Mobile Networks and Applications 9, 517–528, 2004
© 2004 Kluwer Academic Publishers. Manufactured in The Netherlands.

Mobile Agents for Wireless Computing: The Convergence of Wireless Computational Models with Mobile-Agent Technologies

CONSTANTINOS SPYROU and GEORGE SAMARAS
Department of Computer Science, University of Cyprus, CY-1678 Nicosia, Cyprus

EVAGGELIA PITOURA
Department of Computer Science, University of Ioannina, GR 45110 Ioannina, Greece

PARASKEVAS EVRIPIDOU
Department of Computer Science, University of Cyprus, CY-1678 Nicosia, Cyprus

Abstract. Wireless mobile computing breaks the stationary barrier and allows users to compute and access information from anywhere and at anytime. However, this new freedom of movement does not come without new challenges. The mobile computing environment is constrained in many ways. Mobile elements are resource-poor and unreliable. Their network connectivity is often achieved through low-bandwidth wireless links. Furthermore, connectivity is frequently lost for variant periods of time. The difficulties raised by these constraints are compounded by mobility that induces variability in the availability of both communication and computational resources. These severe restrictions have a great impact on the design and structure of mobile computing applications and motivate the development of new software models. To this end, a number of extensions to the traditional distributed system architectures have been proposed [26]. These new software models, however, are static and require a priori set up and configuration. This in effect limits their potential in dynamically serving the mobile client; the client cannot access a site at which an appropriate model is not configured in advance. The contribution of this paper is twofold. First, the paper shows how an implementation of the proposed models using mobile agents eliminates this limitation and enhances the utilization of the models. Second, new frameworks for Web-based distributed access to databases are proposed and implemented.

Keywords: mobile computing, mobile architectures, wireless architectures, mobile agents, client–server, wireless Web, software models

1. Introduction

In mobile wireless computing, most of the assumptions that influenced the definition and evolution of the traditional client/server model for distributed computing [9] are no longer valid. These assumptions include: (a) fast, reliable and cheap communications, (b) robust and resource rich devices, and finally (c) stationary and fixed locations of the participating devices. To accommodate the new computing paradigm introduced by mobile wireless computing, various software models have been proposed including the *client/agent/server* [2,7,21,28,32], the *client/intercept* [12,27], the *peer-to-peer*, [1,24] and the *mobile agent* [5,13,19,29] models. However, most of these models are static in that they have to be set up and configured a priori, a fact that, in some degree, limits their portability and usability. For example, utilizing the *client/agent/server* for distributed database access requires the definition and the a priori installation of a database agent at all participating sites [21]. Extending such infrastructures over the Web becomes even more cumbersome.

In this paper, we present a general framework for dynamically configuring applications to follow each of the proposed models for wireless mobile computing. This is achieved through the deployment of mobile agents. Such dynamic generation of various software models offers flexibility, adaptability and easy of use. Furthermore, the mobile agent implementation of the presented models further enhances their applicability to mobile wireless computing making applications more light-weight, tolerant to intermittent connectivity and adaptable to dynamically changing environments. Viewing mobile agents not only as an emerging computational model but as a communication paradigm as well add a new dimension to their applicability. The proposed framework is illustrated through a specific application, namely wireless web-based access to database systems. Besides serving in demonstrating our framework, this example also introduces new software architectures for wireless Web-based distributed access to databases. The dynamic installation of the models incurs tolerable overhead measured between 9.7 and 10.1 seconds in a wireless network setting. The mobile agent execution environments used in our implementation are IBM'Aglets [13] and ObjectSpace's Voyager [20] which are both Java based frameworks for building mobile agent objects.

The remainder of this paper is organized as follows. Section 2 defines the various software models for mobile computing and deploys them in building a Web application. Section 3 introduces a dynamic implementation of the models through mobile agents. Section 4 presents the effect of mobile agents on the existing models for mobile computing, while section 5 reports on the performance of the implementation. Various implementation issues including the choice of the implementation platform are discussed in section 6. Section 7 concludes the paper.

2. Software models for wireless Web access to distributed databases

2.1. Software models for mobile computing

The inadequacy of the traditional client/server model to support the wireless environment resulted in a number of new computational models. In this section, we briefly discuss the various models that have been proposed for wireless mobile computing. The reader is referred to the [26] for a detailed discussion of these models, their strengths and weaknesses.

2.1.1. Extended client/server models

Most extensions of the client/server model are based on the introduction of stationary agents placed between the mobile client and the fixed server. These agents alleviate the constraints of the wireless communication link by performing various communication optimizations. Furthermore, the introduction of agents alleviates any client-side resource constraints, by undertaking part of the functionality of resource-poor mobile clients. The degree to which this is achieved depends on the placement and functionality of the agents.

The client–agent–server model. A popular such extension is a three-tier or *client/agent/server* (c/a/s) architecture [2,7, 21,28,32] that introduces a server-site agent. Such agents are used in a variety of forms and roles. At one extreme, an agent acts as the complete surrogate of a mobile host on the fixed network. In this case, any communication to and from the mobile host goes through the mobile host's agent. At the other extreme, the agent is attached to a specific service or application, e.g., web browsing [12] or database access [21]. Any client's request and server's reply associated with this application is communicated through this service-specific agent. In this scenario, a mobile host must be associated with as many agents as the services it needs access to. Agents split the interaction between mobile clients and fixed servers in two parts, one between the client and the agent, and one between the agent and the server. However, while the client/agent/server model offers a number of advantages, it fails to sustain the current computation at the mobile client during periods of disconnection. In addition, the agent can directly optimize only data transmission over the wireless link from the fixed network to the mobile client and not in the opposite direction.

The client–intercept–server model. To address the shortcomings of the client/agent/server (c/a/s) model, [12,27] propose the deployment of an agent that will run at the end-user mobile device along with the agent of the c/a/s model that runs within the wireline network. This client-side agent intercepts client's request and together with the server-side agent performs optimizations to reduce data transmission over the wireless link, improve data availability and sustain the mobile computation uninterrupted. From the point of view of the client, the client-side agent appears as the local server proxy that is co-resident with the client. Since the pair of agents is virtually inserted in the data path between the client and the server, the model is also called [12,27] client/intercept/server instead of client/agent/agent/ server model. This model provides a clear distinction and separation of responsibilities between the client- and the server-side agents. The communication protocol between the two agents can facilitate highly effective data reduction and protocol optimization. The existence of the agent pair also facilitates adaptivity since the two agents can dynamically divide the workload among them based on various environmental conditions. The intercept model provides upward compatibility since it is transparent to both the client and the server. Legacy and existing applications can be executed as before since the agent pair shields them from the limitations of mobility and the wireless media. This model is more appropriate for heavy-weight clients with enough computational power and secondary storage to support the client-side agent.

2.1.2. Mobile-agent technologies

In mobile applications data may be organized as collections of objects, in which case objects become the unit of information exchange between mobile and static hosts. Incorporating active computations with objects and making them mobile leads to *mobile agents*.

Besides introducing stationary agents in the path between the mobile client and the server, mobile agents have also been used to accomplish tasks required by mobile clients. Mobile agents are processes dispatched from a source computer to accomplish a specified task [5,29]. Each *mobile agent* is a computation along with its own data and execution state. After its submission, the mobile agent proceeds autonomously and independently of the sending client. When the agent

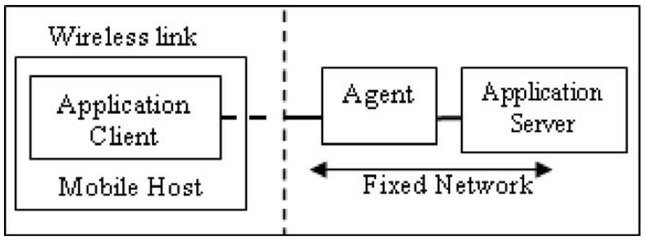


Figure 1. The client–agent–server model.

