Software agents as a versatile simulation tool to model complex systems

FILIPPO NERI
University of Naples "Federico II"
Department of Informatics and System Science
via Claudio 21, 80125 Napoli (NA)
ITALY
filipponeri@yahoo.com http://www.docenti.unina.it/

Abstract: In the paper we will describe how the software agent paradigm can be a powerful and versatile simulation tool to model and study complex systems. Software agents allow to approximate the behaviour of complex systems under several scenario conditions. In some cases, software agents can also be used to approximate solutions to difficult problems occurring in complex systems. To support our position, we will introduce two different domains: consumer decisions in fast moving consumers goods and mobile telecom network management applications. Finally we will show how a software agent modeling approach could be used to study their behaviour in various scenarios.

Key–Words: Software agents, Consumer markets, Telecom network management systems, Cellular networks, Distributed simulation systems.

1 Introduction to the simulation of complex systems

Software agent paradigm can be a powerful and versatile simulation tool to model and study complex systems, allowing to fast prototype approximate solutions to difficult problems and to easily select which ones to take forward at the deployment stage. To support our position, we will introduce two different domains: consumer decisions in fast moving consumers goods and mobile telecom network management applications. Then we will shows two interesting research problems faced today by the experts in those fields. And we will show how a software agent modeling approach could be used to increase the understanding of the problems, their modelization, and the proposal of alternative solutions. In particular, in this paper we expand our ideas initially exposed as an invited talk in one of WSEAS conferences [12] and, we begin by describing how an agent based tool for analysing markets behaviour, under several rate of information diffusion, can be developed. This methodology has allowed for the study of tradeoffs among several variables of information like product advertisement efforts, consumers' memory span, and passing word among friends in determining market shares.

In addition, concerning a mobile telecom domain, we will propose how to use software agent simulation to study the impact of moving from a centralized network management architecture toward a distributed one, where each network management application consists of a controller part and a set of distributed parts running on the individual network elements.

2 Simulating a consumer market

The diffusion of an Internet based economy, that includes even the less valuable transactions, is day by day more evident. The existing information infrastructure has allowed the exploitation of new methods to contract the purchases of goods and services, the most notable of which is probably the agent mediated electronic commerce [8, 11]. In this economy, autonomous agents become the building block for developing electronic market places or for comparing offers across several seller's websites (shopbots) [11, 15]. Our aim is to develop an agent-based market place to qualitatively simulate the diffusion of products' awareness across the Internet and its impact on customer choices. As many commercial scenarios could be selected, we chose to investigate a simple commercial interaction. Different groups of consumers have to choose one product between a set of perfect substitutes that differ in price, advertised lifestyle associated with the product and the advertising effort to initially penetrate the market. Our objective is the to understand how a sequence of repeated purchases is

affected by the trade off among the previous variables, the consumers' desires and limits, and the diffusion of the awareness about the existing products. The ultimate goal would be to capture the common experience of choosing, for instance, among alternative brands of Italian Pasta packages displayed in the webpage or on the physical shelf of our grocery store.

Some researchers take a very long term view about the ecommerce phenomena envisioning economies of shopbots [8, 11, 15]. We try to capture the commercial phenomena in more near future where customers are human beings with their intrinsic limit in information processing, having the need to trust the bought product and to feel supported, and reassured about their purchasing choice as their best possible choice. We share, however, with Kephart et al. the desire to analyse and understand how the information flow can affect such economy. Academics in business schools already report preliminary studies of these situations. For instance, Lynch and Ariely [6] try to understand the factors behind purchases made in a real world experiment of wine selling across different retailers' websites and Brynjolfsson et al. [16] discuss which factors seems to be more likely to impact the consumer choices in the electronic market place. To further extend our work, a more sophisticated approach to modelling the electronic market place may have to be selected in order to take into account negotiation protocols or virtual organisation formation as, for instance, described in [14] or to account for additional brokering agents as described in [17].

