

# Table of Contents

Introduction.....	1
Tools Classification.....	1
Active Measurement.....	1
Passive Measurement.....	1
Objectives.....	2
Experiment Planning.....	2
Software Used.....	2
Selection of Endpoints.....	2
Limitation of Endpoints.....	3
Time Synchronization.....	3
Test Packet Size.....	5
Data Sampling Frequency.....	5
Security.....	5
Our Work.....	6
(A) Understanding How OWAMP Works.....	6
Introduction.....	6
OWAMP Protocol Basics.....	6
Test Endpoints.....	7
Effect of Different Time Sources.....	7
Packet Captures.....	8
Result and Discussion.....	10
Dissecting the Traffic: Complete Test Session Packet Flow.....	10
Connection & Test Setup (OWAMP-Control).....	10
Test Packet Stream (OWAMP-Test).....	14
Post-test (OWAMP-Control).....	15
Same Time Source.....	17
Additional Factors Contribute to the Error of Measured Delay.....	19
PING.....	19
(B) Internet Delays.....	20
Introduction.....	20
Test Endpoints.....	21
Measurement Coverage.....	21
Methods.....	21
Parameters.....	22
Data Sampling.....	22
Result and Discussion.....	23
OWPING Result.....	23
PING Result.....	28
Delay Asymmetry: Forward v.s. Backward Delay.....	31
Conclusion.....	33
References.....	34

# Introduction

Internet, with all its idiosyncrasies, appears to be doing its job rather well. Although it works, it is far from being ideal. To continue its development, internet measurement tools are employed to perform measurement and analysis. Internet measurement is a key to the design of the next-generation Internet and it takes place in various aspects:

- Help us to better understand why it works;
- Help us to know what is happening to the internet traffic;
- Help us to diagnose network problems or connectivity and provide cue to find solutions.

## Tools Classification

Different tools perform various kinds of measurement and return different related information. We must recognize the tools in order to perform measurement and analysis effectively. Throughout understanding help us to determine the usage of different tools and diagnose the problems in a sufficient way.

In this project, we have used OWAMP, PING, TraceRoute and Wireshark as our measuring tools. They can be classified as the following:

- Active Measurement
- Passive Measurement

### Active Measurement

Methods that involve adding traffic to the network for the purposes of measurement are called Active Measurement. For example, PING and OWAMP used in this project are active measurement tools.

PING: Sends ICMP ECHO\_REQUEST and captures ECHO\_REPLY

- Useful for measuring round trip times (RTTs)
- Only sender needs to be under experiment control

OWAMP: A daemon running on the target which listens for and records probe packets sent by the sender

- Useful for measuring one-way delay
- Requires both sender and receiver to be under experiment control
- Requires synchronized clocks or a method to remove clock offset

### Passive Measurement

Methods that capture traffic generated by other users and applications, without injecting any traffic of its own onto the network are called Passive Measurement. In this project, Wireshark served as our passive measurement tool.

Wireshark: A network packet analyzer try to capture network packets and display the packet data as detailed as possible.

- Capture live packet data from a network interface
- Display packets with very detailed protocol information

# Objectives

In this project, we are pointing to these major directions as our major objectives. Using One-Way Active Measurement Protocol (OWAMP) [1.] suite together with PING, TraceRoute and Wireshark,

- to profile one-way network latency (by measure one way path latency);
- understanding how does OWAMP perform measurement (by doing packet captures);
- to compare the results using different measurement methods.

## Experiment Planning

### Software Used

In our study, we adopted OWAMP suite version 3.2RC1 (available at <http://software.internet2.edu/sources/owamp/owamp-3.2rc1.tar.gz>). It was then compiled on both Windows XP (SP3) with Cygwin 1.5.25-15 (available at <http://www.cygwin.com/>); and Linux (Linux linux06 2.6.22.19-0.1-default #1 SMP 2008-10-14 22:17:43 +0200 i686 i686 i386 GNU/Linux) with gcc version 4.2.1 (SUSE Linux) platforms.

Time synchronizations were done on Windows platform with Dimension4 (v4.5 available at <http://www.thinkman.com/dimension4/download.htm>) against Stratum1 and Stratum2 NTP time servers.

Wireshark v1.2.4 (<http://www.wireshark.org/download.html>) was used to perform packet captures on Windows XP (SP3).

Custom developed Perl scripts for data collection automation and parsing of results were included on project CD.

### Selection of Endpoints

Recalling our ultimate objective, we planned to execute a persistent measurement under a physically stable environment. We were most preferred a stable server environment to base our tests on so that they could be executed persistently with minimal external turbulences which otherwise would introduce uncertainties.

**We, however, determined that none of our group members were able to provide/host such an ideal environment.** We then looked into departmental machines and finally reached Linux06 (linux06.comp.polyu.edu.hk) that was equipped with NTP (the primary requirement). As a bonus, it situated in a temperature controlled environment, and its CPU-load gave no sign of stress during our evaluation period. These attributes matched quite well to the requirements as stated in OWAMP cookbook. [2.]

On the other hand, we picked two open OWAMP servers to assume the Server role from perfSONAR Global Service page:

<http://dc211.internet2.edu/cgi-bin/perfSONAR/serviceList.cgi#OWAMP>

All in all, we ended up the topology of hosts involved in our measurement as depicted in Fig 1.

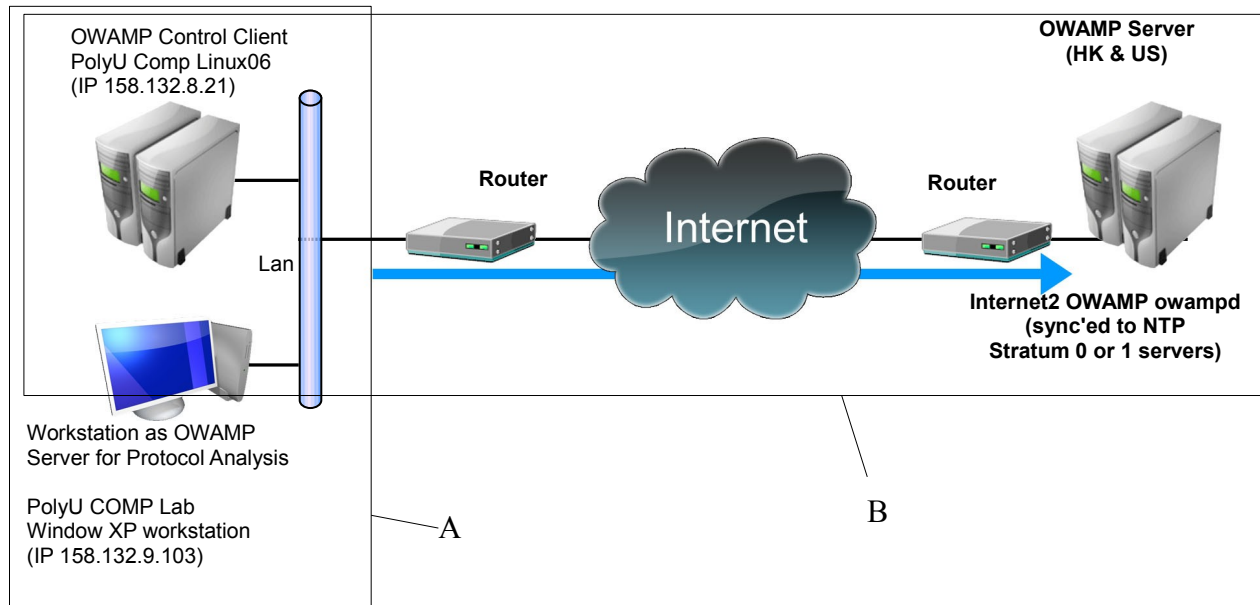


Fig 1: Overall topology for hosts that were involved in our studies in two major parts: (A) Understanding How OWAMP Works, p.6; (B) Internet Delays Measurement, p.20

## Limitation of Endpoints

The use of Linux06 certainly came with limitations due mainly to our non privileged access level to this host, resulting:

1. Permission denied when attempted to listen to socket and campus firewall blocked incoming connections. We could not assume Session-Receiver & Server role. Equally, when executing OWPING (playing the role as test client) against remote servers, only forward one-way delay measurement could be done at any instance (i.e. cannot accept connection from remote server, hence no concurrent backward one-way latency measurement possible) (hence the  $-t$  argument when invoking OWPING).
2. We later found that even Linux06 had NTP installed, it actually synchronized to a Stratum 2 (but not Stratum 0 or Stratum 1 time source as suggested by OWAMP cookbook). Even then, we could not change NTP sync peers without privileged access.
3. As a result of 2., we could not perform cross-time-synchronization between two endpoints
4. Measurement tasks got randomly terminated (!) by server administrator
5. Could not install packet capture software to inspect the test traffic directly (need additional rounds of experiment, hence the upcoming section)

Open OWAMP servers certainly comes with limitations as well:

1. We had no control over the host, as a result, we could not assert the NTP status of the host

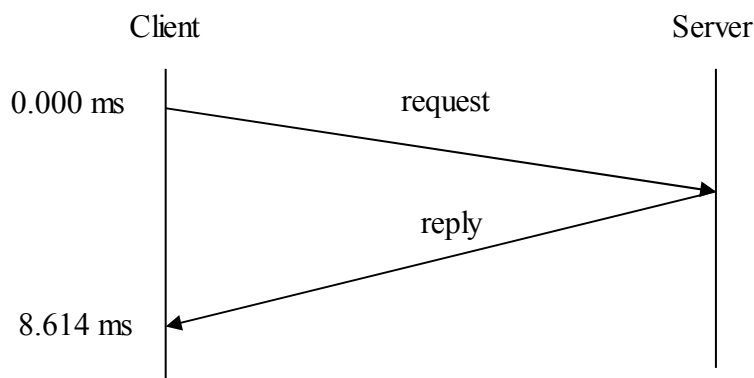
These factors actually impacted our experimental design to a substantial degree.

## Time Synchronization

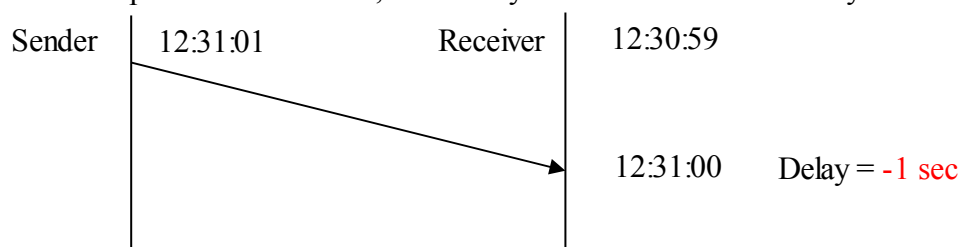
One of the pre-requisites for being an open OWAMP server is that it needs to be synchronized to

more than 3 Stratum 0 or Stratum 1 NTP servers. In our upcoming measurements that involve open OWAMP servers, we assumed those open OWAMP servers were synchronized with Stratum 1 NTP servers, which is the pre-requisite of being an “open” server (2.).

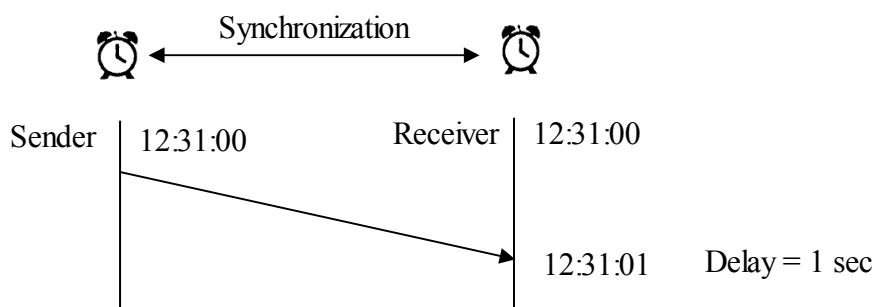
With traditional PING, the delay measurement is based solely on the sending host:



*Drawing 1: The path latency is totally determined by the client endpoint in traditional PING*  
The path latency (Drawing 1) is determined only by the client from the PING start time and the relative response time. Hence, no time synchronization is necessary.



*Drawing 2: The one-way latency without a synchronized clock. If time are not synchronized between to endpoints, the measured delay would not be contributed solely by network latency but may include the time difference between two clocks as well.*



*Drawing 3: Time is synchronized between two endpoints.  
The Receiver reports one-way latency correctly.*

On the other hand, OWAMP (RFC4656) protocol requires computer on both ends participating the measurement to be time synchronized. This is necessary because the Session-Receiver pre-computes the test-packet delivery schedule independent of Session-Sender. Without synchronized time, the interpretation of test-packets' time-stamp would be *different* between two endpoints (Drawing 2). The measurement reported from the Session-Receiver would make no sense to Session-Sender as a result.

With synchronization (Drawing 3), however, both endpoints are referring to the same clock. The Session-Receiver would be able to time the delivery of packets with respect to Session-Sender's packet firing time. Session-Receiver receives the pre-sent delivery schedule from Session-Sender

to compute the delivery time for every upcoming test packets. These *independently computed* packet firing schedule would ideally be the actual packet sent-out time from Sender.

## Test Packet Size

Launching OWPING (Fig 7) with parameter **-s 200** was deliberate: quoting RFC4656 [1., p5]:

```
...because many Internet paths include segments that transport
IP over ATM, delay and loss measurements can include the effects of
ATM segmentation and reassembly (SAR). Consequently, OWAMP has been
designed to allow for small test packets that would fit inside the
payload of a single ATM cell (this is only achieved in
unauthenticated mode).
```

Our plan, however, focused on observing the behavioral difference between OWPING and PING. We thus tried to make packet from OWPING and PING to be similar so that the effect of packet size in the comparison would be negligible. Thus an arbitrary 200-byte padding was introduced to both OWPING and PING test sessions.

## Data Sampling Frequency

After preliminary OWPING exercises, we found that the voluminous amount of data generated by OWPING was difficult to handle in terms of disk space and transfer time. To contain the size of data set, we limited our test packets sending rate to 1 test packet per second in PING and OWPING. Hence the **-i 1** argument.

