Project 5 : Computer Networks

Kriti Joshi : 13358

November 14, 2016

1 Traffic Measurement

1.1

The average packet size comes out to be : 768.180860115 bytes. The calculation was done using the formula

$$\frac{Total\_Bytes}{Total\_Packets}$$

where $Total\_Bytes$ and $Total\_Packets$ are the total bytes and total packets sent across all transactions.

1.2

Main features:

Flow Duration The graph has certain kinks *i.e.* points where graph drops or rises abruptly and is smooth for majority of points. This can be seen in the graph 2 where the graph abruptly drops indicating a large number of entries with high flow duration.

Number of Bytes Graph 3 the sudden rise in graph near origin indicates the presence of maximum entries with low(around $40 - 80$ Bytes) data size. This is shown clearly in 4 where the graph attains value 1 below 100 Bytes and then uniformly decreases to zero indicating almost equal distribution of larger data size entries.

Number of Packets Similar to graph 3, graph 5 also shows the hike on going towards origin as usually less number of packets are sent during networking. This can also be seen in graph 6 where the slope is maximum near low packet values indicating sudden rise. Large number of packets are generally avoided, this is clear from the very sudden drop in the end in graph 6

These features are desirable because we try to optimize the process of sending data so that the transfer is fast(ensured by large packet numbers or data size) and the data loss incurred by dropping of a packet or congestion can be minimum. Hence an optimum size is kept for proper networking

Logarithmic graphs are desirable for better understanding of the scenarios where there is a sudden hike or drop. In linear scale, the values shoot up (slope becomes large) and the data near such points cannot be studied properly. In logarithmic scale, the details get revealed. This is apparent from the graphs 3 and 4 where no details can be inferred by seeing the linear scale graph.

