Ryerson University

**Department of Electrical, Computer, & Biomedical Engineering**
Faculty of Engineering & Architectural Science

| | |
|---|---|
| **Course Title:** | **Advanced Computer Networks** |
| **Course Number:** | **COE865** |
| **Semester/Year (e.g.F2016)** | **WINTER 2022** |

| | |
|---|---|
| **Instructor:** | **Dr. Jaseemuddin** |

| | |
|---|---|
| *Assignment/Lab Number:* | **Project Report** |
| *Assignment/Lab Title:* | **Routing Control System for Inter-domain Routing** |

| | |
|---|---|
| *Submission Date:* | **Sunday, April 10th, 2022** |
| *Due Date:* | **Sunday, April 10th, 2022** |

| Student LAST Name | Student FIRST Name | Student Number | Section | Signature* |
|---|---|---|---|---|
| **Hasan** | **Ali** | [redacted] | **#2** | **A.H.** |
| **D**[redacted] | **James** | [redacted] | **#2** | **J.D.** |
| **T**[redacted] | **David** | [redacted] | **#2** | **D.T.** |

*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work,an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/senate/current/pol60.pdf

# Table of Contents

# 1. Abstract

The purpose of this project was to explore BGP routing to better understand inter-domain routing. This report will demonstrate the design and implementation of a routing control system which evaluates two specific parameters, bandwidth and cost. Subsequently an optimized path along the network through the given inter-domain routes will be established for various configurations. In addition, the BGP inter-domain routing topic will be analyzed and basic implementations for better understanding will be discussed. Python was used for the implementation of the given topology given in the below figure.

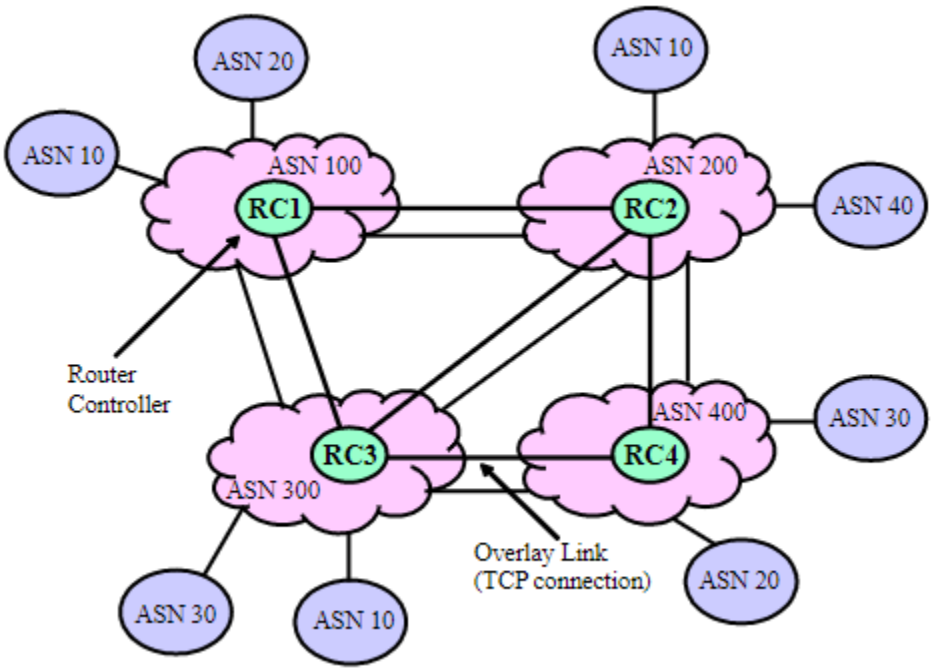


***Figure 1.1.1-Given Topology in Project Manual***

# 2. Introduction

The internet is an extensive computer network that allows people to share information and communicate with anyone around the globe. Computer networks such as the internet are relied upon for everything from work communication to leisure activities. As a result, computer networks as a whole have become ubiquitous in modern society. Networks are made up of a number of interconnected nodes and work by passing information from one computer to another until it reaches its specified destination. In order for these networks to function at their best it is important that messages between the nodes are routed to traverse the best path from their source to destination. The best path for information to travel can depend on what metric is being used to evaluate them. These metrics can include values such as cost, bandwidth, delay, or the number of hops between the source and destination.

