

Commit 1 : task 1 finish

Commit2 : task 2 有先寫出一個完整的版本但不確定，所以只有 commit 到 task2 寫完，但不小心也把 lab4.ortools.out 交出去了

Commit3 : all finish and report

Global index :

```
20  std::map<int, int> gsToSat; // ground station ID → satellite ID
21  std::map<std::pair<int, int>, double> linkRate; // (GS, Sat) → Mbps
22
23  std::map<int, bool> gsStarted; // ground station 是否開始傳送
24  std::map<int, bool> gsFinished; // ground station 是否完成傳送
25  std::map<int, bool> satBusy; // satellite 是否正在接收資料
26
27  std::map<int, double> startTime; // 每個 ground station 的開始時間
28  std::map<int, double> endTime; // 每個 ground station 的結束時間
29  std::map<int, double> collectionTime; // 每顆 satellite 的收集時間
30
31  std::map<int, uint64_t> currentRxBytes; // gsId → 已接收的位元組數
32
33  uint64_t maxbytes = 125000;
```

Task 1: Input File

Get the data rate

讀取 network.graph，先讀
取地面站跟衛星數量，再讀
每個 link 的 datarate，

把 datarate 存在
linkRate[{gsId, satId}]

Get the association results

```
208 // Task 1: Input File
209 // Step 1: Read network.graph for link data rates
210 std::ifstream graphFile("contrib/leo/examples/network.graph");
211 if (!graphFile) {
212     std::cerr << "Cannot open network.graph file" << std::endl;
213     return 1;
214 }
215 int numGs, numSat, numLinks;
216 graphFile >> numGs >> numSat >> numLinks;
217 std::string dummy;
218 std::getline(graphFile, dummy);
219 for (int i = 0; i < numLinks; ++i) {
220     int gsId, satId;
221     double rate;
222     graphFile >> gsId >> satId >> rate;
223     linkRate[{gsId, satId}] = rate;
224 }
225 graphFile.close();
```

先把這次的 network.graph
丟到 lab3 的 lab3_ortools.cc
跟 lab3_greedy.cc 得到

network_ortools.out /
network.greedy.out(放在跟
leo-lab4.cc 同一個目錄)。

讀取檔案時跳過第一行(原本
預估的傳輸需要時間)，讀取
對應的 association。

```
227 // Step 2: Read inputFile (e.g., network.ortools.out) for GS-Sat mapping
228 std::string fullPath = "contrib/leo/examples/" + inputFile;
229 std::ifstream inFile(fullPath);
230 if (!inFile) {
231     std::cerr << "Cannot open input file: " << inputFile << std::endl;
232     return 1;
233 }
234 // Skip the first line (header)
235 std::getline(inFile, dummy);
236
237 int gsId, satId;
238 int i=0;
239 while (inFile >> gsId >> satId) {
240     gsToSat[gsId] = satId;
241     i++;
242     if (i==20) break;
243 }
244 inFile.close();
```

Task 2: Send Packet

Task 2.1: Complete SendPacket(int gsId, int satId)

取得 Ground Station 和 Satellite 節點，初始化地面站的接收數據起始點，並
用地面站當 index，記錄衛星在這次傳輸開始前的數據接收基準點
(GetTotalRx)，以便稍後計算接收的總數據量。(不然會被之前累積接收的別的地
面站傳的 Rx 干擾)

```
85 void SendPacket(int gsId, int satId){
86     // Task 2.1: Complete this function
87
88     Ptr<Node> gs = groundStations.Get(gsId);
89     Ptr<Node> sat = satellites.Get(satId);
90
91     // 初始化該地面站的接收起始點
92     Ptr<PacketSink> sink = DynamicCast<PacketSink>(sat->GetApplication(0));
93     if (sink) {
94         currentRxBytes[gsId] = sink->GetTotalRx(); // 記錄這次傳輸前的總接收量
95     }
```

用 lab3 的方式設定雙向 datarate

```
97 // 設定雙向 DataRate
98 std::ostringstream oss;
99 oss << linkRate[{gsId, satId}] << "Kbps";
100 utNet.Get(gs->GetId())->GetObject<MockNetDevice>()->SetDataRate(DataRate(oss.str()));
101 utNet.Get(sat->GetId())->GetObject<MockNetDevice>()->SetDataRate(DataRate(oss.str()));
```

使用 `GetObject<Ipv4>()` 來存取衛星的 IPv4 屬性，`GetAddress(1, 0).GetLocal()` 獲取衛星的 IP 地址，以便地面站指定目標 IP 來發送數據。

建立 `BulkSend TCP` 應用，指定數據要發送到 `dstAddr` (衛星 IP) 的 `port (9)`，並設置 `MaxBytes` 跟 `SendSize`。