1.2.1 Flow Duration

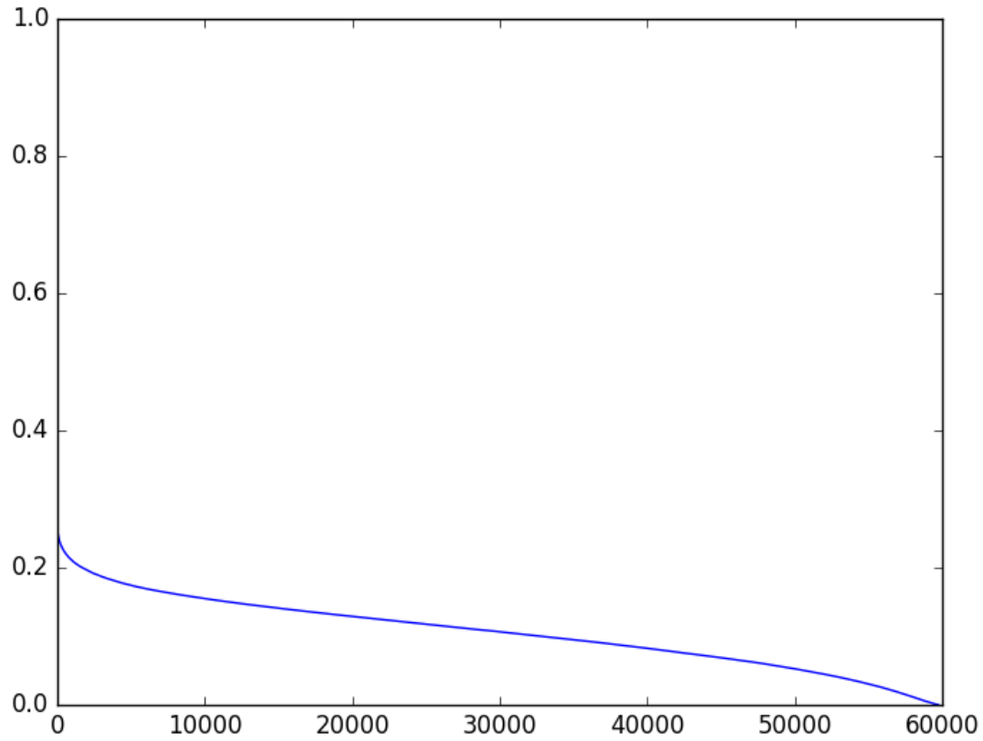

Figure 1: Flow Duration Linear Scale Graph

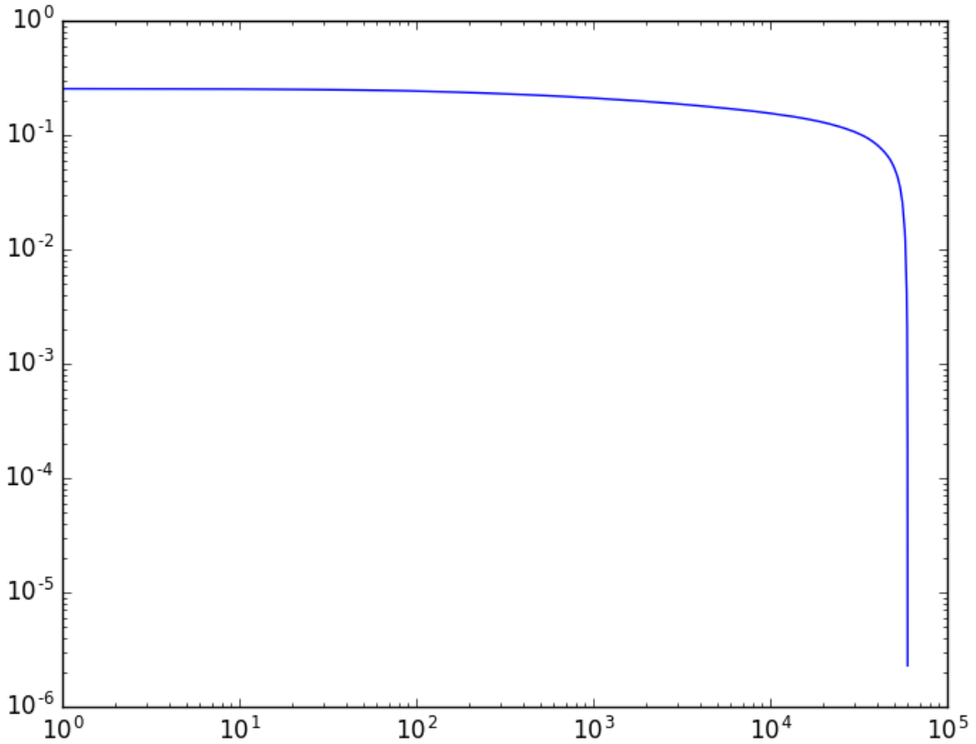

Figure 2: Flow Duration Logarithmic Scale Graph

1.2.2 Number of Bytes

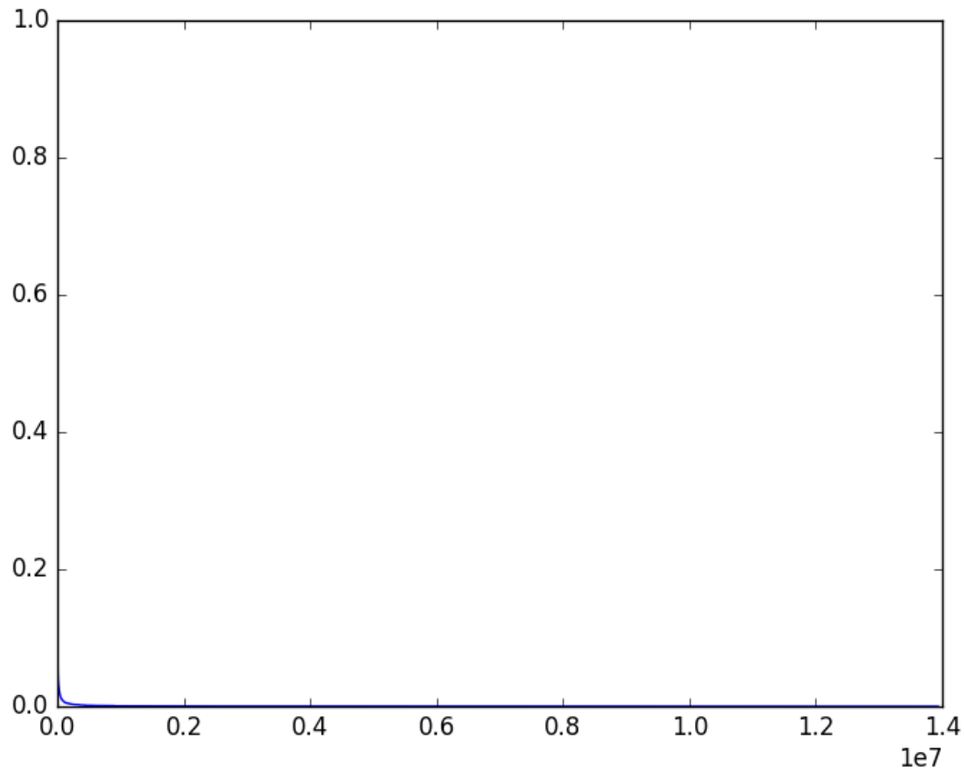

Figure 3: Number of Bytes Linear Scale Graph

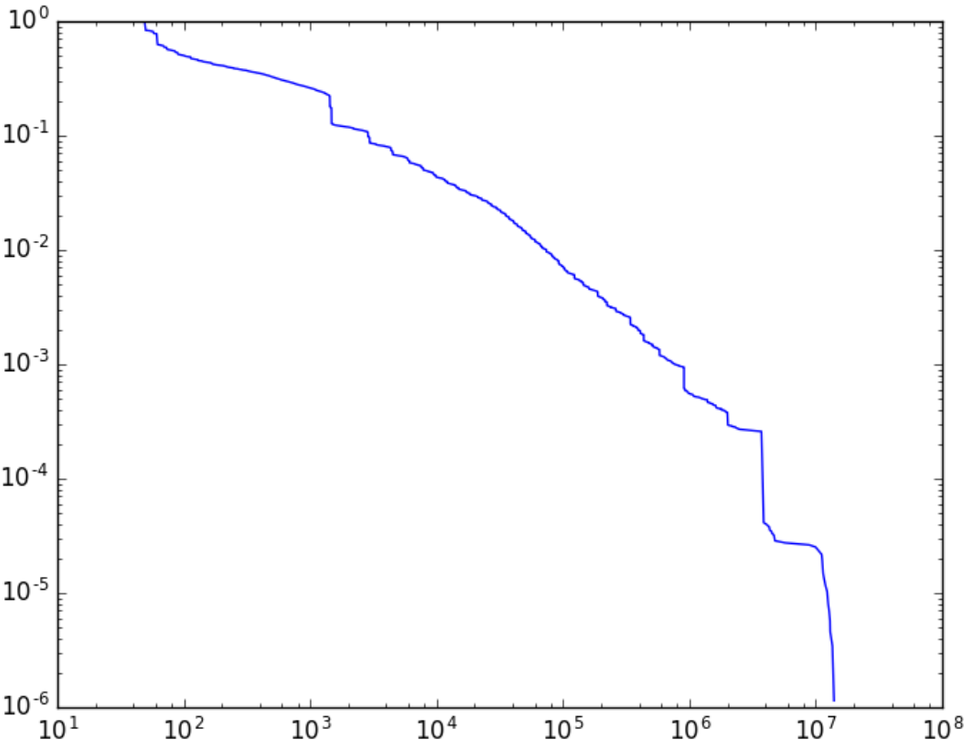

Figure 4: Number of Bytes Logarithmic Scale Graph

1.2.3 Number of Packets

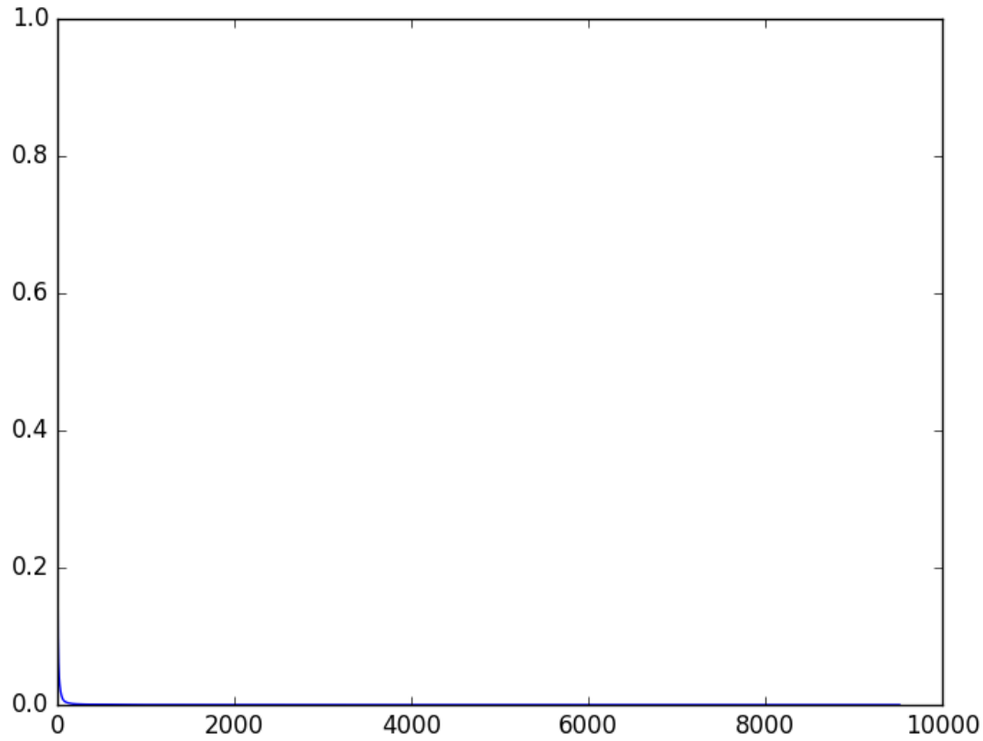

Figure 5: Number of packets Linear Scale Graph

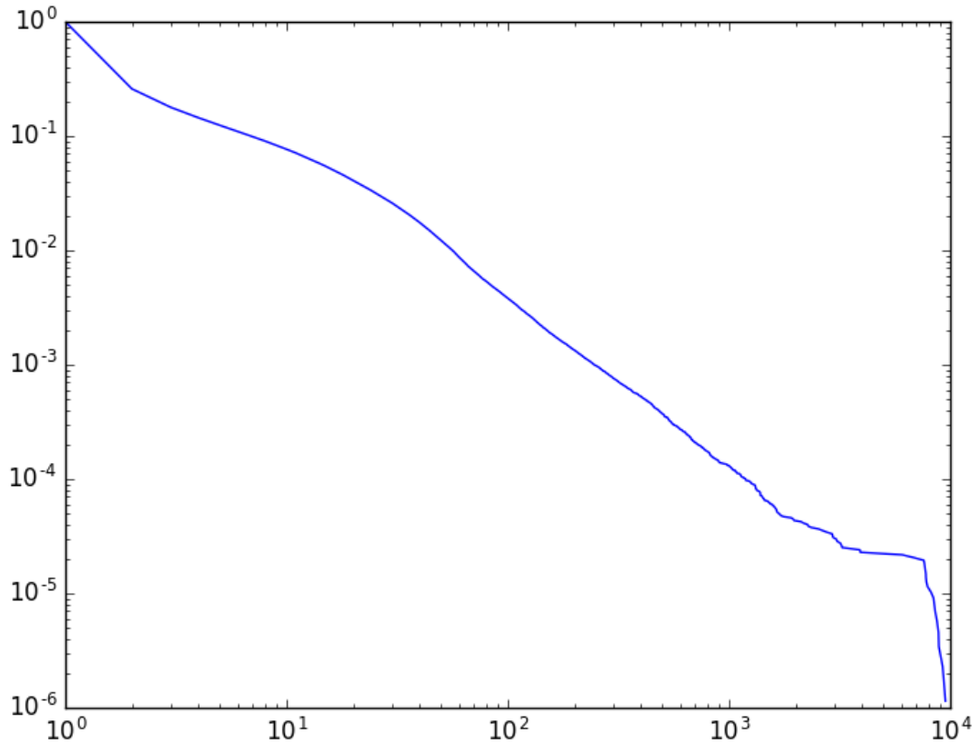

Figure 6: Number of packets Logarithmic Scale Graph

What are the main features of the graphs? What artifacts of Netflow

1.3

Overall top 10% sender ports account to **63.840%** of total traffic while top 10 receiver ports account to 14.203%. Port 80 contributes a large percentage as sender as well as receiving port. It is associated with HTTP applications. As the internet is very much popular and the most used source, the popularity of port 80 is obvious. Port 1935 is the port for Macromedia Flash Communications Server MX, which is used in Flash Player whose popularity can also be