This, however, may coincide with system events that also happens once per second. And that the periodic system events concerned might have impact on the punctuality of packet delivery. So a test running in-sync with this event might produce a highly skewed result. Even then, due to our limited resources and the fact that the tests were carried out on a shared machine, we determined that this gave a right balance between data quality and data quantity.

## Security

A significant portion of the OWAMP protocol definition focus on illustrating its temper-proof measures that include integrity check and secure communication between test peers. Since security is outside the scope of our study, here we only focus on the messages structure (relevant to the delay measurement) and their relative sequence with respect to the OWAMP specifications.

# Our Work

We report our work in two major parts:

1. OWAMP protocol analysis ((A) Understanding How OWAMP Works, p.6); and,
2. Comparison of Internet delays by using OWPING and PING ((B) Internet Delays, p.20).

## (A) Understanding How OWAMP Works

### Introduction

In this section, we correlate the OWAMP messages with OWAMP specifications (RFC4656) by actually capturing packets involved in OWAMP communications. This protocol study experiment was actually part of the feasibility studies of server platform on which the upcoming measurements would be carried out. Note that the parameters (*except destination host and packet delivery interval*) adopted by this test session were identical to the actual persistent measurement in the next part (see p.20, (B) Internet Delays).

### OWAMP Protocol Basics

As stated in RFC4656, in an one-way PING measurement, when a Client executing OWPING sends a packet to Server running owampd, the former does have no knowledge of when the packet will be delivered. Hence, in order to tell this one-way path latency, the Server should somehow report the Client of the delays.

To achieve this, the Client (as Control-Client) who produces a packet delivery schedule(s) sends it to Server in advance of the test session. By using this schedule, Server pre-compute the *expected arrival time* for every packets.

When the OWAMP-test begins, Client (as Session-Sender) starts sending out packets according to the prescribed schedule. Each packet is tagged with a sequence number with respect to the Test Session. The Session-Receiver, upon receiving these packets, compares their arrival time against the *expected arrival time*. Session-Receiver in turn populates its log, detailing the arrival time, if at all, arrival order and the time offset of the respective packets with respect to the prescribed schedule.

When the test finishes, Client (as Fetch-Client) obtains the one-way delay from Server.

OWAMP protocol can be resolved to OWAMP-Control and OWAMP-Test:

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

OWAMP-Test protocol is layered over UDP and is used to send singleton measurement packets along the Internet path under test.

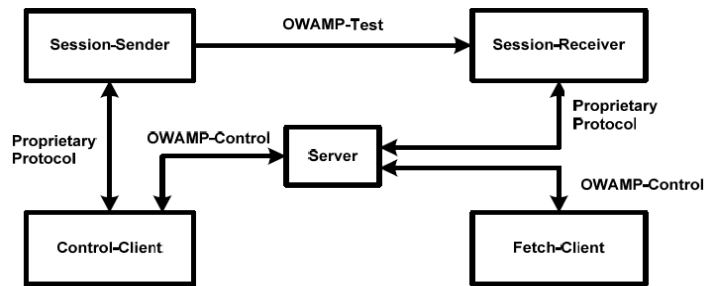


Figure 1 – OWAMP architecture.

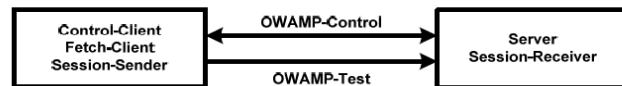


Figure 2 – OWAMP simplified architecture.

Fig 2: OWAMP architecture

## Test Endpoints

To maximize packet capturing experiences, we performed our packet capture on Windows platform *despite OWAMP is not supported on Windows officially*. One major limitations of Linux emulation software, Cygwin, on the Windows built version of OWPING was that it was unable to issue \*nix specific NTP related system calls for doing time synchronizations, which is critical to the accuracy of data [1., p.34]. Still, we found it acceptable for the purpose of understanding the traffic pattern and packets' content while OWPING was executing. Nevertheless, it enlightened us the necessity of accurate time source with respect to one-way delay measurements.

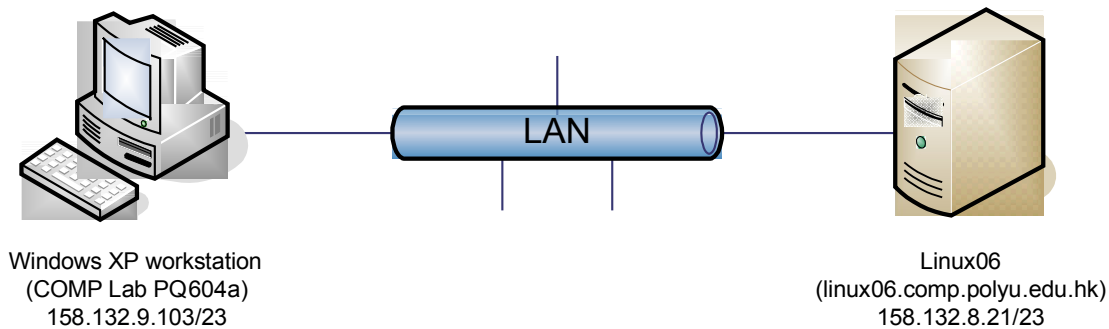


Fig 3: Machines involved for OWAMP packet capture exercise

The packet capturing was done in Computer Science Laboratory (PQ604a), Polytechnic University campus. WireShark (1.4.2) was installed on workstation 158.132.9.103 assuming Server & Session-Receiver OWAMP roles [1., p.5]; whereas Linux06 assumed Control-Client & Fetch-Client & Session-Sender OWAMP roles. The workstation firewall was shut off to ease the connection setup. Another lab PC workstation was used to SSH to Linux06 to launch OWPING test against the OWAMP server (Fig 3).

## Effect of Different Time Sources

Since in no way we could change the NTP settings on Linux06, we realized the effect of time source to the measurement result by synchronizing the workstation to different time servers.

Therefore, we performed two rounds of experiments and the system time of workstation was synchronized to:

1. stdtime.gov.hk (Stratum 1) (the following experiment)
2. Linux06's NTP setting – 203.129.68.14 (Stratum 2) (Fig 4)
3. Cross synchronization between workstation and linux06 was *impossible* because the system policy on linux06 prohibited us to do so

```
(08586615g) ~> ntptrace
localhost: stratum 3, offset -0.031652, synch distance 0.335469
203.129.68.14: stratum 2, offset 0.039070, synch distance 0.191029
192.168.51.202: timed out, nothing received
***Request timed out
(08586615g) ~>
```

Fig 4: Inspecting the NTP time source on Linux06 via ntptrace



# Packet Captures

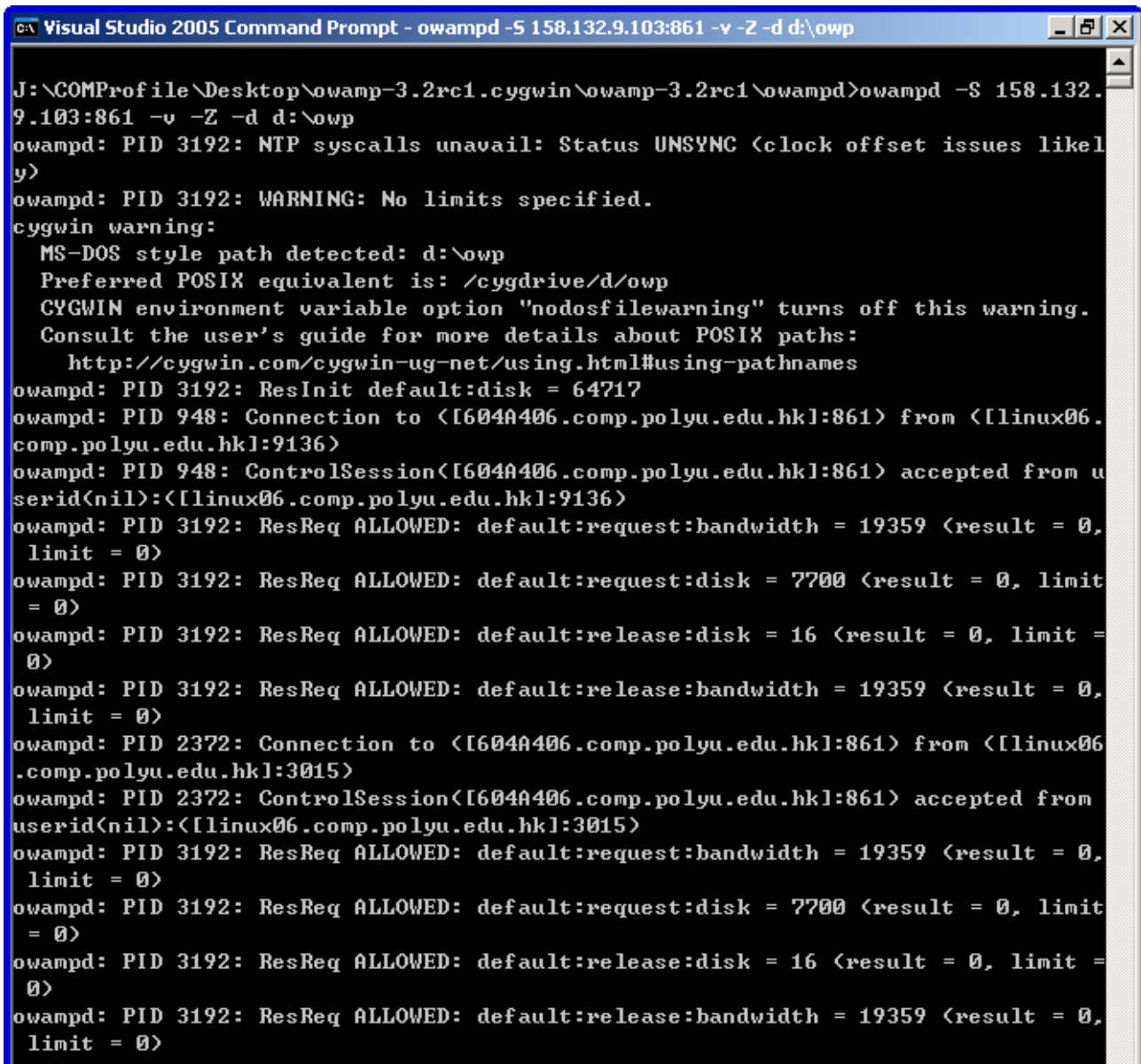
The workstation, playing the role as Server (Server & Session-Receiver) was launched as shown in Fig 5 with these parameters:

```
owampd -S 158.132.9.103:861 -v -Z -d d:\owp
```

Linux06, playing the role as Client (Control-Client & Fetch-Client & Session-Sender) was launched as shown in Fig 7 with these parameters:

```
owping -t -c 300 -s 200 158.132.9.103
```

Meanwhile, packet capturing software WireShark was also passively capturing this traffic (Fig 6).



```
C:\ Visual Studio 2005 Command Prompt - owampd -S 158.132.9.103:861 -v -Z -d d:\owp

J:\COMProfile\Desktop\owamp-3.2rc1.cygwin\owamp-3.2rc1\owampd>owampd -S 158.132.9.103:861 -v -Z -d d:\owp
owampd: PID 3192: NTP syscalls unavail: Status UNSYNC <clock offset issues likely>
owampd: PID 3192: WARNING: No limits specified.
cygwin warning:
  MS-DOS style path detected: d:\owp
  Preferred POSIX equivalent is: /cygdrive/d/owp
  CYGWIN environment variable option "nodosfilewarning" turns off this warning.
  Consult the user's guide for more details about POSIX paths:
    http://cygwin.com/cygwin-ug-net/using.html#using-pathnames
owampd: PID 3192: ResInit default:disk = 64717
owampd: PID 948: Connection to <[604A406.comp.polyu.edu.hk]:861> from <[linux06.comp.polyu.edu.hk]:9136>
owampd: PID 948: ControlSession<[604A406.comp.polyu.edu.hk]:861> accepted from user id<nil>:<[linux06.comp.polyu.edu.hk]:9136>
owampd: PID 3192: ResReq ALLOWED: default:request:bandwidth = 19359 <result = 0, limit = 0>
owampd: PID 3192: ResReq ALLOWED: default:request:disk = 7700 <result = 0, limit = 0>
owampd: PID 3192: ResReq ALLOWED: default:release:disk = 16 <result = 0, limit = 0>
owampd: PID 3192: ResReq ALLOWED: default:release:bandwidth = 19359 <result = 0, limit = 0>
owampd: PID 2372: Connection to <[604A406.comp.polyu.edu.hk]:861> from <[linux06.comp.polyu.edu.hk]:3015>
owampd: PID 2372: ControlSession<[604A406.comp.polyu.edu.hk]:861> accepted from user id<nil>:<[linux06.comp.polyu.edu.hk]:3015>
owampd: PID 3192: ResReq ALLOWED: default:request:bandwidth = 19359 <result = 0, limit = 0>
owampd: PID 3192: ResReq ALLOWED: default:request:disk = 7700 <result = 0, limit = 0>
owampd: PID 3192: ResReq ALLOWED: default:release:disk = 16 <result = 0, limit = 0>
owampd: PID 3192: ResReq ALLOWED: default:release:bandwidth = 19359 <result = 0, limit = 0>
```

Fig 5: The server (workstation) and its debugging traces. Note the UNSYNC warning indicating the unavailability of NTP syscalls on Windows Cygwin.

