Lance Wetzel

Lab 4

1. Does traditional sampled NetFlow/IPFIX sample packets or flows? Explain.

It samples packets. If packets are randomly sampled to create entries in the NetFlow records, there is no method to track flows and there is no guarantee that once a flow is sample it will continue to be sampled.

2. The Flow Sampling primitive described in Sekar et al. (papers/imc070-sekar.pdf) is designed for network-wide deployment. Explain why.

It divides the work of sampling flows over multiple routers. Induvial routers compute a hash of the 5-tuple flow key and check that value off of a range assigned to that router. If the hash value is in that range, then the router updates the flow record that packet belongs to. The distribution of the assigned hash ranges allow the work of sampling every flow to be shared by every router on the network.

3. Consider the following three cases. Given a single vantage point (router), which approach (Sampled NetFlow or Flow Sampling) is more appropriate for each? Briefly explain your rationale for each answer.
(a) Few large flows.

Flow Sampling. Once a flow has been identified to be sampled (either with sample and hold or another algorithm) it will continue to be sampled until the flow is terminated or the router needs to write its logs to long-term storage. This means that large flows will eventually be sampled, and then more detail will be captured on those flows compared to a uniformed sampling approach.

(b) Many small flows.

Sampled NetFlow.

(c) Mix of large and small flows.

4. Explain the difference between the Sample-and-Hold algorithm, and traditional sampled NetFlow.

Sample-and-Hold picks packets according to a probability P. Once a packet has been selected, all other packets with the same 5-tuple are sampled until that flow ends. This results in more complete sampling of large flows.

5. How many flows do you collect?

number of flows sampled: 8095

6. What are the top 5 heavy hitters in your S&H data structure?

heavy hitters by number of bytes

Src IP	Dest IP	Protocol	Source port	Dest port	#packets	#bytes
'77.110.121.11'	'18.161.232.159'	6	51021	39168	137198	205773832
'77.110.121.11'	'18.161.232.159'	6	51016	38926	132228	198323176
'77.110.121.11'	'18.161.232.159'	6	51006	38677	129529	194265988
'77.110.121.11'	'18.161.232.159'	6	51058	39641	123488	185214624
'77.110.121.11'	'18.161.232.159'	6	51038	39431	113325	169971572

7. How do the top 5 heavy hitters found in S&H differ from the true top 5 heavy hitters (w/o sampling)?

The top 5 heavy hitters from the unsampled flows are:

Src IP	Dest IP	Src Port	Dest Port	# of bytes
77.110.121.11	18.101.245.7	54156	1090	103636500
77.110.121.11	18.89.200.124	56586	3491	101208000
77.110.121.11	18.89.200.124	56792	3497	101139000
77.110.121.11	18.89.200.124	56616	3493	96280500
77.110.121.11	18.89.200.124	56722	3495	96225000

The sampled heavy hitters show the same IP address as the source for the top 5 flows, but the top 5 destination IPs are different. Additionally, I would expect the sampled top 5 flows to show fewer bytes in the flows due to missing some number of packets before the flow was being sampled. However, the sampled flows captured more bytes of traffic than the unsampled ones. (this is due to extra traffic being added to the original records processed by the sample and hold program, and would not happen if the same records were analyzed by both methods)

8. How many unique IP source addresses are present in the flows collected?

unique source IP: 5211

9. How many unique IP destination addresses are present in the flows collected?

unique dest IP: 4943

10. What is the minimum, average, and maximum number of bytes per flow?

min bytes: 32 max bytes: 207159832 avg bytes: 429516.17220506485

11. What is the minimum, average, and maximum number of packets per flow?

min packets: 1 max packets: 144000 avg packets: 761.401358863496

12. What fraction is TCP traffic of the total byte count?

total bytes: 3476933414

tcp bytes: 3415807051

tcp/total bytes: 0.9824194611395569

13. Comment on how well SH approximates the answers you derived in the previous lab.

Due to the original records for both labs being different, this is a difficult comparison. The heavy hitters seem to have been accurately detected, especially the source IP for the heavy flows. This comes at the cost of not detecting the shorter flows, and can be seen by comparing the percentage of TCP traffic between the two methods. The sampled flows indicate the 98% of traffic is TCP, but the unsampled ones show that it is actually 84%. If we are interested in non-tcp traffic (such as malicious traffic masquerading as DNS) the sample and hold method is much less likely to observe it.

