Magrathea: A Mobile Agent- and Sensing Platform

Jukka Perkiö[1], Petri Myllymäki[1], Ville Tuulos[2] and Péter Boda[2]
[1]Complex Systems Computation Group
Helsinki Institute for Information Technology
Department of Computer Science, University of Helsinki, Helsinki, Finland
[2]Nokia Research Center, Palo Alto, California, USA

Abstract – *Personal mobile devices are all ubiquitous in many forms like mobile handsets, PDAs, etc. Their features and computational powers make them a very capable platform for wireless sensing and mobile agents. We present Magrathea: A mobile agent- and sensing platform that is built on top of mobile smart phones. We discuss the motivation behind the platform, present the design choices, the architecture and discuss the possible use cases of the platform. We also show empirically how the platform can be used for investigating viral phenomena within human test subjects by modeling the viruses with the mobile agents, for detecting the social networks within the test subjects and how the results can be further refined by simulating other possible agent configurations using the acquired data.*

Keywords: Mobile agents; Sensor networks; Pervasive computing; Contagious agents; Viral Simulations; Social networks

1 Introduction

Today's mobile devices present an unparalleled potential to investigate and simulate different phenomena that is related to the interaction and movement of people. Not only is the nature of most mobile devices very personal – mobile phones are a clear example of that – but their features and computational powers today are at very high level. This makes them a very appealing platform for wireless sensing and mobile agents. In this paper we present Magrathea: A mobile agent and sensing platform that can be used among other things for simulating and investigating the spreading of viral agents in a human population.

Suppose that we are interested in knowing how a certain biological viral agent behaves in a certain human population, e.g. people in a confined area like an office building etc. We can try simulating that 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. Nevertheless 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 that we can somehow greatly improve the quality of the model for the population. That unquestionably would improve the quality of the simulation as a whole. As all computer models are approximations of the real phenomena, then 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 that is not possible to do in a controllable manner both for the ethical and technical reasons. 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. Today almost everyone has a personal mobile phone and more increasingly those phones are smart phones that have a very rich set of features that make them more and more indistinguishable from personal computers. Personal mobile devices with wireless connectivity provide means for the kind of realistic observations 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.

Consider another example, where we are interested in investigating how information spreads through some population. The spreading of information is defined by the social relations and structure of the given population. This can be investigated e.g. in the spirit of *reality mining* [2] and [3] 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 paths of the spreading of information, then we can add some degree of accuracy to the observations by using the same mechanism that was discussed above. By using spreading contagious agents we can observe the exact path the agent moves from individual A to individual B instead of knowing that that specific path is just possible. 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 the observations.

One very important question that is related to above mentioned examples are the privacy and security concerns that raise from the close observation of individuals' daily activities. One can not emphasize that too much. The main principle regarding to the privacy issues has to be that the participants are aware of the purpose and implications and that the research has their consent. Also the technical platform that is used have to 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 designed so that unauthorized agents can not be run on it. In this paper we present an architecture and implementation of such a system.

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 graph theoretic considerations about the platform and modeling epidemics using it. We also present results of empirical- and simulated experiments that validate the functionality of our platform. Finally in Section 4 we present our conclusions.

2 Design, architecture and implementation

In this section we describe the design choices, architecture and the implementation of 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 Magrathea platform is simplicity. The platform should provide only the basic functionality and the more complex application logic should be the responsibility of the spreading agents. On the other hand the platform should provide enough services for the agents so that they can be of reasonable size and complexity.

Even though the nature of our platform is in the research domain, the security and privacy issues are important. In [4] 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 as sole research platform differs from those applications quite a lot, but nevertheless the issues raised in [4] are valid and have to be considered as one can also see a path for our work to be extended to possible 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. In principle there are two barriers that limit the allowed operations in the device. The hard one is the Symbian OS platform security [7], and the soft one are the limits set by the Magrathea platform. Symbian platform security functions by using digital certificates and digital signatures to specify the capabilities each program running on Symbian OS can have. As our platform is coded using Python programming language (with some Python extensions written in C++), the hard limit 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 [6]. Currently we have not implemented this functionality, since the Symbian OS platform security is adequate for our purposes. Nevertheless each agent that spreads in the system has to be signed using at least MD5 checksums.