3 The Virtual Market Place

The architecture of the agent based virtual market place is quite simple: one purchaising round after the other, groups of consumers, modelled as software agents, select which product to buy according to their internal status. The internal status takes into account the consumers' preferences for a product and her awareness about the product's benefits and image. This process based description of the buying experience matches what most people experience when selecting among alternative wholemeal breads or milk chocolate bars at the local grocery store [2]. In the simulator we represent both products and consumers as software agents. A product is a collection of an identifier, a price, an effort to describe its features/benefits on the package, an effort to bound the product to the image of a lifestyle (brand) and an initial advertisement effort to penetrate the market. It is important to note that the scope of this work is to consider products that are substitute one for the others but differ in price or other characteristics. The idea to model products as software agents is new.

A consumer is a (software) agent operating on the market and driven in her purchases by a target price, a need for understanding the product benefits, the lifestyle conveyed by the product brand, and the initial marketing effort put into placing the product in the market. The consumer can remember only a constant number of products (memory limit) for a constant number of rounds (memory duration), and she may share with her friends her opinion about the known products. It is worthwhile to stress that the memory span limits the consumer awareness of the available products. For instance, if a consumer had a memory limit of 3, she would be aware of 3 products at most and she would make her choice only among those three products. A consumer will not remember a product, if its memory has already reached its limit, unless it is better of an already known product thus replacing it. However, round after round, consumers talk to each other and they may review their opinions about the products by updating their set of known products. Our interest lays in forecasting the product market shares (percentage of bought products) on the basis on the previous factors. In order to evaluate the feasibility of our approach, we developed from scratch a basic version of the market place simulator and performed some experimentation under constrained conditions.

In the following the detailed descriptions of both the simulator's architecture and the experimental setting is described. In the simulator, each product is defined by an identifier (Id), a selling price (Price), an effort in describing its benefits on its package, an effort to convey a lifestyle (image), and an effort to initially penetrate the market. As an instance, in the initial series of experiments, all the products prices and characteristics are selected to cover a wide range of significant offers as follow:
Product(Id, Price, Description, Image, InitialAdvertisement)
Product(0, LowValue, LowValue, LowValue, LowValue)
Product(1, LowValue, LowValue, LowValue, HighValue)
Product(2, LowValue, LowValue, HighValue, LowValue)
...
Product(15, HighValue, HighValue, HighValue, HighValue)
The constants 'LowValue' and 'HighValue' correspond to the values 0.2 and 0.8. The Price, Description and Image parameters are used to evaluate a customer's preference for the product, whereas the InitialAdvertisement parameter defines the initial aware-

ness of the product among the customers. So, for instance, a product defined as Product(x, LowValue, LowValue, LowValue, LowValue) is especially targeted toward price sensitive consumers that do not care about knowing much on the product. And with an initial penetration rate of 0.2, on average, 20% of the consumers are aware of its availability at the beginning of the first buying round. Finally, it is worthwhile to note that, in the above list, odd and pair numbered products differ only because of a different initial advertising effort.

A similar representation choice has been made to represent customers. Four groups of consumers are considered. For the scope of the initial experiments, we concentrate on customers whose target product has a low price but differs in the other features. Consumer groups are represented as follows:
Customer(Price, Description, Image)
Customer(LowValue, LowValue, LowValue) (bargain hunters)
Customer(LowValue, LowValue, HighValue) (image sensitive
Customer(LowValue, HighValue, LowValue) (description sensitive
Customer(LowValue, HighValue, HighValue) (image and description sensitive
Through the selection of target values, we tried to capture the following categories of customers: the bargain hunters, the brand sensitive ones, the package sensitive ones (i.e. are interested in its nutrition values, its composition, its ecological impact, etc.), and those that are both brand and package sensitive. It is important to note that each customer does not necessary known the same products than other consumers because of the individual memory and of the initial random distribution of a product awareness among consumers. During each round, a consumer chooses to buy the product that most closely matches her preferences.

According to Bettman [2] and [5], we approximate the product matching process by means of a weighted average function defined as follows:
$Preference(product) =$
$(max(product.Price, target.Price) -$
$target.Price)^2 +$
$(min(product.Description, Description) -$
$target.Description)^2 +$
$(min(product.Image, target.Image) -$
$target.Image)^2$

The preferred and selected product is the one with the lowest value of the Preference function among the ones known by the customer. Alternative expressions are under study.