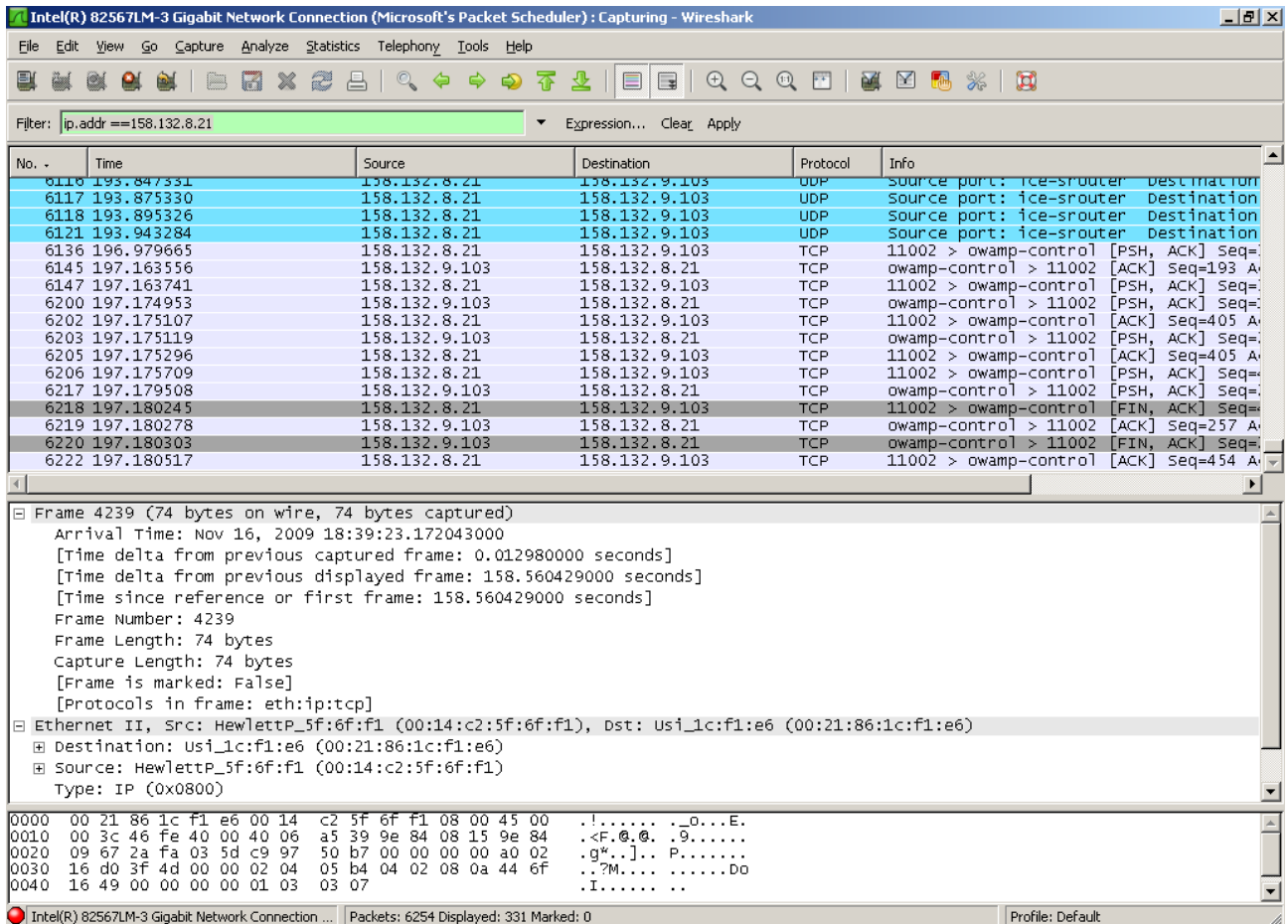


Fig 6: Wireshark running on workstation hooked on the LAN interface to capture OWPING packets. IP filter was applied to display only packets to-from test server and client.

```
[08586615g@linux06] /home/student6/g/08586615g/comp5311/owamp-3.2rc1/owping:> owping -t -c 300 -s 200 158.132.9.103
Approximately 32.9 seconds until results available

--- owping statistics from [linux06]:4076 to [158.132.9.103]:1985 ---
SID: 9e840967ceabae488e64c2f8156137da
first: 2009-11-16T18:56:10.080
last: 2009-11-16T18:56:38.759
300 sent, 0 lost (0.000%), 0 duplicates
one-way delay min/median/max = -577/-576/-575 ms, (unsync)
one-way jitter = 0.7 ms (P95-P50)
TTL not reported
no reordering
```

Fig 7: In a separate run, OWPING gave negative delay in the absence of synchronization.

## Result and Discussion

Here we identify the packet contents from the TCP session one by one (Fig 6) against the OWAMP specification. We also attempt to account for the *negative* time difference from the result (Fig 7) in terms of NTP time synchronization effects on the system clock.

### Dissecting the Traffic: Complete Test Session Packet Flow

According to RFC4656 [1., p.6], OWAMP consists of two inter-related protocols: OWAMP-Control (TCP/IP portion) and OWAMP-Test (UDP/IP portion) that were clearly indicated in the flow (Fig 8,9,10).

#### Connection & Test Setup (OWAMP-Control)

Time	158.132.8.21	158.132.9.103	Comment
0.000	(3015) → (861) SYN		Seq = 0 Ack = 342858246
0.000	(3015) ← (861) SYN,ACK		Seq = 0 Ack = 1
0.000	(3015) → (861) ACK		Seq = 1 Ack = 1
0.000	(3015) ← (861) ACK		Seq = 1 Ack = 1
0.141	(3015) → (861) PSH,ACK - Len: 64		Seq = 1 Ack = 1
0.142	(3015) ← (861) ACK		Seq = 1 Ack = 65
0.142	(3015) → (861) PSH,ACK - Len: 164		Seq = 1 Ack = 65
0.142	(3015) ← (861) PSH,ACK - Len: 32		Seq = 65 Ack = 165
0.179	(3015) → (861) ACK		Seq = 165 Ack = 97
0.179	(3015) → (861) PSH,ACK - Len: 16		Seq = 97 Ack = 165
0.180	(3015) → (861) ACK		Seq = 165 Ack = 113
0.180	(3015) → (861) PSH,ACK - Len: 112		Seq = 165 Ack = 113
0.378	(3015) ← (861) ACK		Seq = 113 Ack = 277
0.378	(3015) → (861) PSH,ACK - Len: 32		Seq = 277 Ack = 113
0.400	(3015) → (861) PSH,ACK - Len: 48		Seq = 113 Ack = 309
0.400	(3015) → (861) PSH,ACK - Len: 32		Seq = 309 Ack = 161
0.400	(3015) → (861) PSH,ACK - Len: 32		Seq = 161 Ack = 341
0.439	(3015) → (861) ACK		Seq = 341 Ack = 193

Fig 8: OWAMP-Control flow (handshaking)

Time	158.132.8.21	158.132.9.103	Comment
1.420	(4107) → (2077) Source port: j-ac		UDP: Source port: j-ac Destination port: tsrmagt
1.564	(4107) → (2077) Source port: j-ac		UDP: Source port: j-ac Destination port: tsrmagt
1.568	(4107) → (2077) Source port: j-ac		UDP: Source port: j-ac Destination port: tsrmagt
.			
.			
.			
29.969	(4107) → (2077) Source port: j-ac		UDP: Source port: j-ac Destination port: tsrmagt
30.065	(4107) → (2077) Source port: j-ac		UDP: Source port: j-ac Destination port: tsrmagt
30.069	(4107) → (2077) Source port: j-ac		UDP: Source port: j-ac Destination port: tsrmagt

Fig 10: OWAMP-Test flow (UDP)

The Data portion of the OWAMP-control packets concerned were translated from binary to their ASCII representation. Each 8-bit of binary inputs were grouped (bit 0 → 0; bit 1 → 1), producing the formatted result (Fig 11).

Time	158.132.8.21	158.132.9.103	Comment
32.175	(3015) → (861) PSH,ACK - Len: 16		Seq = 193 Ack = 341
32.175	(3015) → (861) ACK		Seq = 341 Ack = 209
32.175	(3015) → (861) PSH,ACK - Len: 16		Seq = 209 Ack = 341
32.175	(3015) → (861) ACK		Seq = 341 Ack = 225
32.175	(3015) → (861) PSH,ACK - Len: 16		Seq = 341 Ack = 225
32.363	(3015) → (861) ACK		Seq = 225 Ack = 357
32.363	(3015) → (861) PSH,ACK - Len: 96		Seq = 357 Ack = 225
32.447	(3015) → (861) PSH,ACK - Len: 32		Seq = 225 Ack = 453
32.447	(3015) → (861) PSH,ACK - Len: 1448		Seq = 257 Ack = 453
32.447	(3015) → (861) PSH,ACK - Len: 312		Seq = 1705 Ack = 453
32.447	(3015) → (861) PSH,ACK - Len: 1448		Seq = 2017 Ack = 453
32.447	(3015) → (861) PSH,ACK - Len: 152		Seq = 3465 Ack = 453
32.447	(3015) → (861) PSH,ACK - Len: 1448		Seq = 3617 Ack = 453
32.447	(3015) → (861) PSH,ACK - Len: 152		Seq = 5065 Ack = 453
32.447	(3015) → (861) ACK		Seq = 453 Ack = 1705
32.447	(3015) → (861) PSH,ACK - Len: 1448		Seq = 5217 Ack = 453
32.447	(3015) → (861) PSH,ACK - Len: 1272		Seq = 6665 Ack = 453
32.447	(3015) → (861) ACK		Seq = 453 Ack = 3465
32.447	(3015) → (861) ACK		Seq = 453 Ack = 5065
32.447	(3015) → (861) ACK		Seq = 453 Ack = 6665
32.448	(3015) → (861) FIN,ACK		Seq = 453 Ack = 7937
32.448	(3015) → (861) ACK		Seq = 7937 Ack = 454
32.448	(3015) → (861) FIN,ACK		Seq = 7937 Ack = 454
32.448	(3015) → (861) ACK		Seq = 454 Ack = 7938

Fig 9: OWAMP-Control flow (connection termination)



The client then sent out Request-Session message (Fig 12):

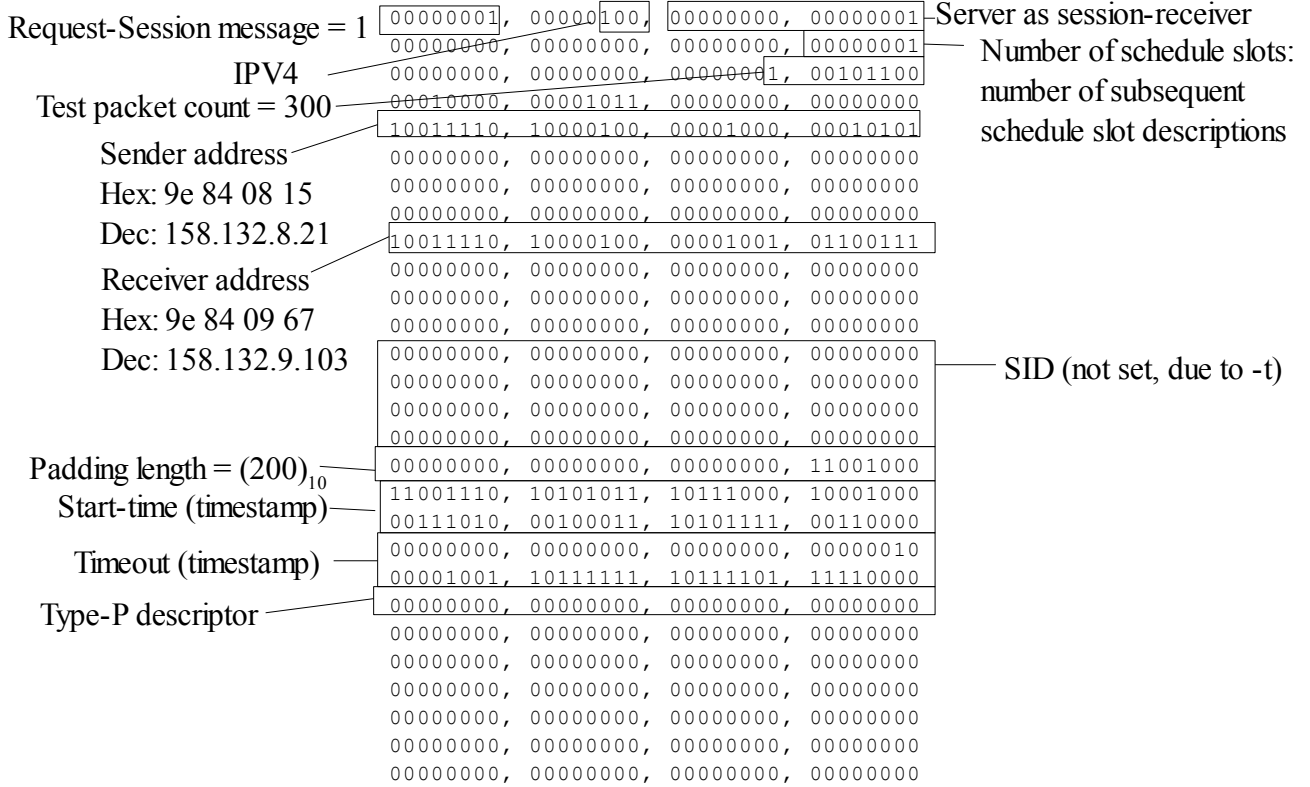


Fig 12: Seq#12, Request-Session message 112 octets sent from client

After this message, and because the *Number of schedule slots* equal to one (Fig 12), it was then followed by a single Schedule Slot Descriptions message (Fig 13):

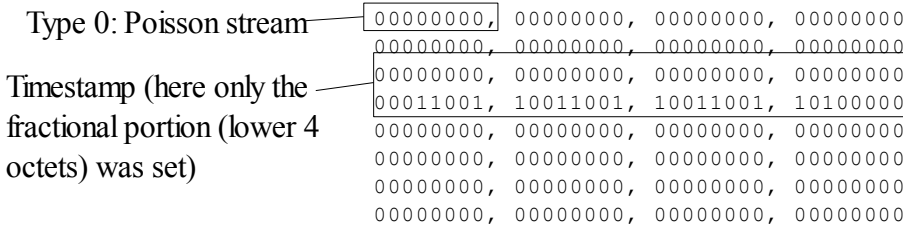


Fig 13: Seq#14, Schedule Slot Descriptions

As we launched OWPING in its default Poisson mode. In this mode, the packet delivery schedule was represented as a timestamp indicating the mean of the exponentially distributed pseudo-random quantity  $m$ . With this value, the Session-Receiver would expand it to a full schedule by using:

$$m = 1/\lambda$$

(where the distribution density function is expressed as  $\lambda \cdot \exp(-\lambda \cdot x)$ )

$x$  varies with the sequence of the test packets. For this experiment using  $(-c \ 300)$  parameter,  $x$  would fall in  $(1..300)$ . The output of this function thus gave the Session-Receiver a clue when a packet with sequence number  $x$  would be sent out by the Session-Sender. Thus the Session-Receiver would be able to tell how long the packet was in transit *iff* the Session-Sender sends out test packets according to the schedule.

