

AGENT-BASED FORWARD ANALYSIS

Ryan Kerekes, Yu (Cathy) Jiao, Mallikarjun Shankar, Thomas Potok, and Rick Lusk
Oak Ridge National Laboratory
Oak Ridge, TN

ABSTRACT

We propose software agent-based “forward analysis” for efficient information retrieval in a network of sensing devices. In our approach, processing is pushed to the data at the edge of the network via intelligent software agents rather than pulling data to a central facility for processing. The agents are deployed with a specific query and perform varying levels of analysis of the data, communicating with each other and sending only relevant information back across the network. We demonstrate our concept in the context of face recognition using a wireless test bed comprised of PDA cell phones and laptops. We show that agent-based forward analysis can provide a significant increase in retrieval speed while decreasing bandwidth usage and information overload at the central facility.

INTRODUCTION

The military is faced with the problem of information overload on a daily basis. Massive amounts of data in such forms as intelligence updates, field reports, surveillance video, and battlefield imagery are generated continually by numerous heterogeneous sources. The ability to retrieve the right information from this mountain of data may be crucial to victory. Robust, intelligent software systems are needed to reduce data redundancy and answer complex queries by retrieving and fusing data in a distributed manner.

Notice: This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. Research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory (ORNL) managed by UT-Battelle, LLC, for the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

U.S. Government work not protected by U.S. copyright.

Because overwhelming quantities of data are generated remotely, distributed data mining algorithms would provide answers more quickly by taking advantage of the local processing capability of the data-generating devices. In many cases, the data is too large to be sent to a centralized location in real time due to bandwidth limitations. Thus, *forward analysis*, or pushing computation to data, would enable real-time analysis of distributed image and video data and reduce the amount of data that is sent back across the network. In other words, the system could send back only a few candidate needles rather than the entire haystack.

In this paper, we propose an agent-based forward analysis approach for distributed data mining and retrieval. Agent-based techniques are well-suited for distributed data mining and forward analysis. First, the autonomous nature of software agents leads to increased system robustness. Second, the existence of agent platforms enables fast system design, development, and deployment. Third, mobile agents can hop from device to device to carry out non-predefined tasks.

The agents in our proposed system are equipped with the capability to analyze data as well as to collaborate with other agents toward a goal-driven solution. Agents are given tasks of varying complexity depending on the processing capability of the target device. The system thus uses the agent paradigm to push computation toward the edge of the network in order to retrieve relevant information while decreasing bandwidth usage.

We use a newly developed mobile agent platform for implementing our agent-based system and providing a means for proof of concept. We employ a wireless test bed consisting of laptops and PDA cellular phones for hosting the software platform. Our platform is Java-based, which provides a certain level of device independence to mitigate the problems introduced by device heterogeneity.

In order to demonstrate the feasibility of the proposed approach, we use the example of distributed face recognition. We pose the problem in the following context: enabling a commander to initiate a search-by-example query over a large number of face images

captured by and stored on devices in the field in order to discover whether any device has photographed a particular suspect. We employ computationally efficient correlation filter techniques to analyze face images stored on the remote devices. Our agent-based computational scheme is organized hierarchically; specifically, agents at the very edge of the network are given lightweight processing tasks such as parsing the metadata associated with images, while intermediate agents perform image analysis to look for a particular person. We evaluate system performance in terms of response time in answering data mining queries, and we show that our distributed agent-based architecture outperforms a centralized solution.

The remainder of this paper is organized as follows. We first provide some background information on agent-based systems. We then provide a review of correlation filtering techniques used in our experiments. Next, we present our technical approach to agent-based data retrieval, including the different types of agents used in the approach. We follow by describing our experimental setup for demonstrating our proposed system, including our mobile agent framework for heterogeneous mobile devices, and we then present some results achieved by the system. We close with concluding remarks.

BACKGROUND

Several mobile agent-based solutions to data processing and retrieval problems have appeared in the literature. Most of these focus on either reducing power and bandwidth requirements or distributing existing algorithms for speed and efficiency. For example, Tei *et al.* [1] proposed a mobile agent system for location-specific data retrieval in which agents focus on choosing optimal geographic locations for reducing the amount of data transfer in an ad-hoc mobile network. Wang and Qi [2] proposed a mobile agent-based method for decentralizing the Bayesian source number estimation algorithm in a sensor network for increased energy efficiency.

