Internet Measurements

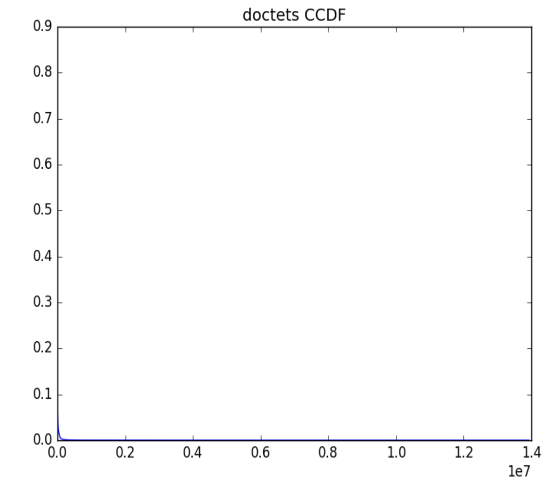
Manikanta Reddy D (13265)

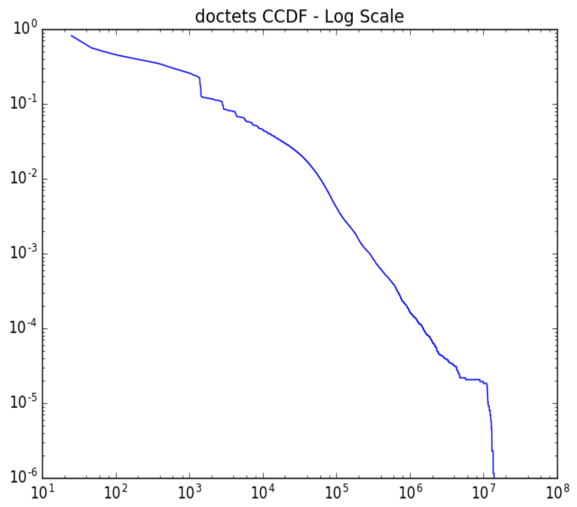
1. Traffic Measurements

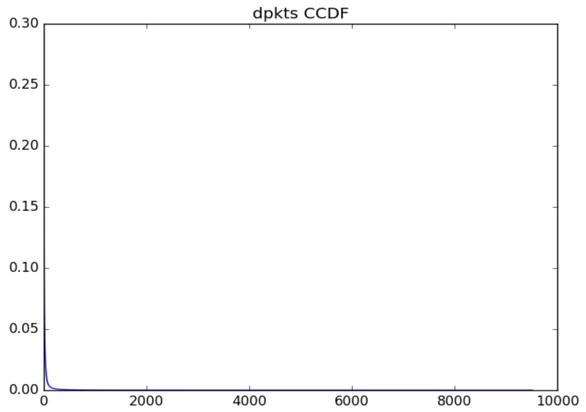
1.1. Average packet size

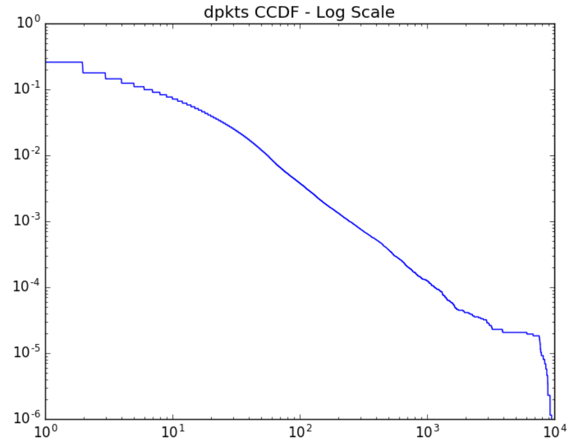
768.180860115 bytes/packet
Method: Compute sum of column 'doctets': doctets_sum
Compute sum of column 'dpkts' : dpkts_sum
Average = doctet_sum/dpkts_sum

