference. In the Telemeeting mode all participants have the same permissions and the same level of control over the components. It is a mode intended for distributed meetings, and also, the mode we used during our experiments. The Teleclass mode makes use of two roles (Teacher and Students) to build an appropriate environment for E-learning. Finally, the Teleconference mode allows only one site to have the entire control of the session. It is intended for distributed congresses with a pre-established structure.

Regarding configuration, there are two simple wizards that were used to setup the test scenario. First, a new session must be established on the flowserver. To do this, we use the Applications menu of the Ubuntu-based Isabel distribution and go to **Isabel -> Start Session Server**. A panel appears (Figure 1) where we have to fill in then the proper session parameters.

The session server will be started by clicking on "start". Notice the ability to specify the maximum bandwidth that will be used by the Isabel modules. This parameter will affect the quality of the video / audio transmissions and can be changed during a session.

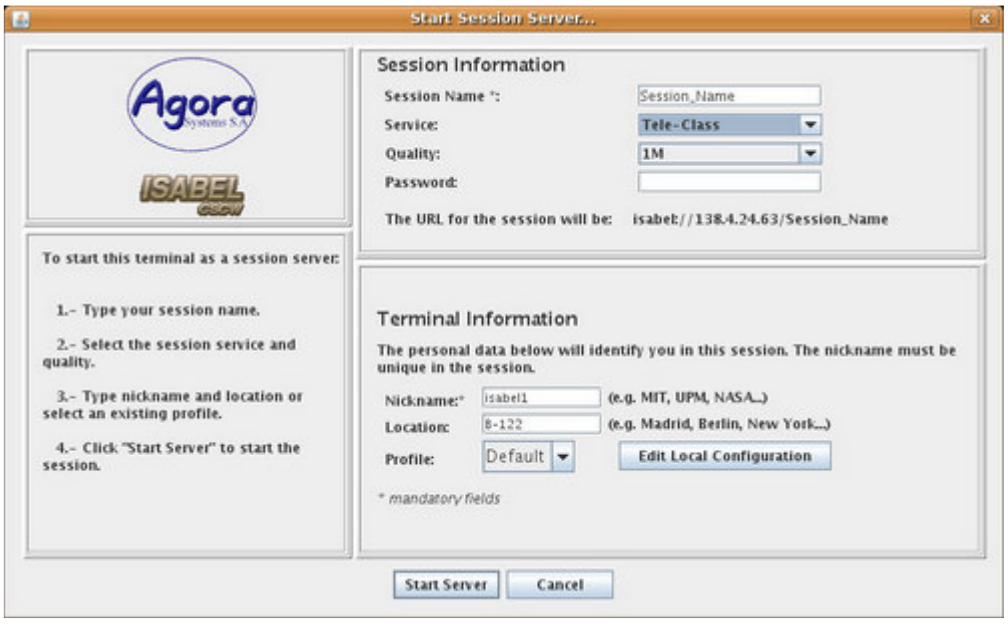

Figure 1. The "Start Session Server" Wizard

On the client side, another Wizard is used to connect to the flowserver, thus establishing an overlaid star topology. To do this, we choose **Isabel -> Connect To** from the Application menu on each of our clients. The panel in Figure 2 appears. We fill in the IP address of our flowserver (84.88.40.26), click "connect" and the session is initiated.

Figure 2. The "Connect to Session" Wizard

2.2. Wireshark

Wireshark is a free packet capture application used for network analysis, troubleshooting, communications protocol development and education. It is a continuation of the Ethereal sniffer which had to change its name in 2006 due to trademark / copyright infringement issues.

It provides functionality similar to **tcpdump**, but it also adds a graphical front-end and numerous information sorting and filtering options. By configuring the network card which is being used for packet capturing to use **promiscuous mode**, it allows the user to see / capture all traffic passed over a network segment. Promiscuous mode is a configuration option of a network card that makes the card pass all traffic it receives to the CPU, rather than just the packets addressed to it. When a network card receives a packet, it checks if the MAC address matches its own. If not, the packet is normally dropped. This is not the case however when promiscuous mode configuration is employed.

Wireshark's native network trace file format is the libpcap format supported by libpcap and WinPcap. This is a very versatile file format that satisfies a wide range of applications. Its advantages come from the fact that the data is organized in blocks that share a common format, and are appended one to another to form the file. Thus it provides an application the ability to easily parse a data file and to skip the data it is not interested in. Another advantage of this format is that two or more files can be concatenated together obtaining another valid file. In our case, the capture files contain a raw dump of the network data, made of a series of Enhanced Packet Blocks. The format of these blocks is shown in Figure 3.

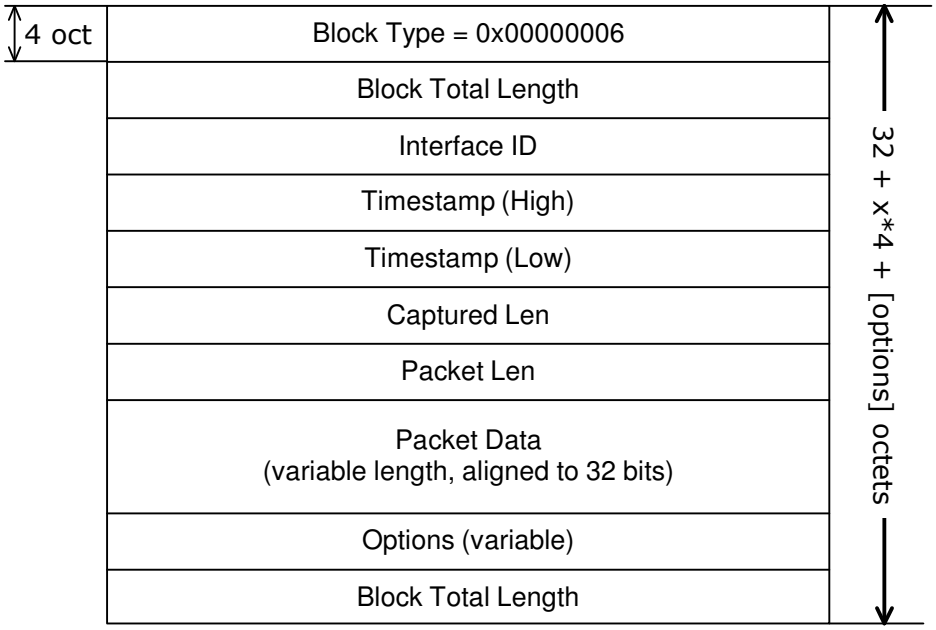

Figure 3. PCAP Enhanced Packet Block

- Block Type: The block type of the Enhanced Packet Block is 6.
- Block Total Length: total size of this block, in bytes.
- Interface ID: it specifies the interface this packet comes from

