Degree Centrality in Network Measurement and Analysis

Carlos Mendoza
Department of Computer Science
University of New Orleans
October 29th, 2022

Abstract
The internet consists of many large groupings of networks called autonomous systems (AS). These ASes communicate and interact with each other in many ways. This can be seen through datasets that show each link between a variety of ASes as well as the types of relations ASes form and the source of these links. This report introduces the concept of ASes, explains important details about ASes, compares the different ways dataset graphs can be examined, and analyzes a dataset from October 1st, 2022 in order to show what a typical dataset usually consists of.

Keywords: degree, betweenness, eigenvector, centrality, autonomous systems, AS, ASN, node, degree, distribution, networks

Introduction

The internet can best be described as what is currently the ideal way of communicating with others around the world. From sharing files to conversations, the internet is helpful from both a business and casual point of view. Access to the internet is made possible with networks. While the internet itself is one large global network, it is also made up of much smaller networks capable of hosting websites, web apps, web servers, and more services that provide privacy and confidentiality between many different parties.

According to AWS's explanation on computer networking, a network relies on nodes and links which can be either physical (e.g. modem, switch, etc.) or virtual depending on how the network is constructed [1]. Another key component to a network's construction is its topology, or the way the nodes and links are organized. Below is a figure showing the different types of topologies [2]. The most commonly used topology seen here is the star topology since it involves a central node that increases the performance and overall running time of each connection made.

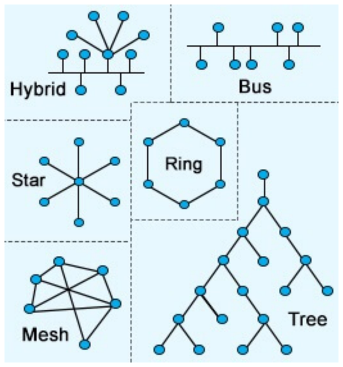

Fig. 1: Different Types of Network Topologies

A particularly large group of networks involving several internet protocols, or IPs, is known as an autonomous system (AS) [3]. Autonomous systems contain autonomous system numbers (ASN) which are used to identify an AS. They can be found in datasets along with the relations and types of relations that each AS has been associated with within the time frame specified in the dataset-- in the case of this assignment, the dataset appears to consist of all the AS relationships of the month. ASes-- at least in this dataset-- can connect to each other through either a provider-customer relationship (p2c) or a peer-to-peer relationship (p2p). In p2p relations, both ASes have the same permissions during the communication between the two. A p2c relation functions similarly to a client server relationship [4]. The provider in a p2c relationship is an ISP.

ASes are assigned into different tiers based on the connections that they make. The tiers that will be focused on in this report are tier 1 and stub ASes. A tier 1 AS is an autonomous system that never appears as a customer. Because it is never a customer, tier 1 ASes never purchase permission to connect to other networks; it only ever provides its services or peers with other tier 1 ASes [6]. Stub ASes can be seen as the opposite of tier 1 ASes; they never appear as providers. Though they aren't related to any of the objectives done for this report, tier 2 ASes are often preferred and can be defined as tier 1 ASes that, unlike tier 1 ASes, can be customers but only when they are the customer of a tier 1 AS [7].

The goal of this report is for both the reader and myself to achieve a better understanding of analyzing networks and degree centrality. Given a dataset from October 10th, 2022 that contains 463,140 AS relationships, I will make statistical observations based on the results I received by performing the following steps through a java file named asnCounter.java:

Step 1: Show how many unique ASNs are in the dataset.
Step 2: Show how many unique links are in the dataset.
Step 3: Show how many unique p2p and p2c relations are in the dataset.
Step 4: Find the top 10 largest ASN in terms of connection size.
Step 5: Find the top 10 largest provider ASNs in terms of connection size.
Step 6: Find the top 10 largest p2p ASNs in terms of connection size.
Step 7: Find the top 10 largest tier 1 ASNs in terms of connection size.
Step 8: Show how many stub ASes are in the dataset.

Related Work

There are many techniques when it comes to examining a graph of ASNs as a dataset. Degree centrality is what was used to calculate and compare the values in steps 4-7. Degree centrality consists of counting the links that an AS makes with the exception of duplicate links (e.g. ASN2-ASN1 is not counted if ASN1-ASN2 was already included). You're effectively counting the edges of a node on a graph which is a great way of showing how important a node, or an AS in this case, is-- you can show how central it is to the graph or network of autonomous systems, much like the central network in a star topology [8].

