UNIVERSITÀ DI PISA

Master Degree in Computer Science, ICT Solution Architect

Peer to Peer and Blockchains

Final Project: Pandemic Flu

Teacher:

Prof. Laura Ricci

Student:

Nunzio Lopardo

600005

Academic Year 2019/2020

Contents

Chapter 1

Introduction

This project aims to implement and study a model describing the diffusion of a virus between smartphones which communicate through Bluetooth connections. Bluetooth is a wireless technology standard used for exchanging data between mobile devices over short distances and has security policies suitable for this experiment. In fact, it provides a coupling mechanism, called $JUSTWORKS$, which allows devices to connect without high security controls. To build this model has been implemented as an application to simulate devices' movements and interactions.

The simulation takes place in a wide area in which different kind of devices moves free, this area is characterized by zones of interests, called *hotspot*, in which the devices aggregate. These hotspots are used to represent places with high density in the real-world that could be, for instance, supermarkets, bars, restaurants, malls, etc.... The devices can move between these places and when they are in can stop or explore them. The following chapters will explain in detail all the development steps. In chapter 2 will present the tools through which the model simulation has been realized. In chapter 3 the details of the code will be shown and the implementation choices will be explained. Chapter 4 will show the results obtained using different parameters. Finally, the chapter 5 will conclude the report.

Chapter 2

Developing Tools

The entire development of the software for the simulation of the Pandemic Flu model was done using the Java language, in particular version SE 14.0.2. To examine the results and to build the graphs, Python was used. This choice was made also because the graph analysis tool, NetworkX, is a library developed in the above mentioned language.

2.1 NetworkX

NetworkX [1] is a Python package for the creation, manipulation and study of the structure, dynamics, and functions of complex networks.

Chapter 3

Implementation Details

This chapter will explain the implementation choice of end code details. As said in the introduction, the scenario that has been created to implement the Pandemic Flu model provides a wide area in which are present hotspots and devices equipped with Bluetooth. To recreate a real-world likely situation, the implementation provides forms of time and space, the execution of a step is equivalent to one minute in reality while space is measured in meters. During the simulation the nodes can assume three different states or behaviors: Halting, Exploring or Travelling. In the first case the device will remain stationary in its position, in the second it will explore the hotspot in which it is located, in the last one it will travel to another hotspot.

3.1 Model

3.1.1 Node

This class is an abstraction of the real-world device. The object of this class can choose the next action to do and in the case in which they are infected to check if there are nodes in their range to infect. Every instance has a series of parameters:

- **id**: a unique identifier for the device;
- **status**: is a variable in which is saved the chosen action, when the node is moving the value of this variable is kept constant until the device reaches its destination;
- **healthCondition**: is 0 if the node is healthy else is 1;
- **os**: this variable indicates the smartphone operating system;
- **os_version**: this variable keeps track of the smartphone OS version;
- **actualLocation**: indicates the actual node position on the map;

- **destination**: this variable contains the destination of the node when its status is exploring or traveling;
- **actualHotspot**: the actual node hotspot;
- **nextHotspot**: this variable contains the next hotspot when the node is in traveling mode.

Also, this class has methods:

- **getNextAction()**: this method compute the next action of the node. Based on the simulation settings, three ranges are defined: halting, exploring, and traveling. When this method is called a random number between 0 and 1 is generating then an if/else statement checks in which range the value falls and returns the relative action.
- **checkNeighborhood()**: this method verifies if there are nodes suitable for virus transmission. To optimize the process this method is called just for the devices that are infected and, if the attribute $actualHotspot$ isn't null, the verification occurs just for the devices in the same hotspot of the infected node.
- **virus_transmission()**: this method verifies the conditions for the virus transmission like OS, OS version, distance and checks the transmission probability.

3.1.2 Hotspot

This class is an abstraction of point of interest in the real-world, zone in which people aggregates and thus increase the possibility to propagate infection between their smartphones. As attributes this class has:

- **id**: a unique identifier for the hotspot;
- **hotspot_center**: the location of the hotspot's center;
- **nodes**: is a HashMap that is used to memorize the nodes that are in the hotspot;

During the simulation, the hotspot instances provide nodes available position to occupy when they decide to travel to the hotspot. This is made by $aviableLocaion()$ that returns destination coordinates to the traveling node.

3.2 Pandemy class

This is the main class of the implementation which is responsible for the initializing of the simulation and the node movement. Also it keeps data structure in which are memorized all instances of nodes and hotspots. The methods within this class are:

- **generateNodes()**: generates the nodes in the area, and keeps track of their position to avoid nodes with the same position. At the end of the generation pick a random node and infect it.
- **generateHotspots()**: this method as his name says generates hotspots in the map and avoid the overlapping of the hotspots area;
- **stepExecution()**: after the initialization, this method is called at each step of the simulation to handle the nodes' action and relative movements.
- **movingNode(Node n, Point destination)**: in this method has been implemented the algorithm to move nodes in the map.

3.3 Implementation Choices

This section of the report will examine implementation choices regarding node and hotspot generation in the map, step execution and node movement.

3.3.1 Map Initialization

The simulation starts with the initialization of the map: nodes and hotspots are placed. In both cases, the coordinates are randomly generated taking care not to create overlaps of the areas covered by the hotspots and not to generate duplicates in the positions of the nodes. This is performed by the two methods *generateNodes* and *generateHotspots*, the result is showed in the 3.1.