accredited to high use of internet. Other popular ports are for SSH(22), HTTPS(443) etc. Among the senders, a few ports cover majority of traffic unlike the receiver's end.

Sender IP Port	% Traffic Contributed
80	43.939
33001	7.362
1935	3.664
22	2.168
443	1.726
55000	1.623
388	1.339
16402	0.762
20	0.672
873	0.584

Table 1: Top 10 sender traffic volume ports

Receiver IP Port	% Traffic Contributed
33002	4.025
80	2.927
49385	2.092
62269	1.229
443	0.774
43132	0.763
16402	0.743
22	0.648
5500	0.648
57493	0.354

Table 2: Top 10 receiver traffic volume ports

1.4

The fraction of total traffic with source mask 0 is 43.26

Percentage	Number of Prefixes	% Traffic Contribution
0.1	4	59.327
1	34	79.771
10	338	97.311

Table 3: Without removing 0 mask

Percentage	Number of Prefixes	% Traffic Contribution
0.1	3	28.317
1	33	64.349
10	337	95.260

Table 4: After removing 0 mask

Where possible, explain what applications are likely responsible for this traffic. (See the IANA port numbers reference for details.) Explain any significant differences between the results for sender vs. receiver port numbers.

1.5

By Bytes	Fraction of traffic sent by Princeton	0.007009