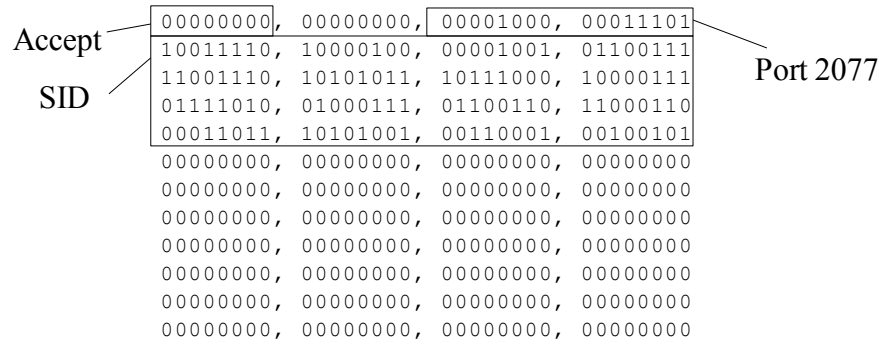


Fig 14: Seq#15, Accept-Session by server

In response to this Request-Session (logical) message (Fig 12 & Fig 13), the Server replied with Accept-Session message (Fig 14).

After all tests and its respective schedules were transferred, the client initiates the test by sending out Start-Session message (Fig 15):

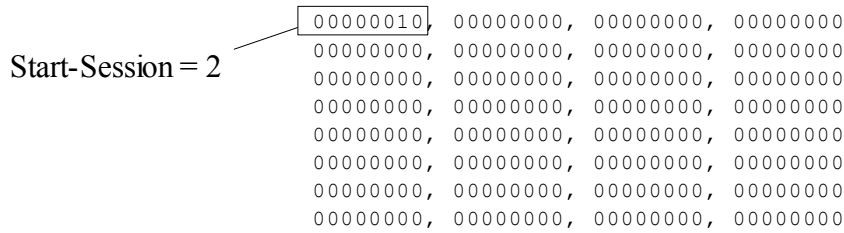


Fig 15: Seq#16, Start-Session by client

The server responded to this Start-Session request by sending out Start-Ack message (Fig 16):

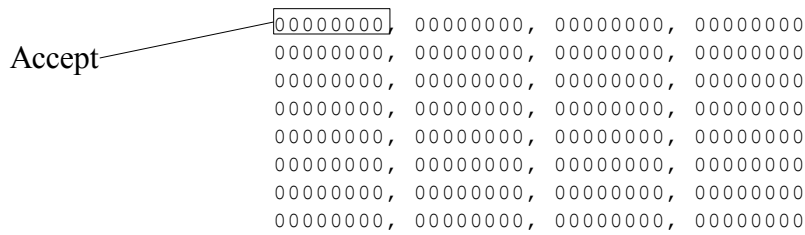
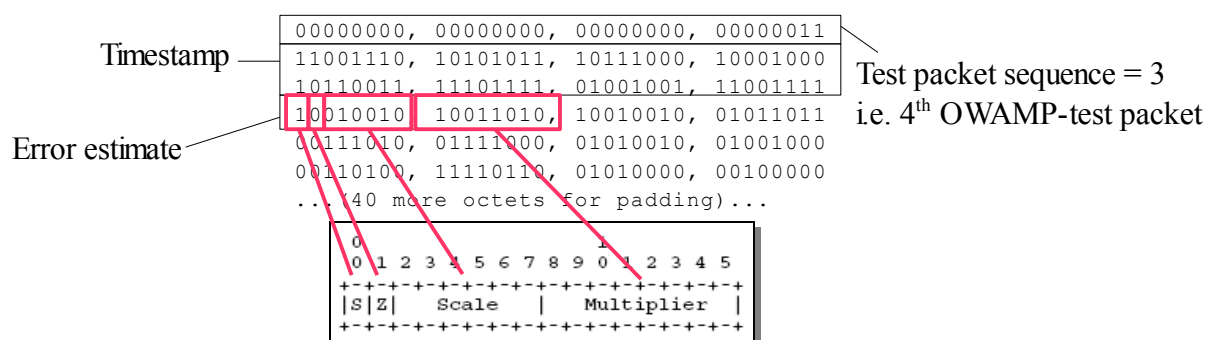


Fig 16: Seq#17, Start-Ack by server

Upon receiving the Start-Ack response, the client replied with a pure ACK, and began sending out test packet stream (Fig 17), note the ~1 second delay when the client received Start-Ack and started sending test packets.

[illegible]

### Test Packet Stream (OWAMP-Test)





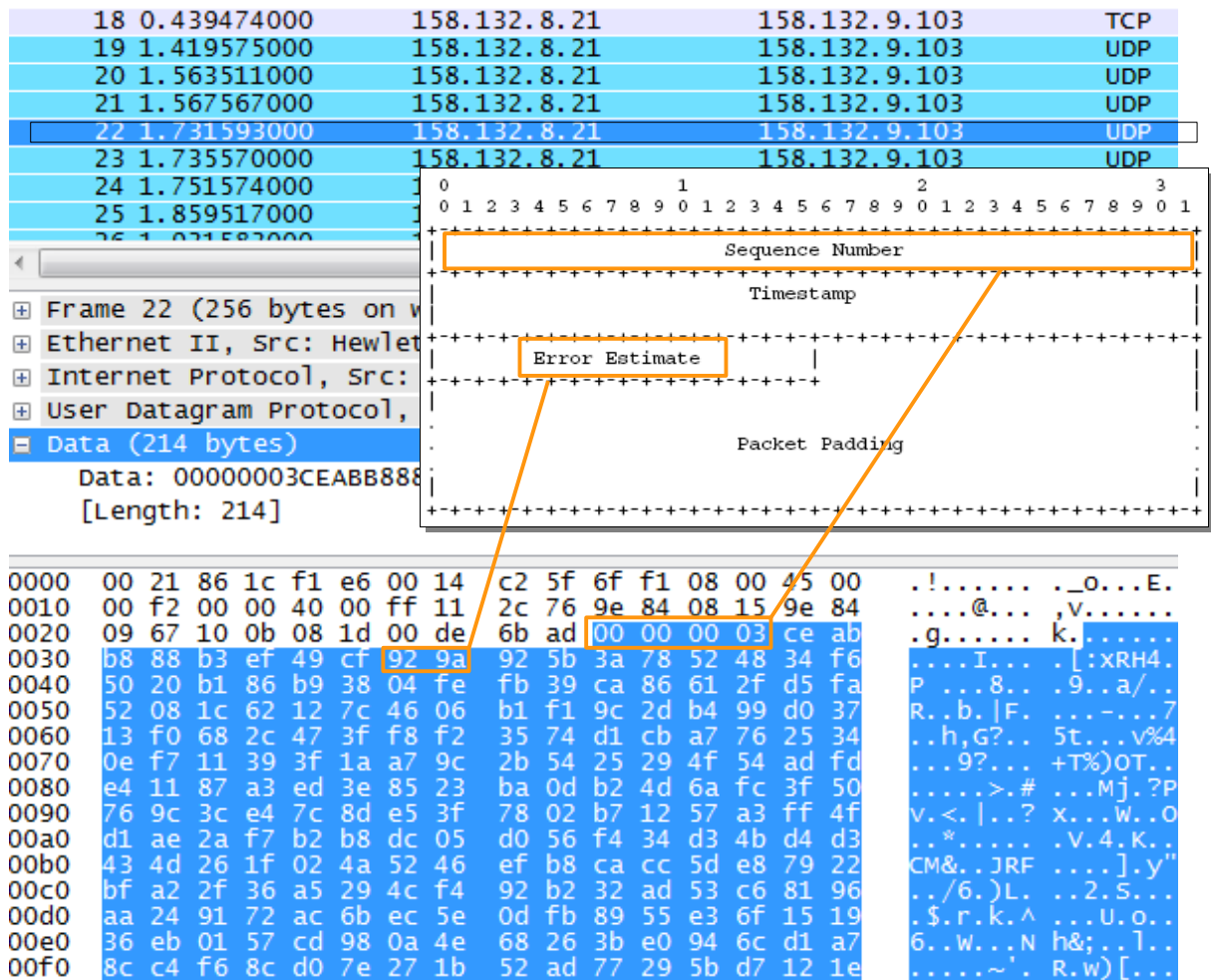


Fig 19: OWAMP-test 4<sup>th</sup> packet at seq#22

As an example of OWAMP-Test packet (Fig 19), the Error Estimate field (Fig 18) did have its most significant bit (left most) set indicating that the client (Linux06) had synchronized to an external time source.

Sequence numbers start with zero and are incremented by one for each subsequent packet.

The Error Estimate specifies the error estimate of a given test packet in that the deviation of its actual send out time against the corresponding delivery schedule.

### Post-test (OWAMP-Control)

The measurement were ended when either side sent out Stop-Sessions message.

In our experiment, Server sent out the first Stop-Sessions message after receiving the 300<sup>th</sup> OWAMP-test packet (Fig 20):

Stop-Session = 3      00000011, 00000000, 00000000, 00000000  
 00000000, 00000000, 00000000, 00000000      Number of sessions = 0  
 00000000, 00000000, 00000000, 00000000  
 00000000, 00000000, 00000000, 00000000

Fig 20: Seq#319, Workstation (Server) sends Stop-Session message to Linux06 (Client)

In our experiment, since Linux06 assumed only the Session-Sender role (and not Session-Receiver), the Number of session is thus zero as reported by Server.

On the other hand, Linux06 was a Session-Sender, it then sent out its Stop-Sessions message at



seq#323 (Fig 21):

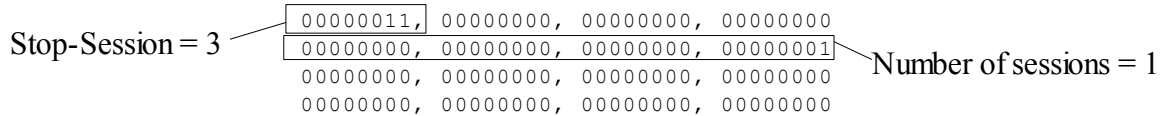


Fig 21: Seq#323, Linux06 (Client) sends Stop-Session message to Workstation (Server)

With -t parameter while invoking OWPING, it instructed OWPING to be Control-Client and Session-Sender and Fetch Client only, but not being Server and Session-Receiver at the same time.

In this run, no packets were skipped, so there were no Skip Range messages.

We found that this version of OWPING might not be fully RFC4656 compliant. Unlike what specifies in RFC4656, no messages with Message Type = 4 at the 1<sup>st</sup> octet were found.

The closest packet signature resembling Fetch-Session occurred in packet seq#325, where the Message Type = 4 appeared at the 49<sup>th</sup> octet of Data (Fig 22).

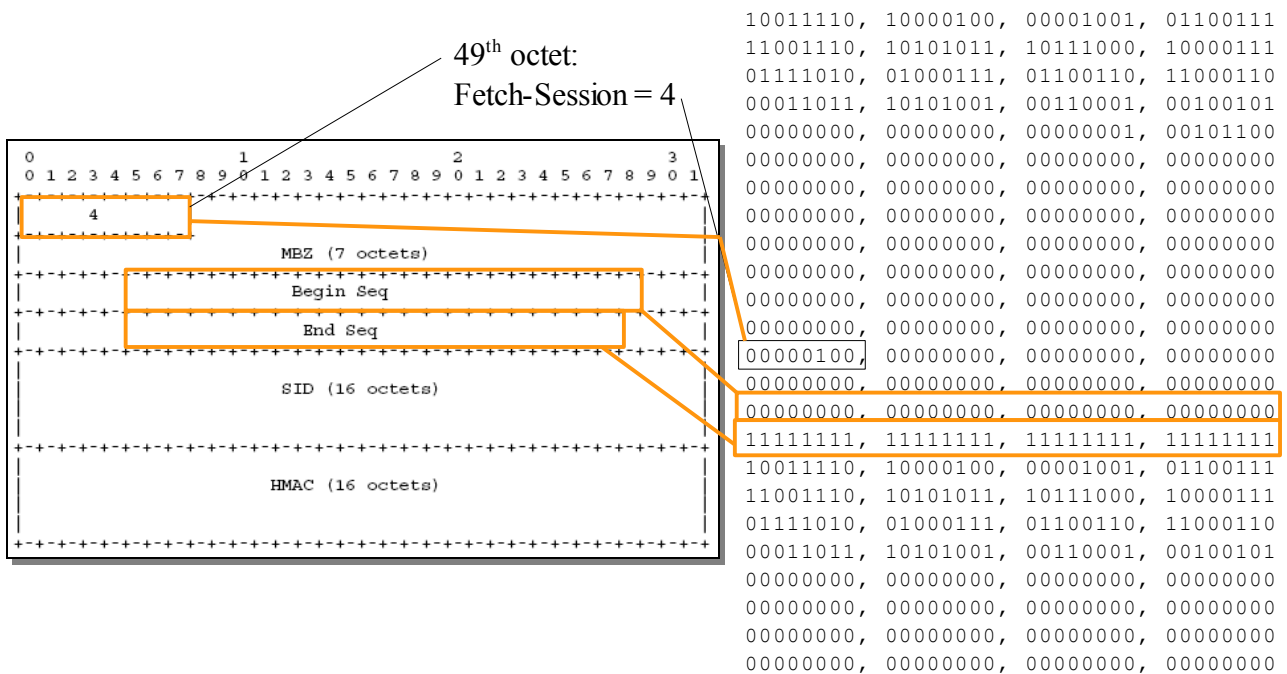


Fig 22: Seq#325, Linux06 (Control-Client) piggy backed Fetch-Session message to Workstation (Server). Request range 0x0000 to 0xFFFF.

