Campus-Wide Smart Grid Communication Network
Project Report

Gao Peng
School of Computer and Communication Sciences
Swiss Federal Institute of Technology, Lausanne
Email: p.gao@epfl.ch

September 16, 2012

Contents

1 How to Read the Document **3**

2 Test Set Up: Point-to-Point Connection using SHDSL **5**
2.1 Objective and motivation 5
2.2 Configuring the point-to-point connection 5
2.2.1 Wiring 6
2.2.2 Configuration of the SHDSL router 6
2.2.3 Miscellaneous details and remarks 9
2.3 Performance tests of the point-to-point connection 9
2.3.1 Round trip time 9
2.3.2 Maximum achievable throughput 10
2.4 Chapter Summary 11

3 Fragmentation Deteriorates Throughput **12**
3.1 Objective and motivation 12
3.2 1470 Bytes: Causing trouble? 12
3.3 Analyzing the trace using wireshark 13
3.4 Does the size of datagrams matter? 16
3.5 Effects of fragmentation in theory 17
3.6 Conclusion on UDP datagram size 19

4 Sending Pattern Affects Packet Dropping Probability **20**
4.1 Objective and motivation 20
4.2 Defining two spacing patterns 20
4.2.1 Spacing Pattern I 21
4.2.2 Spacing Pattern II 21
4.3 Equal sized datagrams with different spacing patterns 21
4.4 Unequal sized datagrams with different sending patterns 23
4.5 Revisit throughput drop caused by fragmentation 25

4.6 Chapter summary 25

5 Maximum Burst Size 27

5.1 Objective and motivation 27

5.2 How do the tests work? 27

5.3 Results and analysis 28

5.4 Chapter Summary 29

6 Summary and Key Findings 31

Chapter 1

How to Read the Document

This document serves as a log file for recording the progress and output of the EPFL campus-wide smart grid communication network project. The general objective of the project is to come up with an operational region which the PMUs are advised to work within in order to better utilize the bandwidth of the SHDSL link. The document is also a complementary document to the design specification document for the EPFL campus-wide smart grid communication network. The document keeps track of technical and hands-on details of setting up the communication network infrastructure in an organized way so that it could be used as a reader-friendly reference document in case someone wants to repeat the procedures again. The document also describes in details the performance issue related tests designed and implemented and presents the corresponding results and analysis so that they could be referenced in the future for trouble shooting reasons. Relative data collected in the tests and source codes for the test programs are also packed alongside the document.

The organization of the document is generally based on the chronological order of the tasks being executed. Each chapter covers a more or less independent topic or task that has been done in order to move the project forward. Chapter 2 is about the procedures to set up a point-to-point connection using SHDSL for performance tests, which is the starting point of the communication network we have designed for the EPFL campus wide smart grid. Also basic performance test results such as round trip time and throughput are presented in Chapter 2. Chapter 3 follows up the throughput drop issue observed when we first used Iperf to test the throughput. We started by analyzing the network traffic trace using Wireshark and found out that fragmentation was the reason for the drastic throughput drop in the Iperf throughput tests using default UDP datagram size. We further devised several tests to explore factors that affect the packet dropping probability in Chapter 4 and concluded Chapter 4 with the finding that spacing between packets is the crucial factor that determines the packet dropping probability given our network configuration. The maximum burst size that the ZyXEL line terminal device could support without losing any data is studied in Chapter 5. The test results show that when the queue is empty, the ZyXEL line terminal device we use can support in maximum a burst of 51 datagrams. Chapter 6 summarizes our key findings and draw a conclusion on the safe region within which the PMU application should work so as to make the best use of the SHDSL link.

Each chapter will be structured in the following way: the first section of a chapter mainly discusses the objective and motivation of implementing the task or test, the link between the task and the final project output; the following sections will present in details about how the tasks are implemented and analysis of the relevant test results; the last section usually summarizes the findings of the chapter and draws conclusion on the topic of the chapter.

Chapter 2

Test Set Up: Point-to-Point Connection using SHDSL

2.1 Objective and motivation

The communication between each measurement site and the concentration point for the wired phase solution will be done over the existing twisted pair cabling infrastructure using the SHDSL technology. In order to test the SHDSL line terminal device that we will deploy, we start by setting up a point-to-point connection using the SHDSL device and exploring different configuration options of the device. Also certain performance tests of the point-to-point connection were carried out to verify that our design specifications meet the network delay and throughput requirements.

2.2 Configuring the point-to-point connection

This section will walk you through the procedure of setting up a point-to-point connection using ZyXEL P-791R v2 SHDSL router over the existing twisted pair cable on EPFL campus. Here it is a bit of abuse of terminology since the SHDSL router is the product name while in our configuration that uses IPv6 it is working as a bridge. To avoid ambiguities, we will call it ZyXEL line terminal device from this point on.

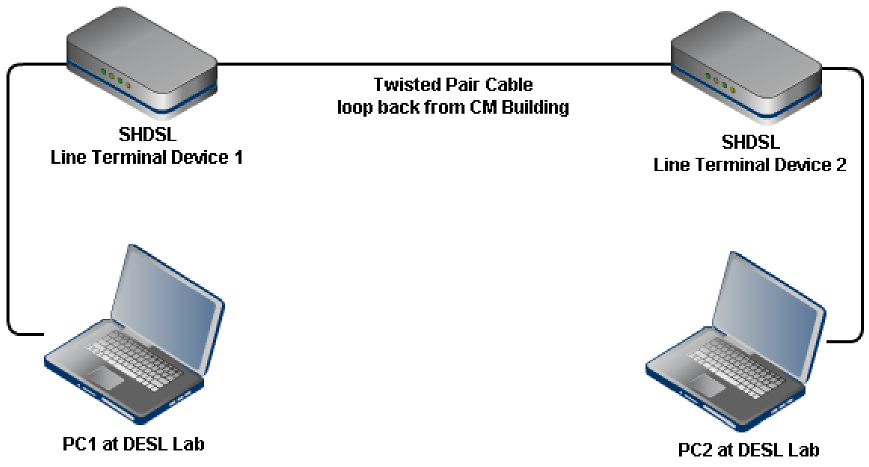


Figure 2.1: Wiring of the point-to-point connection

2.2.1 Wiring

In order to obtain a sense of the real network delay, we used the twisted pair cables at DESL lab that loops back from CM building. The hardware used for setting up the point-to-point connection are two laptop computers with linux operating system and two ZyXEL P-791R v2 SHDSL line terminal devices. The wiring is illustrated by Figure 2.1.

2.2.2 Configuration of the SHDSL router

There are several ways to configure the ZyXEL router.

- Web Configurator. This is recommended for everyday management of the ZyXEL Device using a web browser.
- Command Line Interface. Line commands are mostly used for troubleshooting.
- FTP. Use File Transfer Protocol for firmware upgrades and configuration backup/restore.
- SNMP. The device can be monitored and/or managed by an SNMP manager.

For the purpose of setting up and testing the point-to-point connection, we used web configuration interface and the command line interface. The password for administration is asked to be set after the first login to the web configuration interface. We set the password to **test**. And the configuration files of the ZyXEL line terminal device for the two cases we have explored were backed up for future use and they're packed together with this document in the folder "ZyXEL_Config".

To establish a point-to-point connection, one of the ZyXEL Devices becomes the server (instead of the ISP). The server controls some of the attributes of the DSL connection, such as the transfer rate and the DSL operational mode. Other than that, there is no difference between the server and the client. Either one can initiate the point-to-point connection.

We did the following to configure the server.

