Distributed Systems 2 Project
A distributed sensing network using a broadcast only communication model

Girardi Massimiliano 211999

The objective of the following paper is to introduce an implementation of the protocol described in the Paper by Christian F. Tshudin[1]. The discussion will analyze most aspects of the implementation, including extendability and general usability of the provided simulator, that is based on the RePast framework.

Introduction

This document will not introduce most concepts already extensively described in the original paper, but it will give them as a given in order to expand and explain the specific scenario that was imagined when designing this product, its caveats, and its good properties. The project provided along with this document is an easily extensible tool that is designed to test, for now, a parametric scenario that involves a variable network that has the capability of sensing a medium and transmitting into it. Each of the following chapters will be subdivided into a theoretical description of each of the main components and a subchapter that describes the practical implementation of the concept at hand.

Architecture

Here the architecture of the project is explained, giving each fundamental component a subchapter of its own. The structure can be briefly explained as follows, refer to the specific chapters for more details:

Component	Description
Relay	A relay is one of the components in charge of sending and receiving the packets
Medium	The medium mediates in an opaque way the communication
Map generator	Component in charge of generating the geographical disposition of the relays

The medium

We call medium anything that can propagate a signal by any means. A medium can not be assumed as a reliable mediator for communication nor it is a privacy preserving channel. No assumptions are made in this protocol about it, and so it was intentionally vaguely described as a black box that has an input channel and an output channel. Another important thing to note is that the propagation time and range inside the medium are unknown to the user, and so is the reliability of the channel.

In the simulator provided along with this document it was decided, to facilitate the discussion, stripping it of some useless implementation details, that if a message gets delivered, it is also correct (the content of a message is unchanged after transmission). This is a reasonable assumption, given that it is easy enough to implement a checksum of some sort on the messages. The medium is represented in two views in the simulator, both are geographically accurate concerning the position of the relays. One is is to make the positions clear, and it is useful to consult when the other view, that is called the `FrenView` is clustered with the messages getting exchanged. The `FrenView` displays all of the messages that are getting received in any given moment, the simulator draws each message with a colored arrow, each recipient gets its own color, generated from a pallet so that perturbations are easy to distinguishable and trackable. The positions of the relays are bound to a grid but this serves no purpose other than being esthetically pleasing.

The relay

A relay is an independent host that wants to participate in the network with its purpose in mind. A host is part of the network if it was discovered by at least one peer, and therefore if it was at least once able to send a message in the medium that was correctly sensed by someone else. A host is independent of all the other hosts, and as such, it can have different performance, active time, and objectives.

In the simulator provided along with this document a superclass Relay, with no purpose, is available along with some example implementations like the one that is used to measure the system performance.

The implementation that is provided along with this product was used to get the data that will be presented in the following chapters, this specific implementation beacons the network to join it, getting discovered by the neighbors, and senses the carrier for any message that can come it's way. Thanks to the ARQ mechanism implemented in the superclass, it will be recovered up to speed with its peers in little time after joining.

Map generation

As will be discussed further in the data analysis chapter, the map generation does influence the protocol performance significantly, and this is not really surprising in a way. The main issue is that, given the limited transmission capacity of all the actors involved and the nonreliable nature of the communication, the possibility to traverse geographically the network (and for that matter logically, given any non-trivial logical topology) is strongly influenced by the actor disposition. This is because, given any two nodes A and B, for A to be able to communicate with B it is either true that:

- B is in range of A. notationally: `(B ir A)`
- There is an intermediate node between A and B such that `(B ir C ir A)`
- There is a path § of nodes such that `(B ir P[start]...P[end] ir A)`

Critically this does not account for the non-reliable nature of the network that adds an additional layer of difficulty, mainly the fact that `(B ir A) =/> "B can communicate with A"`, although this assumption is, in the product at hand, true in the long run (for any noise that determines an error rate that is less than 100%).

