Service Decomposition and Task Allocation in Distributed Computing Environments

Malamati Louta and Angelos Michalas
Department of Business Administration
Technological Educational Institute of Western Macedonia
Koila, Kozani, 50100, Greece, louta@kozani.teikoz.gr
Department of Information and Communication Technologies Engineering
University of Western Macedonia
Department of Informatics and Computer Technology
Technological Educational Institute of Western Macedonia
Fourka, Kastoria, 52100, Greece, amichalas@kastoria.teikoz.gr

Abstract. Highly competitive and open environments should encompass mechanisms that will assist service providers in accounting for their interests, i.e., offering at a given period of time adequate quality services in a cost efficient manner. Assuming that a user wishes to access a specific service composed of a distinct set of service tasks, which can be served by various candidate service nodes, a problem that should be addressed is the allocation of service tasks to the most appropriate service nodes. This scenario accounts for both the user and the service provider. Specifically, service providers succeed in efficiently managing their resources, while users implicitly exploit in a seamless way the otherwise unutilized power and capabilities of the provider's network. In general, service task allocation is founded on general and service specific user preferences, service provider's specific service logic deployment and current system & network load conditions. The pertinent problem is concisely defined, mathematically formulated, optimally solved and evaluated through simulation experiments.

1 Introduction

The main role of all players in the liberalised, deregulated and competitive telecommunication market is to constantly monitor the user demand, and in response to create, promote and provide the desired services and service features. In

Please use the following format when citing this chapter:

Louta, M., Michalas, A., 2007, in IFIP International Federation for Information Processing, Volume 247, Artificial Intelligence and Innovations 2007: From Theory to Applications, eds. Boukis, C., Pnevmatikakis, L., Polymenakos, L., (Boston: Springer), pp. 81-91.

accordance with a business model applying to the telecommunications world, five main different entities can be identified, namely, user, service provider, (third party) application (content) provider, broker and network provider. The role of the (third party) application (content) provider is to develop and offer applications (content). The role of the service provider is to provide the means through which the users will be enabled to access the applications (content) of (third party) application (content) providers. The broker assists business level entities in finding other business entities. Finally, the role of a network provider is to offer the network connectivity needed for service provision.

Service provisioning in such open models is a quite complex process since it involves various diverse actors. The following are some key factors for success. First, the efficiency with which services will be developed. Second, the quality level, in relation with the corresponding cost, of new services. Third, the efficiency with which the services will be operated, controlled, maintained, administered, etc. The challenges outlined above have brought to the foreground several new important research areas. Some of them are the specification of service architectures (SAs) [1,2], the development of advanced service creation environments (SCEs) and grid computing architectures [3,4] and service characteristics (e.g., the personal mobility concept), and the exploitation of advanced software technologies, (e.g., distributed object computing [5] and intelligent mobile agents [6]). The aim of this paper is, in accordance with the cost-effective QoS provision and the efficient service operation objectives, to propose enhancements to the sophistication of the functionality that can be offered by service architectures in open competitive communications environments.

In accordance with the SA concept and exploiting advanced software paradigms, the service logic is realised by a set of autonomous co-operating components, which interact through middleware functionality that runs over Distributed Processing Environments (e.g., Common Object Request Broker Architecture - CORBA). Limited by techno-economic reasons or considering administrative, management and resilience/ redundancy purposes it is assumed that each service provider deploys service components realising service logic in different service nodes, residing in the same and/or different domains. Moreover, it can be envisaged that a service will in general comprise a set of distinct service tasks, which could be executed by different service nodes.

Highly competitive and open environments should encompass mechanisms that will assist service providers in accounting for their interests, i.e., offering at a given period of time adequate quality services in a cost efficient manner which is highly associated to efficiently managing and fulfilling current user requests. Thus, assuming that a user wishes to access a specific service composed of a distinct set of service tasks, which can be served by various candidate service nodes (CSNs), a problem that should be addressed is the allocation of service tasks to the most appropriate service nodes. In this paper, the pertinent problem is called service task allocation. The aim of this paper is to address the problem from one of the possible theoretical perspectives and to show the software architecture that supports its solution and how it can be incorporated in service architectures that run in the open environment.