Another graph metric that could've been used was betweenness centrality. Rather than counting the edges, betweenness centrality consists of counting the number of shortest paths for each AS [9]. While this technique is useful for finding the most central AS, it takes slightly more work to implement and would have a longer overall running time (running steps 1-8 with degree centrality took several hours due to the length of the text file). This is because you have to save each link that connects to an AS in it, as you would with degree centrality, and then calculate and compare the paths for each AS pair in order to find the shortest paths. Overall, degree centrality is more efficient for finding the centrality of ASes for network analysis.

While eigenvector centrality would also have a longer running time compared to degree centrality due to its additional implementation, it is useful for network analysis. Eigenvector centrality is best described as a combined value consisting of an AS's degree and its adjacent nodes' degrees. The implementation of this analysis technique shouldn't be too difficult as you can create an n dimensional array, with n being the number of unique ASNs, that can resemble the "adjacency matrix" used for calculating the eigenvector values of each node [10]. It should be noted, though, that the results for steps 4-7 would likely be similar to the results I received with degree centrality. This can be evidenced by the fact that the top 10 ASes have a large number of links, giving them a better chance of being adjacent with the other ASes in the top 10 list which would then contribute to a high eigenvector value. There's no telling which nodes are connecting to which as my analysis did not print this information, but the probability of the top 10 centrality lists being similar to the results I received are very high.

Experimental

In order to ensure that my code worked without having to wait a long time for the output due to the length of the dataset, I created a test.txt file (included in the compressed project code folder) with about 40 random AS relations. The code proved to be a success in both the test.txt file and the ASRelationship-2022-10-01.txt file. However, I produced two different outputs (both as text files in the compressed folder) for two different reasons. The first is that I failed to realize that I had to divide the provider ASNs' degrees in half for step 5. I originally didn't do this because I figured that having ASN1 be a customer and ASN2 be a provider is not the same relationship as ASN1 being a provider and ASN2 being a customer. However, upon realizing how high the degree values were and looking back at the handout's "ASN1-ASN2 is the same as ASN2-ASN1" statement from the lab handout, I decided that my output for step 5 had to change. The second reason as to why I had to create another output file was that step 7 changed from "Show all tier 1 ASNs" to "Find the top 10 largest tier 1 ASNs in terms of connection size." The final version of my output can be seen below:

"

```
STEP 1: The number of unique ASNs in the dataset is 74579.

STEP 2: The number of unique links in the dataset is 231570.

STEP 3: The number of unique p2p relations is 157524 and the number
of unique p2c relations is 74046.

STEP 4: Largest ASNs
1.          ASN6939 degree = 4883
2.          ASN174 degree = 3316
3.          ASN3356 degree = 3211
4.          ASN14840 degree = 2300
5.          ASN24482 degree = 2275
6.          ASN51185 degree = 2258
7.          ASN35280 degree = 2157
8.          ASN1828 degree = 2098
9.          ASN61568 degree = 2000
10.          ASN58511 degree = 1665

STEP 5: Largest Provider ASNs
1.          ASN174 degree = 3271
2.          ASN3356 degree = 3176
3.          ASN7018 degree = 1209
4.          ASN1299 degree = 1139
5.          ASN6939 degree = 1032
6.          ASN6461 degree = 1019
7.          ASN3257 degree = 970
8.          ASN2914 degree = 766
9.          ASN701 degree = 687
10.          ASN209 degree = 594

STEP 6: Largest p2p ASNs
1.          ASN6939 degree = 3848
2.          ASN51185 degree = 2256
3.          ASN24482 degree = 2254
4.          ASN14840 degree = 2165
5.          ASN35280 degree = 2126
6.          ASN1828 degree = 2063
```

```
7.          ASN61568 degree = 1923
8.          ASN58511 degree = 1638
9.          ASN199524 degree = 1592
10.          ASN37497 degree = 1580

STEP 7: Largest Tier 1 ASNs
1.          ASN174 degree = 3316
2.          ASN3356 degree = 3211
3.          ASN51185 degree = 2258
4.          ASN37497 degree = 1595
5.          ASN13237 degree = 1232
6.          ASN137409 degree = 1217
7.          ASN1299 degree = 1167
8.          ASN41805 degree = 950
9.          ASN52863 degree = 925
10.          ASN45352 degree = 886

STEP 8: There are 48791 stub ASNs in the dataset.
```
”

Fig. 2: Lab 3 Output

For the first step, I had my code run through the dataset line by line with “|” as a separator in order to easily find ASN1 and ASN2. I then had each individual ASN go through an ArrayList of the unique ASNs, if the current ASN matched anything in the ArrayList, it wasn’t added to the ArrayList-- to prevent duplicates-- otherwise, it’s added to the ArrayList. There ended up being 74,579 unique ASNs.