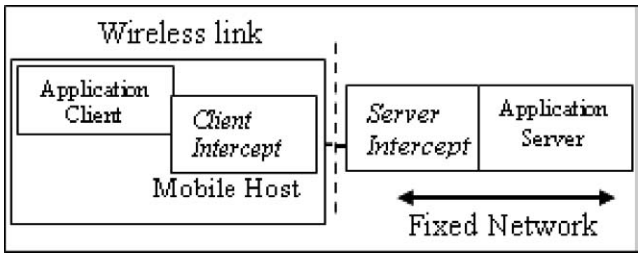


Figure 2. The client–intercept–server model.

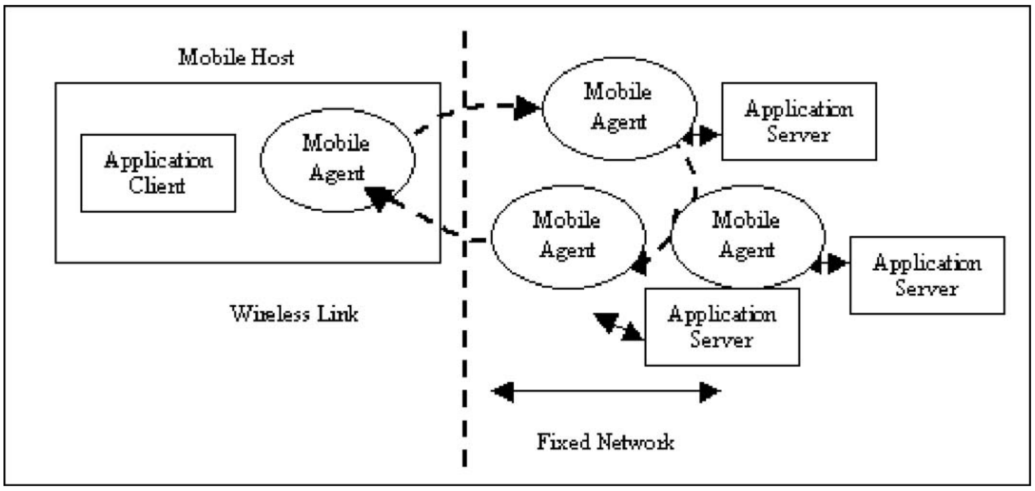


Figure 3. Mobile agent model.

reaches a server, it is delivered to an agent execution environment. Then, if the agent possesses necessary authentication credentials, its executable parts are started. To accomplish its task, the mobile agent can transport itself to another server, spawn new agents, or interact with other agents. Upon completion, the mobile agent delivers the results to the sending client or to another server. By letting mobile hosts submit agents, the burden of computation is shifted from the resource-poor mobile hosts to the fixed network. Mobility is inherent in the model; mobile agents migrate not only to find the required resources but also to follow mobile clients. Finally, mobile agents provide the flexibility to adaptively shift load to and from a mobile host depending on bandwidth and other available resources. Mobile-agent technology is suitable for wireless or dial-up environments (see figure 3).

Mobile agents platforms. Several mobile agent platforms have been proposed. They can be broadly categorized as Java and non-Java based ones. There is an increasing interest in those that are Java-based due to the inherent advantages of Java, namely, platform independence support, highly secure program execution, and small size of compiled code. These features of Java combined with its simple database connectivity interface (JDBC API) that facilitates application access to relational databases over the Web at different URLs make the Java approaches very attractive for our implementation of web database connectivity. The Java-based mobile agents platforms include IBM'Aglets Workbench [13], ObjectSpace's Voyager [20], Mitsubishi's Concordia [31], IKV++ Grasshopper [4] and General Magic's Odyssey [29]. The non-Java-based systems include, for example, TACOMA [16] and Agent Tcl [8]. While all these systems provide the basic functionality expected from such mobile platforms, they differ significantly in their system architecture, the communication mechanism employed, the additional functionality they provide and their performance. For our purposes, namely to demonstrate the abilities and potential of mobile agents, we have chosen to use IBM'Aglets and ObjectSpace's Voyager. Aglets are popular in terms of ease of programming and functionality, while Voyager, as shown in recent studies [6], provides very good performance. For a comprehensive comparison and quantitative performance evaluation of the Java-based mobile agents platforms see [6,25].

2.2. Wireless Web access to distributed databases

To demonstrate the applicability of the models, we employ them in accessing distributed database systems through the Web. The implementation of the models shows that all proposed models outperform the current client/server approach. In fact, in the wireless environment and for average size transactions, the client/agent/server-based framework provides a performance improvement of approximately a factor of ten (see charts 5, 7, 8). Even for the fixed network, the gains are about 40% (see figures 6–8). Furthermore, the current commercial client/server applet-based methodologies for accessing database systems (using the JDBC API and JDBC drivers API [14,15] for database connectivity [9]) offer limited flexibility, scalability and robustness [22]. Within the wireless environment, these setbacks are further exacerbated.

2.2.1. Using the client/server model

Realizing the client/server model requires the web browser to be located at the mobile client and communicate directly with the web server via wireless communications. When the client requires access to a specific database, the client downloads the appropriate database applet (DBMS-applet) to the mobile unit form the database provider. Then, a JDBC [15] connection is established between the client and the database server.

Limitations. Most existing approaches require to some extent the download and initiation of a JDBC driver on the client machine [22]. The limitations of wireless communications are aggregated by the mechanism of establishing a JDBC connection. For every connection, the client must download this JDBC driver whose size ranges between 300–500 Kb. Chart 1

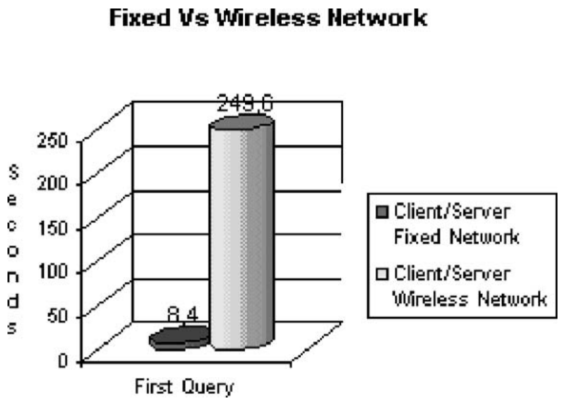


Chart 1. JDBC connection time over the wireless and fixed network. The bandwidth of the wireless and wireline links used are 9600 Kb/s and 10 Mb/s, respectively.

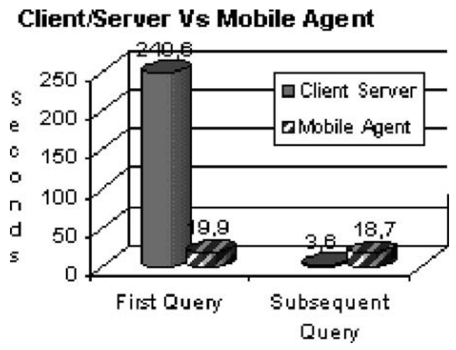


Chart 2. Client/server vs. mobile agents in the wireless environment.

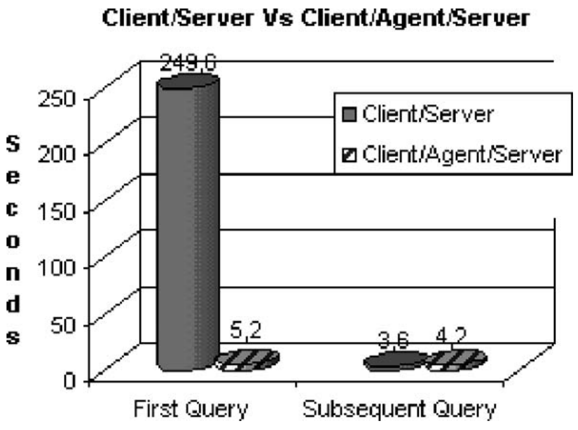


Chart 3. Client/server vs. client/agent/server in the wireless environment.

shows the cost of establishing a connection and executing a query.