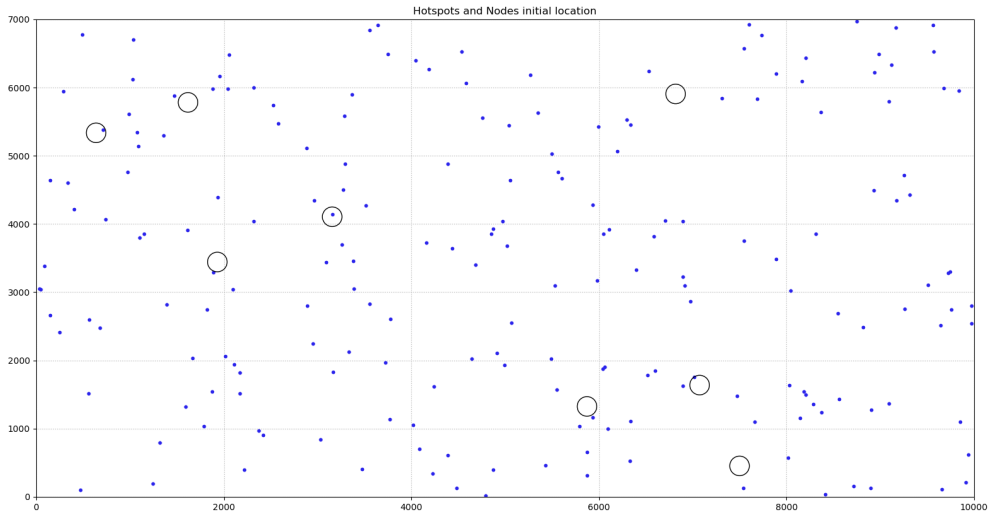

Figure 3.1: The simulation map after initialization

3.3.2 Step Execution

Once the map is filled the simulation starts with a While loop that runs n times the *stepExecution* method. Inside this method the list of nodes is iterated and from each instance is obtained the action to execute through the *getAction* method that returns an integer number between 1, 2 or 3 that corresponds respectively to Halting, Exploring or Travelling. Such integer is passed to a Switch\Case that executes the relative action. At the first iteration of the While cycle, because of the random initialization of the map, the nodes are in random positions and therefore have not yet visited any hotspot so the options are limited to Halting and Exploring and for the following iterations will have all the available actions. When a node decides to move to a new hotspot or explore the one where it is located the *availablePosition* method of the hotspot is called to get an available position. Once the free position is obtained the node starts to move to it. This is done to avoid overlapping positions. When a device starts to move, set its status to Exploring or Traveling and it will remain that until it reaches its destination. Every time an infected node changes position, it checks if there are compatible nodes within its range to transmit the virus to. The infection occurs by checking the compatibility of the operating system and the OS version. In addition to simulating some kind of virus infectivity, a random number is generated and checked if it is below a certain threshold to confirm the infection. The simulation ends at the end of the While cycle and this happens in three cases: the maximum number of steps is reached, the threshold of infected nodes is reached or all nodes are updated.

3.3.3 Nodes Movement

The *movingNode* method is used to manage the movement, which takes the node that is iterated and the destination assigned to it. The nodes move at a constant speed by changing the coordinates at each step until they reach the destination or at a point with a distance less than the speed. In the last case the position is adapted to the coordinates of the destination.

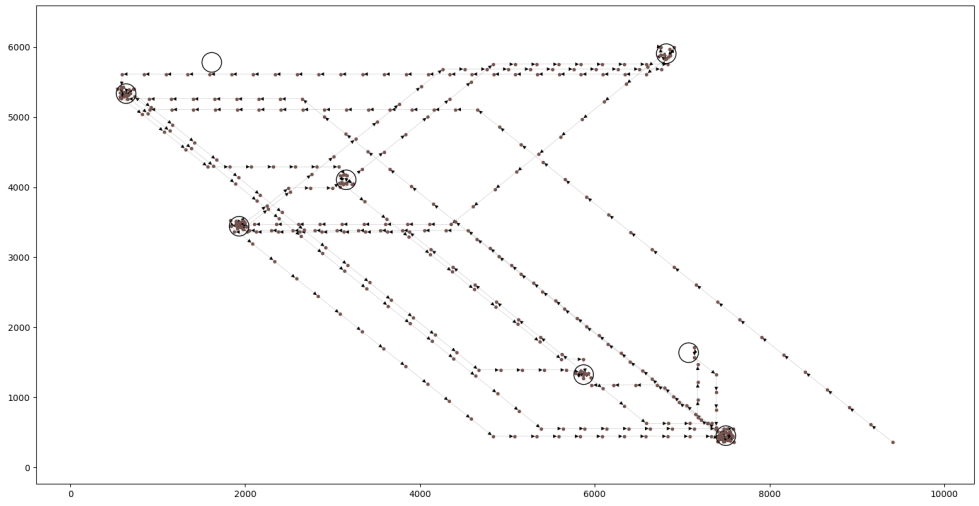

Figure 3.2: Node movements

Chapter 4

Simulations Results

4.1 Settings

All the parameters that need the simulation to work are settable in the class Settings, for instance, map dimensions, infection probability or OS distribution.

