Towards a simulation of AmI environments integrating social and network simulations

Álvaro Sánchez-Picot and **Diego Martín** and **Borja Bordel** and **Ramón Alcarria** and **Diego Sánchez de Rivera** and **Tomás Robles**[1]

Abstract.

We are heading towards a technological and hyper-connected world where every building is going to be full of sensors and actuators to monitor and interact with it, in what is known as an Ambient Intelligence (AmI) environment. The main problem when creating such environment is how expensive it can be, so a tool such a simulator could help to improve the way in which the devices are installed, testing with different configurations until you arrive to the optimal one. Also this simulator could help once the infrastructure is created to detect certain events before they happen, being able to apply a countermeasure. In this paper we propose the architecture to integrate a social and a network simulation in order to create a simulation for an AmI environment.

1 INTRODUCTION

We are heading towards a technologically connected world. More and more devices are installed in our homes and our environment. Some of these devices are not very new such as televisions, air conditioning units or security cameras but others are relatively new such as ambient lights, temperature sensors or microphones to talk with a computer. Currently we want to know much more about what happens in our home and our surroundings than several years ago and thanks to the mobile phones we can easily access this information anywhere and in real time. Nowadays we also want the environment to act proactively depending on what happens, for example, turn on the lights automatically when the nightfall comes and turn it off when there is nobody present or open and close blinds depending on the light outside or the desired temperature inside the building. This corresponds to what is known as Ambient Intelligence (AmI) environment, that is sensitive and responsive to the presence of people and environmental factors.

The idea of an AmI environment is that all its devices cooperate together in order to obtain a desired result. The intelligence behind all these devices resides in a computational system that manages the data of all the sensors an analyzes it to get an idea on what is happening in the environment. Then, using some predefined rules or some instructions from a person the actuators react to a system command to do certain tasks. For example, if the temperature is rising, the system receives a notification from the sensor, the system processes that information and send a command to the air conditioning in order to turn it on, until the temperature lowers and it can be turned off. An AmI environment is a complex one in that there can be lots of different devices recollecting information and the system can control lots of different actuators in order to affect the environment.

Deploying all the infrastructure to create an AmI environment in a building can be a very complex and expensive task, depending on the desired objective, not only for the cost of the devices and the required communication devices, but also the time to select the optimal position of these devices and the testing necessary to check that everything works as expected. In this research paper we propose a tool that helps in this task. That tool would be a simulator that enables the study of the optimal position for placing the devices assuring that the system works as expected. The problem is that we have simulators that perform part of the work but not a simulator that covers all the cases. I.e. there are social simulators able to simulate the behavior and movement of the people inside a building for the social part of AmI, and there are network simulators capable of simulating communications and devices of a network for the communication part of AmI; however, there is no simulator that integrates both simulations and use the outcome from the two simulations. This really is a problem because it's impossible to perform simulations of AmI environments and therefore its design, development and deployment will be very costly, and also many problems will arise that were not taken into account after deploying the AmI environment.

Other valuable functionality that is obtained from joining both simulators and interconnecting them to an AmI real time environment is that the simulator could analyze real time data and predict certain events that are going to happen and act in consequence, trying to avoid them to happen or minimizing the possible damage. The simulation could also use previous data to search for a pattern before certain event happens and use machine learning techniques.

This paper is the continuation of our previous work [8], where we expand the architecture, the models, add the prediction of events and present the current status of the simulation; we also present a tool that integrates both social and network simulators in order to obtain a simulator that covers all information and that is necessary in an AmI environment. In chapter 2 we present the related work this paper is based on. In chapter 3 we show the architecture. Chapter 4 explains several models created for the simulator containing a data model and a sequence model. In chapter 5 we talk about the use of the simulator in a real time environment to predict certain events. Finally in chapters 6 and 7 we present the conclusions and some future work.

2 RELATED WORK

This section describes the related work with the tool that we present in the paper. These includes AmI environments and also the simulations, specifically both the social simulation and the network simulation.