CODE

```
'''
if pck not tracked:
    add w/prob P
else:
        update # pkts
        update # bytes

P = L/#packets

key -> value
(ip.src,ip.dst,ip.p,tcp.sport,tcp.dport) -> [num_packets,num_bytes]

'''
import dpkt
import socket
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sys
import operator
import random

fd = open("peering.pcap", "rb")
pcap = dpkt.pcap.Reader(fd)

sample_hold_dict = {}
num_packets = 0
prob = 10000/40000000
```

```python
error_count = 0

print(prob)




#--------------------------START PCAP LOOP-------------------------
for ts, data in pcap:
    #-----------------break for testing----------------------------

    num_packets += 1
    if num_packets % 10000000 == 0:
        print('10 M sampled')

    '''
    if num_packets > 1000000:
        break
    '''
    #----------------end testing break-------------------------------

    ip = dpkt.ip.IP(data)
    ip_src = socket.inet_ntoa(ip.src)
    ip_dst = socket.inet_ntoa(ip.dst)
    protocol = ip.p
    size = ip.len
    #-------------TCP ports----------------
    if protocol == 6:
        tcp = ip.data
        try:
            d_port = tcp.dport
            s_port = tcp.sport
        except:
            errrr_count += 1
            #print('tcp pkt error: ' , tcp , ' not valid')
            d_port = 0
            s_port = 0

    #-------------UDP ports----------------
    elif protocol == 17:
        udp = ip.data
        try:
            d_port = udp.dport
            s_port = udp.sport
        except:
            error_count += 1
            #print('udp pkt error: ' , udp , ' not valid')
            d_port = 0
            s_port = 0


    #----------check if key is in the dict------------------
    key = (ip_src,ip_dst,protocol,s_port,d_port)
    valid_key = True
    if key[3] == 0 and key[4] == 0:
        #print('invalid key, ignoring')
        valid_key = False
    #print(key)
    if key in sample_hold_dict:
        value = sample_hold_dict.get(key)
        ptk_count = value[0]
        byte_count = value[1]
        ptk_count += 1
        byte_count = byte_count + size
        sample_hold_dict[key] = [ptk_count,byte_count]

    #--------if not in dict, add with prob P----------
    elif valid_key == True:
        rand_num = random.random()
        #print(rand_num)
        if rand_num <= prob:
            sample_hold_dict[key] = [1,size]
            #print('new sample added')



#--------------------------------END PCAP LOOP--------------------------
print('END PCAP LOOP')
print('number of packets: ' , num_packets)
print('number of errors: ' , error_count)
total_flow = len(sample_hold_dict)
print('number of flows sampled: ' ,total_flow)
#print(sample_hold_dict)
sorted_max_bytes = sorted(sample_hold_dict.keys(), key = lambda k: sample_hold_dict[k][1] ,reverse = True)[:5]

sorted_sample_hold = sorted(sample_hold_dict.items(),key=operator.itemgetter(1),reverse = True)

print('heavy hitters by number of packets')
```

```python
for i in range(5):
    print('key : ' , sorted_sample_hold[i])

print('heavy hitters by number of bytes')
for i in sorted_max_bytes:
    val = sample_hold_dict.get(i)
    print(i , ',' , val)

#---------ITERATE THROUGH ALL FLOW RECORDS-----------------------
unique_src_ip = []
unique_dest_ip = []
total_bytes = 0
tcp_bytes = 0
min_bytes = 10000000
max_bytes = 0
min_packets = 1000000
max_packets = 0
total_packets = 0

for i in range(total_flow):
    flow = sorted_sample_hold[i][0]
    data = sorted_sample_hold[i][1]
    ip_src = flow[0]
    ip_dst = flow[1]
    protocol = flow[2]
    #---unique src/dst ip addrs-----------
    if ip_src not in unique_src_ip:
        unique_src_ip.append(ip_src)
    if ip_dst not in unique_dest_ip:
        unique_dest_ip.append(ip_dst)
    #---fraction of tcp traffic-----------
    total_bytes = total_bytes + data[1]
    if protocol == 6:
        tcp_bytes = tcp_bytes + data[1]
    #---min/max packets/bytes--------
    total_packets = total_packets + data[0]
    if min_packets > data[0]:
        min_packets = data[0]
    if max_packets < data[0]:
        max_packets = data[0]
    if min_bytes > data[1]:
        min_bytes = data[1]
    if max_bytes < data[1]:
        max_bytes = data[1]
```

```
print('unquie source IP: ' , len(unique_src_ip))
print('unquie dest IP: ' , len(unique_dest_ip))
print('total bytes: ' , total_bytes)
print('tcp bytes: ' , tcp_bytes)
print('tcp/total bytes: ', tcp_bytes/total_bytes)
print('min packets: ' , min_packets , ' max packets: ' , max_packets , ' avg packets: ' , total_packets/total_flow)
print('min bytes: ' , min_bytes , ' max bytes: ' , max_bytes , ' avg bytes: ' , total_bytes/total_flow)
#  random.randint(0,99)  use for random generation

fd.close()
```