In seq#326, the workstation (as Server) responded as Stop-Sessions message by using Fetch-Ack message:

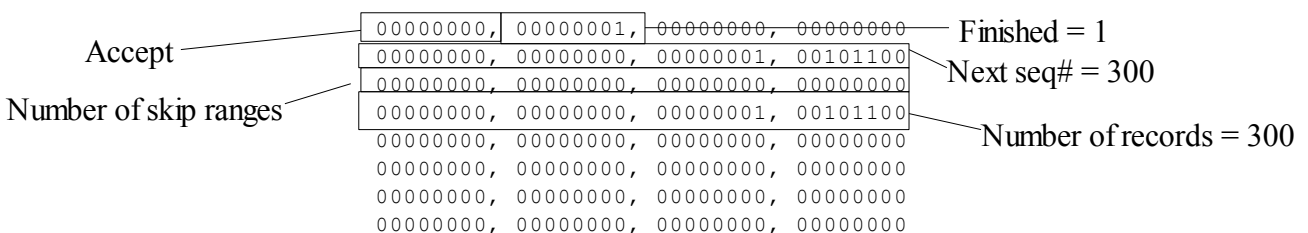


Fig 23: Seq#326, Workstation (Server) sends Fetch-Ack message (32 octets) without being asked by Linux06 (Fetch-Client)

Without any ACK from the client, the workstation (as Server) sent out 6 more packets (seq#327 to 332), which were the concatenated form of OWAMP-Test session data [1., p.26]:

No. .	Time	Source	Destination	Protocol	Info
320	32.175088000	158.132.8.21	158.132.9.103	TCP	Seq#327-332 & 334-335: OWAMP-Test Session Data win=5888 Len=0
321	32.175098000	158.132.9.103	158.132.8.21	TCP	=341 win=65195
322	32.175275000	158.132.8.21	158.132.9.103	TCP	win=5888 Len=0
(Total data length = 7680)					
Seq#326: Fetch-Ack					
326	32.446602000	158.132.9.103	158.132.8.21	TCP	Seq=225 Ack=453 win=65083
327	32.446674000	158.132.9.103	158.132.8.21	TCP	Seq=257 Ack=453 win=65083
328	32.446683000	158.132.9.103	158.132.8.21	TCP	Seq=257 Ack=453 win=65083
329	32.446735000	158.132.9.103	158.132.8.21	TCP	Seq=2017 Ack=453 win=6508
330	32.446742000	158.132.9.103	158.132.8.21	TCP	Seq=3465 Ack=453 win=6508
331	32.446795000	158.132.9.103	158.132.8.21	TCP	Seq=3617 Ack=453 win=6508
332	32.446809000	158.132.9.103	158.132.8.21	TCP	Seq=5065 Ack=453 win=6508
333	32.446888000	158.132.8.21	158.132.9.103	TCP	Seq=453 Ack=1705 win=8832 Len=0
334	32.446903000	158.132.9.103	158.132.8.21	TCP	Seq=5217 Ack=453 win=6508
335	32.446914000	158.132.9.103	158.132.8.21	TCP	Seq=6665 Ack=453 win=6508
336	32.446986000	158.132.8.21	158.132.9.103	TCP	Seq=453 Ack=3465 win=14592 Len=0
337	32.446994000	158.132.8.21	158.132.9.103	TCP	Seq=453 Ack=5065 win=20352 Len=0
338	32.447158000	158.132.8.21	158.132.9.103	TCP	Seq=453 Ack=6665 win=26112 Len=0
339	32.448096000	158.132.8.21	158.132.9.103	TCP	Seq=453 Ack=7937 win=2905
340	32.448113000	158.132.9.103	158.132.8.21	TCP	Seq=7937 Ack=454 win=65083 Len=0
341	32.448148000	158.132.9.103	158.132.8.21	TCP	Seq=7937 Ack=454 win=6508
342	32.448321000	158.132.8.21	158.132.9.103	TCP	Seq=454 Ack=7938 win=29056 Len=0

Fig 24: Workstation reports to Linux06 (Fetch-Client) about the Session-Test Data

Then, it followed with ordinary 4-way TCP connection termination (Seq#339-342).

Producing the result:

```
--- OWPING statistics from [linux06]:4107 to [158.132.9.103]:2077 ---
SID: 9e840967ceabb8877a4766c61ba93125
first: 2009-11-16T19:39:52.519
last: 2009-11-16T19:40:21.169
300 sent, 0 lost (0.000%), 0 duplicates
one-way delay min/median/max = 34/35/35.8 ms, (unsync)
one-way jitter = 0.7 ms (P95-P50)
TTL not reported
no reordering
```

It is important to realize that the presence of (unsync) in the OWPING summary output. Despite the fact that there existed no routers between the two endpoints (inferred by `traceroute`), the median one-way delay was 35ms; comparing to regular PING, which reported an average of 0.24ms RTT (data not shown).

Recalling the workstation (as Server) was synchronized to Stratum 1 NTP server (stdtime.gov.hk) and linux06 (Client) was synchronized to Stratum 2 NTP server. A major portion of this difference must be contributed by the fact that two endpoints were synchronizing to different time sources.

As mentioned in the Time Synchronization section (p.3), when a client running a relatively “slower” clock launched OWPING against the server running a relatively “faster” clock. The reported one-way delays would be inflated. The receiver interpreted the timestamp of the received packets according to its own clock. The time difference inevitably got into the measured result.

## Same Time Source

Even we were unable to perform cross synchronization, we tried removing this time difference as best as we could. We repeated this experiment but with the workstation (Server) synchronizing to the same time source (Fig 25) as Linux06 (Client).

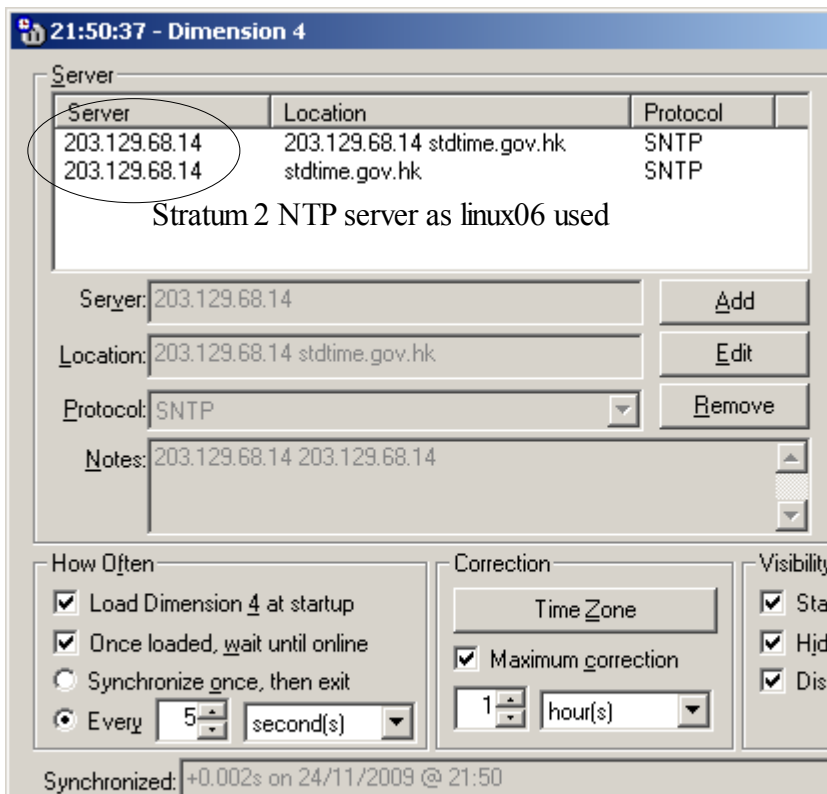


Fig 25: Synchronized to the same Stratum 2 time source like what Linux06 (see Fig 4) did using Dimension4

With 100 packets (each had 200 bytes padding) being used in the respective (ow)PING sessions, the following results were produced:

```
--- owping statistics from [linux06]:8755 to [158.132.8.97]:1746 ---
SID: 9e840861ceb6637b57020c49c9bb4606
first: 2009-11-24T21:51:57.186
last: 2009-11-24T21:53:21.658
100 sent, 0 lost (0.000%), 0 duplicates
one-way delay min/median/max = 0.0358/1.9/3.03 ms, (unsync)
one-way jitter = 0.6 ms (P95-P50)
TTL not reported
no reordering
```

comparing to PING:

```
100 packets transmitted, 100 received, 0% packet loss, time 99002ms
rtt min/avg/max/mdev = 0.117/0.390/5.162/0.556 ms
```

From the result, the one-way delay was not always half to that of RTT. This might be caused by the fact that the medium between two endpoints was actually shared by many workstations that also generate network traffic. This traffic would delay, on data-link layer, the delivery of test packets in this only 100-packet measurement.

Nevertheless, using the PING time as reference, this “same NTP server” version of one-way delay result was more reasonable than the “different NTP server” version as reported earlier.

Even S flag was set (Fig 18, p.14) indicating the client had already synchronized to an external time source, it did not mention the *quality* of the time source and how many NTP time servers a given host was connecting to. This is important in public open OWAMP server scenario in that clients have no access to the server to learn this information. Better confidence on the reported one-way delays could be obtained if endpoint specific NTP meta data is communicated between test peers.

## Additional Factors Contribute to the Error of Measured Delay

Even with time synchronized to the same time source, there exist other factors that contribute to the errors of the reported system time from each individual system. This could be due to the instability of system clock (stressed CPU load, variations in temperature of the server/workstation environment), OS inherent delay while fetching timestamps (delay incurred to process the fetch time instruction), and the delay of the IP stack on the endpoints (the buffer on the receiving end may be stuffed with non-OWAMP related packets that delay the processing of the already-arrive OWAMP packets).

## PING

Given the limitation of linux06, direct reverse one-way delay measurements were not feasible. To approximate the reverse traffic delays, we adopted PING so as to obtain the RTT to a given host at the same time OWPING was executing.

Contrasting to the Windows version, \*nix versions of PING does offer RTT time below 1ms resolution.

We also performed packet captures on a simple 8-round PING exercise (Fig 27).

Time	158.132.8.21	158.132.9.103	Comment
0.000	(0)	Echo (ping) request	ICMP: Echo (ping) request
0.000	(0)	Echo (ping) reply	ICMP: Echo (ping) reply
1.000	(0)	Echo (ping) request	ICMP: Echo (ping) request
1.000	(0)	Echo (ping) reply	ICMP: Echo (ping) reply
2.001	(0)	Echo (ping) request	ICMP: Echo (ping) request
2.001	(0)	Echo (ping) reply	ICMP: Echo (ping) reply
3.001	(0)	Echo (ping) request	ICMP: Echo (ping) request
3.001	(0)	Echo (ping) reply	ICMP: Echo (ping) reply

Fig 27: WireShark PING session traffic profile (ICMP flow)

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	158.132.8.21	158.132.9.103	ICMP	Echo (ping) request
2	0.000020	158.132.9.103	158.132.8.21	ICMP	Echo (ping) reply
3	1.000032	158.132.8.21	158.132.9.103	ICMP	Echo (ping) request
4	1.000048	158.132.9.103	158.132.8.21	ICMP	Echo (ping) reply
5	2.000878	158.132.8.21	158.132.9.103	ICMP	Echo (ping) request
6	2.000892	158.132.9.103	158.132.8.21	ICMP	Echo (ping) reply
7	3.000984	158.132.8.21	158.132.9.103	ICMP	Echo (ping) request
8	3.001002	158.132.9.103	158.132.8.21	ICMP	Echo (ping) reply

0123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901

Fig 26: Packet capture on regular PING request (RFC792 - Internet Control Message Protocol)

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	158.132.8.21	158.132.9.103	ICMP	Echo (ping) request
2	0.000020	158.132.9.103	158.132.8.21	ICMP	Echo (ping) reply
3	1.000032	158.132.8.21	158.132.9.103	ICMP	Echo (ping) request
4	1.000048	158.132.9.103	158.132.8.21	ICMP	Echo (ping) reply
5	2.000878	158.132.8.21	158.132.9.103	ICMP	Echo (ping) request
6	2.000892	158.132.9.103	158.132.8.21	ICMP	Echo (ping) reply
7	3.000984	158.132.8.21	158.132.9.103	ICMP	Echo (ping) request
8	3.001002	158.132.9.103	158.132.8.21	ICMP	Echo (ping) reply

Frame 2 (242 bytes on wire, 242 bytes captured)	
HP-UX Network Tracing and Logging (nettl) header	
Ethernet II, Src: Usi_1c:f1:e6 (00:21:86:1c:f1:e6), Dst: HewlettP_5f:6f:f1 (00:14:c2:5f:6f:f1)	
Internet Protocol, Src: 158.132.9.103 (158.132.9.103), Dst: 158.132.8.21 (158.132.8.21)	
Internet Control Message Protocol	
Type: 0 (Echo (ping) reply)	
Code: 0 ()	
Checksum: 0x105a [correct]	
Identifier: 0xf865	
Sequence number: 1 (0x0001)	
Data (200 bytes)	

0000	00 14 c2 5f 6f f1 00 21 86 1c f1 e6 08 00 45 00	..._o...! .....E.
0010	00 e4 46 7a 40 00 80 01 65 1a 9e 84 09 67 9e 84	..Fz@... e....g..
0020	08 15 00 00 10 5a f8 65 00 01 90 3a 01 4b 98 92	....Z.e ....K..
0030	06 00 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15	.....
0040	16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25	..... !#\$%
0050	26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35	&'()*+,-./012345
0060	36 37 38 39 3a 3b 3c 3d 3e 3f 40 41 42 43 44 45	6789:;<=>?@ABCDE
0070	46 47 48 49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55	FGHIJKLM NOPQRSTU
0080	56 57 58 59 5a 5b 5c 5d 5e 5f 60 61 62 63 64 65	VWXYZ[\] ^_`abcde
0090	66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75	fghijklm nopqrstu
00a0	76 77 78 79 7a 7b 7c 7d 7e 7f 80 81 82 83 84 85	vxyz{ } ~.....
00b0	86 87 88 89 8a 8b 8c 8d 8e 8f 90 91 92 93 94 95	.....
00c0	96 97 98 99 9a 9b 9c 9d 9e 9f a0 a1 a2 a3 a4 a5	.....
00d0	a6 a7 a8 a9 aa ab ac ad ae af b0 b1 b2 b3 b4 b5	.....
00e0	b6 b7 b8 b9 ba bb bc bd be bf c0 c1 c2 c3 c4 c5	.....
00f0	c6 c7	..