[1] Technical University of Madrid, Av. Complutense 30, 28040 Madrid Spain, Telecommunications School - ETSIT, email: alvaro.spicot@gmail.com

2.1 Ambient Intelligence

AmI is a discipline that makes our everyday environments sensitive to what is happening with the use of sensors, actuators, the network that interconnects all these devices and the server that orchestrates all these elements [2]. The main objective of AmI is the improvement of people's life that use the environment. In order to achieve this objective the information generated in the sensors is recollected and processed in the server and then certain orders are sent to the actuators, based on the information gathered from the sensors. The actuators in the end will influence the people that are present in the environment, ideally not being conscious of the technology. In AmI environments we expect several features [3]:

- Sensitive: The system needs to base its decisions on what is happening in the environment reacting to the people in them.
- Adaptive: The system also requires to adjust its behavior depending of the situation, considering the best possible behavior, and ideally anticipating to an event.
- Transparent: The people in an AmI environment should not be conscious of the technology that surrounds them. Thanks to the miniaturization of the technology this is easily achieved nowadays.
- Ubiquitous: An idea behind AmI is that it requires being present in as many places as possible, ideally everywhere. In this way there is more data recollected, and the more information, the best the system can react to a particular event.
- Intelligent: The system works using AI in order to achieve its goals. This is done recollecting the data from the sensors, processing it, and giving orders to the actuators in order to, in the end, influence the environment, specially the people.

AmI is mainly used in home environments controlling the elements of the house such as the air conditioning, the watering of the plants or the security but it can also be extended to larger places such as an office or a cinema to control those elements but also to prevent certain catastrophes such as a fire or, should it happen, manage the evacuation as best as possible guiding the people to the quickest and safer exit [5].

2.2 Simulation

Simulation is the process of designing a model of a real or imagined system and conducting experiments with that model to achieve certain goal [7]. Simulations of a very simple environment can be done with a mathematical model but one that is slightly complex, requires the execution of the simulation in a computer, as there are too many variables to take into account in the mathematical model.

There are many different types of simulations, each aimed for a particular field but in the field of AmI, as there two very important variables people and devices, we are going to focus in two simulations, the social simulation and the network simulation.

Social simulation studies the interaction among social entities taking into account their psychology and their behavior, both between people and with the people and the environment [4]. There are two main types of social simulation, system level simulation that analyzes the situation as a whole and agent-based simulation where we model a person (the agent) and its own behavior, and the interaction between agents will result in the overall behavior. We will focus in these last one as its way of working is more adapted to an AmI environment.

There are different agent based Social Simulators (SS) such as MASON, Repast [1], Swarm, each with its own characteristics and usually particularized for a certain case study. Some of them work with a 2D environment while others have a 3D one. All of them include some kind of physical engine to calculate the collisions between the agents and the environment. These simulators work using steps, so that all the information is updated every step (some seconds defined during the initialization).

The SS specializes in the behavior of the human and it can simulate other elements in an AmI environment such as sensors or actuators but it won't be able to get an very deep simulation of those devices.

In a network simulation, a program models the behavior of a network and each entity present in it, as well as the messages sent between them [2]. It can also simulate in detail the behavior of the entities such as routers or computers.

There are several Network Simulators (NS) nowadays both open-source such as NS or OMNet++ and proprietary such as OPNET or NETSIM [6]. All are event driven, meaning they calculate the next event in the network, where an event could be, for example, sending or receiving a packet or a new device that enters the range of a wireless network. After the simulation ends they generate a log that contains all these events, useful for a future analysis of the network.

NS are very good at simulating the network in an AmI environment and can simulate the other elements in this environment, mainly the people, using specific algorithms for their movement but they can't do a very realistic simulation, specially in their behavior, such a SS would.

3 ARCHITECTURE

In this section we present the proposed architecture that integrates both simulators. In order to achieve this we need to solve certain problems that might arise during the interconnection of both simulators. We have identified the following ones:

- Initialization: Each simulator requires specific information in order to start the simulation. Much of the information is shared between the simulators but probably a different format is necessary. Still some of the information is only required to one of the simulator, for example all related to the behavior of the people is only required by the SS while the NS only needs the position of the people, but nothing more.
- Synchronization: There is a very serious problem with synchronization while the NS is synchronous SS is asynchronous. I.e. the network simulator is based on events, updating the simulation when something happens, while the social simulation is based on steps, updating the position of all the agents every certain time. This requires a special synchronization between both simulators so that events are converted to time and everything can work.
- Visualization: Both simulators have their own visualization module but we need a common one to use with the integration so that the user can operate the whole AmI simulator from a single graphic interface. This visualization will have to manage the information from both simulators.
- Management: Different parameters can be managed before the simulation start as shown in figure 8, so we can set the behavior of the different elements as well as the characteristics of the environment. This helps to run several simulations with different parameters and then analyze the differences in the results.
- Decomposition: Both simulators require different parameters so, we need to keep track of the whole system but we need a way to particularize the information to each simulator as each simulator has its own way of processing the data.

