A Scalable Framework for Policy-based QoS Management in SOA Environments

Elarbi Badidi
Faculty of Information Technology, United Arab Emirates University, PO.Box. 17551, Al-Ain, United Arab Emirates, Email: ebadidi@uaeu.ac.ae

Larbi Esmahi
School for Computing & Information Systems, Athabasca University,1 University Drive, Athabasca, Alberta, T9S 3A3, Canada, Email: larbie@athabascau.ca

Abstract— **The successful integration of the Service Oriented Architecture (SOA) in large distributed environments greatly depends on their support of quality of service (QoS) management. The aim of QoS management is to guarantee diverse QoS levels to users issuing requests from a variety of platforms and underlying networks. In this paper, we present our policy-based framework for QoS management in SOA environment with both usual and mobile users. The framework is based on a federation of QoS Brokers that are in charge of mediating between service requestors and service providers, and carrying out various QoS management operations. We describe an auction-based ranking algorithm of functionally equivalent services, which ranks services according to their ability to fulfill the service requestor QoS requirements. The brokers are also in charge of handling appropriately service requests from mobile users equipped with various handheld devices.**

Index Terms— **Quality of service, QoS management, Service oriented computing, Web services, Service selection.**

I. INTRODUCTION

With the emergence of service-oriented computing and the ubiquitous deployment of enterprise applications on the Web to reach a wide base of customers, many organizations have moved their business online. Service-oriented computing together with Web technologies aim to promote business collaboration and application integration on a global scale. Web services are the current most promising technology founded on the concept of service-oriented computing. They provide the foundation for the development and execution of business processes distributed over the Web and accessible via standard interfaces and protocols.

Another development in the realm of service-oriented computing is the phenomenal rise in the number of mobile workers using a variety of devices including laptops and handheld devices, such as PDAs and SmartPhones, to consume online services. Modern mobile devices are usually fully equipped with broad capabilities. Most of these devices support various wireless communication technologies including Wi-Fi, Bluetooth, GPRS, and EDGE. They also come with advanced multimedia capabilities including streaming, and the ability to play a number of audio and video formats. These devices offer now browsing capabilities that go beyond the basic WAP protocol, to support HTML-based Web sites.

QoS is becoming a key feature in Web services competition due to this rapid growth. End-to-end quality management of Web services is a key issue that is highly related to the changing and dynamic environment of Web services. Most of the research works on QoS support in Web services have focused on the enumeration of QoS requirements and mechanisms for QoS management. Also, some research efforts have addressed the issues of QoS manageability, and security in SOA in general. The success of QoS support for Web services largely depends on the scalability of the proposed mechanisms for QoS management. This is due to the widespread adoption of SOA, which results in a rising number of clients and servers.

To cope with this concern, we propose a policy-based framework for QoS management of Web services, which is capable of supporting at runtime the QoS management phases (QoS specification, QoS negotiation, and QoS monitoring). This framework provides support for mobile users using various handheld devices. Its policy-based management allows dealing with issues of authentication, authorization, QoS policy, user profile management, and mobile devices profile management. We also describe an auction-based selection policy that enables ranking and selecting appropriate Web services according to the user's QoS expectations.

The remainder of the paper is organized as follows. Section 2 describes the main objectives of this work. Section 3 provides related work and background information on the issue of Web services' QoS management, the recent initiatives for provisioning Web services on mobile devices, and the issues of policy-based management and server selection. Section 4 presents an overview of our proposed framework. Section 5 details our auction and QoS based algorithm for services' selection. The paper concludes in Section 6.

doi:10.4304/jsw.6.4.544-553

II. OBJECTIVES

As a result of the rising interest in QoS support in Web services, the limited number of QoS management architectures in SOA environments, and the increasing use of mobile devices to consume online services, our aim in this work is to develop a framework that meets the following requirements:

- Provide a QoS model for the specification of the QoS in Web services. The model should allow providers to specify the classes of services they can deliver to their users.
- Allow description, publication, and discovery of Web services on the basis of their QoS attributes.
- Provide support for QoS management operations such as QoS-based service selection, QoS negotiation, and QoS monitoring.
- Provide support for mobile users, with QoS requirements, using various handheld devices.
- Allow user and device profile management.
- Enable monitoring the provision of QoS agreed between providers and users.
- Allow the implementation of various policies such as authorization policies, policies for monitoring services and QoS, and policies for QoS-enabled Web service selection.

To meet the above requirements, we propose in this work a framework, which is relying on a service domains federation. This federation extends the reach of both services and requestors. A QoS Broker component is in charge of managing each service domain. Our motivation in using brokers is their successful and widespread adoption in various systems. In SOA, brokers mediate between services providers, service consumers, and partners. Several multimedia systems and mobile computing systems use brokers to deal mainly with the issue of QoS management [1][2][3].

III. RELATED WORK AND BACKGROUND

Many research works have proposed broker-based architectures to cope with QoS management for Web services. Yeom et al. [4] designed and implemented a QoS Broker system, which monitors Web services availability, performance, and reliability. Tao et al. [5] proposed a broker-based framework for the composition of QoS-aware Web services with QoS-constraints. The broker's components implement dynamic service composition, service selection and adaptation. Zuquim et al. [6] described a system for QoS management of Web services using QoS Brokers. The brokers provide support for the interaction with the UDDI. They publish QoS information, obtained from providers, in the UDDI, and help customers select relevant services according to their functional and QoS requirements.

All these works share some common goals with our work. However, most of them do provide support for only some of the functionalities and concerns we raised in the objectives section. None of these works has considered the issue of handling mobile users that have different requirements as they use various devices with limited capabilities. Also, none of them has used a policy-based management approach with policies at several levels: authentication, User profile management, QoS specification, QoS monitoring, and service policies in general. Moreover, most of them did not address the scalability issue for QoS management.