The goal of this project is to design and implement a routing control system for inter-domain routing. Using information on link capacity and link rental cost the routing algorithm will find the optimal path from one node to another. There will be a configuration file for every route controller present in the network which contains information such as connected ASNs and the link cost and capacity between them. The routing control system will import necessary information from these configuration files and use a routing algorithm to determine the optimal path. The particular routing algorithm we will use in this project is Dijkstra's algorithm. The composite cost associated with each link will be calculated by dividing the link cost by the link capacity. This value can then be inputted into Dijkstra's algorithm to determine the shortest path between two given nodes. Using a composite cost instead of just using link cost or link capacity allows the routing control system to consider both values when trying to determine the optimal route. This allows for a more balanced approach during path selection and can help break ties in the event of two paths having identical cost or bandwidth. After running the routing algorithm the control system will output the optimal path by printing it to the console.

# 3. Theory

## 3.1 BGP Theory

Inter-domain routing - BGP, this particular routing algorithm helps to route within a domain. Unlike RIP/OSPF (link state and distance vector routing) - the capabilities of these two protocols are limited to within their domain. This specific type of routing is considered to be intra-domain routing (routing inside an autonomous system).

BGP allows the users to be connected on a global level, essentially inter-domain connections (routing between autonomous systems). This type of internet routing, is a domain-to-domain connection, and global connectivity results for a domain to obtain a detailed route to destination subnets, and these are considered subnet routes - e.g. RIP/OSPF.

## 3.2 BGP Relationships

To connect two domains, for example an organization/business connected to its respective service provider - or even service providers connected to each other - Rogers -> Bell (Bell and Rogers customers are able to be connected to each other). This inter-domain connectivity uses border gateway protocol (BGP) and this global routing perspective may be steered routing or steered connectivity. It must be noted that in this topology, every service provider does not share equal breadth of the network - and local ISPs are connected for a small area to regional ISPs. Whereas global ISPs have a point of presence across multiple continents. These ISPs exchange their traffic in a secure site NAP (network access points), within NAP, a secure site where BGP routers are placed.

Another difference between BGP and other routing protocols is the advertisements of the router when calling for traffic. In BGP, sending a prefix is an invitation for the receiver to carry the traffic, a business type of relationship. For instance, two routers Rx and Ry exist, if Rx advertises its x.x.x.x/y to Ry, this means that Rx is practically calling Ry stating - ‘you can send traffic for the destinations within the prefix to us’. Therefore these two routers, Rx and Ry, are considered to hold a business customer relationship - also known as BGP Pairing.

There exists two types of BGP pairing

i. External BGP Pairing

Where two routers are paired together and both routers belong to different ISPs.

ii. Internal BGP Pairing

Where the two routers are paired to each other and both belong to the same ISPs.

## 3.3 BGP Architecture

Through both internal and external pairing, this is how the routes propagate within the domains, external BGP pair and internal BGP pairs have information, but only external BGP possesses the customer/business relationship.

The routing decision is also done based on a relationship, BGP is a policy based routing, whether a prefix that a router should advertise does not depend on connectivity, but rather depends upon the business/customer relationship between the two domains themselves. This proves to be extremely advantageous to both service provider and customer; since there is an ability to choose whether to be transmitting all the traffic, or specific traffic, again this depends upon the policy chosen between the service provider and client in the aforementioned customer/business relations. BGP is heavily based upon policy routing, for example a router, R1 learns a prefix from a domain(customer domain, a group of customers - AKA - Aggregate Route). If this router - R1 decides to propagate this information to its own customers (in maybe a different continent such as Europe) this decision to advance across continents depends upon the policy between the two global ISPs. If the new ISP (destination - Europe) is willing to provide a transit service for customers in the Canadian ISP. If there is no business relationship, the transit service will not be provided.

