Università degli Studi di Pisa

DIPARTIMENTO DI INFORMATICA

Corso di Laurea in Informatica

TESI DI LAUREA

IDENTIFYING AND REMOVING ABNORMAL TRAFFIC FROM THE UCSD NETWORK TELESCOPE

Candidata:
Elif Beraat Izgordu
Matricola: 491044

Relatore:
Luca Deri

Anno Accademico 2016-2017

Index

1 Introduction 4

2 Motivation and Related Work 6

2.1 UCSD Network Telescope 8

2.2 Telescope Usage Example 10

2.3 IP Address Spoofing 12

2.4 Overloading Capture Capacity 14

3 Architecture 15

3.1 Collected Statistics 17

3.1.1 Port Based Statistics 17

3.1.2 Scanner Statistics 18

3.1.3 Receivers Statistics 19

3.2 Algorithms and Data Structures 19

4 Implementation 22

4.1 ndpiReader 22

4.2 Original Contribution 22

4.2.1 Statistics 24

4.3 Memory Concerns 27

4.4 Filters 28

4.4.1 Filter for Packet Burst 28

4.4.2 Filter for Host Burst 30

4.5 Source Code 31

5 Validation 32

5.1 Packet Burst Examples 35

5.2 Host Burst Examples 37

6 Conclusions 39

1 Introduction

In the last 30 years Internet has had a revolutionary impact both on our society and on our daily lives. There are countless studies for each and every aspect of the Internet; it's behaviour and evolution at macro-scale also has been an important source of research data, not for only computer science but for many disciplines, including even social sciences.

Therefore, understanding the evolution of Internet infrastructure is very important. Yet developing instruments and methods that can measure and analyse macroscopic phenomena on the Internet is not trivial.

One of the most important aspects to understand the evolution of the Internet infrastructure is monitoring and studying internet address space utilisation. It's a known issue that IPv4 address space is almost exhausted but as a matter of fact, not all of the allocated addresses are effectively in use. As mentioned in the study of Dainotti, Benson, King, Kallitsis, Glatz, Dimitropoulos [1]

"Macroscopic measurement of patterns in IPv4 address utilisation reveals insights into Internet growth, including to what extent NAT and IPv6 deployment are reducing the pressure on (and demand for) IPv4 address space."

In the course of this study, the present scientific works with the aim of mapping actual utilization of IPv4 addresses, their limitations and how the mapping can be improved in the particular case of CAIDA Network Telescope[2] are going to be introduced.

Two approaches for the mapping problem; active and passive probing, their challenges are going to be analyzed in the Motivations and Related Work chapter. Particularly Network Telescope[3] (or a darknet, which is a portion of routed IP address space in which little or no legitimate traffic exists), it's usage for scientific inferences and the problems that are threatening it's data integrity are going to be introduced in this chapter. After introducing the terminology, limitations of the current approach for data sanitization (in order to overcome data integrity problems) and difficulties of working with the telecope data are going to be described.

In the Architecture chapter these limitations and difficulties are going to shape our approach and decisions taken to deal with the original problem of this work: improving the current approach for data sanitization. Next, in the Implementation chapter details of the original contribution and technologies used to realize it are going to be introduced.

At the end in the Validation chapter, efficiency and validity of the solution is going to be demonstrated with the test results.

2 Motivation and Related Work

Until now there has been two scientific work for monitoring the extent to which allocated IP addresses are actually used[4]. Both of these works have their own limitations. There are two approaches that separate these two scientific work fundamentally, that is monitoring can be implemented by active or passive probing.

First work is the ISI's Internet Census project[5] in which address utilisation has been monitored via actively scanning the entire IPv4 address space. It periodically sends ICMP echo requests(i.e. ping) to every single IPv4 address (excluding private and multicast addresses) to track the active IP address population.

Active scanning approach has four primary limitations: [6]

i) there is a measurement overhead,

ii) measurement infrastructure can be potentially blacklisted

iii) networks filtering ICMP request cause measurement bias,

iv) not scalable for use in a future IPv6 census.

Second is the CAIDA's UCSD Network Telescope [7] project through passive measurments. The Center for Applied Internet Data Analysis (CAIDA) conducts network research and builds research infrastructure to support large-scale data collection, curation, and data distribution to the scientific research community [8]. Project is realized by analyzing two types of passive traffic data: (i) Internet Background Radiation

(IBR) packet traffic captured by darknets (aka telescopes); (ii) traffic (net)flow summaries in operational networks.

Passive traffic measurements overcomes the challenges posed from active probing approach; it doesn't introduce network traffic overhead, doesn't rely on unfiltered responses to probing and could apply to IPv6 as well. It also detects additional active /24 blocks that are not detected as active with ISI's active probing approach.

On the other hand, it introduces new challenges to deal with: [9]

> i) the limited visibility of a single observation point;
>
> ii) the presence of spoofed IP addresses in packets that can affect results by implying faked addresses are active.

If the presence of spoofed packets(packets with a fake source IP address) is significantly large (thousands of IP addresses per minute) it can invalidate the inferences, resulting in a much more densely utilised IPv4 address space. Therefore, packets with spoofed source addresses threaten integrity of the data obtained from network telescope, because many research use of data depends on the source address of the packet.

CAIDA develops and evaluates techniques to identify and remove likely spoofed packets from both darknet (unidirectional) and two-way traffic data. Their work focused on filtering large-scale spoofing by manually isolating and analyzing suspicious traffic and then defining filters to remove them.

These filters are static filters (e.g filter traffic which has TTL > 200 and not ICMP, filter traffic with least significant byte src addr 0 or 255) which cover most of the spoofed traffic cases because they can be determined by well-known patterns which indicate that traffic can be nothing but spoofed. They significantly reduce amount of spoofed traffic over the network but there are still large-scale spoofing events that can invalidate the inferences.

This work contributes to the effort of improving darknet data usage. Primarily contributing to filter spoofed source traffic and packet burst traffic on the UCSD Network Telescope. These non-filtered spoofed traffic have case-specific reasons. Therefore current techniques of CAIDA are extended with a dynamic approach to determine and filter those cases that could not be determined by static filters.

Further in this section, to understand better the problem and it's challenges, Network Telescope data usage is going to be examined with an example. Then the issues that threaten data integrity are going to be covered; specifically IP address spoofing and packet burst cases.

2.1 UCSD Network Telescope

CAIDA hosts The UCSD Network Telescope , one of the largest network telescopes (a /8 network segment - approximately 1/256th of all IPv4 Internet addresses - that observes about 20TB of traffic per month) operated by the University of California San Diego .

A network telescope (aka a black hole, an Internet sink, darkspace, or a darknet)[10] is an Internet system that allows one to observe different large-scale events taking place on the Internet. The basic idea is to observe traffic targeting the dark (unused) address-space of the network.