By Bytes	Fraction of traffic sent to Princeton	0.021915
By Packets	Fraction of traffic sent by Princeton	0.01014
By Packets	Fraction of traffic sent to Princeton	0.014684

2 BGP Measurement

2.1

The results were computed by counting the number of IPV4 updates across all update files belonging to a particular date between (12pm to 2pm)

Date	Average Updates per Minute
January 3, 2014	623.667
February 3, 2014	1020.208
March 3, 2014	1959.050

2.2

Date	Fraction of prefixes with 0 updates
January 3, 2014	0.989
February 3, 2014	0.9649
March 3, 2014	0.9068

2.3

Date	Maximum Update Frequency(per minute)	Max Frequency Prefixes
January 3, 2014	8.5	121.52.150.0/24 121.52.145.0/24 121.52.144.0/24 121.52.149.0/24
February 3, 2014	6.125	85.239.28.0/24
March 3, 2014	6.067	109.161.64.0/20

2.4

Date	Percentage	Number of Unstable Prefixes	% Traffic Contribution
January 3, 2014	0.1	5	0.062
	1	57	0.2022
	10	578	0.4769
February 3, 2014	0.1	17	0.0663
	1	176	0.1960
	10	1760	0.5505
March 3, 2014	0.1	46	0.0500
	1	462	0.1197
	10	4626	0.3826

2.5

Summary:

Briefly summarize your results and what you learned about BGP stability from them.