- *width*: the width of the map;
- *height*: the height of the map;
- $nodeNumber$: the number of devices that will be generated;
- *nodeSpeed*: a constant nodes speed;
- *nodeRange*: the nodes transmission range, in according with the Bluetooth specification it was set at $10m$;
- $hotspotNumber$: the number of hotspots that will be generated;
- *hotspotRange*: defines the size of the hotspot area;
- $haltingProbability$: is the probability that a node will assume this state;
- $exploringProbability$: represents the probability that a device will start exploring the hotspot it's in;
- $travellingProbability$: indicates the probability of a device going to another hotspot;
- $updateProbablity$: the probability that at the current step the device operating system will receive an update;
- $chance\_to\_get\_infected$: represents the probability that a device will start exploring the hotspot it's in;

- *CrossOS*: a boolean variable that if is true gives the virus the possibility to infect devices with different OS;
- *stepNumber*: indicates the duration of the simulation in execution cycles, one step is equivalent to one minute.

4.2 Simulations

In this section will be present the most relevant simulation and the relative results. All simulations have in common some parameters the number of different OSs (Android, iOS and Windows OS), their distribution and the map dimensions width $20000m$ and height of $15000m$ and so an area of $300km^2$. For each set of parameters have been done five execution to avoid random situation due to bad initial positioning of the nodes or unlucky number generation.

4.2.1 Map Changing

This series of simulations aims to verify how much the environment affects the spread of the virus. In particular, you are performing three simulations by changing the number and size of hotspots. For all three simulations the parameters concerning the probability of contagion, pathcing, speed and probability of node behavior remained constant.

In the first case the parameters are:

- $nodeNumber$: 350
- $stepNumber$: 50000
- $hotspotNumber$: 5
- $chance\_to\_get\_infected$: 0.008
- $updateProbablity$: 0.002

These settings were obtained from a series of simulations and proved to be the most balanced.

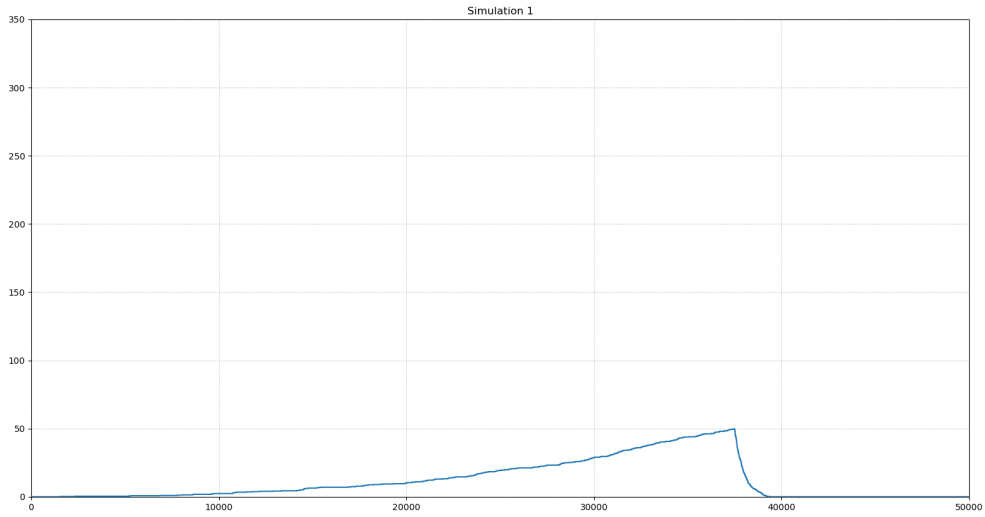

Figure 4.1: Infection trend of the first simulation with five hotspot with range of $100m$.

In this case, at the end of the simulation, the average of 50.2 infection and the curve grows really slow in fact doesn't ever reach the max nodes infection before the patching phase at three-quarters of the steps.

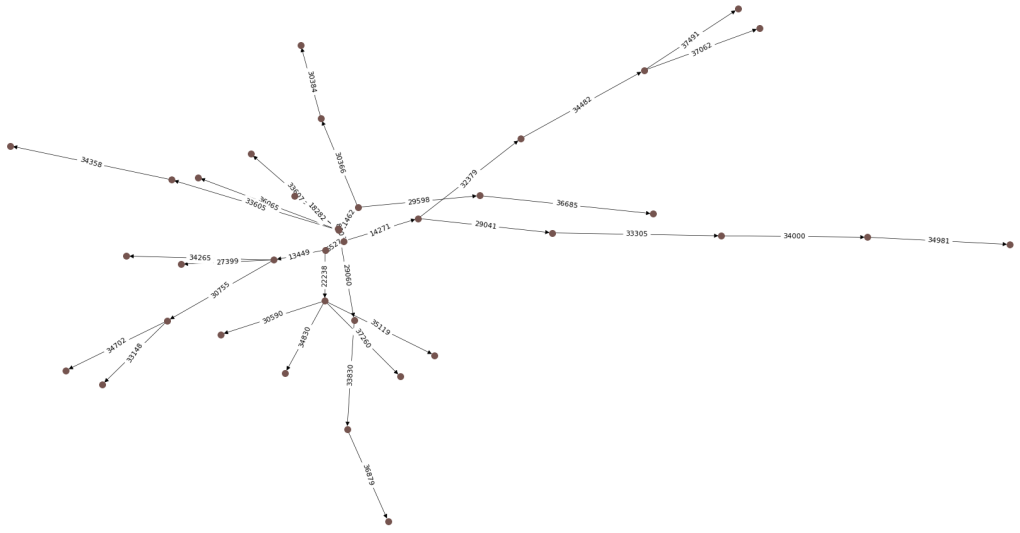

Figure 4.2: The graph shows the infected nodes, the edges are labeled with steps that they contract the virus.