- Results: Once the simulation finishes we need a mechanism to store the data generated so that we can analyze it in the future and compare the results from different simulations.

In order to solve these problems we propose the creation of an engine that will integrate both simulators including also an interface for the interaction with the user and a database to store the information generated. We call this engine Hydra. The general overview of this architecture can be seen in figure 1.

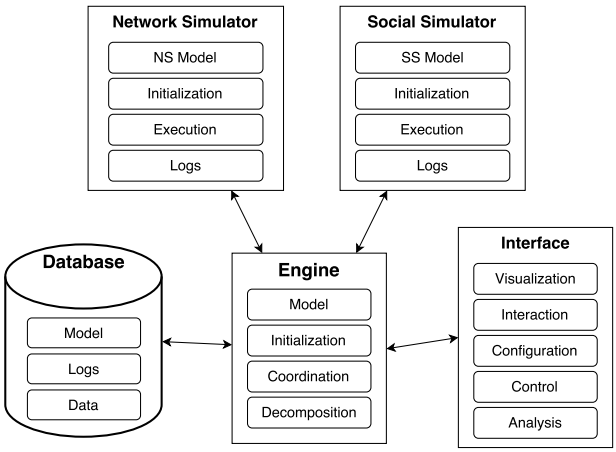

Figure 1. General Architecture

In this architecture we see both simulators communicating with Hydra. Hydra is going to integrate both simulators and is responsible of the following actions:

- The initialization of the simulation. Hydra has to send each simulator all the information it needs to start its own simulation. The user will be required to configure several parameters particular to each simulation specifying a condition that has to be met to end the simulation, such as a specific elapsed time or certain event.
- The synchronization of the simulations. After each step a simulator generates a new state of the elements in the simulation, and then Hydra needs to send the relevant information to the other simulator. For example, if after a step the SS updates the position of a person, this movement needs to be sent to the NS because it could imply the movement of the mobile phone this person is carrying and possibly it could enter or leave a wireless area.
- Ending the simulation. Once the ending condition of the simulation is reached as previously defined in the initialization or if the user manually ends it, Hydra recollects all the information that has been generated during the simulation in order to store it in the database so that it can be processed in the future. It also enables the user to view this information.

Hydra also works as the interface with the user allowing him to configure the initial parameters and to check the information generated once the simulation has finished. Hydra uses certain models presented in the next section and adapts them to each simulator following their requirements.

4 MODEL

In this section we present the different models associated with the architecture explained in the previous section. Firstly the data model used by the simulators which are executed by the engine. And finally several sequence diagrams that explain in detail the communications between both simulators in the different cases: when the simulation starts, when the SS needs to be executed, when the NS needs to be executed and finally when the simulation ends.

4.1 Data model

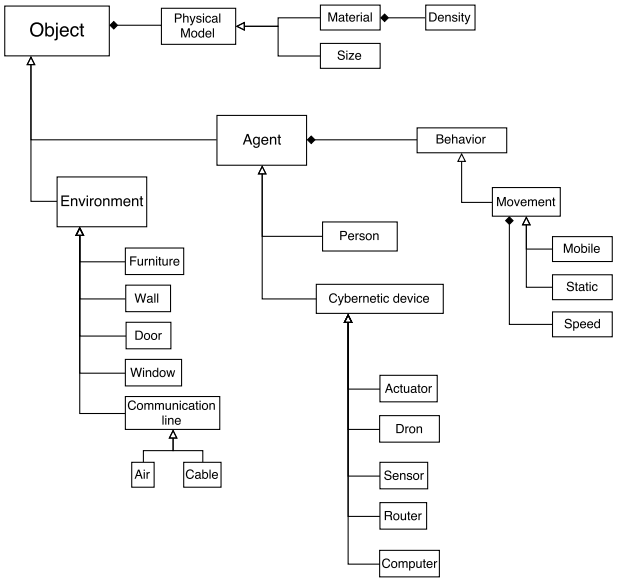

Figure 2. Network Simulator Model

In figure 2 we can see the data model used in the architecture previously presented by the NS.