In the subsequent sub-sections, we present background information on recent research initiatives regarding the description of mobile device capabilities, as well as background information on policy-based management and server selection policies.

A. Web Services on Mobile Devices

Several service providers currently offer services that allow information to be pushed to a mobile device, or offered via a limited Web-browser interface. With the advent of the service oriented architecture, it is becoming possible to offer services that fully use the power of the mobile device. In the last few years, mobile services access was suffering from interoperability and usability problems. This was to some extent attributable to the small physical size of the screens of mobile devices. It was also due to the incompatibility of several mobile devices with the structure of much of the information conveyed to mobile devices.

To deal with these issues, the *W3C Mobile Web Initiative* (MWI), established by the W3C, aims to develop best practices and technologies for creating mobile-friendly content and applications. The goal of the initiative is to enable access to the Web from mobile devices and to make it more reliable and accessible. This typically requires the adaptation of Web content based on the device capabilities. The W3C has published guidelines (Best Practices, W3C mobileOK checker service) for specifying mobile content [7]. The *MWI Device Description Working Group* is actively tackling the problem of device diversity by setting up a repository of device descriptions [8]. Authors of Web content may use the repository to adapt their Web content to best suit the requesting device.

The *Open Mobile Alliance* (OMA) specification defines the *User Agent Profile* (UAProf) to describe capability and preference information of wireless mobile devices [9]. Content providers will mainly use this information to generate content in a suitable format for the device. It is based on the generic framework W3C *CC/PP* (Composite Capabilities/ Preference Profiles) [10]. CC/PP defines a schema for the description of a device's profile, which is composed of components that describe characteristics of hardware, software, network, and so on. A CC/PP profile can be used to adapt Web contents to a specific device.

A UAProf file describes the capabilities of a mobile device, including Vendor, Model, Screen size, Multimedia Capabilities, Character Set support, and more. It consists of seven components, which are *Hardware Platform*, *Software Platform*, *Network*

Characteristics, *Browser UA*, *WAP Characteristics*, *Push Characteristics*, and *MMsCharacteristics*. A server, called the *Profile Repository,* stores user agent profiles. Frequently, mobile device manufacturers manage their profile repositories. For instance, Nokia manages its own profile repository in which it maintains user agent profiles describing the capabilities of Nokia smartphones. The x-wap profile header of requests sent from a given mobile device contains the URL that points to the user agent profile of that device. For example, the x-wap-profile header of a Nokia N97 cell phone has the following value:

"http://nds1.nds.nokia.com/uaprof/NN97r100-2G.xml"

B. Policy-based Management

Policy-based management had been mainly used for network management. A policy is a set of conditions imposed on a subject that permits or prohibits actions. These conditions may apply to the subject, the target, or the state of the system. A policy may be the trigger that allows actions to be performed when conditions are applicable.

The IETF Policy Framework Working Group has developed a policy management framework for the Internet [11,12]. The framework components are the *Policy Management Service*, the *Dedicated Policy Repository,* the *Policy Decision Point (PDP)*, the *Policy Enforcement Point (PEP)*, and the *Local Policy Decision Point (LPDP)*.

Policy-based management may play a crucial role in the management of Web services in general and QoS in particular. The context of Web services is however different from the network context as the intrinsic components are different from one context to the other. In Web services environments, only a Policy Manager, which plays the role of the PDP, and a policy repository in which policies are to be stored, will be required to handle policies.

Utilization of policies in Web services environments has been recognized since the specification of the first standards for Web services. The WS-Policy specification was proposed by IBM, BEA, SAP, Microsoft, and others, to simply define a framework and mechanisms for constructing policy expressions that describe the requirements for accessing a Web service [13].

In this policy-based model for Web services, individual requirements or capabilities of a policy subject are declared using XML policy assertion elements. Policy assertions are the building blocks of policies. Each assertion describes an atomic aspect of a service's requirements. A policy expression can be comprised of one or more policy assertions assembled in Policy alternatives using logical policy operators. This expression can also be associated with a Web service resource, such as a service or endpoint, using WSDL or other mechanisms defined in WS-PolicyAttachment [14].

C. Server Selection Policies

Selection of the most relevant Web services, with regards to the user QoS expectations, is a crucial issue in ensuring user satisfaction. Indeed, users may have different preferences and expectations for QoS. For instance, a user may be interested in minimizing the cost of getting the service while satisfying some other constraints such as response-time and reputation. Another user may give more weight to the service responsiveness than to the cost of getting the service. Therefore, QoS-aware service selection policies are required to match the user QoS requirements with the QoS that Web services can deliver.

In the last two decades, several research works have investigated the issue of server selection in both traditional distributed systems and the Internet. They have studied both static and dynamic policies for scheduling and server selection [22]. Static policies rely solely on information about the average behavior of servers in the system. Random and round-robin are examples of static selection policies. Conversely, dynamic policies rely on the current conditions of servers and react to changes in the system's state. Examples of dynamic server selection and scheduling policies are: the Job Shortest Queue (JSK) [22], the Least Recently Used (LRU) strategy for caching, and the mean number of round trip measurements [18] [19]. Dynamic policies, as opposed to static policies, can respond to system changes and avoid making poor decisions that may lead to poor performance. Nevertheless, the drawback of dynamic policies is the communication overhead that may result from state collection and communication activities.

Sreenath et al. proposed an agent-based approach for Web services selection [20]. This approach uses traditional mechanisms for selection such as collaborative filtering and a reputation system. It relies on using collaborative agents to evaluate service providers. Moreover, each agent can autonomously opt for the weight to give to the recommendations obtained from other agents. Maximilien et al. proposed an agent-based framework and a QoS-ontology for dynamic Web services selection [17]. The QoS-ontology and an XML policy language allow service consumers and providers to expose their QoS preferences and advertisements. They let providers, consumers, and agents collaborate to determine each other's QoS and trustworthiness.