1.2. CCDF plots





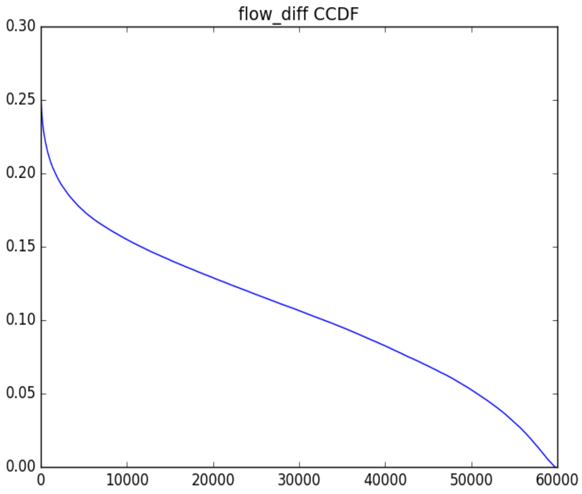


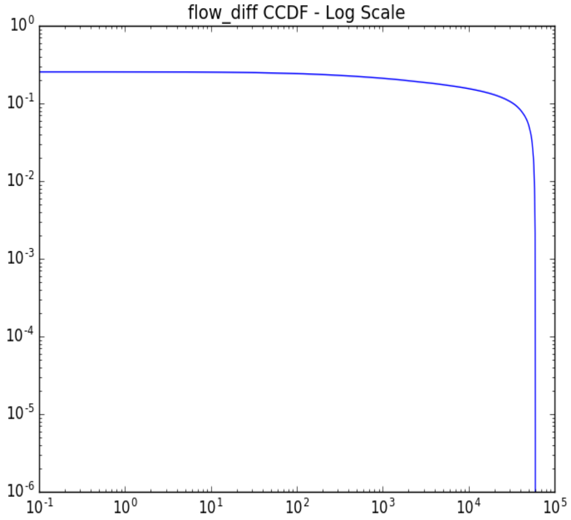


As we can see the linear plots show that the data is skewly distributed. A projection on the log scale elevates the notable details.

Most of the values in dpkts and doctets both die down around 1000 counts, whereas data values are largely spread toward 100000 counts. Since maximum data points are located towards the lower end we can explode that lower range by plotting a log scale plot.

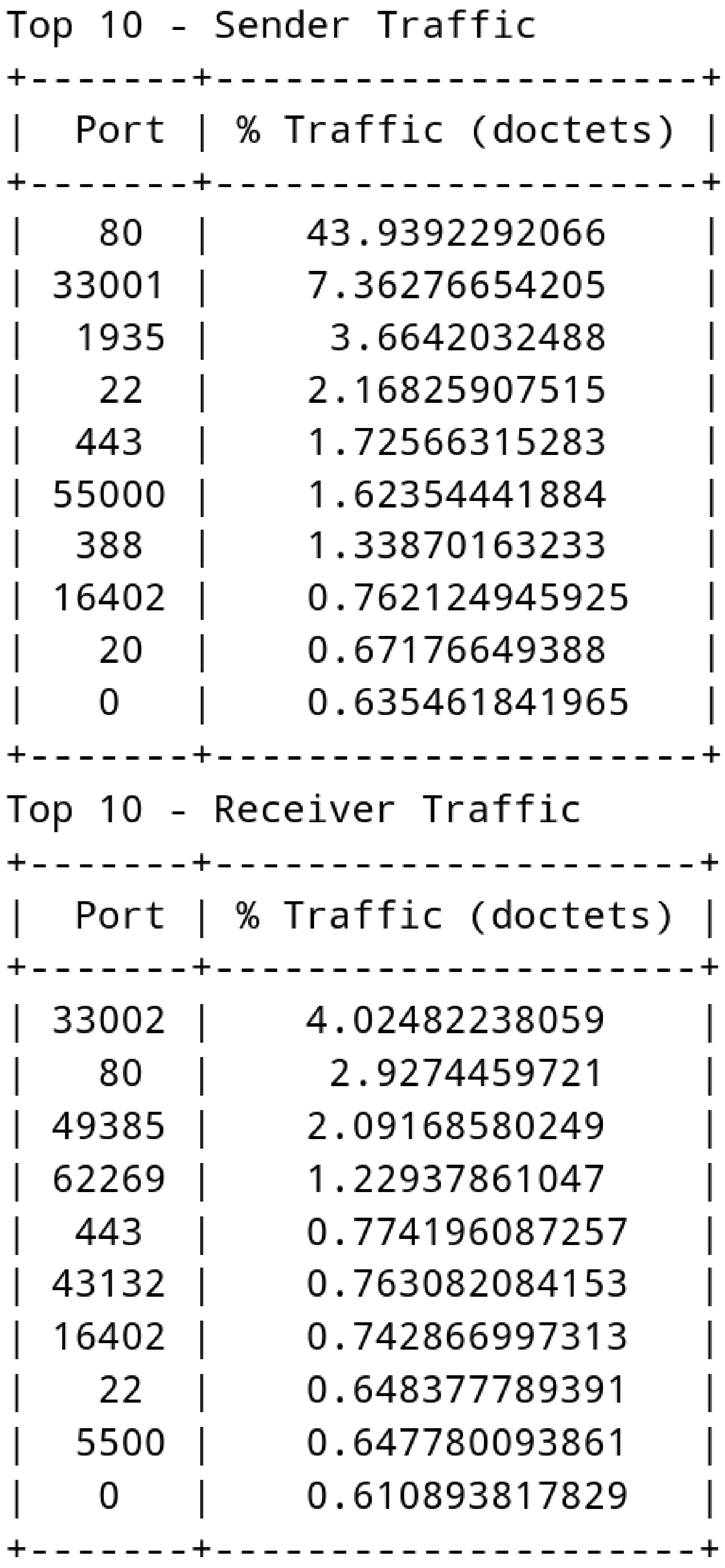
A sudden drop towards the end in log scale plots shows us that the packets are hitting a minimum threshold. That could be attributed to the minimum packet limits.


flow_diff CCDF


flow_diff CCDF - Log Scale

Plots of flow diff show us that many packets have near 0 difference in their flow. This fact is over exaggerated in the log scale, by the sudden drop.