This model is a general conceptualization of the different objects we can find in an AmI environment but oriented towards the network.

We have divided the object in two different fields. First there is the environment that includes all inert objects found in the defined space. Considering a closed space such as that of a building we can find in a room elements such as walls, doors and windows and inside these we can find different types of furniture as well. All these objects are general in an AmI environment but we can also find cyber-physical elements particularly important to the network such as communication lines.

Then we have the agents that represent all that requires certain intelligence in the simulation. It includes two subsequent groups, first the people that contains all the information relevant in the social simulation so that the NS will only be interested in their movement. Then there are also the cybernetic devices such as sensors, actuators, drones, etc.

All objects might possess certain relevant information in the physical model such as its size, weight or the material they have which might be interesting for the NS to check how the wireless communications propagate through the obstacles.

In figure 3 we can see the model specific to the SS. There are lots of elements shared with the NS model but some of them have disappeared as they are not relevant in a social simulation, such as communication lines, routers and computers. Instead there are some new elements such as everything related to the interaction between agents.

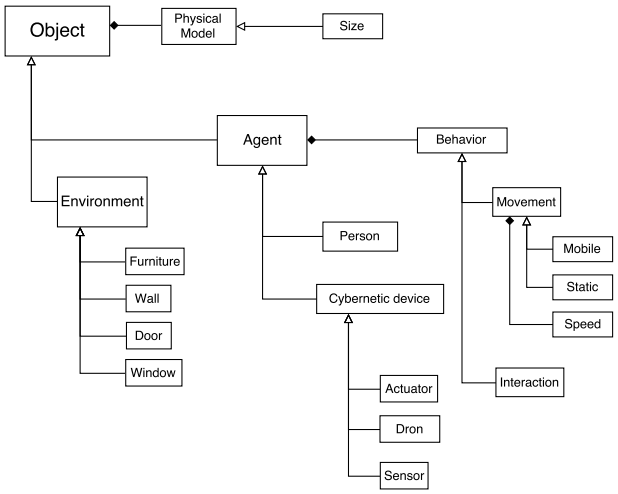

Figure 3. Social Simulator Model

4.2 Sequence model

In this subsection we will see in detail how the communication between both simulators and Hydra is done and in particular the different tasks Hydra needs to execute in order to guarantee that the events are solved in the correct order and that both simulators have the information updated.

There are 4 sequence models that correspond to the four different states the simulation can be, and are later explained in detail. In order to understand these states we need to explain first how Hydra works with both simulators and some key aspect of them.

One of the main tasks of Hydra is to coordinate both simulators and keep track of the current state of the simulation. In order to achieve this, Hydra stores the close future events in a queue ordered by in simulation time, so that the first event in the queue is the next one to be executed. Each event contains information to which simulator does it belongs. Each step Hydra extracts the first event from the queue and informs the corresponding simulator to execute an step in its simulation. Once the simulation of the step finishes the simulator informs the engine, possibly with information about new events generated that are then added to the queue in order. Then Hydra can possibly send information to the other simulator so that it can update its state, and finally a new step starts. All these process is explained in detail later with the sequence models.

Once the user informs Hydra to start the simulation there are four possible situations. The first one is the proper initialization of the simulation where each simulator starts its own simulation The second and third one are during the simulation when different events are extracted from the queue and sent to the NS or the SS as corresponds. Finally the simulation ends when the queue is empty or when a certain condition predefined by the user is met and then the information related to all the simulation is generated, processed and stored in the database.

4.2.1 Initialization of the simulation

The initialization of the simulation happens once the user has configured the parameters of the simulation and starts the simulation, both visually or in batch. This process can be seen in figure 4.

First Hydra has to load the different models from the database, necessary in the selected scene. These models are then particularized

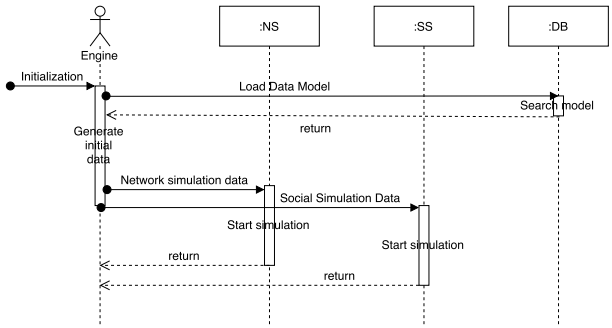

Figure 4. Initialization of the simulation

with the configuration parameters selected by the user and by the scene so that the different elements of the simulation can be placed in its locations and behave as expected. The models are also particularized for each simulator as not both simulators are going to need the same information as explained in the data models in section 4.1.