Updates Per Minute: This was calculated using the corresponding update logs for all three days. The number of entries for IPV4 only were counted and reported after diving by 120 (2 hrs) . The updates almost doubled from January to February and again from February to March indicating many faults in the network (system failures) forcing BGP to restructure paths.

No update: The prefixes from the RIB files were entered in a dictionary and then the updates files were considered. The ones not having any entries were counted and reported after dividing by total different prefixes in the RIB. The output is pretty high indicating only a few percentage of prefixes get updated.

Maximum update frequency: A frequency count for each prefix in the updates file was considered and the max was chosen amongst them. The frequency keeps on decreasing in the coming months indicating more stability.

Unstable Prefixes: Overall contribution of the unstable prefixes keep on decreasing in the following months. Again indicating greater stability.

Appendices

A Traffic Measurement

```
import matplotlib.pyplot as plt
from numpy import inf
from math import ceil
from math import log
import numpy as np

f = open("ft-v05.2010-09-29.235501+0000.csv","r")
data = f.read()
f.close()
# Ignore first line
lines = data.split("\n")
heading = lines[0].split(":")
attributes = heading[1].split(",")
size = len(attributes)
num_packets = 0
sum_packet_size = 0
sender = {}
receiver = {}
prefixes = {}
mask_map = {0:0,1:128,2:192,3:224,4:240,5:248,6:252,7:254,8:255}
total_prefixes = 0
num_lines = len(lines)
princeton_incoming_trafiic_size = 0
princeton_outgoing_trafiic_size = 0
princeton_incoming_trafiic_packet = 0
princeton_outgoing_trafiic_packet = 0
packets_num_log = {}
data_size_log = {}
flow_duration_log = {}
for i in range(1,num_lines):
        line = lines[i]
        if len(line) == 0:
                break;
        list = line.split(",")
        size = len(list)
        entry = {}
        for j in range(0,size):
                entry[attributes[j]] = list[j]
        # print entry
        # Q1.1

        sum_packet_size += int(entry['doctets'])
        num_packets += int(entry['dpkts'])

        flow_duration = int(entry['last']) - int(entry['first'])
        if flow_duration in flow_duration_log:
                flow_duration_log[flow_duration] += 1
        else:
                flow_duration_log[flow_duration] = 1

        if int(entry['dpkts']) in packets_num_log:
                packets_num_log[int(entry['dpkts'])] += 1
        else:
                packets_num_log[int(entry['dpkts'])] = 1

        if int(entry['doctets']) in data_size_log:
                data_size_log[int(entry['doctets'])] += 1
        else:
                data_size_log[int(entry['doctets'])] = 1

        # Q1.3

        if int(entry['prot']) == 6 or int(entry['prot']) == 17:
                sender_port = entry['srcport']
                if sender_port in sender:
                        sender[sender_port] += int(entry['doctets'])
                else:
```

```
                        sender[sender_port] = int(entry['doctets'])

                receiver_port = entry['dstport']
                if receiver_port in receiver:
                        receiver[receiver_port] += int(entry['doctets'])
                else:
                        receiver[receiver_port] = int(entry['doctets'])

        src_prefix = entry['srcaddr'].split(".")
        netmask = int(entry['src_mask'])
        prefix = ""
        # print entry['srcaddr'] + " : " + entry['src_mask']

        for k in range(0,4):
                if netmask>8:
                        src_prefix[k] = int(src_prefix[k]) & mask_map[8]
                else:
                        src_prefix[k] = int(src_prefix[k]) & mask_map[max(0,
                            netmask)]
                netmask -= 8
                prefix += str(src_prefix[k])
                if k!=3:
                        prefix += "."

        # Case 2:
        prefix += "/"
        prefix += entry['src_mask']

        if prefix in prefixes:
                prefixes[prefix] += int(entry['doctets'])
        else:
                prefixes[prefix] = int(entry['doctets'])
                total_prefixes += 1

        # print prefix
        # print src_prefix

        src_prefix_2 = entry['srcaddr'].split(".")
        dst_prefix_2 = entry['dstaddr'].split(".")
        if int(src_prefix_2[0]) == 128 and int(src_prefix_2[1]) == 112:
                princeton_outgoing_trafiic_size += int(entry['doctets'])
                princeton_outgoing_trafiic_packet += int(entry['dpkts'])
        if int(dst_prefix_2[0]) == 128 and int(dst_prefix_2[1]) == 112:
                princeton_incoming_trafiic_size += int(entry['doctets'])
                princeton_incoming_trafiic_packet += int(entry['dpkts'])

