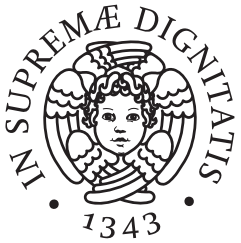


University of Pisa
School of Engineering

Performance Evaluation of Computer Systems and Networks

Epidemic Broadcast

Supervisors
Prof. Giovanni Stea
Ing. Antonio Virdis

Students
Marco Pinna
Rambod Rahmani
Yuri Mazzuoli

March 18, 2021

Contents

Chapter 1

Introduction

In what follows the study on the broadcast of an epidemic message is carried out. The specifications are detailed in the following:

> ### Epidemic broadcast
> Consider a 2D floorplan with N users randomly dropped in it. A random user within the floorplan produces a *message*, which should ideally reach all the users as soon as possible. Communications are *slotted*, meaning that on each slot a user may or may not relay the message, and a message occupies an entire slot. A *broadcast radius* R is defined, so that every receiver who is within a radius R from the transmitter will receive the message, and no other user will hear it. A user that receives more than one message in the same slot will not be able to decode any of them (*collision*). Users relay the message they receive *once*, according to the following policy (*p-persistent relaying*): after the user successfully receives a message, it keeps extracting a value from a Bernoullian RV with success probability p on every slot, until it achieves success. Then it relays the message and stops. A sender does not know (or cares about) whether or not its message has been received by its neighbors.
>
> Measure at least the broadcast time for a message in the entire floorplan, the percentage of covered users, the number of collisions.
>
> In all cases, it is up to the team to calibrate the scenarios so that meaningful results are obtained.

The work is organized as follows:

- Firstly, in Chapter 2, an initial overview and a presentation of the problem are given. Here, meaningful parameters to be tweaked and useful scenarios are identified and some considerations about them are made.
- Secondly, in Chapter 3, a graph-based modelling technique, commonly used in literature, is proposed and some simplified scenarios are analysed.

- In Chapter 4 the development of the simulator is described and the results of its validation are presented.
- Chapter 5 concerns the full simulation and performance evaluation of the system.

Chapter 2

Overview

To perform the analysis of the broadcast of a message that should reach as many users as possible in a 2D floorplan, the following hypotheses were made:

- the floorplan always has a rectangular shape and it is empty, (i.e. there are no obstacles such as walls or pillars in it);
- each user is considered point-like and does not move inside the floorplan;
- the transmission of a message is instantaneous and it happens at the beginning of every time slot; the whole apparatus can therefore be considered a *Discrete Time System*.

Depending on the performance metrics to be analysed, different choices can be made about the parameters to be adjusted. According to the specifications, the main three metrics for this study are:

- the broadcast time `T` for a message to cover as many user as possible in the entire floorplan; the effective duration of the transmissions slots obviously has an effect on the total broadcast time, but it is only a scaling factor on the total number of slots; `T` can therefore be measured in terms of slots and converted to units of time accordingly.
- the percentage of covered users `U`;
- the number of collisions `C`: collisions are detected by nodes, they happen when a node receives more than one message in the same time slot; in order to measure the number of collisions;

The following parameters have been identified: the transmission range of the users, the *per-slot* transmission probability, the floorplan size and its shape, and the density of users in the floorplan per square metre.

To be more detailed:

- the radius of transmission `R`: it represents the maximum distance between two users such that the message sent from one is detected by the other; it is the same for every user on the floorplan.
  R clearly has a great impact on all the performance metrics: the greater this radius, the faster the message moves across the floorplan and the higher the number of users that can be reached; on the other hand, a greater radius is likely to cause more collisions than a smaller one.
  Realistic values for R have been taken from Bluetooth Low Energy standard and range from a minimum of 5 metres to a maximum of 20 metres.

- the *per-slot* transmission probability `p`: it is the success probability for the Bernoullian random variable associated to the transmission. As a probability, it can assume values between 0 and 1.
  The higher the transmission probability p, the faster a message "moves away" from a user; at the same time, a high transmission probability implies a high collision probability in a local area where two or more nodes are transmitting;

- the length of the side of the floorplan rectangle `L`;
  A bigger floorplan area, all else being equal, will require a longer time to be covered entirely by the broadcast;

- the *aspect ratio* of the floorplan rectangle `a`, defined as the ratio between the longer side L and the shorter side W; because of the radial symmetry of the transmission phenomenon, there is no actual need to consider values for `a` lower than 1.
  A very long and very narrow floorplan will probably cause less collisions than a square one with the same area, as the average number of users in the collision range of a random user decreases; on the other hand, the performance of this type of scenario will be highly influenced by the position of the starting node;

- number of users `N`, defined as the number of users dropped on the floorplan;

- users population density `d`, defined as the number of users per square meter; for this parameters, we decided to study two opposite density scenarios (high: $1/m^2$ and low $0.1/m^2$).

For this study, the two main parameters were `R` and `p`. Full sweeps for different values of both were made.
`a` was kept constant and equal to 1 (i.e. the floorplan is always a square) and `L` was either $10m$ ("small" configuration) or $100m$ ("big" configuration). `N` was also kept constant and equal to 100.

Chapter 3

System modelling

Wireless communication networks have been extensively studied in scientific literature and one of the most used mathematical tools to model them and their behaviour is *graph theory*. In this work the same approach was used.
If we think of the users involved in the broadcasting of the message as the nodes of a graph, as the broadcast propagation goes on and the message is transmitted among the nodes, the system goes through different states where each node could be either listening, transmitting or sleeping. We can therefore think of the state of the entire system as a combination of the states of each node. The different nodes behaviours were modelled by means of three different states:

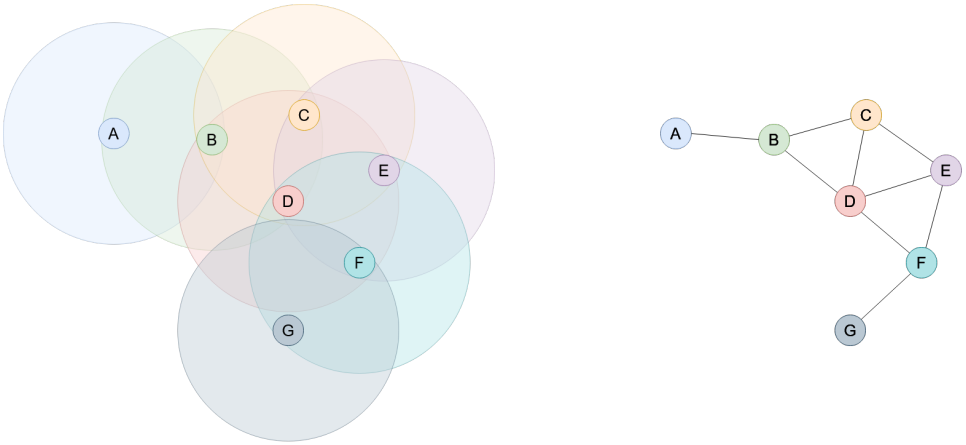
- `listening`: the node has not received the broadcast message yet and therefore it is still listening for incoming messages from other nodes;

- `transmitting`: the node has received the message and during each time slot it is trying to transmit it to adjacent nodes; in what follows, `transmitting` nodes will also be referred to as the *active* nodes;

- `sleeping`: the node has already received and retransmitted the broadcast message; once a node is in the `sleeping` state, it will remain in such state and will therefore have no effect on the system any more.