2.2.2. Using the mobile agent model

The mobile agent approach is based on using mobile agents [11,29] between the client program and the server machine to provide database connectivity, processing and communication. In particular, an appropriate applet creates and fires a mobile agent (or agents if necessary) that travels directly to the remote database server.[1] At the SQL server, the mobile agent initiates a local JDBC driver, connects to the database and performs any queries specified by the sending client [22]. When the mobile agent completes its task at the SQL server, it dispatches itself back to the client machine directly into the DBMS-applet from where it was initially created and fired. Since these kind of mobile agents possess database capabilities, we call them DBMS-agents or DBMS-Aglets [22]. Note that database capabilities are dynamically acquired not at the client but at the server. Chart 2 compares the client/server and the mobile agent models (implemented via IBM'Aglets) over the wireless link for the first and subsequent queries.

Advantages. By using a DBMS mobile agent (namely the DBMS-Agent) to encapsulate all interactions between the client applet and the SQL server machine, the client applet becomes light and portable. This is achieved by: (a) avoiding the unnecessary downloading and initialization of JDBC drivers at the client's DBMS-applet, (b) passing the responsibility of loading the JDBC driver at the SQL server to the DBMS-agent, and (c) not using any JDBC API classes at the client's DBMS-applet. The effect on performance is quite significant; this delegation of responsibility resulted in performance gains of approximately a factor of four (see performance evaluation in section 5, charts 5, 7 and 8). The DBMS-agent is also independent of the various JDBC driver implementations. The DBMS mobile agent cannot (and is not supposed to) be aware of which JDBC driver to load when it arrives at an SQL server. Upon arrival at the SQL server's context, the DBMS-agent is informed of all available JDBC drivers and corresponding data sources. The DBMS-agent is then capable of attaching itself to one or more of these vendor data sources.

Limitations. An agent is fired to the database server and a JDBC connection is established, each and every time a request is issued, thus introducing an unnecessary and undesirable overhead.

2.2.3. Using the client/agent/server model

Employing the client/agent/server model requires the deployment of an agent or proxy on the fixed network [21]. For this example application, a service-specific agent, the database agent, represents the client. The database agent at the fixed network can serve multiple mobile web clients, in which case each client has to register with the agent. All database-related traffic to and from the mobile host goes through the database agent.

Database connectivity is now the responsibility of the static agent. A JDBC connection can be established and maintained for the whole duration of the client's application, thus eliminating the previous limitation of creating an agent and a connection per request. The structure, however, is fixed requiring the a priori installation and configuration of a database proxy. Chart 3 compares the client/server and client/agent/server models in the wireless environment. In fact, as is further shown in charts 5–8 (see performance evaluations in section 5) this approach by far outperforms the tra-

[1] For simplicity and clarity purposes we will refer to the "database server" as the "SQL server".

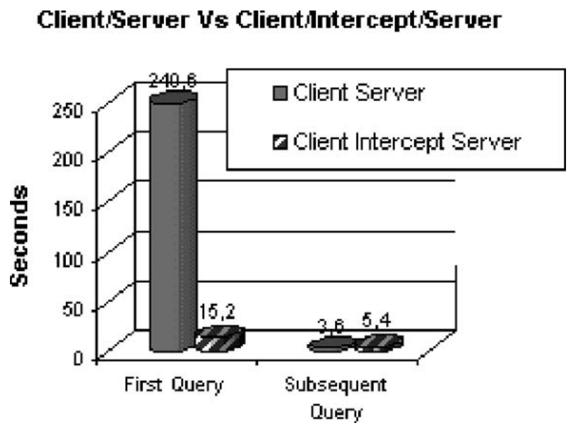


Chart 4. Client/server vs. client/intercept/server in the wireless environment.

ditional (applet) client/server approach (by a factor of ten) as well as the other variations of the framework for direct data access. The optimizations that can be realized by the database agent are mostly optimizations that reduce the client or server computational load and optimizations that minimize the communication overhead only from the server to the client and not vice versa [26].

2.2.4. Using the client/intercept/server model

Employing the client/intercept/server model [27] introduces a database-specific agent residing at the mobile client, in addition to the database agent at the fixed network. We will call the "database" agent at the client, client-side agent and the database agent at the fixed network server-side DBMS-agent. While the server-side agent at the fixed network might serve multiple clients, the client-side agent is unique to the client. In contrast to its server-side counterpart, the client-side agent does not need to posses database capabilities (i.e., JDBC connection capabilities). The agent pair cooperates to intercept and control communications over the wireless link for reducing network traffic and query processing.

Functionality at the client-side agent might include various optimizations such as client-side *view materialization or caching* to support disconnection and weak connectivity [23,30] and an *asynchronous-disconnected mode*, to allow queries that cannot be satisfied by the view to be automatically queued when connectivity is lost and resumed when connectivity is re-established [3,10,12].

3. Mobile agents to the rescue

To employ the proposed models, appropriate components for each model must be configured and set up a priori for each different application. This results in a static configuration. Accessing a new site becomes if not impossible quite cumbersome prohibiting global utilization of network resources. To access a server at a network site, the site must be appropriately configured in advance to include necessary software modules. In this section, we show that combining the various models with the capabilities provided by mobile agents permits dynamic configuration of the various models.

3.1. Realizing the "mobile" client/agent/server model via mobile agents

The mobile agent framework for web database access can be incorporated with the mobile client/agent/server model. The (server-side) agent of the client/agent/server model can be a mobile agent dynamically created at the client, and then sent and parked at the SQL server. This agent is maintained at the server temporarily for the duration of the application. Between this parked agent and the remote client, another agent carries requests and results back and forth.

Based on this variation (see figure 4), upon the first client request, two DBMS-agents are fired from the DBMS-applet at the client. The first one is called parked DBMS-agent and is the client's surrogate for database access located at the fixed network. Its role is to "camp" at the SQL server's agent context, load the appropriate JDBC driver, connect to the database, submit requests and collect and filter the answers. The second agent is the messenger agent. The messenger agent is responsible for carrying the requests and results back and forth to the DBMS-applet. All requests are transmitted to the parked DBMS-agent via the messenger agent.

This scheme is proved to be very efficient in cases where the user issued, through the DBMS-applet, a number of consecutive database requests to the same remote SQL servers. Charts 5 and 6 (see section 5) indicate a 20–30% performance improvement over the mobile agent model. Furthermore, these two agents may own the same itinerary and thus, if the parked DBMS-agent moves to another server the messenger agent can deterministically follow it and thus dynamically maintaining the client/agent/server model. An additional benefit of this approach is the ability of the messenger agent to roam, if needed, around the network before returning to the client. Setting up this model takes about 9.7–10.1 seconds.

3.2. Realizing the "traditional" client/agent/server model via mobile agents

The traditional client/agent/server (c/a/s) model can be materialized by letting the client communicate with the agent through messages instead of mobile agents. This is achieved by replacing the messenger agent with two types of messages. The first type of message, delivered from the DBMS applet to the DBMS-agent, contains the client query and any additional directions to the parked DBMS-agent that might be needed. The second type of message, from the parked DBMS-agent to the DBMS applet, contains the results of the last query. This methodology demonstrates a true service-specific client/agent/server application. The agent is literally inserted into the path between the client and the server communicating with each other via messages.

