02223 Fundamentals of Modern Embedded Systems

Group 14

Luca Cambiaghi - s161162
Salik L. Pedersen - s134416
Alex Incerti - s172125

All members of the group contributed equally.

December 17, 2017

02223 Final Report

Luca Cambiaghi
s161162@student.dtu.dk

Salik Lennert Pedersen
s134416@student.dtu.dk

Alex Incerti
s172125@student.dtu.dk

1. INTRODUCTION

Our project involves the design of a distributed embedded system; the area of interest is Internet of Things. Our *complete* system comprises two principal entities: a smart `Bracelet` featuring a wireless transceiver and a router encapsulated in a small tower offering a touch screen, informally called `Totem`. The context in which we want to deploy our system is crowded events, for example concerts or festivals attended by thousands of people: a good example is Roskilde Festival, a rock festival attracting more than 100.000 people each year over a 7 days time span. The problem we are targeting in this domain is reuniting two people that have been separated: in fact, in these settings it is not always easy to use the phone because of noise, cellular coverage and battery.

We want to equip each of the guest of the event with a smart bracelet and scatter around the area a number of totems they can interact with. Each guest would have his bracelet associated to his friends' at the beginning of the festival. In case a guest wanted to locate a friend, he would swipe his bracelet on a totem and select the desired name among the available ones. The bracelet would then start guiding the lost guest to his friend through a visual interface, for example a LED "screen". The location of each bracelet is calculated by the peer-to-peer network of bracelets, periodically communicating with the bracelets in its range. The system as a whole would therefore have an overview of all the locations.

Since the system featuring the `Totem` would have not been interesting from an energy standpoint, due to the advantages offered by the hybrid architecture client-server and p2p, we decided to focus on an harder version of the problem. Our system comprises of the only `Bracelet`, making the system a pure distributed peer-to-peer system consisting of a unique mesh network.

2. APPROACH

Our objective in this project has been to derive the design of our embedded system through simulation. This has allowed us to understand how the underlying physical components, network protocols and scheduling algorithms affect the performance and quality of service.

Our first step has been to define the scope of our analysis. As not every part of the system was interesting to study in the context of this course and in a limited time frame. Also, not every technology was available to employ because of the nature of the device. In fact, our initial goal was to design a cheap, simple and functional device. This meant coming to compromises such as not including a GPS module or an e-ink screen, because they would have risen up considerably the price and in the case of GPS also the power consumption. Price is also the reason why we have decided to limit our battery capacity to that of a CR2032 coin cell battery(240mAh) [3] however with this amount of battery the bracelet should be able to operate at least 10 days. With these constraints in mind, we focused on modelling the energy consumption of our simplified system and on finding a good balance between battery life and quality of service.

Our simplification was defined by the assumptions we decided to make. As mentioned before, we decided to design a fully decentralized system where totems do not participate in the storage of data or in the routing of messages. Therefore, we can see the totems simply as locations where people start to search. Our first set of assumptions regard this interaction between bracelet and totem, in particular:

1. The algorithm describing the interaction between bracelet and totem has not been implemented. Instead of choosing an ID from the totem's screen, the guest can just start to search whom he wants. The next assumption directly follows.
2. The enrollment of groups of bracelets has not been investigated. The guest can search for any other guest, he is not limited to his group of friends.
3. The guest does not need to reach a totem's location to start the search, he starts it from wherever he is in that instant.

Then, a second set of assumptions had to be made regarding the initial knowledge of the bracelet and the behaviour of its radio, such as:

4. The bracelet knows its own position, even without a GPS module. This is justifiable by the fact that in the real system totems placed in fixed location can be used to triangulate the position.
5. When a bracelet broadcasts a message, it will deliver

it to all bracelets in its range, meaning interference has not been modelled and failed messages are not re-broadcasted.

Finally, an important part of our simulation involves people moving around the area of the festival and gathering in some areas for concerts. Some assumptions had to be made in this regard, like:

6. The movement of people has not been thoroughly modelled. Instead, a simple random walk algorithm has been implemented, such that people move in a random direction and when they reach the boundary of the event they "bounce" in the opposite vectorial direction. We assume that guests that search for a friend walk slightly faster in order to catch up.
7. The geography of the festival has not been analyzed and represented. Instead, a square of fixed size is used as setting of the event. No corridors or obstacles have been placed in the square.

Once our scope and constraints were clearly defined, we finished our research of the current state of the art in the field of wireless peer to peer communication.

3. METHODS

As mentioned in the introduction, our system is defined by a "mesh" network of devices. Each node of the network therefore needs to broadcast and relay messages for the information to propagate in the geographical area of the event. The design of such a network consisted in the definition of the exchanged messages and the scheduling of communication phases.