- Click Network—WAN—Internet Connection.
- Configure the VPI, VCI, Multiplexing, and Encapsulation fields for the point-to-point connection. In the Encapsulation field, select either RFC 1483 or ENET ENCAP.
- In the Service Type field, select Server. The rest of the fields are enabled.
- Configure the rest of the fields, if necessary. For example, you might want to set the Transfer Max Rate to the maximum value.
- Click Apply.

We did the following to configure the client.

- Click Network—WAN—Internet Connection.
- Set the VPI, VCI, Multiplexing, and Encapsulation to the same values set in the server.
- In the Service Mode field, select the same type of connection you selected for the server.
- In the Service Type field, select Client. The rest of the fields will be negotiated with the server.
- Click Apply.

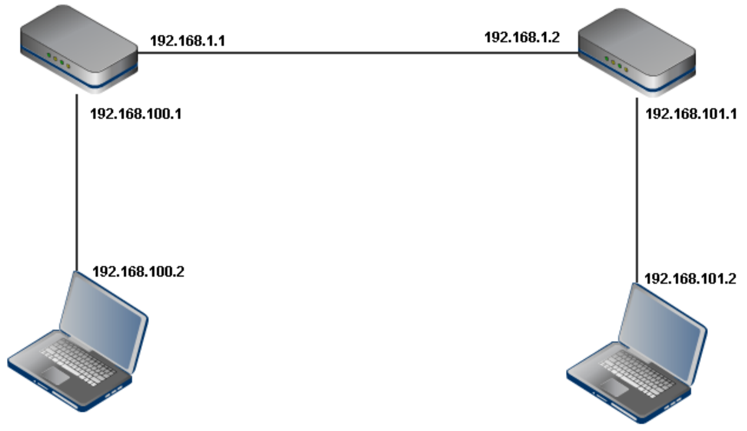


Figure 2.2: Routing mode with IPv4

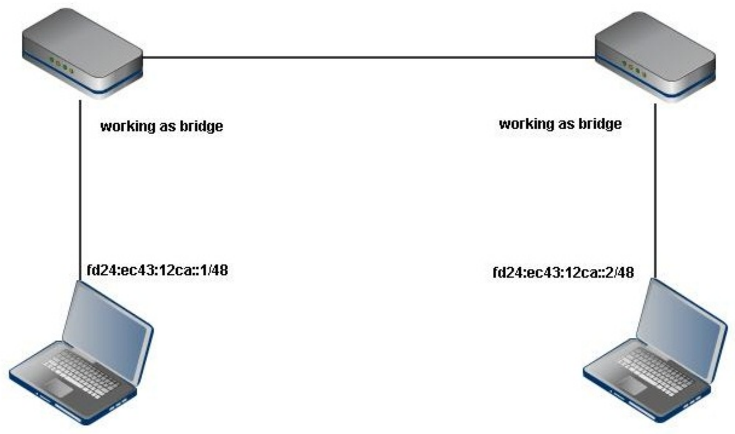


Figure 2.3: Bridging mode with IPv6

After completing the configuration of the server and the client, wait up to about one minute for the router to restart. If the configuration is right, the LED lights for DSL, Ethernet and Internet should be green. If the line terminal device is configured to work as a bridge, the Internet light should be off.

We tested two network configurations. In the first case, the ZyXEL line terminal device worked in the routing mode with IPv4. In the second case, the ZyXEL line terminal device was configured to work as a bridge in order to use IPv6. The two network configurations are illustrated by Figure 2.2 and Figure 2.3 respectively.

2.2.3 Miscellaneous details and remarks

The followings are several things that need to pay attention to when setting up the point-to-point connection.

- If the ZyXEL line terminal device is configured to work in routing mode, the Encapsulation method, VPI, VCI and multiplexing should be set to be the same for the server and the client. Double check these fields' values if the configuration doesn't work.
- If the ZyXEL line terminal device is configured to work in bridging mode, the IP address of the LAN interface still needs to be assigned and it only accepts IPv4 address. If you want to change the configuration of the device through web interface later(e.g. change back to routing mode), use this IP address.
- If you want to ping the WAN interface of the each router, check to make sure that they are configured to respond to both WAN and LAN ping. The configuration is under Advanced—Remote MGMT—ICMP.
- The NAT is set to be on as default. In the routing mode with IPv4 configuration, if you want to ping PC2 from PC1, make sure NAT is turned off. It can be turned off under Network—NAT—Disable all the NAT functions.

2.3 Performance tests of the point-to-point connection

Since our design specification chose to use IPv6 for network layer protocol and UDP for transport layer procotol, we performed several tests using the configuration of the second case illustrated in Figure 2.3 to evaluate the performance of the point-to-point connection and validate our design specification choices.

2.3.1 Round trip time

We first tested the end-to-end round trip time using ping6 command. The result is shown in Table 2.1. The statistics are based on 50 pings. The average rtt between the DESL and the CM building is 3.410ms. Basicly, the rtt between each measurement site and the concentration point will be approximately the same magnitude.

Table 2.1: Ping6 Results

rtt	min/avg/max/mdev	3.105/3.410/9.884/0.840 ms

2.3.2 Maximum achievable throughput

We tested the achievable throughput using Iperf with IPv6 and UDP (in the case of iperf, the value reported is goodput, i.e. the application layer throughput). Without further clarification, in this document, when referring to throughput, we are talking about application layer throughput. The tests were ran in the following way:

- Run # iperf -s -V -u -l 1452 on one PC
- Run # udpthroughput.sh on the other PC

where udpthroughput.sh is the shell script that repeats the iperf throughput tests for different targeted throughput values, i.e. the rate that the application sends data. For each different targeted throughput value, the test runs for 10 seconds and records the achieved throughput and datagram loss rate. After a 5-second pause, the test reruns again with a increase of the targeted throughput value by 25Kbps. The script used for the throughput test and all the relevant test results are packed with the document in the folder "Iperf_Throughput".

Figure 2.4 shows the achieved throughput with respect to the targeted throughput. We claim that the maximum achievable throughput when sending UDP datagram of 1452 bytes (throughout the document, when talking about the size of the UDP datagrams, we are talking about the payload size of the datagrams) without datagram loss in our network configuration in Figure 2.3 is around 1.97Mbps with a precision of 25Kbps. After we reach the maximum achievable throughput, the achieved throughput stays stable at 1.97Mbps.

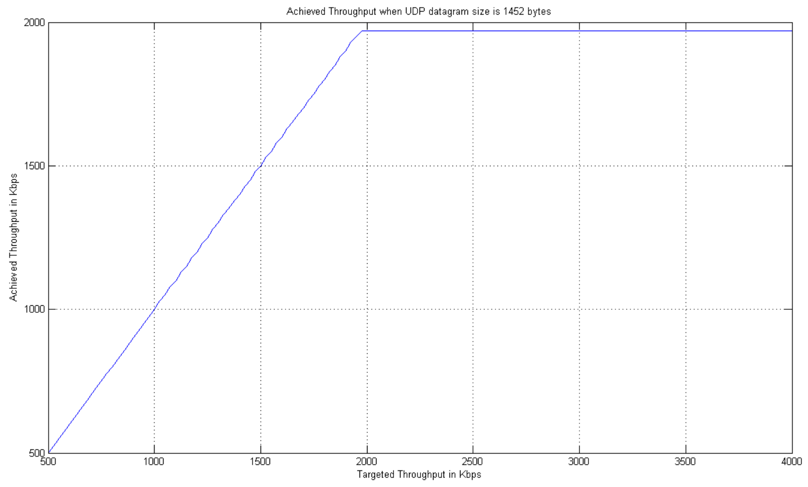


Figure 2.4: UDP Throughput Test

2.4 Chapter Summary

In this chapter, we walked through the procedural details for setting up point-to-point connection using ZyXEL line terminal device step by step. This set up is the starting point of the communication network infrastructure and also works as the testing environment for the following chapters. And performance tests results show that the round trip time from DESL lab to CM building is around 3.410ms and with our network set up, the maximum achievable throughput when sending 1452 bytes UDP datagrams is around 1.97Mbps with a precision of 25Kbps.

