Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 35 (2014) 241 – 250

Procedia Computer Science

18th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems - KES2014

Context-aware multi-agent system in machine-to-machine communication

Hrvoje Maracic, Tihana Miskovic, Mario Kusek*, Ignac Lovrek

University of Zagreb, Faculty of Electrical Engineering and Computing, Unska 3, 10000 Zagreb, Croatia

Abstract

Machine-to-Machine (M2M) communication refers to communication among machines without or with only limited human intervention. The paper deals with context-awareness in M2M communication and proposes rich presence information as a concept to address context information in a multi-agent system supporting networking of machines. Rich Presence Information Data (RPID) extensions suitable for M2M are introduced into Context-aware Mobile Agent Network model. Case study describing context information related to energy consumption and energy saving is presented. Results based on simulations and measurements in a laboratory test-bed are discussed.

Peer-review under responsibility of KES International.

Keywords: machine-to-machine communication; presence; context; agent; multi-agent system

1. Introduction

Machine-to-Machine (M2M) communication is communication between machines that do not necessarily need any direct human intervention. Machines are devices in the range from sensors and embedded processors to computers. Some devices are powered by batteries and need to run for a long period of time without charging or battery change. Consequently, machines must efficiently use energy by turning off communication or stopping operation when it is not necessary. To address the power efficiency challenge, it is necessary to have available the information about a machine's state and context information describing its environmental and operational circumstances.

This paper introduces extensions of the Rich Presence Information Data (RPID)[1] as a tool to address context information in M2M communication and software agents as a way to enable decision making upon collecting and using available context information. Specific elements and attributes are necessary for describing "machines" and context related to them. In the paper special attention is paid to context information suitable for defining energy consumption and providing energy-efficient communication. The case study includes the energy model and the simulation of en-

* Corresponding author. Tel.: +385-1-6129-801; fax: +385-1-6129-832.
E-mail address: mario.kusek@fer.hr

Peer-review under responsibility of KES International.
doi:10.1016/j.procs.2014.08.104

ergy consumption, as well as influence of rich presence information on energy efficiency. Results of simulations and measurements in a laboratory environment are presented.

The rest of the paper is structured as follows: in section 2, rich presence information for machines is defined; section 3 extends the mobile agent network model with context-awareness; section 4 elaborates the energy model and mechanisms that are included in simulations; section 5 provides an analysis of network traffic and its reduction gained from using rich presence information and section 6 concludes the paper.

2. Context based on rich presence information for machines

An M2M system consists of M2M devices, M2M gateways and M2M servers, as shown in Fig. 1. In this hierarchy a number of M2M devices are connected to an M2M gateway, through which they communicate with the M2M server. A number of M2M gateways are connected to one M2M server.

In current research context information is modeled based on an application domain[2,3,4,5] and it is not standardized. In this paper context information is based on the rich presence information standard[1]. Rich presence information is intended to be used for humans, but in this paper it is extended to machines as well. The advantage of this approach is: (i) the context for humans and machines is used in the same way which gives us a unified approach; (ii) part of context information is the same for humans and machines (i.e. environment) and can be collected by machines (i.e. smart phone); and (iii) this approach has a unified format based on XML and current standards for presence information.

Presence information represents information indicating the willingness or ability of a user to communicate with other users in a communication network. In order to use presence information when communicating, a presence service is required. A presence service mediates between users that provide presence information and users that act upon presence information. A basic model for presence and instant messaging has introduced the notion of "presentity", a communicating entity that provides presence information for notifying its state and changes in its state. The roles of the presence service are to receive presence information from presentities, store it, and distribute it towards entities that will use it, called "watchers"[6].

Presence information can be provided explicitly by the entity or derived automatically from its situational information or activity. The initial presence format encoded in XML is defined in the Presence Information Data Format (PIDF)[7]. The PIDF defines the following elements: status, contact and timestamp of change. The PIDF is extended to the Rich Presence Information Data (RPID)[1] in order to introduce elements containing context-related information for an entity defined as a person, service or device that is participating in communication. Some of the RPID elements describe persons, some services (standard refers to them as tuple), and some devices, as shown in Table 1 and marked with "+".