Wang and al. presented a Web services framework for service selection and scheduling under limited resources [21]. They proposed a Web service selection algorithm, which relies on two main threads. The first thread dispatches incoming service requests to the waiting queues of suitable service providers. The second thread is responsible of removing the front requests from waiting queues and submitting them to relevant providers. The main feature of this work is that the proposed scheme provides flexible services selection and schedule with dynamic optimization of resource utilization.

Huang has formulated the problem of ranking Web services by considering multiple attributes of a Web service [23]. He has investigated ranking Web services for a given task, ranking in the context of workflows, and ranking Web services according to the user preferences. This work is the closest one to our selection algorithm, described latter in the paper. As opposed to our proposed algorithm, the author did not consider the weights that may be given by the user to each QoS parameter.

IV. QoS Management Framework

Our proposed framework addresses the issues of handling requests issued from mobile devices, policy-based management, and the already mentioned requirements. It is founded on a service domains federation. The need for service domains federation has already been recognized by the research community [24,25]. A federation of service domains extends the reach of both services and requestors. A service is no longer accessible only to requestors of its domain. Instead, requestors may request services across domain borders (similar to the cellular network). Services, in a single domain, are available to requestors through a QoS Broker.

The service domains of the federation may be, for example, geographical domains where similar services can be provided at different domains. They may also be independent, functional domains where services provided in a service domain are quite different from services offered in other domains.

The aim of the broker is to play the interface between service providers and requestors of a same domain. Providers define the interfaces of their QoS-enabled Web services in WSDL and publish them through an extended UDDI registry [15][6]. Clients may then search the UDDI registry for Web services that are able to deliver their required services with QoS requirements. Given that Web services providers and clients do not normally have the capabilities to negotiate, manage, and monitor QoS, they delegate management tasks, such as Web services selection and QoS negotiation, to the QoS Broker.

Fig. 1 depicts the service domain federation with four domains. Brokers from the various domains collaborate in order to provide services across domain borders. The federation is created by specifying the links between the brokers of the various service domains. A link contains information about the target broker, such as the reference of its *Coordinator* component and the main application area of the services within its domain. The *Link Manager* component of each broker is responsible for managing its links with other brokers of the federation. It provides the following operations: *add_link(), remove_link(), modify_link(), describe_link(), and list_links().*

Fig. 2 depicts the main components of a single service domain. The broker includes several components, which cooperate in order to deliver personalized services to both regular users and mobile users with various devices. These components are the *Request Dispatcher*, the *QoS Negotiator*, the *QoS Monitoring Manager*, the *Profile Manager*, the *Policy Manager*, and the *Link Manager*. They are under the control of the *Coordinator* component. They allow carrying out several management operations such as admission control, QoS negotiation, QoS-based service selection, QoS monitoring, user profile management, policies management, and federation management. Subsequent sub-sections describe these management operations.

The back-end databases maintain information about services' policies, user profiles and preferences, and dynamic QoS information.