print "Total Packets : " + str(num_packets)
                                                        # 3879877
print "Average Packet Size : " + str(float(sum_packet_size)/num_packets)
                            # Average Packet Size : 768.180860115

busiest_sender_ports = sorted(sender, key=sender.__getitem__,reverse=True)
    [:10]
print "Bussiest Sender Ports:"
print busiest_sender_ports

busiest_receiver_ports = sorted(receiver, key=receiver.__getitem__,reverse=
    True)[:10]
print "Bussiest Receiver Ports:"
print busiest_receiver_ports

outgoing_traffic = {}
incoming_traffic = {}

print "Percentage traffic utilisation by individual sender ports"
sum_outgoing_traffic = 0
for i in range(0,10):
        print busiest_sender_ports[i] + " : " + str(float(sender[
            busiest_sender_ports[i]])/sum_packet_size*100) + "%"
        sum_outgoing_traffic += float(sender[busiest_sender_ports[i]])/
            sum_packet_size
```

```
print "Total outgoing traffic by top 10 sender ports : " + str(
    sum_outgoing_traffic*100)

print "Percentage traffic utilisation by individual receiver ports"
sum_incoming_traffic = 0
for i in range(0,10):
        print busiest_receiver_ports[i] + " : " + str(float(receiver[
            busiest_receiver_ports[i]])/sum_packet_size*100) + "%"
        sum_incoming_traffic += float(receiver[busiest_receiver_ports[i]])/
            sum_packet_size

print "Total incoming traffic by top 10 receiver ports : " + str(
    sum_incoming_traffic*100)

print "Total different prefixes : " + str(total_prefixes)
print "With 0 mask analysis"
popular_prefixes = sorted(prefixes, key=prefixes.__getitem__,reverse=True)[:
    int(ceil(0.1*total_prefixes+1))]
print popular_prefixes[:int(ceil(0.001*total_prefixes))]
sum_top = 0
for i in range(0,int(ceil(0.001*total_prefixes))):
        sum_top += prefixes[popular_prefixes[i]]
print "0.1% : " + str(int(ceil(0.001*total_prefixes))) + " : " + str(float(
    sum_top)/sum_packet_size*100) + "%"

sum_top = 0
for i in range(0,int(ceil(0.01*total_prefixes))):
        sum_top += prefixes[popular_prefixes[i]]
print "1% : " + str(int(ceil(0.01*total_prefixes))) + " : " + str(float(
    sum_top)/sum_packet_size*100) + "%"

sum_top = 0
for i in range(0,int(ceil(0.1*total_prefixes))):
        sum_top += prefixes[popular_prefixes[i]]
print "10% : " + str(int(ceil(0.1*total_prefixes))) + " : " + str(float(
    sum_top)/sum_packet_size*100) + "%"

print "Source mask 0 traffic : " + str(float(prefixes['0.0.0.0/0'])/
    sum_packet_size*100) + "%"
print "Without 0 mask analysis"
sum_top = 0
for i in range(1,int((0.001*total_prefixes))+1):
        sum_top += prefixes[popular_prefixes[i]]
print "0.1% : " + str(int((0.001*total_prefixes))) + " : " + str(float(sum_top
    )/(sum_packet_size-prefixes['0.0.0.0/0'])*100) + "%"

sum_top = 0
for i in range(1,int((0.01*total_prefixes))+1):
        sum_top += prefixes[popular_prefixes[i]]
print "1% : " + str(int((0.01*total_prefixes))) + " : " + str(float(sum_top)/(
    sum_packet_size-prefixes['0.0.0.0/0'])*100) + "%"

sum_top = 0
for i in range(1,int((0.1*total_prefixes))+1):
        sum_top += prefixes[popular_prefixes[i]]
print "10% : " + str(int((0.1*total_prefixes))) + " : " + str(float(sum_top)/(
    sum_packet_size-prefixes['0.0.0.0/0'])*100) + "%"

print "Princeton outgoing traffic (By Size): " + str(float(
    princeton_outgoing_trafiic_size)/sum_packet_size*100) + "%"
print "Princeton incoming traffic (By Size): " + str(float(
    princeton_incoming_trafiic_size)/sum_packet_size*100) + "%"
print "Princeton outgoing traffic (By Packets): " + str(float(
    princeton_outgoing_trafiic_packet)/num_packets*100) + "%"
print "Princeton incoming traffic (By Packets): " + str(float(
    princeton_incoming_trafiic_packet)/num_packets*100) + "%"

# Flow duration
x_axis = np.array(sorted(flow_duration_log))
y_axis = np.array(flow_duration_log.values())
ccdf = np.cumsum(y_axis[::-1])[::-1]
ccdf = ccdf * 1.0
```

```
ccdf /= ccdf[0]
plt.xscale('linear')
plt.yscale('linear')
plt.plot(x_axis,ccdf,'b')
plt.show()
plt.xscale('log')
plt.yscale('log')
plt.plot(x_axis,ccdf,'b')
plt.show()

# Packet number
x_axis = np.array(sorted(packets_num_log))
y_axis = np.array(packets_num_log.values())
ccdf = np.cumsum(y_axis[::-1])[::-1]
ccdf = ccdf * 1.0
ccdf /= ccdf[0]
plt.xscale('linear')
plt.yscale('linear')
plt.plot(x_axis,ccdf,'b')
plt.show()
plt.xscale('log')
plt.yscale('log')
plt.plot(x_axis,ccdf,'b')
plt.show()

# Data Size
x_axis = np.array(sorted(data_size_log))
y_axis = np.array(data_size_log.values())
ccdf = np.cumsum(y_axis[::-1])[::-1]
ccdf = ccdf * 1.0
ccdf /= ccdf[0]
plt.xscale('linear')
plt.yscale('linear')
plt.plot(x_axis,ccdf,'b')
plt.show()
plt.xscale('log')
plt.yscale('log')
plt.plot(x_axis,ccdf,'b')
plt.show()
```