Also each customer does not necessarily known the same products than the others because of the different distribution of the products depending on their initial marketing effort. The reported experiments aim to understand the impacts of the following factors in determining the final product market shares: customer preference definition, initial market penetration effort, number of friends in passing the word of known products, and memory limit.

In the initial group of experiments we aimed to investigate some hypothesis on the impacts of the diffusion of product awareness and shift in the consumers' behaviours. The obtained results are promising and confirm the feasibility of the approach. They are however far from being conclusive in term of hypothesis testing. Indeed in order to perform extensive and informative experiments, the virtual market place simulator should be completely re-engineered to facilitate its use and the definition of hypotheses/rules governing the consumers' behaviour.

4 Experimental Results

Goal of the experimentation is to show that our tool can capture some of the inherent complexity behind the determination of the product market shares by considering a variety of factors that impact on this economic phenomena. These factors include the customers' expectations for a product, the limited memory span and duration that consumers reserve to remember available products, and the diffusion of the product awareness among consumers by initial advertisement and further passing by word. Value ranges for this variables have been selected accordingly to past experience with consumers behaviour.

All the reported experiments refer to an hypothetical economy and are based on the following basic settings. During each round, 400 consumers (one hundred for each of the four consumer types) select which of the 16 products to buy. Only products that the consumer remembers (i.e. appearing in its memory list) compete for being purchased. The economic process is repeated for 100 rounds. For each experiments, the reported figures are averaged over 3 runs.

As a baseline for evaluating the economic process, we consider the situation where each consumer is fully aware of all the available products since the first round. As all the consumers are oriented towards products with low price but with different characteristics, it is straightforward to calculate that the product market shares stay constant over the 400 rounds and correspond to the values reported in Fig. 1. In the figure, ideal market share distribution in presence of a perfect product awareness or perfect information flow is shown. In the picture, the product's identifiers appear on the x axis, and the market shares on the y axis.

Thus for instance, Product 6 will achieve a 9.3% market share. It is worthwhile to note that the product from 9 to 16 have a 0% market share because, in the range from 1 to 8, there exists a product with identical features but with lower price.

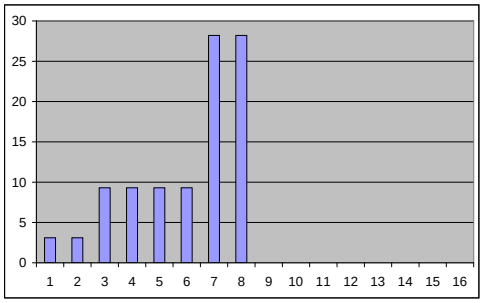

Figure 1: Ideal market share distribution.

If we were in this ideal situation, every consumer would be able to make the best pick among the available products. Unfortunately, in the real world, full knowledge about the available choices is not common and product awareness is the results of a variety of factors including advertisement, passing by word among friends and memory capacity. The impact of these factors on the product market shares is taken into account in the following experiments.

Let us consider the case where consumers do not have any friends or do not talk about products to friends (average number of friends or avgf =0), they can remember only 2 products at the time (memory limit or ml=2), and they remember each product for 20 rounds unless either they keep buying it or they are told about by their friends. The initial (end of round 1) and final market shares (end of round 100) appear in Fig. 2. In the picture, market shares when consumers do not talk each other (avgf=0) and remember at most

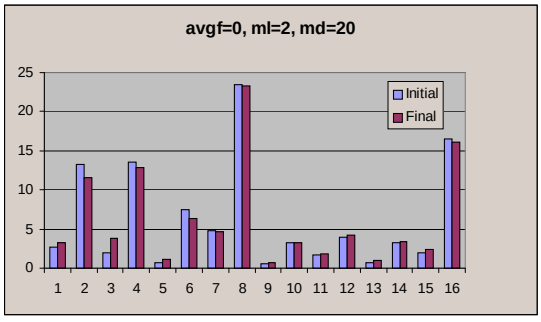

Figure 2: Market shares when consumers do not talk each other.

It appears that the initial and final market shares are very alike and that the higher the effort in penetrating the market the better the market share (compare odd and even numbered products). The market share distribution is biased toward low priced product, this is to be expected given the customers' preferences. But, still, some high price products achieve a significant portion of market because of the limited memory span of the consumers that would prevent him to compare and choose among more alternatives.