3.1 Graph model for wireless systems

The N users dropped on the floorplan make up the set of vertices V of a graph G, whose set of edges E is composed by all the connections between pairs of nodes in reach of each other. Consider the following as a simplified scenario to exemplify this model: in figure 3.1 (a) devices A, C and D are within device B transmission radius. In the equivalent graph, there will be edges that connect B to A, to C and to D. The same goes for all the other vertices. The resulting graph is shown in figure 3.1 (b).



a) Ranges representation b) Equivalent graph representation

Figure 3.1

In general, the existence of an edge from vertex i to vertex j means that nodes i and j are within reach of each other. Two vertices connected by an edge are said to be *adjacent*. The neighbourhood of a vertex v in a graph G is the subgraph of G induced by all vertices adjacent to v, i.e., the graph composed of the vertices adjacent to v and all edges connecting vertices adjacent to v. During the broadcast process, a node can only receive from and transmit to its neighbourhood.

Once a node has transmitted the message and has gone into the `sleeping` state, from the point of view of the graph model, it disappears, along with all the edges connected to it. Therefore, the set of vertices V changes with time. This is what in literature is called a *dynamic graph* or, more specifically, a *node-dynamic graph*[**hararygraph**].

Modelling the system with graphs also allows for a simple computation of the lower bound for the broadcast time `T`, which can be useful for validating the simulator or for gaining further insight about the results. Given a graph $G(V, E)$ which represents the users in the floorplan, let node v^* be the starting point for the broadcast, i.e. the first node with the message. In the best case scenario, the system evolves with no collisions at all and the message moves along the paths of the graph, reaching all nodes. Let $d(u, v)$ be the *distance* between two vertices u and v, i.e. the length of the shortest directed path from u to v consisting of arcs, provided at least one such path exists. Then, the lower bound for the broadcast time is given by the highest distance between v^* and any other vertex. This quantity, in graph theory, is called *eccentricity* of the vertex v^*. More formally, the eccentricity of v^* is defined as follows:

$$\epsilon(v^*) = \max_{u \in V} d(v^*, u) \tag{3.1}$$

3.2 Simplified models

In what follows, incrementally more complex scenarios are presented showing the reasoning that led us to obtain the final complete model.

3.2.1 Single queue configuration

Let us consider a configuration where users are arranged in a line, as shown in figure 3.2. Each user only has two neighbours, except for the outer ones (nodes A and E) that only have one. Assuming node A the starting user with the broadcast message, this is the only node in `transmitting` state while all the other nodes are initially in `listening` state.

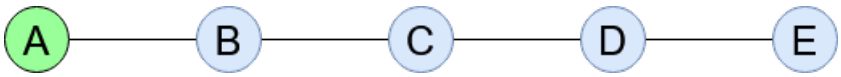


Figure 3.2: Graph of the Single queue configuration

It is clear that, with this configuration, each listening node has a maximum of one active node in its neighbourhood at any time and thus cannot possibly receive the message from two different sources at the same time. This guarantees the absence of collisions. In such a scenario, 100% asymptotic coverage is ensured: during each slot, the active node extracts a Bernoullian RV with success probability p. The probability of the active node transmitting after exactly k slots is a geometric distribution:

$$P(X = k) = (1 - p)^{k-1} \cdot p. \tag{3.2}$$

The successful transmission of the message from an active node to its neighbour is thus guaranteed since $\lim_{k\to\infty}(1-p)^{k-1} \cdot p = 0$. As for the total broadcast time `T`, on average it is equal to the mean value of the geometric distribution, $\frac{1}{p}$, times the number of hops needed to reach the last node:

$$E[T] = \frac{1}{p} \cdot (N - 1). \tag{3.3}$$

3.2.2 Star configuration with one active node (star 1-to-5)

Another useful simple configuration worth analysing is the star-shaped one. In this setup, there is a central node A connected to $N-1$ nodes, all of which are non-adjacent to each other. Let us suppose A to be the broadcast starter. The absence of collisions is ensured in this scenario as well, for the same reason as the previous configuration. At each time slot, A extracts a Bernoulli RV. When the extraction is successful, A broadcasts the message to all of its neighbourhood and total coverage is reached. Hence, 100% asymptotic coverage is ensured in this case too, as the probability of A not transmitting for k consecutive slots is given by Eq. 3.2 and goes to 0 as k goes to infinity.

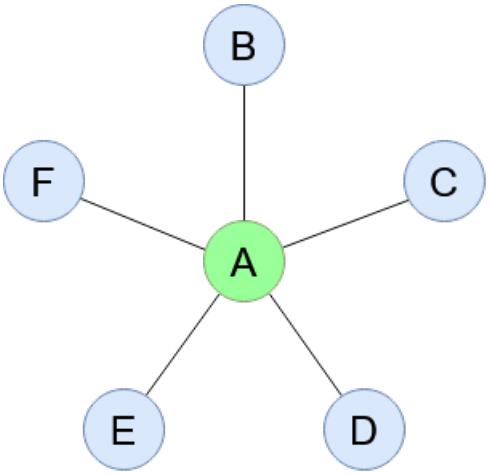


Figure 3.3: Graph of the star configuration with one active node

In this case, the average total broadcast time $E[T]$ is simply equal to the mean value of the geometric distribution, $\frac{1}{p}$, since the central node is placed at one hop from all the remaining nodes.

If the broadcast starter node was not the one placed in the center of the star, there would not be much difference: absence of collisions and total coverage would be ensured as well. As far as it concerns the total broadcast time `T`, its expected value would just be $\frac{2}{p}$, since the are now **two** hops involved in the broadcast: one from the starter node to the center node and the another from the center node to all the $N-2$ remaining ones.

3.2.3 Star configuration with all nodes active but one (star 5-to-1)

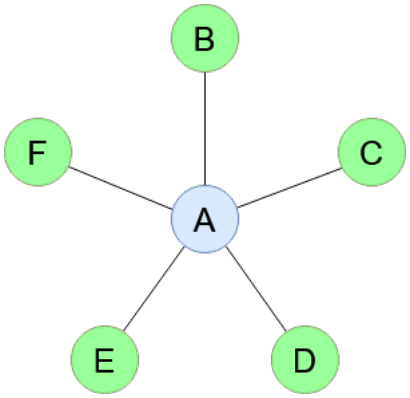


Figure 3.4: Graph of the star configuration with all nodes active but one

This configuration can be seen as the complement of the previous one: each node on the rays of the star is active and trying to transmit the message to the center node. This is actually the first scenario where collisions might happen with the possibility that total coverage might never be reached. To simplify the analysis of this system and obtain some insights, it is useful to model it by means of a discrete-time Markov chain.

Discrete-time Markov chain model for N nodes transmitting to a target

Since the state of the system evolves only once per slot, it can be modelled with a discrete-time Markov chain (DTMC) as follows:

