GoLCoNDa: Geo-IP Lookup for Campus Network NetFlow Data

Swatesh Pakhare and Deep Medhi
University of Missouri–Kansas City, USA

(an experience paper)

***Abstract*—It is a challenging task for network administrators to properly monitor and manage an institution's incoming and outgoing network traffic patterns. While NetFlow is useful to gather flow-level data, its feature is limited to traditional flow-level information such as the source IP address, destination IP address, source port number, destination port number, and the protocol type. Thus, if we are to understand geographic dynamics of any flow connected to hosts at an institution from the outside world, it is not currently possible with NetFlow. To address the geo-location information of such flows, we developed the tool, GoLCoNDa, for use by campus network administrators. This tool allows the correlation of IP addresses with the geo-location information to visualize the geo-location of incoming and outgoing flows. Our tool handles millions of records quickly.**

***Index Terms*—NetFlow, Campus Network, Geo-IP**

I. Introduction

Institutional networks, such as a campus network, provide a number of services to their different customers [1]. A major challenge for campus network administrators is the ability to categorize incoming and outgoing traffic in an efficient way. In this paper, we present our experience on how to geolocate incoming and outgoing traffic efficiently. This grew out of a conversation with our campus network administrators. It should be noted that besides the campus network administrators, various entities within an academic institution benefit from such information; for example, the Admissions Office would like to know specifically from where around the world users are accessing the university's website. While such knowledge can be obtained by co-relating the web server log with the geo-location data, it serves just this entity on campus. Instead, campus network administrators are interested in all incoming and outgoing traffic. For this reason, they collect NetFlow data at a crucial campus network router.

NetFlow data also allows network administrators to understand the flow-level behavior of the campus network including application and network use, network productivity and utilization of network resources, the impact of changes to the network, network anomaly and security vulnerabilities, and long term compliance. Thus, NetFlow gives network administrators the luxury of knowing who, what, when and where, and how network traffic flows. Understanding network behavior gives an insight about network utilization, and reduces the vulnerability of the network as related to failure while allowing efficient operations of the network. Although network monitoring using NetFlow data enables network administrators to troubleshoot network related problems, it does not readily provide network administrators with crucial geographical information about the host IP.

In this work, we present GoLCoNDa, Geo-IP Lookup for Campus Network NetFlow Data. Our goal in this work is to find geographical information for all IP addresses in an optimal amount of time from campus NetFlow data. The geographical information includes fields like the name of the institution holding that IP, its country, region, street address, city, latitude, longitude, ZIP code, time zone, connection speed, Internet Service Provider (ISP) and domain name, IDD country code, area code, and weather station code, etc. GoLCoNDa is a mining tool that serves two purposes: it processes NetFlow data by co-relating with geolocation information in an efficient manner; secondly, it has a visualization component to display geolocation data on Google maps. We used the concept of scraping websites to obtain geographical data for each IP address in a NetFlow log. For this, we used `xidel` [2], a tool for scraping HTML pages. `xidel` takes URL and HTML tags as its argument and extracts the content between the HTML tags that are passed to it. Therefore, we can automate the process of checking the website that has geolocation information for IP addresses using `xidel`. We observed that GoLCoNDa is efficient in processing millions of IP addresses, which we will report on later.

There are a few related works worth mentioning. In [3], the target IP's geolocation is found using MaxMind GeoLite and adding it to NetFlow data before sending it to the NetFlow collector. This allows the network administrators to filter, aggregate and generate statistics based on the geolocated data. Our work is somewhat related to [4] as we also have a web application using Google maps API using NetFlow data. However, there is one significant difference as we focus on obtaining geographical information for a huge set of IP addresses in a short amount of time and then displaying it on Google maps in the form of markers.

II. GoLCoNDa: Approach and Methods

We first briefly review NetFlow data. A NetFlow-enabled router, on processing packets that traverse through it, identifies and records flow level information. Such a record for each

978-3-901882-89-0 @2017 IFIP

flow contains critical information like the source IP address, destination IP address, source port, destination port, layer 3 protocol type, class of service, and router or switch interface. Our institution's core routers collect over 40 GB of NetFlow data per day that contain millions of flow records—this volume is challenging to handle for the network administrators.

