Simulation and Implementation of the Autonomic Service Mobility Framework

Janne Lahti*, Helena Rivas, Jyrki Huusko, and Ville Könönen

VTT
Kaitoväylä 1, P.O.BOX 1100, FIN-90571, Oulu, Finland
{janne.lahti,helena.rivas,jyrki.huusko,ville.kononen}@vtt.fi

Abstract. The increased traffic load, proliferation of network nodes and, in particular, wireless user devices, and the boom in user services and exponential growth of information stored in content distribution networks (CDNs) have brought new challenges for current networks. One major challenge has and continues to be efficient load balancing and information access. The topics have been well studied for wired networks for, for example, process load balancing in distributed computer networks with migration. However, the wireless networks and ubiquitous computing environments create new limitations and additional requirements to perform service or process migration. In this paper, we present a simulation case and proof-of-concept implementation for service mobility as a part of the BIONETS service evolution process with the aim of optimizing service penetration in a pervasive computing environment and balancing the load in the system caused by the high service utilization rate.

Keywords: Migration, mobility, service architecture, Internet, utility function.

1 Introduction

The current network environment can already be characterized by an extremely large number of networked devices possessing computing and communication capabilities. The trend is towards a ubiquitous network environment where the networked embedded devices integrate seamlessly into everyday use. At the same time, the networks are becoming increasingly information and service centric, with the conventional communication and service provisioning approaches starting to become ineffective as they fail to address the device and service heterogeneity, the huge number of nodes with consequent scalability, node and network mobility and device/network management well. From the communication point of view, the scalability and management issues, in particular, decimate the possibility of arranging a global always-connected network infrastructure. In other words, the global network starts to resemble an archipelago of network islands. In such a pervasive, decentralized computing environment, one of the key challenges is to arrange efficient load balancing, and guarantee service penetration and user access.

* Corresponding author.

E. Altman et al. (Eds.): Bionetics 2009, LNICST 39, pp. 65–76, 2010.

In order to tackle the problems of pervasive computing environments, the BIONETS project has proposed the architecture solution SerWorks, which incorporates service and network architectures and benefits from the biologically inspired communication paradigms [1, 2]. According to the BIONETS concept, services and service management are autonomic, and services evolve to adapt to the surrounding environment, just as living organisms evolve by natural selection.

One solution to improving system efficiency, balancing load in the system and optimizing service penetration in order to guarantee access for the majority of users is to utilize the concept of service mobility. Here, service mobility is described as the migration of the service or part of the service between nodes. The service mobility concept includes not only the migration procedure but also the selection of the service, possible replication or deprecation of the original service, migration of service implementation, its execution state and runtime data, adaptation to the target platform and handover of user associations.

Several similar solutions have been introduced on other biologically inspired communication platforms. Nakano and Suda in [3] and [4], for example, have introduced a network framework based on software agents to model the services. In addition, Suzuki and Suda have presented support for autonomic service management on [5] a middleware platform on top of a JAVA virtual machine. The main difference between BIONETS's approach and these solutions is the management of the services and implementation of the decision logics. The BIONETS platform utilizes so-called mediator entities for management and decision-making, e.g., for service migration. The mediators in BIONETS SerWorks are service external, located at each node, and one mediator can serve several individual service components where, as in the agent-based systems, each agent needs to implement algorithms for the decision logic itself. With BIONETS's "semi-centralized" system, the complexity of the services can be minimized by introducing the management functionalities outside the service and, with generic interfaces, it is possible to also provide evolution mechanisms for "legacy" services without re-implementing those as software agents.

In recent years, much work has also been carried out on load balancing using migration. In these cases, the migration is usually referred to as process migration for distributed computer systems [6] and the decision-making processes mainly consider only the CPU and memory load. Bearing in mind the pervasive computing environment and wireless ad-hoc type networks, the load balancing also needs to be taken into consideration, for example, user preferences and network conditions as well as system resources. We have defined a utility function for this to aid the migration decision, and we argue that with such an approach, the system is implementable, and by utilizing controlled service mobility, it is possible to achieve improved quality of service (QoS) also in a disconnected ad-hoc network environment.

