Andy Michel
04/19/2016
CNT6707

Anycast Protocol Implementation

This project is an implementation of the Anycast Networking Protocol. It is meant for the purpose of research so therefore there are a lot of things that are not taken into considerations such as: latency, security and performance. Below is an overview of the implementation that describes the core components:

- **Server**: is a base class that provides an API that implements the common functionalities that the different server types will inherit. The primary methods are: **bind**, **poll**, **connect**, **accept** and **forward**.
- **Target**: inherits from the base class **Server** an act as the endpoint of the network in other when a client submits a request, the client is the source and **Target** will serve as the destination.(i.e Walmart)
- **JoinProxy**: Acts as the router in network through which all requests must go through in order to reach the **Target**.
- **RendezvouxProxy**: This proxy type has knowledge of all the proxies in the network and its role is to find the closest join proxy to forward the requests.
- **IngressProxy**: Acts as an entry point to the network through which all client requests must go through initially. It finds the closest **RendezvouxProxy** which will it turn perform the necessary actions to make sure the request gets to its final destination.
- **Client**: Is a class that is intended to mimic a *user* (i.e John Doe) who would like to access the network and send a request to a **Target**.

For the sake of brevity, there are a few other components in the project that I will mention but will not explain in details because they play a secondary role. Below are the additional classes:

- **AnycastClient**: is the driver of the program
- **Packet**: a class that specifies a format by which the different servers in the network communicates
- **TCPSocket**: medium by which the servers are able to communicate
- **Utils**: that load the configuration of the network (id, ip, port)

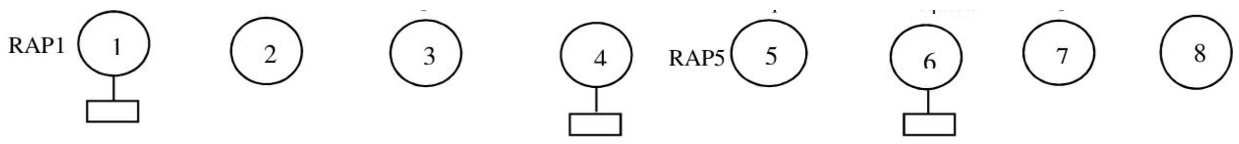
For the simulation, I modeled 8 different clients and logged the number of hops of each packet starting from the client to the: ingress, rendezvoux, join , target and finally back to the client. At the end of the simulation a log file is generated with hops per client and average cost per minute.

To configure a network, there are two files that must be edited although they have been pre-configured for the purposes of a demo.

- **anycast_demo.sh**: This file contains the instructions to build the projects '*make all*' and also creates the spins up the servers on their respective Ips and Ports.
- ***.conf**: There are 4 configurations files: ingress.conf, join.conf, rap.conf and target.conf. They contain all the IP addresses and the ports of the different types of servers. In order to add a new node, modify the files that correspond to the type of node you wish to add and also add the node in anycast_demo.sh.

Simulation & Metrics

Each client fires 1 request that gets routed through the server to the endpoint then back to the client. There is a cost associated with each hop. Given that routing is based on proximity, each server attempts to find the next closest server to forward the packet. In the configuration files the first column is used as the `id` of the server as well as its location therefore when a given server lookups the nearest peer it subtracts its identification number from that of all the peers specifically the *type* of node that it is looking for. Case in point, when an ingress proxy is trying to forward a packet, it calculates the distance of all the rendezvoux proxies nearby and pick then choose the closest one. A distance of 0 emulates a case where the server is simultaneously a Join and a RendezVoux proxy.



The above figure is the initial configuration of the network. My observation through 4 simulations were as follow:
1) 8 ingress, 3 target, 3 join and 2 rendezvoux proxies: 4, 6, 6, 5, 5, 5, 6, 6 hops per client or 43 hops per minute for 8 clients.

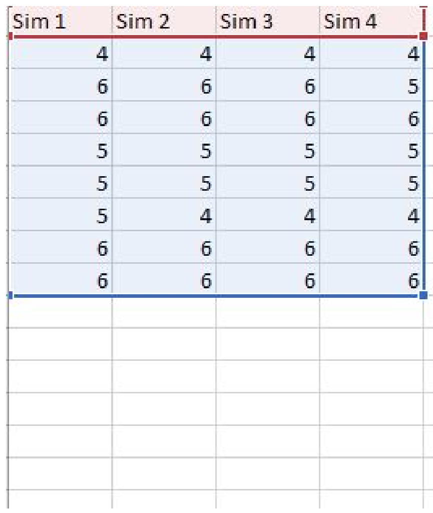
2) 8 ingress, 3 target, 3 join and 3 (third rap added at AC6) rendezvoux proxies: 4, 6, 6, 5, 5, 4, 6, 6 hops per client or 42 hops per minute for 8 clients.

3) 8 ingress, 4 target (fourth target added at AC8), 3 join and 3 rendezvoux proxies: 4, 6, 6, 5, 5, 4, 6, 6 hops per client or 42 hops per minute for 8 clients.

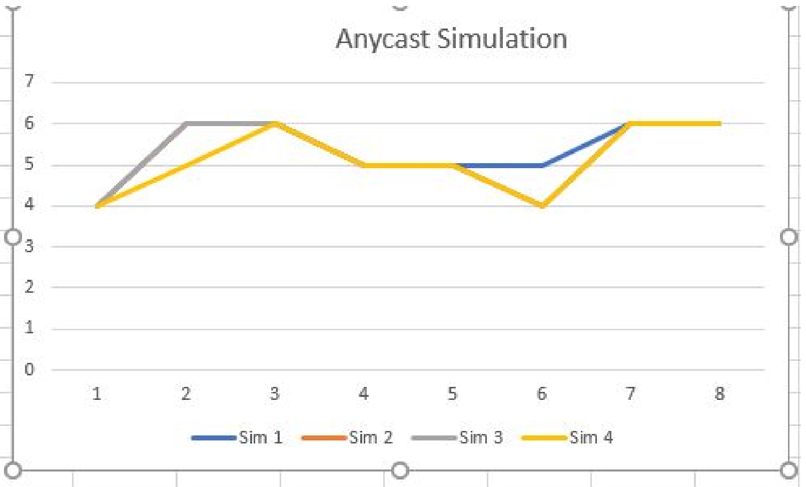
Andy Michel
04/19/2016
CNT6707

4) 8 ingress, 3 target, 3 join and 4 (fourth rap added at AC2) rendezvoux proxies: 4, 5, 6, 5, 5, 4, 6, 6 hops per client or 41 hops per minute for 8 clients.

Below is a graph showing the minimum and maximum cost on average according to the results of the simulation.



Sim 1	Sim 2	Sim 3	Sim 4
4	4	4	4
6	6	6	5
6	6	6	6
5	5	5	5
5	5	5	5
5	4	4	4
6	6	6	6
6	6	6	6



As expected, the line coincide for simulation 2 and 3 in the graph. Simulation 4 showed the most improvement in average cost per minute and simulation 1 was the least performant.