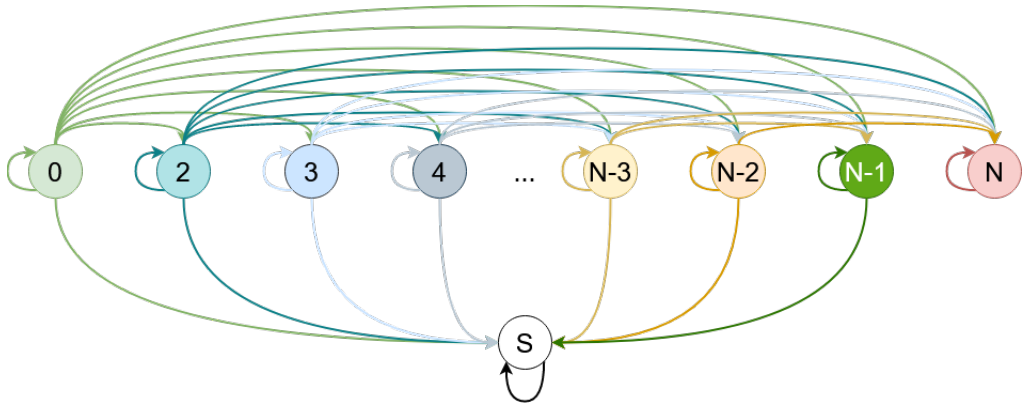


Figure 3.5: Discrete-time Markov chain for a scenario with N active devices

Figure 3.5 shows the DTMC for a generic configuration with N transmitters (transition probabilities are not shown for the sake of clarity).
State 0 and states 2 to N represent the number of `sleeping` devices, namely devices that have already sent the message and have stopped. The number of `sleeping` devices was chosen over the number of `transmitting` devices as the former only allows for non-decreasing sequences of states, since a device cannot become active again once it enters the `sleeping` state.
State S represents the successful transmission state, which the system transitions to when only one device has transmitted the message during the previous time slot.
The initial state X_0 is 0. Any transition from a state i to a state j, $j \neq S$, means that more than one device has transmitted the message, resulting in the target device detecting a collision. Both state S and state N are *absorbing states*, i.e. states that, once entered, cannot be left (as can be seen in Figure 3.5, where the only outgoing arrow from these states goes back to the state itself).
If the system transitions to state N, it stays in it indefinitely since all the devices would be `sleeping` and they cannot become active again. This implies that there will never be total coverage since the target device will forever stay in the `listening` state without ever receiving the message. On the other hand, state S, despite being an absorbing state

as well, should actually be considered as an "exit" state rather than a "sink" state: if the system transitions to this state, the target device has successfully received the message and total coverage has been reached.
Another interesting observation concerns state $N-1$: if the system reaches this state, as there is one and only one remaining node which can transmit, it is guaranteed that the target device will sooner or later receive the message, for the same reason set forth in 3.2.1.

Transition probabilities

Let us now address the challenging part: computing the transition probability.
During the first slot, the probability that j devices out of N transmit the message is given by:

$$P_1(j,N) = \binom{N}{j} p^j (1-p)^{N-j} \tag{3.4}$$

Derivation of (3.4) can be found in Appendix A [6.1].

As for the probability of j devices transmitting at the same time during slot k, be it $P_k(j)$, we can model the system as if it was in the first slot, with the total number of active devices now being equal to N - t, where t is the total number of devices that have transmitted up to the (k-1)-th time slot.
Carrying on with the computation of the transition probabilities this way, leads to unsatisfactory results which are difficult to interpret.

A better way to compute the probability of having j devices sleeping at slot k, is to use the *stochastic matrix* of the Markov chain. If the probability of moving from state i to j in one time slot is $Pr(j|i) = P_{i,j}$, the stochastic matrix P is given by using $P_{i,j}$ as the i-th row and j-th column element, e.g.

$$P = \begin{bmatrix} P_{0,0} & P_{0,S} & P_{0,2} & \dots & P_{0,j} & \dots & P_{0,N} \\ P_{S,0} & P_{S,S} & P_{S,2} & \dots & P_{S,j} & \dots & P_{S,N} \\ P_{2,0} & P_{2,S} & P_{2,2} & \dots & P_{2,j} & \dots & P_{2,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ P_{i,0} & P_{i,S} & P_{i,2} & \dots & P_{i,j} & \dots & P_{i,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ P_{N,0} & P_{N,S} & P_{N,2} & \dots & P_{N,j} & \dots & P_{N,N} \end{bmatrix}$$

Since S and N are absorbing states, $P_{S,j} = 0$ for j $\neq$ S and $P_{N,j} = 0$ for $j \neq N$.
Moreover, all the possible transitions can only generate a non-decreasing sequence of states and there are no transitions between states that differ by one, hence $P_{i,j} = 0 \forall i > j$ and $\forall j = i+1$.
All the other elements of the matrix can be computed using the following formula:

$$P_{i,j} = \binom{N-i}{j-i} p^{j-i} (1-p)^{N-j} \tag{3.5}$$

and $P_{i,S}$ is given by:

$$P_{i,S} = \binom{N-i}{1} p(1-p)^{N-i-1} \tag{3.6}$$

which is just a particular case of (3.5) when $j = i + 1$.
Therefore the stochastic matrix becomes:

$$P = \begin{bmatrix} P_{0,0} & P_{0,S} & P_{0,2} & \dots & P_{0,j} & \dots & P_{0,N} \\ 0 & 1 & 0 & \dots & 0 & \dots & 0 \\ 0 & P_{2,S} & P_{2,2} & \dots & P_{2,j} & \dots & P_{2,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & P_{i,S} & 0 & \dots & P_{i,j} & \dots & P_{i,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 & \dots & 1 \end{bmatrix}$$

Let x_0 be the *initial state vector*, i.e. an $N \times 1$ vector that describes the probability distribution of starting at each of the N possible states.
To compute the probability of transitioning to state j in **k** steps, it is now sufficient to multiply the initial state vector x_0 by the stochastic matrix raised to the k-th power, e.g.

$$P_k(j) = (x_0 \cdot P^k)_j \tag{3.7}$$

In our case, the system always starts in state 0, so we have

$$x_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

which yields

$$P_k(j) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \cdot P^k = (P^k)_{0,j} \tag{3.8}$$

Calculating the k-th power of a matrix can be an intensive task from a computational point of view. To improve the complexity of the computation, rows and columns of P

can be rearranged, moving the S row and the S column as penultimate, thus obtaining

$$P = \begin{bmatrix} P_{0,0} & P_{0,2} & P_{0,3} & \dots & P_{0,N-1} & P_{0,S} & P_{0,N} \\ & P_{2,2} & 0 & \dots & P_{2,N-1} & P_{2,S} & P_{2,N} \\ & & P_{3,3} & \dots & P_{3,N-1} & P_{3,S} & P_{3,N} \\ & & & \ddots & \vdots & \vdots & \vdots \\ & & & & P_{N-1,N-1} & P_{N-1,S} & 0 \\ & & & & & 1 & 0 \\ 0 & & & & & & 1 \end{bmatrix}$$

P is now an upper triangular matrix and this allows for faster computation of its powers in 3.8.

Chapter 4

Simulator

In order to obtain experimental results for the presented scenarios, a simulator was built using OMNeT++ with the support of the INET framework. This allowed us to reproduce the different scenarios, presented in the previous chapters, with different values for the identified parameters.

4.1 Omnet++ and INET framework

OMNeT++ is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators[1].
The INET Framework is an open-source model library for the OMNeT++ simulation environment. It provides protocols, agents and other predefined models for researchers and students working with communication networks. INET is especially useful when designing and validating new protocols, or exploring new or exotic scenarios[2].

OMNeT++ is a library and a framework, and can be used with the dedicated IDE. Not only it allows for development of the simulator itself, but also to export simulation results and to inspect simulation behaviour with a graphical user interface. By taking advantage of the C++ compiler optimizations, it can achieve the lowest simulation duration possible.

Networks are composed by modules; there are two types of modules: simple module and compound module (which can contain other modules itself). INET, on the other hand, is an extension of OMNeT++, dedicated to recreating network simulation environments, with the capability of reproducing the activity of a wireless communication system across multiple nodes. It contains ready to use definitions and implementations of network related modules.
The INET framework was chosen in order to avoid spending too much time on the coding

[1] https://omnetpp.org/
[2] https://omnetpp.org/download-items/INET.html

side and therefore be able to focus more on other aspects such as the problem modelling and analysis.