Many research projects, including the two examples we just described, have demonstrated the benefits of using mobile agents in distributed, wireless computing by using simulations. The feasibility of transferring such systems from theory to reality remains questionable. Concerns about the realism and stability of mobile agent-based solutions grow even stronger when the target system is composed of heterogeneous devices.

Our focus in this paper is on introducing and demonstrating the concept of agent-based forward analysis. Thus, we do not focus on the algorithm or power efficiency but rather on the feasibility of a real-world mobile agent-based forward analysis system in which agents can be dynamically created to employ arbitrary data processing and analysis algorithms. Moreover, we seek to provide a realistic proof-of-concept demonstration of forward analysis on resource-limited wireless devices, something which we believe to be missing from the literature.

REVIEW OF CORRELATION FILTERS

In this section, we provide some background on correlation filtering, an image analysis technique we use in the implementation of our forward analysis concept. Correlation filtering is a technique for image-based pattern recognition whereby test images are compared to a predetermined template in order to detect and locate known patterns [3]. Variations on this basic method have been successfully applied in many application domains; some noteworthy examples include biometrics, automatic target recognition (ATR), road sign detection and optical character recognition (OCR). Properties such as distortion tolerance, graceful degradation in the presence of noise, and shift invariance make correlation filters an attractive solution for many image analysis problems.

An individual correlation filter is typically designed to detect a particular pattern or set of patterns under a small amount of potential distortion. Distortions may include such geometric phenomena as rotation and scale change, as well as more difficult-to-describe changes such as thermal state (for infrared imagery) and facial expression. Correlation filters are good at handling small amounts of distortion but may fail if the appearance of the pattern changes too much.

Many techniques have been proposed for designing correlation filters, but most of these are based on utilizing a set of training images that represent the expected appearance variation of the pattern. The interested reader may refer to the literature for details on correlation filter design [4]. We use a design technique called Optimal Tradeoff Synthetic Discriminant Function (OTSDF) [5]. Once the filter has been designed, it can be applied to any number of test images.

A *correlation output* is computed by correlating the filter template with the input image, i.e., computing the dot product between the two images at every 2-D shift;