The rest of the paper is arranged as follows. In Section 2, we present the system model for service mobility support in BIONETS. Section 3 concentrates on simulation modelling and results. Section 4 discusses proof-of-concept implementation for IP-based networks, and, finally, in Section 5 we draw the conclusions from the results and introduce our future work.

2 System Model

The ecosystem, similar to that presented in the BIONETS project [1], in which the Services live is illustrated in Figure 1. The decentralized and mobile ad-hoc network environment can be characterized by the temporally formed islands of devices that are dependent on the movement patterns of the user. On top of this networking infrastructure we are establishing a Mediator plane that provides autonomic control functionalities, network connections and other platform resources to services. Above the Mediator plane we have a user-centric service environment that allows for seamlessly integrating services of different devices in the user's surroundings to fulfil the user's needs. The atomic services (Service Cells) running on top of the platform can be any kind of small application providing certain functionality to other services or users. As the platform provides all the functionalities related to autonomic migration, the service only needs to implement the basic life-cycle controlling interfaces for starting, stopping, migrating, replicating and deprecating the service. With these interfaces the mediators can control the life cycle of the services located in the node.

The network topology is based on mobile ad-hoc connections between nodes. The devices cannot assume any backbone connection, and all the communication is done with the nodes inside the local connection range. The devices connected over local short-range links have the capability to locate, communicate and provide services for each other.

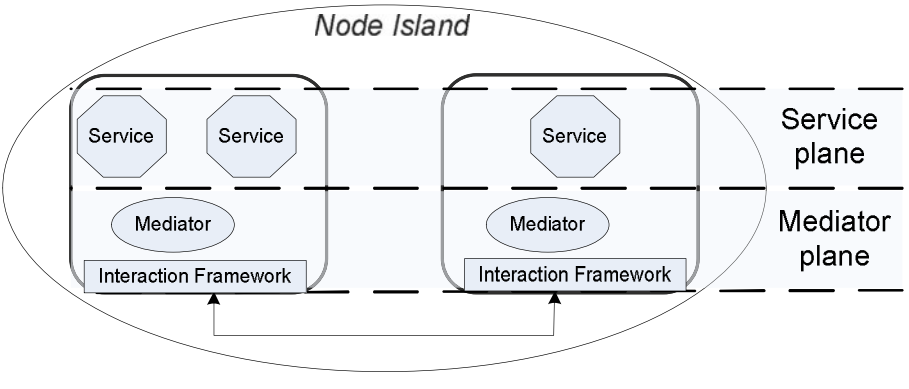


Fig. 1. Service and Mediator planes in U-nodes

The runtime architecture of our proposed service mobility framework is presented in Figure 2. The runtime platform operates on top of the nodes' (devices) operating system and provides the functionalities needed for service mobility. We have chosen a two-tier approach: we have a platform providing the autonomic functionalities for services on top of which we have the service execution layer where the services operate. The platform consists of the following functional blocks: *NodeMediator*, *InteractionFramework*, *ServiceDiscoveryMediator, ServiceCreationMediator*, *MigrationMediator, ServiceRequestHandlingMediator, ServiceMediator, ServiceExecutionRepository, ServiceBuffer and Node Information Repository.*