1.3 Port Traffic Summary

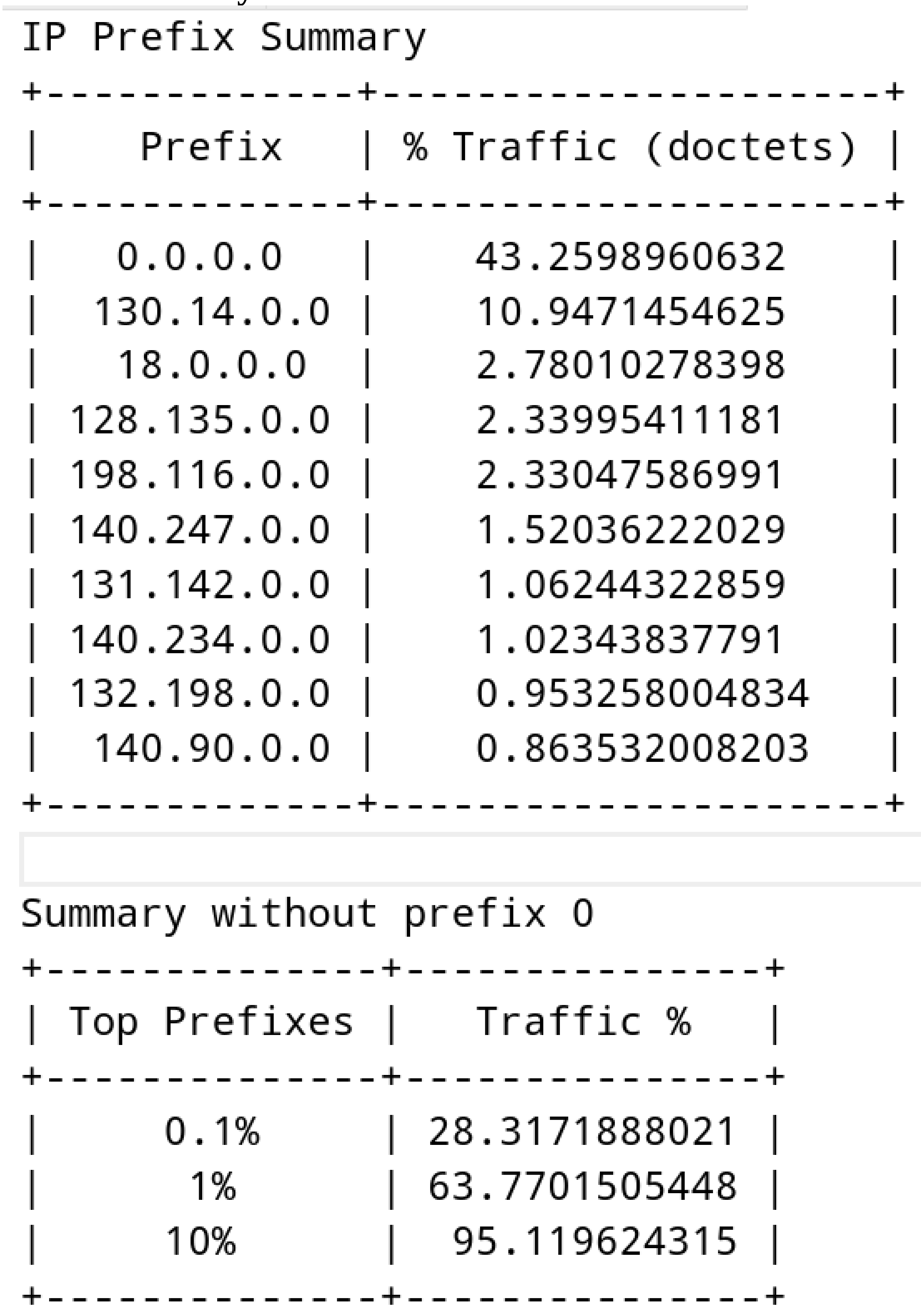


```
Top 10 - Sender Traffic
+-------+---------------------+
|  Port | % Traffic (doctets) |
+-------+---------------------+
|   80  |    43.9392292066    |
| 33001 |    7.36276654205    |
|  1935 |     3.6642032488    |
|   22  |    2.16825907515    |
|  443  |    1.72566315283    |
| 55000 |    1.62354441884    |
|  388  |    1.33870163233    |
| 16402 |    0.762124945925   |
|   20  |    0.67176649388    |
|   0   |    0.635461841965   |
+-------+---------------------+
Top 10 - Receiver Traffic
+-------+---------------------+
|  Port | % Traffic (doctets) |
+-------+---------------------+
| 33002 |    4.02482238059    |
|   80  |     2.9274459721    |
| 49385 |    2.09168580249    |
| 62269 |    1.22937861047    |
|  443  |    0.774196087257   |
| 43132 |    0.763082084153   |
| 16402 |    0.742866997313   |
|   22  |    0.648377789391   |
|  5500 |    0.647780093861   |
|   0   |    0.610893817829   |
+-------+---------------------+
```

This summary of port traffic gives us a few crucial bits.