To have a baseline non-trivial answer to most evaluation metrics, the position of each peer has to be generated so that the resulting physical network is even intermittently connected. The source of intermittence can be the temporary failure of nodes or transmission clashing. This leaves still many possible parameters to adjust in the generation process, but the basic principle is that the resulting network shall have at least one path that connects any node to any other node. This constraint is strong, and not really realistic for the purpose of checking the actual world performance of the protocol, but the addition of intermittence in the connection makes it way more acceptable, so it was decided to place it into action. What this practically means is that each node will be placed in range of at least another node in the network. This does not mean that the network will start or will remain connected for the full duration of the simulation, but that it has the capability to be so.

The simulator has a map generator that is configurable with several network layout directly from the Simulator GUI.

- **Uniform:** All node positions are equally probable, therefore the distribution of the hosts is uniform in the space
- **Clustered:** The nodes are clustered around `cluster count` positions, in a range as big as the `relay range`

Data analysis

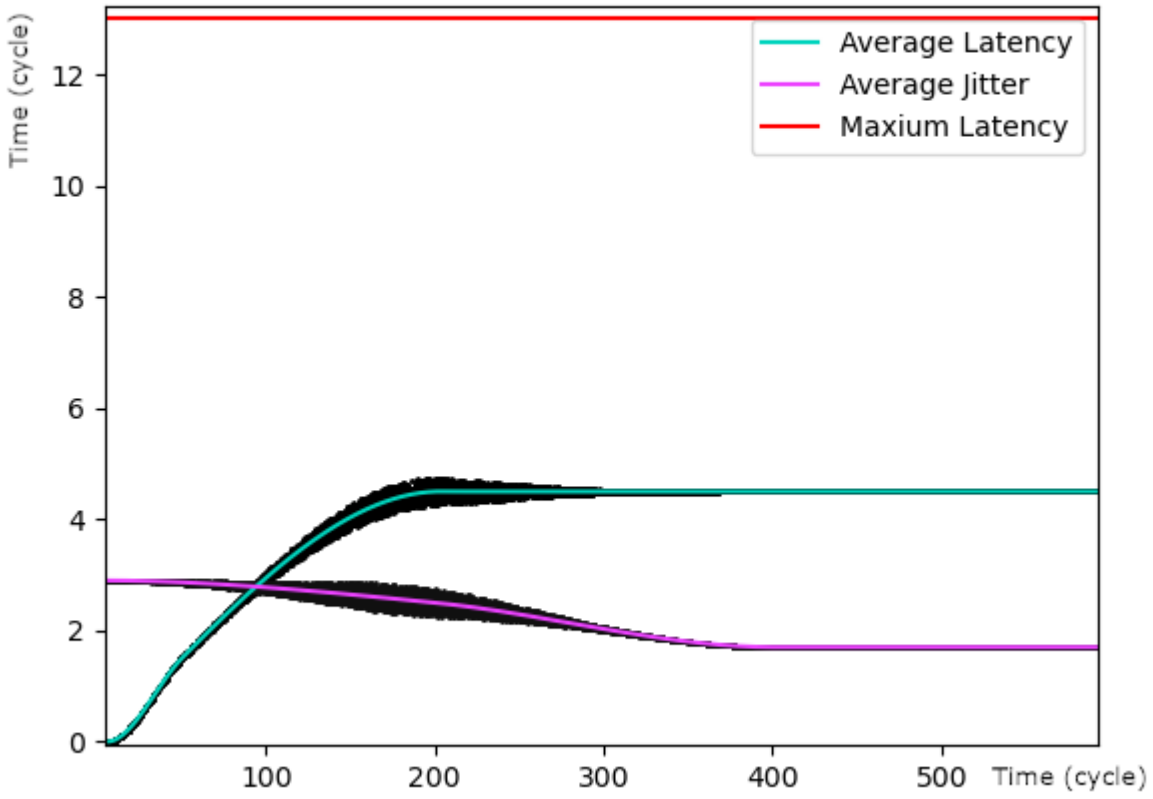
In the following, an analysis of the experiment is presented. The data presented in the graphs are calculated at runtime by the simulator provided and, for the purpose of accuracy, it was saved and re-processed to lower the possibility of initialization bias. What this means is that, although the graphs presented in the present do look similar to the one that the simulator generates at runtime, they are way more statistically significant because they do not refer to any specific initialization seed, but are averaged across multiple of them. This does not mean that execution of the protocol in the simulator does not give a good idea of the evolution of execution with a certain set of parameters but that to have a precise overview of the results caution should be used while reading the runtime graphs. The graphs are also truncated of the initialization part, this is because it is not really significant.