In the second case the parameters are:

- $nodeNumber$: 350
- $stepNumber$: 50000
- $hotspotNumber$: 10
- $chance\_to\_get\_infected$: 0.008
- $updateProbablity$: 0.002

In this case has been tripled the hotspots but keeping the same range. This change has led to a huge drop in infections, this is also given by the fact that having a high number of hotspots reduces the possibility of encountering nodes susceptible to the virus and, in the few cases where it happens, the virus may not be transmitted.

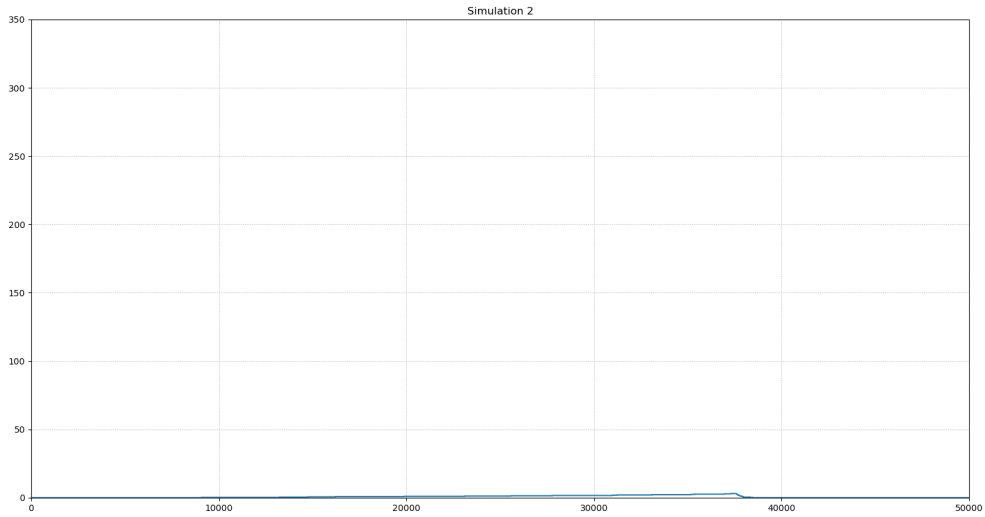

Figure 4.3: Infection trend of the second simulation with fifteen hotspots with range of $100m$.

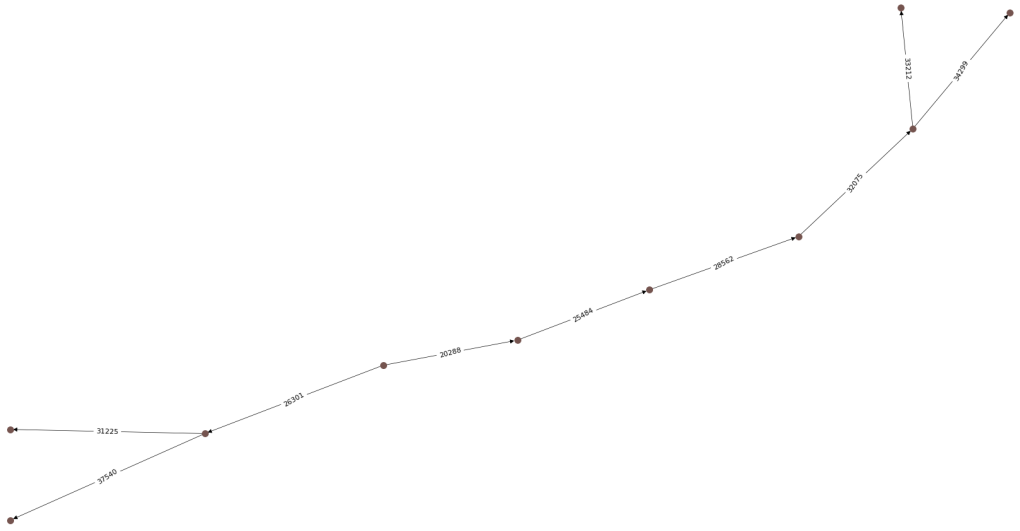

Figure 4.4: Infected nodes in a scenario with fifteen hotspots.

In the third case the parameters are:

- $nodeNumber$: 350
- $stepNumber$: 50000
- $hotspotNumber$: 10
- $hotspotRange$: 50
- $chance\_to\_get\_infected$: 0.008
- $updateProbablity$: 0.002

In the third simulation the number of hotspots was raised to 10 and the hotspots range lowered to 50.

In this case there was a consistent increase in infections for an average of 102.2 among all the simulations performed with these parameters. Moreover, you can see that the contagions curve grew faster, in fact in almost all cases the simulation did not perform all the 50000 steps. Having hotspots with a more limited area tends to reduce the distances between devices and therefore to facilitate infections.