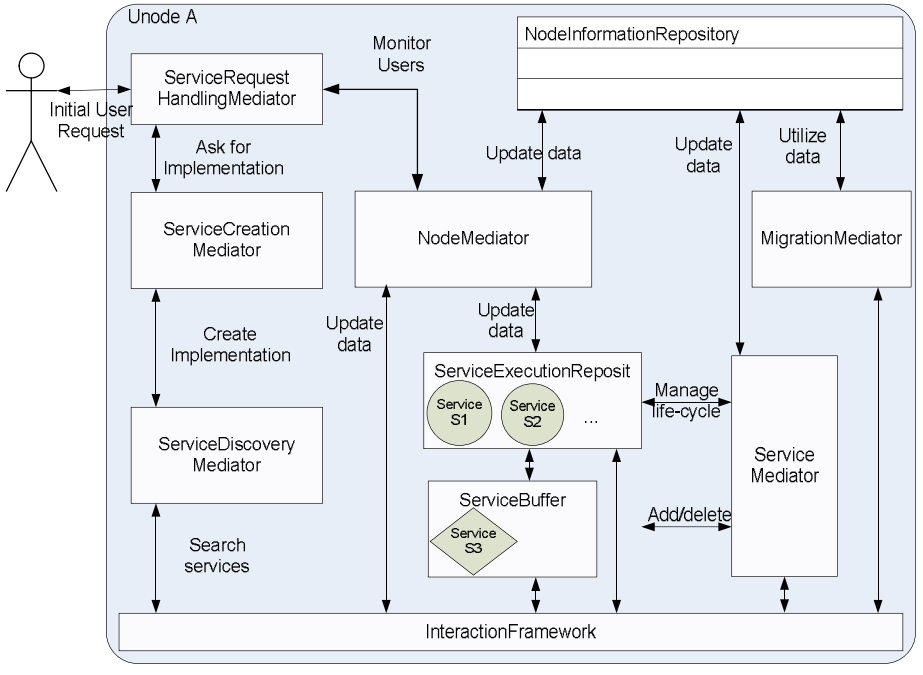


Fig. 2. The main components of BIONETS SerWorks Architecture needed to support service migration

Below, we give a brief description of the main role of each component.

- ***InteractionFramework:*** Handles all the communication between mediators in different nodes and inside the node.
- ***ServiceDiscoveryMediator:*** Discovers the services available in the node island, and makes the local services available to other nodes.
- ***ServiceCreationMediator:*** Builds the core for processing user requests.
- ***ServiceRequestHandlingMediator:*** Builds the interface to the user.
- ***ServiceMediator:*** Suspends service execution, migrates runtime service data and state to a destination node, initiates new service with restored state on the new destination and resumes application execution.
- ***MigrationMediator:*** Enables the movement of service application logic between different nodes.
- ***NodeMediator:*** Gathers and distributes the necessary information for the decision logic needed in the migration process, e.g., node fitness, CPU, battery, memory, speed or signal strength, etc.
- ***NodeInformationRepository:*** Keeps the information gathered by the Node Mediator.
- ***ServiceBuffer:*** Stores the services that are not currently executed in the node.
- ***ServiceExecutionRepository:*** Service execution environment. When an executing service is suspended, it is moved to the ServiceBuffer.

3 Simulation Model and Results

3.1 Simulation Model

The purpose of the simulation work is to model the service mobility in a BIONETS environment in which devices (nodes) move in a limited geographical area and dynamically form Service Islands with nearby nodes. Some of these nodes contain services offering limited sets of services (Service Cell A, B, C and D). In the model there are also entities (users or other services) connected to the nodes using one or all of those services.

The purpose of the simulations is to obtain knowledge about how the selected technologies and solutions would work in real implementations. We compare different decision-making mechanisms, migration models and overall service architecture variations to find out which ones are best suited for BIONETS-like environments. The purpose is to find solutions that ensure the best "Service penetration" for popular services with the minimum overhead cost. For example, we will look into whether the decision logic is more beneficial for implementation in the Service Cell or in the Mediator. The networking aspects related to the Service Migration are left outside the scope of the simulation. The simulation model does not implement the network level functionalities. The connections between different nodes are handled by the InteractionFramework (IF) component. The IFs of different nodes can "see" each other and form logical connections only when the nodes are in the same Node Island (NI), thus inside the given connection range.