The privacy point of view is twofold: How to prevent unauthorized access of the data and how to protect users' privacy as the platform is in use. As our platform is meant for research purposes and not for 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 simulations one is running. As the other main usage of the platform is for investigating social networks, it is quite clear that the privacy concerns have to be dealt with. Most importantly all the participants have to understand the nature of the simulations. 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 in the simulations, the communication between devices is not encrypted, and the platform does not provide services for encryption. 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 spreading 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 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 spreading 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 yet powerful. The aim is to provide all the agents a fair execution and the complex application logic is the responsibility of the spreading 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. [5]) because of the underlying design principle of simplicity and the resulting reliability. At this point also a list of other devices in the experiment and their addresses is downloaded.
4. Receive the possible contagious agents that are introduced in the experiment this way. The main strategy of spreading contagious agents is randomly pass them to the neighboring devices but this way specific devices may be infected (or cured or immunized) if the experiment requires that. Also receive possible configuration changes for the device.

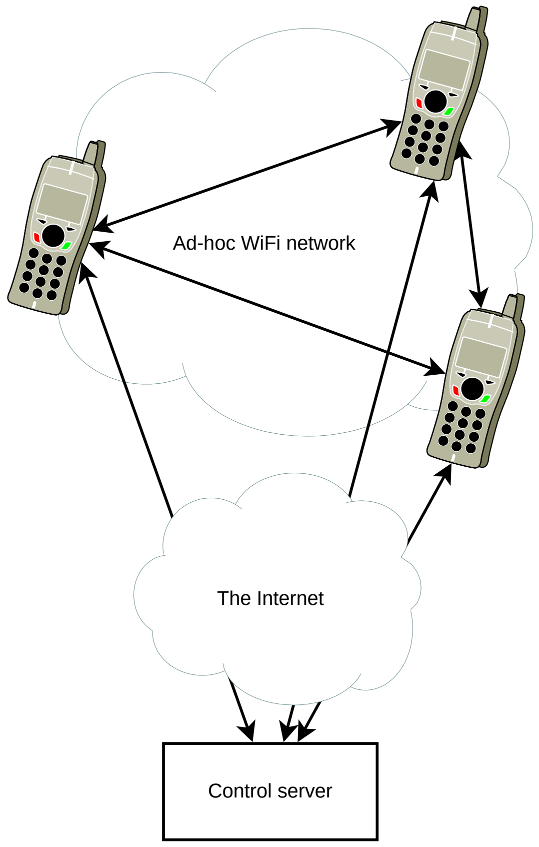


Figure 2: Communication between Magrathea devices is peer to peer. The control server is connected through the Internet.

5. Start up the Magrathea server to listen for connections from other devices through the WiFi ad-hoc network.
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 (infection has happened) 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 contagious agents and configuration changes and act accordingly.

Figure 3 shows an example Magrathea agent.

2.3 The implementation

The platform is implemented on Symbian Series 60 smart phones. The specific model that we use is the Nokia N80

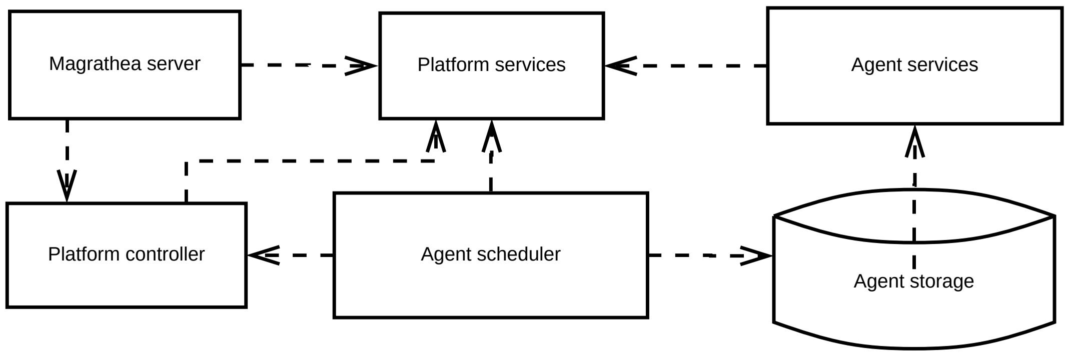


Figure 1: The dependencies of Magrathea platform components.

```
import magrathea, random

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

    magrathea.die_out()
```

Figure 3: Example of a Magrathea contagious agent. The agent propagates to the neighboring devices with the probability of 0.75 and dies out with the probability of 0.1.