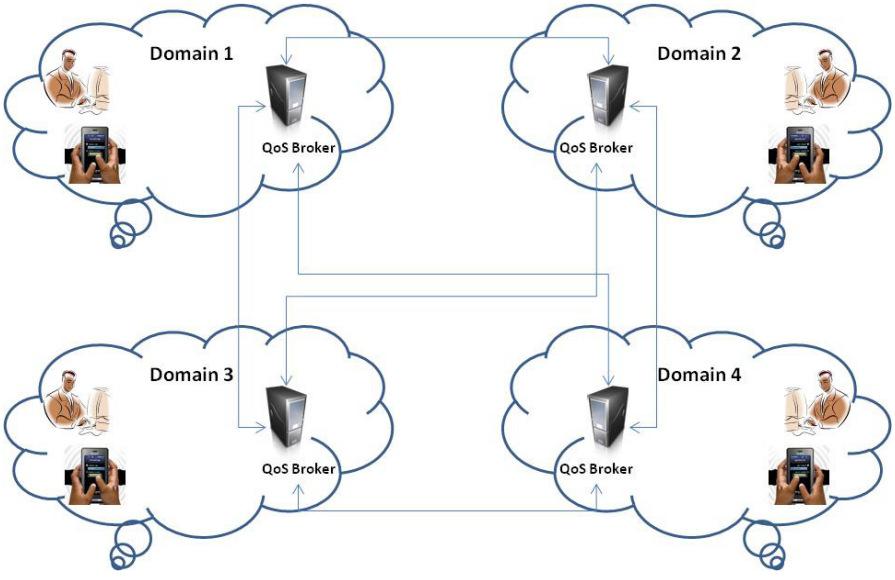

Figure 1. Service domains federation

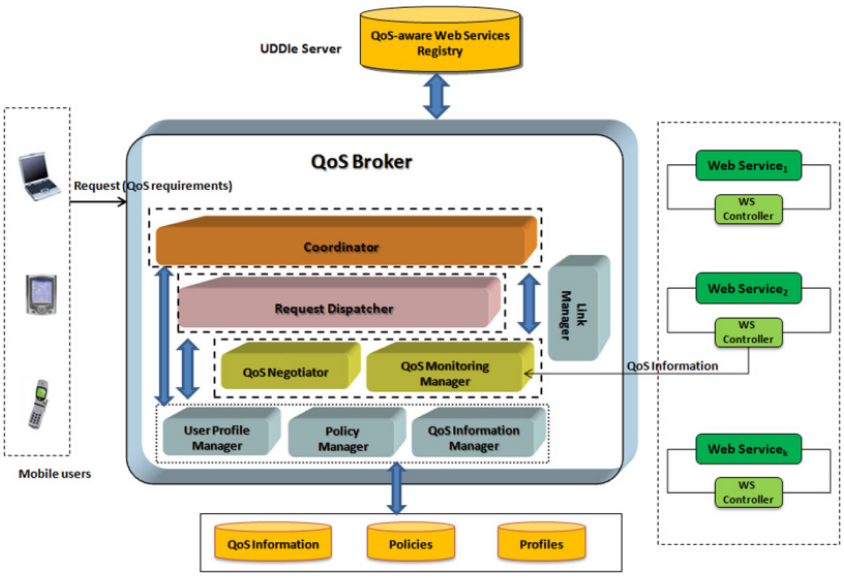

Figure 2. QoS Broker components

A. QoS Specification

Specification of QoS that providers can deliver may be performed by incorporating QoS parameters, such as response-time, throughput and cost, into the WSDL service description. This is the QoS model supported by the UDDIe [15]. Recent works [6] [16] have proposed extensions to the Web services Policy Framework (WS-Policy) to represent QoS policies of Web services. WS_Policy does not define how policies are discovered or attached to a Web service. The WS_PolicyAttachment specification [14] defines such mechanisms, especially for associating policy with WSDL artifacts and UDDI elements. Fig. 3 depicts an example of a QoS policy defined with a policy assertion representing the availability supported by a Web service. In our framework, QoS policies are also stored in the policies repository as well as the other policies concerning authentication, authorization, user profile and preferences management, and mobile devices profile management.

```
<wsp:Policy>
    xmlns:wsp= "http://www.w3.org/ns/ws-policy"
    xmlns:qosp= "Maqam.cit:8080/schema/qospolicy"
    <wsp:ExactlyOne>
      <wsp:All>
        <qosp:Availability
            xmlns:qosp= "Maqam.cit:8080/schema/qospolicy"
            operation="get"   specification="uddi:uddi.org:qos:attribute:availability">0.98
        </qos:Availability>...
      </wsp:All>
    </wsp:ExactlyOne>
</wsp:Policy>
```

Figure 3. Example of a QoS policy

B. Admission Control and QoS-based Service Selection

The *Request Dispatcher* is in charge of the admission control of incoming requests. It is responsible of determining whether the received requests can use the requested services. This means that Web services access is denied to requests from users who did not negotiate the level of QoS with the selected Web services providers.

The *Request Dispatcher* is also in charge of implementing different policies for the selection of the Web service, which will provide the user required service with the required QoS. These policies can range from simple policies, such as random and round-robin, to complex ones taking account of the current state of servers in terms of availability, load, and level of QoS they can provide. The *Request Dispatcher* performs the match-making of the required QoS with the QoS stored either in the UDDIe or the policies repository. We describe a new algorithm for QoS-aware server selection in Section VI. The algorithm takes account of the current conditions and capabilities of potential Web services as well as the QoS required by the user and his/her weights for quality parameters.

If the Web services of the domain cannot provide the user required service or his/her required QoS, then the QoS Broker forwards the user request to a suitable domain that can handle it.

C. QoS Negotiation

The *QoS Negotiator* is in charge of carrying out the negotiation process in order to reach an agreement as to the QoS to be delivered to the user (Fig. 4). First, the user notifies the broker about its required service and its preferred level of QoS. Based upon available QoS information, the *Request Dispatcher* component selects an appropriate server capable of satisfying the required QoS. Then, the broker approaches this server to determine whether it will be able to satisfy the required level of QoS given its current conditions. Then, the client and the provider sign a contract. The contract specifies the service that the provider will offer to the client, the guaranteed QoS, the cost of service, and actions to take when there is a violation of the agreed QoS. If the selected server is not able to deliver the required QoS, the broker selects another server and reiterates the negotiation process.

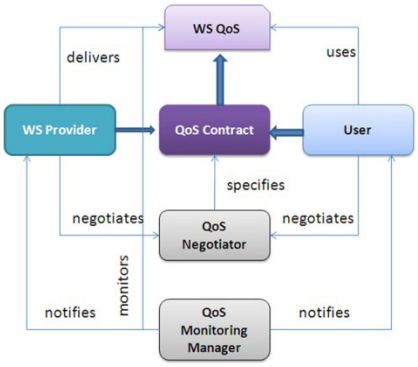

Figure 4. QoS negotiation

If no server is available to meet the required QoS, the broker notifies the user about it. The user may, then, reduce his/her QoS expectations or wait until the conditions of the system may allow obtaining the required level of service. Figure 4 shows the interactions of the broker components, via the *Coordinator*, with the user and the WS provider.

D. QoS Monitoring

QoS monitoring is extremely crucial to assure the compliance of delivered QoS with contracted QoS. It allows the broker to take appropriate actions when it detects that there is a violation of the contract. The *QoS Monitoring Manager* and the *QoS Monitor* of the *Web Service Controller* attached to the Web service are responsible of monitoring QoS of Web services (Fig. 5).

The *QoS Monitoring Manager* continually observes the level of QoS level rendered to clients. The contract specifies the QoS parameters, such as response-time and availability, which need to be observed. The *QoS Monitoring Manager* observes these QoS parameters through periodic measurements at some observation points in both server-side and client-side. There is QoS violation when the measured value of a QoS parameter does not meet the requirements of the agreed upon one.

In this case, the QoS Broker notifies both the client and the provider about the QoS violation. Violation details include information on the violation type, name of violated QoS parameters, period during which the violation occurred, and cause of the violation.

E. Profile Management

The *Profile Manager* is responsible for managing users' profiles, which include their preferences, in terms of personalized services, current location, and required QoS. It is also responsible for negotiating the user profile, authorized services, and devices' profiles described using UAProf and CC/PP.