In order to see if it was possible for such a system to work we started off with two assumptions: infinite battery and infinite range. We designed and implemented a simulator where we could test various setups. Once simulations showed that it was possible we restricted the system, in different ways. By researching different low power components we were able to find a realistic set of parameters for the simulator.

The modelling began with drawing the state diagram (figure 1), refined in iterations as our model gained complexity. During this phase, we defined two different types of communication paradigms in our network: a periodic broadcast and a search-reply interaction.

The periodic message includes information about the sender's location and the locations he recently received from the nearby nodes; the search request message contains the ID of the friend that is being looked for; the search response message contains the location of the friend that is being looked for.

The periodic message is necessary for the service of guiding a guest to a friend's location: with the search-reply message alone each node would only be aware of its location so only the searched friend could reply to the search request. On the other hand, the search-request is not necessary for providing the service: in fact, with a frequent enough flooding by the network, the lost guest could just find the information in his bracelet's database.

This brought us to define two different communication models: the first one featuring only frequent periodic updates and the second with less frequent periodic updates and on-demand search requests (a request is sent when the information is not present in the user's database). We implemented the first model and then expanded it with the search-request variant. In Figure 1 you can see the full state diagram of the system, where the blue edges indicate the transition that only happen in the search-request(SR) model. Referring to the diagram, let's first consider the periodic update model. A bracelet that is not searching for anyone will simply transition from Sleep to Communication Phase to Update Phase, according to the variable timers. During the Communication Phase, each node of the network will broadcast its recent locations and relay the messages received from the other nodes.

When a guest starts a search, its bracelet will transition to the Search Phase. If the location is present in the database it will transition to the LED Phase and start guiding towards that location. If it is not present, the search is failed. A searching bracelet also participates in the Communication Phase.

In the request-response model things differ only slightly. A non-searching bracelet will also transition from Sleep to Listen Phase, where it will relay any received Search Requests. In the following phase, it will search its database for the locations of the requested bracelets. If it finds any, it will transition to the Response Phase. Eventually, it will transition in the Update Phase to update its database with the locations it has processed in other Search Responses, before going to Sleep.

As for the searching bracelet, if a location is not found in the SLookup Phase, it will transition in the Request Phase, where a Search Request is broadcasted and other requests and responses are collected and relayed. The Responses are processed in the following phase and if the location has been received, the bracelet can change to the LED phase and start guiding.

After having understood the behaviour of the system with respect to time, we continued modelling the electrical components, the messages exchanged and the movement of people. In Figure 2 you can see the class diagram of the bracelet entity.

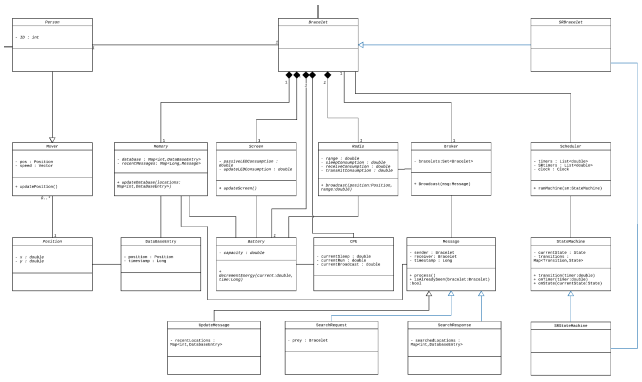

Figure 2: Class diagram - Bracelet

On the left we can see the classes related to the movement of people, in the center the electrical components and on the right those that deal with communication and scheduling. As stated before, the movement of the guests has not been modelled but instead a RandomWalk algorithm implemented. This means that people start in random positions