If we alter the previous scenario just by increasing the number of friends to 20, we obtain quite a different distribution of market shares, Fig. 3. In the picture, market shares when consumers talk to about 20 friends (avgf=20) and remember at most 2 products (ml=2) for 20 rounds, are shown. The pattern of the

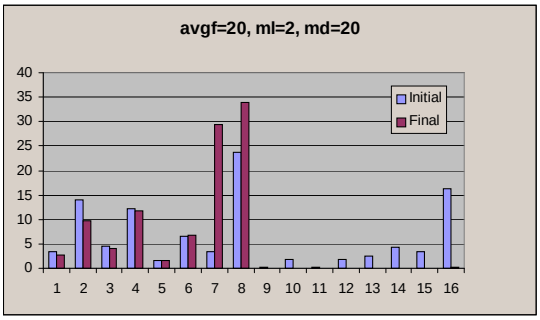

Figure 3: Market shares when consumers talk to each other.

initial market shares is, of course, similar to that of the previous scenario but the final shares tends to converge towards the ideal ones. This can be interpreted that having many friends or collecting many opinions among the same market does actually empower the customer in making the best selection. It is interesting to note that the only initial advertisement cannot compensate for the further product comparisons communicated among the consumers. However, the initial product advertising effort results in the consumers remembering and, then, choosing the more advertised products among the low priced ones.

From additional experiments not reported here, comparing the initial and final distributions of market shares it appears that exchanging information about products with friends and remembering a number of them is the key to make a successful choice in this scenario. Indeed this observation is at the very base for the development of several strategies to deal with comparative on-line shopping.

5 Simulating mobile telecom architectures

The research problem is how to measure and estimate the resource requirements and performances of deploying distributed applications into the network management (NM) system of a mobile telecommunications.

In order to provide some background knowledge about mobile networks and their network management systems, we will start to introduce a simplified model of a mobile network. We present in the next section a more detailed description of the main components of a 2G/3G mobile network and how they cooperate to provide user services, such as voice calls and data transfer. From the point of view of our research, a telecom mobile network can be viewed as a set of several types of network elements responsible for routing the voice or data traffic across the network. In particular for this research, we focus our attention to one Network Element: the OSS (Operation Support System) node and on the set of Network Management applications running on it. The OSS system assists the network operator (for example, a company such as Vodafone, TMobile, etc.) in configuring and monitoring the operation status of each network element and of the network as a whole. Among other activities, the OSS constantly updates a centralised database recording the network configuration, by collecting data from the network elements, and by propagating changes across the network when a new configuration is defined. State-of-the-art commercial telecom networks adopt a centralized solution which resides and runs on the OSS node. As can be easily understood with the centralized approach, the OSS node become the focus of high computational activity and of intense message traffic which in turn require the provision of expensive hardware and large amounts of bandwith. As an example, status data from the Network Elements is copied to the OSS in order to be manipulated and any new configuration data is pushed back to the Network Elements affected by the configuration change. This way of dealing with Network Management tasks results in scalability issues, in term of number of Network Elements that can be controlled by one OSS, and potentially problems also with data consistency, as Network Element data may have to be duplicated across the network. Therefore the centralized approach can result in the OSS having to operate with an outdated picture of the network, which may then result either in the propagation of configuration errors when, for example, new Network Elements have to be included into the network, or it may result in the inability of the network manager to detect when a customer service, (such as video call), is not working properly because of a technical fault somewhere in the network. A potential solution approach to the outlined problems could be to decentralise or distribute more of the Network Management functionality directly to the network elements. This would have the advantage of dealing with configuration changes at, or close to, the data source, thus reducing issues with data consistency, while at the same time freeing computational resources on the OSS node, and reducing bandwith requirements to/from the OSS node, and allow for a more scalable network architecture supporting a greater number of Network Elements being managed from one OSS instance.

While such suggestion is easy to propose, the difficulty lies in proving that the suggestion can actually work and that actually improve the NM operations while at the same time ensuring that the performance of the network, in carrying traffic, is not adversely affected.

It is then with respect to this last point that a software agent simulation methodology will come handy as a simulation environment for novel software architecture.

