Towards Efficient Selection of Web Services

Amir Padovitz
School of Computer Science & Software Engineering, Monash University
Padovitz@bigpond.com

Shonali Krishnaswamy
School of Computer Science & Software Engineering, Monash University
shonali.krishnaswamy@mail.csse.monash.edu.au

Seng Wai Loke
School of Computer Science & Software Engineering, Monash University
swloke@csse.monash.edu.au

ABSTRACT
In the existing frameworks for web services there is no incentive to bind dynamically to a specific web service. However, once runtime information concerning those web services is available to the application, dynamic binding becomes advantageous over a static pre-decided one. We propose a model that provides web service clients with runtime information that is pertinent to its execution and business logic. When faced with multiple service providers who can provide the same (in functionality) service, the client can dynamically select the current best (e.g., in terms of availability for the duration of the service, reliability, and estimated response time) service provider for its required service, according to the client's constraints and information gathered about the service providers at runtime.

1. INTRODUCTION
Web Services are software applications or services that are uniquely identified by a URI (Uniform Resource Identifiers) and expose public interfaces for clients, using XML (extended mark-up language). Those web services can be discovered and used by other client applications using XML based messages and protocols such as HTTP.

The emergence and continued development of web services standards such as SOAP (simple object access protocol) and WSDL (web services description language) [4] enable us to request and describe web services in a standard way. This will increase the ease of use of web services, enable interoperability between heterogeneous platforms and help businesses solve integration problems of their applications. Consequently, it is anticipated that web servers that host the services will be subject to increasing usage and have a higher load. Furthermore, the current simple modus operandi involving client/server activation of a single web service will be enhanced to support more complex scenarios, in which applications and service providers themselves rely on other external web services as part of their business logic. The reliance on third party web services reduces the control of the organization over its application and (sometimes) mission-critical code. The control and information of certain parts of the system is pushed outside organizational boundaries. Scenarios involving reliance on external web services raise several new issues and challenges. An example of common scenario would be of clients consuming external web services, which in turn outsource their computational resources to other service providers.
Furthermore, runtime information such as service load and availability or business related constraints might affect the selection process of an external web service, and not be pre-decided, as it is today. In the existing frameworks for web services there is no incentive to bind dynamically to a specific web service. However, once runtime information concerning those web services is available to the application, a dynamic binding becomes advantageous over a static, pre-decided one. We suggest a model that provides the web service client runtime information that is pertinent to its execution and business logic. The client application can then dynamically bind to the temporarily best service, from a selection of acceptable web services it works with, and according to the client's set of constraints.

A client may want to apply some business rules when dynamically choosing a web service, or may be more concerned with response time or availability. When response time is critical (e.g. stock quotes service etc.) it is important for an application to activate the fastest web service available at that given time, or have some mechanism that ensures availability and reliability. When several clients participate in such a scenario, an indirect load balancing mechanism is created, which helps to direct clients to available and relatively fast web services.

Figures 1 and 2 illustrate a client activation decisions based on information gathered at runtime from the service providers according to the client constraints.
In figure 1, the client is concerned with availability and response times of a web service; after retrieving related information from the service providers, it activates the fastest available web service. This behaviour contributes to the robustness of the client application. Figure 2 shows client activation, based on response time and quality of service. According to the client's business constraints, it may prefer to switch to another service provider when it observe a change in the combination of quality and response time offered by the service providers.