By using a DBMS-agent parked at the server, we avoid the reconnection cost and by using messages instead of the

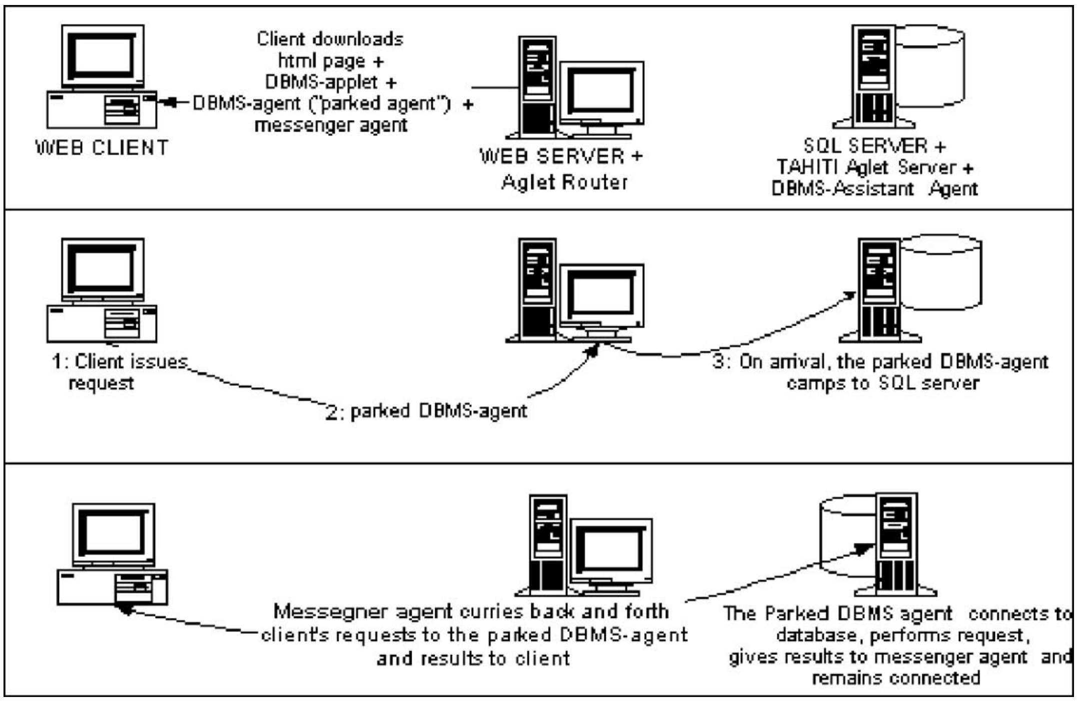


Figure 4. Materialization of the "mobile" client/agent/server model via mobile agents.

messenger agent we eliminate the time of negotiation and the amount of data transmitted between the client and the server.

3.3. Realizing the client/intercept/server model via mobile agents

As in the case of the client/agent/server model, we can implement the client-side agent as a mobile agent. In this case, upon the first client request, the DBMS-applet creates two agents: the client-side agent and the server-side DBMS-agent. The client-side agent remains at the client while the server-side agent is dispatched to the appropriate server. The two agents communicate and cooperate to execute various queries/requests. Again, communication can be performed either via agents or via messages. Both agents are maintained for the duration of the application. Setting up this model (with cross agent communication via messaging) takes about 9.7–10.1 seconds.

4. Mobile agents and the client/server models: New computing models

Realizing the various client/server variations via mobile agents brings out new insights regarding this new distributed computing model. Thus, Mobile agents give rise to new computational models for mobile computing, which we collectively call the *Mobile-Agent model* (see figure 3). This model enables a high degree of flexibility as it incorporates the advantages of mobile-agent platforms. As is been shown the Mobile Agent model is not orthogonal to the client–server model and its extensions, since mobile agents are used to *dynamically materialize* and extend models like the C/S, C/A/S and C/I/S. For example, the server-side agent of the C/A/S and C/I/S model may be seen as a stationary agent, i.e., an agent lacking the ability to migrate to other servers. One can implement these agents, however, as mobile agents that are placed at the client and the server dynamically. Furthermore, the server-side agent may be permitted to move within the fixed network, "following" its associated client, to remain "near" the client and yet within the fixed network. Once the server-side agent starts roaming the fixed network, it can communicate with the client not only via messaging but also via mobile agents. These agents can also roam the fixed network and connect to other servers before returning to the client, thus enhancing the model's flexibility. Such an approach presents many benefits in the wireless and dial-up environments, as well as in the world of Internet services and applications [18].

The combination of the mobile-agent model with "traditional" software models for mobile computing gives rise to new software models. For instance, the employment of mobile agents for client–server communication, instead of messages, leads to the "*mobile*" *client–server* (C/S-MA), "*mobile*" *client–agent–server* (C/A/S-MA) and "*mobile*" *client–intercept–server* (C/I/S-MA) models.

In these cases we denote (d) as messenger *agents* the mobile agents used by the server and the client to communicate. In such a scenario, the client creates a messenger mobile agent and submits it to the server machine. Upon reception, the server processes the information presented by the messenger agent. In a more flexible approach, a server could refine and extend the messenger agent and then launch it to other servers on the network. When the messenger agent finishes its task, it returns with the results to the server. The server filters out any unnecessary information and transmits to the mobile client

Table 1
Software models for mobile computing.

Communication paradigm	Interaction modality between clients and servers			
Messaging	C/S	C/A/S	C/I/S	Mobile agent model
Messenger agents	C/S-MA	C/A/S-MA	C/I/S-MA	

only the relevant data. Such an approach entails enhancing servers and clients with capabilities to process mobile agents, i.e., modifying/refining their state and is, in some respect, in accordance with current research on active networks [28].

In summary, mobile agents play a double role; they participate into these frameworks not only as a computational unit but as a communication mechanism as well. An extended taxonomy of software models for Mobile Computing, according to the connection modality between clients and servers (client–server, client–agent–server, etc.) and the communication paradigm employed to establish and manage connections, is presented in table 1.

5. Performance evaluation

The performance evaluation compares the total time that is required by a Web client to access and query a remote database between the traditional (i.e., client/server) applet-based model and the methodologies that are based on the proposed computational models. Each of the models is dynamically materialized via mobile agents[2] as described in the previous section. The mobile agent execution environment used to realize the models is the Aglet Technology [13], developed by IBM Tokyo, which is a Java based framework for building mobile agent objects. To show that these models can be effectively used in the fixed network as well we performed the same set of tests over both a wireless and a fixed network environment.

Specifically, we are interested in the time required, by each methodology (a) to query the remote database for the first time and (b) to complete any subsequent requests. We consider both short (three queries) and long transactions (six queries) between the client and the remote database. Short or long transactions are the type of transactions generally anticipated by Web users. For each methodology, we performed the evaluation having the client accessing the Web server via:

- a 9,600 b/s wireless dial-up connection to an Internet Service Provider (ISP);
- a 10 Mb/s Ethernet connection (fixed network).

For each methodology and client connectivity case, we performed the tests numerous times and from different remote clients. Specifically, each set of experiments consisted of more than 100 queries randomly distributed between the seven hourly intervals composing the time span between

[2] The implementation of all these frameworks can be found in `http://ada.cs.ucy.ac.cy/~cssamara/dbms-agents`

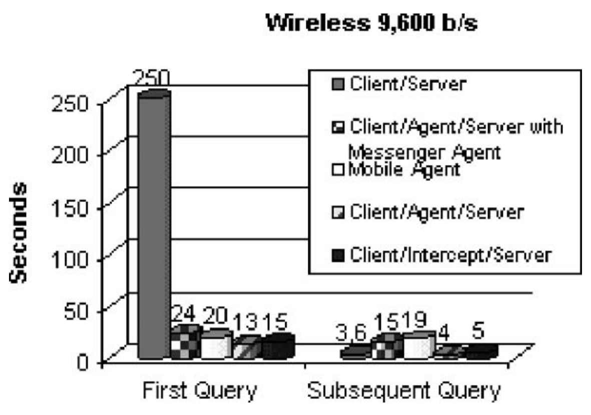


Chart 5. Mean times for 9,600 b/s client connectivity.