F. Policies Management

The *Policy Manager* is responsible for maintaining authorization policies, and polices for monitoring services and QoS. It receives access control requests, processes them against a set of policies, which describe how users can use services, and returns access control responses. Fig. 6 depicts the architecture of the *Policy Manager*. The *Policy Decision Point* is the component that grants or denies access to services. It encompasses a rule engine that determines the policies to apply when a user has issued a request for service. When a user attempts to access a service, with QoS requirements, the *PEP* sends a message to the PDP asking whether to accept or not the user request. It then replies with approval or denial of access to that service depending on the service rules and the request parameters.

G. Scenario of Interactions

Fig. 7 depicts a scenario of interactions between a mobile user and the QoS Broker components. The QoS Broker can initiate and control interactions with Web services, within its domain of control, and with service requestors by exchanging information messages using the SOAP protocol.

1. The mobile user submits a request, which includes the CC/PP profile of the mobile device in use in the case of a mobile user.
2. After processing the user's authentication on an authorized device, the Coordinator requests the user profile from the *User Profile Manager*. Then, it requests from the *Request Dispatcher* to select a relevant Web service to handle the user request.
3. The *Coordinator* requests policies of the selected Web service from the *Policy Manager*.
4. If the user profile is available locally in the profile repository, for example because the user had previously used some services within the domain of control of the QoS Broker, the *Coordinator* may determine whether the Web services of the domain can handle the mobile user request or not. This decision relies on the profile of the mobile user and the policies of the requested Web service.

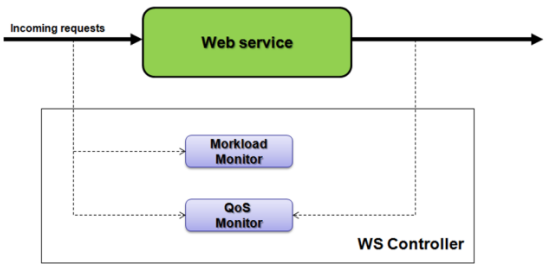

Figure 5. Web service controller

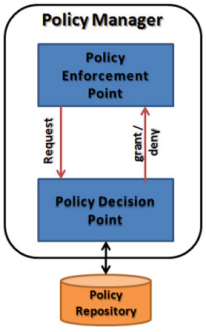

Figure 6. Policy Manager components.

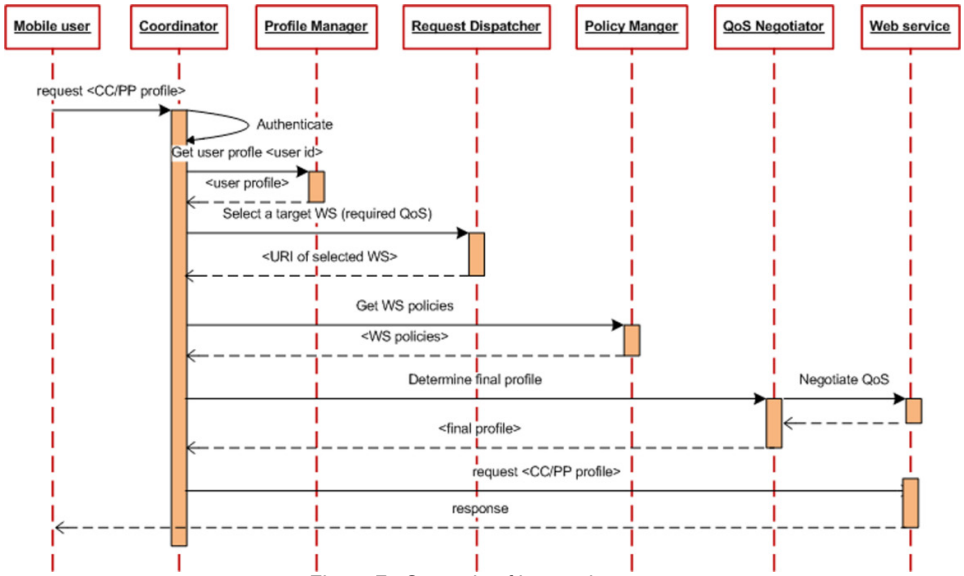

Figure 7 - Scenario of interactions

5. If the requested profile is not available locally, the *Coordinator* asks the user to provide information, such as service preferences and required levels of QoS, in order to create a new profile for the user.
6. If there is at least one Web service of the domain that can handle the user request, the *Coordinator* requests from the *QoS Negotiator* to determine the final profile of the user by negotiating the level of QoS to be delivered.
7. After getting the final profile of the user, the *Coordinator* forwards the user request to the selected Web service in order to be serviced.

V. Auction-based Service Selection Policy

In this section we describe our auction-based service selection algorithm that allows ranking the Web services in the federation domains with regard to the user required QoS, and then finding out the most suitable Web service in the federation to handle the user request. First, we describe how the algorithm works in a single domain. Then, we show in fig. 8 how the selection is done at the level of the federation.

Let $D = \{D_1, D_2, \dots, D_m\}$ represent the set of domains of the federation F. The total number of domains in the federation F is m. Let $B = \{B_1, B_2, \dots, B_m\}$ be the list of QoS Brokers of the federation F.

A. Single Domain Service Selection

The broker B_i of the domain D_i that receives the user request initiates an auction in order to get bids from relevant Web services that can handle the user request. Each bid is a vector, which describes the level of QoS for each of the quality parameters that the Web service may provide given its current conditions. This model corresponds to a *multiple-items auction*. After a predefined period, the broker closes the auction and ranks the bids according to the weight specified by the user for each quality parameter. It, then, selects the winner of the auction as the most suitable Web service to handle the user request in the domain. The auction models, proposed in the auction theory, are *English auction*, *Sealed-Bid auction*, *Dutch auction*, and *Vickrey auction* [26].

As the bidders should not be aware of the bids of each other, the sealed-bid auction is the most relevant in our case. In the following, we describe the process of determining the winner of the auction.

