Programming Assignment: BGP Data Analysis

CS5229 Advanced Computer Networks

1 Assignment Overview

The main objective of this programming assignment is for you to obtain a more concrete understanding of the Internet ecosystem by analyzing the Border Gateway Protocol (BGP) data. In a nutshell, the assignment contains three parts:

- **Elementary part (20%):** You need to download a BGP dataset from website, extract BGP paths using `BGPDump` tools, and construct the Internet topology graph from these BGP paths.
- **Intermediate part (50%):** You need to analyze the business relationships between different autonomous systems (ASes), and then classify these ASes into multiple categories.
- **Advanced part (30%):** On reaching this stage, you should have got a pretty clear understanding of the BGP data, AS relationships, and the AS taxonomy. Based on these background knowledge, you need to further analyze the ten-year's evolution of the Internet topology.

You are required to finish all the three parts step by step, following the instructions listed below. You may implement programs to solve tasks using either `C/C++`, `Python`, or `Java`.

2 Elementary Part (20%)

The BGP dataset contains a collection of *AS paths* describing the Internet inter-domain routing information. It is publically accessible at **RIPE** and **RouteViews** project websites. We suggest using BGP datasets from at least two vantage points to achieve a better view of the AS relationships. For your convenience as well as ease of grading, you can combine and use the provided data collected in July 2008 from two vantage points **LINX** and **SFINX**. The raw dataset is stored in MRT format, a compressed format to store BGP table data. To obtain human-readable data, you need to process the dataset using the **bgpdump** tool. You are supposed to download the latest **bgpdump-1.4.99.15**, decompress it, and compile it into executables. After that, you can extract the human-readable data by executing: `./bgpdump datasetname`. A sample dataset you may obtain is shown as follows:

```
TIME: 01/01/14 00:00:00
TYPE: TABLE_DUMP_V2/IPV4_UNICAST
PREFIX: 1.23.65.0/24
SEQUENCE: 416
FROM: 4.69.184.193 AS3356
ORIGINATED: 12/19/13 20:30:26
ORIGIN: IGP
ASPATH: 3356 6453 4755 45528
```

```
NEXT_HOP: 4.69.184.193
MULTI_EXIT_DISC: 0
COMMUNITY: 3356:3 3356:22 3356:86 3356:575 3356:666 3356:2011

TIME: 01/01/14 00:00:00
TYPE: TABLE_DUMP_V2/IPV4_UNICAST
PREFIX: 1.23.65.0/24
SEQUENCE: 416
FROM: 216.218.252.164 AS6939
ORIGINATED: 12/27/13 10:34:12
ORIGIN: IGP
ASPATH: 6939 1299 6453 4755 45528
NEXT_HOP: 216.218.252.164
```

Throughout this assignment, you may only need to use the AS paths, so the dataset can further be filtered with the command: `./bgpdump datasetname | grep ASPATH > destfile`. A sample dataset after filtering looks like:

```
ASPATH: 13129 3549 701 80
ASPATH: 3549 701 80
ASPATH: 4777 2497 701 80
ASPATH: 7018 701 80
ASPATH: 3257 701 701 80
ASPATH: 4608 1221 4637 701 80
ASPATH: 2914 701 80
ASPATH: 513 209 701 80
```

An AS path may contain `AS_SET`, which integrates several ASes into one set. For example, the path $(a, b, c, \{d, e, f\}, h)$ contains an `AS_SET` with element d, e, f. You should discard all AS paths containing such `AS_SET`. Another technique used in AS path is AS prepending, that is, an AS may appear multiple times in a single path. AS prepending is used for engineering purpose and you can directly compress such path by removing duplicated ASes. A BGP dataset may also contain multiple duplicated AS paths. Such duplication should also be removed.