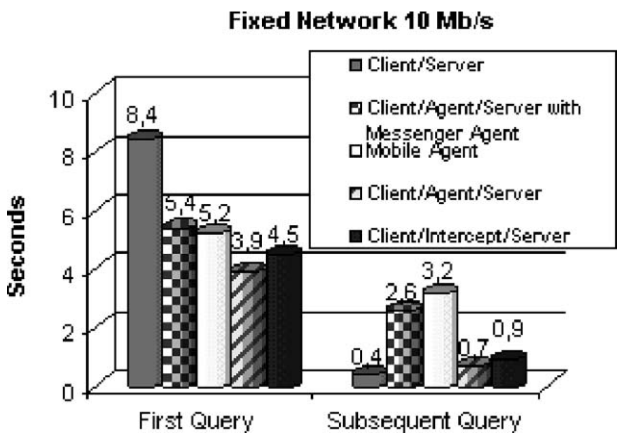


Chart 6. Mean times for 10 Mb/s client connectivity.

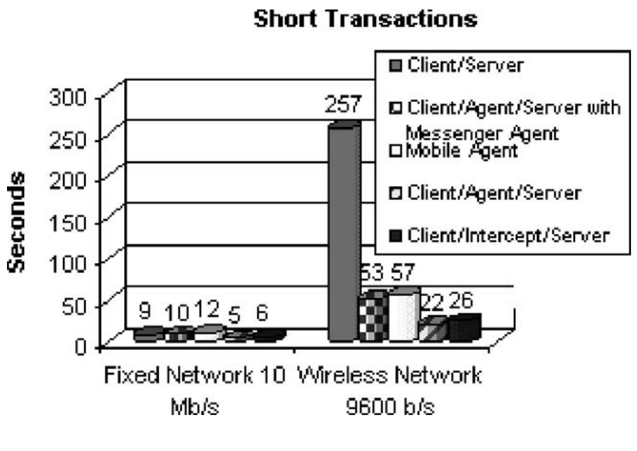


Chart 7. Short transactions graph.

9 AM and 5 PM. Each tested methodology provides two data sets of results (observations): one for the first query and one for the subsequent queries.

As shown in charts 5–8, the proposed software models for wireless mobile computing outperform the current client/server approach. In the figures, the client/agent/server model is the traditional one; i.e., communication is through messages. As indicated by the experiments, the client/agent/server methodology requires considerably less time than any other methodology, while the client/server methodology sig-

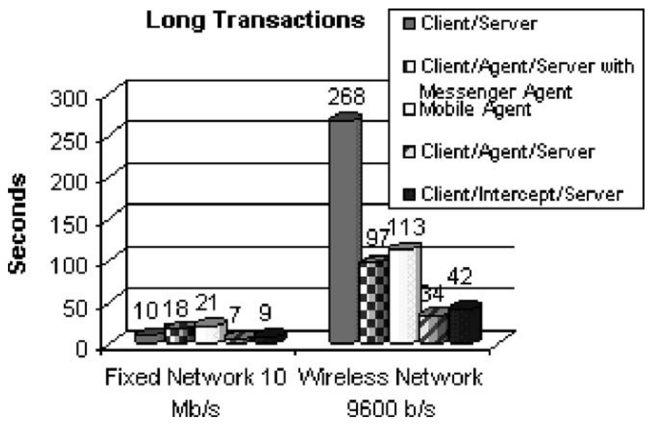


Chart 8. Long transactions graph.

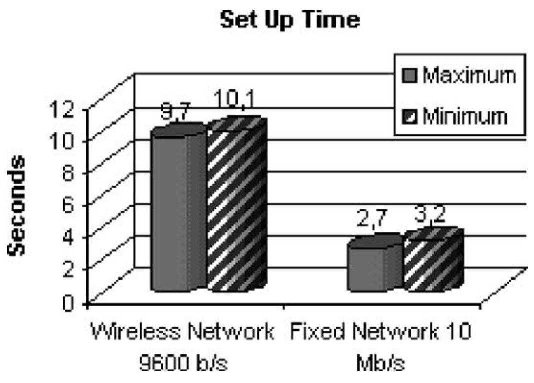


Chart 9. Time needed to set up the models for the fixed and the wireless network (the set up times are the same for all the models since only a single agent is sent to the fixed network).

nificantly more time than any other does, except from the fixed network case. In fact, the client/agent/server variation provides a performance improvement of approximately a factor of ten. Even for the fixed network the gains are considerable, about 40%. This brings about a positive side effect of this work, namely the ability to effectively utilize the proposed models in the fixed network as well.

Note that the performance results demonstrate only the overheads of the proposed models and are not indicative of their full functionality. For example, although, the client/intercept model lacks in performance due to the additional overhead of creating a client-side agent, it supports compatibility, since it can be built on top of existing applications. Similarly, while communicating with messenger agents instead of messages may be more costly, it is more flexible in that messenger agents can perform complicated tasks.

Note that the times recorded for the first query includes the setup time as well. We considered this appropriate since we wanted to charge for the dynamic installation of the models. Chart 9 summarizes the time needed to set up each of the models. For the wireless case, this time is 9.7–10.1 seconds. In the wireline case, to set up the various models takes between 2.7 and 3.2 seconds.

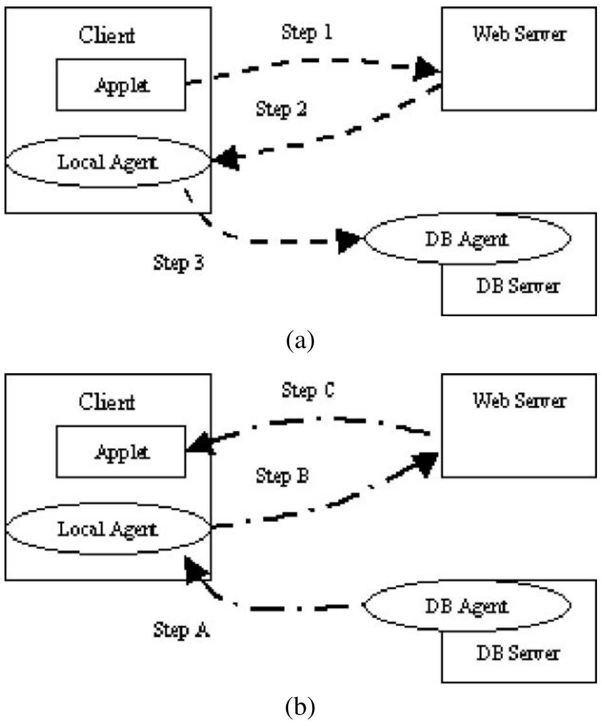


Figure 5. Client–intercept–server model. (a) Request. (b) Result.

6. Implementation issues

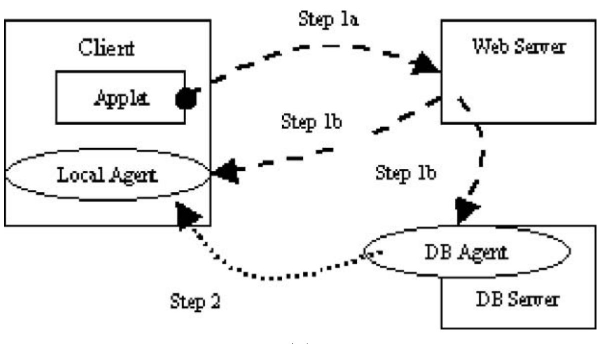
6.1. The role of the environment

Someone cannot avoid noticing the lesser performance provided by the C/I/S model compared to the C/A/S model (see charts 5, 6). The natural question is why is that since the intercept model provides more flexibility that should have been translated into better performance. The intercept model introduces a level of indirection that is not present in the C/A/S model: the client communicates its request to the client-side agent and then to the server-side agent. This indirection creates some delay which is not mask by any added optimizations such caching or communication protocol optimization, simply because we have not at this stage included any.

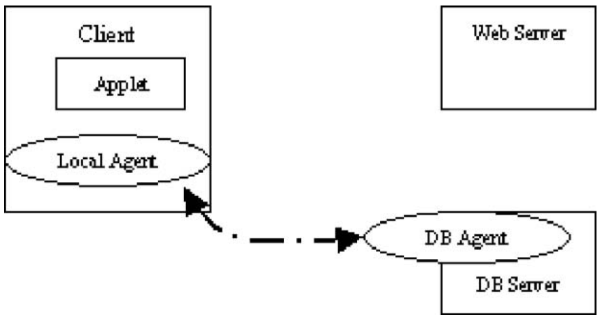
However, the introduced delay far exceeds the anticipated. This is the result of restrictions imposed by the Web technology itself. Web security restrictions add a number of hidden levels of indirection that once removed result in much better performance. Figures 5(a) and 5(b) present the indirections that are introduced by the Web. Once the client-side agent is placed on the mobile unit every request arrives to the "local" agent via the Web server (steps 1, 2, 3).