For an IP address, we are interested in obtaining geographic information such as 1) continent, country, capital, state and city's name, 2) organization and ISP's name 3) AS number, host name and name-servers, and 4) latitude and longitude of country, continent, and city. Based on our initial investigation, we found the website, www.ip-tracker.org, that can return such geographic information for an IP address, to be the best suited for GoLCoNDa. We used the concept of scraping this website to obtain geographical data for each IP address; for this, we found `xidel` to be an ideal tool for scraping HTML pages. `xidel` takes URL and HTML tags as its argument and extracts the content between the HTML tags that are passed to it. Another important tool used in our work is GNU's parallel command. Using t heparallel command, `xidel` can send multiple HTTP requests to website that is to be scraped. We then developed a controller, which is a wrapper function, to process millions of IP addresses. The controller, after receiving geographical data, further processes it to produce comma separated geographical data. These comma separated values are first stored in intermediate csv file and then copied into the database using Postgres COPY command for later use.

In addition to IP address processing, GoLCoNDa includes a web application that visually displays the geographical data of IP addresses on Google maps in the form of markers. We also incorporated a date picker, time slider, and drop down menus in this application for near real time experience. These components act as filters and help to curb cluttering of markers on Google maps.

The basic functionality diagram of our overall approach in GoLCoNDa is shown in Fig. 1. Before presenting the primitive operation, we highlight an important point. In our approach, we considered a semantic approach where geographical information belonging to each IP address was retrieved based on DNS queries or a WHOIS lookup. For this reason, we found www.ip-tracker.org to be best suited for our purpose.

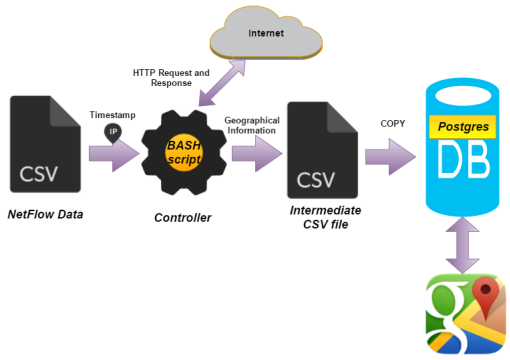

Fig. 1. Basic Block Diagram.

We briefly discuss four methods that we developed for GoLCoNDa. In all the methods, the geographical information belonging to each IP address is obtained by web scraping websites using `xidel`. Note that besides www.ip-tracker.org, we also accessed www.arin.net; the latter was used to obtain the IP prefixes. Henceforth, to avoid confusion between different IP lookups, we define two terms:

- ARIN lookup: IP lookup performed to get IP prefixes from www.arin.net
- Main lookup: IP lookup performed to download geographical data from www.ip-tracker.org

Method 1

In the first method, we started by extracting the IP addresses from the NetFlow data. Note that, these IP addresses were external to University of Missouri-Kansas City. Once the IP addresses are extracted from the NetFlow data, `xidel` was used to download the required data by web scraping www.ip-tracker.org. The downloaded data was further processed by the controller to obtain comma separated values (csv) and stored in an intermediate csv file. The content of this csv file was then copied into the database by using Postgres COPY command.

The advantage of this method is that it was easy to implement. The disadvantage is that it is a naive method; all IP addresses belonging to same IP prefix are looked up repeatedly resulting in downloading redundant information, and thereby, increase the processing time.

Method 2

The framework of Method 2 is similar to Method 1 except that the main lookup is performed based on an IP prefix instead of a host's IP address. This method is also easy to implement. Compared to Method 1, this method has no redundant look ups sinc for IP addressing belonging to the same IP prefix is looked up once. The disadvantage is that all HTTP requests to www.arin.net and www.ip-tracker.org are performed sequentially.

Method 3

Method 3 is different from the first two methods. We started first by extracting the IP addresses and latest timestamp associated with them. The timestamp field in the NetFlow data recorded the time and date when the request was made destined to the UMKC network from outside and vice-versa. When the controller received all IP addresses, it performed ARIN lookups first to get all IP prefixes and then the main lookups were performed to gather all geographical data. Meanwhile, the timestamps were preserved during both ARIN and main lookups. The timestamps preserved during main lookups were then appended to the geographical information downloaded for each IP prefix.

We introduced a 'type' parameter in this method. Initially, when the controller received all IP addresses, it tagged each IP address as either a source or destination depending upon the field from which it is picked. An IP address may also appear either as a source or as a destination for different

NetFlow records. In this case, the IP address was tagged as 'both'. Once the tagging was completed, an ARIN lookup was performed to obtain an IP prefix. Note that, the previous tags associated with IP addresses are now associated with IP prefixex. The controller then finds and replaces the 'type' parameter to 'both' for IP prefixes that were tagged as source and destination. Once this complex process is completed, a main lookup is performed to obtain all geographical data. The timestamps were preserved during the entire process and then finally appended to the geographical data.