Chapter 3

Fragmentation Deteriorates Throughput

3.1 Objective and motivation

We encountered a weird throughput behavior when we first used Iperf to test the throughput. Iperf reported a drastic throughput drop after we reach the maximum bandwidth. And by analyzing the traffic trace using Wireshark we found out that the default UDP datagram size used by Iperf doesn't fit for IPv6 and results in fragmentation which deteriorates throughput. This chapter will first present the issue and tests we performed to investigate the issue, then discuss in further details about how fragmentation affects the application throughput. At the end of the chapter, we'll draw a conclusion on that we should by all means avoid fragmentation. Moreover, based on test results we come up with a recommended size interval of UDP datagrams the application should use to better utilize the bandwidth.

3.2 1470 Bytes: Causing trouble?

If we repeat the throughput test in Section 2.3.2 without the option -l specifying the size of the UDP datagrams, Iperf will use 1470 default UDP datagram size. And the Iperf reported achieved throughput will be as in Figure 3.1. We observed a drastic throughput drop after the targeted throughput reaches value around 1.83Mbps. This is an undesirable behavior and caught our attention to further investigate the issue. We'll present in the following sections the tests we performed to find the reason for the drastic throughput drop.

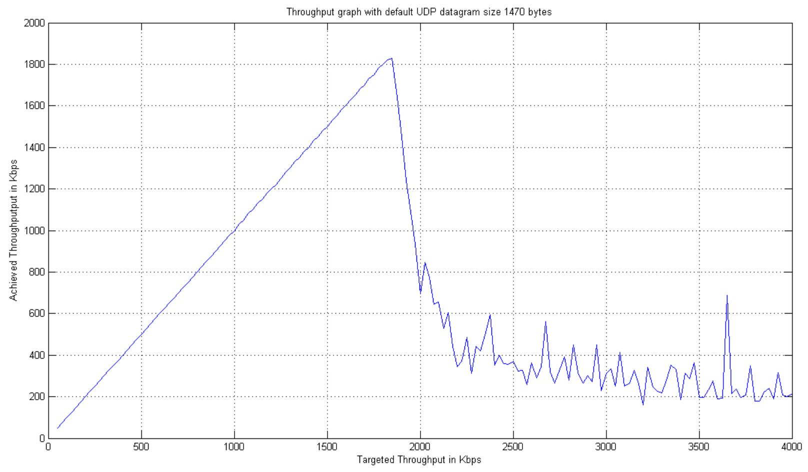


Figure 3.1: Throughput with 1470 bytes UDP datagrams

3.3 Analyzing the trace using wireshark

We start the investigation by analyzing the traffic running on the network. We repeated the throughput tests using Iperf with IPv6 bridging configuration in Figure 2.3, observed and analyzed the trace at sender and receiver with the help of Wireshark. The network configuration and the observation points are indicated in Figure 3.2. The procedures of the tests are listed below:

- We run Iperf on receiver PC in the server mode using the command: #iperf -s -u -V
- We run Iperf on sender PC in the client mode using the command: #iperf -c fd24:ec43:12ca::2 -V -u -b 3m
- Compare and analyze the traffic captured by Wireshark at observation point A and B.

With the above commands, the sender PC sends UDP datagrams of 1470 bytes (default UDP datagram size set by Iperf) at the rate of 3Mbps to the receiver. Figure 3.3 and Figure 3.4 shows the traffic captured at observation point A and B.

By analyzing the trace observed at observation point A and B, we found

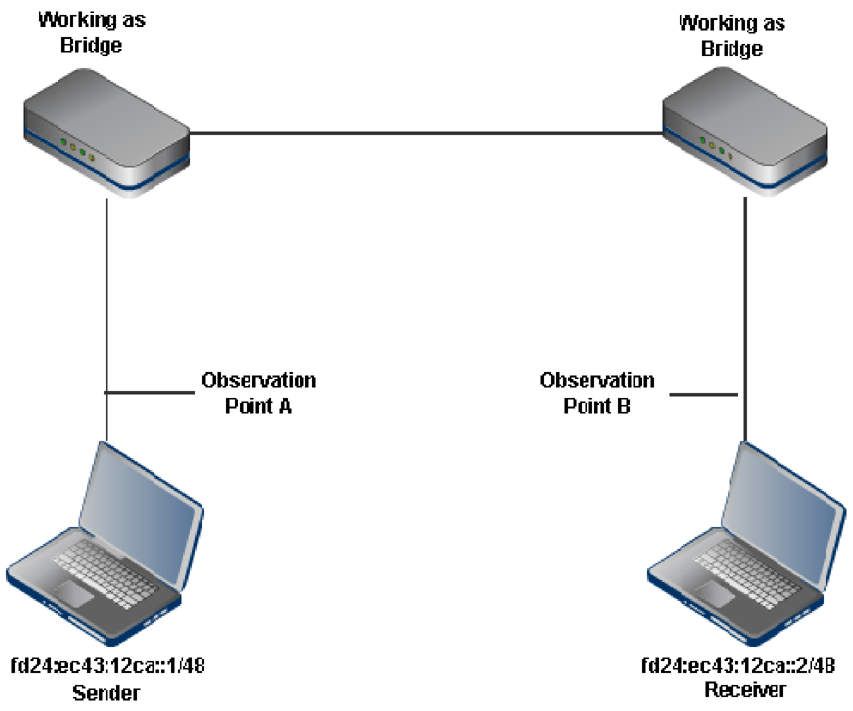
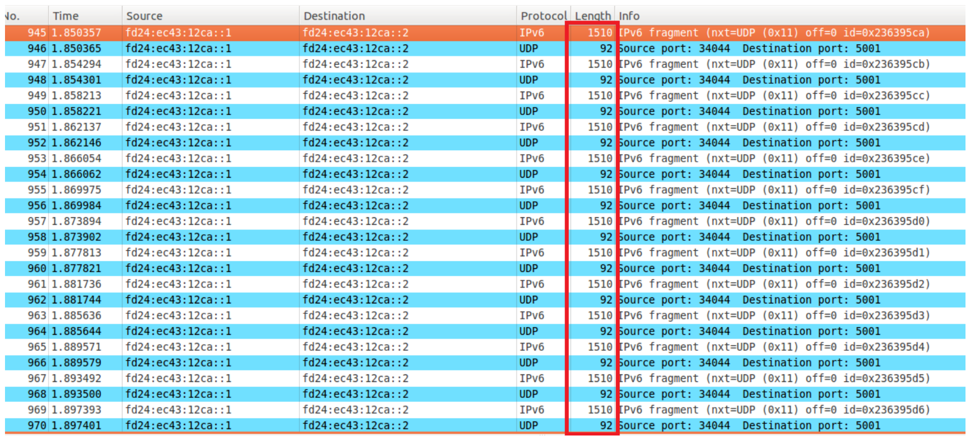
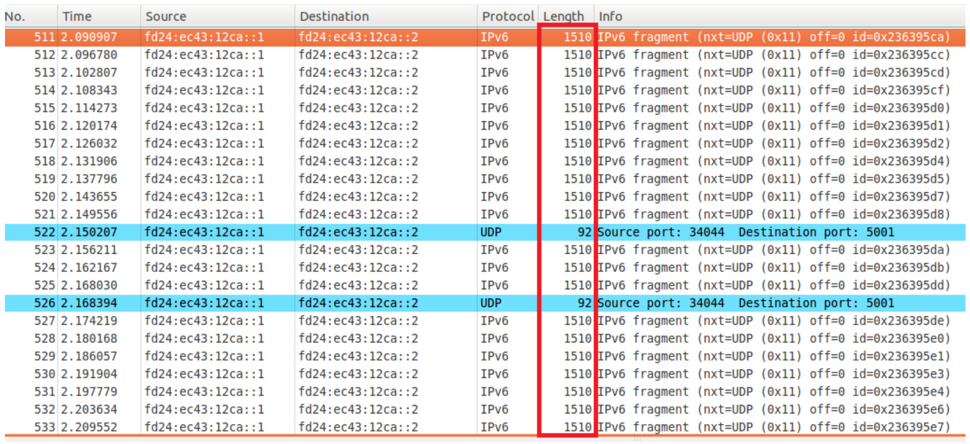


