Magrathea: Building and Analyzing Ubiquitous and Social Systems

Jukka Perkiö and Petri Myllymäki
Helsinki Institute for Information Technology
Department of Computer Science
P.O. Box 68, FI-00014 University of Helsinki, Finland
{*jperkio, myllymak*}*@cs.helsinki.fi*

Abstract

Ubiquitous systems are rapidly becoming a more and more commonplace part of our everyday life. These systems may contain different classes of very heterogeneous components that have to function seamlessly together. A prime example of a class of ubiquitous components is given by the personal mobile devices. They are all pervasive and emerge in many forms: mobile handsets, PDAs, etc. Their features and computational powers make them a very capable platform. We present a pervasive agent- and sensing platform Magrathea that can be run on different kinds of computational devices. Magrathea can be used to build complex pervasive systems. As a practical example of the usage of this platform, we use it on top of personal mobile devices to investigate the structure of social networks of different individuals and to simulate viral behavior of agents. We also discuss analytical tools to further investigate, model and simulate the data obtained through our platform.

1. Introduction

Today's mobile devices offer a very capable and appealing platform to function as a part of pervasive systems. Some requirements for the components of ubiquitous systems are ability to sense, mobility, wireless connectivity, computational powers etc. Since the components of such systems tend to be rather heterogeneous, not all the components have to posses all the properties. Nevertheless, personal mobile devices such as mobile phones, PDAs, etc. tend to have a rather large subset of desired features already available.

The scope of different tasks that these systems are designed to handle is vast and it is growing as the systems become more commonplace. Currently the trend seems to be that the systems are designed for specific tasks and the interoperability between different systems is not very good. In the long run this is bound to change, since the more pervasive systems one strives for, the more interoperable they have to be. Agent Oriented Programming (AOP) as a programming paradigm is one possibility to try to make both designing and implementing interoperable ubiquitous systems easier. AOP also forces modularity in design and implementation, which is a clear benefit when dealing with high requirements for interoperability. We present Magrathea[1] a pervasive agent- and sensing platform that can be used hardware platform independently to build pervasive systems. Magrathea can be run on most computers, PDAs, many mobile phones etc. Magrathea agents are mobile agents and they can move wirelesly between devices that run the Magrathea platform.

Consider a common scenario in which one has context specific information needs. Location, time and the social situation are examples of context that may affect the situation. Classic examples are the menus and price lists when one is close to restaurants or timetables in train stations. There are already services that advertise these things through Bluetooth, SMS or other means of wireless connectivity. The common factor with these kinds of services is that they almost always function in push fashion. The user gets the message regardless whether he or she needs the service at that specific moment. The obvious answer for that is to implement services in pull fashion. This way the users get only the advertisements of services they need. For querying location specific services, the AOP paradigm is a natural choice. The operations are simple and there may be a large number different services that the user is interested in given a specific location and time.

Personalization is an important aspect for any truly ubiquitous system whether it deals with tasks like the example above or just personalizing the user interface of the gadget of choice. Personalized agents are a good starting point for this. A related aspect to personalization is to understand the social relations between users. This information can be used not only for personalization but also for inferring context e.g. whether there is somebody of user's social circle present in a situation. Consider an example, where we are interested in investigating how information or agents spread through a population. The spreading patterns are defined by the social relations of the given population. This can be investigated e.g. in the spirit of *reality mining* [8] and [9] using Bluetooth scan data to infer the social structure of the population. This method is valid and provides nice understanding of the social networks within the population. If one is mainly interested in the spreading patterns, then we can add some degree of accuracy to the observations by using mobile agents.

1. Preliminary results of the platform were reported in [13].

Observing spreading agents we can observe the exact path the agent moves from individual A to individual B instead of just knowing that that specific path exists. By changing the parameters of the agent we can define how long one has to spend time with another so that the hop of the agent happens. We can also define other conditions that have to be fulfilled. This clearly adds to the accuracy and diversity of observations.

Another interesting avenue for research with ubiquitous social systems is to use them for simulating viral behavior. Suppose we are interested in knowing how a certain viral agent behaves in a certain population, e.g. people in an office building etc. We can try simulating this using computer simulations. In reality this kind of simulations may be very complex and computationally expensive but for the sake of illustration we assume a very simple simulation where we build a model for the behavior of the population and another model for the behavior of the viral agent. This kind of approach is well justified and used in many epidemiological simulations. However models are always approximations of the phenomena that one tries to model. Clearly the accuracy of the simulation depends on the quality of the models that we use. In this case we have two models, one for the viral agent, and one for the human population that is in contact with the viral agent.

Suppose now that we can greatly improve the quality of the model for the population. This unquestionably improves the quality of the simulation as a whole. As all computer models are approximations of the real phenomena, the highest possible quality is attainable not through the use of models but through the use of accurate observations of the phenomena. Now the question is how can we accurately observe a real phenomena and at the same time run a simulation about it. In our example case we have two models, one for the viral agent and one for the population. One can observe the behavior of viral agents in laboratory conditions but on a real population it is not possible to do a controllable study both for the ethical and technical reasons. Hence we clearly need a computer model for the behavior of the viral agent. The behavior of the population on the other hand is quite possible to observe to a very high accuracy given a suitable ubiquitous system. Personal mobile devices with wireless connectivity provide means for the kind of ubiquitous system that would make the simulation of our example case more accurate. With wireless connectivity we are able to detect other devices in proximity very fast and with the computing capabilities of these devices we can run a model of the viral agent on that device. The model of the viral agent can also propagate from one device to another, hence infecting other devices in proximity according to the model coded in the agent. In our simple example simulation we have two approximative models that define the accuracy of the simulation. Now we are able to replace the other model – the model for the population – by accurate observations, and hence our simulation in that part is close to perfect.