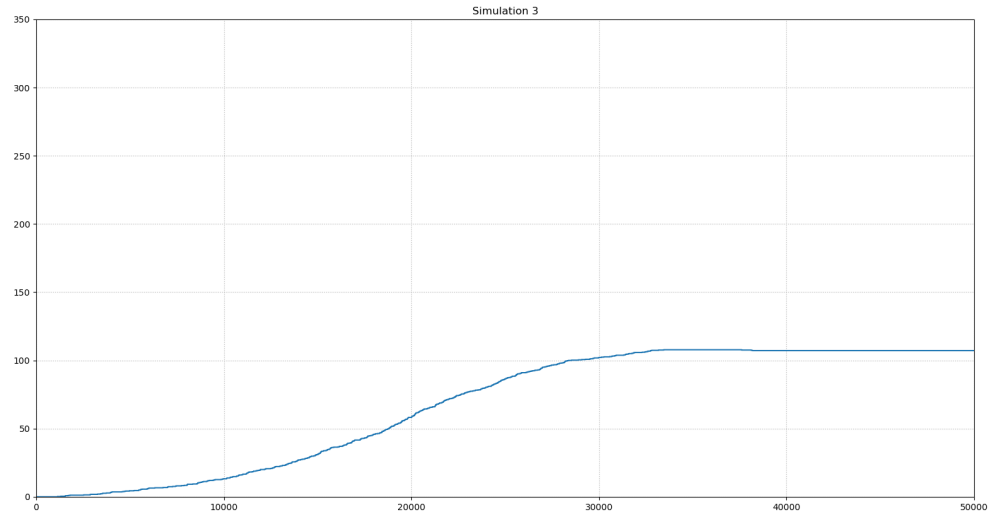

Figure 4.5: Infection trend of the third simulation with ten hotspots with a range of $50m$.

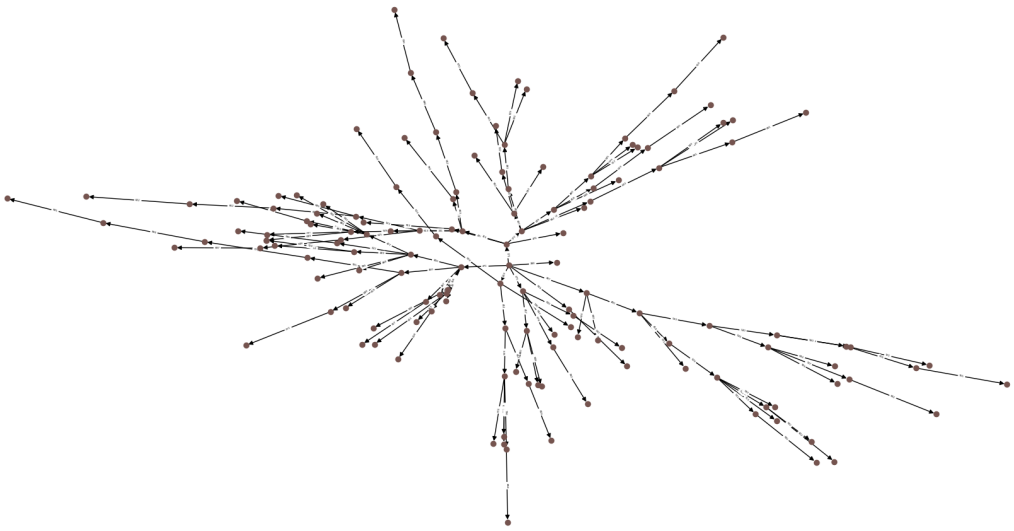

Figure 4.6: Infected nodes in the third simulation with ten hotspots with range of $50m$.

From variations to environmental parameters it emerges that increasing the number of hotspots reduces contagions as devices tend to divide into many small groups. While reducing the range of hotspots leads to a significant increase in infections. So the standard parameters have been updated to find a compromise: seven hotspots and $80m$ range.

4.2.2 Movement Changing

This section will examine the results of the simulation by changing the parameters that affect the movement of nodes, including speed and the probability of performing one action rather than another.

In the first simulation, the speed of the devices was reduced from $350m\backslash m$ to $150m\backslash m$ maintaining the parameters previously used:

- $nodeNumber$: 350
- $nodeSpeed$: 150
- $stepNumber$: 50000
- $haltingProbability$: 0.5
- $exploringProbability$: 0.4
- $travellingProbability$: 0.1
- $chance\_to\_get\_infected$: 0.008
- $updateProbablity$: 0.002

The results showed a small number of contagions, averaging 4.6. This may be due to the fact that they take longer to move around and also the number of encounters with susceptible nodes is reduced.

In the second simulation the standard speed of $350m/m$ was used but the probability of halting was lowered from 50% to 30% and the probability of traveling was raised from 10% to 30%. It results also in this case a drastic drop in contagions, an average of 2 for an average of 413.6 meetings.

In the last simulation only the speed of the nodes has been increased from $350m\backslash m$ to $500m\backslash m$ since the second simulation. In this case the encounters between infected nodes and nodes susceptible to 1918.4 have increased and so have the infections 9.4.In this case, compared to the previous one, the number of encounters turned out to be more influenced by the OS distribution, in fact in the second simulation even though the OS with more devices on the map was infected, the number of encounters remained low.