Figure 3.2: Observation Points



No.	Time	Source	Destination	Protocol	Length	Info
945	1.850357	fd24:ec43:12ca::1	fd24:ec43:12ca::2	IPv6	1510	IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395ca)
946	1.850365	fd24:ec43:12ca::1	fd24:ec43:12ca::2	UDP	92	Source port: 34044 Destination port: 5001
947	1.854294	fd24:ec43:12ca::1	fd24:ec43:12ca::2	IPv6	1510	IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395cb)
948	1.854301	fd24:ec43:12ca::1	fd24:ec43:12ca::2	UDP	92	Source port: 34044 Destination port: 5001
949	1.858213	fd24:ec43:12ca::1	fd24:ec43:12ca::2	IPv6	1510	IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395cc)
950	1.858221	fd24:ec43:12ca::1	fd24:ec43:12ca::2	UDP	92	Source port: 34044 Destination port: 5001
951	1.862137	fd24:ec43:12ca::1	fd24:ec43:12ca::2	IPv6	1510	IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395cd)
952	1.862146	fd24:ec43:12ca::1	fd24:ec43:12ca::2	UDP	92	Source port: 34044 Destination port: 5001
953	1.866054	fd24:ec43:12ca::1	fd24:ec43:12ca::2	IPv6	1510	IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395ce)
954	1.866062	fd24:ec43:12ca::1	fd24:ec43:12ca::2	UDP	92	Source port: 34044 Destination port: 5001
955	1.869975	fd24:ec43:12ca::1	fd24:ec43:12ca::2	IPv6	1510	IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395cf)
956	1.869984	fd24:ec43:12ca::1	fd24:ec43:12ca::2	UDP	92	Source port: 34044 Destination port: 5001
957	1.873894	fd24:ec43:12ca::1	fd24:ec43:12ca::2	IPv6	1510	IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395d0)
958	1.873902	fd24:ec43:12ca::1	fd24:ec43:12ca::2	UDP	92	Source port: 34044 Destination port: 5001
959	1.877813	fd24:ec43:12ca::1	fd24:ec43:12ca::2	IPv6	1510	IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395d1)
960	1.877821	fd24:ec43:12ca::1	fd24:ec43:12ca::2	UDP	92	Source port: 34044 Destination port: 5001
961	1.881736	fd24:ec43:12ca::1	fd24:ec43:12ca::2	IPv6	1510	IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395d2)
962	1.881744	fd24:ec43:12ca::1	fd24:ec43:12ca::2	UDP	92	Source port: 34044 Destination port: 5001
963	1.885636	fd24:ec43:12ca::1	fd24:ec43:12ca::2	IPv6	1510	IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395d3)
964	1.885644	fd24:ec43:12ca::1	fd24:ec43:12ca::2	UDP	92	Source port: 34044 Destination port: 5001
965	1.889571	fd24:ec43:12ca::1	fd24:ec43:12ca::2	IPv6	1510	IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395d4)
966	1.889579	fd24:ec43:12ca::1	fd24:ec43:12ca::2	UDP	92	Source port: 34044 Destination port: 5001
967	1.893492	fd24:ec43:12ca::1	fd24:ec43:12ca::2	IPv6	1510	IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395d5)
968	1.893500	fd24:ec43:12ca::1	fd24:ec43:12ca::2	UDP	92	Source port: 34044 Destination port: 5001
969	1.897393	fd24:ec43:12ca::1	fd24:ec43:12ca::2	IPv6	1510	IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395d6)
970	1.897401	fd24:ec43:12ca::1	fd24:ec43:12ca::2	UDP	92	Source port: 34044 Destination port: 5001

Figure 3.3: Traffic at observation point A



```
No.  Time      Source              Destination         Protocol Length Info
 511 2.090907  fd24:ec43:12ca::1   fd24:ec43:12ca::2   IPv6       1510 IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395ca)
 512 2.096780  fd24:ec43:12ca::1   fd24:ec43:12ca::2   IPv6       1510 IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395cc)
 513 2.102807  fd24:ec43:12ca::1   fd24:ec43:12ca::2   IPv6       1510 IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395cd)
 514 2.108343  fd24:ec43:12ca::1   fd24:ec43:12ca::2   IPv6       1510 IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395cf)
 515 2.114273  fd24:ec43:12ca::1   fd24:ec43:12ca::2   IPv6       1510 IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395d0)
 516 2.120174  fd24:ec43:12ca::1   fd24:ec43:12ca::2   IPv6       1510 IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395d1)
 517 2.126032  fd24:ec43:12ca::1   fd24:ec43:12ca::2   IPv6       1510 IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395d2)
 518 2.131906  fd24:ec43:12ca::1   fd24:ec43:12ca::2   IPv6       1510 IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395d4)
 519 2.137796  fd24:ec43:12ca::1   fd24:ec43:12ca::2   IPv6       1510 IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395d5)
 520 2.143655  fd24:ec43:12ca::1   fd24:ec43:12ca::2   IPv6       1510 IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395d7)
 521 2.149556  fd24:ec43:12ca::1   fd24:ec43:12ca::2   IPv6       1510 IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395d8)
 522 2.150207  fd24:ec43:12ca::1   fd24:ec43:12ca::2   UDP          92 Source port: 34044  Destination port: 5001
 523 2.156211  fd24:ec43:12ca::1   fd24:ec43:12ca::2   IPv6       1510 IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395da)
 524 2.162167  fd24:ec43:12ca::1   fd24:ec43:12ca::2   IPv6       1510 IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395db)
 525 2.168030  fd24:ec43:12ca::1   fd24:ec43:12ca::2   IPv6       1510 IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395dd)
 526 2.168394  fd24:ec43:12ca::1   fd24:ec43:12ca::2   UDP          92 Source port: 34044  Destination port: 5001
 527 2.174219  fd24:ec43:12ca::1   fd24:ec43:12ca::2   IPv6       1510 IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395de)
 528 2.180168  fd24:ec43:12ca::1   fd24:ec43:12ca::2   IPv6       1510 IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395e0)
 529 2.186057  fd24:ec43:12ca::1   fd24:ec43:12ca::2   IPv6       1510 IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395e1)
 530 2.191904  fd24:ec43:12ca::1   fd24:ec43:12ca::2   IPv6       1510 IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395e3)
 531 2.197779  fd24:ec43:12ca::1   fd24:ec43:12ca::2   IPv6       1510 IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395e4)
 532 2.203634  fd24:ec43:12ca::1   fd24:ec43:12ca::2   IPv6       1510 IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395e6)
 533 2.209552  fd24:ec43:12ca::1   fd24:ec43:12ca::2   IPv6       1510 IPv6 fragment (nxt=UDP (0x11) off=0 id=0x236395e7)
```