Then the information from the models is sent to each simulator so that it can start its own version of the simulation. Each simulator will then configure the simulation with the parameters received from Hydra, and once the set up is done, they inform Hydra that the are ready to continue with the simulation when required. Each simulator also send to Hydra information about what are the next events. In the particular case of the SS the only next event is when the next step happens as defined by the user, but in the case of the NS these events can be new packets generated or systems booting up are any other possible event.

Once the first simulator finishes configuring its simulation, Hydra creates the queue where the events will be stored with the information it received from the simulator. Similarly, once the second simulator ends, Hydra will add the events to the queue.

Now the simulation is ready to start. The next step explains how Hydra processes the queue and continues with the simulation.

4.2.2 Update of the social simulation

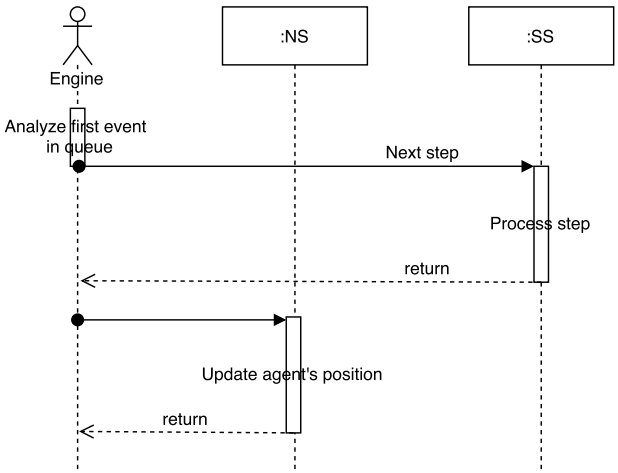

Figure 5. Update of the social simulation

Once the simulation is ready to start and then after each step is

resolved, Hydra extracts the first element in the queue (so it is also the first event in chronological order), removing it from the queue and then processes it.

If the first element is one from the social simulator, Hydra sends a message to the SS informing it that it can simulate the next step. In figure 5 it is explained how does this process work.

Each update in the SS usually requires to update the position of all the agents in the simulation depending on the interaction between them and the surroundings. Once the simulation has been updated, the SS sends a message to Hydra to inform that the simulation of the step has finished but his message also contains information about the new positions of the elements in the simulation and any other information that may be relevant. This message also contains information about when the next event is going to happen in the time of the simulation.

Then the engine processes this information and adds the event of the next simulation to the queue in the chronological order that corresponds. Hydra also processes the updated positions from the agents and sends a message with the information that is relevant to the NS so that it can update its own simulation and it is synchronized with the social simulation. Not all the information might be relevant to the NS, for example, depending on the scenario the position of a person might not be interesting to the NS, but it is the position of its mobile phone the one that is important.

Once the NS confirms that it has updated the new positions of the agents, the step is completed and then Hydra checks if a condition to end the simulation has happened. This condition is defined by the user when configuring the scenario and could be, for example, a certain time of the simulation or a certain region that has to be empty or there could be even no ending condition, as for example when running a visual simulation. In this case, the user will have to manually stop the simulation when he desires.

Now Hydra will extract the next event in the queue and continue with the simulation.

4.2.3 Update of the network simulation

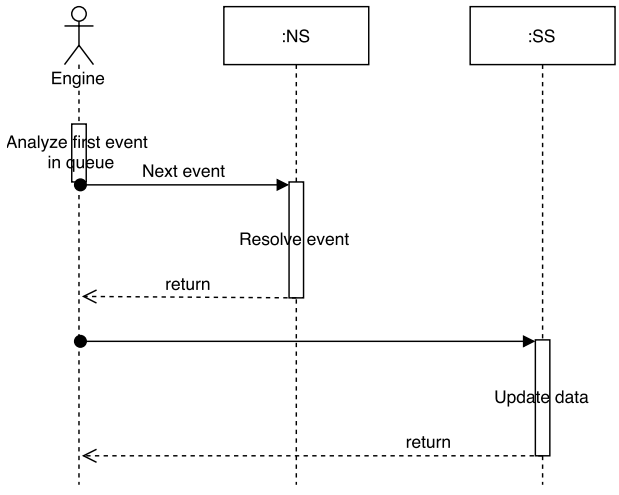

Figure 6. Update of the network simulation

Similarly to the previous case, once the simulation is ready to start or when a new step begins, Hydra extracts the first element in the queue (so it is also the first event in chronological order), removing it from the queue and then processes it.