Fig 28: Packet capture on regular PING reply

After these rounds of protocol analysis, we were interested to know how OWPING performed across the Internet where the packets must traverse many routers. In the next part, we set up prolonged measurements to understand the delay behavior between one-way delay and RTT against Hong Kong peer OWAMP Server and United States (Chicago) OWAMP Server.

## (B) Internet Delays

### Introduction

Delay Asymmetry is one of the main properties of the Internet. Between two endpoints forwarding packets to each other, this asymmetry exists when the network paths used by the forward communication is not the same as the backward one; even if the paths are identical, the transient characteristics of the forward direction against the backward direction may be different.

Traditional PING measures the round trip time (RTT) that cannot reveal the aforementioned properties of a network. As such, one-way delay measurement tools like OWPING has been developed to measure [3.] the unidirectional delay characteristics.

With OWPING, we were interested to obtain the one-way delay characteristics between Hong Kong Polytechnic University network to some external networks through the Wide Area Network. A list of open OWAMP servers were available for OWAMP tests (<http://dc211.internet2.edu/cgi->

[bin/perfSONAR/serviceList.cgi](#)). We adopted two open OWAMP servers: one in Hong Kong (Chinese University, Hong Kong)(Fig 29) and one in the United States (The University of Chicago, United States) (Fig 30).

## Test Endpoints

Given the limitations of our platform, Linux06 could only assume the Client role. Without the ability to listen for connections, we adopted PING to determine the round trip time (RTT). By executing OWPING and PING to the same target server at the same time, we obtained indirectly the backward one-way delay by subtracting RTT by the measured forward one-way delay.

These network schematics shows the relationship between OWAMP servers and Linux06 in the two measurements.

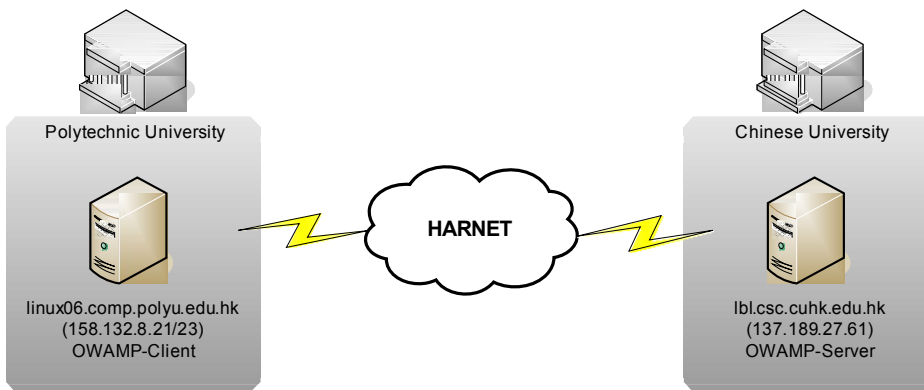


Fig 29: Measuring one-way delay from Polytechnic University to Chinese University (local)

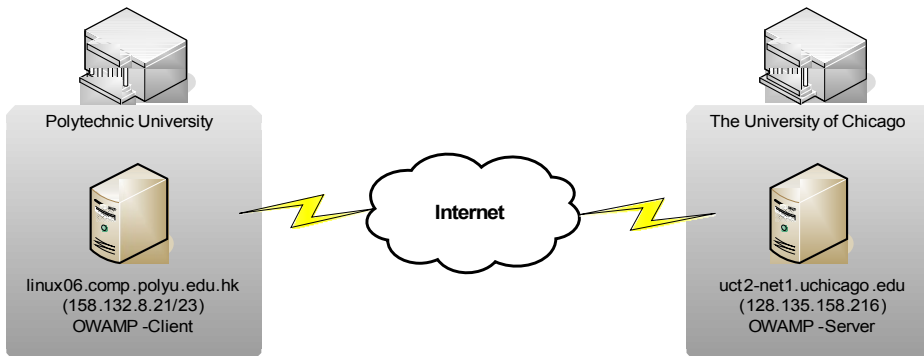


Fig 30: Measuring one-way delay from Polytechnic University to the University of Chicago (overseas)

## Measurement Coverage

	Client	Server
Experiment Location (for OWAMP)	PolyU Linux06	Hong Kong OWAMP Server (lbl.csc.cuhk.edu.hk) United States (Chicago) OWAMP Server (uct2-net1.uchicago.edu)
Measurement Period	Oct 26 to Nov 21, 2009 (27 days)	

## Methods

PING, OWPING and Traceroute were executed in parallel continuously\* for 27 days through custom developed Perl script serving as \*nix cron (necessary for a non-privileged user to launch persistent background tasks).

PING was employed to determine the RTT; whereas OWPING was used to measure the forward one-way delay. By subtracting the RTT with forward one-way delay, the backward one-way delay could be determined indirectly, albeit with the possibility of errors. TraceRoute was executed every 4-hour to help confirm the network path subjected to the measurement was not changing.

Raw logs of PING and OWPING were captured to text files (daily log). These text files were then reformatted by custom developed Perl scripts. Finally the formatted result were exported to Microsoft Excel for analysis.

## Parameters

OWPING and PING were invoked like how they were executed in protocol analysis experiment ((A) Understanding How OWAMP Works, p.6) except the hosts, and that OWPING was executed with interval (1 second per test packet) instead of its default Poisson mode. Please refer to Experiment Planning, p.2 for the related details.

## Data Sampling

Since network traffic and time synchronization can be asymmetric at many sites, this may be the most significant obstacle to measure the delay result. Referring to IETF RFC 4656 standard, we handle the time synchronization with Network Time Protocol (NTP) to provide stable offset estimates that we collected for measurement.

We had carried out two type of protocol ‘PING’ and ‘OWPING’ both on local and oversea (United States-Chicago) public websites. There were four data collections, 24 hours a day, over 27 days.

All data collection had been scheduled on each day by Perl program in Linux OS. Table 1 shows a summary of four data collection we had done on delay and jitter measurements.

Test Code	Date Start	Date End	Protocol	Internet Connection
DM1	26/10	21/11	OWPING	Local
DM2	26/10	21/11	OWPING	Oversea
DM3	26/10	21/11	PING	Local
DM4	26/10	21/11	PING	Oversea

*Table 1: Summary of data collection durations for Internet Delay Measurements*

\*But with system administrator on linux06 randomly terminated our measurement tasks, some hours of data were missing. On the other hand, the instability of OWPING also caused some missing samples. OWPING might somehow could not cope with the change in the network conditions, the parent process remained in memory (looked like as if it was still working) but with its child worker thread terminated. This was verified by using `ps` command.



## Result and Discussion

The measurement results are presented in form of statistics of response. Fig 31-Fig 38 shows the corresponding result of DM1-DM4.

### OWPING Result

Our result in Fig 31 shows that the Hong Kong One Way Ping response result between 26 Oct 2009 to 21 Nov 2009. The minimum and maximum of PING MEDIAN values are 0.1ms and 91.8ms respectively. At first glance, there are great variations of response time on each day especially at noon time. It shows that the network traffic congestion will affect the response time delay.

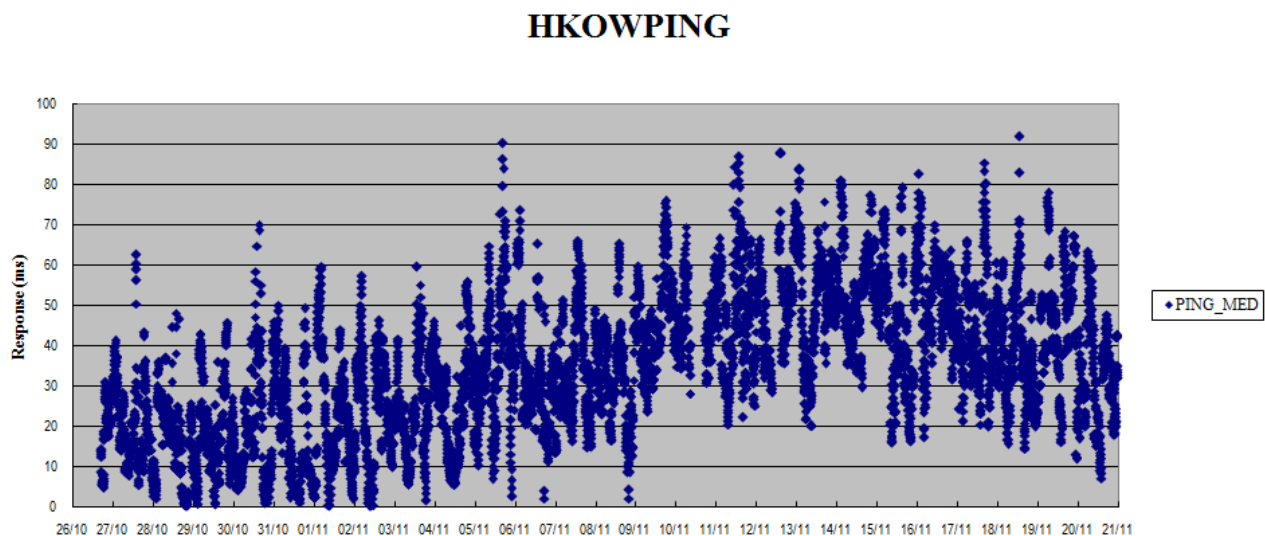


Fig 31: Hong Kong One Way Ping Result –DMI

In comparison, US public website was more stable than Hong Kong public website shows in Fig 32. The average response times are between 94ms and 217ms.

Note that the OWAMP worker thread suicides so there is no response result on Nov 9 to Nov 11.

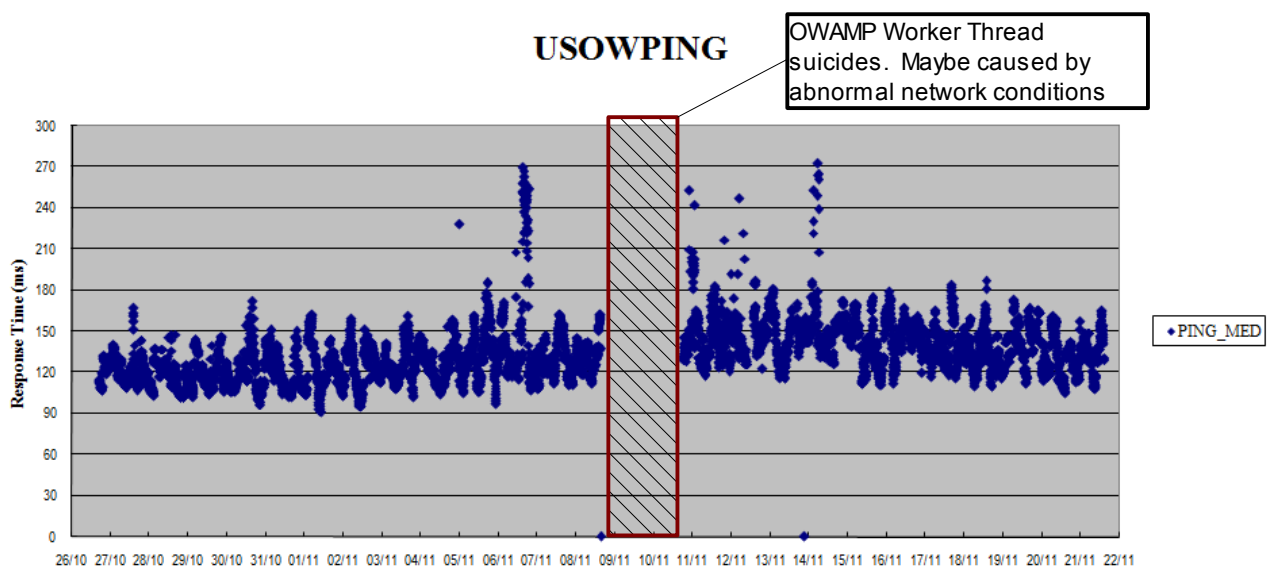


Fig 32: US One Way Ping Result –DM2



In this section, Table 2 and Table 3 give a comparison of two days and timeslots of One Way Ping Delay, the difference between min & max ping and jitter against time. Fig 33 shows the Friday and Sunday response time against time. Fig 34 shows the response time in a single day against time. The data points are connected by lines to merely present the data series more clearly.