Figure 3.4: Traffic at observation point B

out that the default UDP datagram size 1470 bytes used by Iperf was a bad choice for IPv6. After appending the UDP header and IPv6 header, the IP packet payload will be 1470+8+40=1518 bytes which exceeds the MTU size 1500 bytes. Hence, after the datagram of 1470 bytes was passed down to IP layer, fragmentation happens and finally result in two ethernet frames that correspond to the original UDP datagram. As we can see from the traffic from the sender's side, the UDP datagram of size 1470 bytes are fragmented into two packets resulting in two ethernet frames of size 1510 bytes (1448 bytes data + 8 bytes UDP header + 40 bytes IPv6 header + 14 bytes ethernet header) and size 92 bytes (22 bytes data + 8 bytes UDP header + 8 bytes fragmentation header + 40 bytes IPv6 header + 14 bytes ethernet header). Losing either fragment will be treated as that the whole datagram is lost by Iperf which will decrease the throughput reported by Iperf. Careful readers at this point may have noticed the large difference in number of 1510 bytes ethernet frames and 92 bytes ethernet frames shown at receiver end's traffic in Figure 3.4. We'll come back to this in Chapter 4 and discuss the factors that affect the packet dropping probability which in return affects throughput when fragmentation happens.

We now propose the hypothesis that fragmentation is the reason for the drastic throughput drop and then in the following section we'll present a test performed to verify our hypothesis and also to study the effects of the size of UDP datagrams on the throughput.

3.4 Does the size of datagrams matter?

In order to study the effects of different UDP datagram sizes and fragmentation resulted from different UDP datagram sizes, we did the following test. The shell script used for the tests are packed together with this document in the folder "Iperf_SizeTest".

- Fix the targeted throughput at 3Mbps which is larger than the maximum bandwidth.
- Repeat the throughput test with varying UDP datagram sizes. Start with 50 bytes, increase the UDP datagram size with a step size of 10 bytes.
- Plot the achieved throughput versus the UDP datagram size.

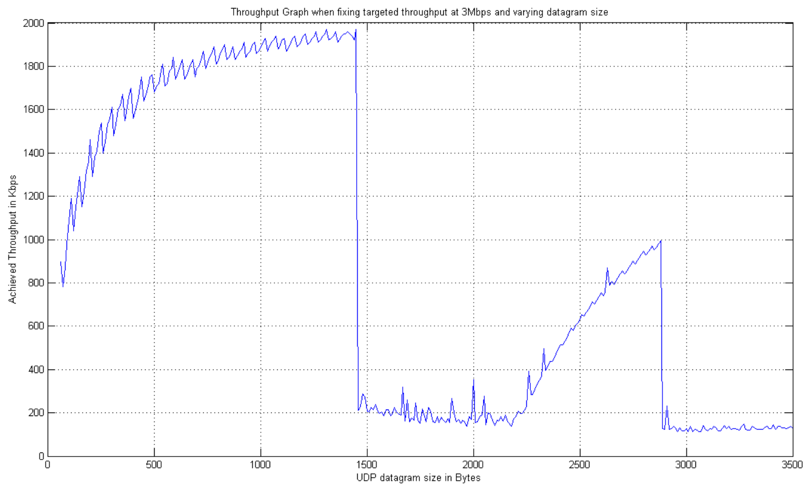


Figure 3.5: Throughput v.s UDP datagram size

Figure 3.5 shows the achieved throughput versus the varying UDP datagram size. Before the UDP datagram size reaches 1453 bytes where fragmentation first happens, the achieved throughput keeps increasing since the relative overhead imposed by protocol headers decreases. The achieved throughput suffers the first drastic drop after UDP datagram size reaches the first fragmentation threshold 1453 bytes. After this, we see that the achieved throughput starts to climb up again as the UDP datagram size increases. This is because as the size of the UDP datagram becomes larger, for the fixed targeted throughput 3Mbps, the spacing between datagrams increases accordingly. This gives the buffer at the line terminal device more time to breathe and thus increases the probability that two consecutive fragments get through. When hitting the second threshold for fragmentation, 2888 bytes after which another fragmentation occurs, the throughput achieved suffers from another drastic drop. This test result verifies our hypothesis that fragmentation is the reason for the drastic throughput drop. Also the throughput behavior for datagram sizes between 50 bytes and 1452 bytes indicates that in order to better utilize the bandwidth we should reduce the relative overhead by sending datagrams as large as possible while avoiding fragmentation. And the optimum choice is 1452 bytes.

3.5 Effects of fragmentation in theory

We have seen that fragmentation deteriorates the throughput since having two fragments increase the probability for the original datagram to be dropped. But exactly by how much should it affect the throughput? In order to analyze the effects of fragmentation, we first define the following quantities and concepts beforehand. And we here assume that there are only two fragments.

- B: in Kbps, the maximum bandwidth offered by the twisted pair cable. Here we assume it to be the largest achievable bandwidth without loss of data.
- b: in Kbps, the required bandwidth by the application. This is the rate that the application sends the data.
- p: the probability that a packet is dropped. We assume that the line terminal device is using dropping tail and each packet will be equally probable to be dropped with the probability p when required bandwidth is larger than maximum bandwidth. Thus $p = (b - B)/b$.
- Type I packet: The first fragment that corresponds to the original UDP datagram. Usually the first fragment is larger compared to the second one.

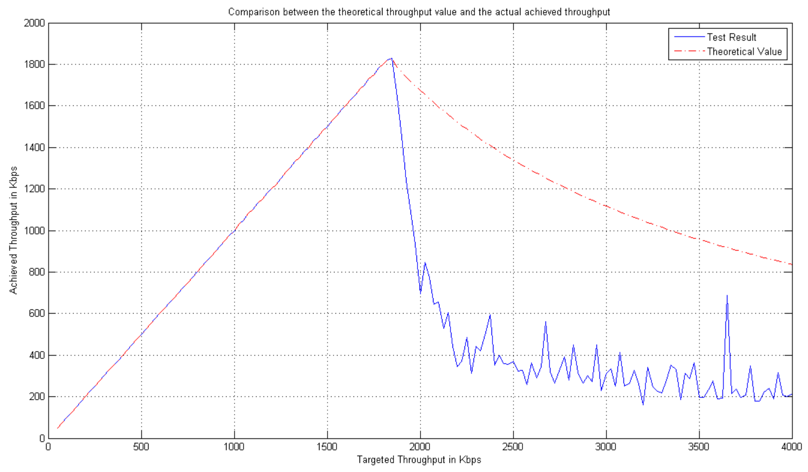


Figure 3.6: Comparison between theoretical value and the test results

- Type II packet: The second fragment that corresponds to the original UDP datagram.

In order to count the original UDP datagram as successfully received, both Type I and Type II packets shouldn't be lost. Assuming that event of dropping of packet is independent, the probability for a UDP datagram to be transmitted successfully is given by $(1-p)^2 = (1-(b-B)/b)^2 = B^2/b^2$. Multiply this probability with the required bandwidth b, we get the theoretical throughput seen by Iperf considering the effects of fragmentation: $b{\cdot}B^2/b^2 = B^2/b$.