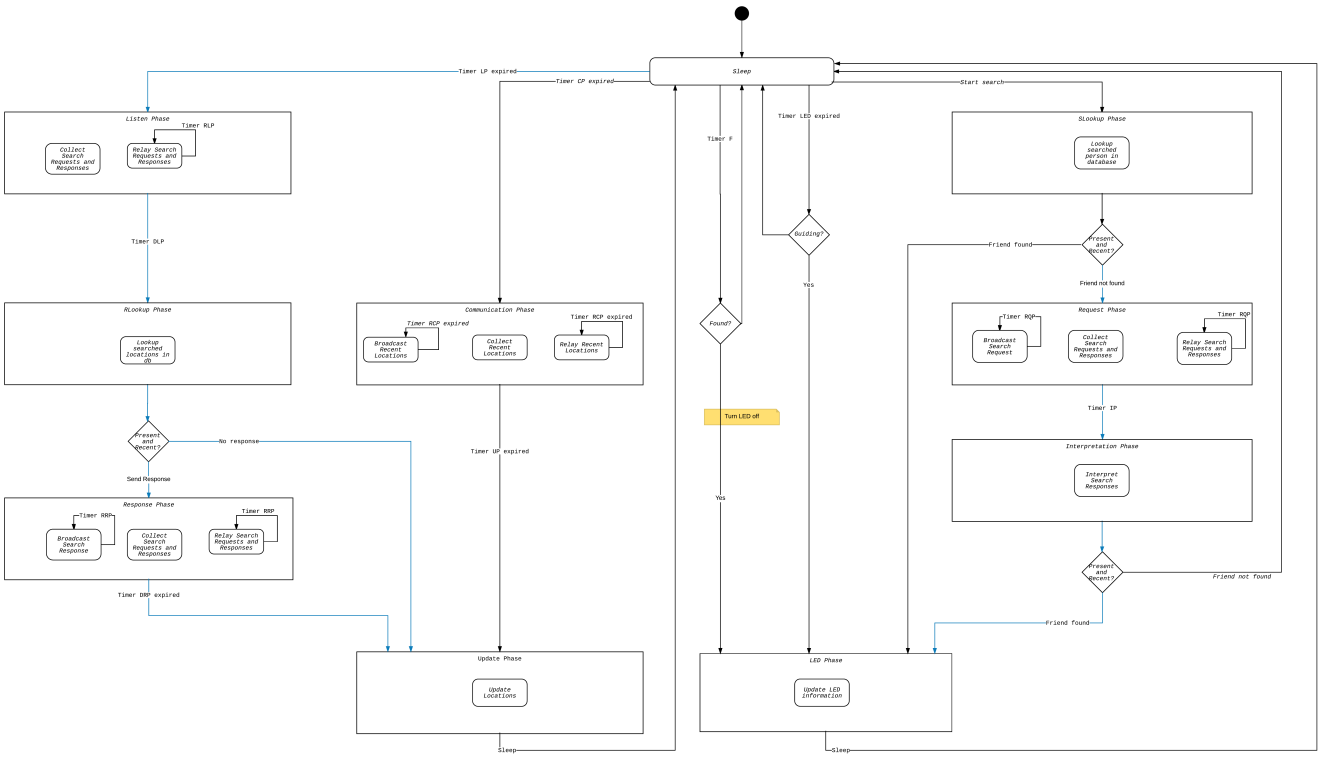

Figure 1: State Diagram

on the map and walk in a random direction on the 2D plane. If they hit a wall they will bounce off the wall at the same angle as they hit it. The components classes act as stubs to decrement energy on the battery, according to the parameters stored in their fields.

The broker class handles exchange of messages between the bracelets. Messages get delivered only to bracelets in range of the sender and only if the message has not been received before already. The state machine class controls how the bracelet transition between the phases designed in the state diagram. We have defined an enum class called State for each phase of the process and we have defined a Transition class which links together two states. Then, in the StateMachine class, we put in a list all the possible transitions so to make sure the bracelet respects the expected behaviour and does not, for example, transition from Communication to Response Phase. This class also contains the logic for each different phase, it is here that we for example broadcast periodic messages in the Communication Phase or update the database in the Update Phase. The Scheduler class stores all the timers that controls the transitions and is in charge to make the transition happen when the simulation clock satisfies the timers' conditions.

This framework has made it easy to expand the periodic update model to the search-request: it was just necessary to have a SRStateMachine class extend the standard StateMachine class. We added to the list of possible transitions the ones related to the additional phases and implemented the methods for the additional states like Listen Phase, Request Phase and Response phase. All the logic for the remaining states is present in the superclass. Then, by having a SRBracelet class extend the standard Bracelet class, we could selectively instantiate different simulations with either of the two different kinds of bracelet, to evaluate the two different models.

The next step was to model the Simulator and design how to extract results from different Simulations. In Figure 3 below you can see the class diagram for the Simulator context. The simulator can load many Simulations at a time. Events and Parameters get parsed from a text file and get injected in the simulation. The Parameters class is used to set the relevant timers and thresholds that drive the results. The Simulation uses a Clock class to coherently administer time of Events and the Bracelet's internal clock. The abstract class Event superclasses the three different kind of events, which have a different effect on the Simulation they belong to. ConcertEvent makes it so that the guests start moving towards the concert's location; ArrivalEvent adds newly arrived guests to the attendees of the simulation; SearchEvent signals the a certain user wants to search for the chosen friend. Events and simulation have been designed to give us a granular control over which cases we want the system to simulate.