In general, service task allocation is founded on general and service specific user preferences, service provider's specific service logic deployment and current system & network load conditions. A high level problem statement may be the following. Given the set of candidate service nodes and their layout, the set of service tasks constituting the required service, the resource requirement of each service task in terms of CPU utilization, memory and disk space, the cost of deploying each service node, the current load conditions of each service node and of the network links, find the minimum cost assignment of tasks to service nodes (in terms of the number of nodes that need to be deployed, the communication cost introduced during the execution of service tasks, and the management cost imposed by the arrangement) subject to a set of constraints, associated with the capabilities of the service nodes.

The approach in this paper is the following. The starting point (section 2) is the service task allocation architecture, presenting the software elements required for the realisation of the assignment process. Additionally, our assumptions regarding the model of service provisioning system are presented. Section 3 presents a concise definition, mathematical formulation and optimal solution of the service task allocation problem, while one possible formulation of the communication cost taken into account in our framework is provided. Section 4 gives a set of experimental results, indicative of the efficiency of the proposed service task assignment scheme. Finally, section 5 gives future plans and concluding remarks.

2 Service Task Allocation Architecture

Service task assignment process, as a first step, requires a computational component that will act on behalf of the user. Its role will be to capture the user preferences, requirements and constraints regarding the requested service and to deliver them in a suitable form to the appropriate service provider entity. As a second step, service task allocation requires an entity that will act on behalf of the service provider. Each role will be to intercept user requests, acquire and evaluate the corresponding service node and network load conditions, and ultimately, to select the most appropriate service nodes for the realisation of the service. Furthermore, a monitoring module is required. Monitoring module consists of a distributed set of agents, which run on each service node of the service provider. Each agent is responsible for monitoring the load conditions and available resources of the service node and delivering them to the service provider related entity. Additionally, a distributed set of network provider related entities will be responsible for providing the service provider entity with network load conditions and managing the network connections necessary for the service provision.

The following key extensions are made so as to cover the functionality that was identified above. First, the *Service Provider Agent* (SPA) is introduced and assigned with the role of selecting on behalf of the service provider the best service task assignment pattern. Second, the *User Agent* (UA) is assigned with the role of promoting the service request to the appropriate SPA. Third, the *Service Node Agent* (SNA) is introduced and assigned with the role of promoting the current load conditions of a CSN. Finally, the *Network Provider Agent* (NPA) is introduced and

assigned with the task of providing current network load conditions (i.e., bandwidth availability) to the appropriate SPA. In essence, the distributed set of the SNAs and NPAs forms the monitoring module. In other words, the SPA interacts with the UA in order to acquire the user preferences, requirements and constraints, analyses the user request in order to identify the service tasks constituting the service and their respective requirements in terms of CPU, memory and disk space, identifies the set of CSNs and their respective capabilities, interacts with the SNAs of the candidate service nodes so as to obtain their current load conditions and with the NPAs so as to acquire the network load conditions, and ultimately selects the most appropriate service task assignment pattern for the provision of the desired service.

Regarding the system model, we consider a set of service nodes SN and a set of links L. Each service node $n_i \in SN$ corresponds to a server, while each link $l \in L$ corresponds to a physical link that interconnects two nodes $n_i, n_j \in SN$. Our system operates in a multi-tasking environment, i.e., several tasks may be executed on a single service node sharing its resources (e.g., CPU utilization, memory, disk space). Let D_i denote a set of nodes grouped to form a domain. A pattern for the physical distribution of the related components to the service task assignment scheme is given in Fig. 1. Each SPA controls the service nodes of a domain. Each SNA is associated with each node, while each NPA is associated with the network elements (e.g., switches or routers) necessary for supporting service node connectivity. The SNA, NPA role (in a sense) is to represent the service nodes or network elements, respectively, and to assist SPA by providing information on the availability of resources of the service node/network element. Domain state information (load conditions of the service nodes of the particular domain and link utilisation) is exchanged between the SPA and the SNAs/NPAs residing in the specific domain, while SPAs residing in different domains exchange their domain state info. This approach increases scalability as it reduces the requirements in terms of computation, communication and storage. At this point it should be noted that for simplicity reasons the network elements needed for the service node connectivity are not depicted in Fig. 1.