1) High traffic through port 80: Attributed to the HTTP trafic
2) Port 443: HTTPS Traffic
3) Port 22: For SSH
4) The high variation between traffic in sender and receiver bytes for port 80 can be explained by a small observation that senders usually send large amount of data for requests whereas the requests generated are very small, so naturally sender side traffic is larger than receiver side traffic
5) Port 33001 and 33002 have unusually high traffic for being IANA unsigned ports. I believe that this might be because of some proxy or firewall within Princeton that might be using these two specific ports.

1.4 IP Prefix Summary



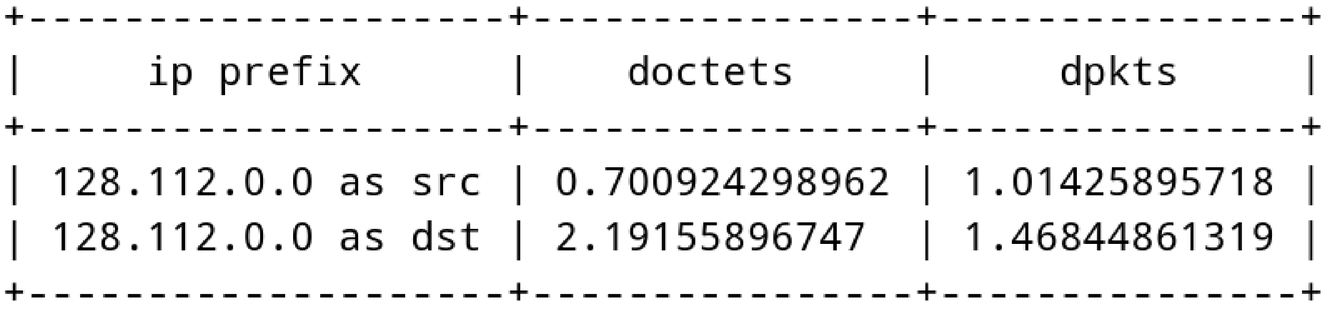
```
IP Prefix Summary
+-------------+----------------------+
|    Prefix   | % Traffic (doctets) |
+-------------+----------------------+
|   0.0.0.0   |    43.2598960632     |
|  130.14.0.0 |    10.9471454625     |
|   18.0.0.0  |    2.78010278398     |
| 128.135.0.0 |    2.33995411181     |
| 198.116.0.0 |    2.33047586991     |
| 140.247.0.0 |    1.52036222029     |
| 131.142.0.0 |    1.06244322859     |
| 140.234.0.0 |    1.02343837791     |
| 132.198.0.0 |    0.953258004834    |
|  140.90.0.0 |    0.863532008203    |
+-------------+----------------------+
```

```
Summary without prefix 0
+--------------+---------------+
| Top Prefixes |   Traffic %   |
+--------------+---------------+
|     0.1%     | 28.3171888021 |
|      1%      | 63.7701505448 |
|     10%      |  95.119624315 |
+--------------+---------------+
```

A large amount of traffic is directed towards prefix 0. We can take a look at the distribution that excludes 0 to get a better view.

Turns out just 10% of total prefixes carry 95% of the traffic. Internet is considerably sparse. Most of the prefixes don't have any traffic at all.

1.5 Princeton's Share

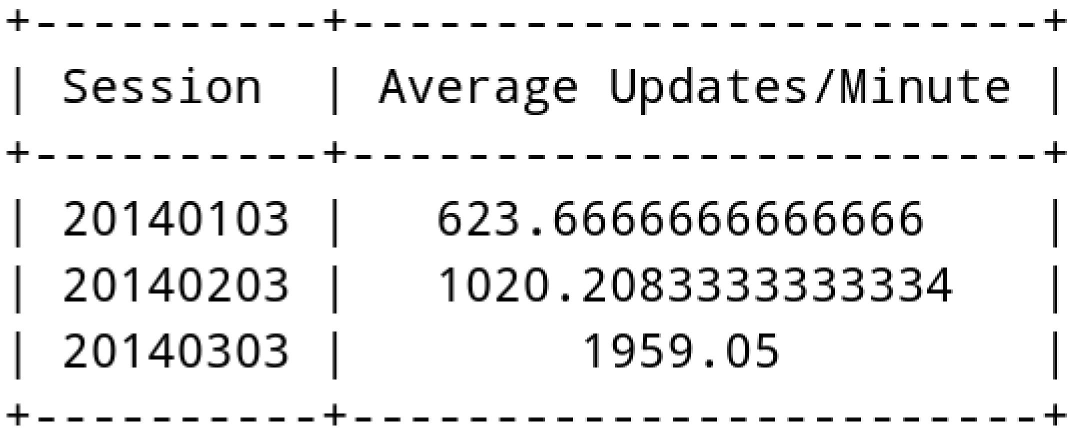


```
+---------------------+----------------+----------------+
|      ip prefix      |    doctets     |     dpkts      |
+---------------------+----------------+----------------+
| 128.112.0.0 as src | 0.700924298962 | 1.01425895718 |
| 128.112.0.0 as dst | 2.19155896747  | 1.46844861319 |
+---------------------+----------------+----------------+
```

Princeton's has a share of about 1% share either way in either measures. 1% of total internet traffic is from and to Princeton. Quite impressive!