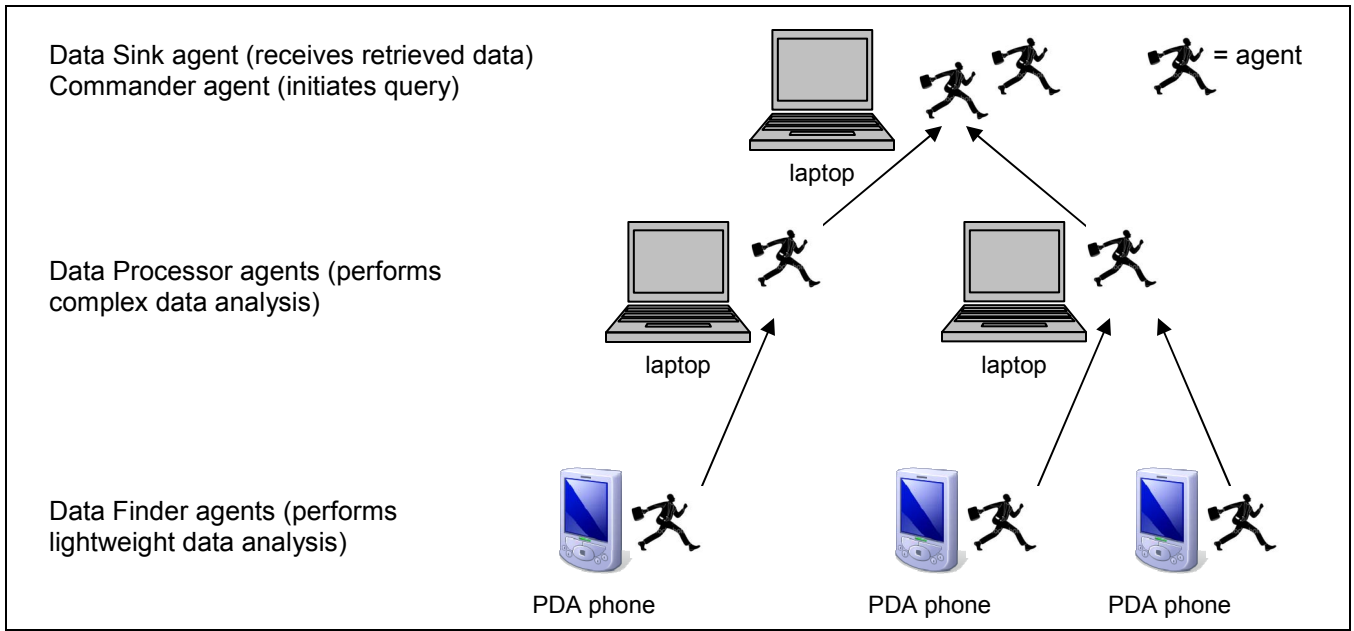


Figure 1. Illustration of experimental setup showing device hierarchy and associated software agents.

specifically, the output $c[m,n]$ is given by the following equation:

$$c[m,n] = \sum_i \sum_j x[i,j] h[i-m, j-n] \quad (1)$$

where $x[m,n]$ is the test image and $h[m,n]$ is the filter template image. This operation normally has complexity $O(mnpq)$, where the test image and filter template are of size $m \times n$ and $p \times q$, respectively; however, this computation may be carried out in the Fourier domain, which reduces the complexity to $O(mn \log mn)$. For small filter templates, however, it may be computationally more efficient to use Eq. (1). For pattern detection, the correlation output is typically compared to a threshold at each location, and every location where the output exceeds the threshold is considered to be a detection.

TECHNICAL APPROACH

Our approach to mobile agent-based forward analysis focuses on search-by-example queries. Specifically, the user specifies certain query parameters and clicks a button to initiate the search. A coordinating agent then deploys several specialized agents to various devices to find, process, and transfer relevant data back to the user. The user is not concerned with the underlying agents, but rather sees only the query parameters used and the search results returned after the search is complete.

For our forward analysis experiments, we designed four types of software agents, which we have given the

following names: (1) *Data Finder* agents; (2) *Data Processor* agents; (3) *Data Sink* agents; and (4) *Commander* agents. Each type of agent is designed with particular hardware resources in mind; e.g., the Commander agent uses a window for user interaction and thus needs a large screen for optimal display. However, the agents are not limited to a particular hardware setup; moreover, since they are Java-based, they can theoretically run on any machine with a compatible JVM (although this does not take into account the available processing resources for speed considerations). We describe the roles of each agent type below.

Data Finder: This agent is deployed to devices on which data is stored and acts as a preliminary filter for the data. The agent will browse through the available data and look specifically at the metadata associated with each data unit (e.g., a text report or an image). This metadata is in text format and includes time and date, GPS location (i.e., where the device was located when the data was generated), and the owner or group associated with the device (e.g., a particular military unit). If the user specifies location and time ranges in the query, then the Data Finder agent will carry these query parameters as it jumps to the device and apply them to each data unit. It will then send only the matching data to a Data Processor agent for further analysis. The Data Finder agent may select an associated Data Processor agent at random, or this association may be specified by the user depending on the system setup.

Data Processor: The role of this agent is to receive data from the Data Finder agent and carry out analysis of the data itself (rather than only the metadata) relative to the user-specified query. For text data, this processing might involve various text analysis algorithms such as TF-ICF [6], latent semantic analysis (LSA), natural language processing (NLP), genetic algorithms [7], etc., while for image data there exist a wide variety of algorithms for image analysis. In our experiments, we focus on image-based face recognition, and we employ a correlation filter-based method for analyzing the images and discriminating between faces. Correlation filters have been shown to perform well in various biometric tasks including face recognition [8-9], and they are known to be computationally efficient, making them well-suited for use in an agent-based forward analysis system.

Each Data Processor agent is equipped with one or more *correlation filters*, specialized image templates designed to recognize the face of a specific person and reject all others. Thus, Data Processor agents are tasked with finding specific persons among the data received from the Data Finder agents. The agent applies the filter to each incoming image and compares the output to a threshold, and if the output exceeds the threshold at any point, the image is declared a match and sent to the Data Sink agent. Each correlation filter template is typically grayscale and is equal in size to that of the pattern (e.g., the face) to be recognized. Thus, if faces are on average 100x100 pixels in size, each filter will be approximately 10KB in file size. This data is transported via the Data Processor agent to the machine on which the agent will run.