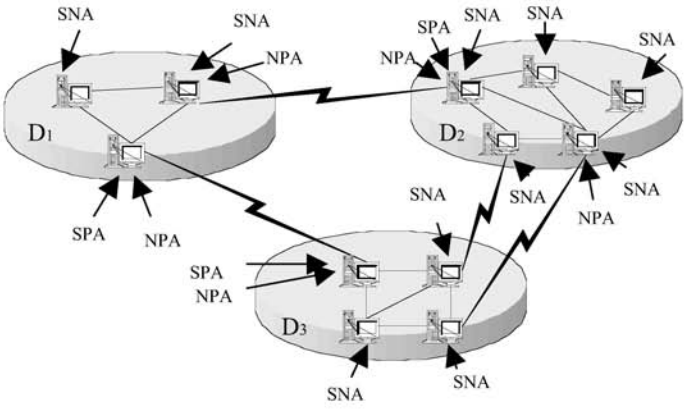

Fig. 1. System Model and physical distribution of the service task allocation related components

3 Problem Formulation & Optimal Solution

User u wishes to use a given service s. A fundamental assumption at this point is that service s may be decomposed in a set of distinct service tasks, which will be denoted as $ST(s)$. Among these service tasks, of interest to the user are those designated in the user profile and will be denoted as $ST(u,s)$ ($ST(u,s) \subseteq ST(s)$).

Let's assume the existence of multiple service nodes for the provision of service s, denoted by $SN(s) = \{n_1,...,n_{|s|}\}$. Each service node-$n_j$ contains a collection of components, denoted as $A_{n_j}(i)$, which inter-work with other components that may reside in the same or in a different service node in order to accomplish each service task $i \in ST(s)$. Let A_{n_j} and C be the total set of components residing in the n_j service node and the various service nodes in total, respectively. Hence, the following relationship holds: $A_{n_j}(i) \subseteq A_{n_j} \subseteq C$. Each service task $i \in ST(s)$ may be executed on an associated set of possible candidate service nodes, represented by the set $SN(i)$, ($i \in ST(u,s)$). Thus, $SN(i) \subseteq SN(s)$. The service logic deployment pattern adopted by service providers determine each of these service node sets.

Task i, ($i \in ST(s)$) requires for its completion consumption of $r_{CPU}(i)$, $r_{mem}(i)$ and $r_{disk}(i)$ resources of service node(s) n_j, ($n_j \in SN(i)$). A realistic assumption is that SPA being in charge of assisting the service providers in the competitive telecommunication market, has a solid interest in as accurately as possible identifying the resources $r_a(i)$ (where $a \in \{CPU, mem, disk\}$) needed for the provisioning of service task i in terms of CPU utilization, memory and disk space. In this respect, the SPA can be the entity that configures these values based on the service task characteristics, user preferences and requirements, exploiting also previous experience.

Let c_D denote the cost of involving service node n_j, ($n_j \in SN(i)$), in the service provision. For notation simplicity it is assumed that the cost of involving a service node in the solution is the same for all service nodes. As an alternative this cost could be taken variant (depending on the cost of acquiring and/or maintaining the node etc.). Notation may readily be extended.

The objective of our problem is to find a service task assignment pattern, i.e., an allocation $A_{ST}(s)$ of service tasks i ($i \in ST(u,s)$) to service nodes n_j, ($n_j \in SN(i)$), that is optimal given the current load conditions and number of service tasks being served by each service node n_j, represented as $r_a^{pre}(n_j)$ and $k^{pre}(n_j)$, respectively. The assignment should minimise an objective function $f(s, A_{ST}(s))$ that models the overall cost introduced due to system/network resources consumption. Among the terms of this function there can be the overall cost due to the deployment of various service nodes to the service provisioning process, the communication cost introduced due to the interaction of the components A_{n_j} residing in n_j service node with the components A_{n_k} residing in service node n_k for the completion of each service task i, ($\forall i \in ST(s)$), as well as the management cost $c_M(i,i')$ introduced due to the assignment of $(i,i') \in ST^2(s)$ service tasks to different service nodes

$(n_j, n_{j'}) \in SN^2(s)$.