One very important question related to all pervasive systems are the privacy and security concerns that raise from the close observation of individuals' daily activities. One can not emphasize that too much. Concerning our platform, the main principle to the privacy issues is that the users are aware of the purpose and implications and that the research has their consent. Also the underlying technical solutions must guarantee some minimum level of protection, so that unauthorized access to the data is not possible. From the security point of view the platform has to be designed so that unauthorized agents can not be run on it.

The rest of this paper is organized as follows: In Section 2 we present the design, architecture and implementation of the platform, in Section 3 we discuss the analytical tools to utilize the empirical data that the platform produces, in Section 4 we present results of empirical- and simulated experiments that validate the functionality of our platform. Finally in Section 5 we present our conclusions.

2. Design, architecture and implementation

In this section we describe the design choices, architecture and the implementation of the Magrathea platform. We also discuss the security and privacy implications of the Magrathea platform.

2.1. Design principles and choices

The most important design principle of the Magrathea platform is simplicity: the platform provides only the basic functionality and the more complex application logic are left as the responsibility of the agents. On the other hand the platform must be able to provide enough services for the agents so that they can be of reasonable size and complexity.

Even though currently the nature of our platform is in the research domain, the security and privacy issues are important. In [11] the authors discuss questions related to specific mobile applications that expose the social location of individual in social networks. They present some guidelines for the design of such systems. Our work, mainly as a research platform, differs from mobile applications quite a lot, but the issues raised in [11] are valid however and have to be considered as one can also see a path for our work to be extended to the commercial direction.

From the security point of view one has to consider questions related to the agent execution and and spreading. There has to be mechanisms to prevent unauthorized agents to be executed and a mechanism for limiting the operations available to the agents has to exist. On the mobile phone domain – which is the domain we use the platform in these experiments – there are two barriers that limit the allowed operations in the device. The hard one is the Symbian OS

platform security [15], and the soft one are the limits set by the Magrathea platform. The latter one is important when one runs the platform on other devices than Symbian based mobile phones, for example desktop computers. The Symbian platform security is based on using digital certificates and digital signatures that specify the capabilities each program running on Symbian OS can have. As our platform is coded using the Python programming language the hard limit, on Symbian based mobile phones, is set by the capabilities that are granted to the Python interpreter that we use. Our principle is to use only the minimum set of capabilities that are needed by a fully functioning platform. The second limit is set by the platform itself. The platform can check which agents are allowed to perform certain operations. This can be achieved through the use of digital signatures [14]. Currently we have implemented resembling functionality in lightweight manner using MD5 checksums.

The privacy point of view is twofold: How to prevent unauthorized access of the data and how to protect users' privacy when the platform is in use. As our platform is not a production environment and not in commercial use, these issues are dealt with trust based approach. This does not mean that we do not take these issues seriously. The solution for preventing unauthorized access is to use digital signatures. That mechanism is good and proven in many fields of computer science, electronic commerce, etc. The privacy issues are dealt case by case depending on the types of systems one is building. As one use case of the platform is investigating social networks, it is quite clear that the privacy concerns have to be dealt with. Most importantly all the users have to understand the nature of the system. The platform does not provide other forms of privacy protection than the fact that for each device only the neighboring devices are visible and the agents' spreading path is not visible in the spreading agents themselves.

As our approach is based on trust, the communication between devices is not encrypted, and the platform does not provide services for encryption. The platform provides services only for digital signatures. There are no limitations for the agents encrypting parts of the data though, but such functionality is solely on the responsibility of the agents.

2.2. Architecture

The Magrathea architecture shares characteristics from both client–server and peer to peer architectures. From the device and communication centric viewpoint the architecture is clearly a peer to peer one, since the devices are communicating directly without any servers in the middle. On the other hand, from the platform centric viewpoint, the architecture is a client–server one, since the platform performs the role of a server for the multitude of agents that play the role of client. In this case the clients use services of both the server on the local device and the remote device.

The Magrathea platform consists of the following components.

- *Control server* is the server – normally in the Internet – that controls remotely the individual devices. The control is done in *pull* fashion, i.e. the devices report to the server periodically and then act according the server instructions. The control server is also used for storing log data from the device.
- *Platform controller* is the local controller in the device that is responsible for controlling the platform behavior in the local device. Platform controller communicates with the control server and uses the platform services.
- *Magrathea server* listens for the connections from other Magrathea devices and uses platform services to authenticate requests and store the agents in the device.
- *Platform services* are the services that the platform controller and Magrathea server use for authenticating requests, communicating with the control server and other communication needs.
- *Agent scheduler* is responsible for executing periodically the stored agents in the device. The scheduler uses the platform services for checking agent privileges.
- *Agent storage* is a simple storage in a device for the agents.
- *Agent services* are the services for the agents i.e. the clients in the device. These services include APIs for agent propagation, controlling the life span of the agents, querying for other agents in the device, querying for other devices in proximity etc. These services also include most of the standard operations of the used scripting language.

Figure 1 shows the dependencies between different components.