B BGP Measurement

B.1

```
import os
import numpy as np

def isIPV4(address):
        bit = address.split(":")
        length = len(bit)
        if length == 1:
                return True
        else:
                return False

count = [0,0,0]

path = "./updates-asst4/"
for root, dirs, files in os.walk(path):
        for name in files:
                f = open(os.path.join(path,name))
                data = f.read()
                f.close()
                name_part = name.split(".")
                if name_part[1] == "20140103":
                        dt = 0
                elif name_part[1] == "20140203":
                        dt = 1
                else:
                        dt = 2
                lines = data.split("\n")
                for line in lines:
                        if len(line) != 0:
                                words = line.split("|")
                                if isIPV4(words[5]):
                                        count[dt] += 1.0

count = np.array(count)
count /= 120
print count
```

B.2

```
import os
import numpy as np

def isIPV4(address):
        bit = address.split(":")
        length = len(bit)
        if length == 1:
                return True
        else:
                return False
mask_map = {0:0,1:128,2:192,3:224,4:240,5:248,6:252,7:254,8:255}
count = [0,0,0]

prefixes = [{},{},{}]

path_rib = "./rib/"
for root, dirs, files in os.walk(path_rib):
        for name in files:
                f = open(os.path.join(path_rib,name))
                data = f.read()
                f.close()
                name_part = name.split(".")
                if name_part[1] == "20140103":
                        dt = 0
                elif name_part[1] == "20140203":
                        dt = 1
                else:
                        dt = 2
                lines = data.split("\n")
                for line in lines:
                        if len(line) != 0:
                                words = line.split("|")
                                if isIPV4(words[5]):
                                        src_prefix = words[5].split("/")[0].
                                            split(".")
                                        netmask = int(words[5].split("/")[1])
                                        prefix = ""

                                        for k in range(0,4):
                                                if netmask>8:
                                                        src_prefix[k] = int(
                                                            src_prefix[k]) &
                                                            mask_map[8]
                                                else:
                                                        src_prefix[k] = int(
                                                            src_prefix[k]) &
                                                            mask_map[max(0,
                                                            netmask)]
                                                netmask -= 8
                                                prefix += str(src_prefix[k])
                                                if k!=3:
                                                        prefix += "."

                                        prefix += "/"
                                        prefix += words[5].split("/")[1]
                                        prefixes[dt][prefix] = 1

print "Read rib"

path_updates = "./updates-asst4/"
for root, dirs, files in os.walk(path_updates):
        for name in files:
                f = open(os.path.join(path_updates,name))
                data = f.read()
                f.close()
                name_part = name.split(".")
                if name_part[1] == "20140103":
                        dt = 0
                elif name_part[1] == "20140203":
                        dt = 1
```

```
                else:
                        dt = 2
                lines = data.split("\n")
                for line in lines:
                        if len(line) != 0:
                                words = line.split("|")
                                if len(words)<=5:
                                        print line
                                if isIPV4(words[5]):
                                        src_prefix = words[5].split("/")[0].
                                            split(".")
                                        netmask = int(words[5].split("/")[1])
                                        prefix = ""

                                        for k in range(0,4):
                                                if netmask>8:
                                                        src_prefix[k] = int(
                                                            src_prefix[k]) &
                                                            mask_map[8]
                                                else:
                                                        src_prefix[k] = int(
                                                            src_prefix[k]) &
                                                            mask_map[max(0,
                                                            netmask)]
                                                netmask -= 8
                                                prefix += str(src_prefix[k])
                                                if k!=3:
                                                        prefix += "."

                                        prefix += "/"
                                        prefix += words[5].split("/")[1]
                                        if prefix in prefixes[dt]:
                                                prefixes[dt][prefix] += 1
                                        else:
                                                prefixes[dt][prefix] = 1

print "Read updates"
count = [0,0,0]

for i in range(0,3):
        print "Total prefixes : " + str(len(prefixes[i]))
        for p in prefixes[i]:
                if prefixes[i][p] == 1:
                        count[i] += 1.0
        count[i] /= len(prefixes[i])

print count
```