- Timestamp (High) and Timestamp (Low): high and low 32-bits of a 64-bit quantity representing the timestamp. The timestamp is a single 64-bit unsigned integer representing the number of units since 1/1/1970.
- Captured Len: number of bytes captured from the packet (i.e. the length of the Packet Data field). The value of this field does not include the padding bytes added at the end of the Packet Data field to align the Packet Data Field to a 32-bit boundary
- Packet Len: actual length of the packet when it was transmitted on the network. It can be different from Captured Len if the user wants only a snapshot of the packet.
- Packet Data: the data coming from the network, including link-layer headers. The actual length of this field is Captured Len.
- Options: optionally, a list of options can be present.

In Figure 4 we present the Wireshark GUI showing a packet generated during a Videoconference session. The functionality of the transport layer in Isabel is accomplished by using two protocols: UDP (User Datagram Protocol) and RTP (Real-time Transport Protocol). This aspect is important when considering techniques to classify the packets into flows and to perform packet matching. These aspects will be discussed in the section over-viewing the Perl tool, but for now it is worth noticing the Source and Destination UDP Ports and the Synchronization Source Identifier from the RTP header.

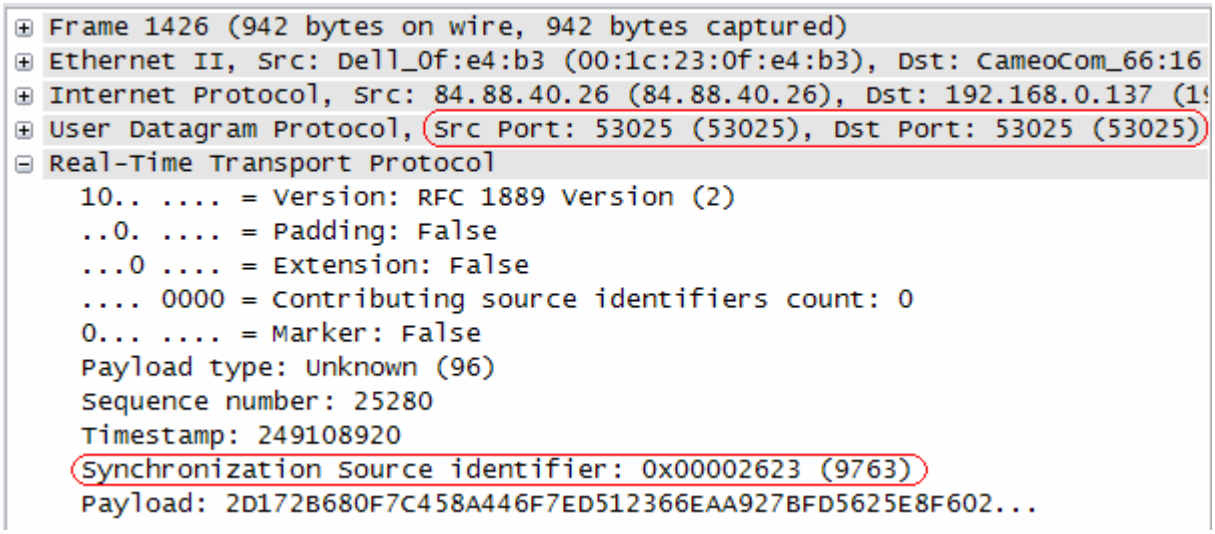

Figure 4. An Isabel-generated Packet in Wiresharks's GUI

2.3. The Perl Tool

For processing the data files captured at the flowserver and the three clients, we wrote a custom Perl script named **rares.pl**. The filtering and extraction of packets from these files is accomplished by using the functions provided by a PCAP wrapper written in Perl, named Net::Pcap.

The functionality of this Perl script can be divided in three parts:

1. Synchronization Checking
2. Flow Listing
3. Packet Matching between source and destination and calculation of OWD and IPDV values

Synchronization Checking

Normally, the clock synchronization between the clients is checked using an NTP-specific tool, called "ntptime". But, to add an extra layer of confidence that these machines are synchronized properly, before proceeding with the videoconference and the data capture, we pinged the flowserver from each client and captured these ICMP packets with Wireshark.

These capture files are then fed to the Perl script using the "-i" command line option. The Perl script identifies the ICMP request packet as it leaves the source host and arrives at the flowserver and computes its OWD. It then identifies the ICMP reply packet as it leaves the flowserver and arrives at the source host and also computes its OWD. The roundtrip time is then derived using the ICMP request and ICMP reply captured at the source host. This is always an accurate measurement since there is only one time source involved. The script then outputs an average of all the one-way delays of the request and reply packets and also the average roundtrip time. The user can then inspect these values to establish if the hosts were properly synchronized or not (Figure 5).

```
karelian@ubuntu:/mnt/d/Master/dataset2/pings$ rares –i upm.eth1.pcap
upc.fs1.pcap
Average OWD1: 0.00870200
Average OWD2: 0.00775875
Average RTT: 0.01647260
```

Figure 5. An example of synchronization checking using the Perl tool

Flow Listing

To separate the relevant data packets from the rest of the bulk traffic, and also, to be able to differentiate between audio and video data, we must classify these packets into flows. First, we must understand how these flows are defined.

There are several ways to define a flow of network traffic between two hosts. The most simple solution is to use a five-tuple containing the source and destination IP addresses, the transport protocol identifier and the source and destination port numbers of the transport protocol. This approach, however, is insufficient when the packets have to travel through NAT or port-forwarding enabled routers. In these cases the destination addresses of packets are altered at the node performing NAT, and the two packets will not be identical in the two capture files.

Therefore, we needed to come up with a new way of packet classification, that focuses on information present at the transport layer of the considered packets. One solution is to use the SSRC (Synchronization Source) field of the RTP header. This is an unique identifier assigned to each host participating in the conference, and it doesn't change its value during one session.

Thus, we needed to add functionality to our tool to list all the flows present in a packet file, based on these new criteria.