We compare the theoretical value above with the actual test results. The comparison is shown in Figure 3.6. The actual throughput achieved is much lower than the theoretical value. This indicates that we've missed some other factors in the picture which compromises the throughput. We've noticed that on receiver end, there's a large difference in number of Type I packets and Type II packets that are successfully received. This urges us to further explore more essential reasons for throughput performance decrease: factors that affects the probability for a packet to be dropped. We'll explore this topic in the next chapter.

3.6 Conclusion on UDP datagram size

On one hand, the application should by all means avoid fragmentation. There are two reasons for this. First, fragmentation will deteriorate the application throughput in congestion. We've verified this both in theory and by tests. And in tests, combined with other factors we'll discuss in later chapters that affects the throughput, we get even worse throughput performance. Second, fragmentation introduces extra overhead. Taking 1470 bytes UDP datagram for example, fragmentation introduces an extra overhead of 70 bytes (8 bytes UDP header + 8 bytes fragmentation header + 40 bytes IPv6 header + 14 bytes ethernet header). This is one reason why the maximum achievable throughput reduces to 1.83Mbps whereas when UDP datagram is 1452 bytes, this value is 1.97Mbps.

On the other hand, it is advised that application sends the UDP datagrams of the size larger than 500 bytes otherwise too much bandwidth will be wasted on overhead. The optimum size is 1452 bytes. So if possible, PMUs could consider packing several measurement data in one UDP datagram in order to reduce overhead.

Chapter 4

Sending Pattern Affects Packet Dropping Probability

4.1 Objective and motivation

We have seen that fragmentation deteriorates throughput but the actual achieved throughput is much lower than the value predicted by simplified model we came up with. The reason is that our assumption that the event that a packet is dropped is independent and identically distributed. However, the trace at receiver's end shows that there's a large difference in the number of Type I packets and Type II packets that get through which breaks our assumption. This urges us to investigate a more essential problem for throughput performance: What are the factors that affect the probability for a packet to be dropped?

4.2 Defining two spacing patterns

We first rewind back to traffic captured at sender side for a moment, we notice that each Type II packet is sent almost immediately after Type I packet where the spacing is of several μs. And each Type I packet is sent after the Type II packet with a spacing of several ms. We thus propose another hypothesis that the packet spacing pattern is a factor that affects the probability for a packet to be dropped. We hereby define two spacing patterns that we would like to study.

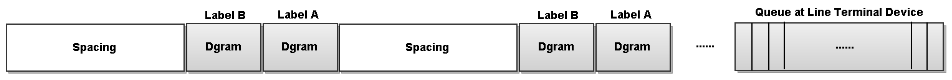


Figure 4.1: Spacing Pattern I

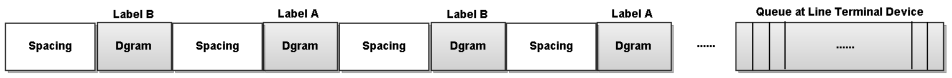


Figure 4.2: Spacing Pattern II

4.2.1 Spacing Pattern I

The spacing pattern I resembles the case when there's fragmentation. Datagrams are sent in blocks of two consecutive ones. Spacings are inserted between blocks. We label the first one in the block as Datagram A and label the second one in the block as Datagram B. The length of spacing can be changed accordingly in order to achieve certain targeted throughput. Spacing pattern I is illustrated in 4.1.

4.2.2 Spacing Pattern II

Spacing pattern II distributes Datagram A and Datagram B evenly in time span. Notice that, given the same Datagram A and Datagram B, in order to achieve the same targeted throughput, the length of the spacing in pattern II should be half of that used in spacing pattern I. The spacing pattern II is illustrated in 4.2.

4.3 Equal sized datagrams with different spacing patterns

In order to study the effects of different spacing patterns, we choose Datagram A and Datagram B to be equal sized and target at same throughput/sending rate. The test programs, source codes of the programs and relevant test results are packed with the document in the folder "Sending_Patterns". The tests are run in the following way.

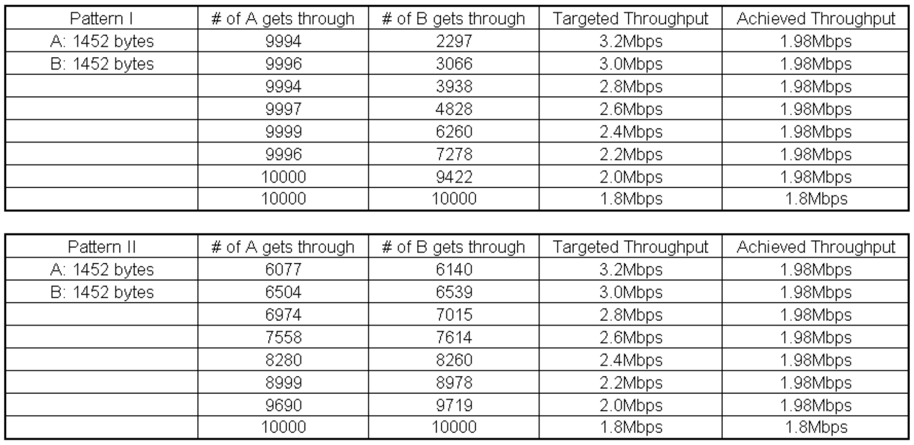
- We run the DropQueueServer test program on the receiver's side with the command: #DropQueueServer
- We run the DropQueueClient test program on the sender's side with

the command: #DropQueueClient $server_ip $dgram_1 $dgram_2 $sending_pattern $targeted_throughput

- $dgram_1 and $dgram_2 are size of Datagram A and Datagram B in bytes, $sending_pattern takes value of 0 or 1 where 0 indicates spacing pattern 1 and 1 indicates spacing pattern 2, $targeted_throughput is the sending rate in Kbps

The DropQueueClient sends 10000 Datagram A and 10000 Datagram B at the specified sending rate with specified spacing pattern. And DropQueueClient will record the achieved throughput and number of Datagram A and Datagram B received in order to get a sense of the packet dropping probability.

Figure 4.3 shows the recorded results for 1452 bytes datagram A and 1452 bytes datagram B with different spacing patterns. With spacing pattern I, Datagram A suffers almost no loss while loss happens mostly to Datagram B. As the targeted throughput decreases, the number of Datagram B that gets through increases. With spacing pattern II, the numbers of Datagram A and Datagram B that get through are more or less the same which indicates that their dropping probability is almost the same. This result illustrates the effect of spacing patterns on the packet dropping probability. The reason for this is that the line terminal device is using dropping tail queueing discipline and when the targeted throughput is higher than the maximum achievable throughput, the queue is almost full all the time. Spacing pattern I is sending a burst of two UDP datagrams, whenever there's a place in the queue available, only the first datagram in the burst gets through. And two places in the queue clearing up when a burst of datagrams arrives is rare case but becomes more likely when targeted throughput decreases which means the spacing length increases. This explains in spacing pattern I, why as targeted throughput decreases the number of Datagram B that gets through increases. As for spacing pattern II, Datagram A and Datagram B are equal sized and evenly distributed in time span, thus are equivalent. From the line terminal device's perspective, there's no difference between Datagram A and Datagram B. So the dropping probability for the two are the same. Generally speaking, the behavior we observed in spacing pattern II is more desirable. So it is recommended that we avoid sending datagrams in a way similar to spacing pattern I.