4.2 Network architecture

The network based architecture is composed by an array of `Host` modules, an `Integrated visualizer` (visualizer) and a `UnitDiskRadioMedium` (radioMedium);

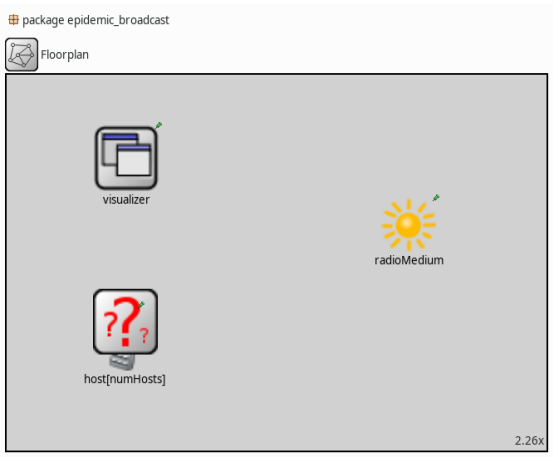


Figure 4.1: Floorplan.ned

- **UnitDiskRadioMedium** is a compound module provided by INET. This radio medium model provides a very simple but fast and predictable physical layer behaviour. It must be used in conjunction with the `UnitDiskRadio` model. It can simulate the behaviour of the wireless communication channel with various levels of abstraction.

- **Integrated visualizer** is a compound module provided by INET. It's resposible for the visual representation of modules properties and events in the graphic user interface.

- **Host** is the compound module developed to represent a node in the network environment.

The `Host` module extends the NodeBase module defined by INET. This module contains the most basic infrastructure for network nodes that is not strictly communication protocol related. The following diagram shows usage relationships between types:

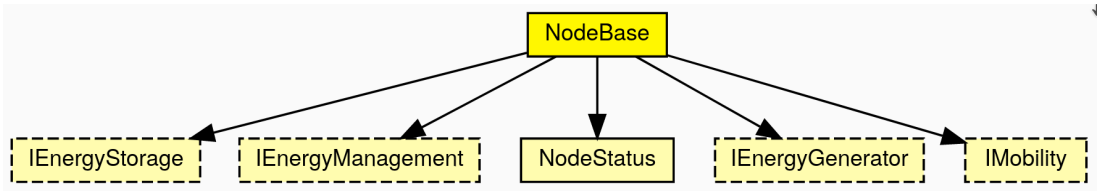


Figure 4.2: NodeBase Diagram.

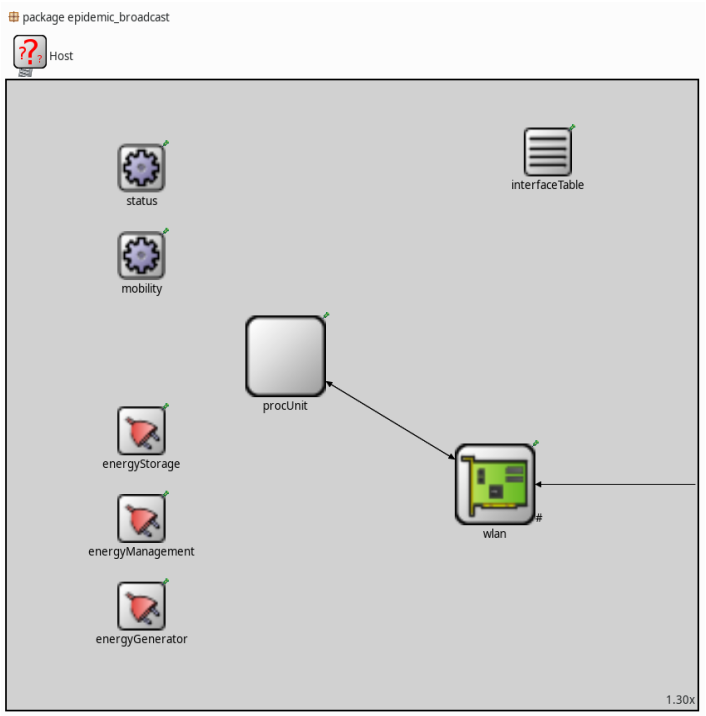


Figure 4.3: host.ned

- the `mobility` module privided by INET manages the position of the parent module `Host`; it allows various types of movements, but in this study was only used for the initial random placement of the nodes. After being placed, all nodes are stationary for the whole duration of the simulation.

- the `interfaceTable` module is provided by INET and is required for correct operation of the radioMedium module.

- the `wlan` module is the wireless interface that allows nodes to communicate with each other. It is an `AckingWirelessInterface` compound module, which is the simplest wireless interface provided by INET.

- the `status` module is provided by INET as well and is required to shut down and restart network interfaces.

- the `procUnit` module is the custom made processing unit, that implements the node behaviour when a message arrives. It is connected to the `wlan` module in order to be able to receive and send messages.
- `energyStorage`, `energyManagement` and `energyGenerator` are modules inherited from `NodeBase` but they are not instantiated as there was no need to model energy-related behaviours.

The `wlan` module is in charge of checking each and every message for collisions, and drop broken packets instead of forwarding them to the processing unit. The processing unit `ProcUnit` implements the behaviours of the nodes; it handles the broadcast message when received, and then its retransmission when the random variable extraction results in a success. Finally it shuts down the network interface, preventing it from receiving any messages or provoking collisions. The network interface is turned off also for the entire duration of the RV extractions.

4.3 Parameters and statistics

During the simulation, signals are used to collect the statistics. They are all collected by the `Floorplan` module:

- The `wlan` module emits a signal every time a collision is detected; this signal is collected by the `packetDropIncorrectlyReceived` statistic of the same module; we are interested in the total number of collisions detected by each node.
- The `ProcUnit` module emits 2 signals when initialized, `hostX` and `hostY`, collected, respectively, by the `hostXstat` and `hostYstat` statistics; those are the coordinates of the parent node in the floorplan.
- The `ProcUnit` module emits the `timeCoverage` singal as well, collected in the `timeCoverageStat` statistic; this is a vector containing, for each node that received the broadcast message, the number of the time slot when the broadcast message was actually received; at the end of the simulation, its size represents the number of covered nodes.

Most significant parameters set up in the initialization file (`floorplan.ini`) are reported below:

- `Floorplan.host[*].procUnit.slotLength = 1`
- `Floorplan.host[*].procUnit.p = 1`

4.4 Design Choices and Optimizations

Using the INET framework for the development of the simulator allowed for the use of pre-built modules for modelling wireless communications; for example, collision detection and statistics collection are already implemented by INET modules. During the

development we choose for every aspect the optimal level of abstraction for our purposes, but it is possible to model other aspects just changing the types of INET modules used, or by adding new ones. We voluntarily avoided taking into account phenomena like path loss and node movement, and we restricted our considerations to a discrete time scenario. However, modelling continuous time scenarios can be done easily by changing few INET modules types and attributes.
INET modules also have pre-built optimization structures, that become indispensable when the number of hosts becomes larger; in order to make the simulator ready for high complex scenarios we used the `neighborCache` structure offered by the `radioMedium` module. This module is in charge of storing proximity information of each and every node, in order to speed up message delivery. By setting the type of this module to `GridNeighborCache`, it is possible to reduce the time needed for a simulation with more than 2000 devices dropped on the floorplan, by a factor of 10; we also found that this type of cache (with the right value for the `cellSize` parameter) is the best trade-off between speed and memory occupancy, for this type of workload[3].

4.5 Validation

To ensure the correctness of the simulator and the meaningfulness of the results, the simulator has been validated by means of two simplified scenarios, namely the single queue configuration and the star configuration, which are discussed in 3.2.1 and 3.2.3 respectively.