On the other hand, Forwarding a route, the transit service in the BGP router does not depend on the routing policy, and this decision is a local decision to make within the router interface itself. A BGP router has capability to introduce peer relationships with multiple other BGP routers, such that a router - Rx, is peering with multiple Service Providers, and this parking most likely occurs in the NAT site. To make a routing decision, router Rx, internally processes a routing structure, and only through each interface will it learn every BGP prefix.

## 3.4 Adjacency RIB (Routing Information Base)

Adjacency Routing Information Base acts as an input process received from the base BGP router, and it is a means of external BGP pairing as an input interface. This adjacency routing information base builds all the prefixes received by one router, and puts into its own adjacency routing information base input. This process builds all prefixes received by one router and inputs into its respective RIB-in. For internal routing there exists an extra application in terms of the policy decisions to be made. These differ depending upon the business/customer relationship previously mentioned. In general these are called the input policy engine, BGP decision processes are applied and routing is subjected independently through a decision base process and all the routes that are separate merge into a local RIB (routing information base). A local RIB is used to generate the routing table which the router forms after making local policy decisions. The next step in this process is a filter based output policy exit, for output interface. The BGP attribute in this regard helps the router decide which peers to advertise to and which to ignore, all these issues are resolved with the policy decision previously mentioned. In summary, this process exhibits routes learnt from peers, and then advertised to peers, it resembles a form of flooding, but instead thanks to BGP attributes, output and input interfaces can resolve problems such as cycles and excessive transmission using the majority of the available bandwidth.

## 3.5 BGP Attributes

Path attributes are used to manipulate and make some paths seem longer than they are, in order for the 'shorter' path to be taken. Path manipulation can be achieved by using BGP attributes.

In this project, every green node in the diagram below is considered to be a Local ISP whereas the blue nodes are considered as clients. The local ISPs are considered to be service providers that allow inter-domain connectivity between each client. As mentioned previously - each blue node is considered a client, but for two clients to connect to each other one acts as a host and the other as the client (master/slave relationship). As mentioned previously BGP attributes have the ability to prevent cycles, this is done by preventing an AS from accepting a specific. This is done through the input and output interface, a tracing method is implemented. Another addition of path attributes is the ability to manipulate the distance, this is directly helpful for hop count - the distance the traffic crosses to arrive at the destination. This hop is between domains, and there is a cost associated with each path, not only counting the number of domains crossed but their respective cost of traversing the link itself. These path attributes can be manipulated in order to bloat the number of domains in the shortest path thus increasing the cost of the path. For example the NAP (network access point) is the point where the BGP routers establish a peer relationship and exchange their BGP routes. The decision made at the NAP is that any router that is connected to the NAP receives the information needed in order to get to the destinations. In order to bloat a path, the value of the shortest path can be the same as the longer path, and shortest tree load balancing commences. To add to a domain, simply repeating the domain number arbitrarily depending on the objective. An example of this from the BGP lecture is given in the following figure.