- **Task 1 (10%): Preprocess BGP dataset.** You are required to implement a program that removes `AS_SET` and AS prepending in the BGP dataset, and then reports the number of distinct ASes and AS paths. The input dataset is stored in a file called "bgpdata", and the output should be the preprocessed BGP dataset, along with the number of ASes and the number of AS paths in it.

  An example of input file is:

  ```
  ASPATH: 5413 3209 55410 38266 {38266}
  ASPATH: 7660 4635 1273 55410 38266 38266
  ASPATH: 3130 2914 1273 55410 38266 {38266}
  ASPATH: 293 6939 50384 31200 31200 {50923,65011,65014,65200}
  ASPATH: 31200 31200 {50923,65011,65014,65200,65500} 2342
  ASPATH: 13030 50384 31200 31200 50923
  ASPATH: 8492 9002 2914 7018 32328 {32786}
  ASPATH: 7660 4635 1273 1273 55410 38266 38266
  ```

  An example of output file is:

```
7660 4635 1273 55410 38266
13030 50384 31200 50923
8492 9002 2914 7018 32328 32786
Number of ASes: 15
Number of AS paths: 3
```

After preprocessing the BGP dataset, you can attempt to construct an *AS graph* capturing the Internet topology. Here, we give a formal definition of AS graph.

- **AS graph.** An undirected graph where the node set consists of ASes and the edge set consists of AS pairs that exchange traffic with each other.

For the ASes along any AS path, they are connected by undirected edges in the AS graph. Such undirected edges linking neighboring ASes also represent certain kinds of business relationships between the ASes. For example, in the path $(AS13129, AS3549, AS701, AS80)$, we have four nodes connected with three undirected edges: $AS13129 \leftrightarrow AS3549$, $AS3549 \leftrightarrow AS701$, $AS701 \leftrightarrow AS80$. A large graph can be constructed using the whole BGP dataset. An important metric in the constructed graph is *node degree*, which is the number of edges connected to a certain node. The node degree of an AS in the AS graph usually reflects the scale of the corresponding AS. The constructed graph can help us further analyze the AS relationships as well as other characteristics.

- **Task 2 (10%): Construct an AS graph.** You are required to implement a program for constructing an unweighted and undirected graph capturing the Internet topology from the BGP dataset. The program input is the filtered BGP dataset you obtained in Task 1, and the output of the program should be the top-ten ASes with the largest node degrees.

  An example of input file is:

  ```
  7660 2516 3356 7018 32328 32786
  386 7018 32328 32786
  293 6939 1299 7018 32328 32786
  7018 32328 32786
  5413 1299 7018 32327 32786
  ```

  An example of output file is:

  ```
  7018
  1299
  32786
  ... (Omitted here.)
  ```

3 Intermediate Part (50%)

At this stage, you will explore the business relationships taxonomy of the ASes in the Internet topology. The two fundamental papers are by Gao et al. [2] and Subramanian et al. [3]. You can begin this stage with analyzing the relationships between AS pairs. Given a collection of AS paths, you are required to figure out the relationships between any of the two directly connected ASes. Generally, there are four types of relationships: *provider-to-customer (p2c)*, *customer-to-provider (c2p)*, *peer-to-peer (p2p)*, and *sibling-to-sibling (s2s)*. The underlying assumption for interring

AS relationships is that each AS sets up its export policies according to its relationships with neighboring ASes. The key idea for AS relationship analysis is to annotate AS graph based on the *Valley-Free* property: after traversing a provider-to-customer or peer-to-peer edge, the AS path cannot traverse a customer-to-provider or peer-to-peer edge. All the AS relationships are captured in an *annotated AS graph.* Here we give the formal definition of an annotated AS graph.

- **Annotated AS graph.** A partially directed graph where each edge is annotated with the AS relationships and only edges between providers and customers are directed.

An example of an annotated AS graph is shown in Figure 1.