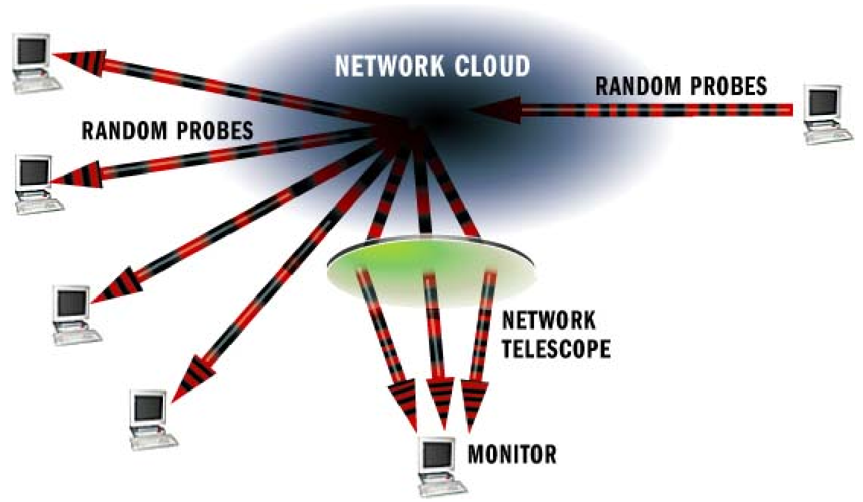


Figure 1: A Network Telescope Representation

UCSD Network Telescope is a passive traffic monitoring system that carries almost no legitimate traffic because there are few provider-allocated IP addresses in this prefix. After discarding the legitimate traffic from the incoming packets, the remaining data represent a continuous view of anomalous unsolicited traffic, or Internet Background Radiation (IBR). IBR results from a wide range of events, such as backscatter from randomly spoofed source denial-of-service attacks, the automated spread of Internet worms and viruses, scanning of address space by attackers or malware looking for vulnerable targets, and various misconfigurations (e.g. mistyping an IP address).[11]

This anomalous unsolicited traffic reaching to the network has its own "normality". In theory, no traffic should reach the darknet, but there are periodic scans (robots) and other activities that are somehow normal. However there is some traffic (e.g. TCP replies) that is definitively unlegitimate as there is no request coming. The goal of this work is not to filter out this traffic but rather to remove those flows of traffic that with its brutality affect the natural shape of the traffic when such phenomena are not observed. Observing the traffic regularly reaching to the telescope from different geographic regions (countries, provinces) or Autonomous System (aggregations) allows global visibility into macroscopic phenomena such as outages, censorship, security-related issues (and revealing insights about their dynamics) and utilisation of IP address resources.

2.2 Telescope Usage Example

As an example for it's usage (to reveal a macroscopic phenomena), CAIDA observed Syria's Internet blackout that occurred on the 29th November 2012 due to the Syrian state telecom's withdrew of the majority of BGP routes to Syrian networks [12].

As Network Telescope receives anomalous unsolicited traffic generated by malware-infected PCs all over the world (infected hosts spreads malware to other vulnerable computers over the Internet by randomly scanning), a country-level Internet blackout causes a significant drop in unsolicited traffic reaching to the network by the malware-infected Syrian PCs. Because Internet access is also denied to malware attempting to infect other hosts. As a result, blackout can be observed in data captured from the UCSD Network Telescope.

Graph below shows number of unique Syrian source IP addresses per hour sending traffic that reaches the UCSD Network Telescope. There is a sudden decrease in the number of transmitting Syrian hosts between 10 and 11am UTC on the 29th which coincides with blackout.

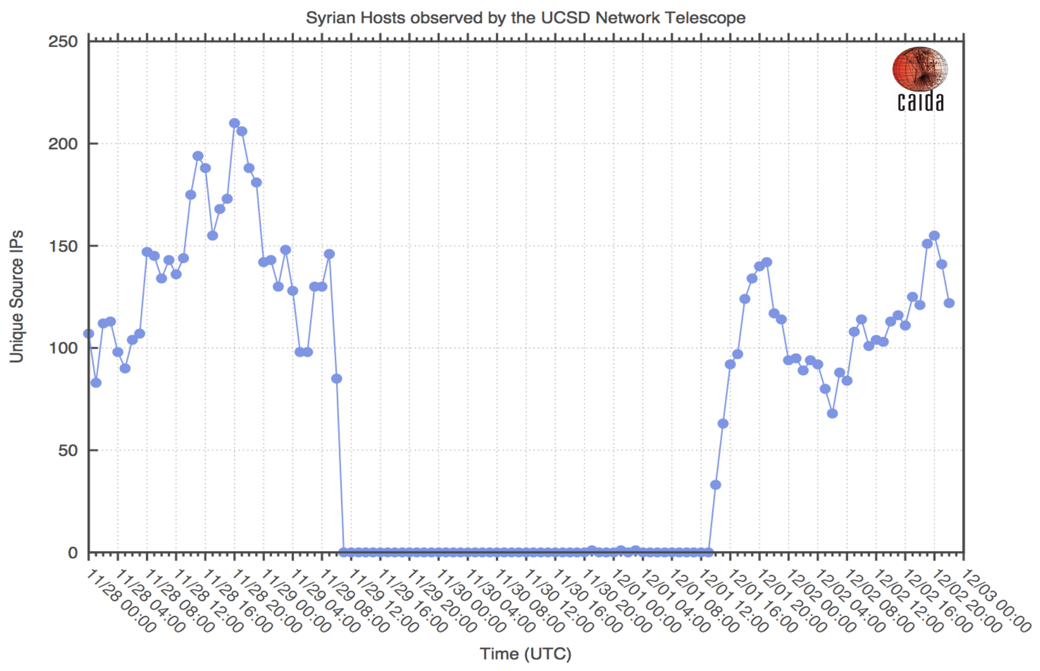


Figure 2 : The Syrian Internet Blackout in Nov 2012 as seen at the UCSD Network Telescope

2.3 IP Address Spoofing

As stated above, passive probing techniques through darknets affected by two main problems:

(i) the limited visibility of a single observation point;

(ii) the presence of spoofed IP addresses in packets

IP address spoofing is the creation of (IP) packets with a fake source IP address for the purpose of hiding the identity of the sender [13]. It's a viable attack method for redirection, amplification, and anonymity over the network. Even though typical reason of address spoofing is to hide the real source (to avoid being caught), it can be produced also due to transmission or programming errors that induce address bit errors. Since there are no hosts to attack, it's unlikely for a darknet to be the target of spoofed DOS attacks, even though it still receives un/intentionally spoofed packets.

Responses to packets with spoofed sources (because responses themselves reaching the telescope have legitimate source addresses) are one useful component of IBR, but packets with spoofed source address directed to the telescope interfere its use for various classes of scientific inferences like detection and analysis of large-scale Internet outages, discovery of new traffic patterns or studying trends in IPv4 address space usage.

The presence of spoofed packets in this traffic will erroneously indicate activity from given sources, leading to incorrect or inaccurate inferences like suggesting a much more densely utilised IPv4 address space or resulting in erroneous detection of outages (false positives).

Since darknets only receive traffic and do not respond, applying bidirectional flow-based data analysis techniques is not possible. In addition, defining “normal” traffic is inherently difficult because traffic received by darknets comes from a variety of unpredictable sources (like malwares or misconfigurations at different layers of the TCP/IP stack)[14].