USOWP	10/30/09				11/01/09			
Time	PING_MIN	PING_MAX	PING_MED	JITTER	PING_MIN	PING_MAX	PING_MED	JITTER
0:02	122	127	122	0.4	105	109	105	0.3
1:05	113	121	114	0.4	107	112	108	0.2
1:57	129	131	129	0.5	117	142	117	24.7
2:05	109	110	109	0.2	143	145	144	0.8
3:08	107	109	107	0.2	149	186	150	0.9
4:10	110	124	110	1.5	154	159	155	0.8
4:27	117	118	118	0.3	160	163	161	0.8
4:32	116	117	116	0.3	137	163	162	0.8
5:12	108	109	109	0.2	142	142	142	0.3
6:16	114	117	114	0.3	132	145	133	0.1
7:18	129	130	129	0.2	125	128	125	0.4
8:20	121	137	121	0.3	108	111	109	0.9
8:59	112	115	112	0.2	92.4	94.7	93.5	1.1
9:22	123	124	123	0.2	98.3	111	104	1.1
9:37	123	178	123	49.3	96.4	102	97.58	1.1
9:54	123	130	128	0.2	89.9	118	91.2	0.6
10:24	127	130	127	0.3	104	111	106	1.1
11:25	121	145	121	11.1	111	118	111	0.2
12:21	114	764	114	6.5	112	113	112	0.3
12:26	114	121	115	1.1	112	119	113	0.3
13:28	126	168	154	8.7	130	135	130	0.4
14:19	143	193	160	29.1	124	137	124	0.4
14:29	137	179	138	28.8	125	130	125	0.3
15:29	142	188	172	12.1	141	143	141	0.4
16:05	111	112	112	0.1	133	141	134	0.5
16:31	126	170	126	29.3	123	133	131	0.5
17:33	112	149	120	22.3	130	135	131	0.5
18:34	102	120	103	1.2	124	127	124	0.3
19:35	106	140	107	5.3	120	122	120	0.2
20:21	95.4	110	96.4	10.3	120	124	121	0.5
20:37	101	103	102	0.9	125	129	125	0.6
21:38	105	112	105	0.7	114	118	114	0.6
22:40	108	116	110	0.5	108	110	108	0.6
23:46	138	143	140	1.3	112	116	113	0.6
		Min		Max				

Table 2: Comparison of US One Way Ping on Friday (30/10) and Sunday (1/11)

Fig 33 shows that the min & max ping response and jitter on both two days. The packet size is 300 and no hop count change in all transmissions. The min and max PING\_MIN of Friday and Sunday are 95.4\143 and 89.9\160, respectively. And the min and max of PING\_MAX are 103\764 and 94.7\186. The Friday jitter values also higher than Sunday, the jitter values are 49.3 and 24.7 respectively. This can be explained by the fact that the Sunday network is less congested compared to the Friday network.

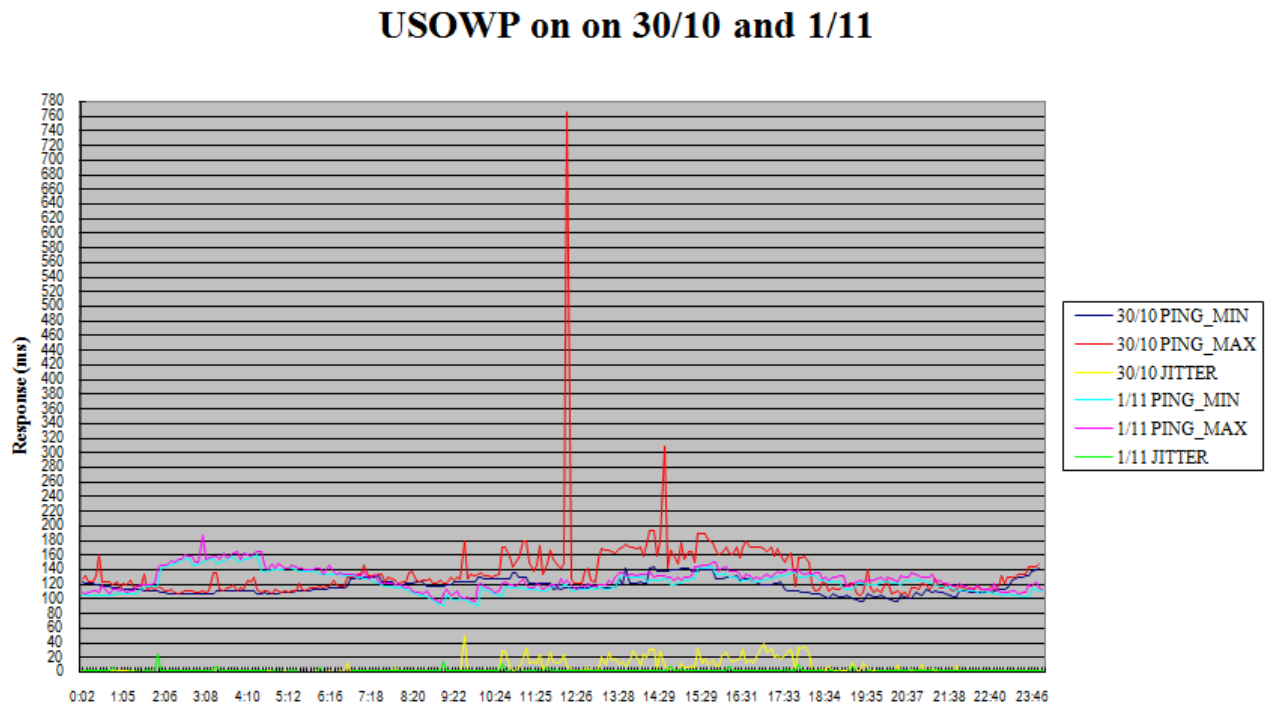
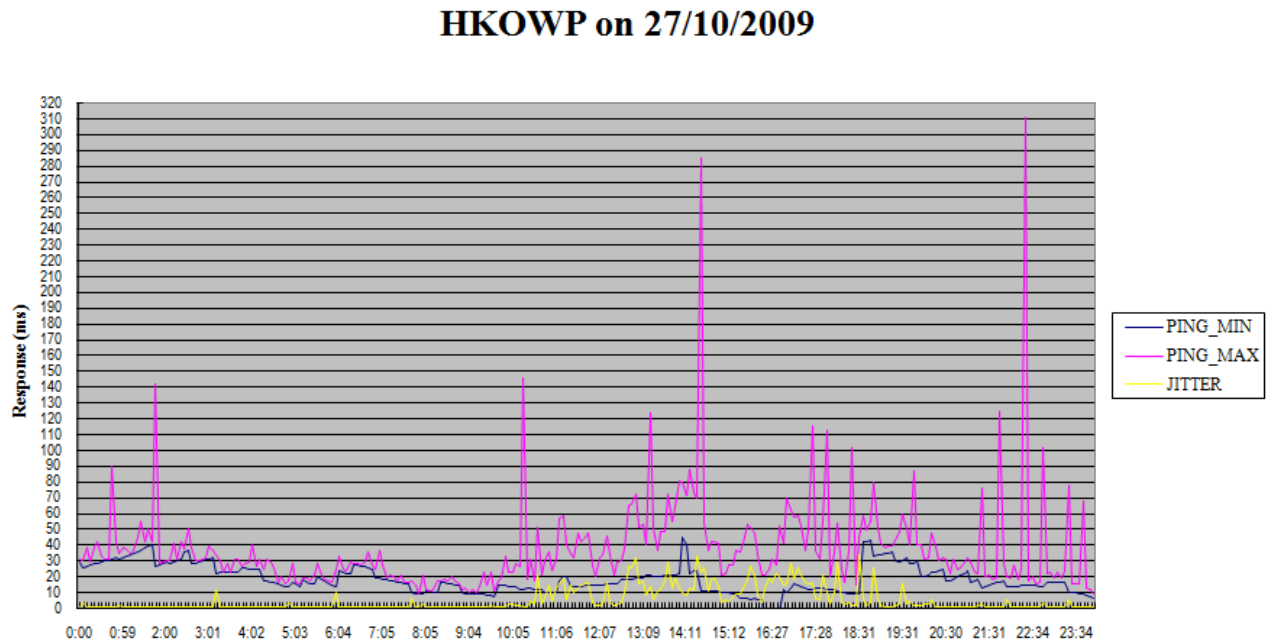


Fig 33: US One Way Ping Result on Friday (30/10) and Sunday (1/11)

HKOWP	10/27/09			
Time	PING_MIN	PING_MAX	PING_MED	JITTER
0:00	29.2	30.7	29.4	0.1
0:59	31.3	37.9	31.9	0.4
2:00	28.5	30	29	0.3
3:01	31.2	39	31.7	0.4
4:02	24.1	40.2	24.4	0.9
5:03	14.7	15.8	15.2	0.5
6:04	23.3	32.8	23.8	0.3
7:05	18	27.2	18.3	0.2
8:05	10.6	11.1	10.8	0.3
9:04	8.84	9.97	9.1	0.3
10:05	13.5	22.9	13.8	1.9
11:06	19.5	58.3	21	19.1
12:07	14.2	31.8	14.6	1.7
13:09	19	52.6	22.8	17.4
14:05	44.7	79.4	62.4	9.2
14:11	39.9	71.1	60.1	7.4
15:12	9.27	27.2	9.8	4.1
15:56	4.72	20.8	5.3	3.3
16:27	11.1	41.1	13.4	14.6
17:28	12.5	36.5	12.9	6.4
18:31	8.28	44.9	8.6	33.5
19:31	30.2	59.5	31.3	14.7
20:30	17.4	30.3	26.1	0.6
21:31	14.1	20.7	14.8	0.7
22:24	13.8	310	14	0.2
22:34	13.8	19.5	13.9	0.2
23:34	9.24	14.7	9.8	0.5
23:59	5.89	8.66	6.5	0.4
		Min		Max

Table 3: Comparison of HK One Way Ping on 27/10

Fig 34 below shows that the ping response and jitter of 27 November. There are no hop count change and reordering in all the simulations. This result indicates the network of morning is better than night time, the max PING\_MED of day and nights are 79.4ms and 310ms.



*Fig 34: Hong Kong One Way Ping Result on 27/10*

## PING Result

We tried to measure the Round Trip Time with traditional ping, the min. and max. of Round Trip Time MEDIAN values are 0.063ms and 40.2ms respectively. The PING result similar to One Way Ping, there is large RTT at noon time. The reason behind HKPING RTT being shorter than one-way delay time must be due to the poor time-source on Linux06. The reported delays were inflated.

### HKPING

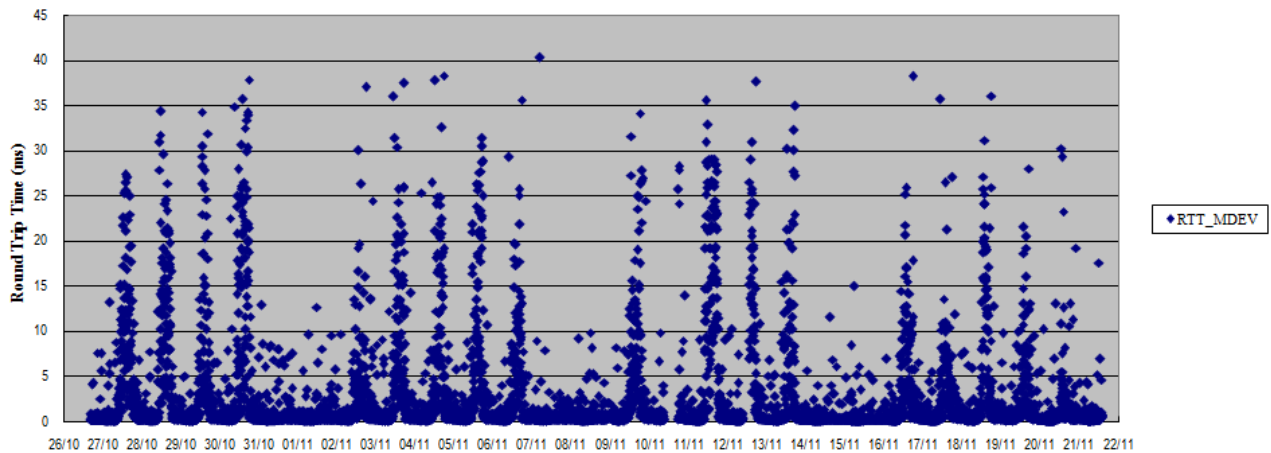


Fig 35: Hong Kong Ping Result –DM3

Fig 36 shows the result of US PING, the min RTT median is 0.081 and the max RTT median is 53466. Hence, the Polytechnic University public server is not stable on 10 November.

### USPING

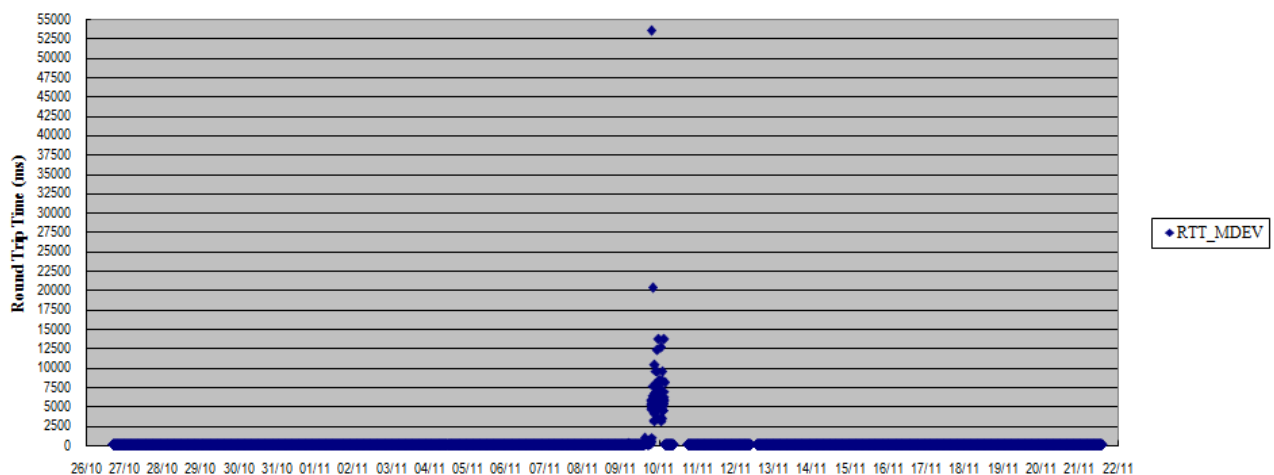


Fig 36: US Ping Result –DM4

Comparison of PING round trip time in Table 4 and Table 5, show the difference between min, max and average of RTT. Fig 37 shows the Friday and Sunday RTT against time using PING protocol. Fig 38 shows the RTT in a single day against time.