Similarly, every result is returned to the client-side agent via the Web server which is then directed to the client applet again via the Web server (steps A, B, C). These extra trips significantly affect performance and are the cost of the lesser performance provided by the C/I/S model compared to the C/A/S model.

Fortunately, mobile agents provide unique flexibility allowing alternate ways of approaching a problem. The result of this flexibility is shown in charts 10(a) and 10(b). In the chart the *New Intercept* model is shown to outperform both



(a)

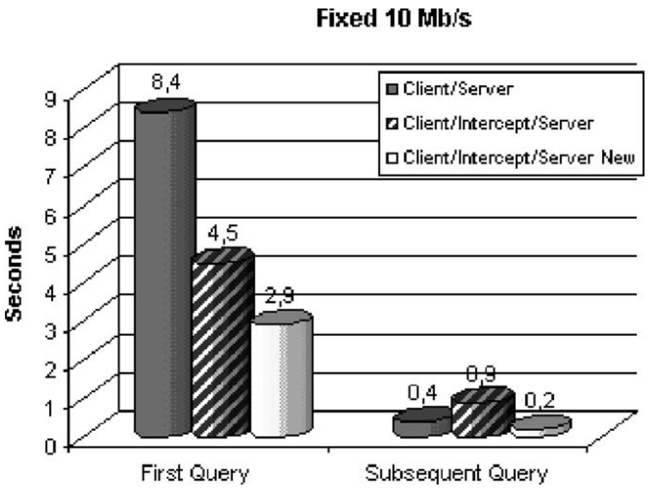


(b)

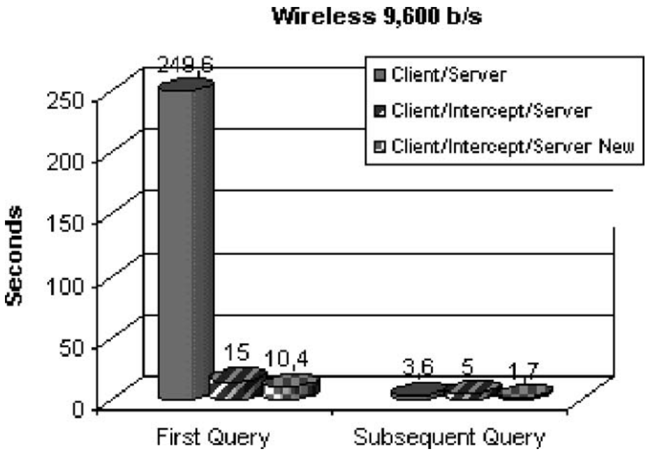
Figure 6. New Intercept model. (a) First request. (b) Subsequent requests.

the C/A/S and C/I/S approaches for the first as well as the subsequent query. This was achieved by enhancing the client-side agent with the functionality of the DBMS-applet interface (see figures 6(a) and 6(b)). Thus for a request to materialize the C/I/S model we send to the client the new and enhanced client-side agent. Once this agent arrives at the mobile unit the applet interface is killed and the interface carried by the agent (which is exactly the same as the DBMS-applet interface) pops out. Via this simple "trick" we bypass all the restrictions of the Web and achieve direct communication between the client, the client-side agent and the server side-agent. The result is shown in charts 10(a) and 10(b).

In similar fashion, for the case of the C/A/S model, creating the "parked" DBMS-agent on the Web server and not on the client, achieve similar flexibility. The created on the Web server agent is also enhanced, namely it includes the applet interface. The client gets the user request, compose the SQL query and alone with the other needed information submits it via a message to the "parked" agent (at the Web server). The "parked" agent receives the info (i.e., the Database's URL, the SQL query, etc.) and moves to park and execute it at the remote destination. There it creates an agent that along with the result carries the interface as well (see figure 7). Back to the client, the DBMS-applet interface is killed, then the agent displays the interface and the results, gets the new query and goes to the server and so on and so forth. Charts 11(a) and 11(b) compares the results of this approach with the "Mobile" Client/Agent/Server model. Attempting to similarly enhance the "traditional" Client/Agent/Server model essentially results into the New Intercept model.



(a)



(b)

Chart 10. Client/intercept/server vs. client/intercept/server new (a) in the fixed environment; (b) in the wireless environment.

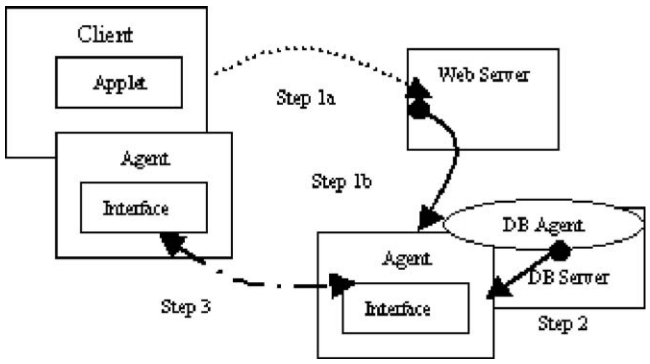
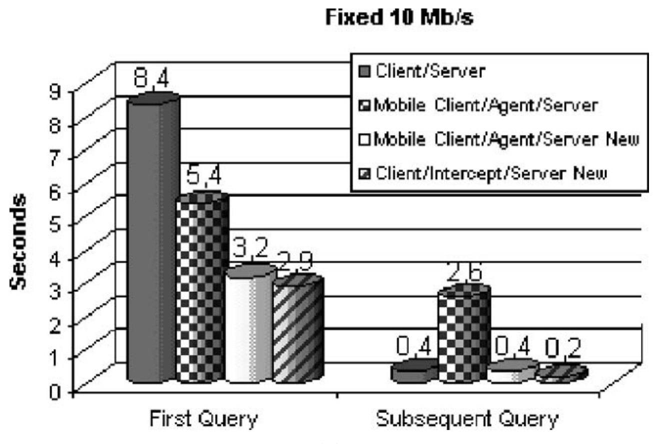


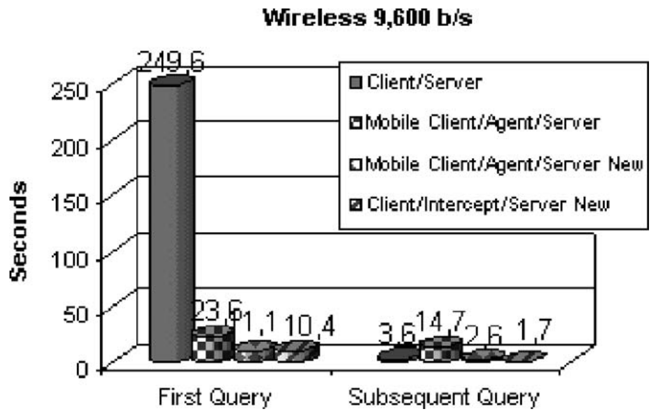
Figure 7. The "Mobile" client/agent/server new model.

6.2. *The role of the implementation platform*

In this work, so far, we have used IBM'Aglets Workbench [13]. Aglets have mainly concentrated on functionality and not, thus far, on performance [17]. To give an example, in the Aglets workbench, whenever an agent is transferred to a new destination, all the reachable objects are transported along with it and this is done even in the cases where the



(a)



(b)

Chart 11. Client/server vs. mobile client/agent/server vs. mobile client/agent/server new vs. client/intercept/server new (a) in the fixed environment; (b) in the wireless environment.