The simulation model is implemented on top of the MASON simulation toolkit [7]. MASON is a fast discrete-event, multi-agent simulation library core in Java designed as the basis for large custom-purpose Java simulations. On top of MASON, we implemented the core BIONETS components presented in Section 2 (see Figure 2) that are required for service mobility purposes. The Mason toolkit executes all the components (U-nodes and Service Cells) at every step, with one step representing one second in real time.

In the simulation model, U-nodes move randomly in an X m*X m playground (City or Open field). The movement of nodes is based on the random waypoint mobility model [8] with an enhancement of "service gravity". This movement model provides semi-random movement with nodes picking a target point somewhere inside the circle with radius *r* and starting to move towards that point at every step. In some cases, we enhance the movement model with a "gravity factor", which makes the nodes less likely to leave an area that is providing services used by that node.

The U-nodes contain Service Cell components' and/or ServiceUser components. The ServiceUser mimics the real-world service users (user or other service) and is fixed to their host nodes (in certain cases the user may "change the device", i.e., move to another node). The services can (depending on the scenario) migrate freely between U-nodes. The U-nodes communicate with each other through the InteractionFramework (best effort messaging). The network part of BIONETS is not implemented in the simulation model. The U-nodes have a connection range and form NodeIslands with other nodes inside the range, enabling the InteractionFramework to exchange messages.

For reasons of simplicity, we do not have separate variables for each U-node's resource (e.g., CPU, memory and disk space), but model all these basic resources as general variables: Runtime Resource (*rr*) and Static Resource (*sr*). Each U-node has an amount of *x* resources *rr* and *sr*. The rr models the runtime resources consumed by services such as CPU and Memory load, and the sr models the static resource consumption such as disk space. Each hosted Service Cell consumes a certain amount of static resources from the host U-node when installed and frees it when it is uninstalled. In contrast, the runtime resources are consumed by Service Cells when executing a service response, and these are reset at each step. Similarly, the Service Cells have the variables Static Resource consumption (src) and Runtime Resource Consumption (rrc), modelling the amount of static resources required by the Service Cell when installed to a U-node and the amount of runtime resources consumed by the Service Cell per service response.

A more detailed view of simulation model implementation is presented in Appendix 1, which presents a sequence diagram of service migration in the Simulation Model.

3.2 Simulation Results

The primary purpose of a simulator model was to provide a (somewhat simplified) view of a model to be tested and studied. In the first phase of the research work we created a simple hypothesis: "It is possible to achieve improved service penetration in a disconnected ad-hoc network environment utilizing controlled service mobility." In order to evaluate the hypothesis, we envisaged a three-parted simulation study to set the base line for future more advances simulation studies. The simulation contained three different scenarios presenting different migration models: *Static services*, *Viral distribution* and *Controlled mobility*.

In the Static services scenario, the services laid completely static in their host nodes. The purpose of the scenario was to provide a reference point where, much like with the current systems, the services do not migrate and thus provide weak service penetration, but at same time require minimum resource consumption. In the Viral distribution scenario, when the U-node hosting service x comes within the connection range of another U-node, the node migrates the service to the new U-node. After the U-node is "infected" with the service x it also starts to distribute it to other U-nodes inside the connection range. The purpose of the scenario was to provide another opposite reference point where the services spread uncontrollably to new nodes resulting in very good service penetration, but with the cost of inflated resource consumption.