The constraints of our problem are the following. First, each service task i ($i \in ST(u,s)$) should be assigned to only one service node n_j, ($n_j \in SN(i)$). Second, the capacity constraints of each service node should be preserved. Lets assume that $r_a^{\max}$ and $k^{\max}$ represent the maximum load and the maximum number of service tasks that a service node may handle. For notation simplicity, these parameters are assumed to be the same for each service node n_j, ($n_j \in SN(s)$). Thus, the constraints are $r_a^{post}(n_j) \leq r_a^{\max}$ and $k^{post}(n_j) \leq k^{\max}$, $(\forall n_j \in SN(s))$, where $r_a^{post}(n_j)$ and $k^{post}(n_j)$ denote the potential load conditions of service node n_j, after the service task assignment process. Notation may readily be extended.

The general problem version presented is open to various solution methods. Its generality partly lies in the fact that the objective and the constraint functions are open to alternate implementations. Thus, the problem statement can be distinguished from the specific solution approach adopted hereafter. In order to describe the allocation $A_{ST}(s)$ of service tasks to service nodes we introduce the decision variables $x_{ST}(i,j)$ ($i \in ST(u,s)$, $n_j \in SN(i)$) that take the value 1(0) depending on whether service task i is (is not) executed by service node-n_j. The decision variables $y_{SN}(j)$ assume the value 1(0) depending on whether candidate service node n_j ($n_j \in SN(i)$) is (is not) deployed (involved in the solution). In addition, we define the set of variables $z_{ST}(i,i')$ ($\forall (i,i') \in ST(u,s)^2$) that take the value 1(0) depending on whether the service tasks i and i' are (are not) assigned to the same service node. The variables $z_{ST}(i,i')$ are related to variables $x_{ST}(i,j)$, $x_{ST}(i',j)$, through the relation $z_{ST}(i,i') = \sum_{j=1}^{|SN(i)|} x_{ST}(i,j) \cdot x_{ST}(i',j)$, which may be turned into a set of linear constraints through the technique of [7]. Allocation $A_{ST}(s)$ may be obtained by reduction to the following 0-1 linear programming problem.

Service Task Assignment Problem:

Minimise

$$f(s, A_{TN}(s)) = c_D \cdot \sum_{n_j \in SN(s)} y_{SN}(j) \cdot \left(1 + b \cdot \sum_{a \in \{CPU, memory, disk\}} w_a \cdot \frac{r_a^{pre}(n_j)}{r_a^{\max}(n_j)}\right)$$

$$+ \sum_{i \in ST(s)} \sum_{n_j \in SN(i)} C(i,n_j) \cdot x_{ST}(i,j) + \sum_{i \in ST(s)} \sum_{i' \in ST(s)} c_M(i,i') \cdot (1 - z_{ST}(i,i')) \qquad (1),$$

where $C(i,n_j)$ denotes the communication cost introduced in case n_j service node has undertaken the responsibility for the execution of service task i ($i \in ST(u,s)$),

subject to the constraints:

$$\sum_{n_j \in SN(i)} x_{ST}(i,j) = 1 \quad \forall i \in ST(s) \qquad (2),$$

$$r_a^{pre}(n_j) + \sum_{i \in ST(s)} r_a(i) \cdot x_{ST}(i,j) \leq r_a^{\max}(j) \cdot y_{SN}(j) \qquad \forall n_j \in SN(s) \qquad (3),$$

$$k^{pre}(n_j) + \sum_{i \in ST(s)} x_{ST}(i,j) \leq k^{\max}(j) \cdot y_{SN}(j) \qquad \forall n_j \in SN(s) \qquad (4)$$

Cost function (1) penalises the aspects identified previously (i.e., cost of the service node involved in the solution, communication cost introduced during the

realisation of each service task, and management cost of service tasks that are assigned to different service nodes). In order for the service providers to better utilize their resources, the cost of each service node deployment introduced in cost function (1) takes also into account the node's current load conditions in order to obtain a load balancing solution. Parameters β, $(\beta < 1)$, and w_a denote the relative significance of load balancing and of each resource type a to the service provider. Constraints (2), guarantee that each service task will be assigned to one service node. Constraints (3) and (4) guarantee that each service node will not have to cope with more load and service tasks than those dictated by its pertinent capacity constraint.