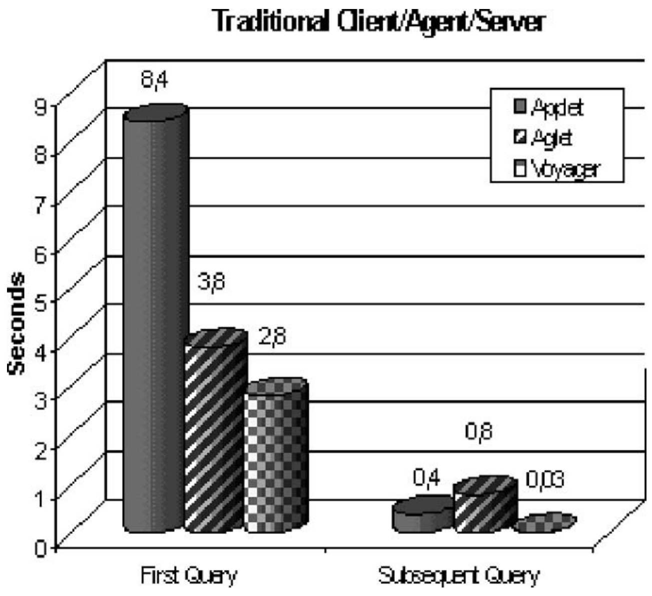


Chart 12. Voyager special implementation of the "Message" approach.

agent is sent to the same destination multiple times. Other Java-based mobile agents platforms include ObjectSpace's Voyager [20], Mitsubishi's Concordia [31], IKV++ Grasshopper [4] and General Magic's Odyssey [29]. While they are all based on Java, each one of them adds its own special implementation features that can significantly affect performance.

In this section, we report on our experiments in implementing our framework using a different agent platform, namely Voyager. The results show that the new implementation of our framework outperforms the applet approach even in the case of the fixed network. After porting the Voyager platform and port into it the DBMS-Aglet framework, we performed the same tests under the same network/system configuration as well as the needed statistical analysis. Worth noting is that the porting was performed smoothly and in a timely fashion.

In all approaches, for the subsequent query the Voyager implementation by far outperforms the Aglet implementation. For the "mobile agent" model and the "mobile C/A/S" model approaches the performance is very close to that of the Applet approach. The "traditional C/A/S" (see chart 12) approach, however, outperforms the Applet approach by a factor of thirteen. This improvement is attributed to Voyager's agent transportation procedure. Voyager is based on RMI while Aglets implement their own agent transport protocol (ATP). Aglets in each and every agent transport transmit all the needed class objects as well, while Voyager transports (following the RMI philosophy) the needed object classes only once and on a need-to-use base. Any subsequent request is satisfied by the now cached at the destination objects.

The performance of the first query, however, is disappointing. In every approach, the Voyager implementation performs the worst. This "bad" performance is the trade off for higher flexibility. This difference in performance is attributed to the fact that during the execution of the first query the Voyager implementation requires the downloading (to the client) of all the needed classes for the dynamic creation of the agent execution environment. Aglets do not need to do that since the required classes are preloaded to the client (the so called aglet plug in). For the cases where we do not need to host at the client the agent execution environment, such a case is the traditional C/A/S model, by remotely creating the needed agents we can achieve tremendous performance even for the first query. Chart 12 shows the performance of such as implementation.

Implementation issues are important, especially in newly created environments such as Mobile agent technologies. In the particular case of chart 12 the current implementation of IBM Aglets Workbench could not provide so readily and effectively the ability to remotely create an agent, the Voyager mobile-agent platform could. This little Voyager feature, namely this ability to remotely create agents, effectively allowed mobile agents to outperform the current applet approach in all communication platforms. The fact that this feature is not fully supported by Aglets is immaterial since (a) it can be easily added to it and (b) we are not aiming to show the superiority of Aglets but the power of mobile agents technology.

7. Conclusions

The challenges that a mobile computing model must meet include support for mobility, disconnections, weak connectivity, adaptivity, upward compatibility with existing and legacy applications and support of variable types of mobile units. In this paper, we presented a suite of software models appropriate for mobile wireless computing. These models either build on the traditional client/server model or are based on the new paradigm of mobile agents. These two types of models are not orthogonal, however, but they can be effectively integrated to provide additional flexibility. In fact, we focus on how to materialize all models dynamically via mobile agents. The combination of the mobile-agent model with "traditional" software models for mobile computing gives rise to new software models in which mobile agents play a double role; they participate into these frameworks not only as a computational unit but as a communication mechanism as well.

New frameworks for Web-based distributed access to database systems based on the suggested software models are proposed and implemented. Qualitative and quantitative analysis of the implementation of these frameworks is also presented. The various implementations, each following one of the proposed models built upon mobile agents, indicate: (a) significant performance improvements over the traditional client/server approach, (b) the flexibility of the mobile agent model in dynamically configuring all other models in a cost-effective manner and (c) the potential presented by the mobile agent model(s).

Acknowledgements

The initial implementation of the web-based application is due to S. Papastavrou. The code for implementing each of the frameworks can be downloaded from `http://ada.cs.ucy.ac.cy/~cssamara/dbms-agents`

References

[1] A. Athan and D. Duchamp, Agent-mediated message passing for constrained environments, in: *Proceedings USENIX Symposium on Mobile and Location-Independent Computing*, Cambridge, MA (August, 1993) pp. 103–1070.

[2] B.R. Badrinath, A. Bakre, T. Imielinski and R. Marantz, Handling mobile clients: A case for indirect interaction, in: *Proceedings of the 4th Workshop on Workstation Operating Systems (WWOS-IV)* (1993) pp. 91–97.

[3] D. Barbara and T. Imielinski, Sleepers and workaholics: Caching strategies in mobile environments, in: *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'94)* (1994) pp. 1–12.

[4] M. Breugst, I. Busse, S. Covaci and T. Magedanz, Grasshopper: A mobile agent platform for IN based service environments, in: *Proceedings of IEEE IN Workshop 1998*, Bordeaux, France (1998) pp. 279–290.

[5] D. Chess, B. Grosof, C. Harrison, D. Levine, C. Parris and G. Tsudik, Itinerant agents for mobile computing, Journal of IEEE Personal Communications 2(5) (1995) 34–39.

[6] M. Dikaiakos and G. Samaras, A performance analysis framework for mobile-agent systems, in: *Infrastructure for Agents, Multi-Agent Systems and Scaleable Multi-Agent Systems, Proceedings of the 1st Annual Workshop on Infrastructure for Scaleable Multi-Agent Systems, The Fourth International Conference on Autonomous Agents 2000*, eds. T. Wagner and O.F. Rana, Lecture Notes in Computer Science, Vol. 1887 (Springer, 2001) pp. 180–187.

[7] A. Fox, S.D. Gribble, E.A. Brewer and E. Amir, Adapting to network and client variability via on-demand dynamic distillation, in: *Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems*, Cambridge, MA (1996) pp. 160–170.

[8] R. Gray, D. Kotz, G. Cybenko and D. Rus, Agent Tcl, in: *Mobile Agents: Explanations and Examples, Manning Publishing*, eds. W. Cockayne and M. Zyda (1997). Imprints by Manning Publishing and Prentice Hall, `http://agent.cs.dartmouth.edu/general/agenttcl.html`

[9] J. Gray and A. Reuter, *Transaction Processing: Concepts and Techniques* (Morgan Kaufman, San Mateo, 1993).

[10] R. Gruber, F. Kaashoek, B. Liskov and L. Shrira, Disconnected operations in the Thor object-oriented database system, in: *Proceedings of the Mobile Computing Systems and Applications*, IEEE, Los Alamitos, CA, USA (1995) pp. 51–56.

[11] C.G. Harrison, D.M. Chess and A. Kershenbaum, Mobile agents: are they a good idea?, Research Report, IBM Research Division (March 1995).

[12] B.C. Housel, G. Samaras and D.B. Lindquist, WebExpress: A client/intercept based system for optimizing web browsing in a wireless environment, Mobile Networks and Applications (MONET) 3(4), Special Issue on Mobile Networking on the Internet (1998) 419–431. See also Technical Report CS-TR*96-18, University of Cyprus (December 1996).