Each flow was identified by a key consisting in: source IP and source UDP port, destination IP and UDP port, and the RTP SSRC field. If a packet doesn't match any existing flow keys, a new key is generated and the packet is considered as belonging to this new flow. On the other side, whenever a packet *matching* a flow key is encountered, its length is added to the flow length, and its timestamp is compared against the minimum and maximum recorded timestamps for that flow, in order to establish if it was the first or last packet of the flow. This is important for calculating the flow throughput. Figure 6 presents the output of the Perl script in this mode, when fed a small test Pcap file.

```
karelian@ubuntu:/mnt/d/Master/scripts$ rares -f test.pcap
192.168.0.137:53023 84.88.40.26:53023 9765       62 [bytes]       1 [packets]
0 [bps]
192.168.0.137:53021 84.88.40.26:53021 9765       1470 [bytes]     5 [packets]
91913.8479886107 [bps]
192.168.0.137:53025  84.88.40.26:53025  9765           800838  [bytes]    1009
[packets]  339776.354441643 [bps]
84.88.40.26:53025 192.168.0.137:53025 9763      170492 [bytes]  202 [packets]
234188.976152668 [bps]
84.88.40.26:53023 192.168.0.137:53023 9763       1178 [bytes]     19 [packets]
1043.71914532372 [bps]
84.88.40.26:53025 192.168.0.137:53025 9764      190522 [bytes]  252 [packets]
196384.849819963 [bps]
```

Figure 6. Example of flow listing

Packet Matching and Calculation of Metrics

Using the information provided by the Flow Listing mode, we were able to draw some conclusions of major importance for the classification of the test traffic:

1. Isabel uses three different UDP port numbers for its audio, video and control connections
2. Each client has its own SSRC which doesn't change during a session
3. The throughput of the video flows is generally higher than that of the corresponding audio flows

Armed with this knowledge, we wrote the last part of the Perl script, that deals with packet matching between the different capture files, and evaluates the OWD, IPDV and packet loss for a given link.

As input arguments, we provide the source and destination PCAP files, and a list of flows for which we want the metrics to be evaluated. A flow is identified here solely by its UDP port number and the SSRC field. Note that the IP addresses of the sending and receiving host are no longer needed, since the sending host is identified by its SSRC and the receiving host is explicitly set by specifying the destination PCAP file.

The script parses the source and destination pcap files and loads into hash data structures all packets matching the aforementioned criteria (UPD port and SSRC tuples). It then sorts the source flow and starts searching for packets in the destination flow. This is accomplished by

using a digest of the UDP payload, since it's the only part that stays unchanged during all sorts of NAT or port forwarding processes. Whenever a match is found, the OWD is evaluated by subtracting the timestamps of the matched packets. IPDV is then evaluated by subtracting consecutive OWD values. If the sent packet wasn't found in the destination flow, a packet loss counter is incremented.

Additionally, a set of descriptive statistics for these metrics can be evaluated. These include the minimum, maximum, median, mean and standard deviation for OWD, IPDV and packet size. An short excerpt of a result file is shown in Figure 7. The columns are, in order: sequence number, source timestamp, destination timestamp, one-way delay, interpacket delay variation and packet length.

```
0 0.0000000000 0.0631880760 0.0631880760 0.0000000000 294
1 0.0005350113 0.0638110638 0.0632760525 0.0000879765 294
2 0.0011169910 0.0642869473 0.0631699562 -0.0001060963 294
3 0.0016679764 0.0646860600 0.0630180836 -0.0001518726 294
4 0.0022120476 0.0651850700 0.0629730225 -0.0000450611 294
5 0.0646250248 0.1225988865 0.0579738617 -0.0049991608 294
6 0.0648729801 0.1230280399 0.0581550598 0.0001811981 294
```

Figure 7. Result file format

2.4. The MATLAB Script

The result files from the Perl tool were processed using the MATLAB integrated mathematics environment. A custom **.m** file was developed for performing various statistical analysis on the data and for graphically depicting the results. The functionality of this script includes:

- Evaluation of the histogram and the Cumulative Distribution Function for the one-way delay and the IPDV. For this we used the MATLAB built-in functions `cdfplot()` and `histc()`.
- Evaluation of certain percentile values and other statistical descriptors for these metrics. The percentile values are used to automatically zoom the histogram and cdf plot to a restricted range, so they don't get affected by outlier values. The other statistical descriptors (mean, median, standard deviation) were used for building the tables shown in the results section.
- Calculating the variation of IPDV and OWD with respect to packet size. Video streams were used for this purpose because they contain a wide range of packet lengths. For a given packet length, we evaluated the average of OWD and IPDV values. The median of several consecutive values was then calculated in order to smooth these graphs.

3. METHODOLOGY

This subsection details the test-bed deployed to collect the measurements, the methodology used to compute the performance metrics and provides an overview of the captured traffic used in the following analysis.

3.1. Test-Bed

Figure 8 presents our distributed test-bed among three universities (UPC, UPM and UPV). All three universities are connected through RedIris, the Spanish academic research network (NLANR).

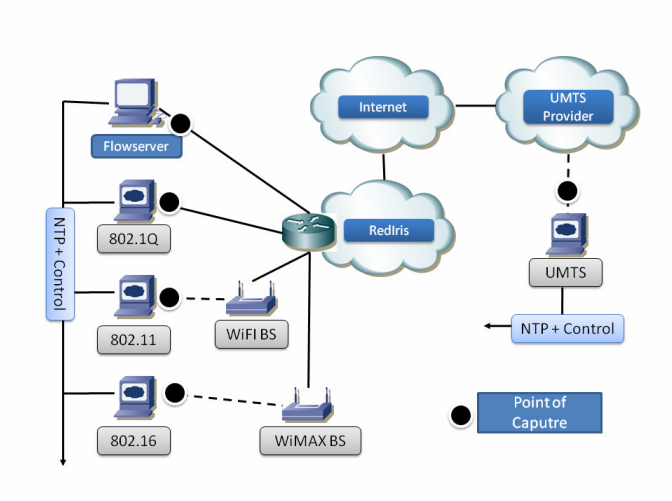

Figure 8. Test-bed Configuration

The test-bed includes five terminals. Three of them are equipped with an IEEE 802.11, an IEEE 802.16 and an UMTS interface. Another one is equipped with an Ethernet (as reference). The last one is the Isabel flow-server. All the machines are Linux-based, at least Pentium III with 1GB of RAM.