If the first element is one from the network simulator, Hydra sends a message to the NS informing it that it can simulate the next event. In figure 6 it is explained how does this process work.

Each update in the NS requires to execute certain event such as a packet that arrives to a router and needs to be processed or a user that moves within range of a Wi-Fi. Once this event is resolved the NS informs Hydra that the update is complete. In the process of solving the event, new events might have been generated with a time-stamp in them. This events are sent to Hydra within the message informing the conclusion of the update.

When Hydra receives the message it adds the new events to the queue, if any, and then parses the information to send the one is relevant to the SS. Similarly to the previous case, not all the information will be relevant to the SS but some might. For example if a mobile phone has lost its signal the SS needs to know this information because the person could react to the event.

Once the SS confirms that it has updated the new information, the step is completed and then Hydra checks if a condition to end the simulation has happened as explained at the end of section 4.2.3.

Now Hydra will extract the next event in the queue and continue with the simulation.

4.2.4 End of the simulation

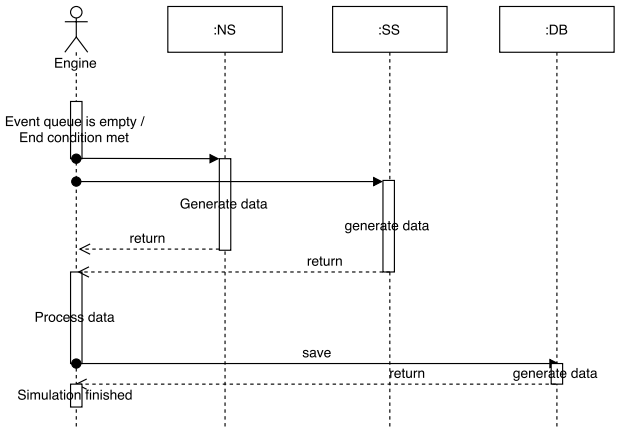

Figure 7. End of the simulation

Once the simulation reaches the end, Hydra has to do certain tasks to store all the information relevant to the whole simulation as explained in figure 7.

First Hydra informs both simulators to finish their simulation. Each simulator end the simulation but also process all the data that has been generated during the simulation and sends it to Hydra. In the case of the SS this information will be the evolution of the position of the elements in the simulation as well as certain parameters that might have changed. In the case of the NS this will be a file with all the packets that have been sent and its content as well as other parameters that might be relevant.

Once Hydra has the information from both simulators, it proceeds to store it in the database including all the parameters selected initially by the user as well as some other information as the date or the duration of the simulation.

Now the simulation has finished. If the user run a batch simulation, then a new simulation might start or if it was the last one then Hydra pauses and waits for new user input. If the user run a visual simulation, now there is a new management screen where he can analyze any simulation and compare them.

5 PREDICTION OF EVENTS

A simulation of an AmI environment should not only be used when designing the real AmI environment testing where the devices should be placed, but it could also be very helpful once the system is ready and the devices are installed in the environment. The idea is that the simulation can use the data from the devices in real time and use this information to predict future events that may cause the system to malfunction or something dangerous that could happen. A simulation is run with the data obtained in the present searching for certain patterns previously defined such as for example a great concentration of people in a small area. Should the simulation find this pattern the system tries to react in order to avoid it, for example indicating the people to abandon the area or, if necessary, informing a supervisor. Each time lapse a new simulation is run with the current data. This time lapse could be shorter or longer as required depending on how fast is going to change the data.

In order for this prediction to work the simulation should run fast enough so that when the data is processed, not enough time has been elapsed and the event has not yet happened so that some measures can be taken to prevent it.

The simulation can also use data from the past in order to predict these events searching for certain patterns that can cause them.