The discussion will be subdivided into two main parts, firstly the protocol was tested in an ideal scenario, where the packet loss is none, to get a baseline measurement or an ideal practical

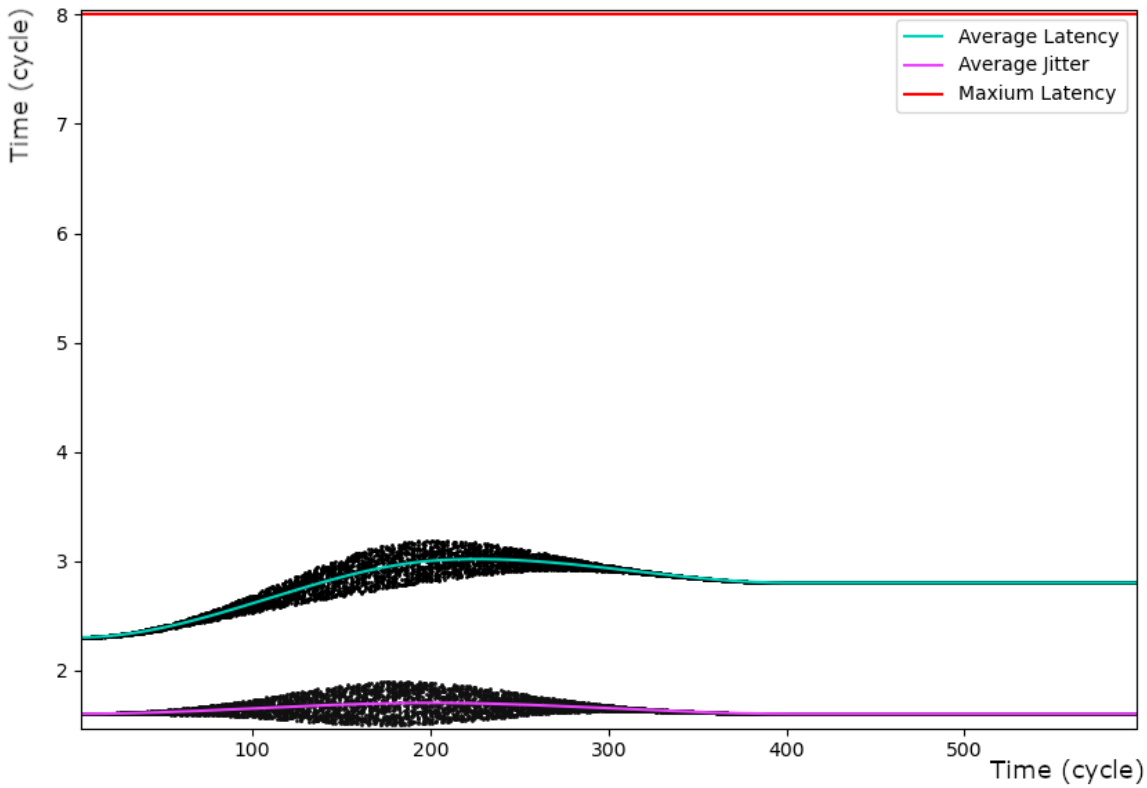
performance of it. Finally a more realistic scenario, that takes under consideration all variables were tested.