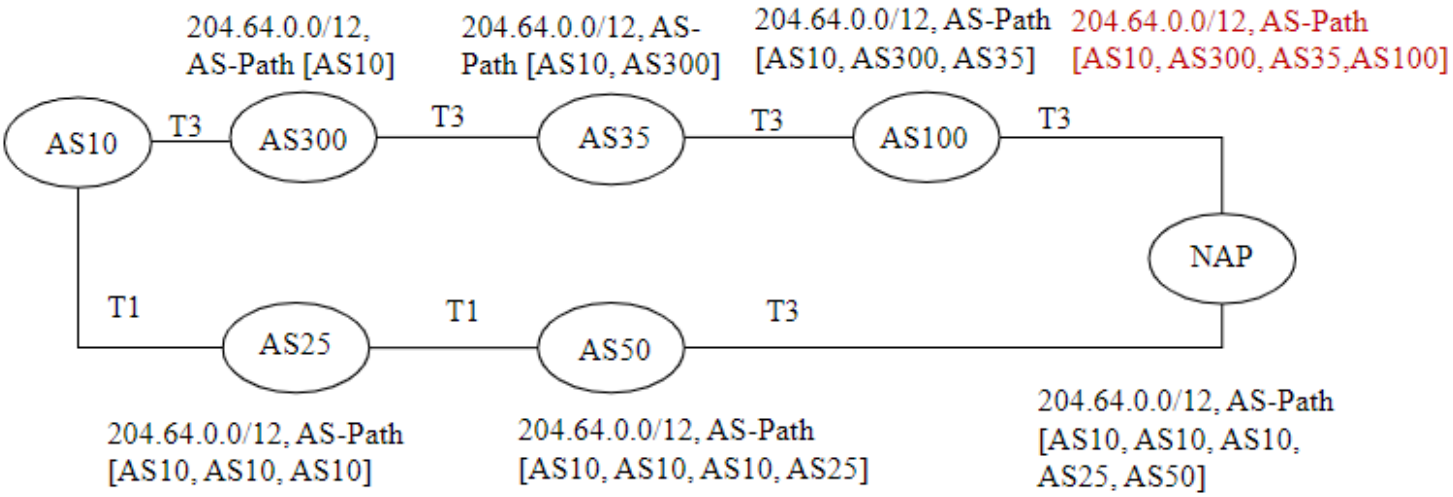


**Figure 3.5.1**

In the above figure, AS10 advertises itself three times, when advertised via AS300. Other path tries to bloat by appending several times the AS#. What the NAP sees is AS10x3 and AS25x1 and AS50x1. Therefore it seems five domains as opposed to four for the other path. Therefore it means that the AS100 path is more preferable for traversal rather than AS50 path. Therefore AS10 receives traffic upon the AS100 path. This is done to manipulate the cost and traffic.

Manipulation is a good reason to serve commercial objectives; other attributes in BGP give preferences. The receiving router assigns priority of the prefix or will send router assigned highest priority on that specific prefix which the receiving BGP router obtains from a neighbor.

The finally attribute to be covered in this report is the MEDI attribute. This is specifically for the ‘sender’ BGP router in order to set the priority of the prefix through he MEDI attribute. In case of MEDI attribute, the lower teh value the ‘better’. Therefore BGP uses the lower value of MEDI, uses that link to send traffic. For example both Rx and Ry receive the same prefix, but with a different MEDI attribute. Through an IGBP session, between Rx and Ry the MEDI value will resolve any issues in the network.

Finally a quick recap of IBGP r EBGP - 2 routers exchange routing tables and handshake but will be using different AS #s which come from two different AS #s. Whereas IBGP sessions exchange routing tables but come from the same BGP AS domain numbers.

# 4. Implementation

## 4.1 Programming Language

For this project, the Python programming language was used. Python is one of the more mature programming languages around and is extremely versatile which is why it was chosen for this project. The other programming language that was considered was Java.

## 4.2 Algorithm Used

The algorithm used for this project was Dijkstra's algorithm. The original version of Dijkstra's algorithm was used to find the shortest path between two different nodes. However, the version used for this project works by specifying the source node and then generating a tree that connects all the other nodes to the source node via shortest path. The result is a shortest-path tree or SPT.

Dijkstra's algorithm is based on the idea that there is a cost associated to go from one node to another. For this project the cost is the dollar value cost of the connection divided by the bandwidth of the connection. So for example if you have a connection that costs $10 and it's bandwidth is 20 Mbps, then the cost that the algorithm will use is 0.5.

## 4.3 Program Structure / Explanation

As stated in section 4.1, the programming language used is python. There are three main chunks of the program, the configuration files processing, implementation of Dijkstra's algorithm, connecting the processed config files with Dijkstra's algorithm and the user input.

The configuration files are each read from the modules/import_config.py file and then put into a json object so that each ASN will have a list of its adjacent ASN's along with the cost and Mbps associated with said connection. The code below is the main function used to read in the configuration files.

```
def read_configs(folder_path):
    result = {}
    config_files = [f for f in listdir(folder_path) if isfile(join(folder_path, f))]
    for config_file in config_files:
        config_path = join(folder_path, config_file)
        config = read_config(config_path)

        result = {
            **result,
            **generate_rc_to_rc_and_host_asn_connections(config),
            **generate_host_to_rc_asn_connections(config),
        }

    return result
```