Isabel uses an overlay network to transmit live audio and video streams. All the streams are routed through the flow-server, creating an overlay star-topology. Each client transmits both audio and video, which is received by the rest of the clients. The microphone and the camera of each client transmit a movie, to setup a realistic environment.

Regarding synchronization each client is configured to use three NTP (Network Time Protocol) sources [16]. At least each client is connected to a Stratum 1 server which is in turn connected to a GPS source. The other two sources are on the outside network. All the NTP traffic is routed through a parallel network (with the local NTP servers). It is possible to access those remote NTP servers through the control network that can use external time sources. The NTP statistics show that, with this setup, we obtain a measurement accuracy of 1ms. Since the delays of the different measured links are higher this accuracy suffices.

3.2. Passive Measurement Methodology

The main focus of the tests was the estimation of the following metrics: One-way Delay (OWD), IP Delay Variation (Jitter), and Packet Loss Ratio (PLR). With this range of parameters we can determine the ability, reliability and the robustness of each wireless technology:

- OWD represents the time that a packet takes to travel through the network from source to destination. For a real-time application, it is important that this parameter stays below a threshold value (RFC 2679).
- IPDV refers to the variation of a packet's one-way delay in respect to the one-way delay of the previous packet (we assume here that both packets belong to the same flow). Erratic variation in delay makes it difficult (or impossible) to support many real-time applications (RFC 3393).
- PLR is a percentage ratio of the number of data packets lost to the total number of packets transmitted by the user into the network (RFC 2680).

As pointed out earlier, the probe traffic was generated using a videoconferencing application. This approach allows us to estimate the performance of the network technologies using both the Constant Bit Rate (CBR) nature of the audio stream and the Variable Bit Rate (VBR) nature of the video stream.

Isabel uses the Real-time Transport Protocol (RTP) over the User Datagram Protocol (UDP) to send its data. As shown in [10], UDP is the most appropriate protocol for determining the uplink/downlink performance of an access technology, since the lack of acknowledgments eliminates the interdependency between the bit rates in the two directions.

The port numbers assigned for the transmission of audio and video flows are known, and the SSRC (Synchronization Source) field of the RTP header has an unique value for each client. Based on this information we developed a custom tool that can load and match packets from the source and destination capture files, even if the packets passed through a point where Network Address Translation was used. The tool uses PCAP's timestamp from each matched packet pair to compute the aforementioned metrics.

3.3. Captured Data Overview

The traffic was captured for the subsequent offline analysis during two test sessions, each one with a duration of approximately 900 seconds.

We used two profiles for the videoconferencing application with different quality settings. The high quality profile was switched on for 300 seconds during each test, and the rest of the test was conducted with the lower quality profile.

During a test session Isabel generated ~21000 and ~32000 packets corresponding to average bit rates of 111.212 kbps and 69.684 kbps for the video and audio flows, respectively.

4. TESTS AND RESULTS

4.1. Ethernet

Table I shows the minimum, maximum, mean, median and standard deviation of the one-way delay and the jitter experienced between the Ethernet client and the flow-server. These parameters are evaluated using the video and audio streams in both the uplink and the downlink directions. The minimum and maximum values are taken as the 2.5 and 97.5 percentiles of the distributions.

TABLE I
ETHERNET CLIENT METRICS STATISTICS

Metric		Max	Mean	Median	Min	Dev
Video Up	OWD	11.5483	9.7942	9.49407	8.9369	1.2901
	IPDV	0.3541	0	0.02193	-1.0117	1.0018
Audio Up	OWD	9.4810	9.0133	8.89993	8.6751	0.6347
	IPDV	0.3221	0	0.01597	-0.6061	0.6613
Video Down	OWD	8.7628	8.4993	8.56996	7.8580	0.3236
	IPDV	0.6180	0	-405 µs	-0.7380	0.3637
Audio Down	OWD	8.2281	8.0075	7.97701	7.8502	0.1580
	IPDV	0.1991	0	-95 µs	-0.3399	0.1745

All values are in milliseconds, unless specified otherwise.

In the *uplink* direction this client performed well, with a low median (50 percentile) value of the OWD (9.49ms). The IPDV distribution is zero-centered with a median value of under 0.03ms. The PLR was 0.088% for the video stream and 0.006% for the audio stream.

In the *downlink* direction the client had to receive audio and video streams from all other three clients. Nevertheless its performance remained high, scoring a 8.499ms median value for the OWD and a well-shaped IPDV distribution. However, we recorded a higher PLR of 0.507% for the video streams and 0.006% for the audio streams.

When we analyzed the OWD and IPDV in respect to the packet length, we saw a slight increase of the IPDV (0.15ms) as the packet size increased from 100 to 900 bytes and a relatively flat OWD characteristic.

As expected, the low delay and jitter values of the Ethernet client enabled it to deliver a high QoE (Quality of Experience) [17] for both the video and the audio transmissions.

4.2. Wifi 802.11

Next we analyzed the performance of the WiFi client. Like in the Ethernet scenario, we built a table showing the minimum, maximum, mean, median and standard deviation of the OWD and IPDV experienced between the Wifi client and the flow-server, and evaluated using the video and audio streams in both the *uplink* and the *downlink* directions.

TABLE II
WIFI CLIENT METRICS STATISTICS

Metric		Max	Mean	Median	Min	Dev
Video Up	OWD	20.3739	4.6778	1.00303	0.5501	7.1284
	IPDV	1.4438	400 µs	0.06485	-3.4165	2.1437
Audio Up	OWD	3.3201	1.0819	0.58794	0.4001	1.7777
	IPDV	0.9828	0	0.01597	-2.0391	1.4886
Video Down	OWD	2.8499	1.2784	0.89598	0.5169	2.0727
	IPDV	0.6863	0	0.05913	-1.7739	1.4879
Audio Down	OWD	1.8242	0.8957	0.60296	0.5112	1.6164
	IPDV	0.6070	0	0.01812	-1.5716	1.4064

All values are in milliseconds, unless specified otherwise.

The MAC Layer of the WiFi link employs a positive acknowledgement system [1], i.e. retransmissions occur if a frame is not acknowledged within a given amount of time. This may explain the higher recorded maximum values for the one-way delay.

In the *uplink* direction the performance of the Wifi client was very good with respect to the Ethernet link, even outperforming the latter with a median value of 1.003ms for the video stream and 0.587ms for the audio stream, close to the minimum values for the computed metrics. The standard deviation values were higher, but in normal limits given the overall more erratic behavior of a radio link compared to a wired link. In terms of PLR, the link proved stable with 0% packet loss for the audio stream (again due to MAC Layer retransmissions) and as little as 0.034% for the video stream.