The following graphs are extracted from the simulations and will be presented in the data discussion further below:

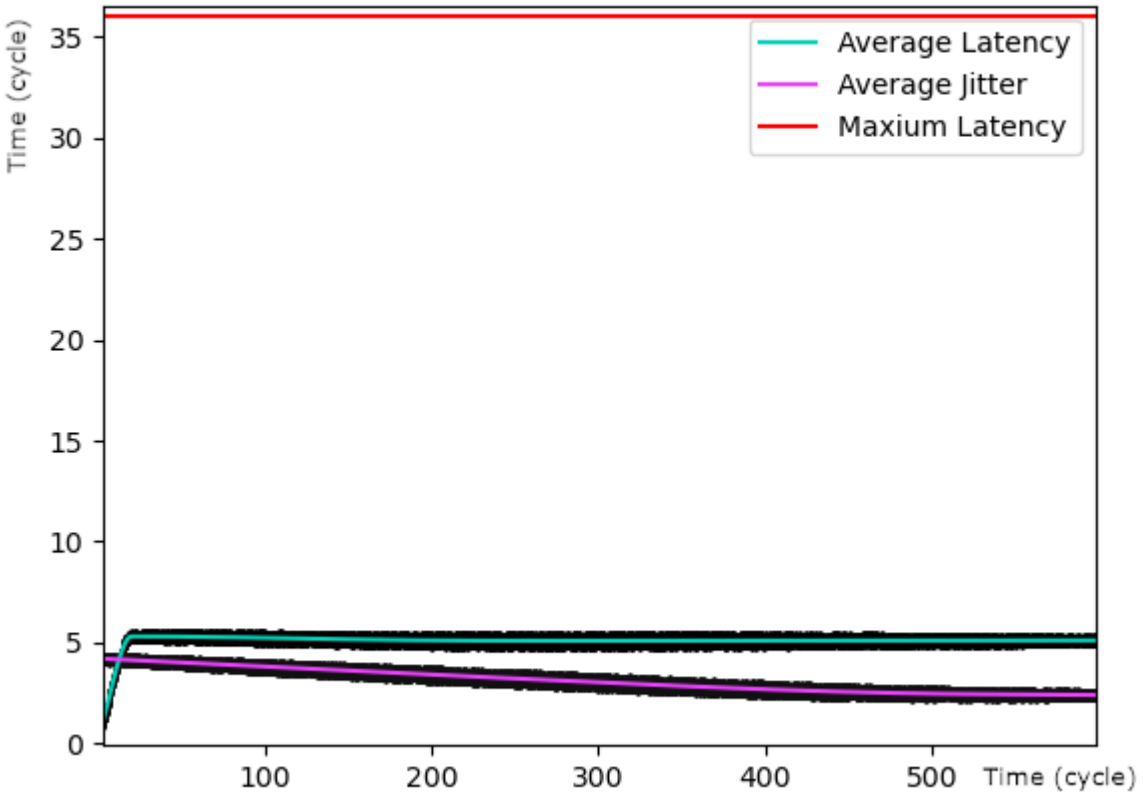
Latency and Jitter, 15 simulations, Ideal network (0% packetloss), default parameters (uniform map)