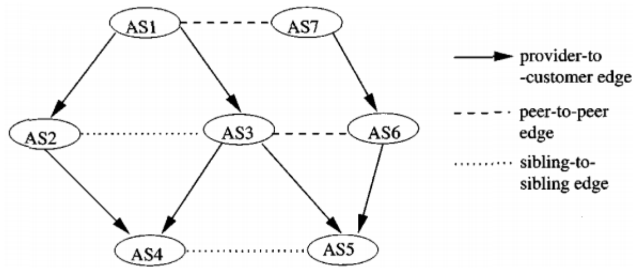

Figure 1: An example of an annotated AS graph [2].

The algorithm framework for annotating the AS graphs is presented in Gao's work [2]. For detailed descriptions, you are encouraged to study the original research paper. Major components of the algorithm (copied from the original paper [2]) are presented in Alrogithm 1. Note that there is a typo in the algorithm of the original paper (line 3 of Phase 3).

By using the above algorithm, you could infer the sibling-to-sibling, provider-to-customer, and customer-to-provider relationships between the ASes. Now you need to further infer the peer-to-peer relationship. An AS pair may have peering relationship if the sizes of these two ASes do not differ significantly. The algorithm is summarized in Algorithm 2.

- **Task 3 (25%): Annotate AS graph.** You are required to write a program that takes as input a collection of preprocessed BGP paths obtained in Task 1 and generates all the AS pair relationships encoded in the dataset. To validate the result, we will randomly pick AS pairs and check whether the generated peer relationships are correct.

  An example of input file is:

```
7660 2516 3356 7018 32328 32786
286 7918 32328 32786
293 6939 1299 7018 32328 32786
7018 32328 32786
5613 1299 7018 32327 32786
```

  An example of output file is:

```
7660 2516 c2p
2516 3356 c2p
3356 7018 p2p
7018 32328 p2c
... (Omitted here.)
```

Algorithm 1 AS graph annotation algorithm

Phase 1: Compute the degree for each AS

```
1: for each AS path $(u_1, u_2, ..., u_n)$ in routing tables do
2:     for each $i = 1, ..., n-1$ do
3:         $neighbor[u_i] = neighbor[u_i] \cup \{u_{i+1}\}$
4:         $neighbor[u_{i+1}] = neighbor[u_{i+1}] \cup \{u_i\}$
5:     end for
6: end for
7: for each $AS\ u$ do
8:     $degree[u] = |neighbor[u]|$
9: end for
```

Phase 2: Count the number of routing table entries that infer an AS pair having a transit relationship

```
1: for each AS path $(u_1, u_2, ..., u_n)$ do
2:     find the smallest $j$ such that $degree[u_j] = max_{1 \le i \le n} degree[u_i]$
3:     for $i = 1, ..., j-1$ do
4:         $transit[u_i, u_{i+1}] = transit[u_i, u_{i+1}] + 1$
5:     end for
6:     for $i = j, ..., n-1$ do
7:         $transit[u_{i+1}, u_i] = transit[u_{i+1}, u_i] + 1$
8:     end for
9: end for
```

Phase 3: Assign relationships to AS pairs

```
1: for each AS path $(u_1, u_2, ..., u_n)$ do
2:     for $i = 1, ..., n-1$ do
3:         if $(transit[u_{i+1}, u_i] > L$ and $transit[u_i, u_{i+1}] > L)$
           or $(transit[u_i, u_{i+1}] \le L$ and $transit[u_i, u_{i+1}] > 0$
           and $transit[u_{i+1}, u_i] \le L$ and $transit[u_{i+1}, u_i] > 0)$ then
4:             $edge[u_i, u_{i+1}]$ = sibling-to-sibling
5:         else if $transit[u_{i+1}, u_i] > L$ or $transit[u_i, u_{i+1}] = 0$ then
6:             $edge[u_i, u_{i+1}]$ = provider-to-customer
7:         else if $transit[u_i, u_{i+1}] > L$ or $transit[u_{i+1}, u_i] = 0$ then
8:             $edge[u_i, u_{i+1}]$ = customer-to-provider
9:         end if
10:    end for
11: end for
```

