# Telephone simulation

Group 3 Denver

May 5, 2019

## Contents

# 1 Introduction

## 1.1 Purpose

This project simulates a system responsible for routing telephone calls.

## 1.2 Scope

The telephone simulation will provide a simulation of a small network of phones that has a central handler that routes phone calls between them over a node. The system is not hooked up to real phones, but will instead read input commands from the keyboard and a configuration file.

## 1.3 Glossary

- node - Routing for a call between phones is simulated to go over a node
- handler - The handler is the central point where all phones and nodes are registered and it is responsible for the routing of calls
- JSON - JavaScript Object Notation

## 1.4 Technologies to be used

- Python
- JSON

## 1.5 Overview

The SRS document has two further section namely the overall description of the project and the specific requirements.

# 2 The Overall Description

## 2.1 Product Perspective

The product is independent and totally self-contained.

### 2.1.1 Interfaces

The simulation does not feature any graphical user interface, but takes inputs and commands via the command line from the user.

#### 2.1.2 Operation

The product operates only on provided inputs and does not feature any operation without direct input from the user.

### 2.2 Product Functions

The software provides the user the ability to simulate phone routing using a command line. The user can specify his initial setup via file. The initial setup consist of a non empty set of nodes, an arbitrary number of phones. With this done it is possible to make calls to other phone that get routed over a node, the other phone needs to be connected to the handler and not busy. The call functionality does include ringing and the possibility to refuse a connection. A phone can be connected or disconnected to the handler, this affects reachability of the phone. Since this is a simulation the user also can issues commands to simulate the malfunction of a phone or a node. When a node malfunctions every call that is routed over the specified node gets terminated. When a phone malfunctions, it is no longer reachable for the other phones. To make this a little bit more realistic the handler only notices a malfunction of a phone when it gets contacted the first time after a malfunction happened. The whole state of the system will be outputted to the command line in every time something changes.

### 2.3 Assumptions and Dependencies

The product must run on Windows 10 and Linux machines (Ubuntu 19.04).

## 3 Specific Requirements

### 3.1 Software Requirements

1. The system has the following parts:
    1.1. A single handler
    1.2. A non empty set of nodes

   1.3. A set of phones

2. The handler keeps track of all phones in the system and their status.

   2.1. The status of a phone can be ready, offline, dialing, ringing, malfunctioning, and talking with XX. Where ready describes the state where it is connected to the system, but is neither dialing, ringing or talking with XX. Offline describes a known phone that at the moment is not connected to the system. Dialing describes the state where a phone wants to connect to another one, but the connection was not yet established. Ringing describes the case where a phone is getting called, but hasn't yet answered the call. Talking with XX is the case after a successful connection is established and the call got answered, where the XX stands for the other phone involved. Malfunctioning describes a state where a phone is malfunctioning and therefore not reachable.

3. The handler keeps track of all nodes and the distance in meters each phone is away. The availability of a node will also be tracked.

   3.1. All nodes are available, unless they malfunction.

   3.2. This distance will be assigned at random for this simulation.

4. The systems handler needs to print the state of every phone and node in its system, when a change in the state of any phone or node occurs.

5. A phone can initiate a call to another phone at the handler via the address of the target, but it needs to be ready to do so.

   5.1. If the dialed address is not valid or it's the same as the source's address, an error should be returned to the source. The status remains ready.

   5.2. If the target phone address is valid, but the phone is not ready, this information should be returned to the source. And the status ready gets assigned to the source phone.

   5.3. If the address is valid and the target is ready the source gets the status dialing and the target the sate ringing. The handler assigns a node as the route of this call.

5.3.1. The route it assigned is the shortest distance between the phones over an available node.

5.3.2. If the target accepts the call, both phones get the status talking with the other address.

5.3.3. Ringing and dialing has no time limit.

6. A phone can terminate a call at any given time, this includes the status dialing, ringing and talking, and both phones will be assigned the status ready again.

7. A phone can connect and disconnect itself at the handler. A disconnect is only possible when the status is ready or malfunctioning, the system will assign the status offline to it. The phone needs to be known to the system and then will be assigned the status ready.

8. A phone is allowed to connect with the handler although already connected, but a phone should not be able to do this while in a call, this includes the states dialing, ringing and talking.

9. If a node malfunctions, all calls over this node will be terminated and the node becomes unavailable. The phones involved in calls over this malfunctioning nodes gain the status ready. If the malfunction is fixed the node regains the status available, this should be possible with a command.

   9.1. If the last available node malfunctions the system should not be able to route calls anymore. When a phone tries to make a call while no node is available an error should be returned to it and the status remains ready.

10. If a phone malfunctions, the system will assign it the status malfunctioning when the malfunction gets discovered.

    10.1. A malfunctioning phone can not be contacted at all by the system, so the next time a request involves this phone the malfunction will be discovered.

    10.2. A malfunctioning phone can be fixed by connecting it to the system.

11. The systems takes commands via command line, these commands cover every possible action of a phone and include commands to simulate the failure (and fixing) of a phone or node. Alternatively a file can be provided as a series of inputs to it.

12. The system reads a configuration file when starting to have an overview over all known phones and all nodes.

    12.1. If no configuration file was found an error is returned and the system terminates.

    12.2. If an error occurs during the parsing of the config file or the file is malformed an error is returned and the system terminates.

    12.3. If the set of nodes is empty an error is returned and the system terminates.

    12.4. Every phone/node has a unique address/id, that is generated when the configuration file is parsed. The configuration file is a JSON object with two members: phones and nodes. The user can specify the number of generated phones and nodes as a natural number.

### 3.1.1 Example Configuration JSON

```json
{
        "phones": 10,
        "nodes": 3
}
```