For the second step, my code ran through each ASN in the dataset (similar to step 1-- all other steps use this implementation from step 1) and appended the two ASNs in the same row in order to create a link. The link was then added to a HashSet. The links didn’t have to be checked for uniqueness since HashSets don’t allow for duplicates. Though, ASN1-ASN2 is not considered a duplicate to ASN2-ASN1. Therefore, I followed the hint from the lab handout and divided the number of links by 2-- I would also go to do this for any other future step involving connections between ASes. As a result, the number of unique links in the dataset was 231,570.

It can be seen that there are roughly 3.1 times more unique links than there are ASNs in the dataset. Considering that there should be more edges than nodes in an active network graph where networks are constantly communicating in a dataset that supposedly occurred over the course of a month, this is a great and expected result.

For the third step, my code created links in a similar fashion to step 2 with the addition of a third column check for each row where it would either be determined as “0” and the link would be added to a p2p HashSet or the third column would be confirmed as “-1” and the link would be added to a p2c HashSet. Both HashSet’s sizes were halved before printing in order to prevent duplicate links. The total number of unique p2p relations were 157,524 and the total number of unique p2c relations were 74,046.

There are roughly 2.1 times more p2p relations than there are p2c relations. This is a fair result as, according to Abdullah Yasin Nur and Mehmet Engin Tozal, both peer ASes in a p2p relationship don't have to purchase the transportation of an IP [12]. It should also be noted that there are as many unique p2c relations as there are unique ASNs. This could imply that most if not all of the unique ASNs were either a customer or provider in at least 1 unique link since duplicate links are not counted. This fact is made a little more clear in the results given with step 7 and step 8.

For steps 4-7, my code first collected the unique ASNs in each step's ArrayList that met the requirements (i.e. checking if the third column of a row is either "-1" or "0" to ensure that the ASN is a provider, peer, or tier-1 depending on the step) and then went through the dataset for each element in the ArrayList to count their instances-- provided the instances meet the requirements for the specified type of AS based on the step. The element in the ArrayList and the value for that specific element divided by 2 (since ASN1-ASN2 is the same as ASN2-ASN1) were then used to construct an object. These objects were placed in an ArrayList made for these objects so that they could then be sorted. Thus, allowing for the ten elements at the end of the object array to be printed as a top 10 largest ASN list by connection size. I should mention that I made a mistake when it came to the sorting algorithm implemented for these four steps. I used BubbleSort as it was easy to implement. However, I failed to remember that BubbleSort has some of the worst running times out of all the sorting algorithms only until after I implemented it and received my output. My code would have had a significantly shorter running time as opposed to the several hours it took for my final code if I had implemented a faster sorting algorithm such as MergeSort. I must also mention that since I used ArrayLists for sorting, I used the Collections class in order to swap any elements that were out of order as I had no familiarity with this class prior to this assignment [11]. The results for these steps can be seen in the table below along with their averages:

Degree Centrality	ASN	Provider ASN	Peer ASN	Tier 1 ASN
1.	**ASN6939** degree = 4883	**ASN174** degree = 3271	**ASN6939** degree = 3848	**ASN174** degree = 3316
2.	**ASN174** degree = 3316	**ASN3356** degree = 3176	**ASN51185** degree = 2256	**ASN3356** degree = 3211
3.	**ASN3356** degree = 3211	**ASN7018** degree = 1209	**ASN24482** degree = 2254	**ASN51185** degree = 2258
4.	**ASN14840** degree = 2300	**ASN1299** degree = 1139	**ASN14840** degree = 2165	**ASN37497** degree = 1595
5.	**ASN24482** degree = 2275	**ASN6939** degree = 1032	**ASN35280** degree = 2126	**ASN13237** degree = 1232
6.	**ASN51185** degree = 2258	**ASN6461** degree = 1019	**ASN1828** degree = 2063	**ASN137409** degree = 1217
7.	**ASN35280** degree = 2157	**ASN3257** degree = 970	**ASN61568** degree = 1923	**ASN1299** degree = 1167
8.	**ASN1828** degree = 2098	**ASN2914** degree = 766	**ASN58511** degree = 1638	**ASN41805** degree = 950
9.	**ASN61568** degree = 2000	**ASN701** degree = 687	**ASN199524** degree = 1592	**ASN52863** degree = 925
10.	**ASN58511** degree = 1665	**ASN209** degree = 594	**ASN37497** degree = 1580	**ASN45352** degree = 886
Average	**2616**	**1386**	**2144**	**1675**

Table I: Steps 4-7 Results and Their Averages

As you can see in the table above, certain ASNs appear in multiple of the top 10 lists. For instance, the three general ASNs with the largest degrees, ASN6939, ASN174, and ASN3356, also appear in the top 10 list of largest provider ASNs. This means that some of the most central ASNs are very profitable providers which can be confirmed by the fact that ISPs seem to handle the most internet traffic [5].