Therefore CAIDA focuses on identifying and filtering out large portions of spoofed traffic (by identifying suspicious traffic components and defining static filters based on network and transport layer packet headers to remove them) to mitigate the effects of spoofing on measurements, rather than first identifying unspoofed traffic like it’s done with bidirectional traffic.

In search of large-scale spoofing from suspicious traffic components, CAIDA looks for two behaviours [15]:

1. bursty behaviour – (i) sudden spikes in the number of unique source IP addresses, unique source /24 blocks, and newly observed source IP addresses (source /24 blocks) per hour; (ii) the same type of events with only source addresses in unrouted network blocks (a /24 block is considered as routed only if covered by a prefix visible by at least 10 BGP peers[16]);

2. long-term consistent behavior: (i) aggregating packets over the entire measurement window into traffic classes by protocol and port (when applicable) and investigating classes with many originating unrouted /24 blocks; (ii) packets are aggregated based on the least significant byte of the source address to look for inconsistencies in address utilisation.

2.4 Overloading Capture Capacity

Another concern for the integrity of the data source is that telescope regularly observes bursts of traffic that exceed its capture capability. They are mainly consequences of large-scale coordinated bursts caused by bot-nets or misconfigurations. These kind of bursts can overload the capture capacity of the infrastructure inducing packet loss and misleading timestamps, causing misinterpretation of phenomena.

For example, when the packet burst overloads the capture infrastructure, it would drop packets and a sudden decrease in the number of unique source IP addresses would be observed which could be erroneously interpreted as an Internet outage (based on their geolocation or assignment).

3 Architecture

This work basically consist of resolving two problems regarding Network Telescope data usage:

(i) Filtering host bursts caused by large-scale spoofed traffic

(ii) Filtering packet bursts saturating capture capacity of the telescope

In this chapter architecture with requirements and related choices that guided the approach to resolve these two problems explained above is going to be introduced.

Since Network Telescope receives only abnormal unsolicited traffic, it's challenging knowing what information to extract and determining reasons of "abnormal" events in the context of a darknet. Bidirectional flow-based data analysis techniques or defining first "normal" traffic then exclude the remaining cannot be applied.

Therefore this challenge requires to firstly studying the traffic manually to have an insight about what kind of information can be useful. Then examining it to see some patterns that could be tracked down and changing trends between burst traffic and non-burst traffic. After gaining first insights, the need for examining traffic by collecting statistics about top producers/consumers in different keys is emerged. These statistics make possible revealing the nature of burst traffic.

This solution provides statistics collected based on the flow information which is obtained from the raw traffic data. With each flow various statistics are updated. At the end of the statistic collection process, top talkers of each type of statistics are saved in a format that can be parsed in order to generate filters. Figures below represent inner architecture and the run-time architecture respectively.

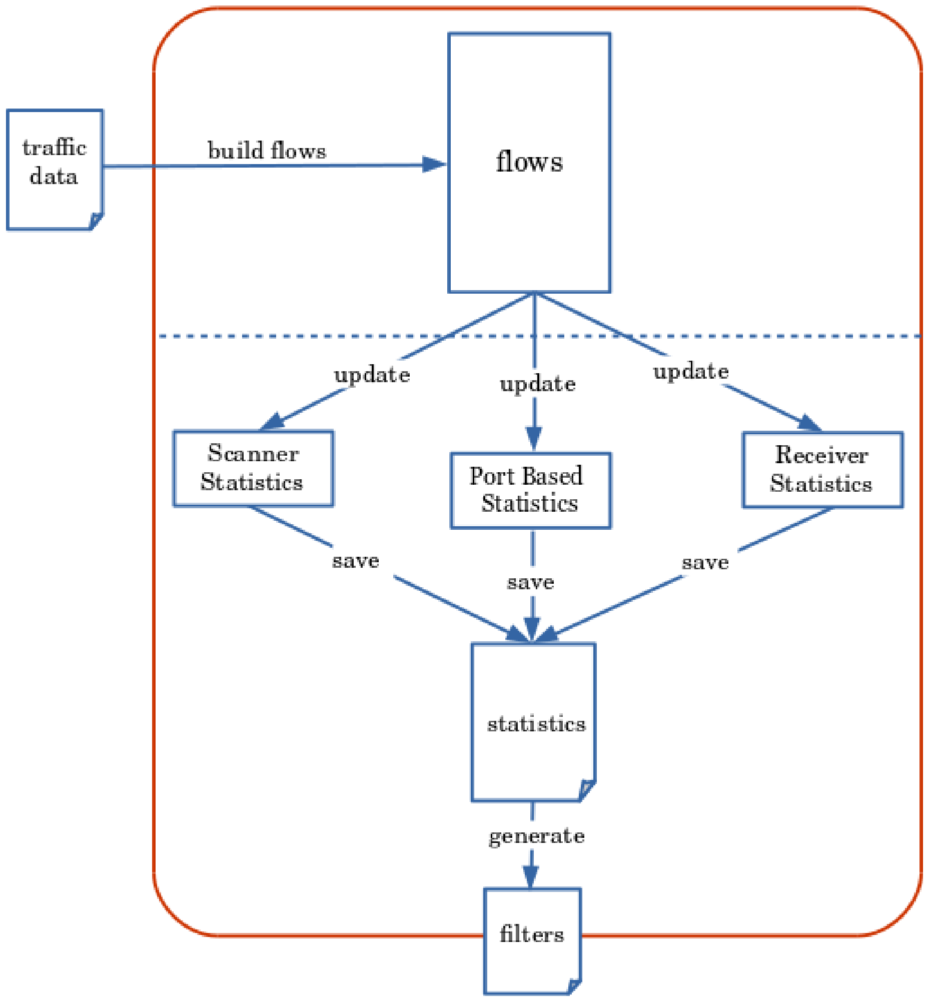


Figure 3: Inner Architecture

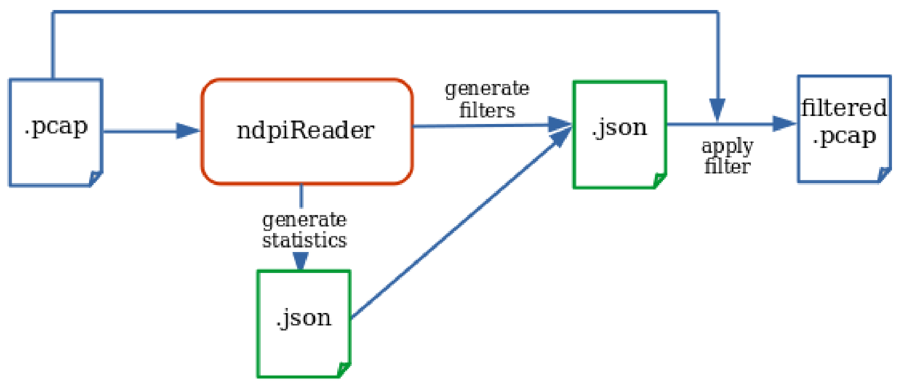


Figure 4 : Architecture at run-time

3.1 Collected Statistics

3 type of statistics are collected for two problems mentioned above:

(i) port based statistics

(ii) scanner hosts statistics

(iii) receiver hosts statistics

Each statistic type is consist of a key value (that statistic is collected based on it) and a series of flow-based information collected to compare and order them respect to the event that we are looking for to filter.