Elements <activities>, <mood>, <place–is>, <place–type>, <sphere> and <time–offset> relate only to a person, the role of the person and the place the person is located. Elements <privacy> and <status–icon> define if the communication service is likely to be observable by others and the current status of the person or service. Service–specific elements are <device ID>, <relationship> and <service–class> specifying a device that contributes to the service, relationships to the users and the way how the service is delivered. Only two elements, <class> and <user–

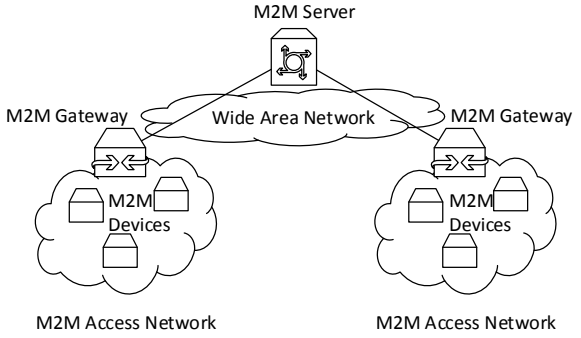


Fig. 1. M2M architecture

Table 1. RPID elements used for describing person, service and device or object

Element	attributes from/until	person	service (tuple)	device	ctx_j
<status>			+		ctx_1
<contact>			+		ctx_2
<activities>	+	+			
<class>		+	+	+	ctx_3
<deviceID>			+	+	ctx_4
<mood>	+	+			
<place-is>*	+	+	+*	+*	ctx_5
<place-type>	+	+			
<privacy>	+	+	+		ctx_6
<relationship>			+		ctx_7
<service-class>			+		ctx_8
<sphere>	+	+			
<status-icon>	+	+	+		ctx_9
<time-offset>	+	+			
<user-input>		+	+	+	ctx_{10}
<load>*			+	+	ctx_{11}
<battery-level>*				+	ctx_{12}
<charging-type>*				+	ctx_{13}
<availability>*	+		+	+	ctx_{14}

input> are common for all the three entities: <class> for grouping of similar persons, devices, or services; <user–input> for recording the user–input or usage state of the service or device, based on human user input.

These elements are not sufficient for devices acting as machines in M2M communication. The proposed extension of the RPID is based on the following approach: a) to extend the existing elements with M2M-specific features, and b) to introduce new elements specific for M2M. These elements are labeled with "*" in Table 1. The column ctx_j is reference to the formal model extension in section 3.

Machines contain a set of elements where the most important are contact, identifier and status. The contact is used for communication (i.e. IP address) and the identifier (i.e. MAC address) to uniquely identify the machine in a network. The status contains information about the activity status (e.g. hibernate, off, on), as well as information collected by various sensors. For example, the status can contain elements that describe the collected contextual information, i.e., an application can receive information from a temperature sensor that records the measured temperature in the <temperature> element contained in the <status> element.

The element <relationship> indicates how a presentity relates to another entity and the element <service-class> describes the type of communication (e.g. electronic, delivery or live).

Some elements may be qualified with the "from" and "until" attributes to describe the absolute time when the element assumed this value and the absolute time until which this element is expected to be valid. All described elements are optional within RPID documents.

The basic problem with energy consumption in M2M communication is that machines consume a large amount of energy when they are available and connected to the network. In order to conserve energy, devices go to some kind of sleep or hibernation and wake up at a specified point in time[8]. If some other machine wants to communicate, both machines need to be available at the same time. Rich presence information can be exchanged for that purpose.

3. Context-aware mobile agent network model

A mobile agent network is a formal model of a multi-agent system placed in a network of nodes. The nodes host agents allowing them to execute services, communicate and migrate from one node to another. It is introduced in[9].

Performance issues related to multi-agent systems are analysed in [10]. Context-aware extensions of the mobile agent network are introduced in [11,12] and represented by the following quadruple:

$CA - MAN = \{A, S, N, C\}$

where A represents a set of agents co-operating and communicating in an environment defined by S and N. S denotes a set of processing nodes in which the agents perform services, and N a network that connects the processing nodes and allows agent communication and agent mobility. The network is defined as an undirected graph, $N = (H, E)$ where H is a set of network elements (hosts) that in this paper corresponds to a set of nodes S and E represents a set of links that connect the network elements. Each processing node s_i corresponds to one network element h_j but each h_j does not have to correspond to some s_i (e.g. the network switch is a network element, but it is not a processing node).

C represents a set of context type of data handled by agents. The context type data is derived from rich presence information for machines from section 2. Each M2M entity corresponds to one agent in $CA - MAN$. This means that there are two types of agents: the M2M device agent and the M2M gateway agent. The M2M device agent only communicates to one M2M gateway agent and the M2M gateway agent is responsible for collecting data from all M2M device agents that communicate to it and send that data to the M2M server. The context C of each M2M entity is defined as $C = \{ctx_1, ctx_2, \ldots, ctx_j, \ldots, ctx_{14}\}$ where ctx_j is a representation of one rich presence information element, as stated in Table 1. In this paper we have used experiments only with availability information combined with "until" attributes. Context information ctx_{14} is defined as $ctx_{14} = \{c_{availability}, c_{from}, c_{until}, c_w\}$. $c_{availability}$ is the state of an entity and can have two values: *available* (corresponds to state *on*), and *not − available* (corresponds to state *hibernate* or off). c_{from} is the absolute time when an entity will change state. c_{until} is the absolute time until the entity will be in the state $c_{availability}$. c_w is a set of watchers that this entity will inform when it changes state (e.g. the M2M gateway agent is the watcher of the M2M device agent and vice versa). The information sent to the watcher is $c_{availability}$, c_{from} and c_{until}. In the special case, when an entity is always active, context information is $c_{availability} = available$, $c_{from} = 0$ and $c_{until} = \infty$.