**Figure 4.3.1 Code snippet representing the function used to read in the config files**

The implementation of Dijkstra's algorithm is done in the modules/dijkstra.py file where a class is created that takes in the processed nodes from the modules/import_config.py file along with a source node. Then the SPT is created with the source node as the root of the tree. The code snippet below is the initialization function for the Dijkstra class. As you can see it iterates through each of the nodes and processes it so as to construct the SPT based on Dijkstra's algorithm.

```
def __init__(self, source, nodes):
    self.source = source
    self.nodes = nodes

    self.setup_default_tree()

    node_to_process = self.source

    while node_to_process:
        self._process_node(node_to_process)
        self.processed.append(node_to_process)
        node_to_process = self._choose_next_node()
```

**Figure 4.3.2 Code snippet representing the function used to setup Dijkstra's algorithm**

And finally, in the app.py file, everything is put together. The config files are read in by calling the code from the modules/import_config.py file, then the user is prompted to input a source and receiver nodes after which the SPT is constructed. The final thing to happen is the defined route is shown in a minimally animated way and then the whole path is printed with the total dollar value cost taking into account as well. The snippet below is the code from the app.py file. As can be seen, it also exports certain objects to be viewed in the output/ directory.

```
from modules import (Dijkstra, choose_node, export_to_output_folder,
                     read_configs)

processed_configs = read_configs("configs")
export_to_output_folder(
    processed_configs,"processed_configs.json"
)

nodes = processed_configs.keys()

print("Choose a source from the list below:")
source = choose_node(nodes)

print("Choose a receiver from the list below:")
receiver = choose_node([node for node in nodes if node != source])

dijkstra = Dijkstra(source, processed_configs)
export_to_output_folder(dijkstra.tree, 'tree.json')

route = dijkstra.find_path_to_node(receiver)
export_to_output_folder(
    route,"route_path.json"
)

dijkstra.animate_route(route)

Dijkstra.print_route(route)
```

***Figure 4.3.3 Code snippet representing the main app code***

Once all of the pieces mentioned above were put together, the topology that was defined by the config files is shown in the figure below.

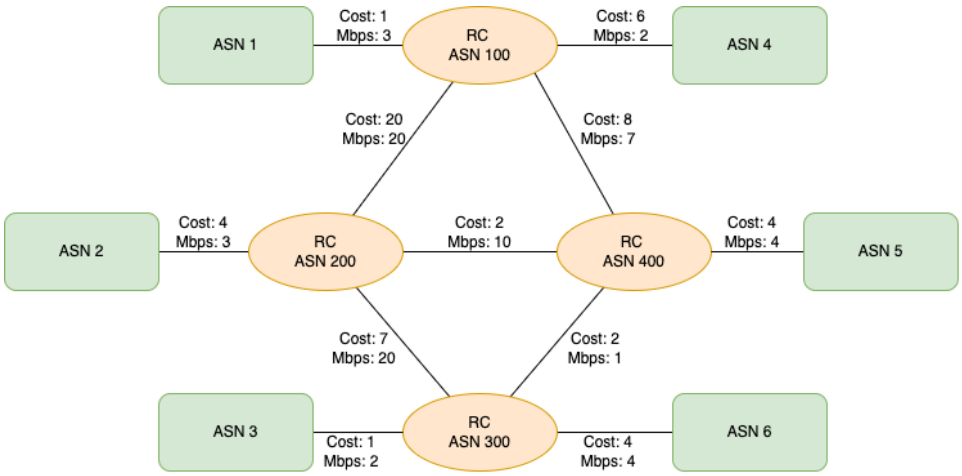


**Figure 4.3.4 Topology Represented in the Configuration Files**

## 4.4 Test case

One of the test cases was to have ASN 1 as the source and ASN 6 as the receiver. When these values were chosen, the result is as shown below.

```
$ python app.py
Choose a source from the list below:
ASN400, ASN5, ASN300, ASN3, ASN6, ASN200, ASN2, ASN100, ASN1, ASN4
> ASN1
Choose a receiver from the list below:
ASN400, ASN5, ASN300, ASN3, ASN6, ASN200, ASN2, ASN100, ASN4
> ASN6
ASN1
.
ASN100
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
ASN200
.
.
.
.
.
.
.
ASN300
.
.
.
.
ASN6
The route cost was 32 and the route taken was: ASN1 => ASN100 => ASN200 => ASN300 => ASN6
```