In the rest of the section, we present a model for the overall communication cost $C(i,n_j)$ introduced in case n_j service node has undertaken the responsibility for the execution of service task i ($i \in ST(u,s)$). In essence, the model covers the case in which the components of set $A_{n_j}(i)$ need to interact with the components of set $A_{n_k}(i)$ residing in service node n_k in order to provide service task i, $(i \in ST(s))$. It should be noted that service nodes n_j and n_k may reside even in different domains. At this point, a major assumption adopted in our study, is that part of A_{n_j} components are implemented as mobile agents, while the rest are supposed to be fixed service agent components. Let $A^M_{n_j}$ and $A^F_{n_j}$ be the subset of components of A_{n_j} that are implemented as mobile and fixed agents, respectively.

The volume of messages exchanged between each pair of components (e.g., dependent on the number of messages and size of each message) for the accomplishment of task i ($i \in ST(s)$) will be represented as $m_{wv}(i)$, $\forall (w,v) \in C^2$ and $\forall i \in ST(s)$. Let $cc(n_j,n_k)$ be the communication cost per unit message that is exchanged between service nodes n_j and n_k, $\forall (n_j,n_k) \in SN(s)^2$. This factor may be proportional to the distance (e.g., number of hops) between the two service nodes and the load conditions (e.g., bandwidth availability) of the communication link interconnecting the two nodes. Another factor that should be taken into account is the cost associated with the migration of a component (implemented as a mobile agent) from one service node to another. In this respect, let $mc(w,n_j,n_k)$ be the migration cost of component-w from service node n_j to service node n_k, $\forall w \in C$ and $\forall (n_j,n_k) \in SN(s)^2$.

The overall cost for the completion of task i $(i \in ST(s))$ can be calculated by the following formula.

$$C(i,n_j) = \sum_{\forall n_k \in SN(s)} \Big[\sum_{w \in A^F_{n_j}} \sum_{v \in A_{n_k}} m_{wv}(i) \cdot cc(n_j,n_k) + \sum_{w \in A^F_{n_j}} \sum_{v \in A_{n_j}} m_{wv}(i) \cdot cc(n_j,n_j) + \sum_{w \in A^M_{n_j}} mc(w,n_j,n_k) + \sum_{w \in A^M_{n_j}} \sum_{v \in A_{n_k}} m_{wv}(i) \cdot cc(n_k,n_k) \Big], \quad \forall i \in ST(s) \quad (5)$$

In the previous formulation three main factors are identified. The first one is related to the communication cost deriving from the fixed components and is proportional to the messages (their number and size) that are exchanged between every pair of components (w,v) and the communication cost per unit message between different service nodes.

The second factor is associated with the migration cost of mobile agent

components between two different service nodes. This factor is dependent on the path which the mobile agent will follow (i.e., number of hops) and the information encryption and code execution cost, as well as the load conditions of the communication links. The last factor is the communication cost within the same service node, which in practice may be negligible, and in the context of this study is taken equal to zero. It is noted that only the involved to the provisioning process components are taken into account.

Apparently, the designation of the components that will be included in sets $A_{n_j}^{M}$ and $A_{n_j}^{F}$ by the service providers may be based on factors such as the overall communication and migration costs as well as estimation of the respective component invocations. In our study, the service logic deployment pattern (i.e., service components/nodes) adopted by the service providers is known.

4 Experimental Results

In this section, indicative results are provided in order to assess the proposed framework, which allows for effective service provisioning. In order to test the performance of the service task allocation scheme, we assume a simple application executing on a single PC performing a configurable number of queries on a database (that is, the service considered is composed of one service task that involves execution of one service component which interacts with the database).

Concerning the implementation issues of our experiments, the overall Service Provisioning System (SPS) has been implemented in Java. The Voyager mobile agent platform [8] has been used for the realisation of the software components as well as for the inter-component communication. To be more specific, the system components (SPA and the monitoring module SNAs, NPAs) have been implemented as fixed agents and the service task constituting the service as intelligent mobile agent, which can migrate and execute to remote service nodes.