將 `BulkSendHelper` 安裝到地面站 `gs` 上，開始發送數據包。

設定 `Start(Seconds(0))` 讓應用在模擬開始時立即運行，不延遲。

```
103 // 取得 Satellite IP
104 Ptr<Ipv4> ipv4 = sat->GetObject<Ipv4>();
105 Ipv4Address dstAddr = ipv4->GetAddress(1, 0).GetLocal();
106
107 // 建立 BulkSend TCP App
108 BulkSendHelper source("ns3::TcpSocketFactory", InetSocketAddress(dstAddr, port));
109 source.SetAttribute("MaxBytes", UintegerValue(maxbytes));
110 source.SetAttribute("SendSize", UintegerValue(512));
111
112 ApplicationContainer srcApp = source.Install(gs);
113 srcApp.Start(Seconds(0));
```

紀錄傳輸狀態以利後面判斷要不要啟動 `SendPacket`

```
115 // 記錄傳輸狀態
116 gsStarted[gsId] = true;
117 satBusy[satId] = true;
118 startTime[gsId] = Simulator::Now().GetSeconds();
```

Task 2.2: Call `SendPacket()` in `main()`

安排地面站向衛星傳送數據：遍歷所有地面站 (`gsId`) 和其對應的衛星 (`satId`)(index 小到大)，如果該地面站還沒開始傳，並且想要傳的衛星有空，就 `call SendPacket`，讓他開始傳輸資料。

如果衛星在忙就跳過去，等衛星接收完當前地面站的資料時，他會再叫下一個地面站傳 `SendPacket` (in `EchoRx`)

```
246 // Task 2.2: Call SendPacket()
247 for (auto &pair : gsToSat) {
248     int gsId = pair.first;
249     int satId = pair.second;
250     if (!gsStarted[gsId] && !satBusy[satId]) {
251         SendPacket(gsId, satId);
252     }
253 }
254
255 Simulator::Schedule(Seconds(1e-7), &Connect);
256 Simulator::Stop (Seconds (duration));
257 Simulator::Run ();
258 Simulator::Destroy ();
```

Task 3: EchoRx()

確認接收的節點是否為衛星，取得 PacketSink 並獲取 totalRx

```
41 // Task 3: Complete this function
42 int nodeId = std::stoi(GetNodeId(context));
43 if (nodeId >= (int)satellites.GetN()) return; // 非衛星不處理
44
45 int satId = nodeId;
46 Ptr<Node> sat = satellites.Get(satId);
47 Ptr<PacketSink> sink = DynamicCast<PacketSink>(sat->GetApplication(0));
48 if (!sink) return;
49
50 uint64_t totalRx = sink->GetTotalRx();
```

遍歷 gsToSat，尋找與 satId 相關聯的地面站 (gsId)，找到現在正在傳給他並且還沒完成的地面站，計算從他開始傳到現在總共接收了多少 Rx，如果達到 maxbytes 表示目前的地面站傳送完成 → 更新衛星的收集時間。

啟動下一個還沒開始的地面站的傳輸，並更新 currentRxBytes

```
52 for (auto &pair : gsToSat) {
53     int gsId = pair.first;
54     if (pair.second == satId && gsStarted[gsId] && !gsFinished[gsId]) {
55         uint64_t rxSoFar = totalRx - currentRxBytes[gsId];
56
57         if (rxSoFar >= maxbytes) {
58             endTime[gsId] = Simulator::Now().GetSeconds();
59             gsFinished[gsId] = true;
60             satBusy[satId] = false;
61
62             collectionTime[satId] = std::max(collectionTime[satId], endTime[gsId]);
63
64             // 嘗試啟動下一筆任務
65             for (auto &next : gsToSat) {
66                 int nextGs = next.first;
67                 int nextSat = next.second;
68                 if (!gsStarted[nextGs] && !satBusy[nextSat]) {
69                     SendPacket(nextGs, nextSat);
70                     break;
71                 }
72             }
73
74             // 更新 currentRxBytes 為目前新基準
75             currentRxBytes[gsId] = totalRx;
76
77             break;
78         }
79     }
80 }
```

Task 4: Output File

開啟 output file (都在同一個目錄)，計算總接收時間(取最久的衛星)

```
256 // Task 4: Output File
257 fullPath = "contrib/leo/examples/" + outputFile;
258 std::ofstream outFile(fullPath);
259 v if (!outFile) {
260     std::cerr << "Cannot open output file: " << outputFile << std::endl;
261     return 1;
262 }
263
264 // 計算總收集時間
265 double totalTime = 0;
266 v for (auto& pair : endTime) {
267     totalTime = std::max(totalTime, pair.second);
268 }
269 outFile << totalTime << std::endl;
```