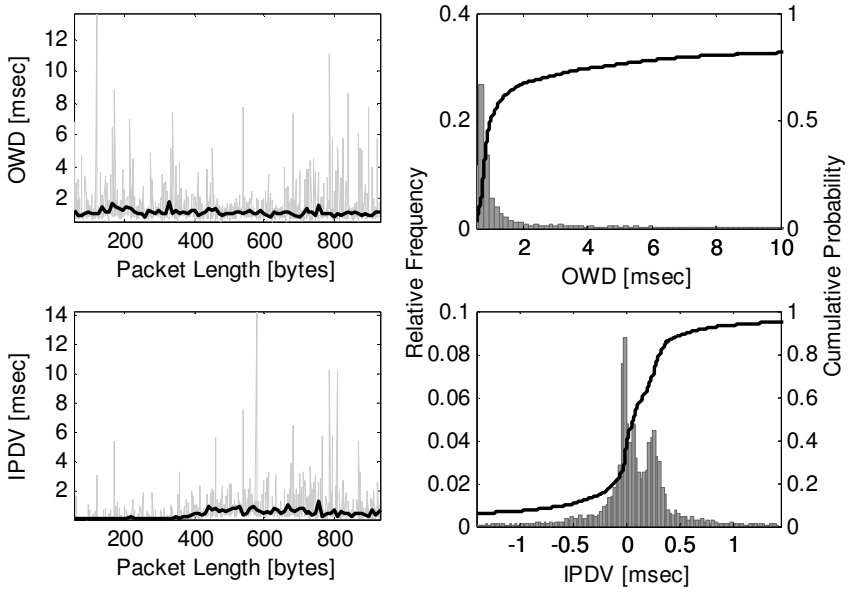

Fig. 9. OWD and IPDV for the Wifi downlink

In the *downlink* direction the client experienced overall lower values for one-way delay. The packet loss ratio was 0.380% for the audio stream and 0.879% for the video stream.

Regarding the relation between the OWD, IPDV and packet size, the WiFi link demonstrated good stability on the *downlink* direction with nearly no variation at all. On the *uplink* direction the OWD characteristic remained unaffected by the packet length, and the IPDV showed increased values with packet lengths greater than 400 bytes (figure 9).

As expected from a client using the now mature 802.11 technology, the Quality of Experience was excellent on both the audio and video transmissions.

4.3. WiMAX 802.16

The performance of the WiMAX client was estimated using the same set of statistics for the OWD and the IPDV as shown in Table III.

TABLE III
WIMAX CLIENT METRICS STATISTICS

Metric		Max	Mean	Median	Min	Dev
Video Up	OWD	275.118	150.672	134.998	54.0566	77.686
	IPDV	49.587	0	8.93307	-80.051	35.0056
Audio Up	OWD	244.949	107.593	81.3001	42.9605	67.9576
	IPDV	39.1852	400 µs	4.61698	-62.378	27.819
Video Down	OWD	158.730	78.5673	67.1921	55.2515	34.5139
	IPDV	10.1601	-300 µs	0.57006	-22.082	9.5404
Audio Down	OWD	104.967	68.0588	62.1218	54.3220	23.1352
	IPDV	8.6916	0	0.27585	-16.799	8.5142

All values are in milliseconds, unless specified otherwise.

The first thing we notice in the statistics is the discrepancy of the delay and jitter values between the *downlink* (from Base Station to Mobile Station) and *uplink* (from MS to BS) directions. The median, mean and minimum one-way delay values of the *downlink* are closely packed within two intervals spanning across 14 and 23ms for the audio and video flow, respectively. In contrast, the same assumed intervals span across 65 and 96ms for the *uplink*, indicating the presence of more outlier delay values. This is further consolidated by the higher standard deviation values of all distributions for the *uplink* metrics. We provide a visual description of this phenomenon in figures 10 and 11, where we present histograms and CDF plots of the one-way delay of packets from the video flows. Different performances in the uplink and downlink directions were also recorded in [10] where the authors have found much lower *uplink* bit rates compared to the corresponding *downlink* bit rates for distances above 2 km. As they point out, the principal reason for this asymmetric behavior is that the power amplifier in the user terminal can only deliver a maximum of 20 dBm, compared to the 28 dBm for the amplifier in the Base Station.

We next take a look at the different performances of this link when subjected to audio (CBR) and video (VBR) traffic. As expected, the higher overall throughput demand and the variable bit rate of the video flow have a negative effect on the link's performance. The recorded differences in the mean values of the one-way delay were ~43 ms for the *uplink* and ~10 ms for the *downlink*, consistent with the asymmetry described in the previous paragraph.

In order to better characterize this behavior we next analyzed the stability of the delay and the IPDV in respect to the packet size. In the uplink direction the OWD characteristic had a slight parabolic shape with an optimal packet size of 500 bytes, and the IPDV modulus showed a monotone increase from ~5ms at 100 bytes to ~30ms at 500 bytes and beyond (figure 10). An interesting result was recorded when analyzing the IPDV of the downlink: as shown in figure 11, we found high instability with small packet sizes (<125 bytes).

Next we analyze the IPDV on the WiMAX link by computing its histograms. We can easily observe that the nature of both the *uplink* and the *downlink* delay variation is discrete (figures 10 and 11). The explanation for this discretization lies in our setup and in the Hybrid ARQ scheme employed by WiMAX's MAC layer for providing reliability. The link was operating in a TDD (Time Division Duplexing) mode, with a frame length of 5ms. This imposes a base extra delay of 10ms for any frame that is retransmitted. Subsequent failures and retransmissions show up as delay variations multiple of the 10ms base value (figure 10). This is consistent with the plots presented in [13], where the authors comparatively analyzed the original WiMAX HARQ scheme and their improved version. The average values of waiting time (the duration from the time the first copy of a data burst is received by the SS until the correct data burst is sent, in sequence, to the upper layer) found by the authors are always multiples of 10ms.