2. BGP Measurements

2.1 Average BGP Updates



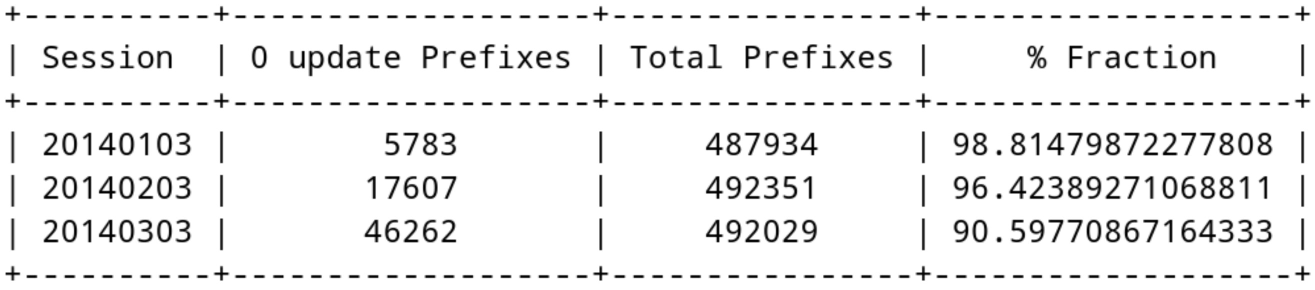
```
+----------+------------------------+
| Session  | Average Updates/Minute |
+----------+------------------------+
| 20140103 |   623.666666666666     |
| 20140203 |   1020.2083333333334   |
| 20140303 |        1959.05         |
+----------+------------------------+
```

The number of updates increased over the sessions. This might be because of the prefixes added during the previous session. This is naturally expected as the internet grows overtime.

Method of computation:

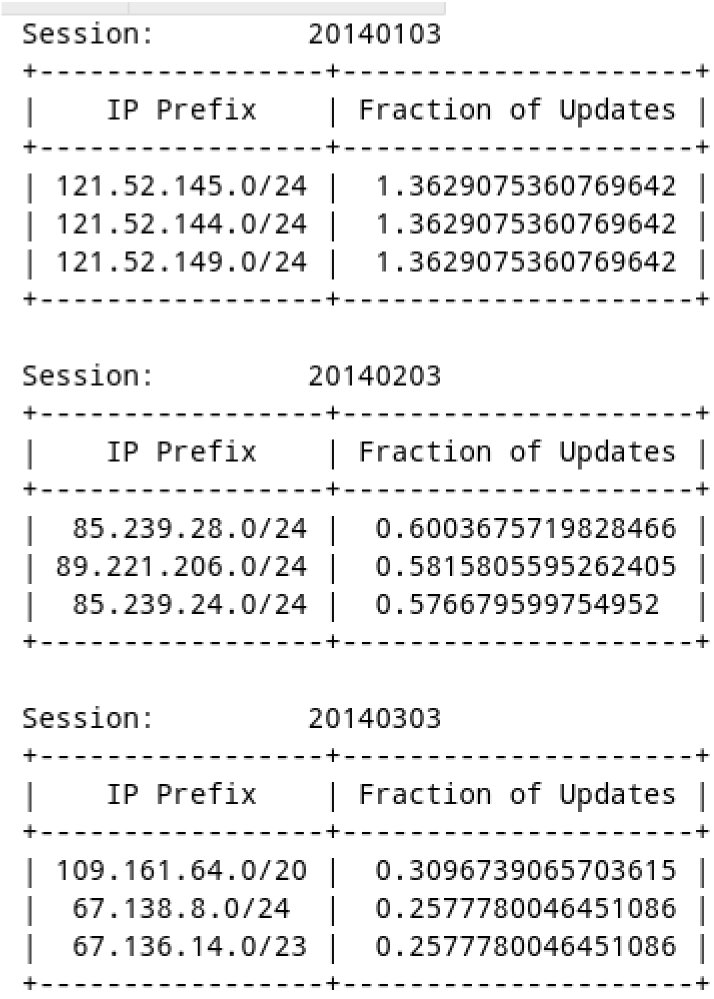
1. Read all the update files per session and count total announcements in it.
2. Every file provides us announcements for 15 minutes and there are 8 dumps per session.
3. so average updates = total updates / 15*8

2.2 & 2.3 IP Prefix fractions



```
+----------+-------------------+---------------+-------------------+
| Session  | 0 update Prefixes | Total Prefixes |     % Fraction    |
+----------+-------------------+---------------+-------------------+
| 20140103 |        5783       |     487934    | 98.81479872277808 |
| 20140203 |       17607       |     492351    | 96.42389271068811 |
| 20140303 |       46262       |     492029    | 90.59770867164333 |
+----------+-------------------+---------------+-------------------+
```

Clearly most of the prefixes in all sessions don't receive any updates, this is because most of them don't change at. We even saw in the traffic measurements how low traffic a large amount of prefixes carry.