4.5.1 Single queue validation

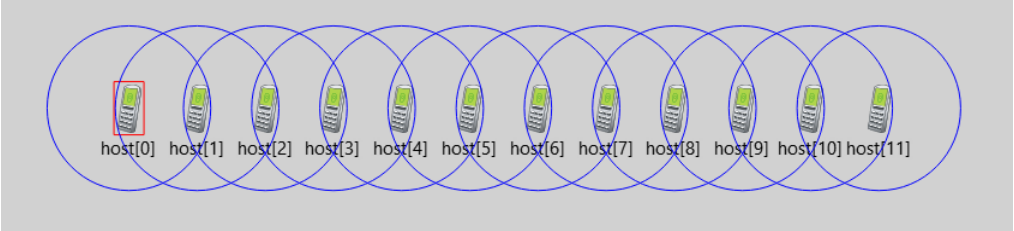


Figure 4.4: Validation configuration with 12 hosts placed on a line

In this configuration, host[0] always broadcasts during the first slot and then the message travels along the queue, with a total of 10 hops. On every hop, the per-slot probability of successful transmission is p, which implies an average number of attempts equal to $\frac{1}{p}$. Therefore, the expected coverage time is

$$E[T] = 1 + 10 \cdot \frac{1}{p} \tag{4.1}$$

[3]https://doc.omnetpp.org/inet/api-current/neddoc/inet.physicallayer.contract.packetlevel.INeighborCache.html

Validation was performed with 9 different configurations, one for each value of p ranging from 0.1 to 0.9, with 200 repetitions each. The following results were obtained:

p	Expected value	Experimental value (mean, interval for 95% confidence)
0.1	101.0	**99.68**, 95.55, 103.81
0.2	51.0	**49.9**, 47.93, 51.86
0.3	34.33	**34.17**, 32.88, 35.46
0.4	26.0	**25.79**, 24.93, 26.65
0.5	21.0	**20.84**, 20.24, 21.44
0.6	17.67	**17.42**, 17.0, 17.84
0.7	15.29	**15.2**, 14.87, 15.52
0.8	13.5	**13.48**, 13.25, 13.72
0.9	12.11	**12.14**, 11.97, 12.3

As can be seen in the table above, the experimental results are consistent with the theoretical expected value, with a 95% confidence level.

4.5.2 Star 5-to-1 validation

In this configuration, host[0] is the target to be reached by the broadcast while all the others already have the message and try to broadcast at every slot with probability p.

This system can be modelled by a discrete-time Markov chain, as explained in 3.2.3. The probability of the system to be in the i-th state during the j-th slot is given by taking the j-th power of the stochastic matrix and taking the i-th element of the first row.

Validation was performed with 4 different configurations, one for each value of p ranging from 0.2 to 0.8 with steps of 0.2, with 1000 repetitions each.
For each configuration, data about the state of the system was recorded and statistics were computed, yielding experimental probabilities. These probabilities were then compared with theoretical predictions, obtained by taking powers of the appropriate stochastic matrix.
All the experimental results proved to be consistent with theoretical computations.

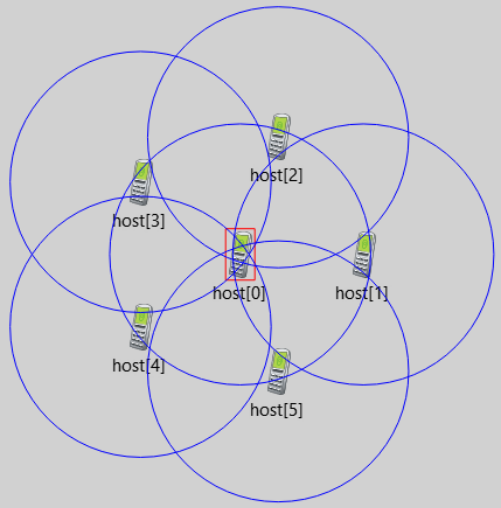


Figure 4.5: Validation configuration with 5 hosts placed on a star and a target in the middle

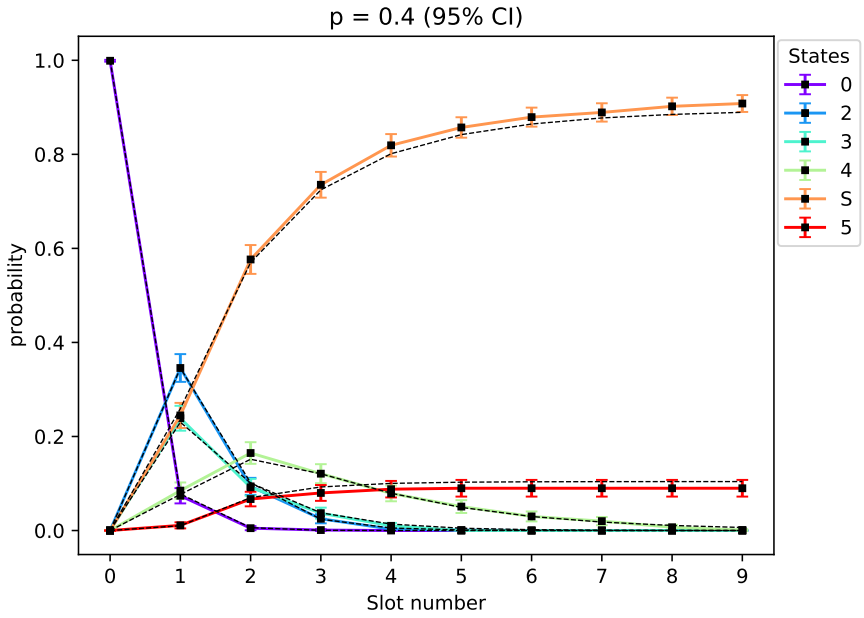


Figure 4.6: Validation data for p = 0.4

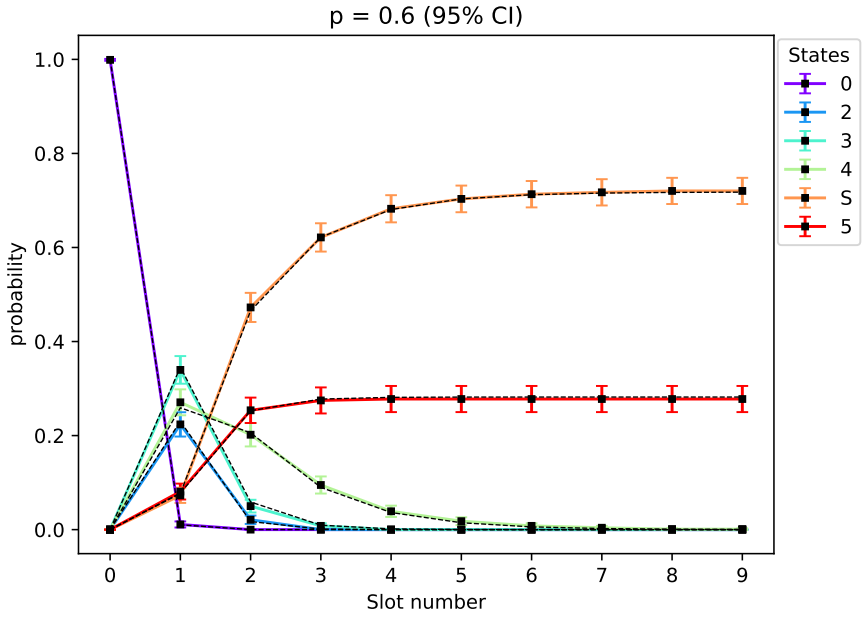


Figure 4.7: Validation data for p = 0.6

Figures 4.6 and 4.7 show graphs of the experimental data (coloured solid lines) along with predictions (thinner black dashed lines).