The functionality of A is defined by a set of elementary services $ES = \{es_1, es_2, \ldots, es_j, \ldots, es_{nes}\}$. Each elementary service can be provided by multiple competing or collaborating agents. A single agent can support multiple elementary services. A set of services that we have defined are: es_1 executing task (e.g. smart meter measures temperature), es_2 sending measurement, es_3 sending context information, es_4 collecting (receiving and saving) measurement, es_5 collecting context information, es_6 creating measurement report and es_7 sending report to server. Services 1 to 3 are the responsibility of an M2M device agent, and services 4 to 7 are executed by an M2M gateway agent. Agents in *on* state consume much more energy then in *hibernate* state. The constraint is that the agent in *hibernate* state cannot execute any service. M2M device agents usually change their state over periods of time. The M2M device agent informs their watchers (in this case it is the M2M gateway agent) by sending availability context information before going to *hibernate* state and after returning to *on* state. For a certain amount of time the M2M device agent is in *on* state and after that it goes to *hibernate* for some time. The M2M gateway agent needs to be in *on* state if it wants to communicate with other agents as well. The M2M device agents can change the phase of periods in order to conserve energy of the M2M gateway agent. This decision is based on context information that the M2M device agent receives from the M2M gateway agent and the algorithm implemented in the M2M device agent.

4. Simulation of energy consumption

For experiments, Waspmote Devices are used as M2M devices and M2M gateway [13]. On each machine (M2M device or gateway) there is software running that is implemented as an agent from CA-MAN.

4.1. Energy model of waspmote devices

Devices used in the experiments are Waspmote v1.1. The data sheet [14] and the technical guide [13] state that a device can be in one of the 4 states: *on* ($I_{on} = 15mA$), *sleep* ($I_s = 55\mu A$), *deep sleep* ($I_{ds} = 55\mu A$) and *hibernate* ($I_h = 0.06\mu A$). In this paper we have used only *on* and *hibernate* states.

The device is consuming different amounts of electrical current in different states, and the transition from one state to the other takes a different amount of time. Since we want to minimize energy consumption, we have used the