In order to get a better understanding of the overall behavior of the system under simulations, a web user-interface has been implemented to show the guests movement on the map and their interactions. The real data about people density along with area and number of guests from Roskilde Festival have been analyzed as a real case scenario for our bracelet. Simulations have been run in a scaled down version of these data, to keep the ratio but allowing us to run them on a relatively smaller area, with lower number of participants

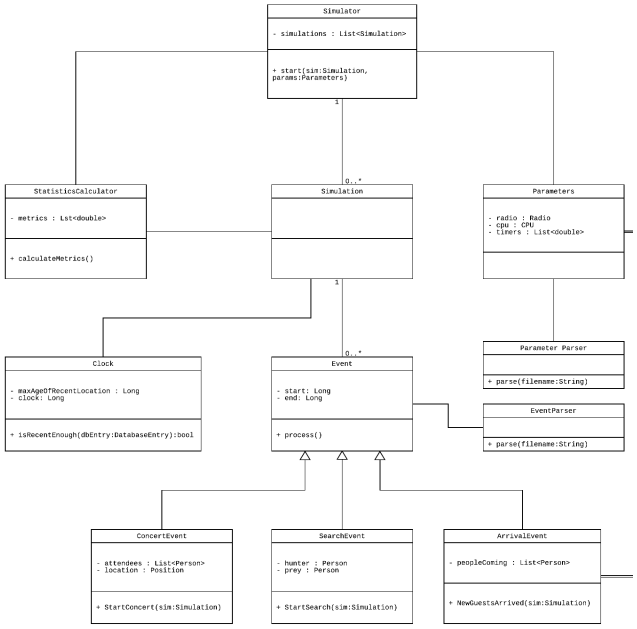

Figure 3: Class diagram - Simulation

and shorter amount of time while keeping the realistic proportions.
We introduced an artificial unit of time called *clock* in order to being able to compare results from different simulations, as using the real time timestamp was suffering from fluctuation from simulation to simulation as every iteration of the system could take slightly longer in the different simulations. The reader can roughly think of the *clock* unity of measurement as millisecond. In order to simulate bracelets being started at different times we decided to offset the simulatorclock in each bracelet by the time people enter the venue. In this way only people being synchronized in the different phases would be able to communicate during the event. We therefore decided to introduce a slight random clockdrift in each bracelet giving asynchronous behaviour of the bracelets.
We developed a metrics collector, to run with the simulations in order to calculate metrics about the behavior of the system. These metrics tells us specific details and allow us to compare the different configurations and models with each other. For some of these metrics, maximum and minimum are calculated over the simulation, to analyze the variance in the system. Some of the main metrics calculated are:

- **Average time to find a friend** The time, on average, it took to complete successfully a search request. Failed searches are not taken into account in this metric.
- **Average age of location in database** Over the whole simulation, the age of the locations present in all bracelets is calculated and summed to give a total average. (Intermediate values are also stored to show the trend over time)
- **Percentage of guests present in the database** How many guests' locations, out of all those present at the event, are in the database of each guest is calculated over the whole simulation and summed to give a total average. (Intermediate values are also stored to show the trend over time).
- **Percentage of recent locations in database** How many of the locations in the database are recent (it depends on the age-threshold).
- **System coverage** How many of the guests are out of range at a specific moment. Being out of range has been defined as not being connected, directly or indirectly through others, to more than half of the guests at the festival.
- **Number of succeeded/failed/started but failed searches** Failed searches are those where the searcher does not receive a recent location of the target and does not start the search. The total number of these three types of outcome for a search are calculated for the whole simulation.

4. DISCUSSION

As mentioned in previous sections we are quite limited in the size of the battery as the bracelet has to be cheap while still providing enough power to last 10 days. Is it in any way possible for a bracelet to last the full 10 days of an event like Roskilde Festival? Using [4] we found some parameters for our model that suggested that using a coin-cell battery like the CR2032 with a capacity of $240mAh$ we could expect in the area of 530 days of broadcasting every $1000ms$ assuming the data-size of each broadcast was 31 Bytes which is the maximum size for user data in a single broadcast packet [6]. This is of course very optimistic as the amount of data transferred can be much larger. As the power consumption grows linear in the amount of data transferred as can be seen in figure 4 we were able to scale the power consumption to any amount of messages.
Depending on which of the communication paradigms is used we can either be broadcasting our own position as well as the most recent entries in the database at regular intervals. In the case of the Search-Request energy model we only broadcast when we want to find another person/bracelet and the data of this broadcast contains less data than the previous model, as we do not include all the recent entries from a database. The broadcast could contain sender and receiver ids which can be as little as two 48bit MAC addresses which gives 12 Bytes and can therefor fit in a single broadcast packet. Once the receiver gets the message it will respond with a position for the sender to see. A database entry should contain the following information: MAC address (6 Bytes), time-stamp (8 Bytes) and position consisting of two double precision floating point numbers (16 Bytes) which gives a total of 30 Bytes pr. entry. With a total of 130000 potential entries at a venue like Roskilde festival the total database would be $130000 * 30Bytes = 3900000Bytes = 3.72MiB$, so the total size of the database should easily fit in cheap flash memory.