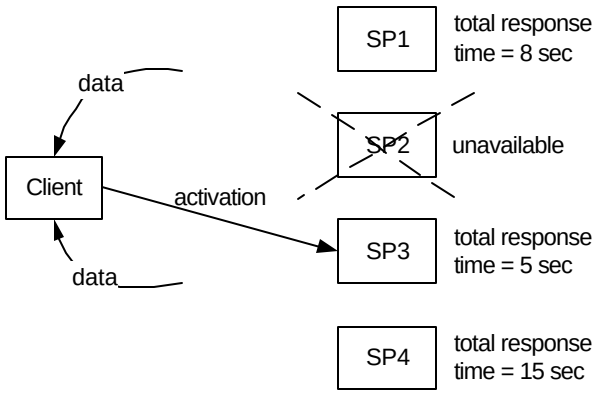

Figure 1 – web service activation according to response time and availability

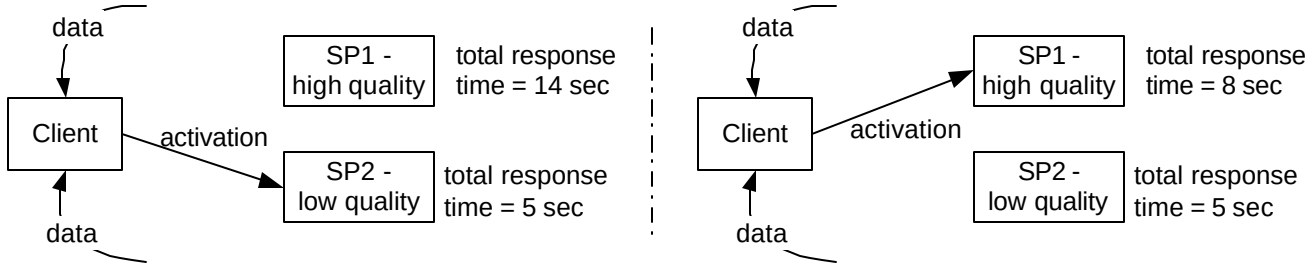

Figure 2 - dynamic activation according to response time and quality of service

In order to facilitate dynamic selection, up to date information concerning parameters that affect the decision of web service activation must be gathered. We are currently developing and investigating a system that retrieves runtime related information from service providers according to the client constraints and specifications and decides which web service to activate. To enable such functionality we examine two paradigms – the traditional RPC (remote procedure call) approach and a mobile agent approach. We draw three different conceptual models based on these technologies and compare their strengths and weaknesses. While the first two are based on applying the well-known RPC and Mobile Agent paradigms, we also introduce a novel Circulating Mobile Agent model that exhibits different characteristics and complements the first two.

The paper is organized as follows. In section 2 we introduce related work. Section 3 presents architectural models for the dynamic selection of web services. Section 4 draws a quantitative analysis for the use of those models. Section 5 discusses implementation details of the prototype that supports the models described in section 3. We conclude and discuss future work in section 6.

2. RELATED WORK

Mobile software agents are units of code capable of migrating to different hosts while maintaining their data and state of execution. Mobile agents display autonomous behavior, which implies a capability to handle various scenarios independently, without a need for some application management layer. Consequently, they are capable of performing asynchronous tasks and reduce communication overheads. Another important feature mobile agents possess is the ability to work in heterogeneous environments. Currently, Agents reside in dedicated server applications, which can potentially be activated on any platform. Furthermore, they present fault tolerance and robustness characteristics, as failure of a specific node in a network will affect only agents physically located on that node at that time. These characteristics can become very useful when integrated in large and heterogeneous networks such as the Internet and provide an alternative to the RPC approach, including web services. Web services may evolve to become more agent like and enjoy the agents' autonomy, interaction capabilities and add robustness and efficiency effects on the system [5]. The mobile agent paradigm can also coexist with web services and both can mutually benefit from each other strengths. In this work we present an attempt to integrate the two approaches in one system to extract the beneficial characteristics of both.

An important step towards the integration of mobile agents and web services is the work being done on the creation of the semantic web and the development of new web markup languages such as DAML (DARPA Agent Markup Language) [6] and OWL [7] as well as ontology of services such as DAML-S that aims to enable the discovery, activation, monitoring and selection of web service by agents [8].

A similar client oriented approach is taken by [1] to perform load balancing of Internet services by moving parts of the load balancing decision making from the server to the clients. Other systems [2] also use client-oriented approaches to enhance the overall file system performance.

3. ARCHITECTURAL MODELS

In this section, we present architectural models that assist in dynamic activation of web services. We present three models: an RPC based approach, a mobile agents approach and circulating mobile agents. In order to illustrate the models and their respective modes of operation we use the following scenario shown in figure 3. As seen, each participant can be both a client and source of information, e.g. a service provider can be the source of information for another client as well as a client requesting information on other web services.

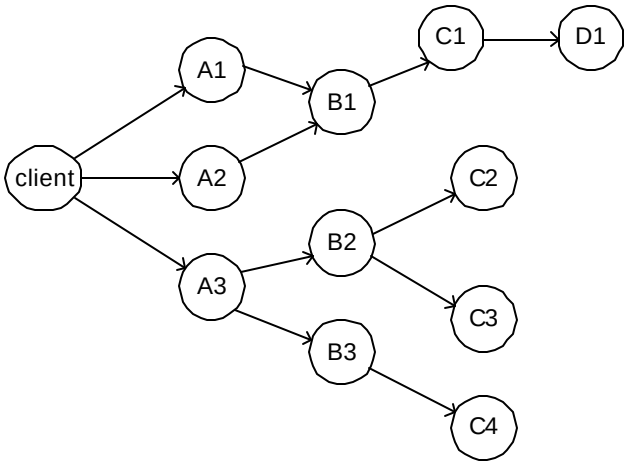

Figure 3 - dependencies between service providers

In this scenario a client wishes to retrieve information on a service of type A. Three service providers are available: A1, A2 and A3. Those service providers are themselves dependent on other web

