Hello, my name is \*\*\* and I will present my team’s experiment.

The topic of my team is ‘optimal condition for caching server’.

In this presentation, we’ll talk about four part, Experiment, Logic, Source code and Result analyzing.

Then let’s start.

Our experiment’s purpose is to find the optimal location policy for arbitrary network topology.

To know optimal location policy, we have to decide the network performance evaluation method to compare what network is better.

Since we can evaluate the network by delay, we will evaluate network performance by the average RTT of each node.

But we want to know how caching server’s location affects to time to send a request and get the desired data.

So we will use the time it takes to get the desired data as a basis for evaluating the network.

From now on, I’ll use RTT to denote time to get desired data and say Round Trip Time to denote original RTT meaning.

Anyway, ‘The average RTT of each node’ will be the scale for network performance.

Then how should we find the optimal location?

Since caching is useful for node requests the same data often, the optimal caching server should be near to this node.

So, our anticipated result is ‘optimal location will be a position which satisfies ‘minimum of sigma 1-variety multiplies square of the distance from a node to cache’.

For our experiment, we have three premises.

First, a delay is related to the distance between nodes.

Next, exceptions in the network like packet loss or congestion control are not considered.

The last one is that there are all data in the origin server.

We have some logic to simplify our experiment.

In our experimental environment, the network is abstracted and simplified.

So, client, server, cache, and link are denoted by node and edge in the graph.

And every packet sending is pseudo-packet sending.

Then what is pseudo-packet sending?

Pseudo-packet sending is just pretending to send the packet and wait until the predicted receive time.

After the predicted time, we do some behavior recorded in the packet.

When there are no desired data in the cache, the cache and client should receive data from the origin server.

But these cache updates are done through pseudo-packet sending.

So, cache update is pseudo-update, too.

Our source code is composed of four big parts.

Let’s start with csv import!

In the left picture, we extract the CSV file path from the command line.

And in the right picture, we read CSV files and extract useful information to make list.

Next step is topology making.

In the first picture, we set some hyperparameters to make experiment simulation more smooth.

The next picture and last picture are making network topology from CSV-extracted information.

Third step is the most important one, packet sending.

In this part, I have to introduce three functions.

The first function is the packet function.

This function controls all time delay and behavior about packet sending.

First of all, we will record how many times the client requested data.

And set sleep time to make propagation delay.

After one sleep time which means the packet arrived at its destination, we check is there desired data in the cache.

And we wait for one more sleep time which means propagation delay from the cache to the client.

If there was desired data, we set sleep time as 0 to ignore sleep function below.

But if there is no desired data, we set sleep time and call the update cache function to update cache data.

After all sleep time over, the packet function records rtt in this data request sequence.

updateCache function is a function for updating the cache.

But since asyncio can run only one asynchronous object, this function is not an async function.

So we considered updateCache runtime not to affect in rtt of the client.

Main function is a function to run asyncio List object.

This function is repeated until the number of sent packets is bigger than the packet limit.

After simulation, lastly, we organize data and export it with a CSV file.

The last step is analyzing the result.

First, we use wolfram alpha to find the optimal location.

Lastly, repeat simulation to gather data. This is one of the five cases of out experiment. For the rest of the data, we attached it to the folder along with the source code.

As you see in the right picture, the closer point to optimal location shows better network performance.

According to the 3D bar graph, there are only a few locations that have better performance than our expected optimal position.

So we concluded our experiment with ‘Optimal location policy will be a condition which satisfies ‘minimum of sigma 1-variety multiplies square of the distance from a node to cache’’.

Thank you for listening.