Let $W_i = \{WS_{1i}, WS_{2i}, \dots, WS_{ki}\}$ be the set of Web services of the domain D_i, with $1 \le i \le m$. Let $P = \{p_1, p_2, \dots, p_l\}$, the set of QoS parameters considered in the system. The total number of these parameters is l. For example,
$P = \{response\ time, availability, throughput, \dots\}$.

Let $B_i = \{b_1, b_2, \dots, b_l\}$ represent the best QoS that can be provided in the domain for each quality parameter. Therefore, b_j is the best QoS that can be provided for quality parameter P_j.

Let r^u, a user request sent to the domain D_i, and let M_u represent the user required QoS. $M_u = \{min_1^u, min_2^u, \dots, min_l^u\}$ with $1 \ge q_j^u \ge 0$ for $1 \le j \le l$. min_j^u represents the minimal value that the user is willing to accept for the quality parameter P_j.

$$min_j^u \le b_j \text{ for } 1 \le j \le l$$

A zero value for one quality parameter means that the user has not specified any constraint on that parameter.

Let $bid_j = \{q_{1,j}, q_{2,j}, \dots, q_{l,j}\}$ be the bid (QoS that may be provided) of the Web service WS_j of the domain D_i. We assume that these quality values are normalized and are between 0 and 1 (a percentage of a maximal value for each quality parameter (b_j)). A value of 1 means highest quality and 0 means lowest quality.

We also assume that for each quality property, higher normalized values represent higher levels of service quality. Therefore, for the *response-time* property, which is normally good when the value is small, we can consider its inverse as the quality parameter. That is,

$1/_{response\_time}$. The more the response-time is smaller the more its inverse is higher.

Let Ω_i the set of Web service from domain D_i that can provide the service required by the user and that have submitted their bids for handling the user request.

$\Omega_i \subseteq W_i$. Let *s*: the cardinality of Ω_i.

$$\Omega_i = \{WS_{1i}, WS_{2i}, \dots, WS_{si}\}$$

Let Q_i represent the matrix corresponding to the QoS that the Web services in Ω_i can provide for each QoS parameter.

$$Q_i = \begin{array}{c} \\ WS_{1i} \\ \\ WS_{ji} \\ \dots \\ \dots \\ WS_{si} \end{array} \begin{array}{c} \begin{array}{cccc} p_1 & p_2 & \dots\dots\dots & p_l \end{array} \\ \begin{bmatrix} q_{1,1} & q_{2,1} & \cdots & q_{l,1} \\ \vdots & & & \vdots \\ q_{1,j} & q_{2,j} & \cdots & q_{l,j} \\ \vdots & & & \\ q_{1,s} & q_{2,s} & \cdots & q_{l,s} \end{bmatrix} \end{array}$$

The user request can be satisfied if there is in the above matrix at least one row, which has values greater than or equal to the user min values; that is:

$$WS_{ji} = [q_{1,j} \;\; q_{2,j} \;\; \dots \;\; q_{l,j}]$$

$$\text{For } 1 \le k \le l \quad q_{k,j} \;\ge\; min_k^u$$

In the following, we define the distance of each offer from the user required QoS. This distance is defined for each quality property by:

$$d_{k,j} = q_{k,j} - min_k^u \qquad (1)$$

$$\text{For } 1 \le k \le l \text{ and } 1 \le j \le s$$

Therefore, we can consider the distance matrix $Dist_i$ for all Web services in Ω_i.

$$Dist_i = \begin{array}{c} \\ WS_{1i} \\ \\ WS_{ji} \\ \dots \\ \dots \\ WS_{si} \end{array} \begin{array}{c} \begin{array}{cccc} p_1 & p_2 & \dots\dots\dots & \dots p_l \end{array} \\ \begin{bmatrix} d_{1,1} & d_{2,1} & \cdots & d_{l,1} \\ \vdots & & & \vdots \\ d_{1,j} & d_{2,j} & \cdots & d_{l,j} \\ \vdots & & & \\ d_{1,s} & d_{2,s} & \cdots & d_{l,s} \end{bmatrix} \end{array}$$

Using the distance matrix, we can say that the user request can be satisfied if there is at least one row in the above matrix with values greater than or equal to zero. Therefore, we can discard from that matrix rows having negative values. The resulting matrix is called $PDist_i$ (all rows have positive values). Let $c_i = Cardinality(PDist_i)$.

The Euclidian distance of WS_{ji} offer from the user required QoS is:

$$d_j = \sqrt{\sum_{1 \le k \le l} \left(d_{k,j}\right)^2} \qquad (2)$$

$$\text{with } 1 \le j \le c_i$$

The highest value of d_j will correspond to the best offer that can fulfill all the user required QoS.

The most suitable Web service, that we call $target^u$, will be the one that maximizes the above Euclidian distance, that is :

$$target^u \leftarrow max\left(\sqrt{\sum_{1\le k\le l}^{1\le j\le Cardinality(PDist_i)} \left(d_{k,j}\right)^2}\right) \qquad (3)$$

$$target^u \leftarrow max\left(\sqrt{\sum_{1\le k\le l}^{1\le j\le Cardinality(PDist_i)} \left(q_{k,j} - min_k^u\right)^2}\right) \qquad (4)$$

In this model, we have assumed that the user has given the same weight to all QoS parameters. This is not always the case as the user may set relative weights for the QoS parameters. For instance, he/she may give more weight to the *response-time* than to the *availability* of the Web service. Furthermore, the weight may depend on the application domain.

Therefore the weights, given by the user to the QoS parameters, can be expressed by the following vector:

$$W^u = \{\omega_1, \omega_2, \dots, \omega_l\}$$

$$\text{Where } 0 \le \omega_j \le 1 \text{ and } \sum_{1\le j\le l} \omega_j = 1$$

ω_k is the weight given by the user to the quality parameter P_k. The normalized weights indicate the user preferences with respect to the quality dimension.