services to complete their tasks. For example service A1 is dependent on service B1, which is dependent on service C1, etc. (A1→B1→C1→D1). Here, the service providers, in turn, gather information about other service providers they depend on. This dependency makes it more difficult for a service provider to accurately estimate information such as the total time for processing a client request. When attempting to retrieve information on a web service several client constraints should be imposed on the system. For example, the client should have the ability to limit the duration of the information retrieval process or the depth (or level) of the network the information is to be retrieved from. Figure 4 illustrates a possible outcome of this approach when activated on figure 1 network, with a client that restricts the depth to 3 service provider levels and has a timeout constraint for the search:

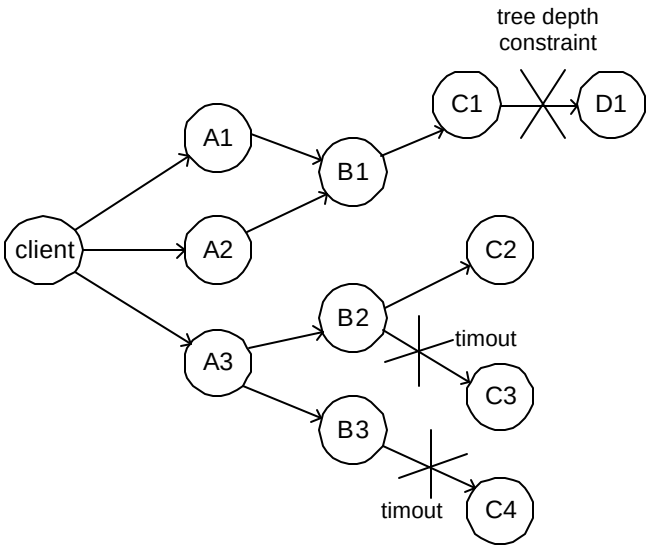

Figure 4 - possible results of info retrieval

In the example above, only the route: client→A3→B2→C2 yields a result, which is communicated back to the client by service provider A3. Other service providers are unable to formulate the desired information for the client. Factors that limited the information gathering routes in this scenario are: a tree-depth constraint between nodes C1 and D1 (which has reached the fourth level), and a timeout constraint between B2 and C3 and B3 and C4 (in this case we assume a relatively long processing time by service provider C3 and C4)

We have shown a possible scenario that depicts a network in which information is gathered for a web service; we now present three models that support this "information gathering for selection before activation" behaviour.

3.1 Models for Dynamic Selection of Web Services

We propose three different models for dynamic selection. Each of these models has its own strengths and weaknesses and is best suited for particular situations. We aim to integrate all three approaches and create a hybrid model to cater for different situations.

3.1.1 RPC based model

In traditional wide network scenarios (e.g internet) the most straightforward approach for gathering information for web services activation would be to use RPC (remote procedure calls) for communicating information between hosts in the network.

For platform independence, web services themselves can be the means by which communication between two hosts is performed. Information could then be easily and generically sent and received between all the participants in the network. This kind of implementation is generally beneficial in wired networks, as multiple connections need to be handled, which may become problematic in wireless environments where connections are less reliable. In this model a component-oriented approach is taken, in which a client/service provider is treated as a black box. When a service provider receives a request, it may become a client and actively request information from other service providers.

This approach simplifies the programming complexity of an environment consisting of many service providers. This model also supports complex scenarios where service providers support several web services and a client requires information on more than one service. This is shown in Figure 5.

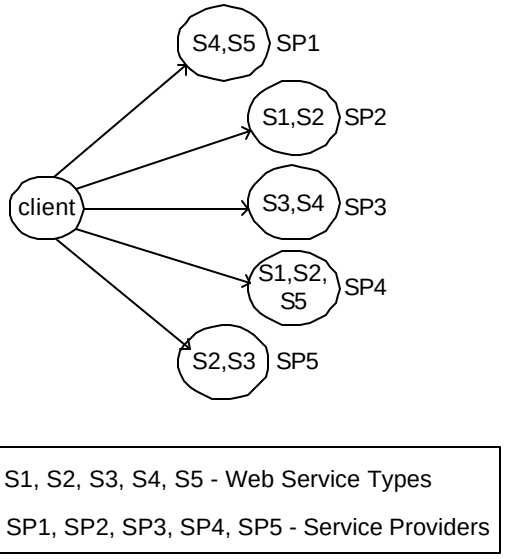

Figure 5

The client sends requests to all the service providers; each request contains information on all the desired services from the recipient. The data that is retrieved is then examined and ranked by the client's system. Following this rule the amount of requests per service desired is minimized (instead of inquiring all service providers several times, each is inquired only once).