In the first two methods, the bottleneck was sending http requests to www.ip-tracker.org and www.arin.net in sequence. We overcame this problem by sending multiple requests at once using GNU's parallel command. The parallel command works based on the number of threads in the computer hardware. The higher the number of threads, a higher degree of parallelism is achieved. Detailed information about the parallel command can be found at [5].

For storing data in the database, the data is first stored in an intermediate csv file and then by using Postgres COPY command, the contents of the intermediate csv file were recorded in the database.

The advantages of this method are: 1) No redundancy as main lookup is performed based on IP prefixes, and 2) HTTP requests are sent in parallel. The disadvantage is an increase in complexity as timestamps and tags that belong to each IP address/IP prefix are preserved throughout the ARIN lookup and main lookup.

Method 4

Method 4 is similar to Method 3 except that it was developed keeping in mind that the database is already filled with geographical data of the IP prefixes ahead of time for records already processed. That is, method 4 is an incremental method.

Comparison of the Methods

In Table I, we present a high level comparison of the four methods. As the main lookup in Methods 3 and 4 is performed based on IP prefixes, we overcome the problem of downloading redundant information multiple times for IP addresses belonging to same IP prefixes. This improved the efficiency of Methods 3 and 4. In addition, Methods 3 and 4 are more optimized than Methods 1 and 2 because HTTP requests are sent in parallel and not sequentially. Since, we introduced tagging of IP addresses and IP prefixes in Methods 3 and 4, they are more complex as compared to Methods 1 and 2. Moreover, the complexity of Methods 3 and 4 increases when the tags and timestamps are preserved during the ARIN lookup and main lookup.

Visualization of GeoLocation Information

Besides the Geo-IP lookup function, GoLCoNDa also has a visualization front end to display processed data. Once the database is filled with geographical information, we display this information on Google maps in the form of markers. The markers, representing geographical data belonging to each IP, have associated latitude and longitude information. We incorporated four filters in this application; they are: date picker, time slider, drop down menus based on continent's name, and 'type' field.

1) The date picker and time slider are used to filter markers based on date and time. With these filters, we can see IP addresses that were accessed on specific dates in a particular time range.
2) The drop down menu with continents allows us to filter markers based on the continent's name. The drop down menu with 'type' field enables us to filter markers depending upon whether the IP address is either the source, destination, or both.

The use of filters allows us to curb the cluttering of markers on Google maps. Also, it helps to visualize data with a near real time experience. In Fig. 2, a screenshot is shown (just for Europe) where the information window consists of the IP prefix, city location, ISP's name, and type field.

TABLE I
SIMILARITIES AND DIFFERENCES BETWEEN METHODS

Properties	Method 1	Method 2	Method 3	Method 4
ARIN Lookup	No	Yes	Yes	Yes
Parallelism	No	No	Yes	Yes
Preserving timestamps and tags	No	No	Yes	Yes
Redundant Geographical information	Yes	No	No	No
Efficient	No	No	Yes	Yes. Better than Method 3
Complex	No	No	Yes	Yes. More complex than Method 3

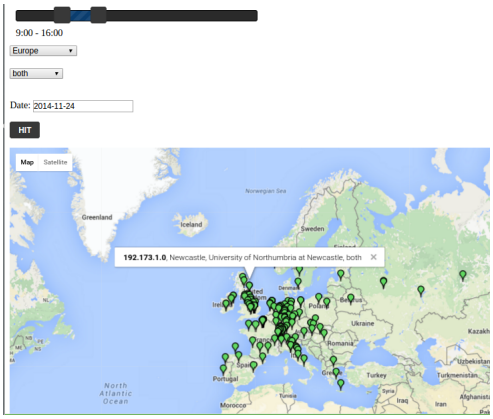

Fig. 2. Google map:Europe

III. RESULTS

Table III represents different sizes of NetFlow data in terms of time duration and number of NetFlow records. Two different

TABLE II
HARDWARE SPECIFICATIONS

Specifications	1 core VM	4 core machine
Processor	Intel(R) Core(TM i5-2410M CPU 2.30 GHz)	8x Intel(R) Core(TM i7-4700HQ CPU 2.40 GHz
Memory	12 GB	8 GB
OS	Ubuntu 14.04.3 LTS	Ubuntu MATE 16.04 LTS