The modus operandi of the Magrathea platform is simple. The aim is to provide all the agents a fair execution and the complex application logic is left as the responsibility of the agents. When a Magrathea device starts up, it performs a series of tasks that are listed and explained in detail below.

1) Start up the platform controller and set up the internal data structures for the existing data in the device. The data includes the Internet access point, control server address and possible existing agents in the device.
2) Set up the Internet connection for the connection to the control server. The access point to use and the control server ip–address are the only parameters that have to be preconfigured in the device.
3) Connect to the control server and receive a static ip-address for the WiFi connectivity. We chose this kind of address assignment over more elegant strategies (see e.g. [12]) because of the underlying design principle of simplicity and the resulting reliability. At this

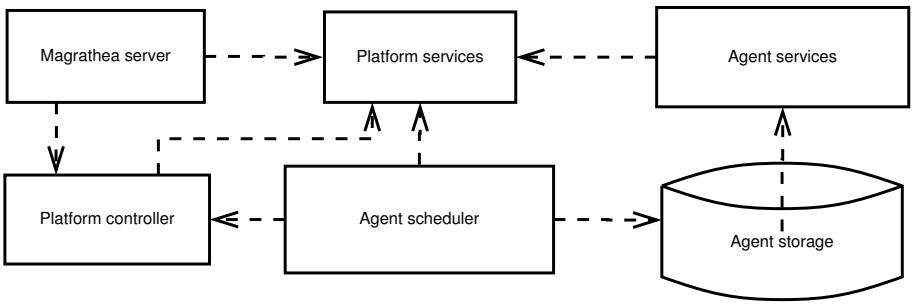

Figure 1. The dependencies of Magrathea platform components.

```
import magrathea, random

magrathea.propagate(0.75)
todie = random.random()
if todie <= 0.1:

    magrathea.die_out()
```

Figure 2. Example of a dummy Magrathea agent without any specific application logic. The agent propagates to the neighboring devices with the probability of 0.75 and dies out with the probability of 0.1.

point also a list of other authorized devices and their addresses is downloaded.

4) Receive the possible new agents. The main strategy for spreading agents is randomly pass them to the neighboring devices but this way specific devices may be handled appropriately if the system requires that. Also receive possible configuration changes for the device.
5) Start up the Magrathea server to listen for connections from other devices through the wireless connectivity.
6) Start up the agent scheduler. The strategy for agent scheduler is simple periodic execution of the agents in the device in random order. That is the only strategy that guarantees fair execution of all agents in case there are more agents on the device than it is possible to execute in one cycle of the scheduler. This strategy is also nicely in line with our design philosophy. For the possible new agents it is guaranteed that they will be executed once before the already existing agents on the device. After the first execution the strategy is same for all the agents.
7) Periodically connect to the control server and pull possible new agents and configuration changes and act accordingly.

Figure 2 shows an example Magrathea agent.

2.3. The implementation

The platform is implemented in Python[2], which is also available[3] for Symbian series 60 mobile phones. For the experiments in this paper we use only these phones. The specific model that we use is the Nokia N80 phone. This phone has Bluetooth and WiFi (802.11g) connectivity additional to the EDGE/GSM 850/900/1800/1900 and WCDMA 2100 standards, a three megapixel camera and support for the Python programming language.

As means of communication between the devices we use WiFi. WiFi is both reliable and fast means of communication. The other option for inter device communication would be the Bluetooth, but we chose WiFi over Bluetooth for the reliability reasons. It appears that the WiFi communication in this context simply is far more reliable than Bluetooth both in connection breaking and the device crashing. On the other hand the WiFi communication is much more expensive in terms of battery life and we use the WiFi radio at its lowest transmission power. Low transmission power also means short range, which is exactly what we need. Combining both means of communication would be ideal but remembering our main design principle, which is simplicity, we chose to use only WiFi.

The implementation is done using Python scripts and Python extensions implemented in C++. This choice is justified by the rapid development phase that the use of Python makes possible. Python is a full interpreted programming language, which provides both simple scripting capabilities and state of the art object orientated features. Aside of being implemented mostly in Python, all the agents are coded in Python as well, which makes the coding of agents a very fast process. Since the Python implementation for series 60 phones is not as complete as the standard version of Python, the needed extensions were implemented using standard Symbian C++ APIs.

2. http://www.python.org
3. http://sourceforge.net/projects/pys60

Symbol	Explanation
G	A graph.
$\mathcal{V}$	Set of nodes (vertices) in a graph.
$\mathcal{E}$	Set of edges in a graph.
$\mathbf{A}$	Adjacency matrix of a graph.
$\mathbf{D}$	Connectivity matrix of a graph.
$\mathbf{S}$	System matrix of a graph.
$\lambda_{1,S}$	Largest eigenvalue of system matrix
$\lambda_{1,A}$	Largest eigenvalue of adjacency matrix
$d_G(i)$	Degree of node i.
η_i	Fitness of node i.
Π_i	Probability of node i acquiring a link to/from a new node in fitness model.
$\rho(\eta)$	Fitness distribution for fitness model.
θ	Multinomial parameters.
l	Number of links to add when adding node to graph.
Z	Scaling constant.
β	Agent birth rate. This is the probability that the agent infects neighboring node and it is coded in the model of the agent.
δ	Agent death rate. This is the probability that an agent dies spontaneously in an infected node and it is coded in the model of the agent.