In this architecture, decisions regarding the ways to query inner depth nodes in the network or decision on which nodes to query and when, are all delegated to the service provider, which then becomes itself a client of the system. In this component-oriented scheme, the initial client delegates future decisions and implementation details to sub-contractors, in the form of the service providers. Once a request is launched, parameters contained in the request cannot be changed, even if sometime along the way the criteria for performing future requests have changed. The initial client no longer has the control over the operation.

Another problem is the high amount of messages sent between client and service providers. This limits the reliability of such a system in a wireless environment, where connections are less reliable. Furthermore, the higher the number of participants, the more messaging is involved, resulting in more traffic congestion.

3.1.2 Mobile Agents based model

To address the issue of wireless connectivity and client control we propose a second model, based on mobile agents. We examine two approaches that differ in the type of behaviour agents perform. In the first model, agents are launched by the client, arrive at the service providers, query information, and if needed, continue to look for dependant services required information. Agents contain client restrictions such as timeouts and maximum number of hops as well as other data that pertain only to the specific client. Following this approach, two agents of different clients may act differently under similar circumstances given different client directions for behaviour.

Figure 6 shows a possible strategy performed by agents *a* and *b* that are launched by the client. Upon arriving at service of type A they are redirected to retrieve information from service providers B1 and B2. Then they are redirected again to gather information pertaining to services C1, C2 and C3. After arriving at B2 agent *b* clones itself into 2 agents, each traveling on a different path to accomplish its task

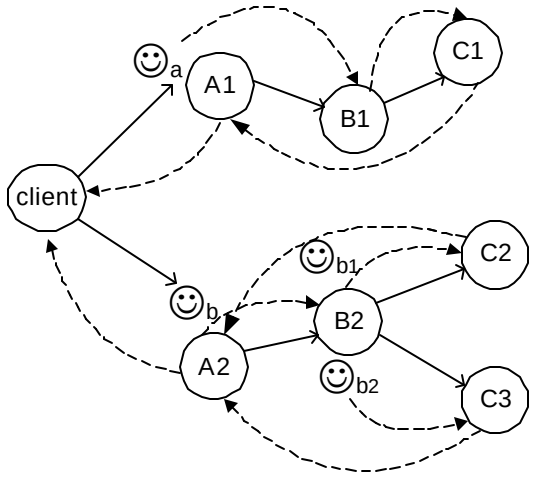

Figure 6 – first mobile agents approach

In this approach we gain a client control of the agent behaviour also in deeper node levels, after the initial encounter with the first service provider. On top of client control, we also gain better reliability in wireless environments, as the amount of connections is highly reduced compared with the RPC model. In this model the client (a wieless device) only maintains connections as per the amount of agents it initially launches. A mobile agent should also be embedded with the ability to change its migration path if encountered with disconnected or unresponsive nodes. Despite their higher level of abstraction, implementing agents to work in a large network of dependable service providers may be more complex. Instead of treating the service providers as black boxes as in the first model, the agents need to be programmed to move around the network autonomously and respond to possible changes in the environment.

The main disadvantage of this approach however is that it is less realistic in a business sense. It is unlikely that a client agent would be permitted to be redirected and interact with nodes that the service provider is concerned with. Service providers may have for example private agreements with other service providers and would not want to send an agent that represents the initial client. It may also be against the best interest of the client, since sometimes a direct request for a service would result in a higher price than if the other service provider handled it.

The second approach assumes interaction of the client agent only with the required first level service providers. This approach is similar to the RPC one, where service providers are treated as "black boxes". Although we loose the client control characteristics, we still maintain better performance in wireless environments, compared with the RPC model.

Figure 7 shows this approach; agents operate on behalf of their clients and are restricted to interact only with the required first level service providers.

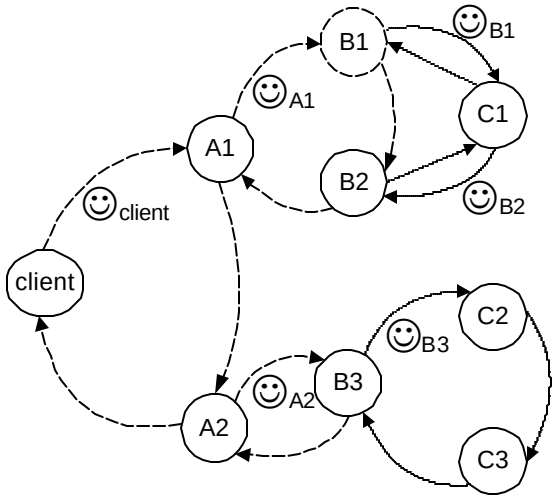

Figure 7 - second mobile agents approach

Figure 8 depicts the differences between the RPC and Mobile Agents models in terms of the number of wireless connections. In the RPC model the client manages 5 connections to all 5 service-providers, while in the Mobile Agents model only 1 connection is established and used to send a single mobile agent that clones itself and arrives to all the service providers autonomously.