Data Sink: This agent acts as an interface between the Data Processor agents and the Commander agent. The tasks of this agent are (1) to receive data from all Data Processor agents and place them in a particular storage location, and (2) to send received data from a particular search to the Commander agent for display to the user. The reason for giving this task to a separate agent rather than including it in the Commander agent's tasks is that the user may desire to store data and perform searches on two separate machines. For example, the user may wish to operate at a small terminal machine, while a large file storage server is available elsewhere to store the large amounts of retrieved data.

Commander: This agent has two main roles: user interaction and agent coordination. First, the Commander agent provides a window whereby the user may initiate queries in the agent system and view search results. This window also allows the user to select

various settings for the query, such as which agent hosts in the system will accept different types of processing agents.

Second, the Commander agent coordinates the deployment of the other necessary agents upon initiating a query. This deployment happens in three stages: (1) a Data Sink agent is deployed; (2) Data Processor agents are deployed and subsequently locate the Data Sink agent; (3) Data Finder agents are deployed and subsequently locate the Data Processor agents. Once all the agents have been successfully deployed, the Commander agent waits for messages from the Data Sink agent, which represent the search results.

The agents communicate in a hierarchical fashion as follows. Data Finder agents locate data at the edge of the network and send the data to one or more Data Processor agents. The Data Finder agents thus need not maintain connectivity to central machines, but rather only to those machines on which their associated Data Processor agents are located. The Data Processor agents send data to the Data Sink agent, which in turn communicates search results to the Commander agent. This communication scheme is illustrated in Fig. 1.

The usefulness of this communication scheme can be motivated by the following example. Suppose that soldiers in the battlefield are equipped with agent platform-enabled devices. A commander may want to search the devices for reports and/or relevant image data, but not all soldiers may be within communication range of the central facility at all times. However, other communication devices may be placed strategically between the two (e.g., at an outpost), providing a link between the center and the edges of the network. Data Processor agents could thus be deployed to these intermediate machines, creating a robust and efficient data flow mechanism.

EXPERIMENTAL SETUP

In order to demonstrate our forward analysis proof of concept in a relevant environment, we need to have an agent platform in place. JADE-LEAP is a well-known mobile agent platform that supports handheld devices [10]. There are reports of varying degrees of success in using JADE-LEAP for the development of mobile agent-based systems [11, 12]. However, JADE-LEAP has only three configurations: j2se, pjava, and midp. The handheld devices in our testbed use the J2ME CDC configuration [13], which is not supported by JADE-LEAP.

We developed a new Java-based mobile agent framework, which we refer to as the Knowledge Acquisition Ubiquitous Agent Infrastructure (KAUAI). The design of KAUAI aims at high performance and reliability. This framework runs on both standard J2SE JVMs and a mobile JVM for Windows Mobile called CrE-ME developed by NSIcom. Because CrE-ME does not provide all the functionality of J2SE, it requires the Java code to be compatible with Java 1.3; for this reason, we designed the platform to meet Java 1.3 specifications.

The platform allows agents to be deployed at runtime to any remote machine running an agent host. A “locator daemon” is used to store the location and contact information (IP address and port) of each agent and agent host in the system. Any agent may communicate directly with any other agent after requesting the appropriate contact information from the locator daemon. Also, agents may move from one host to another at any time provided that they update the locator daemon of the move. A screenshot of the KAUAI agent host GUI is shown in Fig. 2. This GUI allows the user to select an agent type and a destination host for deployment.

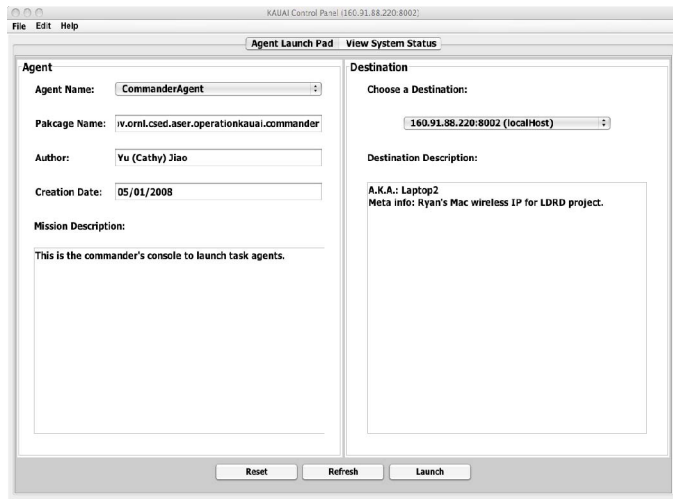


Figure 2. (U) Screenshot of GUI for interacting with KAUAI agent host.