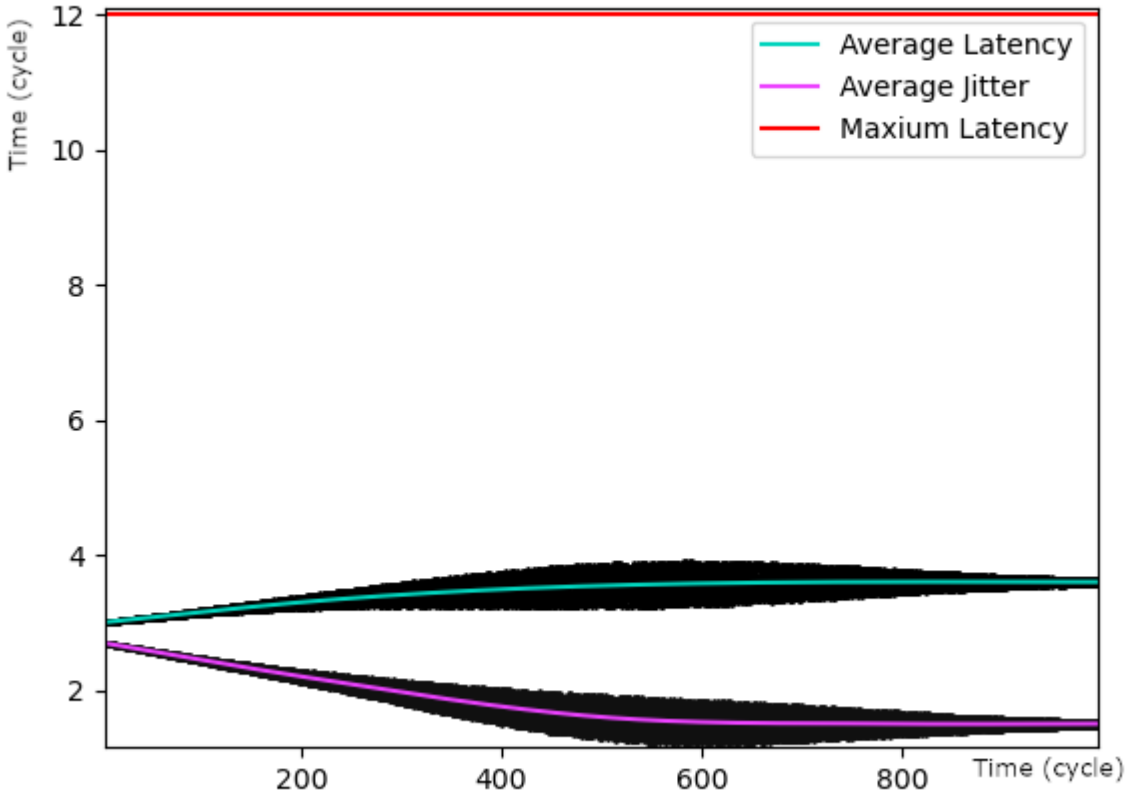
Latency and Jitter, 15 simulations, Ideal network (0% packetloss), default parameters (clustered map)



Latency and Jitter, 15 simulations, Noisy network (5% packetloss), default parameters (uniform map)



Latency and Jitter, 15 simulations, Noisy network (5% packetloss), default parameters (clustered map)



All the graphs present the aggregated (averaged) data of a simulation (the gray area) and an aggregated estimator (average) of an expected value for each measure. As an added bonus, the maximum latency was added, to give a worst-case scenario, that is important for the application at hand.

Data discussion

In the following, the results for each metric are presented and commented on. One thing that is important to underline is that there is no silver bullet metric in this paper and the discussion will keep this into account, although it is true that the average latency, for example, is important for some applications, for many others, like distributed sensing networks, it can be almost irrelevant as long as it is bounded. For example, there is no need to have a 2ms latency on a pollution sensing network of a city, a one-day latency is more than enough for the purpose. With that being said, although further some considerations will be made on some specific application of this protocol, in the following each metric will be treated as the silver bullet in its own chapter.

Average latency and jitter

The average latency is the average time that a packet takes to get from any host in the network to any other host. Firstly we took into consideration a scenario where all the hosts are scattered uniformly in the field. The expectancy is that in this situation the average latency is the maximum value for any casual communication pattern, where all the hosts have the same probability to communicate with any other neighbor.