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. Combining both means of communication would be ideal but remembering our main design principle, which is simplicity, we chose to use only WiFi. The on board camera is a nice addition to the sensors on the phone even though in the experiments in this paper we do not use the camera functionality.

The platform is implemented in Python and using Python extensions implemented in C++. This choice is justified by the rapid development phase that the use of Python makes possible. Python[1] 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. That 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.

[1]http://www.python.org

3 Experiments

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 most contagious agents possible spread within the test subjects. 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 time till you leave for home.
2. When you leave, turn off the phone and put it to 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 user.
- 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.

Table 1 summarizes the settings for the user experiment.

This setting resulted data that contains the daily spreading patterns of viruses and complete graph of the social network within the test subjects. This data can be analyzed and used for simulating other agent configurations.

Parameter	Value
Number of participants	10
Length of the experiment	8 working days
Length of the agent (virus) life cycle	1 day
Initial number of agents (viruses) per device	1
Contagiousness of agents (viruses)	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 1: Summary of the experimental setting for the user study.

Symbol	Explanation
G	A graph.
$\mathbf{A}$	Adjacency matrix of a graph.
$\mathbf{D}$	Connectivity matrix of a graph.
d_{ij}	Normalized connectivity between nodes i and j.
$\mathbf{S}$	System matrix of a graph.
$\mathbf{I}$	Identity matrix.
$d_G(i)$	Degree of node i.
N	Number of nodes (devices).
$\mathcal{K}$	Maximum degree of any node.
$\mathcal{P}$	Maximum number of nodes to propagate at one time point.
β_0	Agent birth rate. This is the probability that the agent infects neighboring node and it is coded in the model of the agent.
β	The effective agent birth rate.
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.
ϵ	Error probability in agent propagation.
τ	Epidemic threshold.

Table 2: The symbols and their explanations.

3.1 Graph theoretic preliminaries

In this section we discuss graph theoretic preliminaries that are related to our experiments. Table 2 summarizes the used notation.

In [8] a model for the evolution of an epidemic and epidemic threshold τ is presented for static network topologies and in [1] that model is further developed for arbitrary dynamic network topologies. Both of these models present a threshold condition for an epidemic in a network that is based on eigenvalue decomposition of a square matrix that describes the network. In the case of [8] that matrix is the adjacency matrix of the network and in the case of [1] it is something that the authors call the *system matrix* of the network. System matrix is also discussed in [8] but there it is obtained differently than in [1]. Nevertheless, the system matrix describes the dynamics of the system so that the diagonal elements relate to the node/virus/agent survival and the off-diagonal elements relate to the virus/agent propagation to neighboring nodes.

Let $\mathbf{A}$ be an adjacency matrix of an arbitrary graph G. For a static network the system matrix [8] is defined as:

$$\mathbf{S} = (1-\delta)\mathbf{I} + \beta\mathbf{A}. \qquad (1)$$

We assume that $\mathbf{A}$ is symmetrical, i.e. our graph is undirected and if the agent propagation path $i \rightarrow j$ is possible then the path $j \rightarrow i$ is also possible.

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.

Let $\mathbf{D}$ be the connectivity matrix of the graph G, which we have acquired by running and observing our agent platform. We define the system matrix

$$\mathbf{S}_G = (1-\delta)\mathbf{I} + \beta\mathbf{A}_G \otimes \mathbf{D}_G, \qquad (2)$$

where $\otimes$ denotes a component wise multiplication. Since our adjacency matrix is binary, the system matrix is

$$S_{ij} = \begin{cases} 1-\delta & \text{if } i=j \\ \beta d_{ij} & \text{if } i \neq j \end{cases}, \qquad (3)$$

where

$$\beta = \begin{cases} \beta_0(1-\epsilon)-\psi & \text{if } \mathcal{K} > \mathcal{P} \\ \beta_0(1-\epsilon) & \text{if } \mathcal{K} \leq \mathcal{P} \end{cases}, \qquad (4)$$