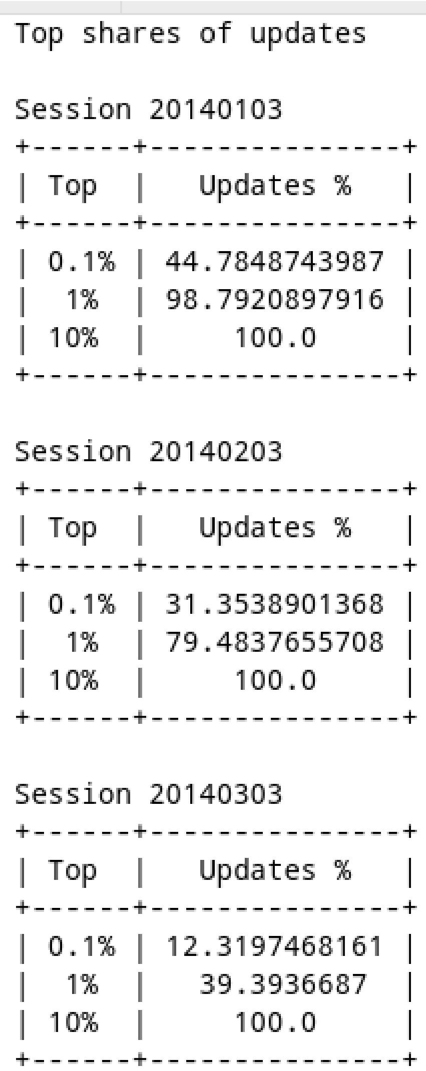
```
Session:        20140103
+----------------+--------------------+
|    IP Prefix    | Fraction of Updates |
+----------------+--------------------+
| 121.52.145.0/24 |  1.3629075360769642 |
| 121.52.144.0/24 |  1.3629075360769642 |
| 121.52.149.0/24 |  1.3629075360769642 |
+----------------+--------------------+

Session:        20140203
+----------------+--------------------+
|    IP Prefix    | Fraction of Updates |
+----------------+--------------------+
|  85.239.28.0/24 |  0.6003675719828466 |
| 89.221.206.0/24 |  0.5815805595262405 |
|  85.239.24.0/24 |  0.576679599754952  |
+----------------+--------------------+

Session:        20140303
+----------------+--------------------+
|    IP Prefix    | Fraction of Updates |
+----------------+--------------------+
| 109.161.64.0/20 |  0.3096739065703615 |
|  67.138.8.0/24  |  0.2577780046451086 |
|  67.136.14.0/23 |  0.2577780046451086 |
+----------------+--------------------+
```

We can see by this table that the top prefixes that change are usually around the same subnet.

This might be because the entire block is changing.

This detail is strengthened by the fact that the fraction of updates is nearly same within a session.

2.4 Most Unstable Prefixes



```
Top shares of updates

Session 20140103
+------+---------------+
| Top  |   Updates %   |
+------+---------------+
| 0.1% | 44.7848743987 |
|  1%  | 98.7920897916 |
| 10%  |     100.0     |
+------+---------------+

Session 20140203
+------+---------------+
| Top  |   Updates %   |
+------+---------------+
| 0.1% | 31.3538901368 |
|  1%  | 79.4837655708 |
| 10%  |     100.0     |
+------+---------------+

Session 20140303
+------+---------------+
| Top  |   Updates %   |
+------+---------------+
| 0.1% | 12.3197468161 |
|  1%  |   39.3936687  |
| 10%  |     100.0     |
+------+---------------+
```

This table is rather interesting. We can see that most of the updates are done to very unstable prefixes.

Just 10% of total prefixes garner up all updates. The rest of them never experience any considerable updates at all.

That is, if a prefix is changing a lot, most probably it'll change again. The ones that have no updates will probably have no updates at (we saw it in 2.2)

This trend being constant over all sessions strengthens it even more.