Given these weights, the most suitable Web service will be the one that satisfies condition (1) and which maximizes the sum of quality offers for all QoS indicators:

$$target^u \leftarrow max\left(\sqrt{\sum_{1\le k\le l}^{1\le j\le c_i} \left(\omega_k\left(q_{k,j} - min_k^u\right)\right)^2}\right) \qquad (5)$$

and $\sum_{1\le k\le l} \omega_k = 1$

B. Multiple Domains Service Selection

The previous subsection describes how the ranking of Web services is achieved within a single domain. In order to find out the most suitable Web service within the federation of service domains, the broker that received the user request forwards the user QoS requirements to each broker of the other domains with whom there is a link. Each of these domains conducts, then, a similar auction in order to find the best Web service to handle the user request within the domain. Selected Web services from the domains of the federation are then ranked to find out the best Web service, which maximizes the above Euclidian distance expressed by (4) and (5). Fig. 8 summaries the steps of the algorithm.

C. Proof of Concept

As a proof of concept, we describe in this section a scenario of how the auction-based selection algorithm works within one domain of service.

Assume that the QoS indicators considered in the system (indicators in the set P) are in order of *availability*, *1/response-time*, *reliability*, and *throughput*.

The normalized minimum QoS requirements of the user are given in Table 1.

Assume that four Web services of the domain have submitted their bids to the QoS broker, WS_1, WS_2, WS_3, and WS_4. Table 2 describes The QoS offer of the four Web services.

The initial Broker, from domain D_i, which has received the user request for service carries out the following steps:

***Step-1**: Construct the vector $\boldsymbol{M_u}$ of minimum QoS requirements the user can tolerate. We assume that all values of the vector are normalized to be in the range [0,1].*

***Step-2**: Forward the vector $\boldsymbol{M_u}$ to the QoS Brokers of the other domains of the federation that are linked with the domain D_i. Each broker conducts, then, a similar auction (step3, step4, step5, and step6) to find out the most appropriate Web service within its domain.*

***Step-3:** Construct the normalized matrix $\boldsymbol{Q_j}$ of the QoS offers of all Web services, of the domain D_j , which submits their bids to their QoS Broker.*

***Step-4:** Calculate the distance matrix $\boldsymbol{Dist_j}$ that represents the linear distance between the QoS offers of the bidders and the user QoS requirements for each quality parameter considered in the system. If a row of this matrix has negative values, then it means that the bidder cannot meet the QoS requirements of the user. Only rows with positive values will be considered in the next steps. The resulting matrix is called $\boldsymbol{PDist_j}$ (all rows have positive values).*

***Step-5:** Calculate from the matrix $\boldsymbol{PDist_j}$ the Euclidian distance for each bidder using equation (2).*

***Step-6**: Find out the maximum Euclidian distance using equation (4) or (5), depending on whether the user has specified weights for its QoS requirements. The Web service associated with that Euclidian distance is considered as the most suitable for handling the user request in the Domain D_j.*

***Step 7:** Rank the best Web services from the domains in order to get the best one in the federation.*

Figure 8- Auction-based Service Selection Algorithm

By computing the distance matrix and the Euclidian distance for each Web service, we get the following ranking of the Web services from highest offer to lowest:

$WS_4(0.0310)$, $WS_2(0.0218)$, $WS_3(0.0087)$

The value between parentheses corresponds to the computed Euclidian distance. The QoS offer of WS_1 does not meet the minimum QoS requirements of the user.

TABLE I.

Minimum QoS Requirements of the User

	availability	1/RT	reliability	throughput
ω_k	0.15	0.35	0.15	0.35
M_u	0.35	0.97	0.50	0.96

TABLE II.

QoS offers of four Web Services

	availability	1/RT	reliability	throughput
WS_1	0.38	0.94	0.55	0.98
WS_2	0.45	0.98	0.60	0.97
WS_3	0.40	0.97	0.53	0.96
WS_4	0.35	0.98	0.70	0.98

VI. CONCLUSION

In this paper, we have presented a new framework for the management of the QoS in a SOA environment. The framework is based on a federation of service domains; each one managed by a QoS Broker that is in charge of mediating between service requestors and service providers, and carrying out various QoS management operations. The framework is capable of handling service requests from mobile users using various handheld devices. Policies are a crucial part of the framework. They are used at different levels: Authorization, QoS specification, QoS service monitoring, and description of service policies. The framework is scalable in terms of the ability to add new service domains to the federation as well as the ability to handle the user request by services from different domains. This is illustrated through our proposed auction-based service selection algorithm. The algorithm ranks, first, functionally equivalent services from a same domain, according to their ability to fulfill the service requestor QoS requirements. Best services from each domain of the federation are then ranked to find out the most suitable Web service to satisfy the user QoS requirements.

A prototype of our proposed framework is under development. The implementation platform includes: NetBeans, the UDDIe registry [15] with MySQL, and Apache Neethi, which provides a general framework for developers to use WS-Policy. Some components of the QoS Broker such as the QoS Monitoring Manager and the Request Dispatcher with basic selection policies have been partially implemented in our previous work. QoS monitoring is achieved using observers that are called SOAP handlers. As a future work, we intend to extend our proposed architecture with security policies related to SOA using standards such as WS_Security and WS_Policy.

REFERENCES

[1] G. Stattenberger and T. Braun, "QoS Provisioning for Mobile IP Users," In H. Afifi and D. Zeghlache, editors, Conference on Applications and Services in Wireless Networks, ASW 2001, Paris, July 2001.

[2] V. Marques et al., "An Architecture Supporting End-to-End QoS with User Mobility for Systems beyond 3rd Generation," http://newton.ee.auth.gr/summit2002/papers/SessionW9/2602138.pdf

[3] D. Chalmers and M. Sloman, "A Survey of Quality of Service in Mobile Computing Environments," *IEEE Communications Surveys*, vol. 2, no. 2, 1999.