Algorithm 2 Algorithm for annotating peer-to-peer relationship

Phase 2: Identify AS pairs that can not have a peering relationship

```
1: for each AS path $(u_1, u_2, ..., u_n)$ do
2:     find the smallest $j$ such that $degree[u_j] = max_{1 \le i \le n} degree[u_i]$
3:     for $i = 1, ..., j-2$ do
4:         $notpeering[u_i, u_{i+1}] = 1$
5:     end for
6:     for $i = j+1, ..., n-1$ do
7:         $notpeering[u_i, u_{i+1}] = 1$
8:     end for
9:     if $edge[u_{j-1}, u_j] \ne$ sibling-to-slibling and $edge[u_j, u_{j+1}] \ne$ sibling-to-sibling then
10:        if $degree[u_{j-1}] < degree[u_{j+1}]$ then
11:            $notpeering[u_j, u_{j+1}] = 1$
12:        else
13:            $notpeering[u_{j-1}, u_j] = 1$
14:        end if
15:    end if
16: end for
```

Phase 3: Assign peering relationships to AS pairs

```
1: for each AS path $(u_1, u_2, ..., u_n)$ do
2:     for $i = 1, ..., n-1$ do
3:         if $notpeering[u_j, u_{j+1}] \ne 1$ and $notpeering[u_{j+1}, u_j] \ne 1$
           and $degree[u_j]/degree[u_{j+1}] < R$ and $degree[u_j]/degree[u_{j+1}] > 1/R$ then
4:             $edge[u_j, u_{j+1}]$ = peer-to-peer
5:         end if
6:     end for
7: end for
```

The previous task helps to generate a directd graph that describes the relationships between ASes. AS A has a directional link to AS B if B is a customer of A. Also, AS A and AS B share a bidirectinal link if they have *peer-to-peer (p2p)* or *sibling-to-sibling (s2s)* relationship. We can further explore the AS taxonomy using the technique proposed in section V of Subramanian et al.'s work [3]. This paper also proposes an alternate method to infer AS relationship, which is not used in this assignment because the method is similar to Gao's work [2]. Here, you are required to classify the ASes into several categories, including *stubs*, *regional ISPs*, and *cores*. The classification method is simplified as follows: all the leaf nodes, defined as having no peers or downstream customers, in the AS graph obtained in the previous task are classified into the *stub* category; after removing the stub nodes in the AS graph, we have a residual AS graph. Then, we keep removing leaf nodes in the remaining graph until there is no leaf node. The ASes removed through this process are classified as *regional* ISPs. All the remaining ASes in the residual AS graph are classified as *cores*. The AS cores can be further classified into *dense cores*, *transit cores*, and *outer cores*. The algorithm is still quite simple: any AS core with no upstream provider are classified dense core; any AS core that peers with the dense core and each other are classified transit cores; all the remaining AS cores are classified as outer cores.

- **Task 4 (25%): Classify ASes.** You are required to implement a program for classifying the ASes in an AS graph into multiple classes, including stubs, regional ISPs, dense cores, transit cores, and outer cores. The input of the program is the AS graph and the AS relationships you got from previous tasks, and the output of the program is the AS classification results. To validate the result, we will randomly pick ASes and check whether the generated AS classification are correct.

  An example of the output file looks like:

  ```
  7660 stub
  2516 regional ISP
  3356 outer core
  7018 transit core
  ... (Omitted here.)
  ```

4 Advanced Part (30%)

The Internet ecosystem keeps evolving all the time [1]. At this stage, you are encouraged to figure out the evolutional trends of the Internet ecosystem over the last ten years, and explore where the Internet is heading. You can use the BGP data collected from **RIPE** in **2001**, **2006**, and **2011**. Table 1 and Figure 2 present the inter-connectivity of ASes across levels and the distribution of the size for different level ASes in a certain year. Details can be found in Subramanian's work [3]. You are required to generate such table and CDF for the year 2001, 2006 and 2011, and analyze evolutional trends. There are also some open questions below. You can choose some or all of them to answer based on your understanding of the network ecosystems.