In the Controlled mobility scenario, the services migrate, applying the rules given by the MigrationMediator's decision process. The assumption was that the service penetration is better in this scenario than in the Static scenario, but with reasonable resource consumption. In order to evaluate the different scenarios, we run extensive simulations using the presented simulation model (Section 3.1). We set the square size L to 2 km, the communication range R to 50 m and the speed of the U-node V to 4 m/s. In each simulation run, we injected one hundred U-nodes into the environment in random locations. We also implemented four different Service Cell types (SC1, SC2, SC3, SC4), each providing simple calculation tasks (SC1 = add{x, y}, SC2 = subtract{x, y}, SC3 = multiply{x, y}, SC4 = divide{x, y}), and injected 10 of each

into the random U-nodes. We also randomly injected 20 Users into the nodes. The Users were set to create queries to one (random) service type at random every 5 steps (seconds). We measured the number of successful replies (SRN) to the User on average for the sent service request in each simulation run, thus giving a reasonable presentation of the service penetration among U-nodes. We also measured the average resource consumptions for static resources (ASR) and runtime resources (ARR). We scaled all the numerical results to [0.1] for better comparisons.

We ran the simulation 20 times with each simulation setting. Each simulation run lasted 10000 steps (seconds). From histograms presented in Figure 3 we can see how the number of successful replies varied in different scenarios. Figures 3a, 3b and 3c present the distribution of measured SRN values for separate simulation runs in each case. As expected, the best service penetration was in the Viral distribution scenario in which the Users received responses to sent service queries on average about 49.6% of the time. In the Static scenario, the percentage was significantly lower at 10.4%. In the Controlled migration scenario, in which the services migrate according to predefined rules, taking account of the resource load, the percentage was 29.8%.

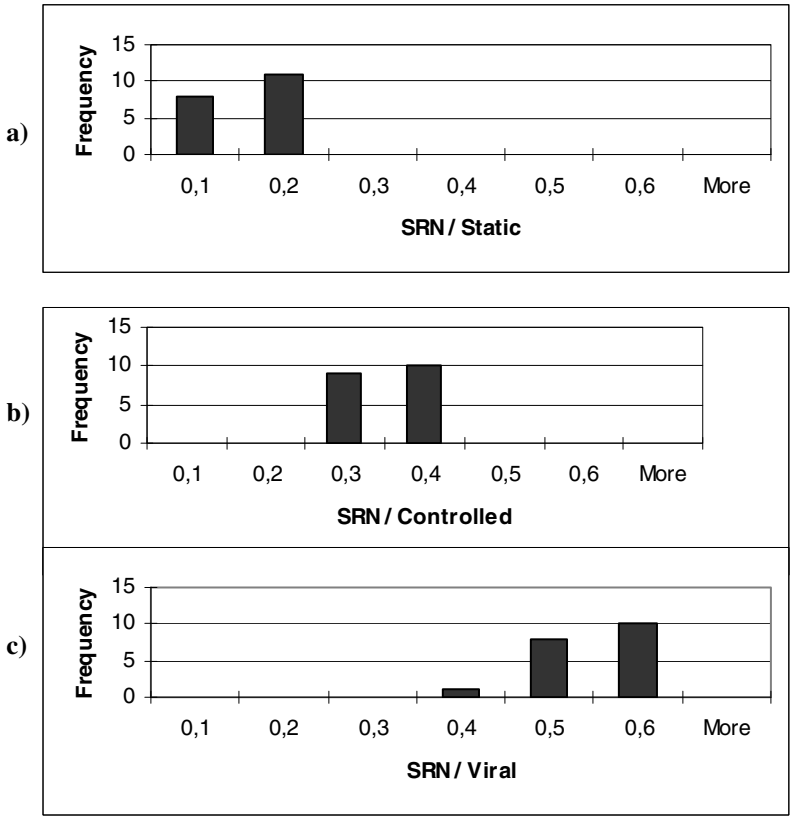


Fig. 3. Distribution of SRN values for each case

The average resource consumptions for same simulation runs are illustrated in Figure 4 below. As we can see, the resource consumption is somewhat reversed compared to the SRN values. The AVG variable presents the average of both ASR and ARR values. We can compare the "goodness" of each scenario by calculating a value Q = SNR / AVG for each scenario (bigger values are better). For the Static services scenario, we get Q = 0.846. For the Viral distribution scenario, we get Q = 0.728. Finally, for Controlled mobility, we get Q = 0.937, which supports the original hypothesis that we can achieve better service penetration with reasonable resource consumption with controlled service mobility.