Given a square field of side(s) and a number(n) of hosts with range(r) the question is, how much is the average distance in hops between hosts? Remembering that the distance of two hosts is calculated as:

$$Dist((x_1, y_1), (x_2, y_2)) := \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Taking advantage of the fact that the hosts are disposed on a discrete grid and that the space is square, we the size of a subdivision of the space as:

$$SubSize(space_s, range) = \frac{space_s}{range}$$

Therefore, given `sub_space = SubSize(s, r)` we can assume that the average number of hops from each node to each other is:

$$N\_hops = \sum_{x=1}^{sub\_space} \frac{Dist((0,0),(x,x))}{sub\_space}$$

This metric is enough to calculate the average latency that is simply:

$$AvgLatency = N\_hops * (Transmission_t + Reaction_t)$$

Where the transmission time is the time that a message takes (on average) to go from one host to another, and the reaction time is the average time a host takes to react to a message received.

When the hosts are not uniformly distributed in the space, the average distance among them is bound, by the construction of the metric, to get lower (intuitively, once the nodes are closer between each other, there are fewer subs, given that the range is the same and the space occupied in total by the nodes is lower).

The experiment going further expected that the uniformly distributed network was almost a worst-case scenario for the latency and this was proved to be true with some interesting edge cases. What the simulator underlined very clearly is that clustering together multiple nodes does reduce the average latency in the ideal case, as expected, and this remains true for the more realistic scenario where messages may be lost. This is not surprising, although it has to be underlined that the data that was used to plot the graphs that are now going to be presented was generated using some network configurations that do not highlight the weaknesses of the clustered method. Given that the hosts are more concentrated in some zones of the map, it is way more likely for this approach to create a network that does not let all the hosts communicate with all the others.

Another interesting finding of this analysis, that seems to present itself in a wide variety of configurations, here the graphs of two separate “extreme” cases were presented, is that although all the configurations of similar geographies have a different initialization time and behavior, they all seem to converge to a similar average value for both metrics, and this validates the theory presented above.

Specific application

In this simulator, the protocol was imagined to be used in a distributed sensing network, where hosts are not constantly sensing the carrier to receive new packets to save energy. For a situation like this, the average latency and jitter can be important, if the sensing is done to keep under control an environment. The most important thing is that both these measures are bounded and low enough for the application at hand. Of course, and intentionally, the simulator does not by itself provide a unit of measure for these sizes. What is most important is the relationship that they have. To study a specific application case a user can vary the sizes that are given, keeping in mind only one limitation, that can be changed but has yet to be implemented in the GUI, the propagation time of a message in the range of a relay is always one cycle and the reaction time of the relay to it is similarly one cycle. This can be changed in code, and will, in a future version, be changeable from the simulator GUI. This is though a non-limitation for most cases, given that in the scenario presented the signal of the radio is really fast, therefore the travel time is negligible. The period one was chosen to enforce a follow by relation between the send action and the receive action, this makes also the graphs more pleasant to watch given that this was one of the secondary objectives.

Installation

In the directory, there is an installer.jar file that can be used to install the simulator, execute it, and follow the instructions.
The installation process will create a directory that will contain, among others, two executable script files that can be used to launch the project.

> Do not make the tick period too low if you plan to execute without a `stop at` option set. There is a bug that makes the GUI freeze and if you do!

Further work

As it can be seen, there are attributes of the project that were not tackled in this paper, and features of the simulator that were not expanded or did not find a proper visualization on the GUI. This is a release of the simulator that is by no means final, there will be updates to expand the GUI and the library of example relay implementation in general.

Conclusions

The protocol seems to behave fairly well in situations where the reliability of the network is low or very low, this is a good news for any application that is not too sensitive to jitter or can withstand high, although converging, average latency. The implementation of the protocol using the RePast framework can be considered succesful, as the project behaves according to the source paper.

References

1 Christian F. Tschudin. A Broadcast-Only Communication Model Based on Append Only Logs