In the following section we will review the tools available for simulating or modeling part of the telecom wireless network and their limitations.

6 Related works

In [18], the authors show that predictive algorithms can be successfully employed in the estimation of performance variables and the prediction of critical events in system management tasks.

In [3], the authors describe an off-line modeling tool able to predict the impact of changes to a network's topology, configuration, traffic, and technology.

In [10], Mobile Ad hoc NETworks (MANETs) are studied. MANETs are dynamic networks populated by mobile stations.

Ns-2 [13] is the de facto standard for network simulation. We are considering to use the ns-2 simulator [13] for some advanced stage work on mobile network modeling.

GloMoSim [20] from UCLA is the second most popular wireless network simulator. Lack of documentation makes difficult to adopt and exploit the simulator.

The OPtimized Network Engineering Tools, OPNET, [4] is a network simulator proposed by MIT in 1986. It is a well-established and professional commercial suite for network simulation.

Researchers at IBM [19, 9] report two approaches to measure and predict performance behaviour under

growing workload for a centralized system.

7 Distributing NM to deal with new services and more nodes in telecom wireless networks

As mentioned earlier, the current centralized NM system may run into problems when attempting to scale up when new customer services and network nodes will be deployed across a mobile network. This consideration has prompted a research activity to find alternative architectures that could overcome the limits of the current centralized approaches.

A straightforward idea to deal with increasing network size and complexity is to move from a centralized NM system to a distributed/decentralized one where each NM application previously residing in the OSS node is broken into a controller part running on the OSS node and a distributed part running on each affected network element. While such suggestion is easy to promote, the difficulty lies in proving that the suggestion can actually work and that actually improve the NM operations while at the same time ensuring that the performance of the network, in carrying traffic, is not adversely affected. In the following section we will review the tools available for simulating or modeling part of the telecom wireless network and their limitations.

8 Selected mobile network simulators

In our research, we would like to be able run code both on NEs and on the OSS node, and we then would like to use an experimental mobile network to verify the effect produced by our code. This is easier said that done as our experience proved.

Accessing an experimental mobile network is not an easily achievable task as priority is obviously given to the commercial production projects. Therefore we decided to run our experiments on the JADE platform [1, 7] and then to move the code to the experimental mobile network when the procedures and the code for the experiments are tuned.

The JADE (Java Agent DEvelopment Framework) is a java based middleware that simplifies the implementation of multi-agent systems. We can use it as we have based the development of our distributed NM applications on Object Oriented programming and the Software Agent paradigm. It is therefore straightforward to prototype alternative versions of an application in JADE and simulate the distributed NM system of a mobile network by using a network of cooperating agents where each agent implements one NE.

9 Identifying relevant Performance measures

In order to measure the impact of deploying distributed applications in a mobile network, a set of measures has to be defined which should allow to identify requirement for relevant limited resources. For the scope of this research, we have defined the following measures as relevant, but the list can be extended on a case by case need:

1. CPU usage - average CPU usage by the decentralized application over time
2. Memory utilization - average memory usage over time
3. Storage - maximum amount of hard disk space needed
4. Bandwith - average bandwith utilization.

In particular, we are interested in monitoring variations in those measures under different load conditions and ideally we would like to be able to estimate how an application will perform under unseen load conditions (number of nodes in the network).

10 The PIRR estimation methodology

Given our research problem, we would like to develop a performance and resource estimation methodology which could predict the impact of deploying a distributed application, NewApp, into the mobile network. Because of the complexity of the problem we are dealing with, and the limits of the available simulators, we decided to base our estimation methodology on a combination of empirical and analytical practices.

We will call our methodology PIRR (pronounced as the word 'peer') from Performance Indexes and Resource Requirements estimation, and it consists of a two step approach.

In the **first phase**, we will run NewApp in our simulation environment and collect as much relevant performance data as possible by using profiling tools. Unfortunately not all the information we are interested in is ready available from a profiling tool.

In the **second phase**, we will analyze the code of NewApp in order to understand what part of the algorithm affects the application's performances. On the

base of this analysis, we will develop a mathematical model of NewAppl's performances that should be able to explain the set of observed performance data and should allow to estimate the PIRR values when NewApp runs in different network loads.