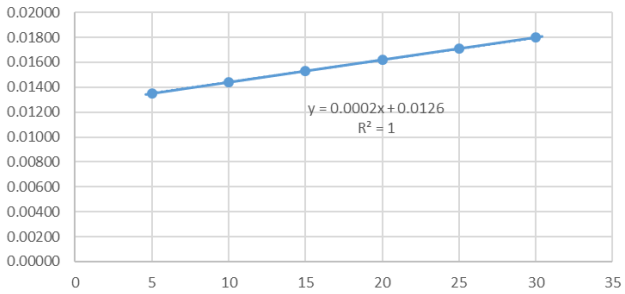

Figure 4: Average mA as a result of #Bytes pr. Broadcast packet every 1s

As we can only fit a single database entry in a single BLE broadcast packet, two options for transferring more than 31 Bytes were considered.
The first option was to let the bracelet connect to other bracelets thereby allowing for as much data as we want. However that would require every bracelet to connect to each individual bracelet in order to transfer data. Energy wise this would be troublesome as we would have to advertise that we want to connect via a broadcast packet and then use multiple additional packets. Furthermore it would be tedious to keep track of which bracelets have already exchanged data and which ones have not.
The second option that we chose was to send multiple broadcast packets each containing a single database entry. By limiting the number of packets in each broadcast session to 30 and broadcasting every 3 seconds we were able to get just around 18 days of battery-life from a CR2032 coin-cell battery. This was however in the base case with no searching sessions, only broadcasting and receiving updates from other bracelets. Once a bracelet starts searching and guiding the power consumption rises as the visual feedback has to be updated periodically. As can be seen in figure 5, where the battery has been scaled down by a factor 100 and the time spans 144 minutes (0.1 day), once the bracelet starts guiding there is a massive increase in consumption.

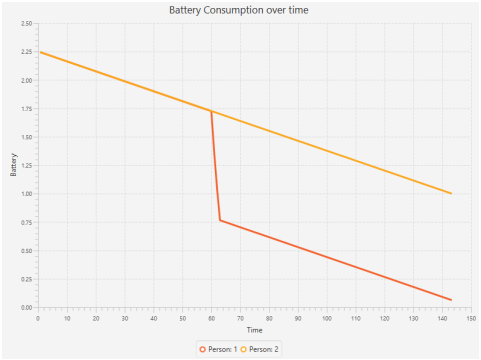

Figure 5: Power consumption when guiding using LEDs

This is with a single LED drawing 20mA being kept on while guiding. In this case the bracelet stops guiding after 4 minutes. Assuming 4 minutes to find a friend we would have enough battery to do a single search each day of a 10 day Roskilde festival. However in a real event covering upwards of 2500000m^2 [5], 4 minutes to find a friend seems very low. This is a highpower LED and using a low power LED of 1mA like [2] the runtime will increase, furthermore one could turn the led on and off making it blink at an interval. A simple 50% duty cycle could cut the consumption in half, however the frequency of the cycle is also a factor as a high a frequency would make the LED seem dim instead of blinking and therefore harder to see in direct sunlight. Another approach could be to use another technology like an E ink screen that only requires current when updating [1]. As can be seen in figure 6 the consumption is much lower for the 4 minutes guiding phase.

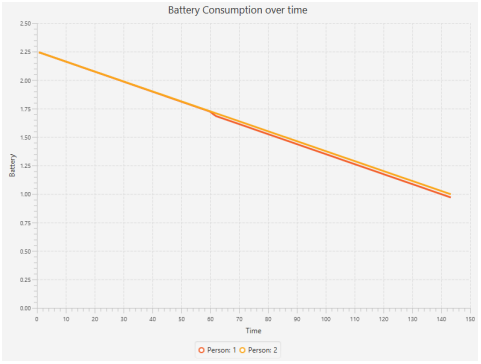

Figure 6: Power consumption when guiding using an E ink screen

This would allow for multiple searches per day while using a CR2032 coin-cell battery. The downside to using an E ink screen is the high cost and the fact that it is much more fragile than simple LEDs. As we aim for a cheap and robust bracelet we have chosen to use low power blinking LEDs. Using a CR2032 battery, in the event that a bracelet runs out of battery it can easily be replaced by a new one.

4.1 Simulations

The starting point for our simulations are a fixed number of events, generally mocking a normal day at a music festival.

- In the first part of the simulation, all the 30 guests arrive, divided in groups, each group at a different time. (Arrival)
- A first round of searches are initiated in the second part of the simulation along with two concerts attracting only a couple of guests (During the day, people search friends and small concerts take place)
- In the third part of the simulation two big concerts take place and attract almost all the guests. (Main concerts in the evening)
- In the last part, a new round of searches is initiated. (Reuniting with friends after the end of concerts)

Regarding the parameters changed from one simulation to the other, these are the common ones:

- The time threshold relative to which a position in the database is defined either recent or not. This affects the system since only recent positions are taken into account in the different phases.

- The range of the radio component in the bracelet.
- The scheduling for how often to communicate.