We setup our proof of concept experiments using two AT&T® Tilt PDA phones running Windows Mobile® 6 with the CrE-ME JVM (referred to as Phone 1 and Phone 2) and an Apple® MacBook Pro laptop with a 2.4-GHz Intel Core 2 Duo processor running Mac OS X Leopard®. All devices were connected via an 11Mbps wireless network. We instantiated one Commander agent, one Data Sink agent, one Data Processor agent, and two Data Finder agents. The Commander, Data Processor, and Data Sink agents were located on the laptop, while each Data Finder agents was dispatched to

a different phone. A screenshot of the agent platform running an agent on one of the phones is shown in Fig. 3.

Each phone was loaded with a set of face images and associated metadata. The images ranged in size from 96x120 pixels to 160x120 pixels. Phone 1 contained 100 images, while Phone 2 contained 96 images. We initiated 4 different queries, referred to as A, B, C, and D. Each query specified a single person (i.e., a correlation filter designed for that person), a time range, and a location range. The Data Finder agents were responsible for matching the location/time portion of the query (as described earlier), while the Data Processor agent was responsible for the face matching portion. Table 1 shows the actual percentage of matching data items for each query, both for the Data Finder (DF) portion separately and for the whole query.

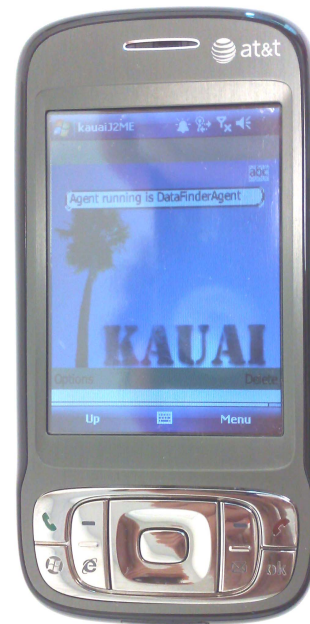


Figure 3. (U) Screenshot of KAUAI wireless mobile agent platform hosting a Data Finder agent on an AT&T® Tilt PDA phone.

Table 1. (U) Percentage of data relevant to each test query, both on the individual devices and overall.

Query	Query part	Phone 1	Phone 2	Overall
A	DF	0%	17%	9%
	Whole	0%	0%	0%
B	DF	20%	33%	27%
	Whole	20%	0%	9%
C	DF	40%	50%	45%
	Whole	20%	0%	9%
D	DF	60%	67%	64%
	Whole	20%	0%	9%

RESULTS

We measured the response time for each of the four queries, i.e., the amount of time between initiating the search and having received all the search results. This quantity includes the time taken to deploy the Data Processor and Data Finder agents, which we observed to take approximately 4s of the total search time for each query.

Response times for each query are shown in Fig. 4; we compare our distributed agent-based solution to a centralized solution in which all data is sent back to the central facility. We make two observations from this data: first, the distributed solution outperforms the centralized solution in terms of speed for each query; second, the speed of the distributed search depends on the amount of query-relevant data in the system (refer to Table 1). This second phenomena is due to the fact that in the distributed solution, only data that matches the query in the Data Finder agent is sent across the network, which turns out to be the primary bottleneck in the wireless network environment. Thus, the agent-based forward analysis approach can significantly reduce the network transfer volume and consequently the search time.

In both the centralized and the distributed solutions, we observed that the search results matched the corresponding query exactly, i.e., only the correct face images were returned, and only those matching the specified time/location range. This result indicates that the agent-based solution was able to reduce the search time without sacrificing search accuracy, which is to be expected since the agents employed the same processing algorithms as the centralized solution but at a remote processing location.

