UNIVERSITÀ DI PISA

Department of Information Engineering

Aeronautical Communication System

Performance Evaluation of Computer Systems and Networks project

Pietro Gronchi
Amedeo Pochiero
Giovan Battista Rolandi

A.Y. 2019-2020

Contents

1 Introduction

Aeronautical Communication System is a cummunication system between aircrafts (AC) and a control tower (CT). Connection between AC and CT is provided by ground base stations (BS), which are deployed over a grid, at a distance M from neighbors.

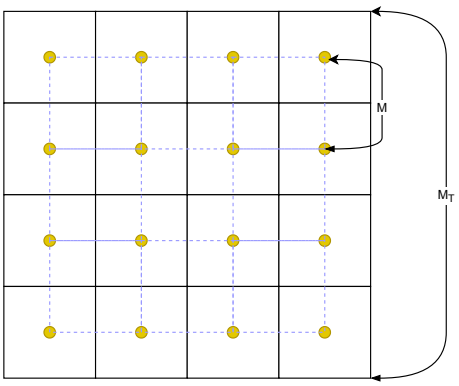

Figure 1: System topology

Deployment grid has a square topology, eg. number of rows is equal to number of columns. In Figure **??**, deployment grid is drawn with dotted blue lines, while solid black lines delimit the area, called *cell*, where an AC is closer to the inner base station. M_T is a shortcut to indicate $M_T \cdot n_l$, where n_l is the number of BSs that lay on a row (or a column).
Each AC selects only one BS at a time as its serving BS, and can transmit only one packet at a time. Possibly, packets are backlogged in a queue on the AC.
ACs execute the handover procedure every t seconds, accordingly with the following algorithm:

```
if (serving BS is the closest one) then
  do nothing
else
  select closest BS as serving BS
  pause transmissions for penalty time (p)
endif
```

1.1 Constant and variables summary

- **k**: it represents the interarrival time of packets;
- **M**: it represents the distance between two consecutive BS on the same row, or column, and it is fixed at 25 km;
- **d**: it represents the distance between AC and BS;
- **s**: it represents the service time of a packet, and is given by $s = T \cdot d^2$;
- **T**: it is a tuning parameter for service time, whose value is costant for every AC in the system;
- **t**: it represents the handover period;
- **p**: it represents the handover penalty;

2 Model

Service time depends on distance between AC and BS. Distance between AC and BS, namely d, belongs to interval $[0, d_{max}[$ where d_{max} can be computed as $d_{max\_in} + d_{max\_out}$, where:

- $d_{max\_in} = \frac{M\sqrt{2}}{2}$
- $d_{max\_out} = vt$

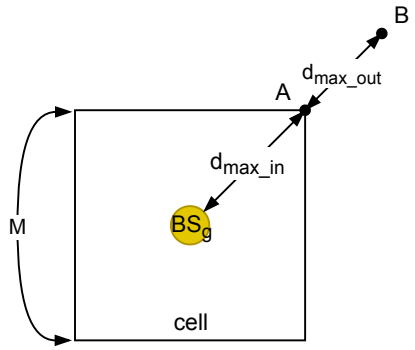

Figure 2: Computation of d_{max}

What is the worst case? Referring to Figure **??**, let's assume that an AC is moving in direction $BS_g \rightarrow B$, and that, at a given time t_g, $\overrightarrow{AC} \in cell$ and $\overrightarrow{AC} - \overrightarrow{A} < \epsilon$ with ϵ small as you wish, eg. AC is at point A but still inside the cell where the nearest BS is BS_g; and let's assume that, at this time, AC initiates the handover procedure and discovers that the nearest BS is BS_g. Immediately after this handover procedure, performed without any penalty time, AC is out of the cell and will not initiate the handover procedure before t time: this means that, when it is in point B, it has traveled for, $d_{max_{out}} = vt$, since the last handover procedure, and that the longest distance d_{max} between AC and BS is given by:

$$d_{max} = \frac{M\sqrt{2}}{2} + vt$$

2.1 Movement of AC

Movement of AC is given by the following algorithm, implemented using the `TurtleMobility` module from INET:

- generate a vector $\vec{s}$ such that $|\vec{s}| \in \mathcal{U}(0, M_T\sqrt{2})$ and $\angle s \in \mathcal{U}(0, 2\pi)$; upper bound of $\vec{s}$ is maximum possibile distance before AC surely wraps around;
- given that AC is in position $\vec{a}$, move to position $\vec{a} + \vec{s}$, at constant v speed, possibly wrapping around at the edges of the simulated topology;
- once reached new position, re-do from the beginning;

3 Application

3.1 Requirements

3.2 Compile

To compile the application, just run:

```
$ cd $REPO
$ make -j$(nproc)
```

Application's two executables, `client` and `server`, will be found in the `bin` directory. For simplicity, it is suggested to run them from the repository's root directory.
To compile this documentation, just run:

```
$ cd $REPO/doc
$ make
```

3.3 Run

To run the application just type the following commands in your shell.

3.4 Repository

Aeronautical Communication System's repository is organized in directories:

- `bin` contains the `client` and `server` executable files;
- `cert` contains all the certificates; if the application's server is run on the same host as the client, then this directory can be shared.
- `conf` contains server configuration, eg. a file with authorized clients list;
- `debug` contains some useful commands and configuration files for some common debugging tools like *valgrind* and *gdb*;
- `doc` contains the LaTeXsource code of this document;
- `downloads` is the default location for files downloaded from the server by the client;
- `obj` contains object files before their linking;
- `root` is the default and only location allowed for files uploaded to the server;
- `src` contains the application's source code.