B.3

```
import os
import numpy as np

def isIPV4(address):
        bit = address.split(":")
        length = len(bit)
        if length == 1:
                return True
        else:
                return False
mask_map = {0:0,1:128,2:192,3:224,4:240,5:248,6:252,7:254,8:255}
count = [0,0,0]
max_freq = [0,0,0]
prefixes = [{},{},{}]

path_updates = "./updates-asst4/"
for root, dirs, files in os.walk(path_updates):
        for name in files:
                f = open(os.path.join(path_updates,name))
                data = f.read()
                f.close()
                name_part = name.split(".")
                if name_part[1] == "20140103":
                        dt = 0
                elif name_part[1] == "20140203":
                        dt = 1
                else:
                        dt = 2
                lines = data.split("\n")
                for line in lines:
                        if len(line) != 0:
                                words = line.split("|")
                                if isIPV4(words[5]):
                                        src_prefix = words[5].split("/")[0].
                                            split(".")
                                        netmask = int(words[5].split("/")[1])
                                        prefix = ""

                                        for k in range(0,4):
                                                if netmask>8:
                                                        src_prefix[k] = int(
                                                            src_prefix[k]) &
                                                            mask_map[8]
                                                else:
                                                        src_prefix[k] = int(
                                                            src_prefix[k]) &
                                                            mask_map[max(0,
                                                            netmask)]
                                                netmask -= 8
                                                prefix += str(src_prefix[k])
                                                if k!=3:
                                                        prefix += "."

                                        prefix += "/"
                                        prefix += words[5].split("/")[1]
                                        if prefix in prefixes[dt]:
                                                prefixes[dt][prefix] += 1
                                        else:
                                                prefixes[dt][prefix] = 1
                                        max_freq[dt] = max(max_freq[dt],
                                            prefixes[dt][prefix])

print "Read updates"
count = [0,0,0]
max_freq_list = [[],[],[]]

for i in range(0,3):
        print "Total prefixes : " + str(len(prefixes[i]))
        for p in prefixes[i]:
                if prefixes[i][p] == max_freq[i]:
```

```
                max_freq_list[i].append(p)

print max_freq_list
print max_freq
```

B.4

```
import os
import numpy as np

def isIPV4(address):
        bit = address.split(":")
        length = len(bit)
        if length == 1:
                return True
        else:
                return False
mask_map = {0:0,1:128,2:192,3:224,4:240,5:248,6:252,7:254,8:255}
count_lines = [0,0,0]
max_freq = [0,0,0]
prefixes = [{},{},{}]

path_updates = "./updates-asst4/"
for root, dirs, files in os.walk(path_updates):
        for name in files:
                f = open(os.path.join(path_updates,name))
                data = f.read()
                f.close()
                name_part = name.split(".")
                if name_part[1] == "20140103":
                        dt = 0
                elif name_part[1] == "20140203":
                        dt = 1
                else:
                        dt = 2
                lines = data.split("\n")
                for line in lines:
                        if len(line) != 0:
                                words = line.split("|")
                                if isIPV4(words[5]):
                                        count_lines[dt] += 1
                                        src_prefix = words[5].split("/")[0].
                                            split(".")
                                        netmask = int(words[5].split("/")[1])
                                        prefix = ""

                                        for k in range(0,4):
                                                if netmask>8:
                                                        src_prefix[k] = int(
                                                            src_prefix[k]) &
                                                            mask_map[8]
                                                else:
                                                        src_prefix[k] = int(
                                                            src_prefix[k]) &
                                                            mask_map[max(0,
                                                            netmask)]
                                                netmask -= 8
                                                prefix += str(src_prefix[k])
                                                if k!=3:
                                                        prefix += "."

                                        prefix += "/"
                                        prefix += words[5].split("/")[1]
                                        if prefix in prefixes[dt]:
                                                prefixes[dt][prefix] += 1
                                        else:
                                                prefixes[dt][prefix] = 1

print "Read updates"
count = [0,0,0]

for i in range(0,3):
        total_prefixes = len(prefixes[i])
        top_10 = int(0.1*total_prefixes)
        top_1 = int(0.01*total_prefixes)
        top_01 = int(0.001*total_prefixes)
```

```
popular_prefixes = sorted(prefixes[i], key=prefixes[i].__getitem__,
    reverse=True)[:top_10]
sum_10 = 0
for j in range(0,top_10):
        sum_10 += prefixes[i][popular_prefixes[j]]
sum_1 = 0

for j in range(0,top_1):
        sum_1 += prefixes[i][popular_prefixes[j]]
sum_01 = 0
for j in range(0,top_01):
        sum_01 += prefixes[i][popular_prefixes[j]]

print "Top most unstabel 0.1% prefixes : " + str(top_01)
print "Total traffic % by 0.1: " + str(float(sum_01)/count_lines[i])
print "Top most unstabel 1% prefixes : " + str(top_1)
print "Total traffic % by 1%: " + str(float(sum_1)/count_lines[i])
print "Top most unstabel 10% prefixes : " + str(top_10)
print "Total traffic % by 10%: " + str(float(sum_10)/count_lines[i])
```