A copy of the database exists on each service node, thus, the communication cost in practice is negligible and is taken equal to zero. In this case, only the service node deployment cost factor is considered and the performance of the system is tested using as decision parameter the load conditions of the service nodes.

The network topology that has been adopted for the experiments consists of five service nodes residing in a single domain. Specifically, all service nodes reside on a 100Mbit/sec Ethernet LAN. The configuration of the service nodes is as follows: two service nodes with 2GHz CPU and 2 GB RAM and three service nodes with 1GHz CPU and 1 GB RAM. All service nodes are running the Linux Redhat OS.

The idle states of the CPUs of the service nodes are simulated to follow the Exponential distribution, with mean value 50,000 ms and maximum value 100,000 ms. In all cases, the duration in which the CPU load of the service nodes is above 50% is 20,000 ms.

The graphical user interface of the SPA module, which implements the service task assignment process, is given in Fig. 2.

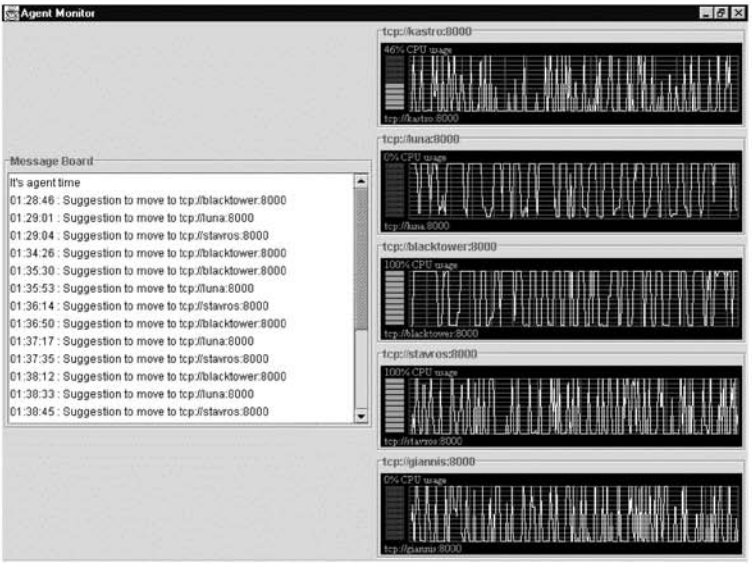

Fig. 2. User interface of the SPA module

We have performed 100 experiments with the mobile agent realising the service logic performing tasks varying from 100 to 1000 queries (with interval 100 queries). The same experiments have also been conducted without using our service task allocation scheme. In the latter case, service tasks are assigned randomly to service nodes.

The mean execution time when the service task assignment process is applied and when the service node is selected randomly are illustrated in Fig. 3. From the obtained results, we observe a decrease in the service completion time when the service task assignment system is used. At this point, it should be mentioned that the performance improvement introduced is tightly related to the number of queries the service task needs to perform at the remote service node and the time that the service node's CPU is idle. It may be observed that for small and large tasks (from 100 to 400 and from 800 to 1000 queries) the improvement in performance is bigger than in medium sized tasks (from 500 to 700 queries). It may also be derived that we have about 6% improvement for small tasks and about 9% for the large ones, while for medium sized tasks the improvement in performance is minor. This could be explained as follows. From Fig. 3, it could be extracted that the mean time required for initialisation of the mobile agent on a service node is approximately 35,000 ms. Also the execution of a task consisting of 100 queries when CPU is idle is 5,500 ms. Thus, small tasks can be performed during one slope of a CPU load (i.e., time during which CPU load is below 50%), while large tasks require for their completion one CPU slope, one CPU peak (i.e., time during which CPU load is above 50%) and finally another CPU slope. The completion of medium tasks usually requires one CPU slope and one CPU peak. Thus, the application of service task allocation process results in minor performance improvement.