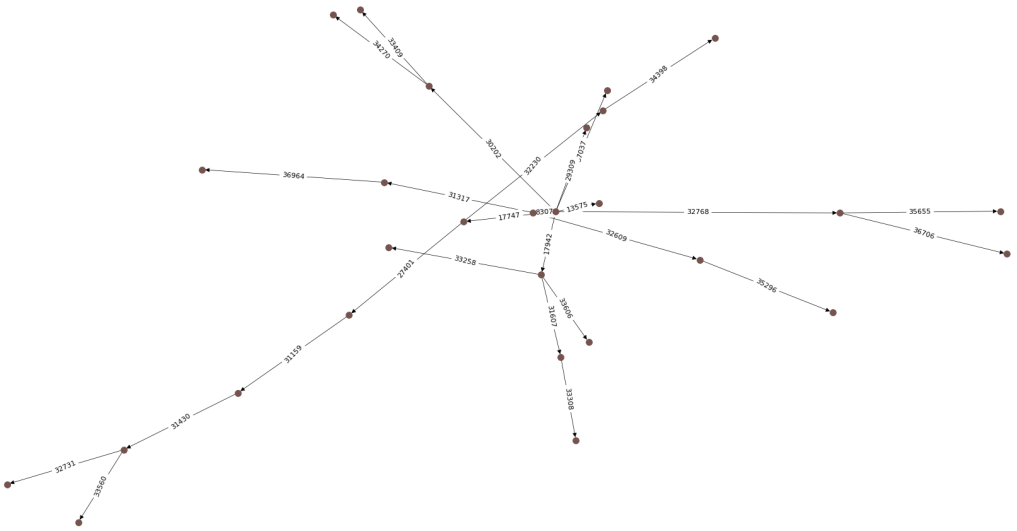

Figure 4.7: Infection graph of the third simulation with node movement speed at $500m\backslash m$.

4.2.3 Infection Changing

For this set of simulations, the parameters that govern the infectious capacity of the virus have been modified to verify the contribution they give to the simulation and analyze the consequences. The modified parameters are only: $chance_to_get_infected$ and the possibility of infecting other OS.

In the first simulation the possibility of infecting devices with different OS has been given and the probability of infection of the previous simulations has been left unchanged, 0.0045.

The results show that the virus, although having an average probability of infecting, is able to infect 80% of total devices within 30000 steps. This is due to the fact that every encounter with an infected node has a chance of being fatal to the healthy node.

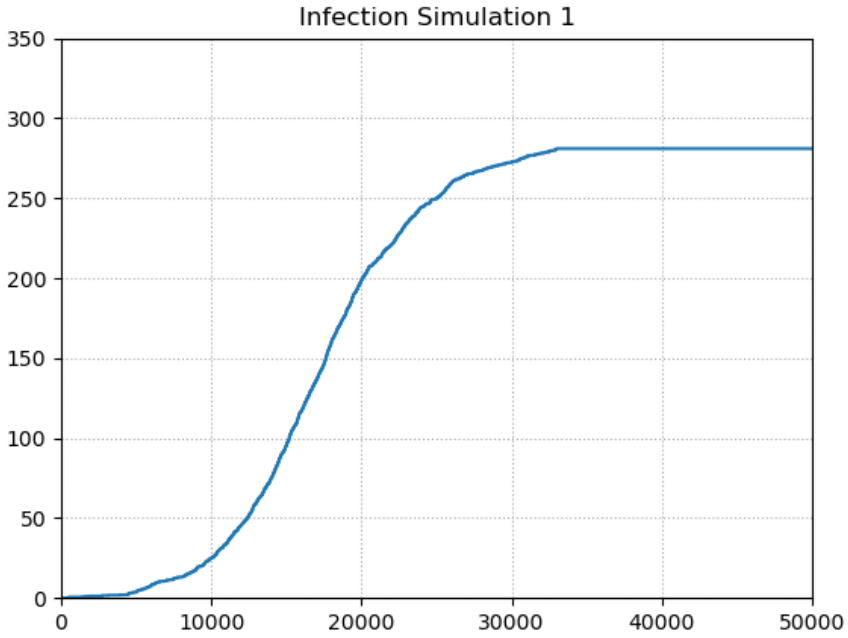

Figure 4.8: Infection trend with a virus that is able to infect also other OS.

In the second simulation the ability to infect other OS was removed but the infectious capacity increased from 0.0045 to 0.065.

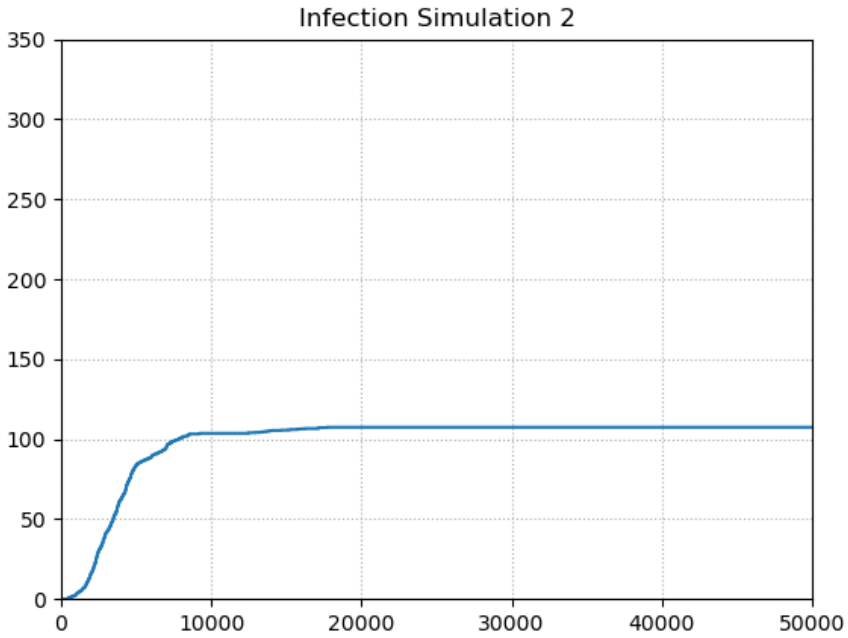

Figure 4.9: Infection trend with a virus increasing the chance to get infected.