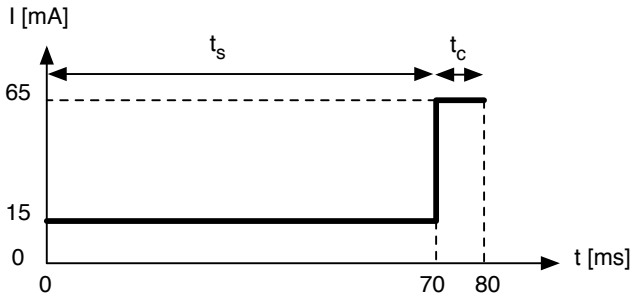


Fig. 2. Waspmote current consumption during wakeup

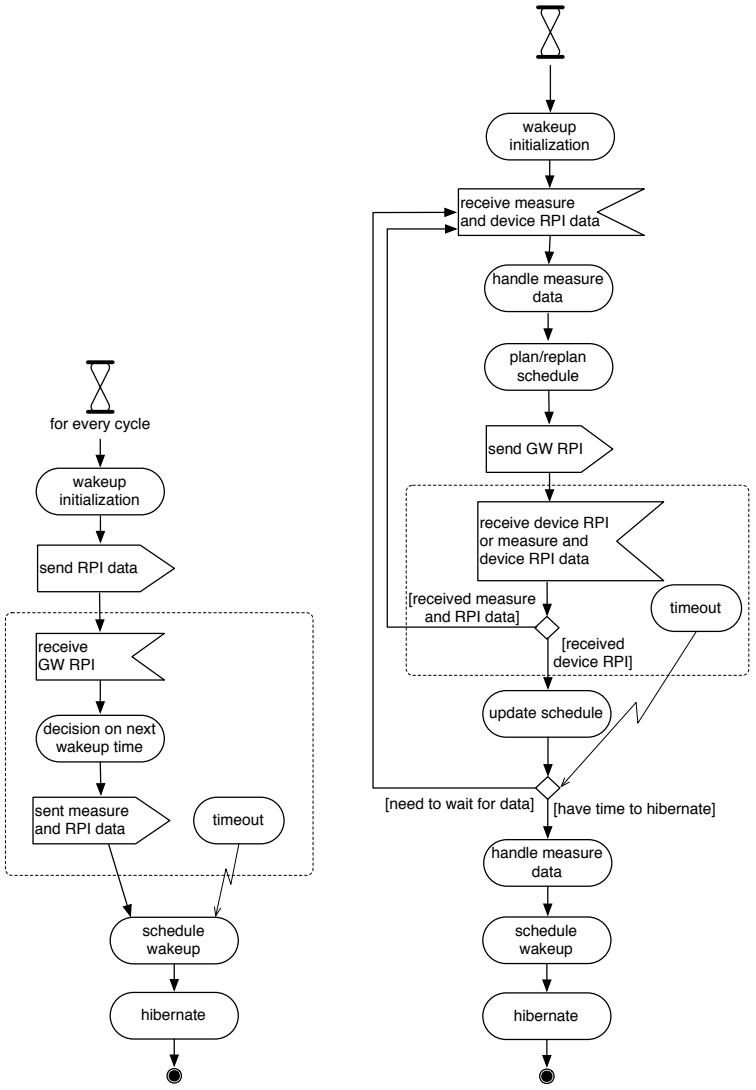
hibernate state because it has the lowest energy consumption. The wakeup time is $t_w = t_s + t_c$ where t_s is startup time and t_c is communication startup time. When the device is connected and powered up it starts the bootloader; there is a waiting time of $62.5ms$ before beginning the first instruction. This time is used to start loading a new compiled program [13]. When the device is in the *hibernate* state the microcontroller (Atmel ATmega 1281) is reset by an external real-time clock. The specification [15] explains that that kind of state is the power-down processor mode. When waking up from the power-down processor mode, there is a delay until the wake-up becomes effective. This procedure allows the clock to restart and become stable after having been stopped. The worst case scenario is a delay of 14 clocks + $69ms$. 14 clock cycles at $8MHz$ is $1,75\mu s$. For the purpose of the simulation we have defined $t_s = 70ms$.

After starting up the communication, initialization is started; there is no defined boot time in the specification [16] but there is a wakeup time from the *hibernate* state and it is $10.2ms$. From that we have defined $t_c = 10ms$. The consumption of the communication module is $I_{receive} = 45mA$ when receiving and $I_{transmit} = 50mA$ when transmitting [17]. We have chosen the worst case $I_c = 50mA$. The operating voltage of the Waspmote is $U = 3.3V$.

The diagram in Fig. 2 shows the electrical current consumption during the wakeup period. During the first period (t_s) the consumed energy is $E = UIt$ from that follows $E_s = 3.3V \times 15mA \times 70ms = 3.465mJ$. During the next period (t_c) when the communication module is initialized, the consumed energy is $E_c = 3.3V \times 65mA \times 10ms = 2.145mJ$. The total energy consumption during the wakeup period is $E_w = E_s + E_c = 5.61mJ$. From that the average wakeup power consumption is calculated $P_w = E/t = 70.13mW$. While running, the power consumption is $P_r = U(I_{on} + I_c) = 3.3V \times 65mA = 214.5mW$

4.2. Negotiation mechanism

Energy efficient wake-up scheduling will maximize the lifetime of the whole network [18]. Fig. 3 shows activity diagrams of the mechanism used in the M2M device and gateway agents. The M2M device agent activity is in Fig. 3a. It is executed in each cycle. First the device wakes up and performs the initialization as explained in section 4.1. After that, the M2M device agent starts measuring some data, which depends on the application (e.g. temperature measuring). The measurement process has its duration which can vary from milliseconds to minutes, depending on what is measured. After measuring, the M2M device agent sends the measured data together with presence information about its availability $c_{availablity} = available$ to the M2M gateway agent. Meanwhile, the M2M gateway agent needs to wakeup and start receiving the data from M2M device agents. When the M2M gateway agent receives the first data from an M2M device agent, it processes measured data and informs the M2M device agent of its own presence information which includes periods when the M2M gateway agent will be in state *on*. Based on that presence information, the M2M device agent decides whether it will try to adjust its wakeup time by moving closer to those periods, or stay on its ideal schedule or for some other reason select a different wakeup time. The M2M device agent makes this decision based on its tolerance parameter which is explained in the following example. In time $t_1 = 100s$ the M2M device agent needs to decide when to wakeup next time. If the period at which the M2M device needs to do measurement is $t_{period} = 10s$ and the tolerance is $t_{tolerance} = 1s$, the next ideal time to wakeup is $t_2 = 110s$. That is the ideal schedule. But if the M2M device agent has the presence information from the M2M gateway agent and in that information is the scheduled period $ctx_{14} = \{available, 108s, 108.1s, ...\}$, then the M2M device agent can