[13] IBM Japan Research Group, Aglets workbench, `http://aglets.trl.ibm.co.jp`

[14] B. Jepson, *Database Connectivity: The Lure of Java, Java Report* (Wiley Computer, 1997).

[15] B. Jepson, *Java Database Programming* (Wiley Computer, 1997).

[16] D. Johansen, F.B. Schneider and R. van Renesse, What TACOMA taught us, in: *Mobility, Mobile Agents and Process Migration – An Edited Collection*, eds. D. Milojicic, F. Douglis and R. Wheeler (Addison-Wesley, Reading, MA, 1998), see also `http://www.tacoma.cs.uit.no/`

[17] D.B. Lange and M. Oshima, *Programming and Deploying Java Mobile Agents with Aglets* (Addison-Wesley, Reading, MA, 1998).

[18] D.B. Lange and M. Oshima, Seven good reasons for mobile agents, Communications of the ACM 42(3) (1999) 88–91.

[19] Mobile Agents, `http://www.agent.org/` and `http://www.cs.umbc.edu/agents`

[20] ObjectSpace, Voyager™ technical overview, available at `http://www.objectspace.com/voyager/whitepapers/VoyagerTechOview.pdf`

[21] Oracle, Oracle Mobile Agents Technical Product Summary (June 1997), `http://www.oracle.com/products/networking/mobile/agents/html/`

[22] S. Papastavrou, G. Samaras and E. Pitoura, Mobile agents for WWW distributed database access, in: *Proc. of 15th International Data Engineering Conference (IEEE-ICDE'99)*, Sydney, Australia (March 1999).

[23] E. Pitoura and G. Samaras, *Data Management for Mobile Computing* (Kluwer Academic, Dordrecht, 1998).

[24] P. Reiher, J. Popek, M. Gunter, J. Salomone and D. Ratner, Peer-to-peer reconciliation based replication for mobile computers, in: *Proceedings of the European Conference on Object-Oriented Programming, 2nd Workshop on Mobility and Replication* (June 1996).

[25] G. Samaras, M. Dikaiakos, C. Spyrou and A. Liberdos, Mobile agent platforms for Web-databases: A qualitative and quantitative assessment, in: *The Joint Symposium ASA/MA'99, 1st International Symposium on Agent Systems and Applications (ASA'99), 3rd International Symposium on Mobile Agents (MA'99)*, USA (1999) pp. 50–64.

[26] G. Samaras, E. Pitoura and P. Evripidou, Software models for wireless and mobile computing: Survey and case study, Technical Report TR-99-5, University of Cyprus (March 1999).

[27] G. Samaras and A. Pitsillides, Client/intercept: a computational model for wireless environments, in: *Proceedings of the 4th International Conference on Telecommunications (ICT'97)*, Melbourne, Australia (April 1997).

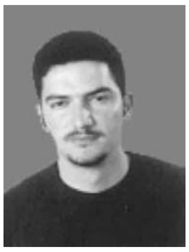
[28] D.L. Tennenhouse, J.M. Smith, W.D. Sincoskie and G.J. Minden, A survey of active network research, IEEE Communications Magazine 35(1) (1996) 80–86.

[29] J.E. White, Mobile agents, General Magic White Paper (1996), `http://www.genmagic.com/agents`

[30] O. Wolfson, P. Sistla, S. Dao, K. Narayanan and R. Raj, View maintenance in mobile computing, SIGMOD Record (1995) 22–27.

[31] D. Wong, N. Paciorek, T. Walsh, J. DiCelie, M. Young and B. Peet, Concordia: An infrastructure for collaborating mobile agents, in: *Lecture Notes in Computer Science*, Vol. 1219 (Springer, Berlin, 1997) `http://www.meitca.com/HSL/Projects/Concordia/`

[32] B. Zenel and D. Duchamp, General purpose proxies: Solved and unsolved problems, in: *Proceedings of the Hot-OS VI* (1997) pp. 87–92.



Constantinos Spyrou is a Ph.D. candidate at the Department of Computer Science, of the University of Cyprus. He received his B.Sc. (May 1999) and M.Sc. (June 2001) in computer science, from the University of Cyprus. He is the recipient of the award for the best Academic progress during the years 1995–1999 that is given jointly by the Computer Science Department (University of Cyprus) and IBM Italia (Cyprus Branch). His research interests include mobile agents technology, Web database connectivity and mobile data management. He has already published a number of papers on these areas in international conferences and journals.
E-mail: cspgcs1@ucy.ac.cy

George Samaras received a Ph.D. in computer science from Rensselaer Polytechnic Institute, USA, in 1989. He is currently an Associate Professor at the University of Cyprus. He was previously at IBM Research Triangle Park, USA and taught at the University of North Carolina at Chapel Hill (adjunct Assistant Professor, 1990–1993). He served as the lead architect of IBM's Distributed Commit Architecture (1990–1994) and co-authored the final publication of the Architecture (IBM Book, SC31-8134-00, September 1994). He was member of IBM's Wireless Division and participated in the design/architecture of IBM's Web*Express*, a wireless Web browsing system. He recently (1997) co-authored a book on data management for mobile computing (Kluwer Academic). He has a number of patents relating to transaction processing technology and numerous technical conference and journal publications. His work on utilizing mobile agents for Web database access has received the best paper award of the 1999 IEEE International Conference on Data Engineering (ICDE'99). He has served as proposal evaluator at a national and international level and he is regularly invited by the European Commission to serve as project evaluator and auditor in areas related to mobile computing and mobile agents. George Samaras has served as program co-chair and program committee member on a number of conferences. He also served on IBM's internal international standards committees for issues related to distributed transaction processing (OSI/TP, XOPEN, OMG).
E-mail: cssamara@cs.ucy.ac.cy

Evaggelia Pitoura received her B.Sc. from the Department of Computer Science and Engineering of the University of Patras, Greece in 1990 and her M.Sc. and Ph.D. in computer science from Purdue University in 1993 and 1995, respectively. Since September 1995, she is on the faculty of the Department of Computer Science of the University of Ioannina, Greece. Her main research interests are data management for mobile computing and network-centric databases. Her publications include several articles in international journals (including IEEE TKDE, ACM Computing Surveys, Information Systems) and conferences (including VLDB, ICDE, ICDCS, CIKM) and a recently published book on mobile computing. She received the best paper award in the IEEE ICDE 1999 for her work on mobile agents. She also co-authored a tutorial on mobile computing presented in IEEE ICDE 2000. Evaggelia Pitoura has served on a number of program committees and was program co-chair of the MobiDE workshop held in conjunction with MobiCom 99.
E-mail: pitoura@cs.uoi.gr

Paraskevas Evripidou was born in Nicosia Cyprus in 1959. He received the Hn.D. in electrical engineering from the Higher Technical Institute in Nicosia, Cyprus in 1981. In 1983 he joined the University of Southern California with a scholarship from the US Agency for International Development. He received the B.S. in electrical engineering, M.S. and Ph.D. in computer engineering in 1985, 1986 and 1990, respectively.

Currently he is an Associate Professor at the Department of Computer Science of the University of Cyprus. From 1990 to 1994 he was on the Faculty of the Department of Computer Science and Engineering of the Southern Methodist University as an Assistant Professor (tennure track). His current research interests are in parallel processing, computer architecture, parallelizing compilers, real-time systems, Java-powered tools for teleworking, parallel processing with mobile agents and parallel input/output and file systems.

Dr. Evripidou is a member of the IFIP Working Group 10.3, the IEEE Computer Society and ACM SIGARCH. He is also a member of the Phi Kappa Phi and Tau Beta Pi honor societies. He was the Program Co-Chair and General Co-Chair of the International Conference on Parallel Architecture and Compilation Techniques in 1999 and 1995, respectively.
E-mail: skevos@cs.ucy.ac.cy