and ϵ is the error probability in transmission and

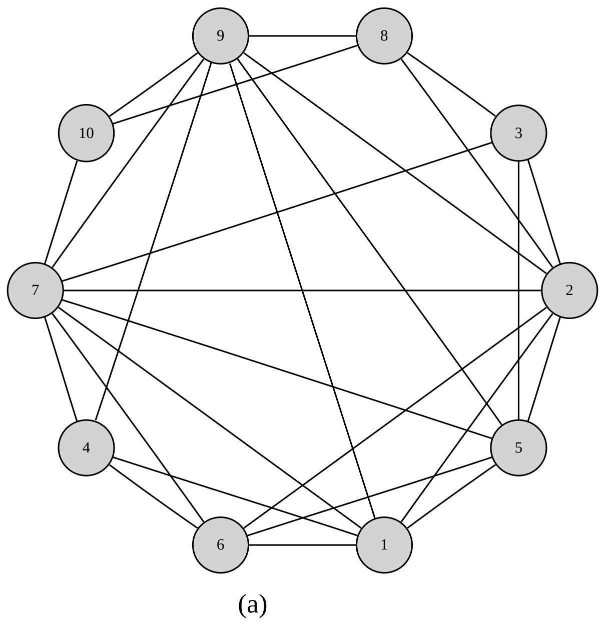
$$\psi = \frac{1}{N}\sum_{i:d_G(i)>\mathcal{P}} \frac{\mathcal{P}}{d_G(i)}. \qquad (5)$$

3.2 User experiment

The user 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 viral spreading patterns are defined mostly by the social graph within the test subjects we first discuss that.

In Figure 4a it can be seen that the graph is quite densely connected but most of the connections are weak, which is shown in Figure 4b, which shows only connections that are stronger than average. This is further illustrated by the basic statistical figures shown in Table 3 and by the visualization of the connectivity matrix of the graph shown in Figure 5. In Figure 4b we can see six cliques in the graph and hubs between the cliques. Four of these cliques are clearly separated by a hub node. When one considers the viral behavior of the agents it is these hubs that are most interesting.

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. One can easily see that the hub nodes in the network are the ones that are the most infected. 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 experimen-



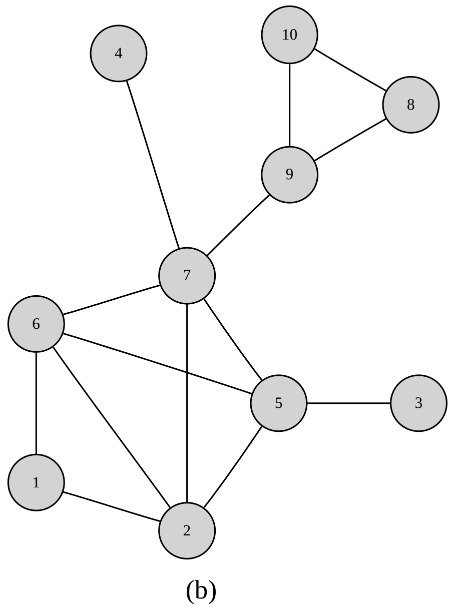


Figure 4: The resulting social graph of our experiment visualized showing all the connections between the nodes (a) and only strong connections (b). This way the hub-like behavior of certain nodes is more visible.

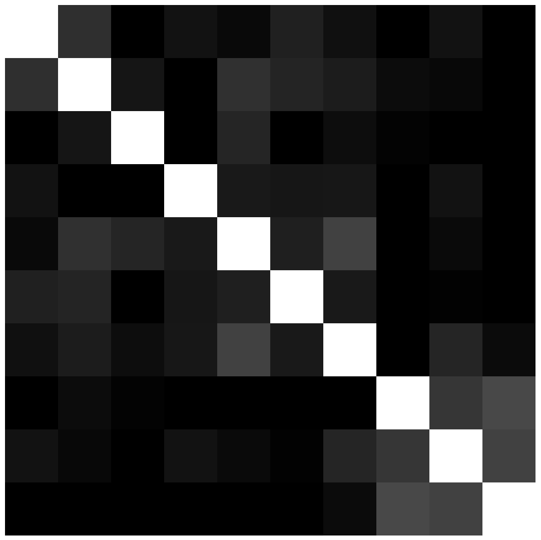


Figure 5: Visualized connectivity matrix for the nodes in the social graph of our user experiment.