3.1.1 Port Based Statistics

Ports are very characteristic information about the traffic. Tracking down port based informations helps to reveal unusual events on the darknet. “Sender side” and “Receiver side” statistics based on source port and destination port keys respectively are collected separately. Since it’s a darknet, the traffic is one-directional that is only coming in to the network but not going out. For each port collected:

- port number (as key value)
- number of packets sent from the port
- number of flows the port is involved
- percentage of flows respect to total flows for the given traffic
- ratio between number of flows (the port is involved) and number of packets (sent from the port). If the ratio is 1 (or close to 1) then all (or almost all) traffic on the given port is consist of single packet flows.
- the most encountered source/destination host address that is involved in more than 95 percent of the traffic for the (source/destination respectively) port. This host is called as “aggressive host”. It’s going to be distinctive information because not every port has an aggressive host.
- Application level protocol of the aggressive host. This information is not used in the ultimate analyses to generate filters, but collected to gain insight about the traffic.

3.1.2 Scanner Statistics

One of the sources of the network telescope data is traffic reaching from scanner hosts, that are scanning randomly to find vulnerable hosts. These scanner hosts can contribute to the packet burst. Therefore we collect scanner host statistics. Considering only TCP traffic, hosts that send single packet flows and send more than 1000 flows per minute are definitely scanner hosts. Because TCP within a minute makes transmissions.; For the flows which satisfy the condition above, collected:

- source host address of the scanner (as key value)
- number of flows that scanner host is involved
- top 10 destination ports that scanner host targeted
    - destination port
    - number of flows that scanner host involved on this destination port.

3.1.3 Receivers Statistics

Third and last kind of statistics is to obtain most targeted destination hosts. This statistic is useful to cover a packet burst case which we will explain more in detail in further. For the receivers collected:

- destination host address (as key value)
- number of packets that host address receives
- percentage of number of packets respect to total number of packets

3.2 Algorithms and Data Structures

A good choice of algorithms and data structures affects performance and efficiency more than any other aspect of the program. Especially working with Network Telescope data requires to pay attention for the memory usage because of data dimension.

Since the solution approach requires to collect statistics within large number of unique items (port based statistics have potentially 65535 different keys and destination host address based statistics have potentially 2^{24} different keys) in order to improve memory usage periodic process is adopted to collects statistics in fixed time intervals. At the end of each interval memory used for data structures to store statistics are freed and the collecting process is started again. It helps to keep memory usage limited with an upper bound. The time interval value is determined at run time, passed as a parameter to the program.

Choice of data structures for statistics collection has a direct impact on the performance. In the terms of time complexity hash table is the best choice for our purposes. With a hash table, search, insert and delete operations has O(1) complexity in average and O(n) at worst case [17].

An ulterior improvement for the memory usage is required for receiver statistics. Collecting statistics based on destination host address indicates a range of 2^{24} different keys (Network Telescope's resolution is /8). Keeping a hash table with that dimension in memory is not possible. Therefore it requires adopting an algorithm that can keep the hash table in reasonable sizes without loosing top players at the end of the process.

To resolve this problem top-k algorithm [18] is used which is a generic solution to compute sorted top-n views from very large numbers of flow information records where storing individual counters per aspect components is not possible. It’s a simple but efficient algorithm that fits perfectly the problem. It’s implementation details will be explained in the next chapter.

4. Implementation

4.1 ndpiReader

ndpiReader is an example tool that uses nDPI library, which is a ntop-maintained superset of the popular OpenDPI library [19]. ndpiReader is able to read from a pcap file or capture traffic from a network interface and process it with deep packet inspection library. Although it implements only some basic features just to show what can be done with nDPI library, it is still a strong tool that provides many information about the traffic and especially the flow information which is essential for our work. Implemented in C language, all nDPI library, ndpiReader tool and the original contribution of this work are open source and have GNU Lesser GPL license.

4.2 Original Contribution

ndpiReader basically process the traffic data and builds flows based on the packets. For the implementation of architecture introduces in the 3. chapter, ndpiReader is extended in order to collect statistics about the traffic based on these flow informations and to generate filters based on these statistics.

ndpiReader parse pcap file to builds flows and stores them in a binary tree. Once it builds all the flows, extension code traverses the tree and updates statistics with each flow (node of the tree). At the end of the collection process, statistics are sorted and top 10 items for each type of statistics are saved with JSON format. Top 10 is preferred because analyses showed that typically first 1, 3 or 5 players are the ones to filter but there is also need to observe how it differs from "normal traffic". Therefore top 10 is a good range for this purpose. This process is repeated periodically based on the analyses duration time interval (expressed in seconds as command option). For each interval a new JSON object is created. Therefore if the process repeats, generated file will be a list of JSON objects.

JSON format is preferred at least for 3 reasons:

- ✔ it has a compact yet human-readable format.
- ✔ ndpiReader needs to parse statistics in order to generate filters, JSON format is a convenient format to parse objects.
- ✔ generated statistics could be useful for further diagnostic operations. Saving them with a well-known format like JSON makes it easier for who in future needs to operate on the produced data.

Then with a second command ndpiReader parses the JSON file in order to generate BPF filters based on the conditions determined by analyses which we will be explained further in this section. Generated filters are also saved with JSON format.

4.2.1 Statistics

Each kind of statistic type (port based, scanner and receiver statistics) is represented with a C struct which holds related informations explained in the previous chapter. To store items of a certain statistic kind uthash [20] is used, a minimalistic and efficient hashtable implementation for C structures.

Statistics file consist of:

- (i) duration of time interval in seconds
- (ii) timestamp for the beginning of time interval
- (iii) list of port based statistics
- (iv) list of scanner hosts statistics
- (v) list of receiver hosts statistics

for each analyses duration time interval.

An example statistics file can be seen below. This file is generated for a data file which has a traffic of 60 seconds time interval. Analysis duration time interval is also set as 60 seconds(-m option in command). Therefore it generates a single JSON object. This statistic file is generated with the command:

```
$ ./ndpiReader -i data.pcap -m 60 -b statistics.json
```

```
{       "duration.in.seconds": 60,
        "statistics": [{
                "time": "2017-09-11T18:29:00Z",
                "scanner.stats": [{
                        "ip.address": "195.3.146.96",
                        "total.flows.number": 610805,
                        "top.ports": [{
                                "port": 3001,
                                "flows.number": 19852
                        }, {...}]
                }, {...}],
                "top.receiver.stats": [{
                        "ip.address": "X.166.40.124",
                        "packets.number": 8082,
                        "packets.percent": 0.002
                }, {...}],
                "top.src.pkts.stats": [{
                        "port": 45962,
                        "packets.number": 611014,
                        "flows.number":  610987,
                        "flows.percent": 3.690,
                        "flows/packets": 0.999,
                        "aggressive.host": "195.3.146.96",
                        "host.app.protocol": "Unknown"
                }, {...}],
                "top.src.host.stats": [{
                        "port": 0,
                        "host.number": 3862,
                        "host.percent": 0.054,
                        "flows.number": 80278
                }, {...}],
                "top.dst.pkts.stats": [{
                        "port": 34001,
                        "packets.number": 0,
                        "flows.number":  4004958,
                        "flows.percent":  24.193,
                        "flows.num_packets": 0,
                        "aggressive.host": "X.217.31.103",
                        "host.app.protocol": "Unknown"
                }, {...}],
                "top.dst.host.stats": [{
                        "port": 23,
                        "host.number": 3882552,
                        "host.percent": 33.142,
                        "flows.number": 4481540
                }, {...}]
        }]
}
```

Reasons behind statistics criterias are discussed below.

