# What does the code do?

My implementation of this assignment makes use of UDP sockets to send data between client and server. I create the airport nodes by using different instances of the command line for each airport, as well as one extra instance used to initialize and route the passengers (instructions below).

I have an airport.py file, where the Airport class lives. For every instance of the Airport class, a UDP server socket is created for the specified airport and bound to the correct address and port. An improvement to this would be to implement the simulation using multithreading instead of using multiple command lines. In my implementation, I’m using localhost as the IP address. The method run() of the Airport class processes the passengers based on whether the current airport is the origin, layover, or destination airport. If a passenger needs to be forwarded further, a UDP client socket is created in the forward\_passenger() method and the payload (passenger info) is sent to the next airport server. If you wanted to scale up the simulation, you could extend the payload by adding birthdate, passport number,...

I also a have a route\_passenger.py file, which is run to initialize passengers using user input in the console and route them if they are correctly initialized. The routing is accomplished by creating a UDP client socket that sends the passenger information to the origin airport. Currently you can initialize and route one passenger per command line input, this could be improved by randomly generating a bunch of passengers and sending them all on the same route, like a real plane.

I used the argparse library to make creating airports and passengers through the command line faster and more customizable.

Each airport node prints an informative statement to the command line and the log when the server is started up, whenever a passenger leaves from or arrives there or passes through, and when the server is shut down. In addition, on faulty input, warnings are printed to the log file.

If you wanted to add more airports to the simulation, you would add them in the mapping either in the hub or the spoke dictionary and assign them an unused port number. An improvement to this program could be made by assigning the port dynamically on server creation.

# How to run the code

Requirements

Python 3.10

pip 23.2.1

standard (built in) libraries: socket, json, logging, signal

external libraries: argparse==1.4.0 (you can run ‘pip install -r requirements.txt’ to install the argparse library)

Running the code:

1. Open at least 3 different terminals, one for each airport that you want to use (minimum 2) and one to initialize the passengers that need to be routed. Navigate to the directory with the files in each terminal.

2. In each terminal that is used to initialize the airports, type: 'python airport.py {code of airport}'

The different airport codes are: ANC, FAI, SEA, BRW, OTZ, SCC, BET, JNU

ANC, FAI and SEA are hubs. BRW, OTZ, SCC, BET, and JNU are spokes.

You must have at least one hub airport.

3. Use the last terminal to route passengers. To do that type: 'python route\_passenger.py {first name of passenger} {last name of passenger} {origin airport code} {destination airport code}’.

As an optional argument you can append --layover {layover airport code}

You can initialize and route one passenger per command.

4. If you would like to gracefully close the airport server, you can type CTRL+C in each airport.py console.

# Discussion

*In your report “README,” compare and contrast how such a simulation compares with the Internet traffic. Do you see parallels with systems such as Internet eXchange Points and Domain Name Servers and how we simulated the traffic with a hub-and-spoke topology here? Please state your assumptions and your thoughts inspired by this exercise in the context of Computer Networks.*

This airport traffic simulation was modeled after a hub-and-spoke topology, which is what the Internet also relies on. There are more small airports (spokes) than big airports (hubs), and the big airports are the only central points for layovers and connections between the small airports. This more or less matches up with how the network of the Internet is structured.

Small access ISPs (spokes) aren’t typically connected directly to each other because this doesn’t scale well when more are added. Instead, they are connected to a larger regional ISP, a tier 1 global ISP, or an IXP, which all act as the large hubs in the center and are interconnected between each other.

These larger ISPs or IXPs act as the hubs that make routing of Internet traffic faster and more efficient. The hubs reduce the distance and the number of hops that packets need to travel to reach their destination. How? At a larger ISP and especially IXP, many networks are connected directly, which means that it’s faster for packets on the originating network to get routed to an IXP - where they are then routed directly to the destination network, instead of traversing multiple intermediate smaller networks (like access ISPs), which would involve additional hops and added latency at each hop.

A similar process can be seen with our airport hub-and-spoke model. It might seem faster to have direct flights going from one small airport to another small airport, to another small airport, ... etc. on a direct route until the destination airport is reached. Instead, it’s more efficient to fly from a small airport to a big hub airport, then from there to the destination small airport, even if it looks like the big airport is far out of the way and the route would be longer. Why? Because the number of layovers/airplane transfers is less, which ultimately reduces the time of the trip, and because big hubs may have the capacity and infrastructure to transport/handle more people at once.

The hub-and-spoke topology also reduces the number of routes that exist for data packets/airplanes to travel on, which reduces routing complexity.

A difference between the air traffic simulation and the Internet is that in this simulation, small airports never communicate with each other directly. However, the Internet isn’t really a strict hub-and-spoke model because there exists a lot of redundancy and messiness in how networks are and aren’t connected to each other. For example, not every connection goes hierarchically up from access ISP -> Regional ISP -> Global ISP -> IXP. Sometimes an access ISP is directly connected to an IXP and so forth. Sometimes spoke nodes are directly connected each other on the Internet whereas with our simulation all layovers need to be at hubs.