[4] G. Yeom and D. Min, "Design and Implementation of Web Services QoS Broker," In Proceeding of The International Conference on Next Generation Web Services Practices (NWeSP 2005), 2005, pp. 459- 461.

[5] Y. Tao and K.J. Lin, "A Broker-based Framework for QoS-aware Web Service Composition," In Proceedings of The 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service, 2005 (EEE'05), pp. 22-29.

[6] D. Zuquim Guimares Garcia and M.B. Felgar de Toledo, "A Web Service Architecture Providing QoS

Management," In Proceeding of The Fourth Latin American Web Congress (LA-WEB'06), 2006, pp. 189-198.
[7] W3C, "W3C MobileOK Checker [Beta]," http://validator.w3.org/mobile/
[8] W3C, "MWI Device Description Working Group," Retrived from: http://www.w3.org/2005/MWI/DDWG/
[9] Open Mobile Alliance, "WAG UAProf, Version 20-Oct-2001," http://www.openmobilealliance.org/tech/affiliates/wap/wap-248-uaprof-20011020-a.pdf
[10] W3C, "Composite Capability/Preference Profiles (CC/PP): Structure and vocabularies 2.0," W3C Working Draft 30 April 2007. http://www.w3.org/TR/2007/WD-CCPP-struct-vocab2-20070430/
[11] IETF Network Working Group, "Policy Framework Architecture," http://tools.ietf.org/html/draft-ietf-policy-arch-00
[12] D.C. Verma, S. Calo, and K. Amiri, "Policy-based Management of Content Distribution Networks," *IEEE Network Magazine* (2002), vol. 16, pp. 34-39.
[13] S. Bajaj, et al., "Web Services Policy 1.5 – Framework," W3C Candidate Recommendation 28 February 2007. http://www.w3.org/TR/2007/CR-ws-policy-20070228/
[14] W3C, "Web Services Policy Attachment," http://www.w3.org/Submission/WS-PolicyAttachment".
[15] A. ShaikhAli, O.F. Rana, R. Al-Ali, and D.W. Walker, "UDDIe: an Extended Registry for Web Services," In Proceedings of The IEEE Symposium on Applications and the Internet Workshops, Jan 2003, pp. 85 - 89.
[16] S. Chaari, Y. Badr, and F. Biennier, "Enhancing Web Service Selection by QoS-based Ontology and WS-Policy," In Proceedings of The 2008 ACM Symposium on Applied Computing (SAC 2008), Fortaleza, Ceara, Brazil, 2008, pp. 2426-2431.
[17] E.M. Maximilien and M.P. Singh, "A Framework and Ontology for Dynamic Web Services Selection," *IEEE Internet Computing*, 8(5), 2004, pp. 84–93.
[18] M.E. Crovella and R.L. Carter, "Dynamic Server Selection in the Internet," In Proceedings of the Third IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems (HPCS'95), 1995.
[19] R.L. Carter and M.E. Crovella, "Server Selection Using Dynamic Path Characterization in Wide Area Networks," In Proceedings of Infocom '97, the Sixteenth Annual Joint Conference of the IEEE Computer and Communication Societies, 1997.
[20] R.M. Sreenath and M.P. Singh, "Agent-based Service Selection," Web Semantics: Science, Services and Agents on the World Wide Web, 2004, pp. 261–279.
[21] X. Wang, K. Yue, J.Z. Huang, and A. Zhou, "Service Selection in Dynamic Demand-driven Web Services," In Proceedings of the IEEE International Conference on Web Services (ICWS'04), 2004, pp. 376-383.
[22] Y. Wang and R.J. Morris, "Load Sharing in Distributed Systems," *IEEE trans. On Computers*, Vol. c-34 (3) , 1985, pp. 204-217.
[23] X. Huang, "WSRank: A New Algorithm for Ranking Web Services," New Technologies, Mobility and Security, Springer Netherlands, pp. 529-539, 2007.
[24] T. Koponen and T. Virtanen, "A Service Discovery: A Service Broker Approach," In Proceedings of the 37th Hawaii International Conference on System Sciences – 2004.
[25] J. Hu , X. Fan , and R. Fischer, "Business-Driven Trust Federation Management for Service Marketplaces", in the 2010 IEEE International Conference on Services Computing. pp. 594-60.
[26] V. Krishna, "Auction Theory, Second Edition," Academic Press; 2009.

Elarbi Badidi is currently an Assistant Professor of computer science at the Faculty of Information Technology (FIT) of United Arab Emirates University. Before joining the FIT, he held the position of bioinformatics group leader at the Biochemistry Department of Université de Montréal from 2001 to July 2004. He received a Ph.D. in computer science in 2000 from Université de Montréal, Québec (Canada). Dr. Badidi has been conducting research in the areas of object-based distributed systems, bioinformatics tools integration, and Web services. He is a member of the IEEE, IEEE Computer Society, and ACM. He served on the technical program committees of many international conferences. His research interests include Web services and service oriented computing, middleware, cloud computing, and bioinformatics data and tools integration.

Larbi Esmahi is an Associate Professor of the School of Computing and Information Systems at Athabasca University. He was the graduate program coordinator at the same school during 2002-2005. He holds a PhD in electrical engineering from Ecole Polytechnique, University of Montreal. His current research interests are in e-services, e-commerce, multi-agent systems, and intelligent systems. He is an associate editor for the Journal of Computer Science, and the Tamkang Journal of Science and Engineering. He is also member of the editorial advisory board of the Advances in Web-Based Learning Book Series, IGI Global, and member of the international editorial review board the International Journal of Web-Based Learning and Teaching Technologies.