Table 1. The notation used throughout the paper.

3. Modelling Magrathea behavior

At the conceptual level the system consists of agents and carriers. If the carriers are people, as is the case here, the resulting graph resembles the social graph of the population. It is not the true social graph however as it also contains connections between people who just happen to be in the proximity of each other even though they may be total strangers to each other. The empirical data that the system produces relates to three aspects of the system that are:

- The evolution of the network of relations between the carriers,
- the behavior of the population of carriers and
- the spreading of viral agents within the population.

We discuss each of the three aspects below. Table 1 summarizes the used notation.

3.1. The evolution of the social graph

A simple generative model for graphs is the variant of Erdős-Rényi model [10], which assumes edges to be added independently to the graph with uniform probability. This model generates graphs that one hardly finds in realistic situations such as the Internet, citation networks or some social networks [1]. This model is not interesting for our purposes since the degree distribution of nodes will be such that the model does not produce scale-free graphs.

Fitness model [3] is based on the idea that nodes compete in order to acquire a link to/from another node. The fitness of a node is the inherent competitive quality of each particular node. The more fit the node is, the more it acquires links to other nodes. The model starts with a small number of nodes and then at each step samples the fitness parameter η from distribution $\rho(\eta)$ and adds new nodes to the system so that a node i already in the system acquires a link to the new node with probability

$$\Pi_i = \frac{\eta_i d_G(i)}{\sum_j \eta_j d_G(j)}. \quad (1)$$

The basic form of fitness model assumes that the number of new links added to the system when a new node is added is constant. When one considers the formation of social networks in real social situations this may not be realistic. We take this into account by additionally sampling the number of links l to add at each node addition from a special case[4] of multinomial distribution, i.e.

$$l \sim Multinomial(1, \theta).$$

Fitness model also assumes that the fitness is a static property of a node. This may not be realistic either as it is reasonable to assume that the interestingness of a person, e.g. an actor or politician is not constant but changes during time. We do not consider these aspects in our experiments but one can consult e.g. [7], [2] for further reference.

The most important aspect of this model in our setting is that it can be used to extend the empirical results for simulations. One only estimates the fitness distribution, the multinomial for the number of links to introduce to the system at each time and the link weight distribution for new links in order to generate much larger networks with similar properties as the empirical data.

3.2. The behavior of carriers

One can also investigate some behavioral aspects of the carriers using the data our system produces. The two main aspects that are available are:

- Temporal social interactions and
- the location of the carriers as a function of time.

The former is readily available from the logs of the system and the latter is available as well, provided that there are stationary nodes whose location are known. Given such nodes and the recorded interactions between the carrier nodes and the stationary nodes one can also estimate to some degree the location of carriers that are not in proximity of any known stationary node.

A simple model for the pair-wise co-occurrence behavior of the nodes i and j is

$$p(i, j) = \frac{1}{Z}\mathbf{D}_{ij}. \quad (2)$$

This means that the link structure of the graph and the connection weights $\mathbf{D}$ are generated by the process presented in Section 3.1 and then at each time step the appearance of nodes i and j together is sampled as above.

4. Sometimes this distribution is called categorical distribution or simply discrete distribution.

3.3. The spreading of viral agents

The spreading of viral agents in a given graph $G = (\mathcal{V}, \mathcal{E})$ is defined by the properties of the viral agent and the properties of the graph. The simplest case is when all the connection weights between nodes are equal and the viral agent has constant probability β to infect its neighbors and constant probability δ of spontaneous death. In this case the survival of viral agent is defined solely by the ratio β/δ. It is shown in [16] that if this ratio

$$\beta/\delta \begin{array}{ll} < & \\ > & \end{array} \frac{1}{\lambda_{1,A}}, \begin{array}{l} \text{viral agent dies out,} \\ \text{epidemic happens,} \end{array} \quad (3)$$

where $\lambda_{1,A}$ is the largest eigenvalue of the adjacency matrix of graph G. In the above, if the (equal) connection probability (i.e. the weights) between nodes is not 100%, one has to take that into account in the agent birth rate β.

It is not realistic to assume equal connection weights between nodes or constant agent birth- or death rate. In realistic social situations the strength of connections between individuals vary and some individuals are more resistant for some viral agents than others. The simple threshold condition presented above does not hold for dynamic graphs.

In [5] a model for information survival in dynamic graph topologies is presented. This model takes into account different connection strengths between nodes, node specific transmission probabilities and possibility of nodes dying and resurrecting. The survivability of agents, information, etc. is predicted by following procedure. One forms a *system matrix*, which describes the graph so that the diagonal elements give the probability of nodes staying alive and off-diagonal elements give the probability of node i infecting node j. The survivability of agent or information is defined by the largest magnitude eigenvalue $\lambda_{1,S}$ of the system matrix so that if

$$\lambda_{1,S} \begin{array}{ll} < & 1, \\ > & 1, \end{array} \begin{array}{l} \text{viral agent dies out,} \\ \text{epidemic happens.} \end{array} \quad (4)$$

For further reference one should look at [4], [5], [6], [16].

One can apply the above model with slight modifications for analyzing the Magrathea behavior. The difference is that instead of having node specific death and resurrection probabilities, we use still viral agent death probabilities. If one wants to model more complex configurations, e.g. consider the effects of temperature or the lightness of environment etc. these models do not suffice anymore. This is the place where one needs simulations and empirical observations combined and our pervasive platform enables us to do exactly that.