We will show on an example how the PIRR methodology can be applied.

11 The PIRR methodology applied to the OK-PING service

In order to show the PIRR methodology at work, we selected two versions of the simplified OK-PING application.

In the rest of the section, we will described the two versions of the OK-PING application and we will show how we were able to collect and estimate PIRR data by using our methodology.

The simulation environment for the reported experiments is based on JADE running on JVM 1.5. In JADE, each NEs and the OSS node run as software agents implemented as separate Java threads.

11.1 Centralized OK-PING

The OK-PING application in real mobile network has a centralized architecture with the OSS node initializing the communications towards to NEs and then collecting their replies. If a NE does not reply before a time out period, a fault management procedure is activated. The abstract code of the application parts running on the OSS node and on each NEs follows:

```
// Location: OSS node; Application: OK-PING.
   Every t seconds do {
      for any NE in the mobile network
         send a OK-PING request
      NEset={the set of all the NEs in the network}
      while ((some timeout is not exceed) and
            not(empty(NEset))) {
         collect OK-PING replies from any NE
         remove from NEset the NE who replied
      }
      if not(empty(NEset))
         {some NEs cannot be reached,
         start fault management process }
   }


// Location: NE; Application: OK-PING.
   Every t seconds {
      prepare status data
      wait to receive a OK-PING request from OSS
         send OK-PING reply to OSS with status data
   }
```

For our work, the message size of a OK-PING request is 10 bytes, an OK-PING reply consists of 60 bytes and the interval of time, t, is set to 60 seconds. The values of these parameters in real mobile networks depends on the chosen configuration.

We are interesting in measuring PIRR data for the concentrated OK-PING application when different numbers of NEs are in the network. Those data will provide information about the scalability of the application. We will also use the collected PIRR data to compare them against a different implementation of the same application, reported below, where a more distributed architecture is adopted.

By looking at the code, we can calculate that for each NE two messages have to be exchanged (the request and reply ones) between the OSS node and the NE to accomplish the task.

11.2 Decentralized OK-PING

This version of the OK-PING application is experimental and is not deployed on real mobile network. In this implementation, each single NE has the liability to periodically update the OSS node about its status by using a 'fire and forget' approach. If no communication is received by every NEs before a timeout expires, than a fault management procedure is activited for those NEs who have not sent the 'ok-ping' message. In this application, the OSS node periodically listen to the incoming communication and update its network status.

The abstract code of the application part running on the OSS node and on each NEs and follows:

```
// Location: OSS node; Application: OK-PING.
   Every t seconds do {
      NEset={the set of all the NEs in the network}
      while ((some timeout is not exceed) and
            not(empty(NEset))) {
         collect OK-PING replies from any NE
         remove from NEset the NE who replied
      }
      if not(empty(NEset))
         {some NEs cannot be reached,
         start fault management process }
   }


// Location: NE; Application: OK-PING.
   Every t seconds {
```

```
    prepare status data
    send OK-PING message to OSS with status
}
```

For our study, the size of a OK-PING message is 60 bytes and the interval of time, t, is set to 60 seconds.

Also by examining the code, it appears that for each NE only one message is needed to inform the OSS node that the NE is working properly.

11.3 The PIRR methodology applied to the centralized OK-PING

Ten experiments per each network configuration (using a simulated network of 20, 40, 80 NEs) have been run to average the collected PIRR data by using the concentrated version of the OK-PING application.

Applying the PIRR methodology
In order to collect some of the PIRR data we are interested in, we have used NetBeans 5.5 Profiler. The profiler application allows for measuring the CPU time used by every NEs and by the OSS in networks with a different number of NE elements.

We then performed a linear interpolation in order to estimated how much the OSS node's CPU usage would vary depending on the number of NEs present in the network. This simple calculation, allows us to represent PIRR data in a parametric form which is useful when predictions for unseen condition have to be made.

In order to measure the memory utilization, the information provided by the profiler is not directly useful as only the total memory used (Heap and Stack) by the JVM is reported. So we have estimated the memory used by each NE node to run the OK-PING application, by averaging the total memory variations, when running the simulator with different NE load, over the number of NEs in the network. In practice, we used the following formula:

```
((TMU(80) - TMU(40))/40 + (TMU(40) +
- TMU(20))/20)/2
```

where TMU(n) stands for Total Memory Used (averaged over the 10 run experiments) when the number of NEs in the network is n.