a) M2M device agent b) M2M gateway agent

Fig. 3. M2M device/gateway agent activity diagram

decide to set the wakeup time to be closer to that. The best next wakeup time, regarding energy consumption, is $108.1s$ because in that case the M2M gateway will not go to *hibernate* and waste energy to wakeup. But this is not possible because the M2M device needs to periodically measure data and in this case it will measure data $8.1s$ after the last measurement. The tolerance parameter defines how much is the tolerated difference in the periods of measurements. In this case it is $1s$ and the M2M device agent can schedule the next wakeup time in $109s$. In the following periods it can further adjust its wakeup time and after a number of periods it will schedule the wakeup time in a way that is best for energy consumption.

After making a decision, the M2M device agent informs the M2M gateway agent of this selected time by sending its presence information (ctx_{14}) and then it goes to hibernate. The M2M gateway agent updates its schedule based on the received presence information from the M2M device agent. The M2M gateway agent then decides whether to hibernate or whether it will stay *on* and *available* to wait for the next M2M device agent. This decision is based

on the $c_{availability}$ information of the next scheduled M2M device agent and the M2M gateway agent wakeup period. If the time to the next M2M device agent to be *available* is less than the gateway's wakeup period, it will not go to *hibernate* before receiving the data from the next M2M device agent. The M2M gateway agent will always be in *on* state and *available* according to the schedule, and its energy consumption is determined by the M2M device agents' ability to fit into the M2M gateway's scheduled availability periods.

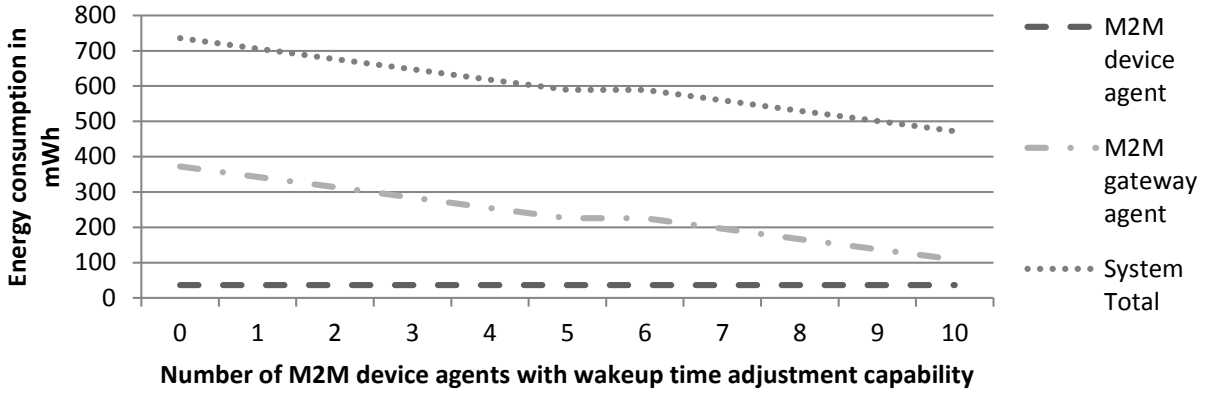


Fig. 4. 24 hour Waspmote energy consumption

4.3. Simulation

The simulated system consisted of a constant number of M2M device agents, ten, connected to a single M2M gateway agent. As defined earlier in this work, we assumed an 80 ms Waspmote wakeup time and a 30 ms measurement delay. The link was 250 kbps. Although all the M2M device agents and the M2M gateway agent were using rich presence information, some M2M device agents have tolerance set to $t_{tolerance} = 500ms$ while others have $t_{tolerance} = 0s$. The consequence, in M2M devices that have tolerance set to $t_{tolerance} = 0s$, is that they can not adjust the wakeup time and they always wakeup at their ideal schedule.

Shown in Fig. 4 is the energy consumption of the system during simulated 24 hours relative to the ratio of devices with and without the wakeup time adjustment capability. By shifting their available time for the tolerance value, M2M device agents with adjustment get inside the M2M gateway's already scheduled available periods (or close enough to them) and allow for more efficient operation of the gateway by avoiding unproductive wakeup time. The M2M gateway's energy consumption decreases significantly, to less than a third. But as more and more M2M device agents do not have the wakeup adjustment capability, more and more wakeup overhead is introduced on the M2M gateway, increasing its energy consumption by almost eight times. Despite the M2M gateway being a single device, compared to ten M2M device agents, we can see that optimized scheduling can reduce consumption of the whole system by 35%.