Pattern I	# of A gets through	# of B gets through	Targeted Throughput	Achieved Throughput
A: 1452 bytes	9994	2297	3.2Mbps	1.98Mbps
B: 1452 bytes	9996	3066	3.0Mbps	1.98Mbps
	9994	3938	2.8Mbps	1.98Mbps
	9997	4828	2.6Mbps	1.98Mbps
	9999	6260	2.4Mbps	1.98Mbps
	9996	7278	2.2Mbps	1.98Mbps
	10000	9422	2.0Mbps	1.98Mbps
	10000	10000	1.8Mbps	1.8Mbps

Pattern II	# of A gets through	# of B gets through	Targeted Throughput	Achieved Throughput
A: 1452 bytes	6077	6140	3.2Mbps	1.98Mbps
B: 1452 bytes	6504	6539	3.0Mbps	1.98Mbps
	6974	7015	2.8Mbps	1.98Mbps
	7558	7614	2.6Mbps	1.98Mbps
	8280	8260	2.4Mbps	1.98Mbps
	8999	8978	2.2Mbps	1.98Mbps
	9690	9719	2.0Mbps	1.98Mbps
	10000	10000	1.8Mbps	1.8Mbps

Figure 4.3: Test results for equal sized datagrams with different spacing patterns

4.4 Unequal sized datagrams with different sending patterns

In order to explore whether the size of datagrams also plays a role, we repeated the tests in Section 4.3 by changing the length of Datagram B to 726 bytes and 363 bytes. Figure 4.4 and Figure 4.5 shows the results for 726 bytes and 363 bytes respectively.

Introducing the size difference doesn't change the story a lot. Datagram A still has a higher probability to get through compared to Datagram B. This is because the spacing length is of several ms while with 100Mbps ethernet link, transmission of a 1452 bytes UDP datagram (including overhead the length is 1514 bytes) only takes $0.12112ms$. So the length of the spacing is the dominating factor here. Thus it is almost always true that with spacing pattern I, the first one in the burst will have higher probability to get through. In order to verify this, we also did the test with Datagram A of size 736 bytes and Datagram B of size 1452 bytes. Figure 4.6 shows that though we changed the size of Datagram A to be the smaller one, it still maintains a higher probability to get through.

To summarize, the size of datagrams also affects packet dropping probability. However, with an incoming link of 100Mbps, the transmission time on the wire of datagrams is much smaller than the spacing. So spacing is the dominating factor that affects the packet dropping probability. When using spacing pattern I, we can claim that in general the first packet in the burst will have a higher probability to get through.

Pattern I	# of A gets through	# of B gets through	Targeted Throughput	Achieved Throughput
A: 1452 bytes	9121	432	3.2Mbps	1.98Mbps
B: 726 bytes	9374	724	3.0Mbps	1.98Mbps
	9817	1450	2.8Mbps	1.98Mbps
	9994	2489	2.6Mbps	1.98Mbps
	9998	4408	2.4Mbps	1.98Mbps
	9998	5776	2.2Mbps	1.98Mbps
	9998	8196	2.0Mbps	1.98Mbps
	10000	10000	1.8Mbps	1.8Mbps

Pattern II	# of A gets through	# of B gets through	Targeted Throughput	Achieved Throughput
A: 1452 bytes	5837	5613	3.2Mbps	1.98Mbps
B: 726 bytes	6381	6119	3.0Mbps	1.98Mbps
	6687	6684	2.8Mbps	1.98Mbps
	7136	7413	2.6Mbps	1.98Mbps
	7754	7984	2.4Mbps	1.98Mbps
	8444	8576	2.2Mbps	1.98Mbps
	9425	9460	2.0Mbps	1.98Mbps
	10000	10000	1.8Mbps	1.8Mbps

Figure 4.4: Changing Datagram B to 726 bytes

Pattern I	# of A gets through	# of B gets through	Targeted Throughput	Achieved Throughput
A: 1452 bytes	7703	302	3.2Mbps	1.98Mbps
B: 363 bytes	8326	321	3.0Mbps	1.98Mbps
	8745	518	2.8Mbps	1.98Mbps
	9020	540	2.6Mbps	1.98Mbps
	9590	1532	2.4Mbps	1.98Mbps
	9995	2784	2.2Mbps	1.98Mbps
	9996	8259	2.0Mbps	1.98Mbps
	10000	10000	1.8Mbps	1.8Mbps

Pattern II	# of A gets through	# of B gets through	Targeted Throughput	Achieved Throughput
A: 1452 bytes	5660	5432	3.2Mbps	1.98Mbps
B: 363 bytes	6373	6232	3.0Mbps	1.98Mbps
	6834	6634	2.8Mbps	1.98Mbps
	7360	7207	2.6Mbps	1.98Mbps
	7768	7631	2.4Mbps	1.98Mbps
	8286	8235	2.2Mbps	1.98Mbps
	9610	9605	2.0Mbps	1.98Mbps
	10000	10000	1.8Mbps	1.8Mbps

Figure 4.5: Changing Datagram B to 363 bytes

Pattern I, 726, 1452	# of A gets through	# of B gets through	Targeted Throughput	Achieved Throughput
A: 726 bytes	9550	4247	3.2Mbps	1.98Mbps
B: 1452 bytes	9654	4600	3.0Mbps	1.98Mbps
	9902	5321	2.8Mbps	1.98Mbps
	9997	5989	2.6Mbps	1.98Mbps
	9998	6959	2.4Mbps	1.98Mbps
	9997	8481	2.2Mbps	1.98Mbps
	10000	9035	2.0Mbps	1.98Mbps
	10000	10000	1.8Mbps	1.8Mbps

Figure 4.6: Datagram A 736 bytes, Datagram B 1452 bytes

4.5 Revisit throughput drop caused by fragmentation

Now we can revisit the drastic throughput drop issue caused by fragmentation. We have analyzed that fragmentation itself by cutting original datagram into two pieces will deteriorates throughput performance since losing either fragment means losing the original datagram. Another issue resides within fragmentation is that whenever fragmentation happens we'll be sending packets using spacing pattern I since two fragments are always sent in a burst. The throughput performance becomes worse since the probability that the first fragment gets through is generally higher than the second fragment. This completes the story of why fragmentation causes such a drastic throughput drop.

4.6 Chapter summary

This chapter presents the tests and analysis about how the spacing pattern, the size of UDP datagrams, the order of datagrams affect the packet dropping probability. We defined two spacing patterns and tested the packet dropping behavior associated with the two spacing patterns. We found out that given our network configuration, among several factors that affect the packet dropping probability, the spacing is the dominating one. In general, when sending using spacing pattern I, the first one in the burst will have a higher probability to get through while sending using space pattern II, both Datagram A and Datagram B will have almost same dropping probability.

We have seen that by avoiding fragmentation, we can always get the maximum achievable throughput. However, this chapter's tests and analysis show light on another perspective of the performance consideration: in order to achieve a uniform packet dropping probability distribution and avoid consecutive datagram losses, PMUs are advised to evenly distribute the datagrams in the time span.

Another point that might worth further exploring is that for the implementation of fragmentation at IP layer, whether inserting a spacing between fragments rather than sending them in a burst will improve the throughput performance in general.

Chapter 5

Maximum Burst Size

5.1 Objective and motivation

Though PMUs are expected to be sending data using UDP at the frequency of 50Hz, there is still a possibility that due to unexpected reasons a burst of data will be generated by PMUs. In this chapter, we tested and estimated the maximum burst size in the number of UDP datagrams that the ZyXel line terminal device can support without losing any data. Given this number, we come up with a safe region where PMUs should limit its burst size within so as to avoid congestion.