4.1.1 Broadcasting model

In broadcast model (B) bracelets periodically send the recent locations in their database and update their database according to the locations received, if more recent. In this model, another parameter is present and it is the frequency with which bracelets broadcast positions (only the recent ones).
As starting values for the three parameters, we decided to use optimistic ones. The range of the radio component has been set to 10m which is only 10% of the nominal maximum range of BLE, but more realistic. The frequency of broadcasting for 3000 clocks and sleeping for 1000 clocks. The age-threshold affects the age of the positions in the system: if a too old position is used in a search, the friend could have moved far away from that location and hence the position does not contain much information. It was decided to start not allowing people to move more than 1/3 of the width away from the position and that gave a threshold of 166000 clocks (given 1000px as width and $1px/500clock$ as speed). This simulation showed that with these parameters, 100% of all searches could be satisfied and only during concerts, the metric of people out of range and the percentage of recent locations in the database slightly dropped from 100% where it has been otherwise, stably.
It was decided to reduce the range to see how short it could be while still allow an acceptable quality of service. With the same parameters as before but with range shortened to 5 meters, only 55% of searches where satisfied. In the broadcasting model, if a search fails, it means there is no recent location of the searched friend in the database. Our first thought was to raise the age-threshold but that did not help. It turned out that at the time of the first round of searches, which were those failing, the average percentage of people in the database was very low. Postponing the first round of searches a bit later, gave the system time to distribute locations and when guests tried to initiate searches, they could find the needed information in their local database.
As it can be seen in Figure 8 (blue and red lines), the system takes longer, if the range is shorter, to reach a point where many people's locations are spread around. With this change, only 15% of searches failed, and they were all after the big concerts in the evening. This time the threshold was the problem. Guests had the location of the friend in the database (in the evening the system had reached a good spread level of information) but that location was discarded because too old.
Looking at the duration of the concert and the threshold, it turned out the threshold was lower than the duration of the concert and hence the position of friends at the other concert outdated. A possible solution in a real case scenario could be either to raise the threshold to be higher than a normal concert duration, let people wait after the concert to collect new and recent information or trigger a search request. We reduced the range further (down to 2 meter) and also the frequency of broadcasting (1000 clocks and sleep for 10000) and this slowed the system even more in distributing positions. All the searches failed. Making the concert last longer raised the percentage of completed searches to 20% of those after the concerts. This means that clusters of people staying close to each other for a long time can help in this case, but in a real case scenario these performances would not be acceptable.
The simulation showed that 2 meter as range and a broadcasting frequency quite low, can give improved results only if the density of people is high enough, which can be the case in some part of the festival and only in some specific occasions. The range of 2 meter could be realistic in a crowded scenario, due to high interference and it could maybe work thanks to the high density. Longer range can on the other hand be reached in less crowded and with less interference scenarios.

4.1.2 Search request model

The search request based model (SR) allows a bracelet to request the position of the searched friend if no recent location of her is found in the local database. In this way it should be possible to broadcast less often since in case of a search request, this triggers the system (hence, the other bracelets) to send the location of the requested guest, if they have a recent one. Moreover, it is hoped that this will allow to initiate a search earlier than in the broadcast model, where it turned out to be necessary to wait for the system to reach a good state of distribution of locations before being able to perform searches with high chance of succeeding.

Parameter	**Description**
Consumption parameters (fixed)	
currentSleep	Current consumed while in Sleep state
currentRun	Current consumed when updating the database
currentBroadcast	Current consumed while broadcasting
Scheduling parameters	
timerCP	Delay between Sleep and Communication Phase
timerUP	Delay between Communication and Update Phase
timerRCP	Delay of rebroadcast while in Communication Phase
timerLED	Delay between updating the LEDs while searching
Search-Request scheduling parameters	
timerIP	Delay between Request and Interpretation Phase
timerLP	Delay between Sleep and Listen Phase
timerDLP	Delay between Listen and RLookup Phase
timerDRP	Delay between Response and Update Phase
timerRQP	Delay of rebroadcast while in Request Phase
timerRRP	Delay of rebroadcast while in Response Phase
timerRLP	Delay of rebroadcast while in Listen Phase
Other parameters	
recentThreshold	Clock cycles under which a location is recent
radioRange	Range of the transceiver
LEDOnOffRatio	Frequency arrow is shown on LED