6 PROPOSAL OF VALIDATION

We are currently working in the validation of the architecture, creating Hydra to coordinate both simulators and an interface on each Simulation to do as intermediate between Hydra and the proper simulator.

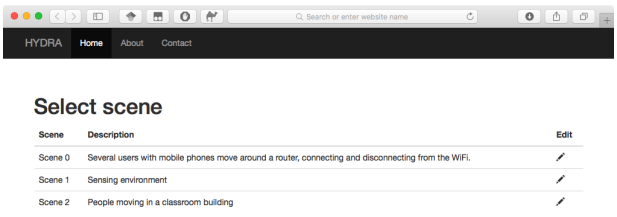

Figure 8. Scenario selection

Once Hydra and the simulators are started, the user can access the graphic interface using a browser. The first screen the user sees is a table with the scenarios as can be seen in figure 8. Currently three scenarios appear with a small description but ideally the user could create a new one or edit one that exists. In order to create a new scenario, a new screen would appear enabling the user to drag and drop different elements predefined to create the desired scenario and then add the configurable parameters.

Once the user selects a scenario, a new screen appears as can be seen in figure 9. The selected scenario is one with a router and different users with mobile phones walking around the router so that their

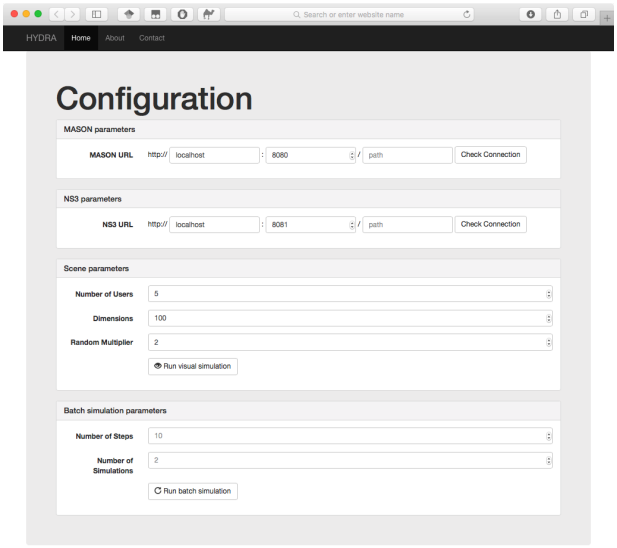

Figure 9. Configuration of the scenario

phones connect and disconnect from the network. The configuration screen enables the user to change the IP and port where the simulators are running as well as configure certain parameters proper to the simulation selected. In this case the user can choose how many people are in the simulation, the size of the area where the people can move and a parameter to define the movement of the people. Once the parameters are configured, the user can click to run a visual simulation what will take him to the next screen.

Alternatively, once the scene parameters are configured, the user can, rather than running a visual simulation, select to run a batch simulation. In this case the user can select how many simulations to run and each one, after how many steps are stopped. In this case the simulations will run in background and once finished the user will be taken to a screen where he can analyze the data generated in the simulations.

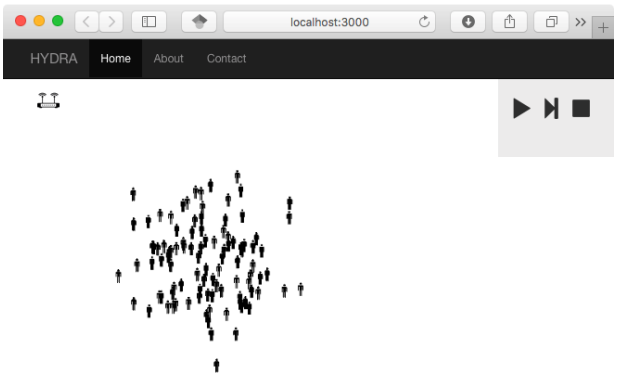

Figure 10. Running the simulation

Once a visual simulation starts, the user has control of the flow of the simulation, being able to run a step by step simulation or running the simulation as fast as possible. The user can pause the simulation and click in any of the elements in the screen to access its information. Once the user decides to stop the simulation, he is then taken to a screen where the data of the simulation appears.