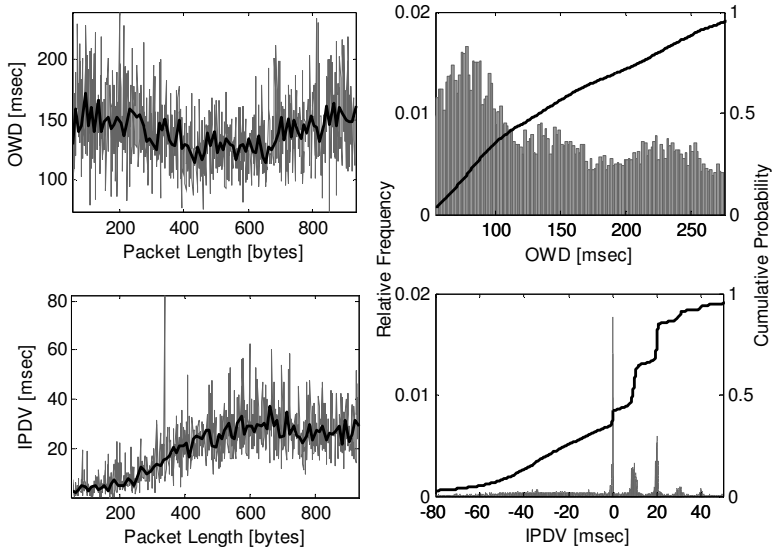

Figure 10. OWD and IPDV for the WiMAX uplink

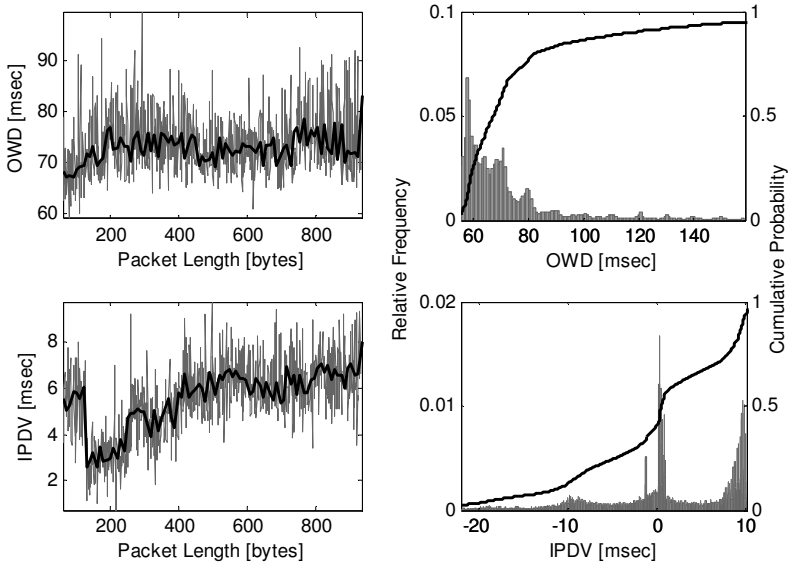

Figure 11. OWD and IPDV for the WiMAX downlink

The multiple spikes present in the *uplink* IPDV histogram at 10, 20, 30 and even 40 ms show that for a significant number of frames, more than one retransmission was required.

In terms of Packet Loss Ratio the link demonstrated the same asymmetric behavior: 0.857% and 6.540% on the audio and video *downlink* compared to 3.237% and 10.621% on the audio and video *uplink*. This surprising result may be the effect of the experimental nature of our WiMAX implementation.

As for the QoE, this client offered satisfactory results for audio, but on the video channels, we experienced interruptions and image blockiness.

4.4. UMTS

The last step of our analysis was the UMTS (HSPA) client. Like in the WiMAX case, we observed a highly asymmetric tendency between the *uplink* and *downlink* computed metrics. Table IV shows the statistics for the video (Variable Bit Rate) stream.

TABLE IV
UMTS CLIENT METRICS STATISTICS

Metric		Max	Mean	Median	Min	Dev
Video Up	OWD	258.251	153.877	95.3851	75.9034	263.481
	IPDV	27.7925	-258 µs	1.0581	-66.573	30.2289
Video Down	OWD	98.1764	74.2441	67.5008	50.0475	43.5242
	IPDV	11.7511	-1.2 µs	1.8251	-22.244	15.3494

All values are in milliseconds, unless specified otherwise.

As the access setup most commonly offered by commercial UMTS providers consists of a dedicated channel with up to 64 kbps and 384 kbps for *uplink* and *downlink* transmissions respectively, packets from the time interval when we used Isabel's higher quality profile were discarded and the UMTS client was setup not to send any audio data, thus avoiding congestion in the *uplink* direction.