5. Simulated and real environment network traffic comparison

In this section, the concept of energy conservation and efficiency of the *on* state from the previous section is further improved, since energy consumption of machines is most influenced by communication. Therefore, reduced network traffic results in better energy efficiency. In paper [12] the generated network traffic was measured as an indirect measure of energy efficient communication. That was accomplished in a simulated environment. A similar experiment is repeated in a real environment to compare the results. The purpose of the experiment is to determine the influence of Rich Presence Information (RPI) on energy efficiency in M2M communication. To determine the effect of RPI on the network traffic generated, experiments are performed in two situations, without RPI and with this information available to M2M gateways. In these experiments RPI is the availability information of the M2M devices (ctx_{14}), which M2M gateways use for defining the time when to contact M2M devices. There are three types of M2M entities

in this experiment: the M2M server, the M2M gateway and the M2M device, which contain one agent on each node: the M2M server agent, the M2M gateway agent and the M2M device agent respectively. There is one M2M server in the system and it is connected to one M2M gateway. The M2M gateway is connected to the M2M server and to a set of M2M devices. For the purpose of energy efficiency, M2M devices turn off communication and go to *hibernate* state for some period of time and the M2M gateway is always in *on* state. In *hibernate* state an M2M device cannot communicate and cannot be woken up. When an M2M gateway wants to communicate with an M2M device, it needs to send data in the time frame when the M2M device is active. The M2M gateway agent will be in the watcher role and the M2M device agent in the presentity role in the presence domain. Each M2M entity is implemented on one Waspmote device. Waspmote devices communicate through the Xbee wireless interface that uses the 802.15.4 protocol[19].

The M2M server agent initiates communication by sending a request to the M2M gateway agent that is an intermediate agent between the M2M server agent and the M2M device agents. An M2M gateway agent receives from the M2M server agent a single, aggregated request for all the M2M device agents of M2M devices connected to it. The M2M gateway agent then checks whether it has RPI data for each M2M device agent to which it needs to forward the request. If the RPI data is available, it schedules the forwarding request, according to the RPI, for the next scheduled availability of the M2M devices. If the M2M gateway agent does not have RPI data for the requested M2M device agent, the request is scheduled for an immediate transmission. The M2M gateway agent then waits for the response. If the response is not received in a defined time–out period, the M2M gateway agent resends the request and the process repeats as if it were a new request for that M2M device agent (Fig. 5). The M2M device agent waits for a request and replies to it. When an M2M device agent replies, it forwards the response back to the M2M server agent. The M2M device duty cycle is a period when the M2M device is in *available* state. In that state the M2M device agent will reply to a request. If the request comes outside of this active period, the M2M device agent will receive it and simply ignore it in a simulated environment and it will not receive it in a real environment because the M2M device is in hibernation.

The parameters of the experiment are the number of M2M devices per M2M gateway and the M2M device duty cycle. The real environment experiment consists of two cases:

Case 1 The number of M2M devices per M2M gateway is changed from 1 to 8. There is 1 M2M gateway per M2M server and the M2M device agent duty cycle is 10%.

Case 2 The M2M device agent duty cycle is changed from 10% to 99%. There are 2 M2M devices per M2M gateway and 1 M2M gateway per M2M server.