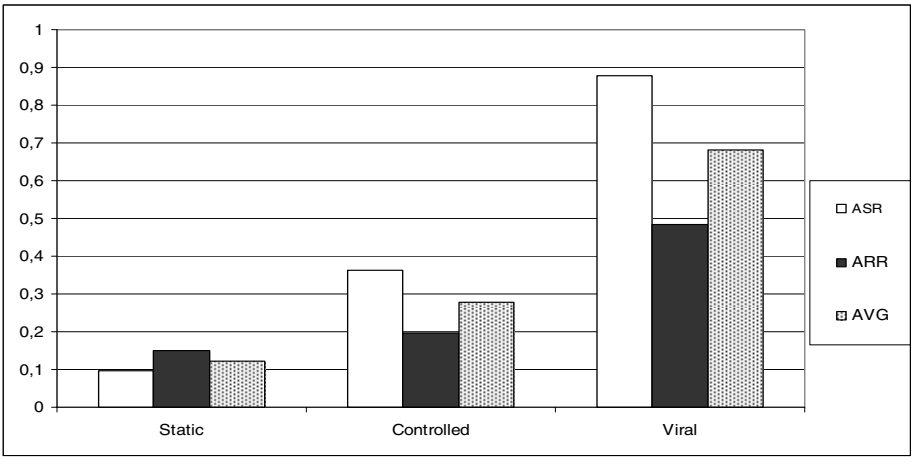


Fig. 4. Resource consumption values for different cases.

4 Prototype

The prototype model implements the basic BIONETS components required for service mobility purposes. The prototype implementation was developed on top of the J2EE platform and the services were implemented as web services. PC/Linux was selected as the implementation platform mainly because of its flexibility in collecting network information. It also provides easier access to system parameters required for decision-making. SIGAR API [9] was used to access the system information required for decision-making. This SIGAR (System Information Gatherer and Reporter) is a cross-platform, cross-language library and command-line tool for accessing operating system and hardware-level information in Java, Perl and .NET technologies. The SIGAR API enables the developed platform to also run on top of Windows OS. Glassfish [10] was chosen as the web server due to its programmatic Application Server Management Extensions (AMX) [11] API. AMX is a superset of the JSR 77 interfaces built on JMX, which simplifies and smoothes out the management and monitoring process. The information gathered with SIGAR (system information, e.g., memory usage, CPU consumption, networking, etc.) and AMX (web-services

information: response times, throughput, total number of requests, faults, etc.) is collected in the Node Information Repository.

In the following, we present simplified decision logic that we implemented for real devices based on the research work and simulation results. We want to emphasize that the parameterization and utility functions here are only preliminary, though they are general enough to cover many real world tasks and will be extended later.

Each time a node makes its decision, it has a fixed set of options. The fitness utility value is computed for each possible option using the utility functions (see below). An option with the highest utility value, that is the best fit for this purpose, is then selected for execution. In a similar way to the simulation model, we implemented four simple integer calculators (add, subtract, multiply and divide) as services, which migrate over the developed platform running inside real devices (Linux OS laptops).

Below is an example of a simple utility function for calculating the fitness of Service Cells based on the memory usage and operation-related CPU. A detailed explanation of the migration process implemented in the prototype is presented in Appendix 2.

Parameters:

Number of requests (SC_{NoR}), size of service: memory allocation required (SC_{MA}), operation-related CPU (SC_{CPU}) and memory usage (SC_{MU}), user evaluation (SC_{UE}), battery (U_B) and free memory (U_{FM}).

Utility function for SC:

c(SC_{NoR}, SC_{MA}, SC_{CPU}, SC_{MU}, SC_{UE}, U_B, U_{CPU}, U_{FM}), $c \in L$

c is a function of the parameters of a service cell and the platform on which the service is running. L is a partially ordered set, e.g., the unit interval [0,1].