For the port based statistics top 10 source/destination ports which have an aggressive host(explained in section 3.1.1) are saved. The reason behind this choice is that, top ports without considering aggressive host are typically standard port numbers for known protocols like 0(ICMP), 23(FTP), 80(HTTP) which are truly most used ports but they are not significant in the search of burst causes. Because we are looking for high number values in short time intervals. Instead these ports always receive abundant traffic. Adding aggressive host condition eliminate these ports and give us truly significant ports which can potentially be involved in burst.

For the scanner statistics, considering only TCP traffic, hosts that send single packet flows and send more than 1000 flows per minute are definitely scanner hosts. Because TCP within a minute makes transmissions. In our statistics we consider top 10 scanner hosts in the terms of number of flows they send and their top 10 destination ports.

During testing phase port based and scanner statistics are failed to produce true filters for some packet burst cases. When there is a packet burst, they typically have some top players with significantly differing values. But there is a different kind of packet burst in which there are very distributed values for these statistics. With analyses discovered a different packet burst case which requires to collect a different kind of statistic. In this case there are no significant values on the sender side but on the receiver side one or few destination host receives significantly greater traffic respect to other hosts but distributed to a range of destination ports. Therefore receiver statistics collected and top 10 destination host addresses with most packets received are saved.

4.3 Memory Concerns

As mentioned in the 3.2 , collecting receiver statistics requires to adopt a memory-friendly algorithm because of large dimension key domain. Therefore top-k algorithm is implemented to be able to keep a reduced size hash table in memory.

Algorithm utilise two hash tables one as primary and other as secondary with size max2 and max1 (typically max2 = max1*2) respectively. It updates the primary hash table with every new item until it reaches to size max1. At this point adds new items only if they likely have an impact on top items (this is determined by a heuristic function). If item count reaches to size max2 then it sorts the primary hash table and cut it back to size max1 in order to merge it with secondary hash table(which is initially empty). After the merge operation, if secondary hash table exceeds size max1 then it is going to be sorted and cut back to the size max1. At the end of the collection process secondary hash table will have final top max1 receivers(destination host addresses).

For the efficieny of algorithm, heuristic function must be simple and more importantly ‘cheap’ to implement memory and cpu wise. The heuristic function used in this implementation accepts a new item(a new dest. host address) only if it’s flow has more than 10 packets. For the max1 and max2 thresholds we tested different values with the aim of minimizing necessary dimensions to get true results. For a domain of 2^{24} possible values, 4096 as max1 and 8192 as max2 worked well enough to get satisfying results. With this algorithm, items keeps at most 3*max1 size in memory in any moment.

4.4 Filters

In this chapter, conditions that used on statistics to generate filters are discussed.

4.4.1 Filter for Packet Burst:

Packet bursts have more than one reason in their occurrences. Analysis showed 2 different cases in which paket burst occurs. Therefore 3 kind of statistics are used to determine the filter.

Starting with the port statistics, number of packets sent from a given port is significantly bigger if it is involved in the packet burst and flows are mostly single packet flows. Between top 10 source port statistics if a port has a flows/packets ratio greater than 9 percent(it means traffic is consist of mostly single paket flows) and flows percent are greater than 1 percent(this threshold eliminates the rumor) then it contributes to traffic significantly. But even between this top source ports there can be significant differences in the terms of their contribution to the total traffic. Therefore we use an ulterior condition just between top 10 ports. If number of packets sent from a given port is greater then average of top 10 ports then we filter this source port number. So our condition can be expressed like this:

```
if flows/packets > 0.9 and
    flows.percent > 0.1 and
        packets.number > average

then we will eliminate this src port
```

A scanner host will be filtered if it's number of flows is significantly higher than average. To be able to determine who has a differing flows number standard deviation of flow numbers for the top 10 scanner hosts is calculated. To filter a scanner host the condition is as below:

```
if total.flows.number > average + standard deviation
    then we will filter this src host address
```

Top receivers hosts which are involved in the burst traffic are typically have more than 1 percent of total packets. As observed during analyses, this is a high value in a network with a 2^24 IP addresses. Hence, those destination host addresses are filtered to risolve the second kind of traffic burst that is mentioned above. So the condition expression:

```
if packets.percent > 0.1
    then we filter this dst host address
```

4.4.2 Filter for Host Burst:

Analysis showed that host bursts are occurring typically because of great number of sender hosts targeting only one or few destination hosts. These destination host(s) result as the aggressive host on the collected destination side port statistics. An aggressive host involved in the host burst typically has a flows percent greater than 2 percent. This threshold is observed by producing statistics for many burst and non burst traffic intervals. Therefore it's is used as a condition to get the destination hosts to filter. This one simple condition filters the root cause of host burst perfectly.

```
if flows.percent > 0.2
        then we will filter this dst host address
```

Based on these conditions on collected top statistics, BPF filters are created. More specifically source ports, source host addresses or destination host addresses are filtered depending on the burst case. ndpiReader analyses the traffic and tries to create a filter for both packet and host burst, when it's possible. Then the relevant filter is applied (for packet burst or host burst) to the traffic. Filters are saved as a json object with two pairs respectively:

(i) pkt.peak.filter : for packet burst filter

(ii) host.peak.filter : for host burst filter

An example filter file generated with the following command can be seen below:

```
$ ./ndpiReader -x statistics.json
```

```
{
     "pkt.peak.filter": "not (src port 45962 or 44473
                         or 42619) and not (src
                         195.3.146.96 or 95.215.1.37)",
     "host.peak.filter": "not (dst X.217.31.103)"
}
```

4.5 Source Code

For further implementation details you can refer to the online repository at:

https://github.com/beratx/nDPI

5 Validation

In this chapter explained how the solution is tested and how the results are evaluated.

The solution to the original problem is tested with 3 months of data from Network Telescope. Within 3 months, different packet burst and host burst cases occuring in different moments are picked up. A burst is considered as arrival of excessive number of packets or hosts over a short period(around 5 minutes) .

To determine scale and timing of a burst another instrument is used; IODA (Internet Outage Detection and Analysis), that is another CAIDA project which monitors the Internet, in near-realtime, to identify macroscopic Internet outages affecting the edge of the network, i.e., significantly impacting an AS or a large fraction of a country [21].

IODA Explorer visualises traffic reaching to Network Telescope as a continuous graph in time. Thanks to IODA it's easy determining when a burst occurs and apply analysis techniques to only related part of the traffic. It allows to see how traffic is evolving in time with different keys like number of hosts/packets per unit of time. A burst is observed as a peak in the graph. Below there are 2 screenshots from IODA Explorer that shows a host burst and a packet burst respectively.