Table 1: Description of simulation parameters

All parameters of the B model (range, age-threshold), expect the frequency of broadcasting, are present in this model as well. The new parameters can be seen in table 1 but the most important ones are the frequency of listening for search requests (and processing them) and the time span a bracelet waits for search responses containing the friend's location, after sending a search request (search-response-waiting-time).
The starting value for the range was 5 meters and an age-threshold longer than concerts' duration. The frequency of listening phase was 2000 clocks every 1000 clocks and a search-response waiting time of 10000 clocks.
This simulation showed that without the B part of the model, 50% of searches could be satisfied (as shown in Figure 7, part of which were the searches performed early in the day (that failed in some B simulations), as expected. The average percentage of people in the database showed to be very low: 26%. With very low distributions of information, the triggering of search requests allowed to complete some searches.
The range was shortened to 2 meters but the listening phase was made more frequent (3000 clocks then sleeping 500 clocks). This improved both the percentage of successful requests and the average of people in the database (32%) as more search requests get replied, more positions get sent into the system.
Unfortunately also the percentage of searches started and then failed increased compared to the B model. This is due to the fact that once the search is initiated, it is expected to receive updates with newer locations, from people met on the way to the friend. This does not happen, if there is no broadcasting or if no one requests her position again. The simulation showed that it is needed to re-send a search request even when the search has started, if no new location is received on the way.

4.1.3 *Hybrid model*

The hybrid model (B & SR) uses both the broadcasting and the search request model in conjunction. The simulation has been performed with a low range, 2 meters. The broadcasting frequency picked was quite low (1000 clocks every 10000 clocks), as the goal of SR is to allow more rare broadcasts. The listening frequency was reduced as well to a more realistic 3000 clocks followed by sleeping for 1000 clocks. The threshold was kept higher than concerts' duration. The percentage of succeeded searches was 90% and a reduced number of started but failed searches (10%) thanks to introduction of B.
The system reached a good distribution of locations, with 88% of all guests in database, on average. Only 20% of searches required a search request, though. This was due to the high age-threshold and the good distribution of guests' locations in others' database. Since that value was introduced to solve the problem of guests trying to find friends after the concert, it can be removed as now bracelets can use SR to solve this problem. In the new simulation, with lower threshold, 50% of searches required to send a search request, 80% of all searches were successful. Even in this hybrid version, some searches fail.

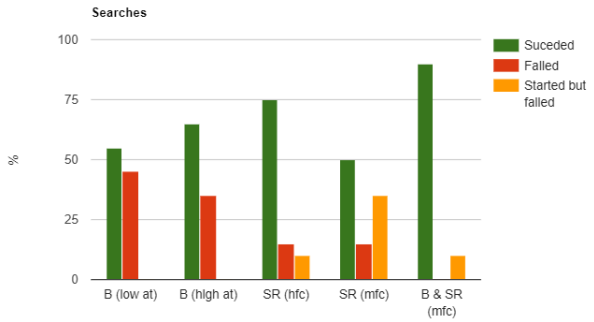

Figure 7: Results of searches over different simulations [at = age-threshold. hfc/mfc/lfc = high/medium/low frequency of communication]

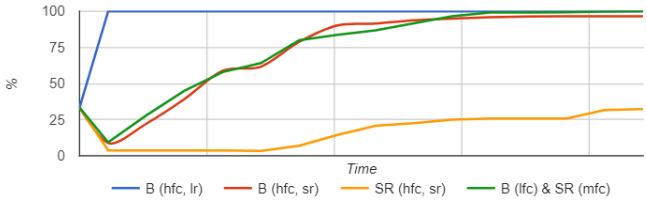

Figure 8: Guests in database on average [hfc/mfc/lfc = high/medium/low frequency of communication. lr/sr = long/short range]

4.1.4 *Results*

The simulations allowed us to detect some particular behaviors and characteristics of the system under certain conditions. The system configurations that provides the fastest search time are unsurprisingly the B and SR with high frequency of communication as shown in Figure 9. The hybrid and more realistic model provides a quite short search time and combined with the highest percentage of successful searches (Figure 7) indicates this as a valid option.

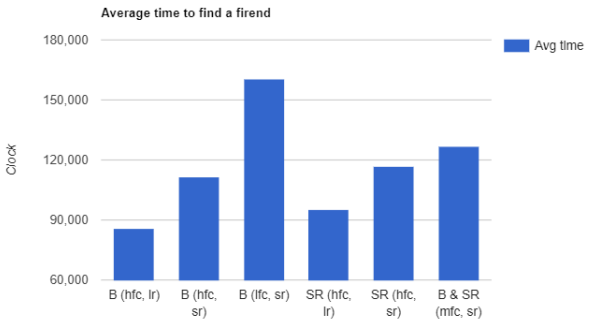

Figure 9: Average time to find a friend [hfc/mfc/lfc = high/medium/low frequency of communication. lr/sr = long/short range]

The simulations have showed that the shorter the range and the lower the frequency of communication, the longer the

system takes to reach a good distribution of information (Figure 8). This is crucial for the B model to complete a search, but in the SR this is not fundamental and the combination of the two can allow searches even in an early state of the system.
A problem underlined by the results obtained are started but failed searches. If a search is started (the location of the target is found in database or received after a search request) it is often needed to receive updates while moving, since the target is probably also moving from the target location. If updates are not received, the search fails.
In SR model, a new search request could be sent and in B model only more frequent broadcasts can solve the problem.
The reproduced scaled version of the festival keeps the average density of people as the real festival. A problem of coverage has been outlined, as shown in Figure 10.

