Mobile Radio Networks project (OpenRAN)

Milestone 2: Requirements and design notes

Andrea Caravano

Academic Year 2023–24

1 Project requirements

1.1 Specifications

Implement an xApp that collects PHY/MAC metrics:

- Per-UE RSRP
- Per-UE BER (uplink and downlink)
- Per-UE MCS (uplink and downlink)
- Cell load (i.e. allocated PRBs)

All values timestamped and saved in a CSV file
500ms data collection loop

1.2 Meaningful values

- RSRP: -44 dBm (optimal) to -144 dBm (worst)
- BER: 0 to 1 (or 0 to 100, in percentage)
- MCS: inspired by lectures exercises, 3 Mb/s (worst) to 100 Mb/s (optimal)
- Cell load: 24 (worst) to 275 (maximum) PRBs

2 Protobuf specification

2.1 RAN parameters

An overall RAN parameter have been added for the cell load.
Note that, since this is not treated as a per-UE parameter, this is added only once to the response (as expected).

```
enum RAN_parameter{
    GNB_ID = 1;
    UE_LIST = 3;
    CELL_LOAD = 4; // added
}
```

2.2 per-UE parameters

RSRP, BER and MCS measurements have been added. BER and MCS are split in uplink and downlink directions, as per project specifications.

```
message ue_info_m{
    // this is to identify the ue
    required int32 rnti=1;

    // specific ue's measurements (these will come from the gnb)
    optional float rsrp=2;
    optional float berul=3;
    optional float berdl=4;
    optional float mcsul=5;
    optional float mcsdl=6;
}
```

3 Snippets

3.1 gNB message handler

The described Protobuf specification translates into the following code snippets, in which the most important behaviours of the application is shown.

```
void ran_read(RANParameter ran_par_enum, RANParamMapEntry* map_entry){
...
case RAN_PARAMETER__CELL_LOAD:
    cell_load = (rand() % (275 - 24 + 1)) + 24;
    map_entry->value_case = RAN_PARAM_MAP_ENTRY__VALUE_STRING_VALUE;
    map_entry->string_value = int_to_charray(cell_load);
    break;
...
}
```

```
UeListM* build_ue_list_message(){
...
ue_info_list[i]->has_rsrp = 1;
ue_info_list[i]->rsrp = -((rand() % (144 - 44 + 1)) + 44); // dBm
ue_info_list[i]->has_berul = 1;
ue_info_list[i]->berul = (double)rand() / (double)RAND_MAX; // [0, 1]
ue_info_list[i]->has_berdl = 1;
ue_info_list[i]->berdl = (double)rand() / (double)RAND_MAX;
ue_info_list[i]->has_mcsul = 1;
ue_info_list[i]->mcsul = (rand() % (100 - 3 + 1)) + 3; // Mb/s
ue_info_list[i]->has_mcsdl = 1;
ue_info_list[i]->mcsdl = (rand() % (100 - 3 + 1)) + 3;
...
}
```

3.2 xApp

The xApp mantains the general overall behaviour, but report requests are sent every 500 ms, writing received data to a timestamped CSV file, as per specifications.