輸出每顆衛星的總時間、每個地面站的開始跟結束時間

```
271 // 每顆衛星的 collection time (ID 排序)
272 v for (uint32_t i = 0; i < satellites.GetN(); ++i) {
273     double time = collectionTime.count(i) ? collectionTime[i] : 0.0;
274 v     if (time != 0) {
275         outFile << i << " " << time << std::endl;
276     }
277 }
278
279 // 每個 ground station 的 start / end time (ID 排序)
280 v for (uint32_t i = 0; i < groundStations.GetN(); ++i) {
281     double start = startTime.count(i) ? startTime[i] : 0.0;
282     double end = endTime.count(i) ? endTime[i] : 0.0;
283     outFile << i << " " << start << " " << end << std::endl;
284 }
285 outFile.close();
```

Result :

1. Compare lab4.greedy.out and lab4.ortools.out

```
≡ lab4.greedy.out

1 0.682102
2 1 0.108083
3 2 0.220336
4 3 0.452024
5 4 0.228448
6 5 0.337939
7 6 0.220952
8 8 0.682102
9 0 0 0.108083
10 1 0 0.107541
11 2 0.107541 0.220336
12 3 0 0.11425
13 4 0.11425 0.230778
14 5 0.230778 0.34166
15 6 0.34166 0.452024
16 7 0 0.11592
17 8 0 0.110286
18 9 0 0.112418
19 10 0.11592 0.228448
20 11 0.110286 0.228071
21 12 0.112418 0.220952
22 13 0 0.108485
23 14 0.108485 0.222657
24 15 0.222657 0.332943
25 16 0.332943 0.452296
26 17 0.452296 0.566116
27 18 0.566116 0.682102
28 19 0.228071 0.337939
```

```
≡ lab4.ortools.out

1 0.278296
2 0 0.26545
3 1 0.278296
4 2 0.254399
5 3 0.235131
6 4 0.230025
7 5 0.225653
8 6 0.220952
9 7 0.246872
10 8 0.222106
11 9 0.237353
12 0 0 0.113262
13 1 0.113262 0.26545
14 2 0 0.118675
15 3 0.118675 0.278296
16 4 0 0.128846
17 5 0.128846 0.254399
18 6 0 0.107363
19 7 0 0.11592
20 8 0.11592 0.230025
21 9 0 0.112418
22 10 0.107363 0.235131
23 11 0 0.112785
24 12 0.112418 0.220952
25 13 0 0.122693
26 14 0 0.116814
27 15 0 0.108286
28 16 0.122693 0.246872
29 17 0.108286 0.222106
30 18 0.116814 0.237353
31 19 0.112785 0.225653
```

greedy：會有某些衛星承擔過多負載，導致總收集時間較長。

Ortools：採用數學最佳化方法，通常能達到更短的數據收集時間、每顆衛星更平均。

2. Explain why the collection time will not be the same as the solutions in network.xxx.out

network.xxx.out 沒有任何協議開銷(Protocol Overhead)，只是單純用資料大小跟 datarate 來計算需要的時間

而用 ns3 模擬的：TCP 有額外的數據握手 (3-way handshake)會造成額外延遲。數據封包、應用層設定、網路擁塞控制 也可能影響真實的收集時間。

3. Explain the meaning of MaxBytes and SendSize in BulkSendHelper

MaxBytes (最大傳輸量)：限制該應用最多能傳輸多少位元組(總共要傳多少)

SendSize (傳輸區塊大小)：每次 TCP 傳輸的數據片段大小 (bytes)。

MaxBytes 決定總傳輸時間 → 數據量越大，衛星處理的時間越長。

SendSize 決定傳輸效率 → 較小的值可能增加 TCP 開銷，但較大的值可能導致擁塞。

4. Adjust the value of MaxBytes and observe the changes in transmission time

Maxbytes	Ortools	greedy
500000	1 0.51008	1 1.35204
250000	1 0.355557	1 0.905449
125000	1 0.278296	1 0.682102
62500	1 0.238689	1 0.570428
31250	1 0.209382	1 0.49782
15625	1 0.199704	1 0.469841

MaxBytes 越大，傳輸時間越長

這是因為 地面站需要傳輸更多的數據，因此衛星接收完成的時間會延遲。

MaxBytes 越小，傳輸時間越短

當 **MaxBytes** = 15625 時，**ORTools** 的傳輸時間降至 0.199704 秒，**Greedy** 則降至 0.469841 秒。

但是 **MaxBytes** 越小時，傳輸時間的差距也越小，好像有一個下限值，可能是因為 tcp overhead。

而 **MaxBytes** 的調整對於 **greedy** 的影響較大，因為衛星分配不均，所以延遲造成的代價較大，會一個加一個累積下去，衛星越跑越遠。