TABLE III
NETFLOW DATA IN TERMS OF HOURS AND RECORDS

DataSet	NetFlow Data (GB)	Duration (hours)	Number of records $\times 10^6$
1	0.22	0.25	1.51
2	6.00	3.00	44.48
3	15.00	6.00	104.23
4	27.00	12.00	182.83
5	42.00	24.00	288.00

compute machines were used for this work; the specifications of the 1 core virtual machine (VM) and the quad core machine that we used to capture the results are shown in Table II. We conducted three categories of studies as follows:

1) **Category 1**: Processing of 222 MB of NetFlow data and retrieving geographical information using Methods 1, 2, and 3.
2) **Category 2**: Processing of 6 GB, 15 GB, 27 GB and 42 GB of NetFlow data using Method 3.
3) **Category 3**: Processing of 42 GB of NetFlow data with Method 4, given that 27 GB of NetFlow data were already processed.

For Category 1's processing 222 MB of NetFlow data, we found that Method 1, Method 2, and Method 3 using 1 core VM took 690.60 min, 272.20 min, and 33.05 min, respectively. This shows the efficiency of Method 3 compared to the first two methods. Then, we applied Method 3 to larger data sets in Category 2 that showed further gain, as expected, with a 4 core machine; see Table IV. Using the incremental approach of Method 4, we can further reduce time as shown in Table V.

TABLE IV
PROCESSING TIME OF METHOD 3 FOR DIFFERENT NETFLOW DATA SIZES (CATEGORY 2)

Data Set	Time (min): 1 core VM	Time (min): 4 core Machine
2	47.16	7.60
3	74.00	12.00
4	175.01	27.42
5	291.10	42.63

TABLE V
METHOD 3 VS METHOD 4 ON DATA SET-5 (CATEGORY 3)

Method	Time (min): 1 core VM	Time (min): 4 core machine
3	291.10	42.63
4	235.80	31.56

GoLCoNDa also has additional features such as obtaining the top ten ISPs and AS from where the flows to and from are observed. Furthermore, we can plot the geographic distribution on the basis of different continents; see Fig. 3. We plan to add a feature to display based on each country.

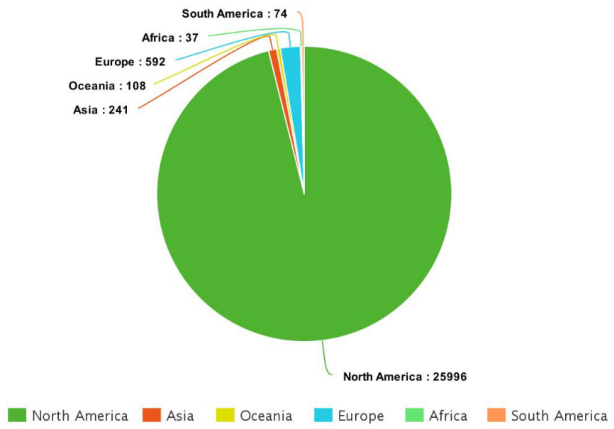

Fig. 3. Pie chart representation for continents

IV. CONCLUSION

In this paper, we report on our experience processing NetFlow data for a campus network for geo-location information. While our approach is fairly straight forward, it is surprisingly efficient in handling millions of NetFlow records. Furthermore, a visualization tool was developed to give near real time experience by displaying geographical data on Google Maps in the form of markers. This tool was found to be useful by the campus network administrators.

ACKNOWLEDGEMENT

We thank Frank Magrone and Andy Goodenow of UMKC Campus Information Services for their input. We also thank Shuai Zhao for extracting the raw NetFlow data used in this paper. This work is partially supported by National Science Foundation Grant # 1541455.

REFERENCES

[1] S. Zhao, K. Leftwich, M. Owens, F. Magrone, J. Schonemann, B. Anderson, and D. Medhi, "I-CaN-MaMa: Integrated campus network monitoring and management," in *Proc. of 2014 IEEE Network Operations and Management Symposium (NOMS)*, Krakow, Poland, May 2014.
[2] B. Zander, "Xidel - html/xml/json data extraction tool." [Online]. Available: http://www.videlibri.de/xidel.html
[3] P. Celeda, P. Velan, M. Rabek, R. Hofstede, and A. Pras, "Large-scale geolocation for netflow," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, 2013, pp. 1015–1020.
[4] R. Hofstede and T. Fioreze, "Surfmap: A network monitoring tool based on the google maps api," in *2009 IFIP/IEEE International Symposium on Integrated Network Management*, 2009, pp. 676–690.
[5] O. Tange, "Gnu parallel - the command-line power tool," *;login: The USENIX Magazine*, vol. 36, no. 1, pp. 42–47, Feb 2011.