Appendix
Code:

The scripts are written in python and can be found here...

https://github.com/manikantareddyd/InternetMeasurements

Traffic Measurements Code

```
import pandas
import config
import matplotlib.pyplot as plt
import numpy as np
import ipaddress
from prettytable import PrettyTable

def get_ip_prefix(ip, mask):
    bi = ''.join([bin(int(x)+256)[3:] for x in ip.split('.')])
    n = bi[:mask] + '0'*(len(bi)-mask)
    ip_prefix = str(
        ipaddress.IPv4Address(
            '%d.%d.%d.%d' %
            (int(n[:8],2),
            int(n[8:16],2),
            int(n[16:24],2),
            int(n[24:32],2)
            )))
    return ip_prefix

def compute_src_ip_prefix(row):
    return get_ip_prefix(row['srcaddr'],row['src_mask'])

def compute_dst_ip_prefix(row):
    return get_ip_prefix(row['dstaddr'],row['dst_mask'])

class TrafficMeasurements:
    def __init__(self):
        self.dataframe = pandas.read_csv(config.FLOW_RECORD_FILE)
        self.dataframe['flow_diff'] = self.dataframe['last'] - self.dataframe['first']

    def average_packet_size(self):
        self.datadescribe = self.dataframe.describe()
        mean_dpkts = self.datadescribe['dpkts']['mean']
        mean_doctets = self.datadescribe['doctets']['mean']
        print(
            "Average packet size: ",
            mean_doctets/mean_dpkts,
            "bytes/packet"
            )

    def plot_all(self):
        self.plot_ccdf('flow_diff')
        self.plot_ccdf('dpkts')
        self.plot_ccdf('doctets')
```

```
def plot_ccdf(self, column):
    column_data = np.copy(self.dataframe[column].values)
    values, base = np.histogram(column_data,bins='auto')
    cumulative = np.cumsum(values)/len(column_data)
    fig, ax = plt.subplots()
    ax.set_title(column + ' CCDF')
    ax.plot(base[:-1], 1-cumulative)
    fig.savefig('plots/' + column + ' ccdf.png')
    fig, ax = plt.subplots()
    ax.set_xscale('log')
    ax.set_yscale('log')
    ax.set_title(column + ' CCDF' + ' - Log Scale')
    ax.plot(base[:-1], 1-cumulative)
    fig.savefig('plots/' + column + ' ccdf log.png')

def print_port_summary(self):
    data = self.dataframe[[
        'srcport',
        'dstport',
        'doctets'
        ]].copy()
    print('Top 10 - Sender Traffic')
    src_port_grouped = data.groupby('srcport').sum()
    sorted_src_port_grouped = src_port_grouped.sort_values(by='doctets',ascending=False)
    sorted_src_port_grouped['doctets'] = 100*sorted_src_port_grouped['doctets']/data['doctets'].sum()
    index = 0
    x = PrettyTable(["Port","% Traffic (doctets)"])
    for row in sorted_src_port_grouped.itertuples():
        if index >= 10: break
        x.add_row([row[0],row[2]])
        index +=1
    print(x)
    print('\n')
    print('Top 10 - Receiver Traffic')
    dst_port_grouped = data.groupby('dstport').sum()
    sorted_dst_port_grouped = dst_port_grouped.sort_values(by='doctets',ascending=False)
    sorted_dst_port_grouped['doctets'] = 100*sorted_dst_port_grouped['doctets']/data['doctets'].sum()
    index = 0
    x = PrettyTable(["Port","% Traffic (doctets)"])
    for row in sorted_dst_port_grouped.itertuples():
        if index >= 10: break
        x.add_row([row[0],row[2]])
        index +=1
    print(x)

def get_princeton_share(self):
    data = self.dataframe[['srcaddr','src_mask','dstaddr','dst_mask','doctets','dpkts']].copy()
    data['doctets'] = 100*data['doctets']/data['doctets'].sum()
    data['dpkts'] = 100*data['dpkts']/data['dpkts'].sum()
    data['src_ip_prefix'] = data.apply(compute_src_ip_prefix,axis=1)
    data['dst_ip_prefix'] = data.apply(compute_dst_ip_prefix,axis=1)
    data_src = data[['src_ip_prefix','doctets','dpkts']].copy()
    data_dst = data[['dst_ip_prefix','doctets','dpkts']].copy()
    del data
    src_prefix_group = data_src.groupby('src_ip_prefix').sum()
    dst_prefix_group = data_dst.groupby('dst_ip_prefix').sum()
    x = PrettyTable(['ip prefix','doctets','dpkts'])
    x.add_row([
        '128.112.0.0 as src',
        src_prefix_group['doctets']['128.112.0.0'],
        src_prefix_group['dpkts']['128.112.0.0']
        ])
    x.add_row([
        '128.112.0.0 as dst',
        dst_prefix_group['doctets']['128.112.0.0'],
        dst_prefix_group['dpkts']['128.112.0.0']
        ])
    print(x)

def aggregate_ip_prefix_traffic(self):
    data = self.dataframe[['srcaddr','src_mask','doctets']].copy()
    data['doctets'] = 100*data['doctets']/data['doctets'].sum()
    data['ip_prefix'] = data.apply(compute_src_ip_prefix, axis=1)
    ip_prefix_group = data.groupby('ip_prefix').sum()
    self.ip_prefix_group = ip_prefix_group.sort_values(by='doctets',ascending=False)
    print("IP Prefix Summary")
    x = PrettyTable(["Prefix","% Traffic (doctets)"])
    index = 0
    for row in self.ip_prefix_group.itertuples():
        if index >= 10: break
        x.add_row([row[0],row[2]])
        index +=1
    print(x)
    fraction_list = np.array(self.ip_prefix_group['doctets'])
    without_0 = fraction_list[1:].sum()
    _0_1 = int(len(fraction_list)*0.1/100)
    _1_0 = int(len(fraction_list)*1/100)
    _10_0 = int(len(fraction_list)*10/100)
    top_0_1 = fraction_list[1:_0_1+1].sum()
    top_0_1 = 100*top_0_1/without_0
    top_1_0 = fraction_list[1:_1_0+1].sum()
    top_1_0 = 100*top_1_0/without_0
    top_10_0 = fraction_list[1:_10_0+1].sum()
    top_10_0 = 100*top_10_0/without_0
    print("\nSummary without prefix 0")
    x = PrettyTable(["Top Prefixes","Traffic %"])
    x.add_row(["0.1%",top_0_1])
    x.add_row(["1%",top_1_0])
    x.add_row(["10%",top_10_0])
    print(x)
```