Chapter 5

Simulation

As previously described, two different scenarios were considered, each with a population of 100 users:

- **Small**: a 10m x 10m floorplan, with the transmission radius ranging from 1m to 4.5m, with a step of 0.5m, and a Bernoullian RV with success probability p ranging from 0.05 to 0.95 with steps of 0.05;
- **Big**: a 100m x 100m floorplan, with the transmission radius ranging from 1m to 19m, with a step of 1m, and a Bernoullian RV with success probability p ranging from 0.1 to 0.9 with steps of 0.1;

Methodological Note

In order to analyse all the possible choices for p and R, we have to explore the space formed by the cartesian product of the possible values for each parameter. Every combination identifies a scenario, for which we have to approximate performance indexes; in our model every index is a random variable, and we want to compute its mean value, with a confidence interval according to a specific confidence level (at least 95%). We do not have enough details to identify the probability distribution of those variables, and we are not even sure that they will be symmetric, so we chose to estimate the mean value of every distribution by the sample mean and also by the sample median. We know that sample mean and sample median converge to the same value for a large number of samples; we simply want to plot the one with the small confidence interval, and we cannot know `a priori` which one will be. For confidence intervals, we chose algorithms that do not require to know the distribution of the samples; in order to apply those algorithms we only have to verify that our distributions have limited variance. We can ensure that because:

- the **broadcast time** is bounded from below (because it cannot be less than 1) and from above (see `single queue configuration`)
- the **percentage of covered users** is bounded from 0.01 (1%) to 1 (100%)

- the **number of collisions** is bounded from below (because it cannot be less than 0) and from above by the value 2305 for a floorplan with 100 hosts. Derivation of the formula for the upper bound can be found in appendix D.

For each of the scenarios, the floorplan coverage, the broadcast duration and the number of collisions are mesured and plotted in graphs in order to extrapolate insighs about the behaviour of the system.
In order to achieve statistically significant results with a minimum accuracy of 95%, each scenario configuration – with the same parameters – was run 200 times; then, mean and median values of the performance indexes were computed, along with their confidence intervals.

5.1 Big

5.1.1 Coverage

Figure 5.1 shows the floorplan coverage achieved as a function of the success probability p, for different values of the transmission radius R.
As we can observe, the coverage is maximum for lower values of p; this is due to the fact that a low transmission probability minimizes the number of collisions, increasing the number of correct transmissions. For values of $R < 8$ we observe an almost linear behaviour: collisions are almost absent with such a small transmission radius; therefore, variations of the value of p do not play a fundamental role as far as total coverage is concerned. For $R \geq 8$ the achievable coverage decreases with the increase of p, but quite slowly and also it does not degenerate to 0; instead it reaches a value between 10% and 40% of the maximum.

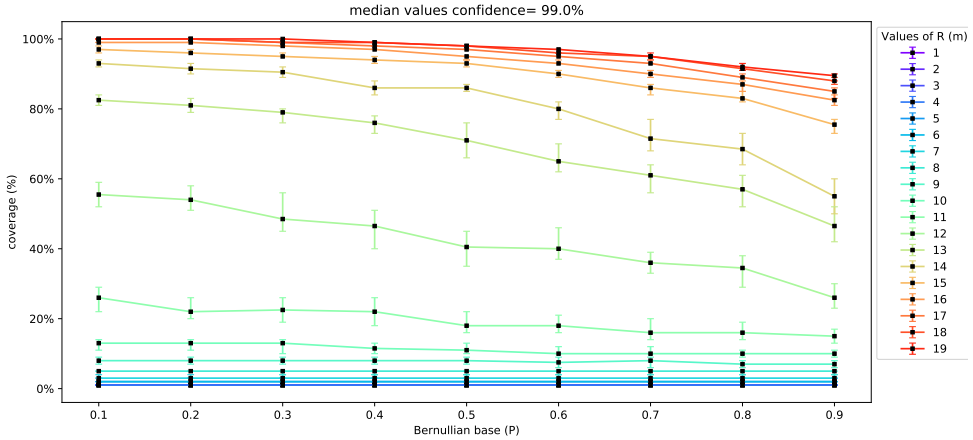


Figure 5.1: Floorplan coverage as a function of p for different values of R

For values of R between 11 and 15, the number of collisions has a strong effect on the

coverage percentage; this effect tends to decrease for larger values of R; we know as a matter of fact that for $R \to L\sqrt{2}$ the coverage tends to 100%, independently of any other parameter.

Figure 5.2 shows the floorplan coverage achieved as a function of the transmission radius R, for different values of the success probability p. As expected, accordingly with the previous plot, the final coverage of the flooplan is deeply influenced by the transmission range R.

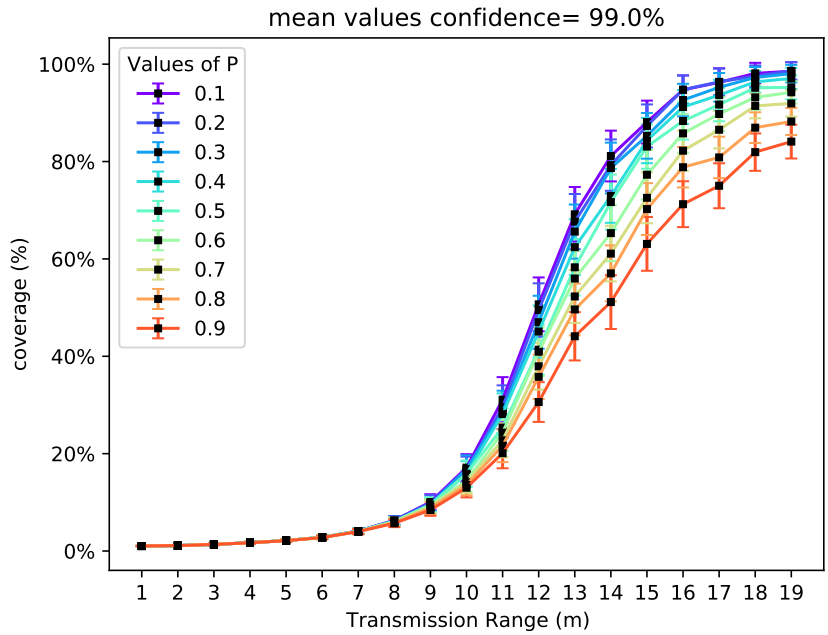


Figure 5.2: Floorplan coverage as a function of R for different values of p

In this second plot we can observe that the coverage increases exponentially with the transmission range. When the transmission range becomes large ($>$ 16m), the coverage tends to 100%. We recognized the shape of the sigmoid[1] function, which can be express as an hyperbolic tangent (tanh); this function is always increasing and can be manipulated to remain between 0 and 1 (like the parameter we want to fit). A good fit for this curve is given by:

$$C = \frac{1 + \tanh(aR + b)}{2}$$

where C is the floorplan coverage, R the transmission radius, and a and b depend on p as shown in the following table:

[1] https://mathworld.wolfram.com/SigmoidFunction.html

p	a	b
0.1	0.3628314535334378	4.356732935967713
0.2	0.3572920723369711	4.3206795324164835
0.3	0.34371134773480694	4.193385575628803
0.4	0.3217942874156316	3.9814759531755515
0.5	0.3091448275788839	3.88645016495332
0.6	0.2809863550485405	3.6000814495973996
0.7	0.25932462296148107	3.3996924196436975
0.8	0.23603288616377227	3.1648817545125527
0.9	0.20917405109220505	2.929033169711656

5.1.2 Duration

This following figure shows the plot of the duration of the simulation (in seconds) as a function of p, for different values of R.