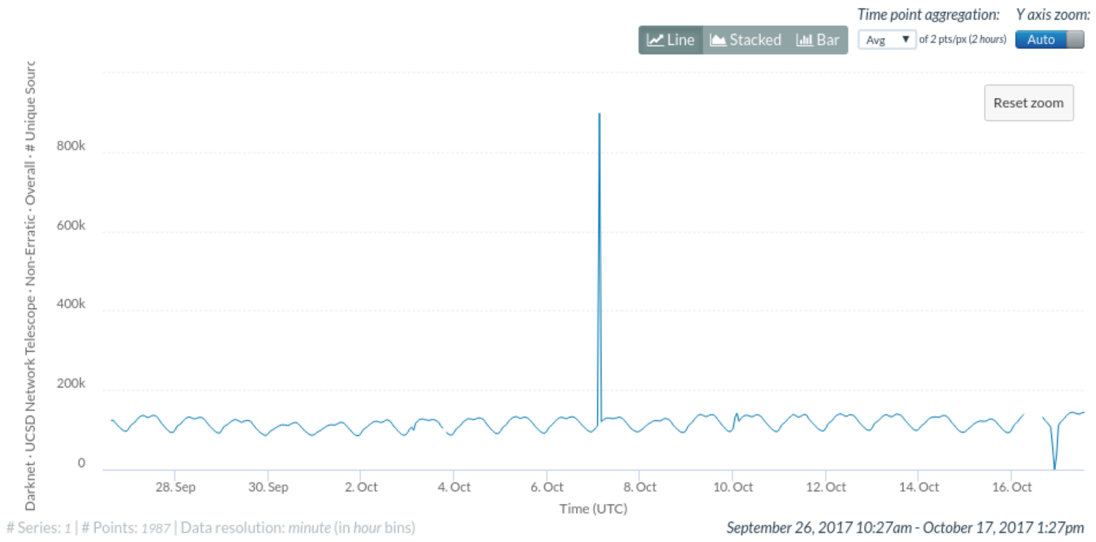


Figure 5 : IODA Explorer Graph shows number of distinct IP address received in time

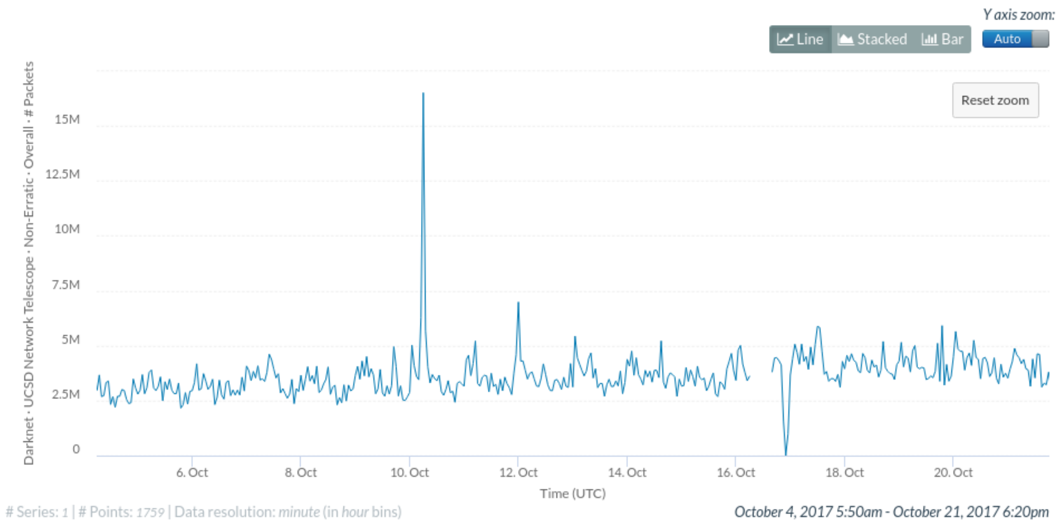


Figure 6 : IODA Explorer Graph shows number of IP packets received in time

Traffic data is stored as per-hour pcap files in the servers of CAIDA. After determining in which day and hour burst occurs, relevant pcap file is sliced to get only relevant part of the traffic. Then ndpiReader is launched with the obtained pcap file in order to generate filters. Produced case-specific filter is applied to the same interval that is sliced from pcap.

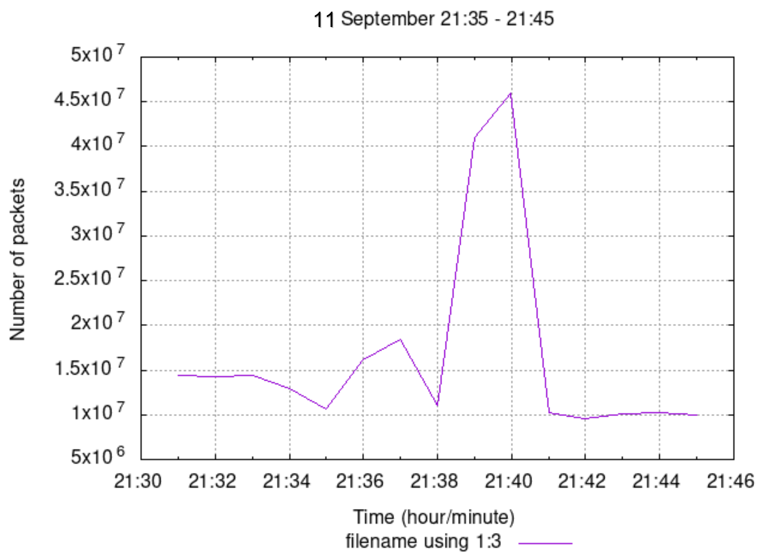
To see if filters effectively remove the burst traffic same slice of pcap is plotted before and after the filtering operation tramite gnuplot [22], which is a command-line program that can generate two- and three-dimensional plots of functions and data.

Particular nature of the darknet data makes it difficult to evaluate results of this work. Since all the reaching traffic is abnormal, "abnormality" in the context of the darknet is defined apart from the regular meaning of it. The abnormality that is wanted to eliminate is determined with the excess of something respect to the average (that is, "normal") traffic.

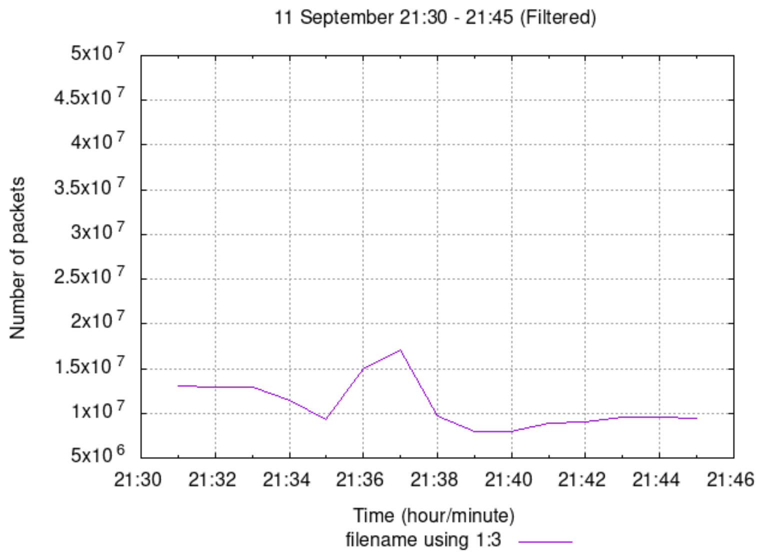
Therefore results are evaluated as satifying because produced graphs gives expected results as the peak seen before filtering disappears after applying filter but the rest of the traffic remains almost the same. (Traffic causes the burst is removed from the whole interval, not only from the moment of burst. That's how you can be sure that only responsible traffic is effectively removed).