4. User experiment

A user experiment was performed in February 6-19, 2008 using Nokia N80 smart mobile phones running the Magrathea platform. The aim of the experiment was to evaluate the platform and to investigate how the spread of agents can be used for analyzing the structure of the social network of participants. The experiment consisted of ten test subjects carrying the devices and they were instructed as follows:

1) When you go to work, turn on the phone and keep it with you all the time until you leave for home.
2) When you leave, turn off the phone and put it in the charger.
3) You may recharge the phone during the day when sitting in the office but only if you are sure that you remember taking it with you any time you leave the office.

The experiment was repeated identically daily and the setting was following:

- When the phone is turned on, it deletes all the agents in the system and initializes the platform with one maximally contagious agent that is specific to the device.
- When the phone detects another device in proximity, the agent migrates to that device with the probability of 1.0.
- Because of high power consumption of WiFi network it is kept on only periodically making the propagation functionality also periodical. The WiFi radio transmission power is set to its lowest level possible.

Parameter	Value
Number of participants	10
Length of the experiment	8 working days
Length of the agent life cycle	1 day
Initial number of agents per device	1
Contagiousness of agents	Maximum (prob. 1.0)
Propagation on time	300 ± 120 s.
Propagation off time	300 ± 120 s.
Max. number of agents to propagate at one cycle	5
WiFi transmission power	4 mW
Physical setting	Office building

Table 2. Summary of the experimental setting for the user study.

Table 2 summarizes the settings for the user experiment.

The experiment produced some 167000 lines of log data that includes the agent spreading patterns and some debug info of the system. The graph of the social network within the test subjects is also found in this data. As the spreading patterns are defined mostly by the social graph within the test subjects, we first discuss that.

In Figure 3 the empirical social graph of our user experiment is shown. The graph is quite densely connected but many of the connections are rather weak, as can be seen in Figure 4, which shows only strong connections. This is further illustrated by the basic statistical figures shown in

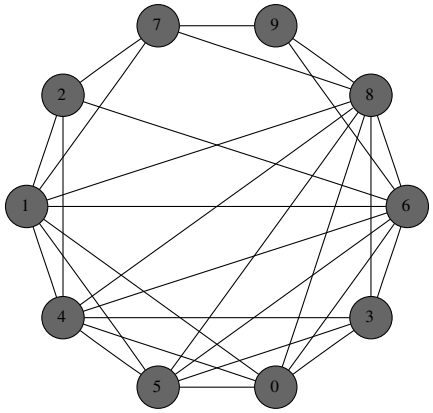

Figure 3. The resulting social graph of our experiment visualized showing all the connections.

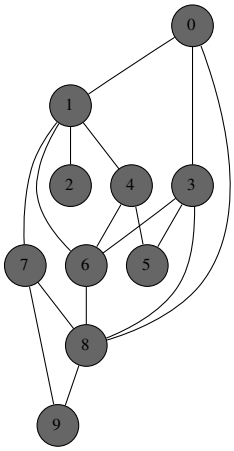

Figure 4. The resulting social graph of our experiment visualized showing only important connections when both the succeeded and failed connection attempts are treated similarly.

Table 3 and by the visualization of the connectivity matrix of the graph shown in Figure 5.

It seems that within this experimental setting the fitness of the nodes (see Table 1 and Eq. 1 for definition) is defined firstly by the presence of the subject at the office and secondly by the social interactions. There are reasons to believe that the social interactions would be more important than the mere presence at the office had the experiment been larger, however. This is based on the fact that our experiment was done in a very confined setting, whereas truly pervasive setting would result richer social interactions, e.g. at home, at hobbies, etc. It is also worth noticing that the strong correlation of presence at the same place at the same time implies social relation but does not necessarily mean it.

Parameter	Value
Average connectivity between any two nodes	0.011
Highest connectivity between any two nodes	0.080
Lowest connectivity between any two nodes	0
Average connectivity within neighborhood of any node	0.019
Highest connectivity within neighborhood of any node	0.049
Lowest connectivity within neighborhood of any node	0.007
Average degree of nodes	5.8
Highest degree of nodes	8
Lowest degree of nodes	3

Table 3. Summary of the resulting social graph.

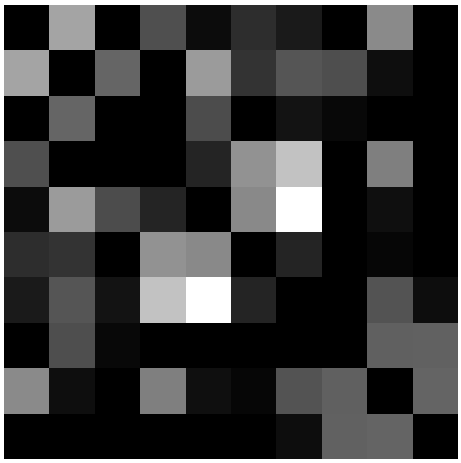

Figure 5. Visualized connectivity matrix for the nodes in the social graph of our user experiment.