USPING	10/30/09				11/01/09			
Time	RTT_MIN	RTT_MAX	RTT_AVG	RTT_MDEV	RTT_MIN	RTT_MAX	RTT_AVG	RTT_MDEV
0:03	205.763	212.878	206.608	1.414	205.72	207.576	206.087	0.466
1:03	205.708	207.416	206.108	0.453	205.571	206.762	206.033	0.38
2:02	205.683	206.498	205.957	0.65	205.626	206.594	205.92	0.422
3:02	205.645	206.81	205.897	0.549	205.683	279.824	206.347	5.62
4:02	205.667	209.275	205.867	0.426	205.603	206.803	205.818	0.357
5:02	205.615	207.785	205.88	0.486	205.6	206.511	205.816	0.382
6:02	205.622	206.058	205.8	0.485	205.614	206.242	205.781	0.572
7:02	205.632	208	205.817	0.479	205.617	207.812	205.846	0.675
8:02	205.625	206.611	205.808	0.627	205.608	206.092	205.775	0.468
9:02	205.622	211.258	206.165	0.939	205.608	207.777	205.874	0.385
10:02	205.678	211.676	206.046	0.634	205.606	206.08	205.816	0.442
11:02	206.129	296.506	231.763	23.414	205.609	206.404	205.887	0.489
12:02	205.89	265.699	212.32	11.207	205.65	206.866	205.905	0.303
13:28	205.787	269.549	213.335	12.506	205.663	206.406	205.94	0.392
14:02	206.057	294.687	243.265	22.293	205.721	206.965	206.045	0.386
15:02	205.738	216.826	206.757	1.479	205.73	207.186	206.076	0.44
16:02	205.933	290.966	219.611	18.315	205.723	207.092	206.061	0.375
17:02	205.811	297.854	235.419	30.068	205.694	206.713	206.045	0.664
18:02	205.897	302.104	235.863	36.449	205.69	207.011	206.015	0.441
19:02	205.769	265.395	208.233	9.219	205.74	206.786	206.058	0.414
20:02	205.754	208.425	206.164	0.649	205.72	207.085	206.073	0.545
21:02	205.694	206.661	206.046	0.273	205.694	207.744	206.046	0.557
22:02	205.739	207.396	206.12	0.598	205.603	211.064	206.11	0.614
23:01	205.682	208.043	206.118	0.59	205.731	206.771	206.071	0.418
	Min			Max				

Table 4: Comparison of US Ping on Friday (30/10) and Sunday (1/11)

### USPING on 30/10 and 1/11

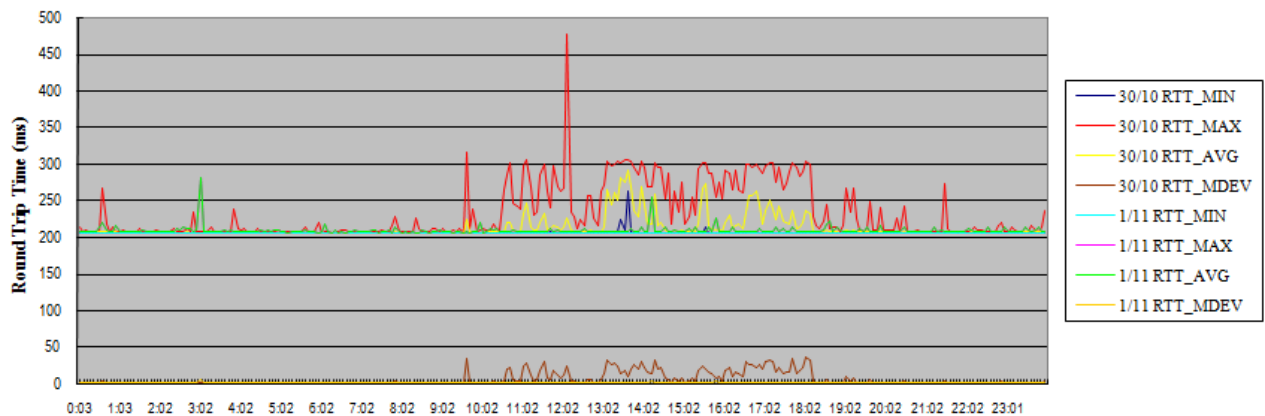


Fig 37: US Ping Result on Friday (30/10) and Sunday (1/11)

Fig 37 above shows the min, max, average and median of round trip time on Friday and Sunday. As a result, the max RTT\_MAX, RTT\_AVG and RTT\_MDEV, 302.104\243.265\36.499 of Friday are higher than Sunday are 279.824\206.347\5.62 respectively.

HKPING	10/27/09			
Time	RTT_MIN	RTT_MAX	RTT_AVG	RTT_MDEV
0:02	1.959	9.682	2.175	0.451
1:01	1.967	10.274	2.195	0.804
2:01	1.941	3.543	2.093	0.136
3:01	1.945	10.301	2.158	0.644
3:16	1.929	2.335	2.039	0.094
4:01	1.941	38.528	2.858	3.936
4:21	1.941	229.944	2.289	13.134
5:01	1.947	7.046	2.07	0.3
5:06	1.944	2.237	2.034	0.085
6:01	1.946	10.174	2.075	0.524
7:01	1.948	2.258	2.025	0.067
7:06	1.939	2.262	2.022	0.066
7:11	1.936	2.357	2.013	0.077
8:01	1.993	10.271	2.094	0.482
9:01	2.048	10.424	2.218	0.502
10:01	2.019	24.272	2.846	1.875
11:01	2.009	25.083	3.642	3.264
12:00	2.12	17.873	2.861	1.567
13:00	2.346	95.233	20.773	21.641
14:00	2.404	108.573	67.284	27.292
14:05	59.383	110.392	86.038	10.072
15:00	2.047	35.277	8.386	3.406
16:00	2.195	25.519	3.22	2.315
17:00	2.12	85.968	25.689	19.267
18:00	1.996	93.087	12.441	19.508
19:00	2.046	41.194	4.471	4.744
20:00	1.992	21.342	3.403	2.34
21:00	1.98	10.521	2.559	0.791
21:59	1.065	8.058	2.241	0.369
22:59	1.965	11.559	2.668	1.161
23:59	1.986	3.21	2.324	0.201
	Min			Max

Table 5: Comparison of HK Ping on 27/10

## HKPING on 27/10

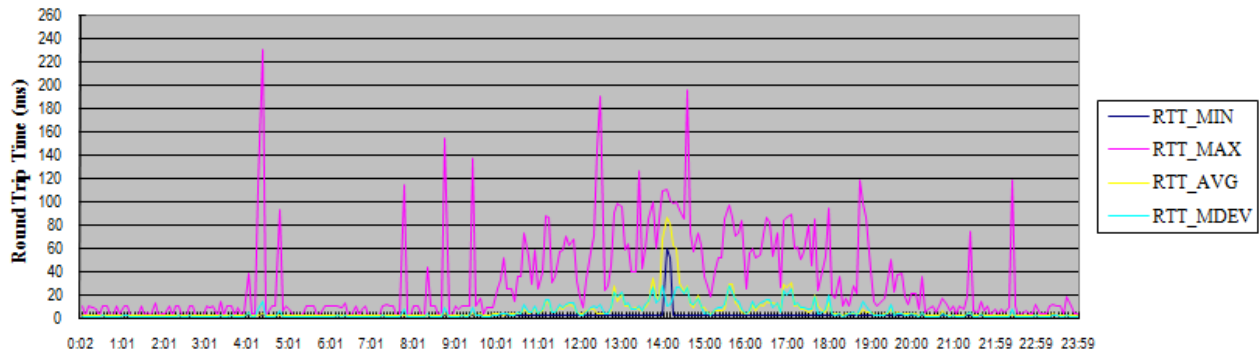


Fig 38: Hong Kong Ping Result on 27/10

In Fig 38, the average RTT for 07:11 and 14:05 were reported to be 2.031ms and 86.038ms respectively. The RTT\_MDEV for 7:06 and 14:00 were reported to be 0.066ms and 27.292ms. This comparison indicates that the network congestion occurred in noon time.

## Delay Asymmetry: Forward v.s. Backward Delay

As per our data collection setup, `linux06` prohibits a regular user to listen on socket. We chose a detour by executing PING in parallel with OWPING to obtain the RTT.

We tried to draw a relationship between PING's RTT and forward one-way delay. By random selection, we picked USOWP (one-way ping) and USP (ping) data for analysis. We calculate the backward delay (by subtracting PING RTT and One-way ping delay) of US one-way ping and shows in Fig 39 below.

Since PING reports Average RTT and OWPING reports Median one-way delay, there should exist deviations in the estimated backward delays.

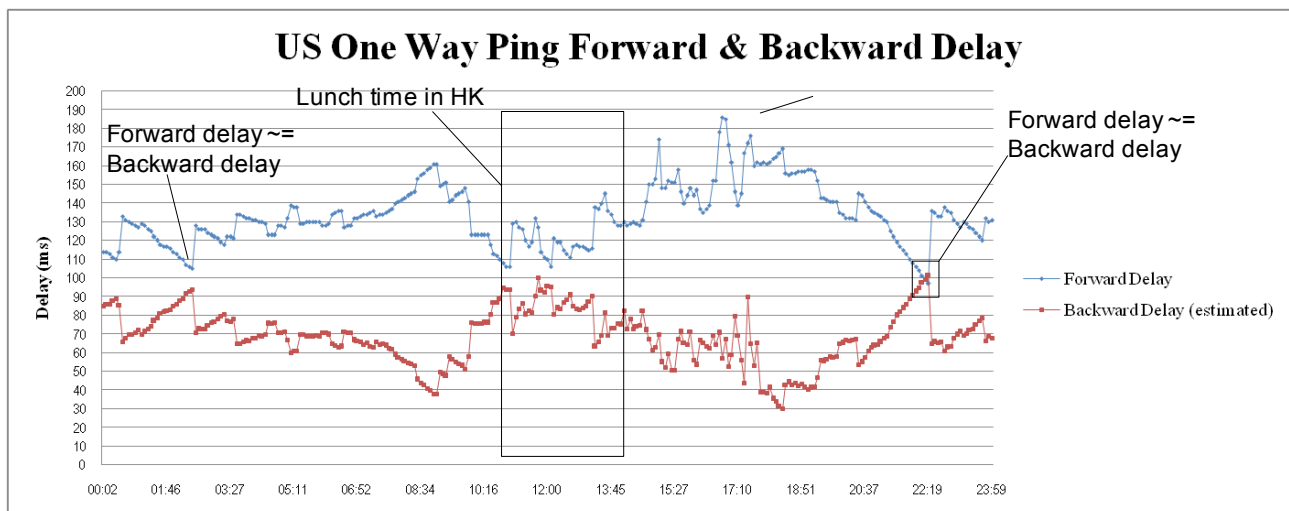


Fig 39: US One Way Ping Forward & Backward Delay on Thursday, 5 Nov 2009

From the result above (Fig 39), it is interesting to note that the trend of forward delay versus the calculated backward delay resembled a mirrored relationship. As in theory, the RTT should be twice of a single way delay. But this mirrored result shows a **highly asymmetric relationship**



between RTT and forward one-way delay. This asymmetric pattern could be attributed to the offset of “business” hours between two time zones: Hong Kong is UTC+0800, whereas US Chicago is UTC-0600 (a 14- hour difference). For example, during lunch hours in Hong Kong, the difference between forward and backward delays were not so much as compared with the afternoon business hours of Hong Kong (1300~1800). We speculate that fewer network activities were going on when everyone was having their lunch; however, more network activities from Hong Kong during “business” hours **inflated the delay in forward direction**.

It is important to understand that the clock `linux06` synchronized to was based only on a Stratum 2 time source. As from the previous protocol analysis experiment (p.17), it generally was slower than Stratum 1 time source by 30~40ms. So if we account for such offset in Fig 39, the blue line should be down-shifted: the backward one-way delays should be higher than forward one-way delays more often. Despite this offset, the asymmetric trend between forward and backward delays were highly represented in our measurement.

In conclusion, the one-way delay was not equal to half of the round trip time as expected. This might be caused by some reasons as below:

1. Clock synchronization in one-way delay, the response result will be affected without an accurate timestamp
2. Network traffic congestion, it makes delay on the response
3. Firewall or third parties program including port filter, policy or prioritize setting
4. Medium rounder, unexpected routers or workstations are shared between two endpoints actually.

# Conclusion

The very requirements of OWAMP measurement turned us to departmental time sharing machine `linux06` which came with its limitations. These limitations in turn sculptured our OWAMP project scope/evaluation strategies.

In particular, instead of trying to perform accurate measurements & compare the result against RTT, we switched our focus on studying the protocol itself, and speculated on the traffic pattern in a relative sense. Despite the failure in obtaining high quality time source in our tests, we researched heavily about Network Time Protocol (NTP), and gained insights between the relationship of NTP and OWAMP.

The result could be much realistic if it existed a place to host our own computers for data collections.

As from the Internet Delays measurement, the asymmetric pattern was obtained. This reported [3.] asymmetry was re-produced somewhat.

As concluded from our OWAMP protocol analysis experiments. Even with two endpoints synchronized to NTP time sources that were "perceived" accurate, there existed a noticeable gap between two clocks.

Therefore, for one-way delay time to be highly creditable, the quality of time sources should be communicated between test peers. The current OWAMP protocol infrastructure does not convey such important information between two endpoints. Furthermore, the following factors could also contribute to the overall error and should be shared explicitly: the frequency of one host synchronizing to the upstream NTP servers and their host IP, and the average CPU loading at the time of measurement.

With this proposition, it would help remote endpoint to evaluate the *correctness of error estimate* in each test packet. On the other hand, it could possibly be used to offset the time error between two hosts (soft-sync) without the need to change the system clock that requires privileged access to the hosts concerned. OWPING could become as popular as PING if this is at all possible.

Finally, the stability and the compliance of the reference implementation might need to be addressed. As for the latter, it might be caused by the implicitness of RFC [1., p.24] concerning the Fetch-Session message. We actually took some hours to realize the fact that the Fetch-Session message was piggy backed.

## References

1. [RFC4656] A One-way Active Measurement Protocol (OWAMP) (<http://www.rfc-editor.org/rfc/rfc4656.txt>) <http://tools.ietf.org/html/rfc4656>
2. OWAMP CookBook (<http://www.internet2.edu/pubs/owamp-cookbook.pdf>)
3. Pathak et al, *Characterizing Internet Delay Asymmetry* (2007), <http://web.ics.purdue.edu/~pathaka/papers/2007-sigcomm-pathak.pdf>