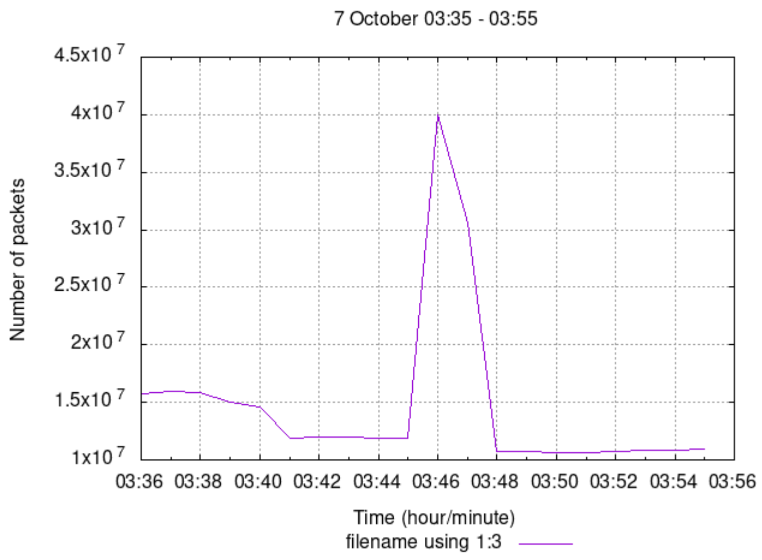
Last part of this chapter is reserved for 2 example cases for each kind of burst that are produced in test and validation phase. First graph shows unfiltered burst traffic and the second graph shows the result after applying BPF filter generated by ndpiReader. Filter used by each example can be seen between graphs.

5.1 Packet Burst Examples

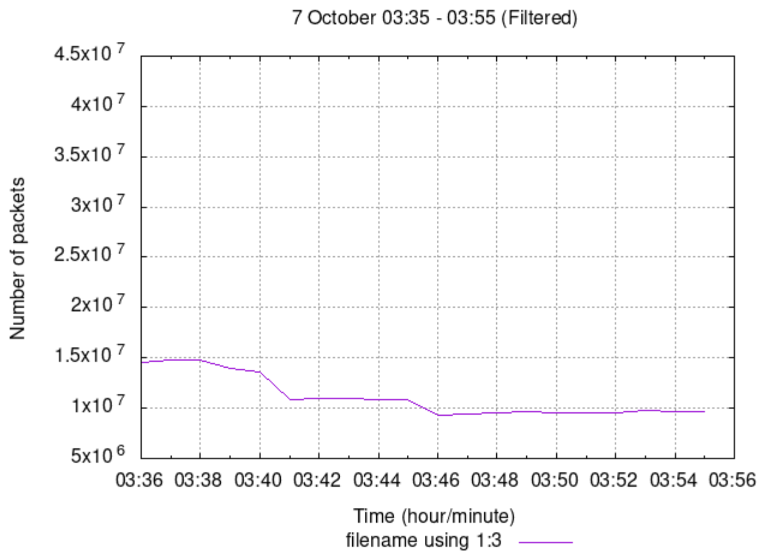


```
pkt.peak.filter : "not (src port 44473 or 5062 or 52304)
                   and not (src 45.55.21.121)"
```

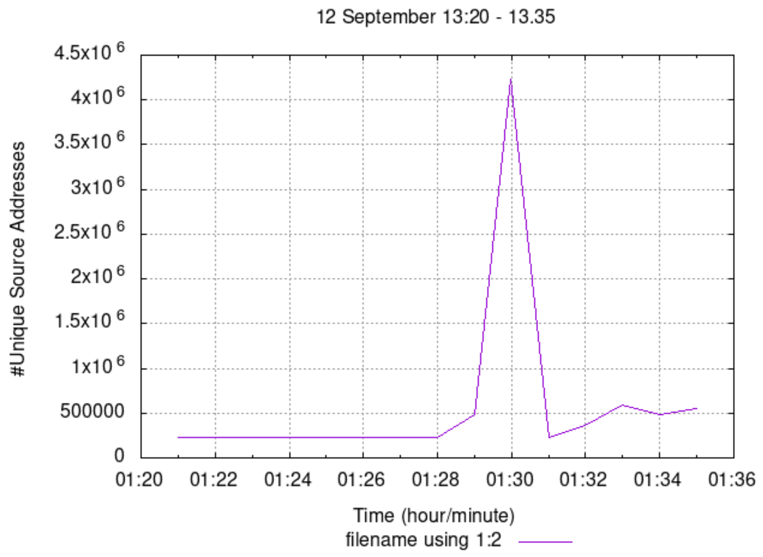




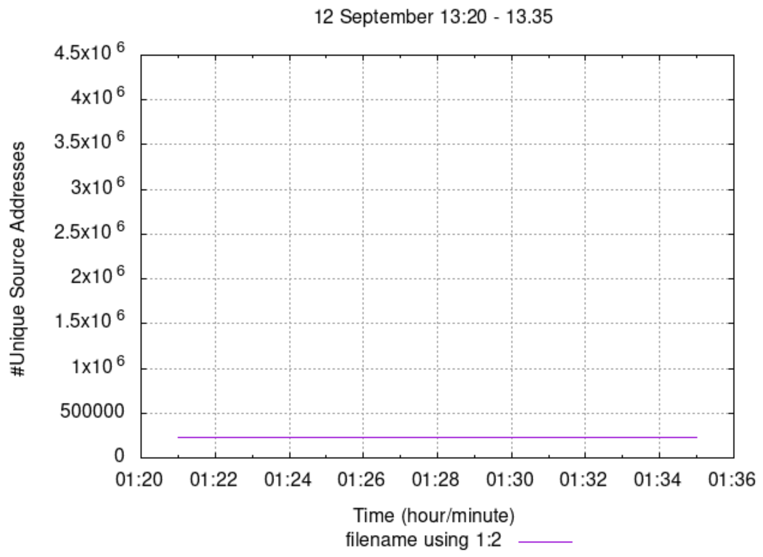
`pkt.peak.filter : "not (dst X.33.13.233)"`