Example utility function:

$$c(SC_{CPU}, SC_{MU}) = \alpha(U_{CPU} - SC_{CPU}) + (1-\alpha)(U_{FM} - SC_{MU})$$

α is a parameter in the unit interval [0,1], which weighs the importance of memory and CPU usage.

5 Conclusion and Future Work

In this paper, we presented the service mobility framework in the BIONETS concept together with the results of the service level simulations with a MASON simulator and proof-of-concept demonstration for an IP network with PC hardware. The framework can be utilized for balancing the service load and resources, and to optimize the service penetration in and between the network islands, when the connection is not always guaranteed. We discussed basic principles of the service ecosystem and presented a system model for service mobility support in BIONETS. We also presented a simulation model for service migration and a simplified prototype implementation. Finally, we presented preliminary simulation results. The main results confirmed that the controlled service mobility can provide better service penetration with reasonable resource and energy consumption.

The next step in our research is to better optimize the service mobility management by applying more extensively, for example, game theory and learning capabilities to the decision-making. One possible approach is also to utilize more efficient mobility triggering and define the triggering events needed for the mobility management as presented in, for example, [12] for node mobility. In addition, we are aiming to continue the prototyping activities by applying the service mobility framework to, for example, energy-aware cloud computing and distributed systems, implementing the more complex utility functions for decision logics in order to also better the network conditions and provide more extensive results from the hardware implementation testing.

Acknowledgments

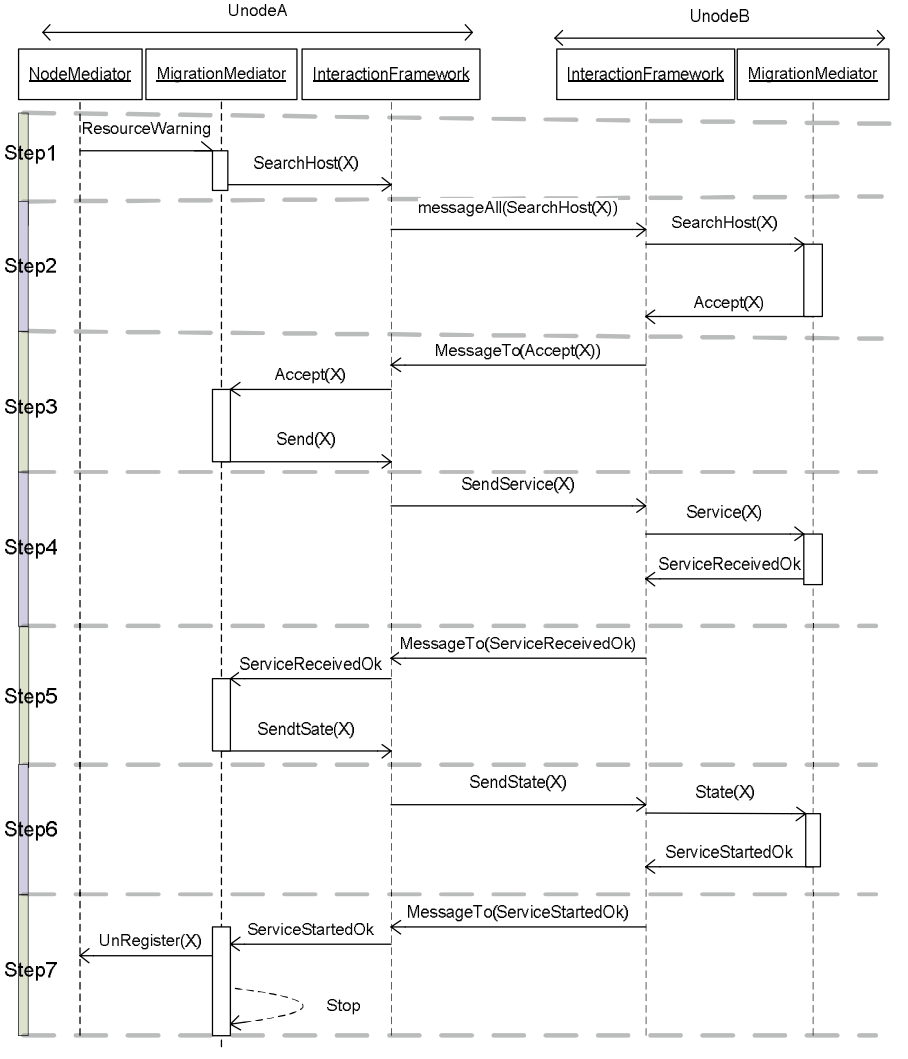
The authors would like to thank all their colleagues in the EU IST FP6 BIONETS project. The research was partially funded by the European Commission.