The user experiment was repeated daily from the state where each device is infected by only one viral agent. Figure 6 shows the average number of viral agents that have spread to the device during a single day in the course of the experiment. We measured the size of the epidemic by the number of foreign viral agents in the system. An agent is foreign to a device if it has not originated from the device where it is found. In our experimental setting the maximum possible number of foreign agents in the system is 90 (see Table 2). Figure 7 shows the average, fastest and slowest progression of the epidemic measured. As we can see the progression of the spreading agents does not reach the maximum during any day of the experiment. Nevertheless had we run the experiment continuously long enough without starting it over every day, the epidemic would have spread to the maximum given the parameters we used.

4.1. Simulations

We run simulations using two kinds of data. First we used the empirical data and second we generated larger networks from the empirical data. In the remaining section we present

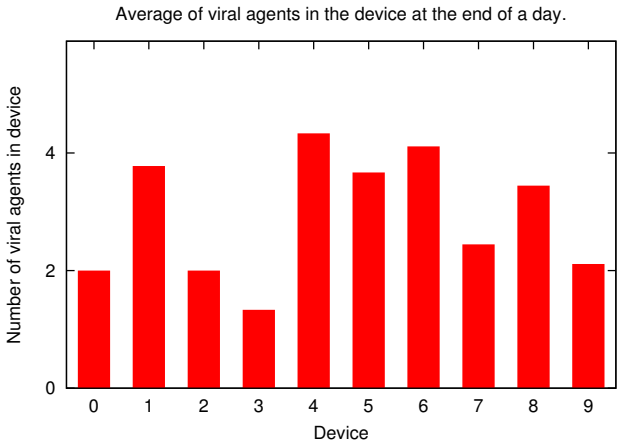

Figure 6. Daily average of foreign viral agents in a device.

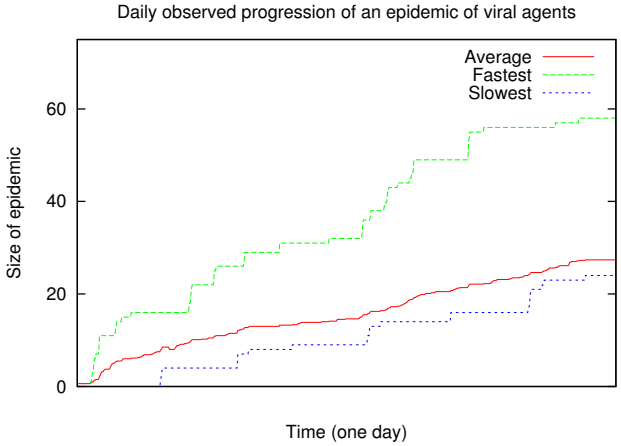

Figure 7. The daily progression of the spreading agents when the experiment is repeated. The y-axis shows the number of foreign agents in the whole system.

the results of simulations for two small cases as they serve only as examples how to do simulations on top of the data our platform produces.

As a result of the user experiment we have the complete log of all activity within the system, which can be also observed in real time during experiments. As an illustrative example we chose to investigate how the epidemic would have spread had we not started the experiment from the beginning each day. We chose to use only the real recorded interactions between the test subjects and examined how the parameter β affects the evolution of the epidemic within our test subjects. Figure 8 shows the results for this simulation. As the network is connected and the agent death rate δ is zero, the epidemic will always eventually spread to the whole network.

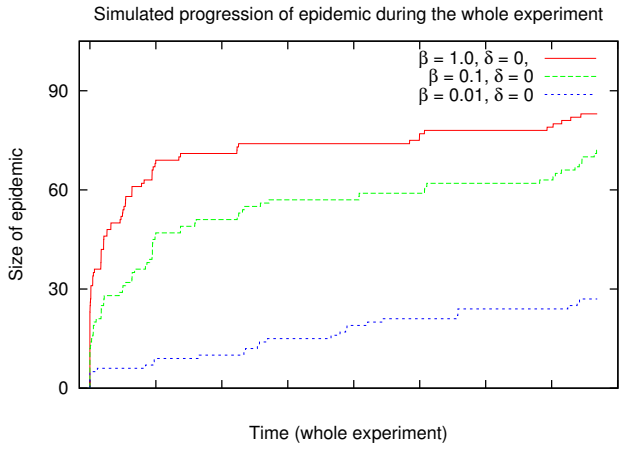

Figure 8. Size of the simulated epidemic with different parameters. The simulation was done using the real recorded connections between devices. Y-axis shows the number of foreign agents in the system as a function of time.

For generating graphs with similar properties as the empirical data we use the fitness model with the multinomial sampling for the number of new links and we also sample the connection weights for new links. One has to estimate following parameters in order to generate a graph.

1) The fitness distribution: we estimated it by defining the fitness of each empirical node to be the mean of connections it received from other nodes and normalizing that by the most fit node. This resulted the fitness being between 0 and 1. The normalized fitness scores of the nodes were then linearly interpolated to 100 discrete bins.
2) The link distribution, i.e. the number of links to add at each node addition: we set it to $Multinomial(1, \theta)$, where $\theta = (0.7, 0.15, 0.1, 0.025, 0.025)$. This is not the link distribution over the whole graph.
3) The connectivity distribution, i.e. the weight for each new link added: we used the exact empirical distribution from our user experiment.