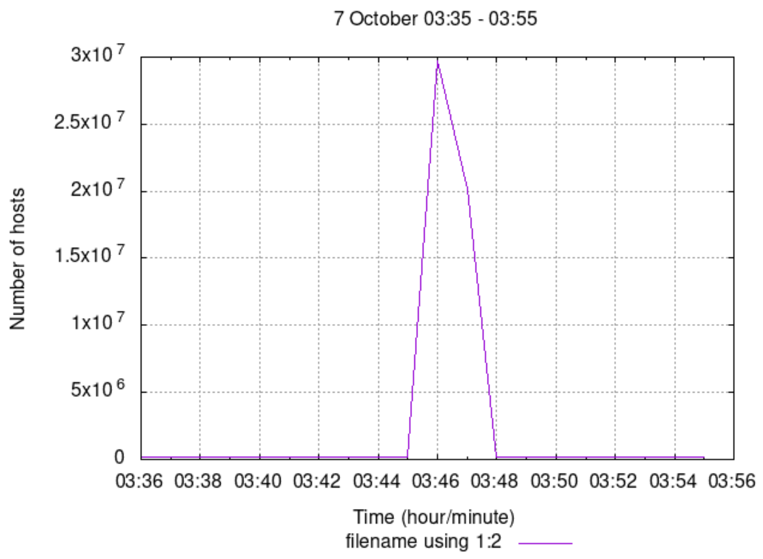


5.2 Host Burst Examples :

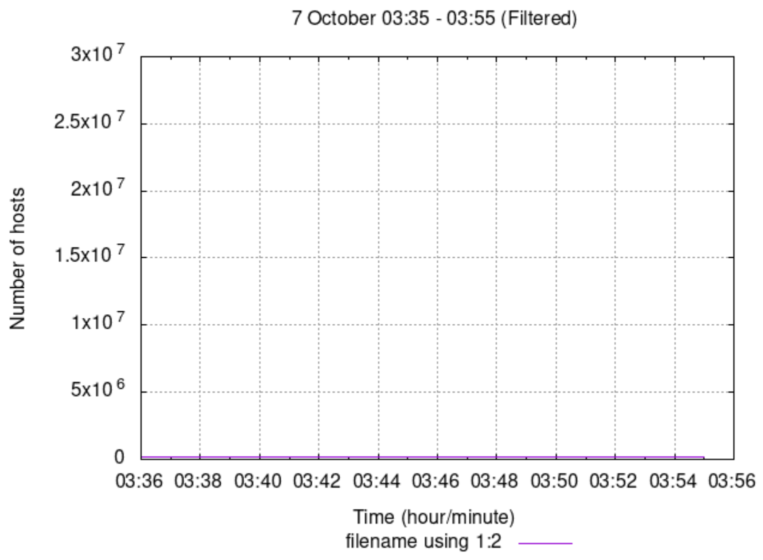


host.peak.filter : "not (dst X.217.31.103)"





```
host.peak.filter": "not (dst X.33.13.233)"
```



6 Conclusions

In the 2. chapter we introduced darknet data usage to analyse and measure macroscopic phenomena on the Internet and the problems threatening data integrity. Primary problems were IP address spoofing and infrastructure saturation due to packet bursts.

In the 3. chapter we introduced challenges of working with darknet data and how these challenges determined our decisions for the architecture. We presented our architecture based on collecting statistics about the traffic data to track down abnormal events and producing appropriate filters to remove these abnormal traffic. Especially memory concerns has been important due to data and key domain dimensions in the statistic collection process.

In the 4. chapter we gave information about the implementation details and the technologies we used to realize decisions we made in the 3. chapter. We also presented our base tool ndpiReader.

Our work is basically a contribution to the improvement of data sanitization process for the Network Telescope carried out by CAIDA. In the 5. chapter with the validation of our contribution we obtained expected results. As we had satisfying results and proved feasibility of dynamic determination and elimination of abnormal traffic, CAIDA will integrate our work in to the Network Telescope structure.

References

[1,4,6,14,15] A. Dainotti, K. Benson, A. King, k. claffy, M. Kallitsis, E. Glatz, and X. Dimitropoulos. Estimating Internet address space usage through passive measurements.2014,URL:http://www.caida.org/publications/papers/2014/passive_ip_space_usage_estimation/

[2] D. Moore, C. Shannon, G. M. Voelker, S. Savage. Network Telescopes: Technical Report. 2004,CAIDA URL:http://www.caida.org/publications/papers/2004/tr-2004-04/tr-2004-04.pdf

[3] M. Bailey, E. Cooke, F. Jahanian, A. Myrick, Sushant Sinha. Practical Darknet Measurement. 2006, 40th Annual Conference on Information Sciences and Systems.URL:http://ieeexplore.ieee.org/abstract/document/4068042/

[5] J. Heidemann, Y. Pradkin, R. Govindan, C. Papadopoulos, G. Bartlett, and J. Bannister. Census and survey of the visible Internet. 2008, Proceedings of the 8th ACM SIGCOMM conference on Internet measurement. URL: https://dl.acm.org/citation.cfm?id=1452542

[7] Network Telescope Project. URL: http://www.caida.org/projects/network_telescope/

[8] CAIDA infosheet, 2016, URL: http://www.caida.org/publications/posters/eps/caida-infosheet-2016.pdf

[9] A. Dainotti, K. Benson, A. King, kc claffy, E. Glatz, X

Dimitropoulos, P Richter, A Finamore, A. C. Snoeren. Lost in Space: Improving Inference of IPv4 Address Space Utilization. 2014, CAIDA. URL: http://www.caida.org/publications/papers/2014/lost_in_space/lost_in_space.pdf

[10] D. Moore, C. Shannon, G. M. Voelker, S. Savage. Network Telescopes: Technical Report. 2004, URL:http://www.caida.org/publications/papers/2004/tr-2004-04/tr-2004-04.pdf

[11] Network Telescope Project. URL: http://www.caida.org/projects/network_telescope/

[12] CAIDA blog.URL:http://blog.caida.org/best_available_data/2012/12/05/syria-disappears-from-the-internet/

[13] M. Tanase, IP Spoofing: An Introduction. 2003, URL:https://www.symantec.com/connect/articles/ip-spoofing-introduction

[16] A. Dainotti, K. Benson, A. King, kc claffy, E. Glatz, X Dimitropoulos, P Richter, A Finamore, A. C. Snoeren. Lost in Space: Improving Inference of IPv4 Address Space Utilization. 2014, CAIDA URL: http://www.caida.org/publications/papers/2014/lost_in_space/lost_in_space.pdf

[17] T. H. Cormen, C. E. Leiserson, R.L Rivest, C. Stein, Introduction to Algorithms. 2009, MIT. ISBN: 978-0-262-03384-8

[18] K. Henderson, T. Eliassi-Rad. Solving the Top-K Problem with Fixed-Memory Heuristic Search. 2009, URL:http://www.eliassi.org/papers/henderson-llnltr10.pdf

[19] nDPI library URL: https://www.ntop.org/products/deep-packet-inspection/ndpi/

[20] uthashUserGuide.URL:https://troydhanson.github.io/uthash/userguide.html

[21] IODAproject.URL:http://www.caida.org/projects/ioda/

[22] P. K. Janert, Gnuplot in Action, Understanding Data with Graphs. 2010, Manning Pub.