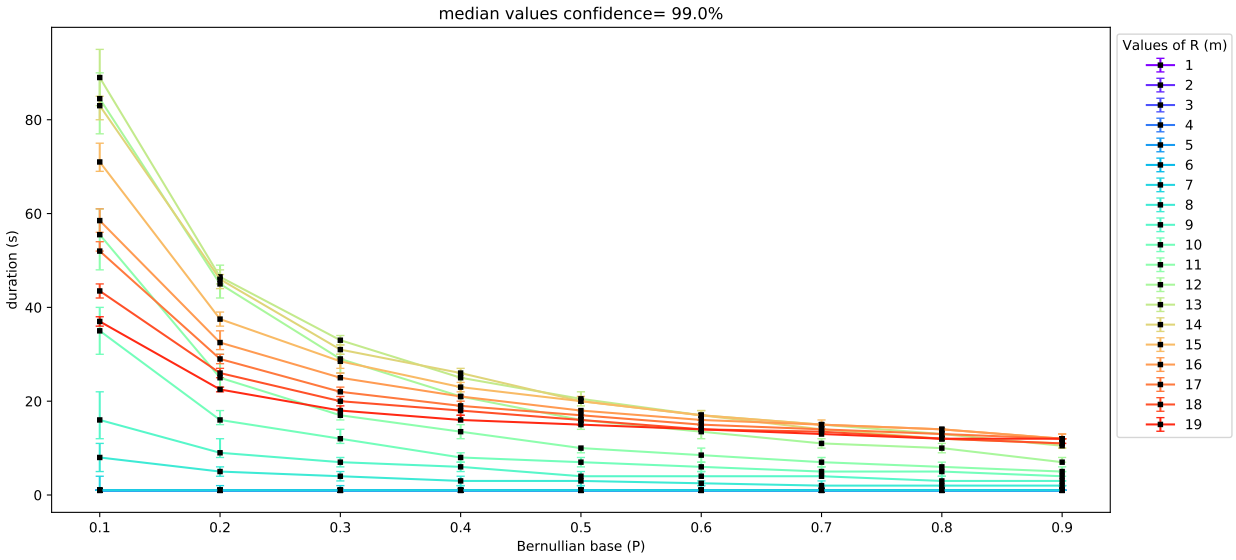


Figure 5.3: Simulation duration as a function of p for different values of R

The simulation duration increases for lower values of p; this behaviour can be explained as a consequence of two main factors:

- the probability of retransmission is low, so nodes will spend most time slots waiting, without actually transmitting;
- with a smaller number of collisions, a higher number of nodes is reached; reaching a higher number of nodes requires a higher number of retransmissions.

We tried to interpolate the curves shown in Figure 5.3 with an hyperbolic function, because it fits well the parameters we want to model; first of all, the hyperbole tends to infinity as p tends to 0, and this is coherent with the reality, because if the probability of retransmission becomes low, then the duration keeps increasing. If instead p gets close to 1, the duration time tends to its minimum. We fit those shapes using:

$$D = \frac{a}{P} + b$$

where D is the duration of the simulation, p the probability of retransmission, and a and b depend on R as specified in the following table:

R	a	b
1	2.4046845625846913 e-09	0.999999999244136
2	2.4046845625846913 e-09	0.999999999244136
3	0.7313966348173997	0.9495446821974095
4	1.3471717522270503	0.9454326532617134
5	1.7744177255115228	1.08724762049118
6	4.652279267805434	1.0487610714204172
7	9.586489452244717	1.088347295851843
8	15.841670397411658	1.0643797063994824
9	28.264640360031194	0.49558107430696624
10	43.141734847778814	0.22815570364285975
11	64.55944043148898	−0.07517936077209139
12	86.56257725232254	−0.09919809415424388
13	89.72686483087844	1.2578386401845827
14	82.15919574473708	3.0965825999956778
15	67.95268283187643	5.415446394742158
16	56.442159145137495	6.991880399220213
17	45.71412729773696	7.548465010848107
18	39.164500694866284	7.991652317096266
19	29.71836400235831	9.075854630828385

To be able to compute the correct values the first two entries for a and b, a larger dataset is needed; as a matter of fact, if the transmission range is too short nodes can not communicate: they are not in reach by each others.

The following plot shows the duration of the simulation (in seconds) as a function of R, for different values of p.

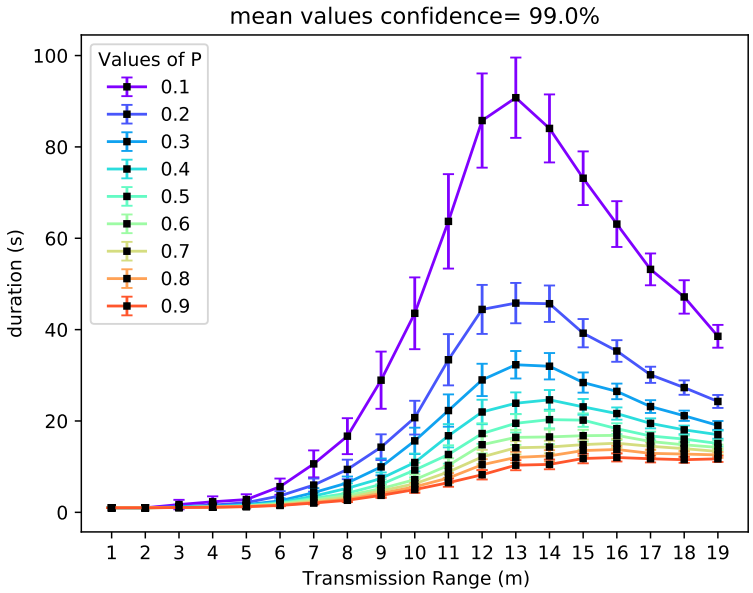


Figure 5.4: Simulation duration as a function of R for different values of p

The duration of the simulation tends to increase with the transmission range reaching a peak around 13m; the maximum value depends on the probability of retransmission (p); This is an interesting result, and can be explained by:?

5.1.3 Collisions

This plot show the Number of collisions in function of P, for different values of R.