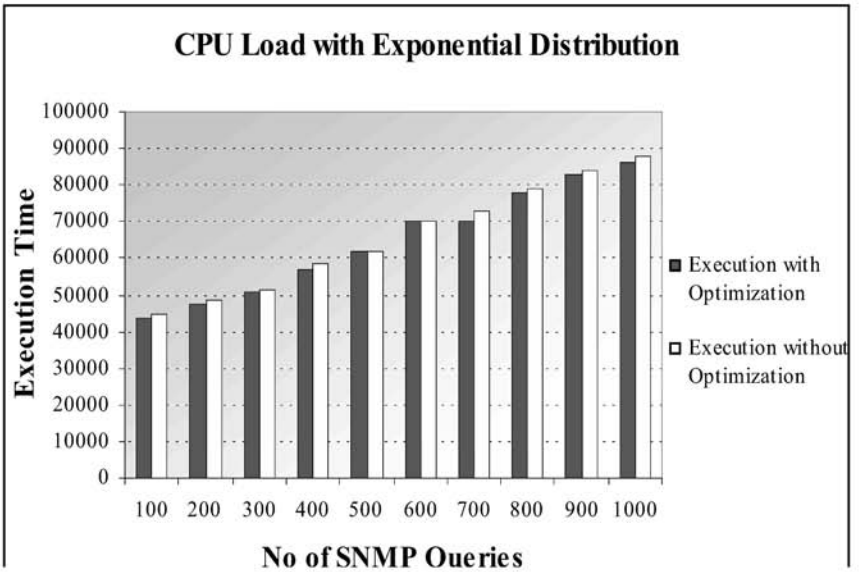

Fig. 3. Execution times with/without optimization for exponential CPU load distribution

5 Conclusions

The highly competitive communications markets should encompass mechanisms that will assist service providers in accounting for their interests, i.e., offering at a given period of time adequate quality services in a cost efficient manner which is highly associated to efficiently managing and fulfilling current user requests. This paper presented such mechanisms. Specifically, the contribution of this paper lies in the following areas. First, the definition and mathematical formulation of (one possible version) of the service task allocation problem, while a model for the communication cost between the service components involved during the provision of a service task was also provided. Through this work it is shown that the problem can be reduced to well-known optimisation problems, which can be solved by relevant standard algorithms. Second, the presentation of a software architecture, which is adopted for acquiring the best service task configuration pattern, i.e., assignment of service tasks to service nodes for efficient service provisioning.

Experimental results indicate that the proposed framework produces good results in relatively simple contexts (e.g., a service, which is composed of one service task that involves execution of one service component). Specifically, when the load conditions of the service nodes is the only factor considered for deciding on the most appropriate service node for the service provisioning, an overall improvement in service completion time of about 7% is introduced (especially, for the small and the large sized service tasks). What remains is to evaluate the performance of the proposed service task allocation scheme in complex contexts where communication cost factor is also involved.

Directions for future work include, but are not limited to the following. First, the realisation of further wide scale trials, so as to experiment with the applicability of the framework presented herewith. Second, the experimentation with alternate approaches (e.g., market-based techniques) for solving the service task allocation problem.

References

1. Trigila S., Raatikainen K., Wind B., Reynolds P., 1998. "Mobility in long-term service architectures and distributed platforms", IEEE Personal Communications, vol. 5, no. 4, pp. 44-55.
2. Magedanz, T., 1997. "TINA-Architectural basis for future telecommunications services", Computer Communications, vol. 20, no. 4, pp. 233-245.
3. Tag M., 1996. "Service creation environment engineering", Proc. Interworking'96 Conference, Japan.
4. Special Issue, 2003. "Special section on grid computing", ACM SIGMETRICS Performance Evaluation Review, vol. 30, no. 4, pp. 12-49.
5. Vinoski S., 1997. "CORBA: Integrating diverse applications within distributed heterogeneous environments", IEEE Commun. Mag., vol. 35, no. 2, pp. 46-55.
6. Morreale P., 1998. "Agents on the move", IEEE Spectrum, vol. 35, no. 4, pp. 34-41.
7. Papadimitriou C., Steiglitz K, 1982. Combinatorial optimization: Algorithms and complexity. Prentice Hall, Inc.
8. The Voyager Platform, Recursion Software Inc. http://www.recursionsw.com/