This increase has led to excellent results in terms of infections in fact the virus infects 80% of devices with the OS infected within 5000 steps if it is the OS with the highest

presence on the map otherwise 20000 steps for the OS with the lowest distribution.

In the last simulation the patching parameters have been modified, in fact in the previous ones the OS updates were released after about 40000 steps.In this case, the updates begin from step 0. In order to not come patched all the devices in little time the probability of patching has been lowered to $0,00000085$ and raised infectivity to $0,06$.

Using these parameters you can notice a very rapid growth of infected people initially and then be balanced by a constant increase of updated nodes.

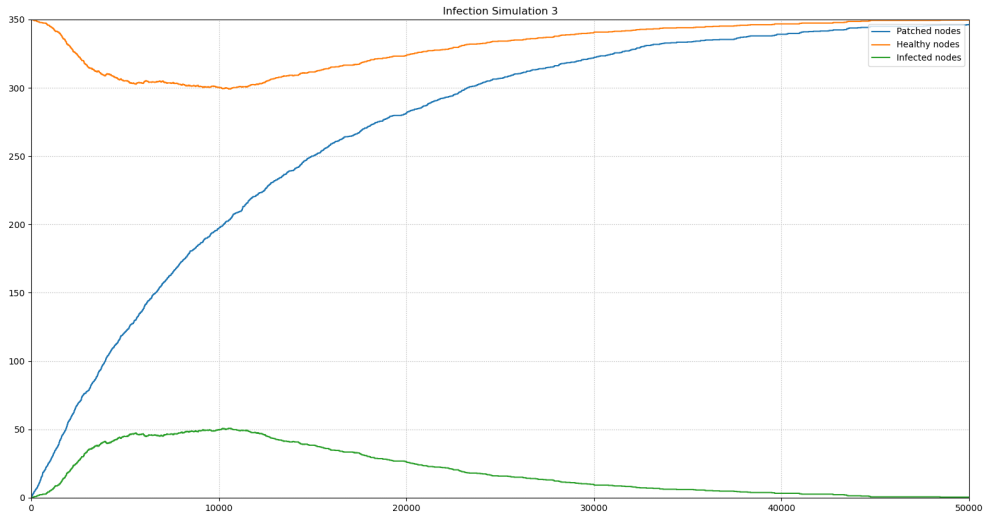

Figure 4.10: Infection trend with earlier update step.

4.3 Main Simulation

Based on the results obtained by modifying the parmeters according to the type of role they play in the simulation, it was possible to obtain a set of balanced settings:

- $width$: 20000
- $height$: 15000
- $nodeNumber$: 350
- $nodeSpeed$: 330
- $hotspotNumber$: 7
- $hotspotRange$: 80
- $haltingProbability$: 0.4
- $exploringProbability$: 0.4
- $travellingProbability$: 0.1
- $updateProbablity$: 0.000025
- $chance\_to\_get\_infected$: 0.06
- $startUpdateStep$: 2500
- $CrossOS$: $false$
- $stepNumber$: 50000

The simulations carried out were experiments in order to find the set of settings to create a situation similar to the real world such as an adequate number of hotspots, the correct speed of the nodes or the right probability of infection.

These settings have created a likely situation, shown in the graph below 4.11 where the node status is shown from three points of view, the blue line indicates the number of infected nodes, the orange line indicates healthy nodes and the green one indicates the number of updated devices and so immune to the virus. The graph (Figure 4.11) shows a constant initial growth of infected nodes in the initial steps and the relative decrease of healthy nodes. From the 2500 step starts the release of updates for operating systems, but it fails to stop the increase of infected nodes, this is due to the fact that the updated nodes are not infected. Constantly growing patched devices curve imposes a limit for the curve of infected nodes that reached this asymptote begins to decline.

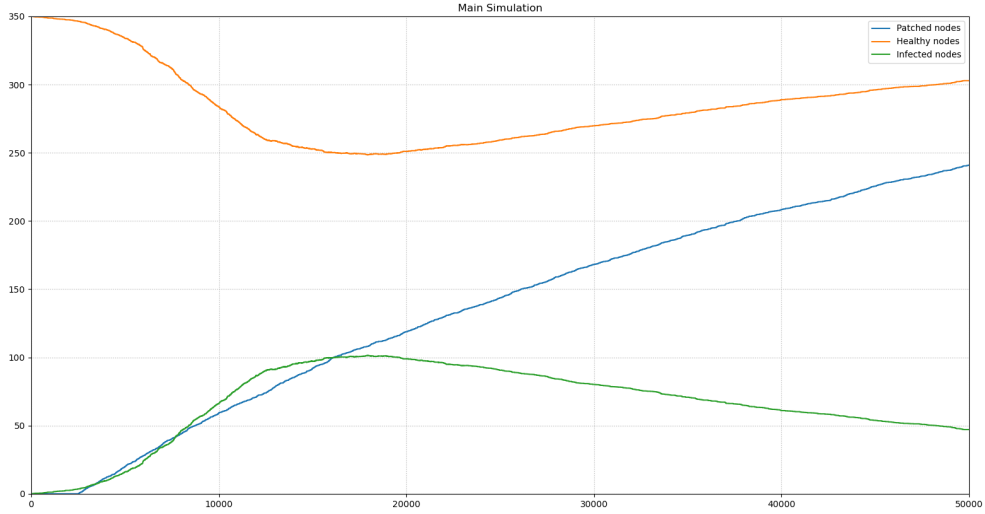

Figure 4.11: Infection trend of the main simulation, the blue line indicates infected nodes, the green line the patched ones and the orange one represents the number of healthy devices.