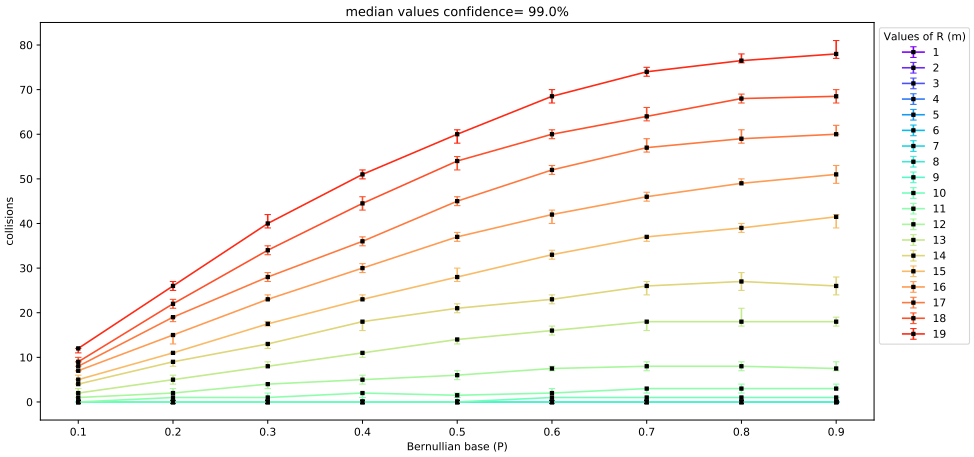


Figure 5.5: Caption this

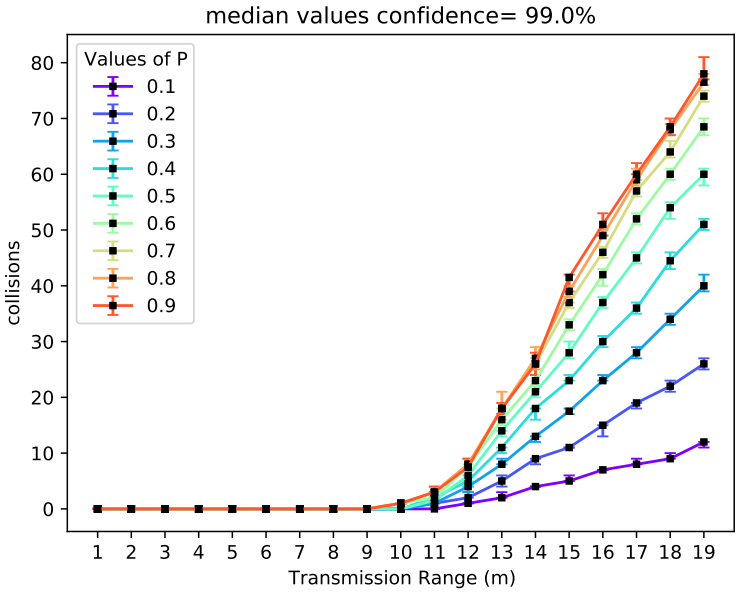


Figure 5.6: Caption this

Chapter 6

Appendices

6.1 Appendix A

Given a scenario with N transmitter devices and one target device T in reach by all of the transmitter devices, let us define the probabilities $P_1(j, N)$ as the probability of j devices out of N transmitting at the same time during the time slot 1 and $P_i(j)$ as the probability of j devices transmitting at the same time during slot i.
By specification, the successful reception of the message by device T occurs when **one** and only one of the transmitters sends the message during the time slot. Furthermore, the successful transmission of a device is "*a Bernoullian RV with success probability* p *on every slot, until it achieves success*". Therefore, we can model $P_1(j, N)$ as follows:

$$\left.\begin{aligned}
&P_1(0,N) = (1-p)^N \\
&P_1(1,N) = Np(1-p)^{N-1} \\
&P_1(2,N) = \binom{N}{2}p^2(1-p)^{N-2} \\
&P_1(3,N) = \binom{N}{3}p^3(1-p)^{N-3} \\
&\dots \\
&P_1(N-1,N) = \binom{N}{N-1}p^{N-1}(1-p) \\
&P_1(N,N) = \binom{N}{N}p^N
\end{aligned}\right\} P_1(j,N) = \binom{N}{j}p^j(1-p)^{N-j}$$

As for $P_i(j)$ we can model the system as if it was in the first time slot, with the total number of active devices now being equal to $N - t$, where t is the number of devices that have transmitted in the $(i-1)$-th slot.

6.2 Appendix B

An example of stochastic matrix for a system with $N = 5$ and $p = 0.4$:

$$P = \begin{bmatrix} P_{0,0} & P_{0,2} & P_{0,3} & P_{0,4} & P_{0,S} & P_{0,5} \\ & P_{2,2} & 0 & P_{2,4} & P_{2,S} & P_{2,5} \\ & & P_{3,3} & 0 & P_{3,S} & P_{3,5} \\ & & & P_{4,4} & P_{4,S} & 0 \\ & & & & 1 & 0 \\ 0 & & & & & 1 \end{bmatrix}$$

Here with numerical values (rounded to 4 decimal places):

$$P = \begin{bmatrix} 0.0778 & 0.3456 & 0.2304 & 0.0768 & 0.2592 & 0.0102 \\ & 0.216 & 0 & 0.288 & 0.432 & 0.064 \\ & & 0.36 & 0 & 0.48 & 0.16 \\ & & & 0.6 & 0.4 & 0 \\ & & & & 1 & 0 \\ 0 & & & & & 1 \end{bmatrix}$$

6.3 Appendix C

Given a generic convex shape as floorplan, we can compute its area (A); if we imagine placing a point mass transmitter (`trx`) in a random point of such floorplan, it is easy to compute the probability for `trx` to be located in a given point x as:

$$P(x) = 1/A$$

Given that the transmission radius in r, we can define a circle centered in x, as the transmission range of the first `trx` (R). If the `trx` in far enough from the border of the floorplan, all it's transmission range fall into the floorplan; in this case, the probability for another (random positioned) `trx` to be placed inside this transmission range is equal to:

$$P(x') = \int_R P(x) = \frac{2\pi r^2}{A}$$

If the first `trx` device is placed at a distance less than r from one or more borders of the floorplan, the probability of randomly placing another `trx` device in its range is less than in the previous case; this is because part of the area of the transmission range will fall out of the floorplan and, by construction, `trx` devices can't be placed there. Depending on the floorplan' shape, we can identify the partition of the shape where the transmission range will fall out of the shape itself, and we will do specific computation for this area.
In general, by the middle value theorem for integrals, and the total probability theorem, the probailiy for 2 transmitters (let x and y be their coordinates) randomly placed to be close enough to be able to communicate is:

$$\rho = P(|x - y| < r) = \frac{1}{A} \int_A P(x')$$

Since we know that $P(x')$ is not constant for all the area of the shape, this integral might be difficult to compute, even for a square floorplan; in any case its value, it's constant for a given floorplan and a given r. Taking into consideration only one `trx`, we can compute the probability density function for the number of `trx` in its transmission range, in function of the number of `trx` in the entire floorplan $C(N)$. Every time we put a `trx` in the floorplan, the probability that it falls in the transmission range of the highlighted one is ρ. This is true for every $N > 1$, because of independence.

$$C(N) = \left(\frac{1}{p}\right)^{(N-1)}$$

This is a Bernoullian distribution, and we can use this result only taking one `trx` at time; in other words we cannot use this distribution for all the `trx` in a random configured floorplan, because they are not independent. The distribution for one `trx` will be influenced by the position of others. Anyway, we can apply this distribution to many transmitters, each one taken from a different floorplan; in this way we can validate the independence of different random configurations.

6.4 Appendix D

Below is a demonstration of the upper bound for the number of collision in a scenario with N hosts.

The necessary condition for a collision to happen is that at least two hosts transmit during the same slot. Each host placed in the intersection of the transmission circles of the two transmitting hosts detects one collision.
Therefore, to maximize the number of collisions, it is necessary in every slot, both to a) maximize the number of hosts that detect the collision and b) minimize the number of transmitting hosts (while still assuring that a collision can happen) so that during the next slot the number of hosts available to detect a collision is the highest.

Let us suppose to have a floorplan with 100 hosts.

- During slot n. 1, host 0 transmits and, being the only host transmitting, absence of collision is ensured. Let us suppose that the message is received by host 99 and host 98.
- During slot n. 2, host 99 and host 99 broadcast the message and, out of the 97 active hosts, only host 97 and host 96 receive the message correctly. The remaining 95 detect a collision.
- During slot n. 3, host 97 and host 96 broadcast the message and, out of the 95 active hosts, only host 95 and host 95 receive the message correctly. The remaining 93 detect a collision.
- And so on...

Following this line of reasoning, the number of listening hosts decreases by two every slot, and so does the number of collisions.
When only 5 hosts are remaining on the floorplan, 2 of which are transmitting and the other 3 are receiving, 2 of the latter 3 receive the message correctly and the last one detects a collision.
Finally, the last 2 out of 3 broadcast the message and the last one detects a collision again.
The total number of collision is therefore $95 + 93 + 91... + 3 + 1\mathbf{+1} = 2305$.

It is trivial to prove that that if $N \leq 3$, no collisions can happen.
In general, for a configuration with N hosts, $N \geq 4$, the upper bound for the number of collision is given by the following formula:

$$C_{max}(N) = \begin{cases} 1 + (\frac{N}{2} - 2)^2 & \text{if N is odd} \\ 2 + (\frac{N-1}{2} - 2)(\frac{N-1}{2} - 1) & \text{if N is even} \end{cases} \tag{6.1}$$