We tried a series of simulations on generated graphs of different sizes. The graph in Figure 9 was generated using the above parameters. The empirical links and nodes are shown in darker color than the generated links and nodes. Figure 10 shows the size of an simulated epidemic on that graph, when the initial condition was such that every node was initialized to have one node specific viral agent. This setting was similar to our user experiment. As it can be seen the virus death rate δ controls whether the epidemic forms or not and the birth rate β affects more on the speed of the spread of the epidemic. Figure 11 shows results on the size of the epidemic when the initial condition was such that all

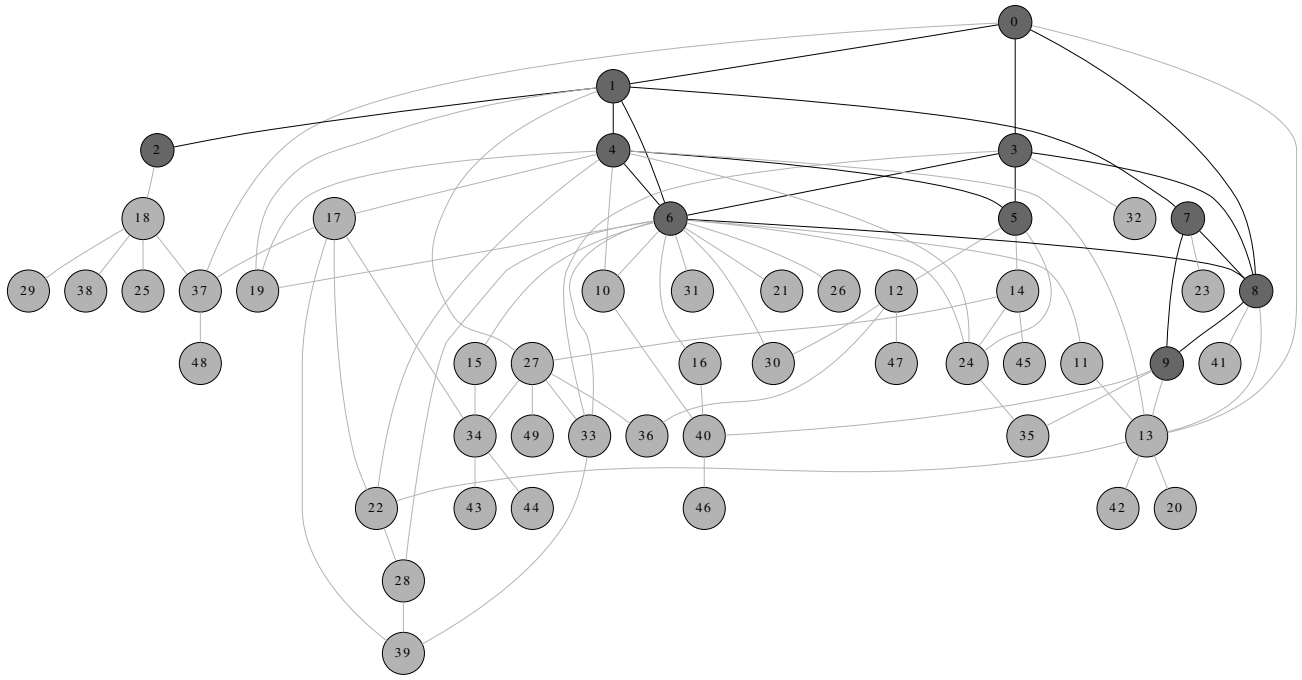

Figure 9. A generated graph of 50 nodes using fitness model and multinomial sampling for the number of new links to add at each node addition. The fitness distribution is estimated from the empirical data from our experiments. The empirical links and nodes are shown with darker color than the generated links and nodes.

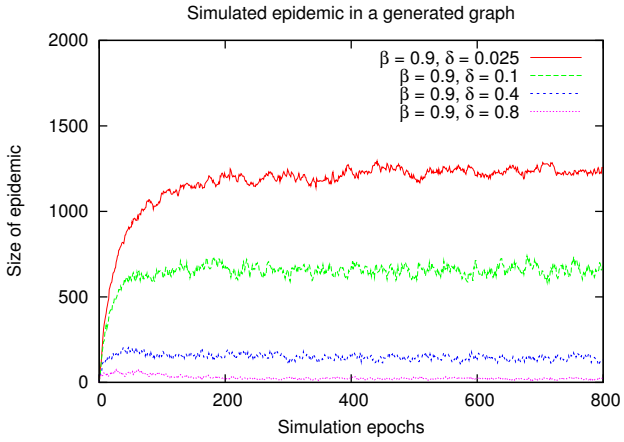

Figure 10. Size of the simulated epidemic with different parameters in a generated 50 node graph with similar properties as the empirical data. The initial condition was such that each node had one node specific agent present in the beginning of the simulation. This setting was similar to our user experiment.

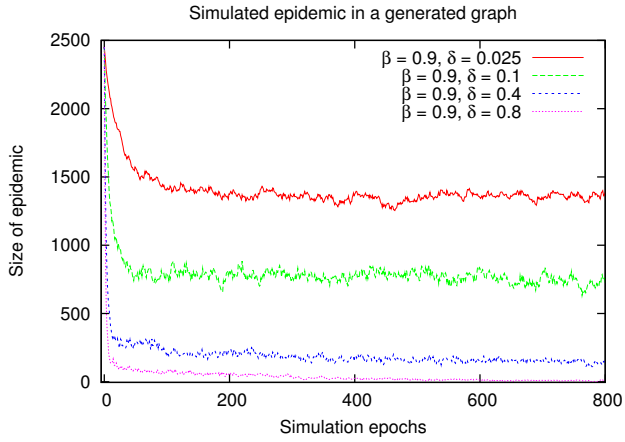

Figure 11. Size of the simulated epidemic with different parameters in a generated 50 node graph with similar properties as the empirical data. The initial condition was such that every node had all the node specific agents present in the beginning of the simulation.

the nodes had all the agents present in the beginning of the simulation. For both cases above the maximum size of an epidemic is 2450.