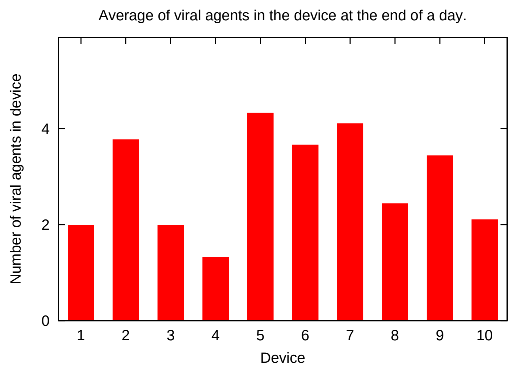


Figure 6: Average number of foreign viral agents in the device after one test day.

tal setting the maximum possible number of foreign agents in the system is 90 (see Table 1). 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.

3.3 Simulations

In this section we will briefly show how the acquired data can be further used for simulations. The spreading patterns of the viral agents are mostly defined by the ad-hoc network formed by the devices running the platform. The log of all activity in this network is readily available from our platform. It can be also observed in real time. There are two main options for running the simulations. We can use both the observed network structure and the observed agent activity as a basis for the simulations or we can use the network structure alone. Either way, one varies the parameters of Equations 3, 4 and

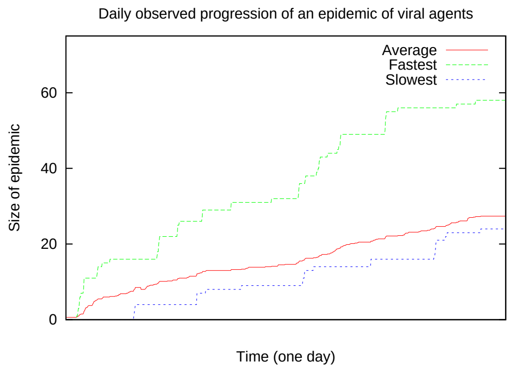


Figure 7: The daily progression of the spreading viral agents when the experiment is repeated. The y-axis shows the number of foreign viral agents in the whole system.

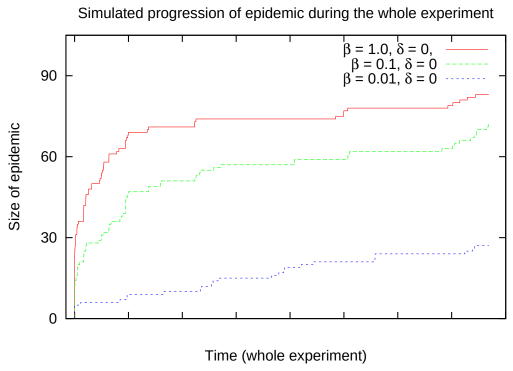


Figure 8: Size of the simulated epidemic with different parameters. Y-axis shows the number of foreign viral agents in the system as a function of time.

5 and observes how they effect the agent behavior. As an illustrative example we show in Figure 8 how the parameter β affects the spread of the agents in the observed network structure shown in Figure 4. It can be seen that as the network is connected and the virus death rate δ is zero the epidemic will always eventually spread to the whole network. Further simulations and analysis of the network are out of the scope of this paper.

4 Conclusions

How to use mobile ad-hoc networking, mobile agents and sensor networks for realistically simulate viral behavior? We have presented Magrathea: a mobile agent- and sensing platform that uses those technologies for simulating all kinds of networked behavior. We have presented the motivation, architecture and design choices as well as performed empirical and simulated studies to verify the the functionality of our platform.

Main advantages of our platform for viral simulations are:

- It combines accurate observations with realistic simulations best of the both worlds manner.
- It is possible to create very complex simulations in distributed manner as the application logic is coded in the agents.
- Simulations can be observed in real time and further refined later.

Even though we have concentrated in this paper only on the viral simulations, the scope of our platform is not limited to simulations only. Same ideas can be used e.g. for agent based sensor network maintenance systems, wireless sensing, surveillance etc. The presented platform is very simple yet powerful. In the future we plan to further develop the platform and use it for larger scale 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] 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.

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

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

[4] 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.

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

[6] 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.

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

[8] 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.