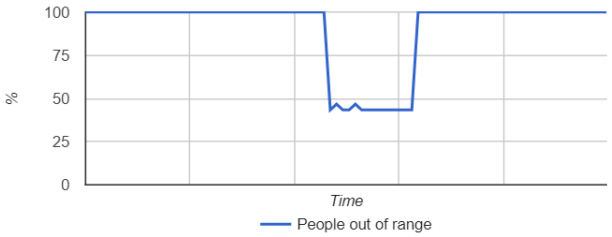

Figure 10: People out of range in the hybrid simulation. The reason for the drop are the evening concerts. One concert was attended by more than 50% of the guests. All the other where out of range

For most of the time, in simulations with range of 5 or 2 meters, people where not able to reach, directly or indirectly, more than half of all guests at the event. The problem was partially solved in our simulations by the frequent movements of the guests. In a low frequent broadcast scenario (as the hybrid model simulated) having many unconnected clusters can break the system since if the searcher is in a cluster different than the one with the target, where no-one has a recent location of the target, and these two clusters are unconnected the response time to the search request could be extremely long and often unacceptable. Simulations with more realistic clustering and movements of guests would be required.

5. CONCLUSION

Reuniting with friends a large concerts or festivals can be very difficult. Guests at venues like Roskilde festival might not bring their phone or other smartdevice either of fear of loosing it or simply because the batterylife is to poor with limited charging possibilities that they decide to leave it at home. In order to investigate weather a cheap solution exist to this problem we decided to design and implement a simulator. Using this simulator we simulated and designed a distributed embedded system consisting of cheap, robust bracelets. The bracelets are able to communicate in a peer-to-peer Bluetooth Low Energy network where they broadcast their own position as well as recently seen bracelets periodically.
We have simulated two different approaches to finding another bracelet. The first approach was, using **frequent broadcasts** we kept a local database with all known bracelets and their most recent location. Only if another bracelet was in the database a guiding phase could be initiated. Our simulations showed that the performance of this approach very much depend on the level of distribution of locations throughout the network of bracelets. If this is low, the chance to find the target in the database is low and hence the search cannot be initiated. Simulations underlined that the spreading time depends on the range and frequency of communication. The lower they are, the longer it takes.
In our controlled environment a satisfying level of spreading was reached under certain conditions but it is unlikely that this would happen in a real world scenario with thousands of people. The second option was to broadcast a **search request**. This broadcast would be relayed through the network until it either, reached the receiver or not. If the receiver was found it would reply with the current position. Simulations have showed that acceptable performance can be obtained under certain conditions but also that a strong limitation was the presence of started searches, after receiving the position of the target, but not succeeded due to the lack of updates on the target position on the way to him.
A third option which was a hybrid of the first two was also simulated. This approach was to have less frequent broadcasts but with the possibility to do search-requests if the other bracelet was not found in the database.
Simulations have revealed this to be the best approach, allowing a low frequency of broadcasting while triggering the system to react on demand to search requests and receiving updates using broadcasts on the way to the target. Unfortunately the coverage of the peer-to-peer network has been shown to be a problem, with a low radio range and people clustering at events.
We saw from the simulations of battery life that using a CR2032 coincell battery it is possible for the bracelet to last the 10 days we initially set as a lower limit. However there are some limitations as to how many times a bracelet can search and guide. We saw that using cheap LEDs we might have as little power as to a single search pr. day but using a blinking low power LED will help in that regard. As the battery is replaceable people should be able to search as much as they want.
The simulations showed that the system could work with the initial strong assumptions (no interference and no need for a GPS component). As we introduced some more realistic constraints (short radio range and lower frequency of communication), in order to simulate a more realistic scenario, the system started to show some limitations. Searches that were initiated but did not succeed and disappointing level of coverage started to show up.
As part of the possible future work, it would be interesting to simulate the behavior of the system with real data from an event such as Roskilde festival (people clustering in particular area, their movement and speed, the scheduling of events during the festival periods). This could either confirm or solve some of the discovered limitations. It could also reveal new ones not showed by our simplified models.

6. REFERENCES

[1] P. Displays. Why e-paper displays will run for 22 years in coin cell powered iot applications. [Online; accessed 07-December-2017].

[2] M. Electronics. Hlmp-k155 broadcom. [Online; accessed 07-December-2017].

[3] Energizer. Cr2032 spesifications. [Online; accessed 03-December-2017].

[4] T. Instruments. Power calculation tool for bluetooth low energy. [Online; accessed 03-December-2017].

[5] T. Magazine. 20 fun facts om roskilde festival. [Online; accessed 03-December-2017].

[6] B. technology. Core specification. [Online; accessed 03-December-2017].