111550022 施采緹

Lab2 – report

<commit history>

support run example code for calculate_delay
HelloHe110 committed 3 weeks ago

The first commit is in the first week of the lab, and the modified parts are all about the *calculate_delay* method, and as we know right now, the method is actually wrong. Here is what I do: (1) for every 'Tx', I record the current time, and for every 'Rx', I update the overall delay time of packet with this sequence number. Finally, with those stored delay time, I can calculate the average packet end to end delay time! Sadly, if we take a deeper look to the simulation process, we will figure out that there are actually many packet with same sequence number, which shows that I mis-update the recorded send-time of the packet! This is the main issue of this commit, and will later be corrected!

Task1: Topology Configuration
HelloHe110 committed 2 weeks ago
adf3523

for the second commit, I mainly focus on task 1 – setting up the topology for the ground-station(GS) as both source and destination to *(6.06692, 73.0213)* according to the spec. And thanks to the hint from TA, we know that the x-y-z coordinates can be called by `pos_tx = users.Get(0)->GetObject<MobilityModel>()->GetPosition();`, and this is the way to get both the coordinates of GS and SAT! For the convenience of further processing in MATLAB bf / no-bf pathloss calculation. And the methodology of writing the file is through ofstream, which is a normal way to write out file in C++. In order to better the readability of the code, I grasp the write file process in to a lambda function `auto write_txt = [](Vector pos_tx, Vector pos_rx, Vector pos_sat1) {`. The last thing is the new xyz_positions.txt which store the information of the x-y-z coordinates of GS and SAT!

support task1~5
HelloHe110 committed last week
08801a9

In this commit, the basic structure of this lab was set up! The $txPowerDbm$, $noisePowerDbm$ and $bandwidthHz$ are all defined as what being mentioned in the spec. And, I got the $rxPowerDbm$ by $CreateObject < LeoPropagationLossModel >$, which is further considered no suitable...

```
Ptr<MobilityModel> txMobility = users.Get(0)->GetObject<MobilityModel>();  // U
Ptr<MobilityModel> rxMobility = satellites.Get(1)->GetObject<MobilityModel>();
double rxPowerDbm = propagationLossModel->DoCalcRxPower(txPowerDbm, txMobility, rxMobility);
```

After $rxPowerDbm$, we could head in to the data rate calculation by Shannon Capacity.

```
double snrDb = rxPowerDbm - noisePowerDbm;
double snrRatio = std::pow(10.0, snrDb / 10.0);
```

```
double shannonCapacity = bandwidthHz * std::log2(1 + snrRatio);
```

After we got the value of $shannonCapacity$, we can therefore udpate the new $DataRate$ of GS to $shannonCapacity$; furthermore, I set the $DataRate$ of SAT to 1Gbps as required. Let's move on other files, the $examples/pathloss.txt$ is there for storing the calculated result of $pathloss$ from MATLAB code, and $model/leo-propagation-loss-model.cc$ and $.h$ are also changed as the requirement in spec.

```
double rxc = txPowerDbm - m_atmosphericLoss - pathLoss - m_linkMargin;
```

update support for report & error fixed

HelloHe110 committed now 0071525

Due to the fact that the methodology of calculate-delay being modified, the first change in this commitment is to update the $func\ EchoTxRx$ by setting up the constraint for not overwrite the send-time if there exist an earlier time while the Rx part remain unchanged.

```
if (context.find("/Tx") != std::string::npos && sendTime[seq] == (Time)0) {
    sendTime[seq] = now;  // 記錄發送時間
}
```

The second part is to remove the rounding bias in MATLAB by sending the x y z in numerical value instead of scientific notation.

```
outfile << "" << std::to_string(pos_tx.x) << " " << std::to_string(pos_tx.y) << " " << std::to_string(pos_tx.z) << "\n";
```

Also, I noticed that we can't directly create a object for calculating the $rxPower$ since there are some extra variables like $m\_atmosphericLoss$ set during the simulation?! If we create a new object directly, those variables would be the default value, which is honestly wrong!

```
-    Ptr<LeoPropagationLossModel> propagationLossModel = CreateObject<LeoPropagationLossModel>();
+    // Ptr<LeoPropagationLossModel> propagationLossModel = CreateObject<LeoPropagationLossModel>();
+    Ptr<NetDevice> dev = utNet.Get(SAT_Idx);
+    Ptr<LeoMockNetDevice> mockDev = DynamicCast<LeoMockNetDevice>(dev);
+    Ptr<LeoMockChannel> channel = DynamicCast<LeoMockChannel>(mockDev->GetChannel());
+    Ptr<LeoPropagationLossModel> propagationLossModel = DynamicCast<LeoPropagationLossModel>(channel->GetPropagationLoss());
```

And there are also some modifications in order to fit the need of generating the output of result. Overall they are based on the formula!

```
double euclidean_dis = sqrt((pos_tx.x - pos_sat.x)*(pos_tx.x - pos_sat.x) + (pos_tx.y - pos_sat.y)*(pos_tx.y - pos_sat.y) + (pos_tx.z - pos_sat.z)*(pos_tx.z - pos_s
```

```
double snrDb = rxPowerDbm - noisePowerDbm;
double snrRatio = std::pow(10.0, snrDb / 10.0);
double shannonCapacity = bandwidthHz * std::log2(1 + snrRatio);
```

```
// HERE
// printf("\n====== w/o bf - Results ======\n");
printf("\n====== w/  bf - Results ======\n");
printf("Q1: Euclidean Dis: %.6f m\n", euclidean_dis);
printf("Q2: PathLoss: %.6f dBm\n", readPathloss());
printf("Q3: RxPower: %.6f dBm\n", rxPowerDbm);
printf("Q4: SNR: %.6f dB\n", snrDb);
printf("Q5: DataRate: %.6f Mbps\n", shannonCapacity / 1e6);
```

<further commit>

If there exist further commit, it'll be only about the basic change like uploading the report/result file, neither core idea of the code nor the concept will be modify!