References

1. Chlamtac, I., Miorandi, D., Steglich, S., Radusch, I., Linner, D., Huusko, J., Lahti, J.: BIONETS: Bio-Inspired Principles for Service Provisioning in Pervasive Computing Environments. In: Di Nitto, E., Sassen, A.M., Traverso, P., Zwegers, A. (eds.) At your service: service engineering in the Information Society Technologies Program. MIT Press, Cambridge (2008)
2. Miorandi, D., Huusko, J., De Pellegrini, F., Pfeffer, H., Linner, D., Moiso, C., Schreckling, D.: D1.1.3/3.1.3 Serworks architecture v1.0. BIONETS Deliverable, D1.1.3/3.1.3 (2008)
3. Nakano, T., Suda, T.: Adaptive and Evolvable Network Services. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 151–162. Springer, Heidelberg (2004)
4. Nakano, T., Suda, T.: Self-organizing network services with evolutionary adaptation. IEEE Trans. on Neural Networks (2005)
5. Suzuki, J., Suda, T.: A middleware platform for a biologically inspired network architecture supporting autonomous and adaptive applications. IEEE Journal on Selected Areas in Communications 23(2), 249–260 (2005)
6. Milojičić, D.S., Douglis, F., Paindaveine, Y., Wheeler, R., Zhou, S.: Process migration. ACM Computing Surveys (CSUR) 32(3), 241–299 (2000)
7. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K.: MASON: A New Multi-Agent Simulation Toolkit. In: Proceedings of the 2004 SwarmFest Workshop (2004)
8. Yoon, J., Liu, M., Noble, B.: Sound mobility models. In: Proc. of ACM MobiCom, San Diego, CA (2003)
9. SIGAR API, `http://www.hyperic.com/products/sigar.html`
10. GlassFish, `https://glassfish.dev.java.net`
11. Appserver Management Extensions, `https://glassfish.dev.java.net/javaee5/amx`
12. Mäkelä, J., Pentikousis, K., Majanen, M., Huusko, J.: Trigger management and mobile node cooperation. In: Katz, M., Fitzek, F.H.P. (eds.) Cognitive Wireless Networks: Concepts, Methodologies and Visions – Inspiring the Age of Enlightenment of Wireless Communications, pp. 199–211. Springer, Heidelberg (2007)

Appendix 1: UML Sequence Diagram of Service Migration Implementation in the Simulation Model

Bellow we present the UML diagram showing the migration process implemented in the simulation implementation. In this figure, we show the process sequence for a scenario where the NodeMediator of the U-Node triggers a service migration of Service Cell X to a new host. The MigrationMediator offers the X to U-Node B (which accepts it) and then migrates the Service Cell to a U-node B. It also shows the interaction between *Mediators* and the exchange of messages between them.



Appendix 2: UML Diagram for a Prototype Implementation

Bellow we present the UML diagram showing the migration process implemented in the prototype implementation. In this figure, we show the *requestMigration* of U-Node A to U-Node B to migrate its *ServiceX* to U-node A. It also shows the interaction between *Mediators* and the exchange of messages between them.