5.2 How do the tests work?

Figure 5.1 illustrates how the tests work. The MaxBurstServer application runs on the receiver side and listens on port 4950 for incoming UDP datagrams. The MaxBurstClient application runs on the sender side and listens on port 5050 for incoming UDP datagrams. For each test round, sender first sends a start signal to receiver to notify the start of a new round of test. Then sender sends a burst of n datagrams without spacing between datagrams. Each datagram has a sequence number. After finish sending the burst, sender sends an end signal with the information of the current burst size to indicate the end of the round and waits for the feedback from the receiver. Receiver keeps a counter of received datagrams and calculates the number of datagrams lost in the burst and send this information back to sender in the feedback. Another round of test begins after a pause of 5 seconds and the burst size increase by one to $n + 1$.

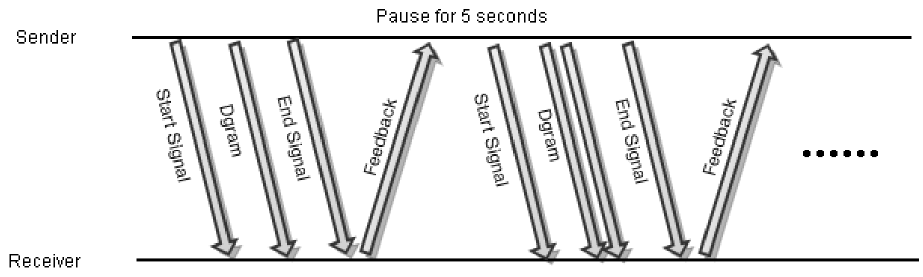


Figure 5.1: Test program flow illustration

The starting value of the burst size and the rounds of tests to run are controlled by input arguments for MaxBurstClient. The size of the datagrams can also be chosen by changing the input arguments for MaxBurstClient. The source codes of MaxBurstServer and MaxBurstClient are packed with the documents. Followings are how to use the test programs.

- Run the MaxBurstServer test program on the receiver's side with the command: #MaxBurstServer
- Run the MaxBurstClient test program on the sender's side with the command: #MaxBurstClient $server_ip $dgram_size $start_burst_size $rounds_to_run
- $dgram_size is size of the UDP datagrams to be sent in bytes, $start_burst_size is the burst size in the first round, $rounds_to_run is the number of rounds of tests to run

5.3 Results and analysis

We ran the maximum burst size tests for different UDP datagram sizes, 50 bytes, 500 bytes, 1000 bytes and 1452 bytes respectively. Each test starts from a burst size of one and runs for 100 rounds until the current burst size reaches 100 datagrams.

Figure 5.2 shows the number of UDP datagrams lost with respect to the burst size. And the results show that ZyXEL line terminal device uses a packet-based queue since the number of UDP datagrams lost in a burst doesn't depend on the size of the UDP datagrams. For cases of 50 bytes,

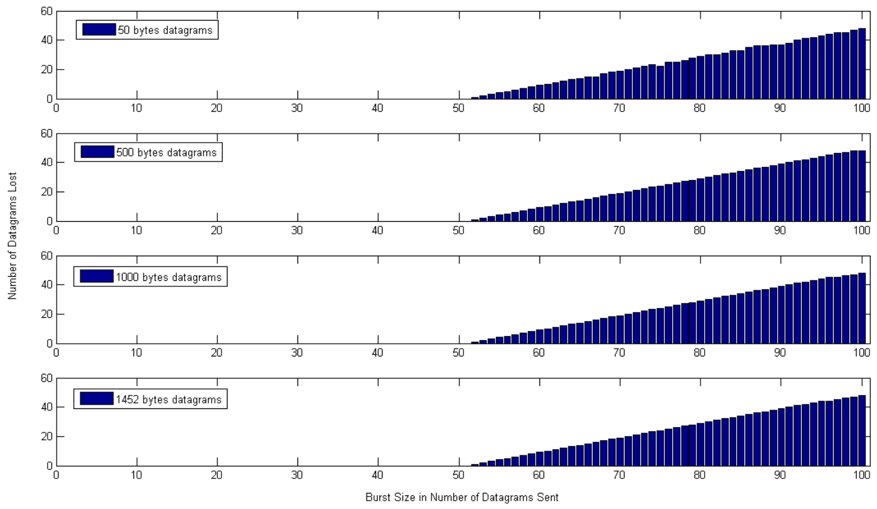


Figure 5.2: Datagram loss v.s Burst Size

500 bytes, 1000 bytes, and 1452 bytes, all the first UDP datagram loss happens when the burst size hits 52.

Furthermore, by observing the sequence number of the datagrams received at the receiver's end, we confirmed that the ZyXel line terminal device is employing dropping tail queueing discipline since all the datagrams after the 51th one were lost in the burst.

As indicated in the design specification document, at each measurement site, two PMUs will be connected to the ZyXel line terminal device. We verified by testing that the line terminal device is able to support two parallel bursty streams with burst size of 25. Thus we claim that when the total average sending rate of the two PMUs are below the maximum bandwidth, as long as both PMUs limit their burst size under 25 datagrams, there won't be any loss of datagrams.

5.4 Chapter Summary

This chapter focuses on the maximum burst size that the ZyXEL line terminal device could support without losing any data. Test result shows that with an empty queue, the ZyXEL line terminal device could in maximum handle a burst of 51 datagrams. Also the test result confirms that ZyXEL line terminal device is using dropping tail queueing discipline.

Since two PMUs are connected to one ZyXEL line terminal device at each measurement site, and the expected required throughput is well below the maximum achievable throughput. The queue will be empty for the most of the time. Hence, as long as the PMUs limit their burst size under 25 UDP datagrams, we shouldn't expect any data loss.

Chapter 6

Summary and Key Findings

The document described procedural details on setting up the point-to-point connection test environment using SHDSL and ZyXEL line terminal device and presented the basic performance test results such as round trip time and throughput performance both of which indicate our design specification meets the network delay and throughput requirement. Based on the test environment we set up, we designed and implemented several tests to investigate the factors that jeopardize the throughput performance, factors that affects the packet dropping probability and the maximum burst size the ZyXEL line terminal device could support. The key findings could be summarized as follows:

- The round trip time between CM building and DESL lab is around 3.410ms and with our current network configuration the maximum achievable throughput when sending 1452 bytes UDP datagrams is 1.97Mbps (with a precision of 25Kbps). And these results indicate that our design specifications meet the network delay and throughput requirements.

- Fragmentation happens when UDP datagram size exceeds 1452 bytes and deteriorates the throughput performance.

- The ZyXEL line terminal device uses dropping tail queueing discipline. We discovered that there are several factors that affect the packet dropping probability, such as spacing pattern, size of the datagrams, order of the packets.

- When the queue of ZyXEL line terminal device is empty and with an incoming link of 100Mbps and outgoing link of around 2Mbps, the maximum number of datagrams in a burst that the device could handle without losing any data is 51.

Based on our findings and analysis of the test results, we give the following advices on how the PMUs could make the best use of the bandwidth in the communication infrastructure we've set up:

- PMUs should by all means avoid fragmentation, i.e. sending datagrams larger than 1452 bytes.
- PMUs are also advised to send datagrams of sizes in the interval between 500 bytes and 1452 bytes since sending smaller datagrams would waste too much bandwidth on overheads. The optimum choice is 1452 bytes.
- In order to achieve an uniform packet dropping probability distribution and avoid consecutive datagram losses, PMUs should evenly distribute the datagrams in the time span.
- For each of the two PMUs attached to the ZyXEL line terminal device, they are advised to limit their burst size under 25 UDP datagrams.

Acknowledgements

I would like to acknowledge and extend my heartfelt gratitude to Professor Jean-Yves Le Boudec for the constructive criticism, stimulating suggestions and invaluable guidance he has offered during the course of the project, and Miroslav Popovic for the support and inspiration he extended, the time and energy he spent on discussing and revising the report with me.