7 CONCLUSIONS

In this paper we present an integration of a social simulation and a network simulator in order to get an enhanced AmI environment simulator that can precisely simulate the whole environment. A simulator for Ambient Intelligence environments is very useful due to they can be tested before being developed and deployed and checking if it's feasible; as a result these environments can be designed, developed and deployed more efficiently and effectively. Several difficulties have arisen during the development of this research work and we gave them solution with the proposal of an engine that integrates and coordinates and orchestrates both simulators. This engine is responsible of the initialization and coordination of both simulators keeping track of the different events that happen and the finalization of the simulation, storing the data generated. Apart from the engine there is also a visualization element that allows the user to follow the simulation as it advances and the inspection of the data generated, enabling him to check if everything worked as expected and comparing this data with one from a previous simulation. It also allows the user to configure the different parameters before starting the simulation and also configure an execution of a batch of simulations.

8 FUTURE WORK

The architecture presented in this paper enables several improvements. Here we comment some of them.

The most important task is the realization of a validation of the architecture. This would include a deployment of a simulated scenario containing several people and cyber-physical devices and the comparison of the data obtained in the real environment with that obtained in the simulation. We are currently working on this, but there is still a lot of work to do.

The scenarios we are working with are very basic and are created by hand. The user should be able to create its own scenario adding graphically the elements he wants, from those defined in the model, and configuring their parameters or being able to define the ones that can be configured later, just before the execution.

During the simulation the user should be able to modify the simulation on the run, so that he can experiment with new changes in the simulation, like moving certain agents, or adding or removing new ones. This would enrich the visual simulation so that it is not just a visual representation of the batch simulation.

Finally the screen that enables the user to analyze the simulation should be the most important one because this is why the user runs a simulation in the first instance. This screen currently shows the logs sent by each simulator, but it should present the information in a more visual way, enabling the user to see the simulation in each step as well as compare it with other simulations previously run. It should also enable the user to filter the parameters he is interested in.

Another idea for the simulator is the inclusion of different simulators in the engine. We have only included a social simulator and a network simulator but several others could be added depending on the scenario simulated. For example, a fire simulator, that precisely simulates the advancement of a fire inside a building, could be included in order to improve a simulation to test the evacuation time of a building. Other ideas could be a weather simulator a day and night simulator that can influence in the behavior of both the people and the cyber-physical devices in the simulation.

Finally, another possible future work could be the distribution of the simulation so that each component runs in one or several machines and the data is shared between all. This could be really important when the simulation works with the real environment firstly because the great quantity of data that it can process but also, should some of the machines stop working, the simulation could keep working with other machines continuing the work of the one that failed.

ACKNOWLEDGEMENTS

This work has been partially supported by the Autonomous Region of Madrid through program MOSI-AGIL-CM (grant P2013/ICE-3019, co-funded by EU Structural Funds FSE and FEDER) and has also received funding from the Ministry of Economy and Competitiveness through SEMOLA project (TEC2015-68284- R).

REFERENCES

[1] Robert John Allan, 'Survey of agent based modelling and simulation tools', Technical report, (2009).

[2] Lee Breslau, Deborah Estrin, Haobo Yu, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, et al., 'Advances in network simulation', *Computer*, (5), 59–67, (2000).

[3] Diane J Cook, Juan C Augusto, and Vikramaditya R Jakkula, 'Ambient intelligence: Technologies, applications, and opportunities', *Pervasive and Mobile Computing*, **5**(4), 277–298, (2009).

[4] Paul Davidsson, 'Agent based social simulation: A computer science view', *Journal of artificial societies and social simulation*, **5**(1), (2002).

[5] Augusto Morales, Ramon Alcarria, Diego Martin, and Tomas Robles, 'Enhancing evacuation plans with a situation awareness system based on end-user knowledge provision', *Sensors*, **14**(6), 11153–11178, (2014).

[6] Saba Siraj, A Gupta, and R Badgujar, 'Network simulation tools survey', *International Journal of Advanced Research in Computer and Communication Engineering*, **1**(4), 199–206, (2012).

[7] John A Sokolowski and Catherine M Banks, *Principles of modeling and simulation: a multidisciplinary approach*, John Wiley & Sons, 2011.

[8] Álvaro Sánchez-Picot, Diego Martín, Diego Sánchez de Rivera, Borja Bordel, and Tomás Robles, 'Modeling and simulation of interactions among people and devices in ambient intelligence environments', in *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 784–789. IEEE, (2016).