As far as the resulting graph and its structural features are concerned, it is possible to highlight a series of characteristics. The degree of nodes is the number of edges adjacent to the node, in this case vary from one, for infected nodes that have not re-transmitted the virus, to then increase for those nodes that have infected other devices. The diameter of a graph is the shortest maximum path between two nodes of the network, in this case it is infinite because the graph is not strongly connected and this is due to the presence of devices with different OS. As for the clustering coefficient of the graph, i.e., the measure of the degree to which nodes in a graph tend to cluster together, is equal to 0 because the relationships between nodes are limited to one since the neighbors of a node being already infected do not establish other relationships. Finally, the density of the graph, which indicates how interconnected the entities of a graph are is low because the virus expands only between a certain type of nodes.

Nodes Degrees	$[1:69, 2:32, 3:18, 4:9, 6:5, 5:2, 7:1]$
Diameter	∞
Clustering	0
Density	0.0011051985264019647

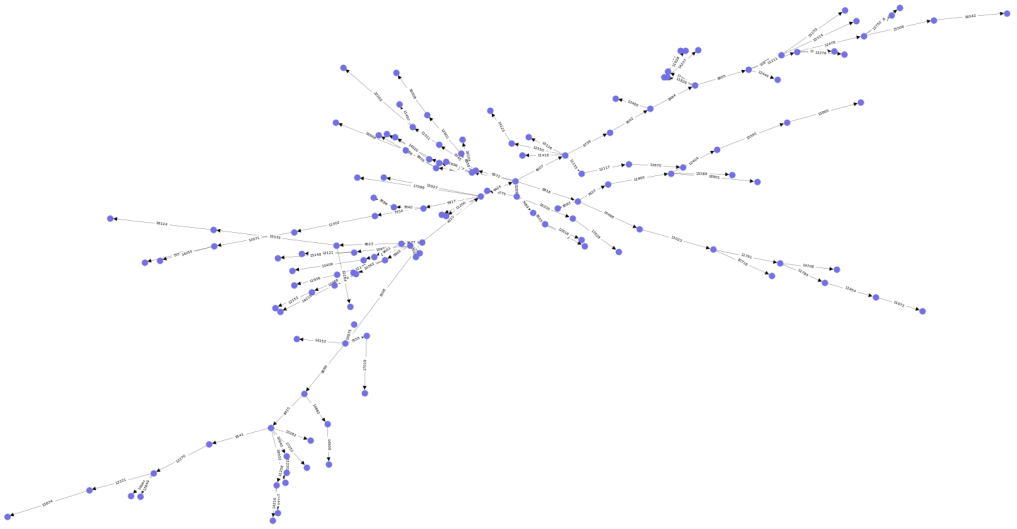

Figure 4.12: Infection graph of the main simulation.

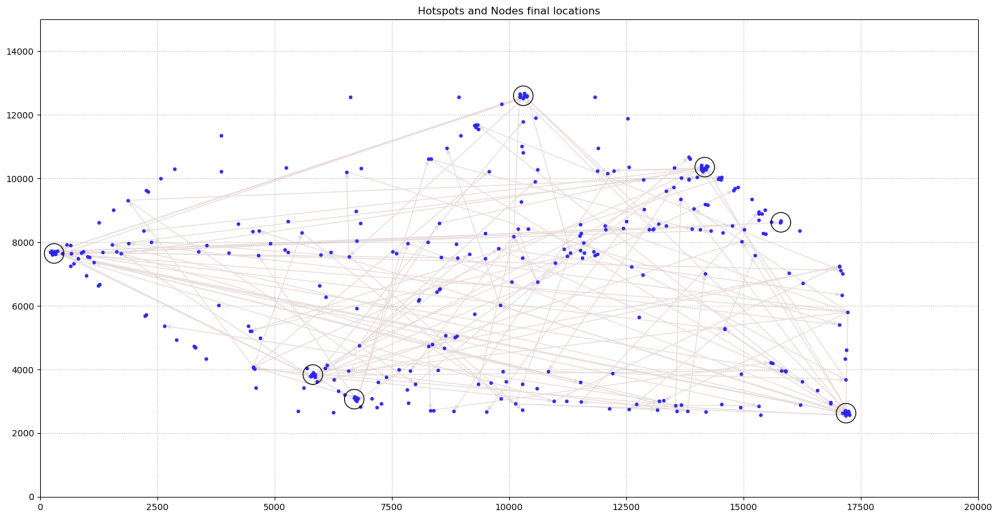

Figure 4.13: Nodes positions and relations at the end of the main simulation

Chapter 5

Conclusion

The objective of this project is to create a model able to simulate the spread of a virus between smartphones in an environment with certain characteristics, such as hotspots. The results showed that several factors can influence the transmission of the virus: the size of the areas where device users cluster, the behaviors they assume, the infectious capacity of the virus and the ability to release updates. Considering the small number of parameters used, the model is not able to represent reality with sufficient accuracy, so it is only qualitative.

Webography

[1] NetworkX `https://networkx.github.io/` 2