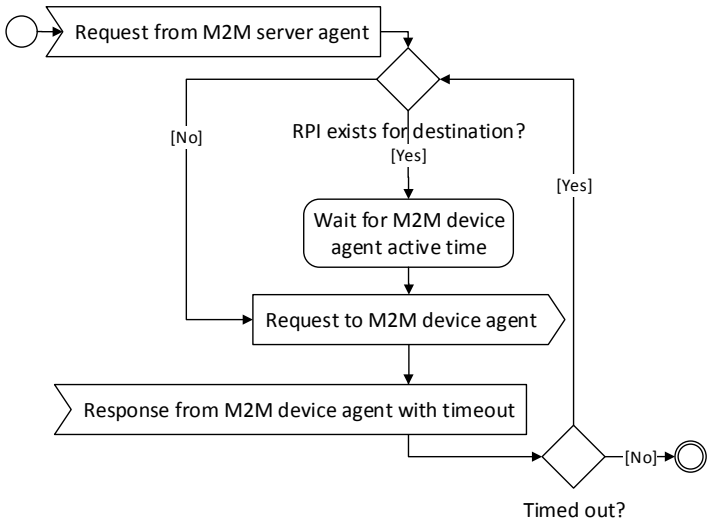


Fig. 5. Request processing activity diagram for M2M gateway agent

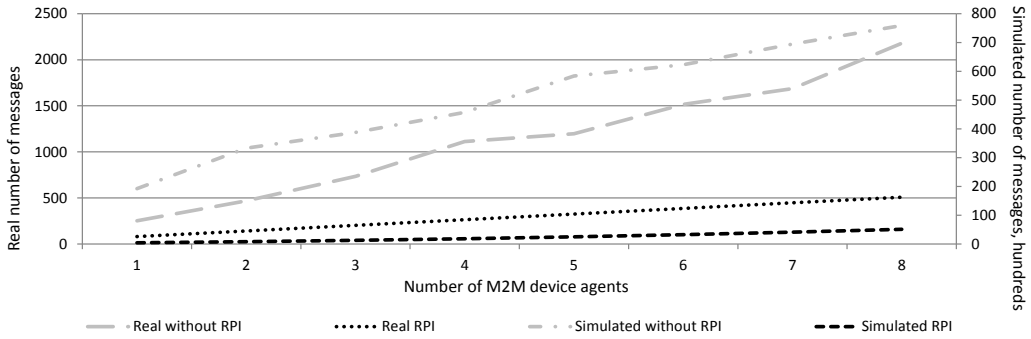


Fig. 6. Case 1, traffic generated for different number of M2M device agents per M2M gateway agent

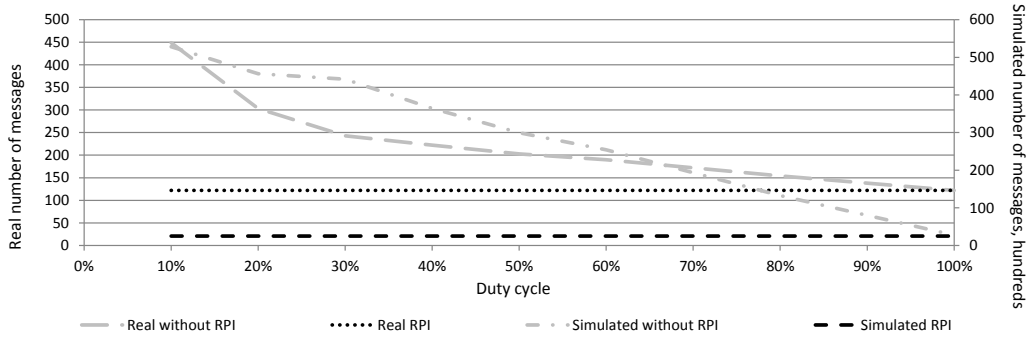


Fig. 7. Case 2, traffic generated for different M2M device agent duty cycles

For each of the cases, a simulation has been run 20 times. Results for Case 1 (Fig. 6) show a dependence of generated traffic on the number of M2M devices in the network. Compared to the situation with RPI, the traffic generated in a situation without RPI is rapidly growing with introduction of new devices into the network, with an example of 2179 data units with RPI versus 508 data units without RPI for 8 M2M device agents. In the situation with RPI, the results were equal for all the runs with the same number of M2M devices. This is due to the fact that the M2M gateway agent in each run had to send only one request per M2M device agent to receive a response, since it knew when the M2M device agent would be available. The difference in size of the generated traffic in the two situations is due to the number of requests from the M2M gateway agent to the M2M device agent when they are in an inactive state and do not respond to requests.

This result from the real environment shows the same trend as the result from the simulated environment[12]. The only difference is that in a simulated environment there was 5 M2M gateways per one M2M server and the duty cycle was 5%.

In Case 2 (Fig. 7), for the situation without RPI the trend of decrease of generated network traffic can be noticed as a consequence of the duty cycle increase. This is in line with expectations, since the probability of the M2M gateway agent transmitting during the active period of the M2M device agent increases with the increase of the duty cycle, therefore, lowering the number of retransmissions and generated traffic. Real environment results show a sharp increase in the generated traffic from 20% to 10%, which is slightly different from simulation environment results in[12]. Simulation results show roughly a linear decrease of the generated traffic from 1% to 99%. In the RPI situation, the value of the generated traffic is constant and it does not depend on the duty cycle. This is the expected behavior as long as the length of the active period is at least equal to the length of time needed to transmit the request from the M2M gateway agent to the M2M device agent. This is because the M2M gateway agent's scheduling does not take into account the delay of data propagation across the link. The results of the real environment experiment with RPI situation are the same as those of the simulated environment.

6. Conclusion and future work

The paper has introduced rich presence information for machines and that was used as context information in experiments. Mobile agent network formal model is extended with context information. The energy model of Waspmote devices has been defined and used in experiments, which show that optimized scheduling based on presence information exchange can reduce consumption of the whole system by more than 40%. Simulated and real environment experiments are compared and the results are the same in both environments.