- You could measure the growth of the number of ASes and inter-AS links. Using ten datasets from the last continuous ten years, you can count the number of ASes and AS links for each year and plot the trend. You need to print out the top-ten ASes with the highest node degree for each year. You also need to count the average length of the AS paths and measure how the BGP connectivity changes.

Table 1: Inter-connectivity of ASes across levels.

Level	0	1	2	3	4
0	312	626	1091	958	6732
1	183	850	1413	665	3373
2	29	145	1600	543	3752
3	0	0	0	212	2409

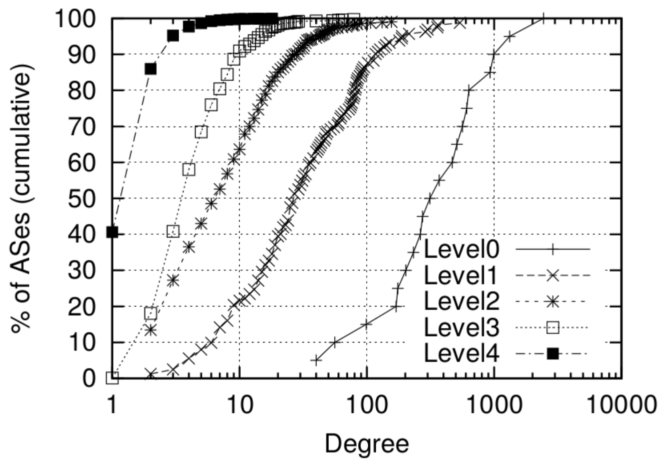

Figure 2: Cumulative distribution of AS degree by level.

- You could explore the evolution of AS types. You have already learned how to capture the AS taxonomy in the intermediate part. Here, you can use more advanced techniques to classify the ASes into several categories: enterprise customers, small transit providers, large transit providers, content providers. Several works have proposed algorithms for classifying AS types. Here, any classification approach is allowed. What you need to do is to figure out how each AS types evolve during the last ten years (e.g., how the percentage of each type changes?).
- You could investigate the evolution of AS relationships. Run the program you have implemented for AS relationship annotation, and explore how the peer relationships evolve throughout the last ten years. You may measure how the number of providers and customers changes in each year, and capture the high-level evolution trend.
- You could figure out how the Internet ecosystem changes geographically. You can check the nationality of each AS based on the dataset you downloaded from the website. You can observe how the AS topology of each continent changes throughout the last ten years, and which continent grows fastest, and how the ecosystem of each continent influences the world.
- Can you conclude the growth trend of the Internet ecosystem? Any reasonable inspiration is encouraged.

5 Submission

Please submit your assignment **BEFORE 23:59 13th November, 2015**. Submissions after the deadline will not be graded. The submission should include source code and program report. Please add comments in the source code and explain the implementation details in the program report. The source code is supposed to be compiled and executed directly on the Tembusu cluster. Any corresponding compilation-related files such as `makefile` or `pom.xml` should also be submitted. Add your name and matric number as a comment on all files you submit. Please zip all your documents and name the zip file as "*MatricNumber*.zip". Submissions in any other formats will be penalized. You may discuss and consult with your TA (Wang Zixiao zixiao@comp.nus.edu.sg), but plagiarism is strictly prohibited.

References

[1] A. Dhamdhere and C. Dovrolis. Ten years in the evolution of the internet ecosystem. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 183–196. ACM, 2008.

[2] L. Gao. On inferring autonomous system relationships in the internet. *IEEE/ACM Transactions on Networking*, 9(6):733–745, 2001.

[3] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz. Characterizing the internet hierarchy from multiple vantage points. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 618–627. IEEE, 2002.