In Figures 10 and 11 one also notice that the size of the epidemic is not the same at the end of both runs. That is because we show only 800 first simulation epochs and the state of the simulation has not stabilized completely yet.

5. Conclusions

We have presented Magrathea: A pervasive agent- and sensing platform in order to investigate how to use mobile agents

- as basic building blocks for ubiquitous systems,
- to investigate the structure of social networks and
- to simulate viral behavior within a population.

The motivation for the Magrathea platform is two-fold: First the design and implementation of ubiquitous systems, especially with their interoperability requirements pose a challenging task. One answer to this task is to use a single (mobile) agent oriented programming platform on multiple hardware platforms. Second we wanted to investigate the use of ubiquitous systems for analyzing social networks and run simulations of the spread of viral agents. The more pervasive the systems are, the more possibilities for analysis and simulations they provide.

We presented the design and implementation of the Magrathea platform. We also discussed the generative models in order to further analyze different aspects of the platform, i.e. the evolution of social networks, the behavior of agent carriers (users) and the spread of agents within users. We also performed a user experiment and simulations in order to validate the functioning of our platform.

Our experiments provided us with positive results on the main aspects of the platform. Agent oriented programming provides means for rapid development of complex pervasive systems on different hardware platforms. Ubiquitous systems have potential to be used for analyzing social networks. Distributed computing and simulations on ubiquitous systems are yet a neglected niche but there is a huge potential for that when the pervasive systems become more matured, general and commonplace.

One last thing worth mentioning are the almost endless possibilities for biological viral modelling using all the time richer sensing capabilities of personal mobile devices. The models for viral agents can be very sophisticated using the sensor information that the devices provide. Examples include temperature, lightness, location, acceleration, humidity, etc. The limits are set by the capabilities of the devices and the devices are getting better all the time.

In the future we plan to further develop the platform and use it to build and study larger scale pervasive systems and to run larger simulations combined with richer sensing capabilities.

Acknowledgments

This work was supported in part by the SensorPlanet project of Nokia, by the IST Programme of the European Community under the PASCAL Network of Excellence, and by TEKES under the PUPS project.

References

[1] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74(1):47–97, Jan 2002.

[2] L. A. Amaral, A. Scala, M. Barthelemy, and H. E. Stanley. Classes of small-world networks. *Proc Natl Acad Sci U S A*, 97(21):11149–11152, October 2000.

[3] G. Bianconi and A.-L. Barabasi. Competition and multiscaling in evolving networks. *EPL (Europhysics Letters)*, 54(4):436–442, 2001.

[4] D. Chakrabarti. *Tools for large graph mining*. PhD thesis, Pittsburgh, PA, USA, 2005. Chair-Faloutsos, Christos.

[5] D. Chakrabarti, J. Leskovec, C. Faloutsos, S. Madden, C. Guestrin, and M. Faloutsos. Information survival threshold in sensor and p2p networks. In *INFOCOM*, pages 1316–1324. IEEE, 2007.

[6] D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec, and C. Faloutsos. Epidemic thresholds in real networks. *ACM Trans. Inf. Syst. Secur.*, 10(4):1–26, 2008.

[7] S. N. Dorogovtsev and J. F. F. Mendes. Evolution of networks with aging of sites. *Phys. Rev. E*, 62(2):1842–1845, Aug 2000.

[8] N. Eagle. *Machine Perception and Learning of Complex Social Systems*. MIT, 2005.

[9] N. Eagle and A. S. Pentland. Reality mining: sensing complex social systems. *Personal Ubiquitous Comput.*, 10(4):255–268, 2006.

[10] P. Erdös and A. Rényi. On random graphs, i. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.

[11] G. Iachello, I. Smith, S. Consolvo, M. Chen, and G. D. Abowd. Developing privacy guidelines for social location disclosure applications and services. In *SOUPS '05: Proceedings of the 2005 symposium on Usable privacy and security*, pages 65–76, New York, NY, USA, 2005. ACM Press.

[12] M. Mohsin and R. Prakash. Ip address assignment in a mobile ad hoc network. In *Proceedings of MILCOM 2002*, 2002.

[13] J. Perkiö, P. Myllyṁaki, V. H. Tuulos, and P. Boda. Magrathea: A mobile agent- and sensing platform. In H. R. Arabnia and V. A. Clincy, editors, *Proceedings of the 2008 International Conference on Wireless Networks, July 14-17, 2008, Las Vegas, Nevada, USA*, pages 494–500. CSREA Press, July 2008.

[14] R. L. Rivest, A. Shamir, and L. M. Adelman. A METHOD FOR OBTAINING DIGITAL SIGNATURES AND PUBLIC-KEY CRYPTOSYSTEMS. Technical Report MIT/LCS/TM-82, 1977.

[15] M. Shackman. Symbian os v9 – Platform security, 2005.

[16] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos. Epidemic spreading in real networks: An eigenvalue viewpoint. In *srds*, volume 00, pages 25–34, Los Alamitos, CA, USA, 2003. IEEE Computer Society.