In future work we will introduce a self–organized mechanism and negotiation for deciding which M2M device agent will be in the role of the M2M gateway agent for a certain amount of time.

Acknowledgements

This work was supported by strategic project "Energy Efficient M2M Device Communication" in cooperation with FTW Forschungszentrum Telekommunikation Wien GmbH, Austria.

References

1. Schulzrinne, H., Gurbani, V., Kyzivat, P., Rosenberg, J.. RPID: Rich Presence Extensions to the Presence Information Data Format (PIDF). RFC 4480 (Proposed Standard); 2006. URL: http://www.ietf.org/rfc/rfc4480.txt.
2. Siewruk, G., Legierski, J., Grabowski, S., Sredniawa, M.. Integration of context information from different sources: Unified communication, telco 2.0 and m2m. In: [3]; 2013, p. 851–858.
3. Ganzha, M., Maciaszek, L.A., Paprzycki, M., editors. *Proceedings of the 2013 Federated Conference on Computer Science and Information Systems, Kraków, Poland, September 8-11, 2013*. 2013.
4. Makris, P., Skoutas, D., Nomikos, N., Vouyioukas, D., Skianis, C.. A context-aware backhaul management solution for combined h2h and m2m traffic. In: *Computer, Information and Telecommunication Systems (CITS), 2013 International Conference on*. 2013, p. 1–5. doi:10.1109/CITS.2013.6705719.
5. Ahmed Surobhi, N., Jamalipour, A.. A context-aware m2m-based middleware for service selection in mobile ad-hoc networks. *Parallel and Distributed Systems, IEEE Transactions on* 2014;**PP**(99):1–1. doi:10.1109/TPDS.2014.2307875.
6. Day, M., Rosenberg, J., Sugano, H.. A Model for Presence and Instant Messaging. 2000. URL: http://portal.acm.org/citation.cfm?id=RFC2778.
7. Sugano, H., Fujimoto, S., Klyne, G., Bateman, A., Carr, W., Peterson, J.. Presence Information Data Format (PIDF). RFC 3863 (Proposed Standard); 2004. URL: http://www.ietf.org/rfc/rfc3863.txt.
8. Jurdak, R., Ruzzelli, A.G., O'Hare, G.M.P.. Adaptive radio modes in sensor networks: How deep to sleep? In: *SECON*. IEEE; 2008, p. 386–394. URL: http://dblp.uni-trier.de/db/conf/secon/secon2008.html#JurdakRO08.
9. Sinkovic, V., Lovrek, I.. Generic model of a mobile agent network suitable for performance evaluation. In: Howlett, R.J., Jain, L.C., editors. *KES*. IEEE. ISBN 0-7803-6400-7; 2000, p. 675–678.
10. Kusek, M., Jurasovic, K., Jezic, G.. Verification of the mobile agent network simulator - a tool for simulating multi-agent systems. *International Journal of Software Engineering and Knowledge Engineering* 2008;**18**(5):651–682.
11. Lovrek, I.. Context awareness in mobile software agent network. *RAD Hrvatske akademije znanosti i umjetnosti* 2012;(513):7–28.
12. Kusek, M., Lovrek, I., Maracic, H.. Rich presence information in agent based machine-to-machine communication. In: *KES*. 2013, p. 321–329.
13. Waspmote technical guide. http://www.libelium.com/downloads/documentation/waspmote_technical_guide.pdf; 2014. (Visited on 2014-02-08).
14. Waspmote datasheet. http://www.libelium.com/downloads/documentation/waspmote_datasheet.pdf; 2014. (Visited on 2014-02-08).
15. 8-bit Atmel Microcontroller with 64K/128K/256K Bytes In-System Programmable Flash. http://www.atmel.com/Images/doc2549.pdf; 2014. (Visited on 2014-02-08).
16. XBee/XBee-PRO RF Modules. http://ftp1.digi.com/support/documentation/90000982_M.pdf; 2014. (Visited on 2014-02-08).
17. XBee Multipoint RF Modules. http://www.digi.com/pdf/ds_xbeemultipointmodules.pdf; 2014. (Visited on 2014-02-08).
18. Mirza, D., Owrang, M., Schurgers, C.. Energy-Efficient Wakeup Scheduling for Maximizing Lifetime of IEEE 802.15.4 Networks. *Wireless Internet, International Conference on* 2005;**0**:130–137. doi:http://doi.ieeecomputersociety.org/10.1109/WICON.2005.14.
19. Waspmote 802.15.4 networking guide. http://www.libelium.com/uploads/2013/02/waspmote-802.15.4-networking_guide.pdf; 2014. (Visited on 2014-02-08).