BGP Measurements Code

```
import os
import config
import csv
from prettytable import PrettyTable
import operator
import numpy as np
class BGPMeasurements:
    def __init__(self):
        self.sessions=["20140103","20140203","20140303"]

    def compute_updates(self):
        bgp_updates = {}
        bgp_files_in_updates = {}
        for session in self.sessions:
            bgp_updates[session] = 0
            bgp_files_in_updates[session] = 0
        for filename in os.listdir(config.UPDATES_DIR):
            for session in self.sessions:
                if session in filename:
                    bgp_files_in_updates[session] += 1
            with open(
                config.UPDATES_DIR + filename,
                "r") as infile:
                for line in infile:
                    fields = line.split("|")
                    if "." in fields[5]:
                        for session in self.sessions:
                            if session in filename:
                                bgp_updates[session] += 1
        x = PrettyTable([
            "Session",
            "Average Updates/Minute"
            ])
        for session in self.sessions:
            minutes = 15*bgp_files_in_updates[session]
            fraction = bgp_updates[session]/(minutes)
            x.add_row([
                session,
                fraction
                ])
        print(x)

    def compute_prefix_fractions(self):
        prefix_list = {}
        total_updates = {}

        for session in self.sessions:
            prefix_list[session] = {}
            total_updates[session] = 0

        for filename in os.listdir(config.RIB_DIR):
            with open(
                config.RIB_DIR + filename,
                "r") as infile:
                for line in infile:
```

```
                prefix = line.split("|")[5].strip()
                if "." in prefix:
                    for session in self.sessions:
                        if session in filename:
                                prefix_list[session][prefix] = 0

    for filename in os.listdir(config.UPDATES_DIR):
        with open(
            config.UPDATES_DIR + filename,
            "r") as infile:
            for line in infile:
                prefix = line.split("|")[5].strip()
                if "." in prefix:
                    for session in self.sessions:
                        if session in filename:
                            total_updates[session] += 1
                            if prefix in prefix_list[session]:
                                prefix_list[session][prefix] += 1
                            else:
                                prefix_list[session][prefix] = 1

    for session in self.sessions:
        print("\nSession:\t",session)
        x = PrettyTable([
            "IP Prefix",
            "Fraction of Updates"
            ])
        for prefix in prefix_list[session]:
            counts = prefix_list[session][prefix]
            total  = total_updates[session]
            prefix_list[session][prefix] = 100*counts/total
        for row in sorted(
            prefix_list[session].items(),
            key=operator.itemgetter(1),
            reverse=True
            )[:3]:
            x.add_row(row)
        print(x)
    self.prefix_list = prefix_list

def compute_no_update_fraction(self):
    self.compute_prefix_fractions()
    x = PrettyTable([
        "Session",
        "0 update Prefixes",
        "Total Prefixes",
        "% Fraction"
        ])
    for session in self.sessions:
        zero_count = 0
        for prefix in self.prefix_list[session]:
            if self.prefix_list[session][prefix] != 0:
                zero_count += 1
        total_prefixes = len(self.prefix_list[session])
        fraction =100 - 100*zero_count/total_prefixes
        x.add_row([
            session,
            zero_count,
                total_prefixes,
                fraction,
                ])
        print(x)

    def compute_distibution_unstable_prefixes(self):
        self.compute_prefix_fractions()
        print("\nTop shares of updates")
        for session in self.sessions:
            fraction_list = np.array(
                sorted(
                    self.prefix_list[session].items(),
                    key=operator.itemgetter(1),
                    reverse=True)
                    )[:,1].astype(float)
            _0_1 = int(len(fraction_list)*0.1/100)
            _1_0 = int(len(fraction_list)*1/100)
            _10_0 = int(len(fraction_list)*10/100)
            top_0_1 = fraction_list[:_0_1].sum()
            top_1_0 = fraction_list[:_1_0].sum()
            top_10_0 = fraction_list[:_10_0].sum()
            print("\nSession",session)
            x = PrettyTable(["Top","Updates %"])
            x.add_row(["0.1%",top_0_1])
            x.add_row(["1%",top_1_0])
            x.add_row(["10%",top_10_0])
            print(x)
```