The bandwith required by the application has been estimated by performing a code analysis, calculating the total byte size of the messages exchanged over a period of time and dividing the total byte size per the seconds in the time period. A communication set up overhead, whose value we were not able to measure, has to be added.

The storage requirement have been calculated by summing up the size of the compiled java file (class files) that make up the application plus estimating the size of temporary file used by the applications. No temp files are used by the studied applications.

Reporting PIRR data
In the table below, PIRR data for a configuration with 40 NEs reporting to the OSS are reported in a parametric format:

	OSS (40 NEs in the net)	1 NE
CPU	0.02% * 40	0.02%
Memory	α	35 KB
Bandwith	6.4 B/s + (40 * conn set up) + 60 B/s + (40 * conn set up)	0.16 B/s + conn set up + 1 B/s + conn set up
Storage	9 KB	6 KB

Note that the Memory need (α) for the OSS node cannot be estimated. We know, by the code analysis, that the OSS node will use a fixed amount of memory for its internal objects and that amount will not change during runtime. If we run the system with only the OSS system to measure the total heap memory use (inclusive of JADE environment) we notice that it stays constant over time and does not exceed 700KB. That means that the memory usage for the OSS node is stable and its upper bound is 700KB. Further if we inspect the application code running on the OSS and we compare it with the code running on a NE, then a better estimate for an upper bound for the OSS node memory need is 50KB.

It is important to note that the PIRR data are reported in a parametric format obtained by combining the code analysis step with the experimental phase of the PIRR methodology. We will then be able to use the parametric data to estimate PIRR values for the application when it might run in different configuration settings.

11.4 The PIRR methodology applied to the distributed version of OK-PING

Again by using the distributed OK-PING application, we run ten experiments per each network configuration (using a simulated network of 20, 40, 80 NEs), and the averaged PIRR data have been collected.

In the table below, parametric PIRR data for a mobile network with 40 NEs reporting to the OSS are shown:

	OSS (40 NEs reporting to it)	1 NE
CPU	0.02% * 40	0.02%
Memory	β	35 KB
Bandwith	60 B/s + (40 * conn set up)	1 B/s + conn set up
Storage	5 KB	5 KB

As in the previous subsection, the reported values are parameter based, so they show the relationships between the application cost drivers and the number of

NEs in the network. As obvious, they could be used to estimate PIRR data for the OK-PING, distributed version, under different conditions.

11.5 Two ways to exploit PIRR tables

PIRR tables, associated with information about the network configuration, could be used

1. to compare alternative versions of an application at the design time to decide which one to further engineer, or

2. to decide at runtime if or not to run a distributed application by dynamically estimating its resource requirements.

As an example of the first use of PIRR tables, comparing alternative implementation of the same application, the results reported in the above tables could be used to support the case to substitute the decentralized version of the OK-PING application for the concentrated one. The decentralized version requires less communication bandwith, as less messages are exchanged, and thus could help reducing the amount of network management traffic running through the mobile network. Also the storage requirements for the two versions, show that the distributed version can be stored in a smaller bytecode file.

Coming to the second use of PIRR tables, any application to be deployed into the mobile network could have associated a parametric table calculated a priori in a simulator. Then depending on overall load condition on the mobile network, a controller process could estimate if the mobile network has enough resources to run the application, and decide if to grant permission to execute to the application or postpone its execution when enough resources are available. This usage of the parametric PIRR table could support the move from a centralized and static NM approach, such as the one currently used in commercial systems, to a distributed and dynamic one where the NM system could deploy across the mobile network just the right combination of NM applications, from a portfolio of many, depending on the available resources and with a reasonable certainty that they would run as required and not adversely affect the network.

The control at runtime over which applications are allowed to run or not, could allow to reduce both the economic investment in network hardware and bandwith, and the interference in the core network between payload traffic and NM traffic.