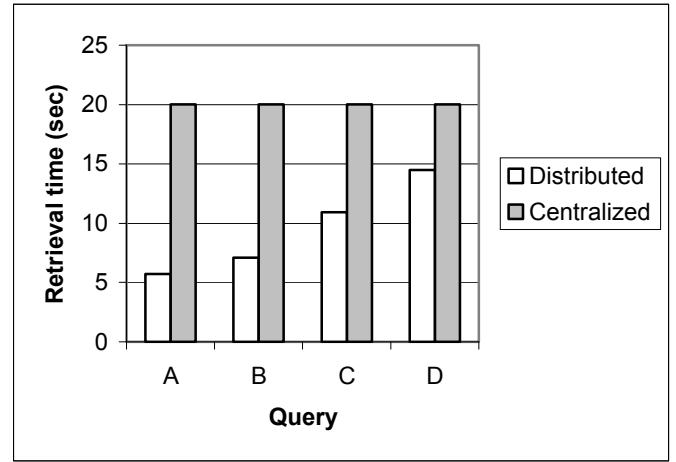


Figure 4. (U) Comparison of query times for centralized and distributed approaches for the four queries A, B, C, and D (see Table 1). Each value is the average over four trials.

CONCLUSIONS

We proposed the concept of agent-based forward analysis for data analysis and retrieval in a heterogeneous network of sensors and storage devices. We demonstrated this concept using a platform-independent mobile agent framework developed specifically for this task. Results showed that our proposed method outperformed a centralized solution in terms of speed while retaining the same search accuracy. We believe that for larger-scale problems, i.e., more data distributed over a greater number of devices, the performance improvement realized by agent-based forward analysis would be greater for two main reasons: (1) increased parallelism from multiple devices, and (2) diminished relative overhead in deploying agents.

REFERENCES

- [1] K. Tei, N. Yoshioka, Y. Fukazawa, and S. Honiden, "Using mobile agent for location-specific data retrieval in Manet," *Intelligence in Communication Systems*, Springer, pp. 157-168, 2005.
- [2] X. Wang and H. Qi, "Mobile agent based progressive multiple target detection in sensor networks," *ICASSP'04*, vol. II, pp. 285-288, 2004.
- [3] B. V. K. Vijaya Kumar, A. Mahalanobis, and R. Juday, *Correlation Pattern Recognition*, Cambridge University Press, 2005.
- [4] R. Kerekes and B. V. K. Vijaya Kumar, "Selecting a composite correlation filter design: a survey and comparative study," to appear in *Optical Engineering*, vol. 47, no. 6, 2008.
- [5] Ph. Refregier, "Filter design for optical pattern recognition: multi-criteria optimization approach," *Opt. Lett.*, vol 15, pp. 854-856, 1990.
- [6] J. W. Reed, Y. Jiao, T. E. Potok, B. A. Klump, M. T. Elmore, A. R. Hurson, "TF-ICF: a new term

- weighting scheme for clustering dynamic data streams,” *ICMLA’06*, pp. 258-263, 2006.
- [7] R. M. Patton and T. E. Potok, “Characterizing large text corpora using a maximum variation sampling genetic algorithm,” *GECCO’06*, pp. 1877-1878, 2006.
- [8] B.V.K. Vijaya Kumar, M. Savvides, K. Venkataramani, and C. Xie, “Spatial Frequency Domain Image Processing For Biometric Recognition,” *Proc. 2002 IEEE International Conference on Image Processing (ICIP)*, pp. 53-56, 2002.
- [9] M. Savvides, R. Abiantun, J. Heo, S. Park, C. Xie and B.V.K. Vijayakumar, “Partial & Holistic Face Recognition on FRGC-II data using Support Vector Machine Kernel Correlation Feature Analysis,” *Proc. CVPRW’06*, p. 48, 2006.
- [10] G. Caire, “LEAP User Guide,” <http://www.iro.umontreal.ca/~dift6802/jade/doc/LEAPUserGuide.pdf>.
- [11] R. Jedermann and W. Lang. “Mobile Java Code for Embedded Transport Monitory System,” *Proc. Embedded World Conference 2006*, p. 771-777.
- [12] A. Moreno, A. Valls, A. Viejo, “Using JADE-LEAP to Implement Agents in Mobile Devices,” *Research Report 03-008, DEIM, URV*.
- [13] <http://java.sun.com/javame/downloads/index.jsp>