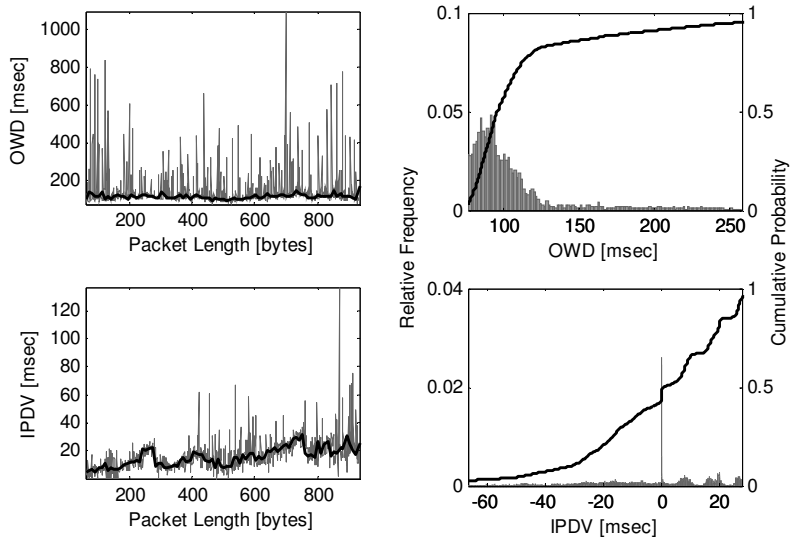

Figure 12. OWD and IPDV for the UMTS uplink

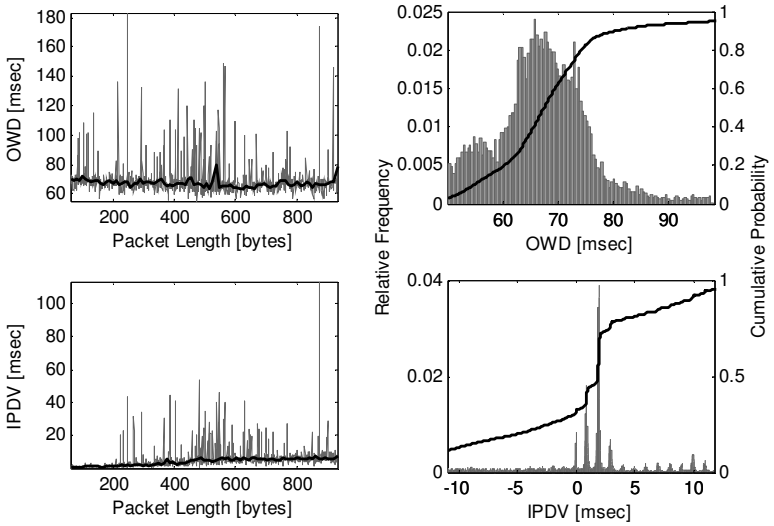

Figure 13. OWD and IPDV for the UMTS downlink

On the uplink the UMTS link experienced a maximum (97.5 percentile) OWD of 258.25ms. As pointed out in [8], these outlier delay values are normal for a commercial UMTS link, if we take into account that under fully-loaded network conditions, the average latency for 3G data services can increase to beyond 1s. However the IPDV values on the UMTS uplink were smaller than those of the WiMAX uplink with a surprising 1.05ms median value, and, the same interesting result regarding the discrete nature of the jitter histogram was observed (see figure 12). This is an effect of the UMTS link temporization employed at the RLC (Radio Link Control) sublayer of the MAC Layer. As the authors of [7] point out, the RLC layer always requires an integer number of TTI's (Transmission Time Intervals) to recover from a loss. When a lost frame is recovered RLC can deliver to upper layers several packets in one go. If these packets were transmitted at regular intervals, like in our case, discretization would occur. Another positive aspect was the stability of the uplink OWD and IPDV in respect to the packet size (see figure 12).

In the downlink direction the client performed better: the maximum OWD value was 98.17ms (compared to 258ms for the uplink), the mean value was close to the median, and the overall spread of the data was smaller (43.524ms standard deviation). The IPDV distribution had a mean value of -1.2 μs, close to the ideal 0 and the harmonics were again observed in its histogram (figure 13). The link demonstrated good stability of the metrics in respect to the packet size: a flat characteristic for the OWD and a slight end-to-end increase in jitter.

Given the packet loss ratio, we can conclude that the ARQ mechanisms employed by the UMTS link are efficient. The recorded values were 0.250% and 0.242% for the downlink, respectively uplink. This is consistent with the results depicted in [7] were the authors have found that the packet losses never climbed over 0.5%.

Although the bandwidth limitations of the UMTS link for data services imposed a special setup, the Quality of Experience was satisfactory with the lower quality settings.

5. CONCLUSIONS

In this paper we have presented a measurement-based comparison of the performance of three wireless access technologies: an IEEE 802.11 Wifi, an IEEE 802.16 WiMAX and a commercial HSPA-based UMTS link. Our analysis is based on statistical interpretations of principal delay measurements: one-way delay and inter-packet delay variation. Packet loss was also taken into account as we tried to emphasize on key asymmetries and differences between the access technologies.

Link reliability and stability in respect to packet size variations have proved that WLAN is a suitable technology for a generic videoconferencing application.
The WiMAX client lacked in reliability. High packet loss made this technology unsuitable for our Isabel application. An overall asymmetric downlink/uplink behavior was present, and we have seen uplink instability with varying packet sizes. We have also seen the jitter of the WiMAX link has a discrete nature because of frame retransmissions. The OWD and IPDV values were higher than those of UMTS and Wifi. The recorded PLR may be the effect of the experimental nature of our WiMAX implementation.

The UMTS link exhibited asymmetric behavior, a discrete nature of the delay variation was seen, but, under normal loading conditions, the commercial setup has shown very low jitter values. Extreme delay values can be the result of high network load. A probable better way to mitigate the low data bit rates offered by this link and the demands of high-quality videoconferencing would be the integration of these services with the inherent 3G video call capabilities.

REFERENCES

[1] WLAN – IEEE Std 802.11-2007, “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications”
[2] UMTS – “Radio interface protocol architecture,” 3GPP, Technical Specification 25.301-v4.3.0 Release 4, June 2002.
[3] WiMAX – IEEE Std 802.16e-2005 and IEEE Std 802.16-2004/Cor1-2005, “Air Interface for Fixed and Mobile Broadband Wireless Access Systems”
[4] A Mishra, M Shin, W Arbaugh, "An empirical analysis of the IEEE 802.11 MAC layer handoff process" - ACM SIGCOMM Computer Communication Review, 2003
[5] Amit P. Jardosh et al. "Understanding link-layer behavior in highly congested IEEE 802.11b wireless networks", ACM SIGCOMM 2005
[6] Albert Cabellos Aparicio, René Serral-Gracià, Loránd Jakab, and Jordi Domingo-Pascual "Measurement Based Analysis of the Handover in a WLAN MIPv6 Scenario" PAM 2005, Boston, USA.
[7] Jose Manuel Cano-Garcia, Eva Gonzalez-Parada, and Eduardo Casilari, “Experimental Analysis and Characterization of Packet Delay in UMTS Networks”, NEW2AN 2006
[8] Wee Lum Tan, Fung Lam, Wing Cheong Lau, “An Empirical Study on 3G Network Capacity and Performance”, Proceedings of IEEE Infocom May 2007
[9] Imperatore, P. Salvadori, E. Chlamtac, I. "Path Loss Measurements at 3.5 GHz: A Trial Test WiMAX Based in Rural Environment", TridentCom 2007
[10] Grondalen, O. Gronsund, P. Breivik, T. Engelstad, P, "Fixed WiMAX Field Trial Measurements and Analyses", Mobile and Wireless Communications Summit 2007
[11] Yousaf, Faqir Zarrar Daniel, Kai Wietfeld, Christian, "Performance Evaluation of IEEE 802.16 WiMAX Link With Respect to Higher Layer Protocols", ISCWS 2007
[12] Jurianto, Joe Hazra, S. K. Toh, S. H. Tan, W. M. Pathmasuntharam, Jaya 5.8 GHz Fixed WiMAX Performance in a Sea Port Environment, IEEE VTC 2007
[13] M. Moh, T. Moh, Y. Shih, “On Enhancing WiMAX Hybrid ARQ: A Multiple-Copy Approach”, Proceedings of IEEE CCNC 2008
[14] Isabel - http://www.agora-2000.com/pdfs/isabel_sheet_en.pdf
[15] LibPCAP - http://www.tcpdump.org/
[16] Internet2 Consortium: OWAMP - NTP Configuration http://e2epi.internet2.edu/owamp/details.html#NTP (2004)
[17] Testing MPEG based IP video QoE/QoS http://www.shenick.com/pdfs/Testing_MPEG_IPTV_VOD_QOE.pdf

Appendix A – Reference Manuals

1. The Perl Script

Synopsis

```
./rares.pl –i source_pcap destination_pcap                          (1st form)
./rares.pl –f pcap_file                                             (2nd form)
./rares.pl –r flow_id_1,... [-s] source_pcap destination_pcap       (3rd form)
```

Description

In the 1st form, the utility evaluates the one-way delays of the ICMP echo request and reply messages present in the source and destination pcap files. It then computes the round-trip time for all echo request-reply pairs and displays the average values for these metrics.