**Figure 4.4.1 Test case of where source = ASN1 & receiver = ASN6**

This resulted in the following path through the topology:

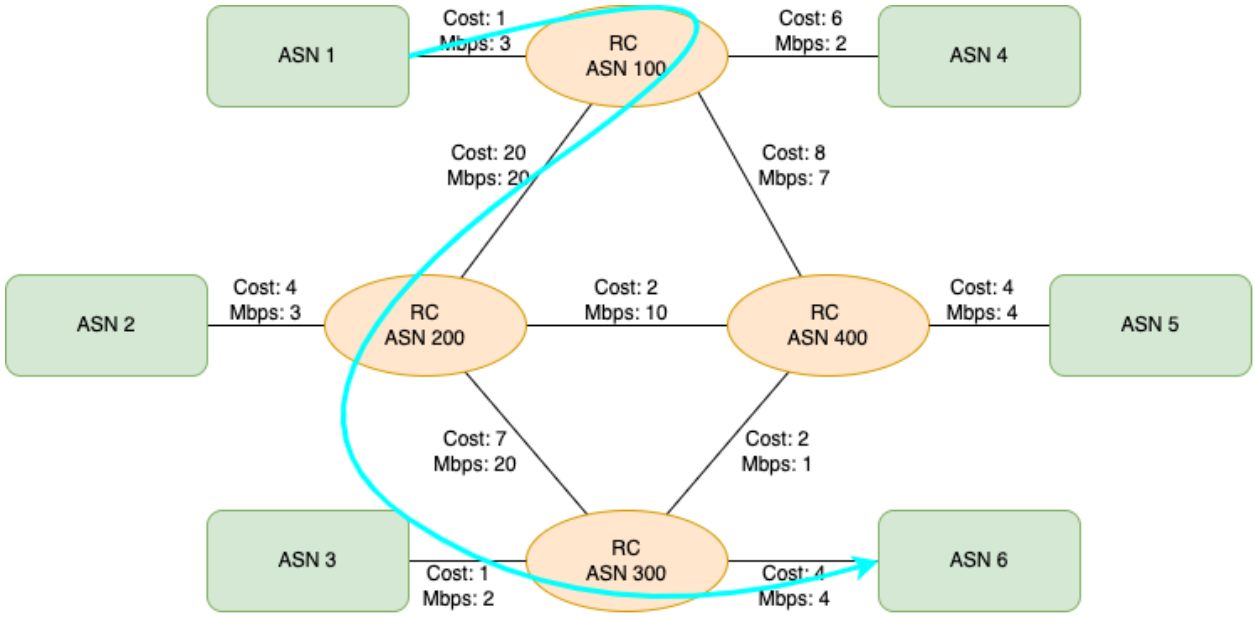


**Figure 4.4.2 Path the test cases took through the topology**

As you can see, it took the connection from ASN 100 => ASN 200 which costs a lot more than ASN 100 => ASN 400. However, ASN 100 => ASN 200 has a much larger bandwidth which was enough to cause that pathway to be chosen.

# 5. Conclusion

The purpose of this project was to explore BGP routing to better understand inter-domain routing. This report demonstrates the design and implementation of a routing control system which evaluates two specific parameters, bandwidth and cost. Subsequently an optimized path along the network through the given inter-domain routes will be established for various configurations.

Our group was able to successfully meet our goal of designing and implementing a routing control system for inter-domain routing. We designed and implemented a path selection algorithm that used a composite cost function derived from the link cost and capacity to determine the optimal path that should be taken from one node to another. For this project we chose to write our program using python because it allowed for easy sharing of code between members and quick development. The path selection algorithm used was Dijkstra's algorithm because it has a low complexity while still being effective for this use case. The program our group designed and implemented was able to collect relevant information from configuration files located in the directory and use Dijkstra's algorithm to determine the optimal path based on our chosen metric.

Altogether, every member participated and played a role in making this project a success.