12 Conclusions

The results from our research support the use of a simulation methodology based on software agents to approximate the behaviour of complex systems, such as, for instance, economics markets and mobile telecom networks. An important finding, when exploiting our simulation methodology, is that alternative solutions to difficult problems occurring in the domains of interests may be approximated in the simulated environment with relatively low development cost. In case, the empirical results from the simulations were good enough, the approximate solutions could be further developed and adopted in the real world version of the domain to solve the original problem. We have discussed with examples how such an approach could be employed in the domains of consumer decisions in fast moving consumers goods and mobile telecom network management applications. We believe that the description of the use of the software agent based simulation in the two case studies has allowed the reader to appreciate how flexibile such a methodology can be and how to practically use it in real world domains.

References:

[1] F. L. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent Systems with JADE*. Wiley, 2007.

[2] J. Bettman. *An information processing theory of consumer choice*. Addison Wesley, Reading, MA (USA), 1979.

[3] Cisco Systems. Using cisco network planning solution for capacity planning and optimization. Technical report, Cisco Systems, 2006. $http : //www.cisco.com/en/US/products/$.

[4] F. Desbrandes, S. Bertolotti, and L. Dunand. Opnet 2.4: An environment for communication network modeling and simulation. In S. for Computer Simulation, editor, *European Simulation Symposium*, pages 64–74, 1993.

[5] W. Hoyer. An examination of consumer decision making for a common repeat purchase product. *Journal of Consumer Research*, 11:822–829, 1988.

[6] J. J. G. Lynch and D. Ariely. Wine online: search costs and competiotion on price, quality and distribution. *Marketing Science*, pages 1–39, 2000.

[7] JADE authors. *JADE - Java Agent DEvelopment Framework*, 2007. Available at http://jade.tilab.com/.

[8] J. O. Kephart, J. E. Hanson, D. W. Levine, B. N. Grosof, J. Sairamesh, R. Segal, and S. R. White. Dynamics of an information-filtering economy. In *Cooperative Information Agents*, pages 160–171, 1998.

[9] S. R. Kunkel, R. J. Eickemeyer, M. H. Lipasti, T. J. Mullins, B. O'Krafka, H. Rosenberg, S. P. VanderWiel, P. L. Vitale, , and L. D. Whitley. A performance

methodology for commercial servers. *IBM Journal of Research and Development*, 44(6):851–872, 2000.

[10] P. B. Luc Hogie and F. Guinand. An overview of manets simulation. *Electronic Notes in Theoretical Computer Science*, 150:81–101, 2006.

[11] P. Maes. Agents that reduce work and information overload. *Communications of the ACM*, pages 31–40, 1994.

[12] F. Neri. The software agent paradigm as a simulation tool to model complex systems. In *7th WSEAS Conference on System Science and Engineering Simulation*, pages 159–163. Wseas Press, 2008.

[13] ns2. The network simulator. ns-2., 1999. http://www.isi.edu/nsnam/ns.

[14] A. P. Rocha and E. Oliveira. Agents advanced features for negotiation in electronic commerce and virtual organisations formation process. *Agent Mediated Electronic Commerce - An European Perspective*, LNAI 1991, 1999.

[15] J. A. Rodriguez-Aguilar, F. J. Martin, P. Noriega, P. Garcia, and C. Sierra. Towards a test-bed for trading agents in electronic auction markets. *AI Communications*, 11(1):5–19, 1998.

[16] M. D. Smith, J. Bailey, and E. Brynjolfsson. Understanding digital markets: review and assessment. *Draft available at http:ecommerce.mit.edupapersude*, pages 1–34, 2001.

[17] M. J. Viamonte and C. Ramos. A model for an electronic market place. *Agent Mediated Electronic Commerce - An European Perspective*, LNAI 1991:3–28, 1999.

[18] R. Vilalta, C. Apté, J. L. Hellerstein, S. Ma, and S. M. Weiss. Predictive algorithms in the management of computer systems. *IBM Systems Journal*, 41(3):461–474, 2002.

[19] E. Weyuker and A. Avritzer. A metric for predicting the performance of an application under a growing workload. *IBM Systems Journal*, 41(1):45–54, 2002.

[20] X. Zeng, R. Bagrodia, , and M. Gerla. Glomosim: A library for parallel simulation of large-scale wireless networks. In *Workshop on Parallel and Distributed Simulation*, pages 154–161, 1998.