Introduction:

This is a platform to quickly help you implement satellite network experiments, you can give this PDF to the large language model. Say your needs, and let it automatically generate experimental code for you. If you encounter an error where your code doesn't run, give more information to the large language model. Once you understand the satellite network, you can further modify the code in this tool to better suit your actual needs.

The satellite network consists of multiple layers of satellites, each layer of satellites is called a shell. Each shell contains a number of orbits, and each orbit contains a specific number of satellites.Because of the dynamic nature of satellite networks, you need to specify how often to perform a calculation and how many times to repeat this behavior.

You should only modify the inputs and try not to modify the APIs code.

Inputs Requirement:

1.con_name: string

Name your constellation for test

2.number_of_shells: int

Indicate how many shells of test constellation

3. shell_data:matrix(list of list)

Each row means one shell's parameters:altitude,period,orbit angle,phase(default is 0),orbit number,satellite number per orbit

4. time_interval:int

Time interval between two time slice.

5. num_intervals:int

Number of time slices in this experiment

6. connectivity:list of int(the length must be 4)

Considering that the current satellite has four inter-satellite links, designate four neighbors ' IDs of satellite 1

7. user_path,cell_path,output_path: string

The input and output file paths.No change is recommended without special requirements

APIs:

1.constellation APIs:

Constellations based on the satellite information you provided.Calculate and save satellite parameters in each time slice

2. construct network and measure hops and RTTs APIs:

Communication using randomly generated user coordinates. Calculate their delay performance in different time slices.

If connectivity is not defined or defined wrongly, we will use +grid for default.

3. coverage statistics APIs:

The coverage rate of constellation satellites to each block of the ground in different time segments was counted

4. Output and analyze APIs:

Output the results obtained in this experiment are and analyze further.

Demo:

```
from utils import constellation_generator
from utils import printer
from utils import reader
from utils import connection
from utils import coverage

# input should be identified
con_name = "test"
# Kuiper
number_of_shells = 3
shell_data = [
    [630, 5830, 51.9, 0, 34, 34],
    [610, 5805, 432, 0, 36, 36],
    [590, 5780, 33, 0, 28, 28]
]
time_interval = 500
num_intervals = 10
connectivity = []
# connectivity = [35, 67, 1123, 1125]    # Satellite_1's neighbour
user_path = "data/input/random_20_user.csv"
cell_path = "data/input/earth_coordinates.csv"
output_path = "data/output/result/"

# 2.constellation
ret_path = reader.create_constellation_xml(con_name, number_of_shells, shell_data)
constellation = constellation_generator.constellation_generator(con_name, ret_path, time_interval, num_intervals)
printer.constellation_printer(constellation)

# 3.construct network and measure hops and RTTs
users = reader.read_users(user_path)
hop, rtt = connection.delay_test(constellation, users, num_intervals,connectivity)

# 4.coverage statistics
ret = coverage.coverage(constellation, num_intervals)

# 5.output and analyze
hop_output_path = output_path + con_name + "_motif_hop.csv"
rtt_output_path = output_path + con_name + "_motif_rtt.csv"
printer.write_2d_array_to_csv(hop, hop_output_path)
printer.write_2d_array_to_csv(rtt, rtt_output_path)
```