Not only is ASN6939 a top provider, but it is also *the* top peer ASN, meaning that it relies on peering in order to appear among its descendents in order to provide its services [12]. Also, based on the averages of each type of ASN in the table, p2p connections are overall more prominent in the dataset which, like what was mentioned with step 3's results, p2p relationships involve a lack of costs [12].

The top tier 1 ASNs appear to contain provider (ASN174, ASN3356, and ASN1299) and peer (ASN51185 and ASN37497) ASNs. This would make sense seeing that tier 1 ASNs only consist of peers and providers; no customers.

For the eighth and final step, my code copied the ArrayList of unique ASNs from step 1 and removed any of the ASNs that ever appeared as a provider. The result that I received was the size of this new ArrayList: 48,791.

Earlier I mentioned that the number of unique p2c relations and the number of unique ASNs implied that each unique ASN was at least either a customer or provider at one point throughout the dataset. I wanted to confirm this presumption by using the original output from step 7, the fact that there were 26,409 tier 1 ASNs (seen in outputV1.txt-- I used word count to determine the number of ASNs), and the output from step 8, the fact that there were 48,791 stub ASNs. To be a tier 1 ASN means that the ASN never appeared in the dataset as a customer. This would mean that 74,579 - 26,409 (unique ASNs - tier 1 ASNs) = 48,170 of the ASNs in the dataset were customers at some point. Using this same logic, subtracting 74,579 and 48,791 (unique ASNs - stub ASNs) shows that 25,788 of the ASNs in the dataset were providers at some point. Combining these values of guaranteed providers and customers shows that 73,958 out of the 74,579 unique ASNs (99.03%) were in a p2c relationship in this dataset consisting of AS relationships that occured over the course of a month.

Conclusion

In conclusion, network analysis, specifically the analysis of AS relationships, is crucial from a business and computational point of view as it not only shows the most important entities based on the number of connections they make, but it also shows the overall flow of traffic of the internet as a whole. Finding the centrality of the different nodes in a dataset graph can be done in several ways, from simply counting the connections to determining the shortest distance from one node to another. Calculating the degree centralities and the number of certain types of ASes in the provided dataset successfully helped me achieve a better understanding of network analysis and its importance.

References

[1] AWS, “What is Computer Networking?”
[2] Reddy, S. B. (2018, February 20). “Different Types of Network Topologies,” Inst Tools.
[3] J. Hawkinson and T. Bates. “Guidelines for creation, selection, and registration of an Autonomous System (AS)”. RFC 1930, Mar 1996.
[4]] L. Gao, “On Inferring Autonomous System Relationships in the Internet, IEEE/ACM Transactions on Networking”, 2001
[5] Abdullah Yasin Nur, “Analysis of Autonomous System Level Internet Topology Graphs and Multigraphs”, IEEE International Symposium on Networks, Computers and Communications (ISNCC), 2021
[6] Van der Berg, R. (2008, September 2). “How the 'net works: An introduction to peering and transit,” Ars Technica.
[7] Dhliwayo, J. (2019, May 13). “IP transit services: ISP Tiers – Tier 1, tier 2, tier 3,” Fiberguide.
[8] Golbeck, J. (2015). “Degree Centrality,” ScienceDirect.
[9] “Betweenness Centrality (Centrality Measure),” GeeksforGeeks. (2022, July 21).
[10] Golbeck, J. (2013). “Eigenvector Centrality,” ScienceDirect.
[11] “How to Swap Two Elements in an ArrayList in Java?” GeeksforGeeks. (2021, July 1).
[12] Abdullah Yasin Nur and M. E. Tozal, “Identifying Critical Autonomous Systems in the Internet”, Springer Journal of Supercomputing, SI: Cyber Threats against Critical Infrastructure and their Countermeasures, Volume 74, Issue 10, pp. 4965-4985, 2018