In the 2nd form, the script returns a summary of all UDP/RTP flows present in a capture file. The flows are classified by a five-tuple: source IP address, source UDP port, destination IP address, destination UDP port and the Synchronization Source (SSRC) field from the RTP header. The summary includes this flow identity, the flow length in bytes and packets, and the flow throughput.

In the 3rd form, the script does packet matching between the source and destination pcap files based on the flow id's given as arguments. The format for a flow ID is `port:ssrc`. Multiple flow id's can be specified. If the option `[-s]` is used, the script will also output statistics for OWD, IPDV, packet length and throughput.

Workflow Example

Suppose we want to compute the OWD and IPDV for the video flows on the WiMAX downlink. The capture files are named according to their respective links: `wimax.pcap, wifi.pcap, eth.pcap, umts.pcap` and the flowserver capture file is named `fs.pcap`.

We first need to determine the identity of these video flows. This means, we need to find out what SSRCs were used by the WiFi, UMTS and Ethernet clients during the test session. Another thing we must find out is the UDP port number associated with the transmission of video data. We already know the topology of the network (see the table), so we can run the **rares.pl** in the 2nd form (-f option) on the WiMAX pcap file to see what SSRCs are associated with the other clients. We can also sort this summary in ascending order of flow throughput.

Network Node	**IP Address**
flowserver	84.88.40.26
ethernet client	138.4.24.109
wimax client	158.42.255.180 / 10.1.24.2 (NAT)
wifi client	84.88.40.28
umts client	84.88.40.28 / 192.168.0.137 (NAT)

We execute the command:

```
./rares.pl -f wimax.pcap | sort -n -k 8
84.88.40.26:53023 10.1.24.2:53023 2799  62 [bytes]      1 [packets]     0
[bps]
10.1.24.2:53023 84.88.40.26:53023 2797  86552 [bytes]   1396 [packets]
712.361771930419 [bps]
84.88.40.26:53023 10.1.24.2:53023 2800  124124 [bytes]  2002 [packets]
1023.13695317195 [bps]
84.88.40.26:53023 10.1.24.2:53023 2798  126480 [bytes]  2040 [packets]
1027.61645786489 [bps]
10.1.24.2:53029 84.88.40.26:53029 2797  3440 [bytes]    40 [packets]
2405.61755591261 [bps]
84.88.40.26:53021 10.1.24.2:53021 2799  1470 [bytes]    5 [packets]
53393.1455886938 [bps]
84.88.40.26:53025 10.1.24.2:53025 2798  6522987 [bytes] 9473 [packets]
53413.0966755407 [bps]
10.1.24.2:53021 84.88.40.26:53021 2797  6594126 [bytes] 22429 [packets]
54265.592203588 [bps]
84.88.40.26:53021 10.1.24.2:53021 2800  9402414 [bytes] 31981 [packets]
77477.4654917522 [bps]
84.88.40.26:53021 10.1.24.2:53021 2798  9572346 [bytes] 32559 [packets]
77732.720655864 [bps]
84.88.40.26:53025 10.1.24.2:53025 2799  13797253 [bytes]        18922
[packets] 114471.869318717 [bps]
10.1.24.2:53025 84.88.40.26:53025 2797  15813456 [bytes]        21005
[packets] 129649.381310021 [bps]
84.88.40.26:53025 10.1.24.2:53025 2800  16449591 [bytes]        21973
[packets] 136517.393398608 [bps]
```

From the output we can draw several conclusions:

- The last three flows have the highest throughput and they all use UDP port 53025. So, this is the port used for sending video data.
- The data coming from the flowserver to the WiMAX client (highlighted in blue) has the SSRCs 2798, 2799, 2800. So, these must be the values used by the other three clients.
- The list of flows we will use as argument for the 3rd form is therefore: 53025:2798, 53025:2799, 53025:2800.

```
./rares.pl -r 53025:2798,53025:2799,53025:2800 fs.pcap wimax.pcap > results
```

The results file will contain OWD and IPDV values for all the packets from the three video flows. The format of this file is:

```
Sequence_Number Source_Timestamp Dest_Timestamp OWD IPDV Packet_Length
```

2. The MATLAB Scripts

Synopsis

There are two **.m** files that we developed for interpreting / depicting the results: **plot.m** and **len.m**. The main routine is in **plot.m** and the functions for evaluating the stability of metrics in respect to packet size are found in **len.m**.

For purposes that deal with processing automation, a file named **list.txt** must be placed in the directory containing the result files. This file must include all the filenames that are to be processed, one per line. For example:

```
cat list.txt
e2e.eth.wifi.audio.txt
e2e.eth.wifi.video.txt
e2e.eth.wimax.audio.txt
e2e.eth.wimax.video.txt
```

Various parameters are set in the beginning of the source code of **plot.m**:

```
cd('D:/Master/_results/dataset2-hop');
```
This is the directory were the results files and the index file **list.txt** are located.

```
ipdv_histogram_precision = 0.025;
```
This is used for setting the bin width of the IPDV histogram. The value is in milliseconds.

```
owd_histogram_nuber_of_bins = 128;
```
This is used for setting the number of bins of the OWD histogram.

```
percentile_limits = [2.5 97.5];
```
This is used for automatically zooming the histogram / CDF plots to the values specified by the two percentile limits.

Notes

- The result files that these script process must be produced with the Perl script *without* using the **[-s]** (statistics) option.
- If MATLAB complains about the script being in an unknown location, click the **"Add to Path"** button.
- In the **Command Window** you can see the statistics for OWD and IPDV for each file being processed.
- To process / plot the next result file press **Space**.