3.1.3 Circulating Mobile Agents model

The two previous models are suitable to work with web services that are expensive to purchase and/or consume an overall long processing time. Since the task of collecting information is time consuming, it is less likely these approaches will be utilized when it is imperative to perform fast activation of short-term and inexpensive web services. For such scenarios, a third model is proposed that can provide service providers related information "on demand". The idea behind this model is having mobile agents periodically circulating the path of the service providers and retrieving information. The information is then given to the client, which then performs web service activation. In this scenario, information that arrives is more updated and is available sooner.

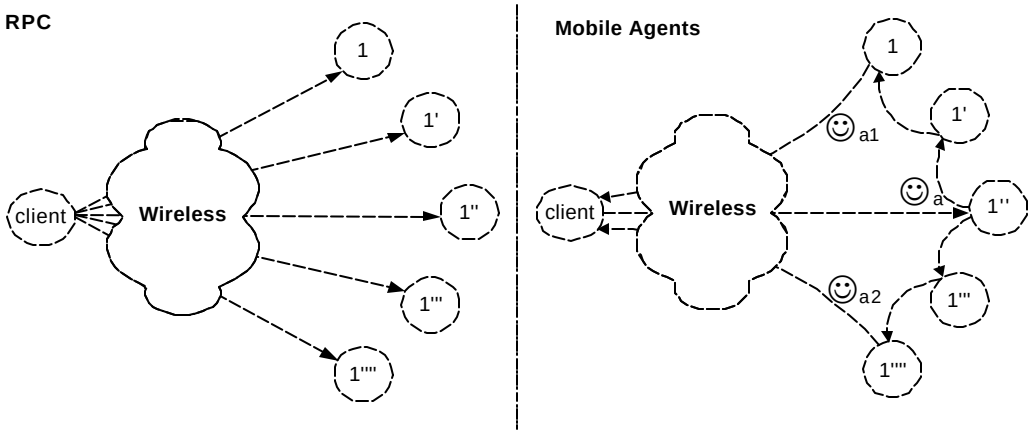

Figure 8 - differences in number of wireless connections between the models

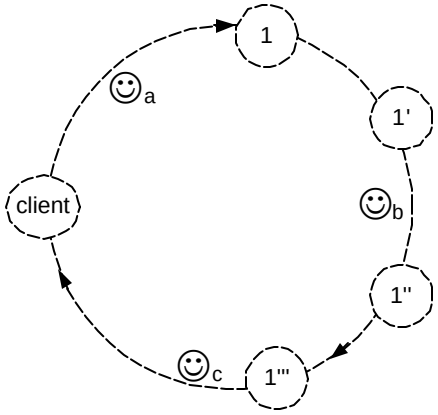

Figure 9 – circulating mobile agents

Along with its beneficial characteristics, this model may also suffer from a varied amount of redundancy. Depending on the client application, agents may circulate the network, retrieving information without any current need to do so. To minimize this redundancy, an ability to control the amount of circulating agents and the duration of their life cycle will be introduced.

Table 1 – activation considerations of the different models

Service Characteristics	**RPC**	**MA**	**Circulating MA**
Expensive, long processing time	X	X	
Fast response is important			X
Network Characteristics			
Wired	X		
Wireless		X	X

4. QUANTITATIVE ANALYSIS FOR USAGE OF THE MODEL

Having presented the three models, we now analyse the suitability of the models with respect to the response time criterion. However, we note that clients may have additional criteria for activation such as price. In this paper, we focus on response time and draw a quantitative representation of the processing time of the different models. We also present rules governing the system activation decision based on response time for each mode of operation.

We define *Treqi* as the time it takes for a request to be sent to a specific service provider i, *Tresi* as the time it takes for a response to be sent back from a specific service provider i, *Twsi* as the average time it takes for a web service to complete for a specific service provider i and Tans i is the total time it takes for a service provider i to produce information about the user requested parameters. We define *T*req*, *T*res* and *T*ws* as the request, response and processing times of the web service that is selected by the system.

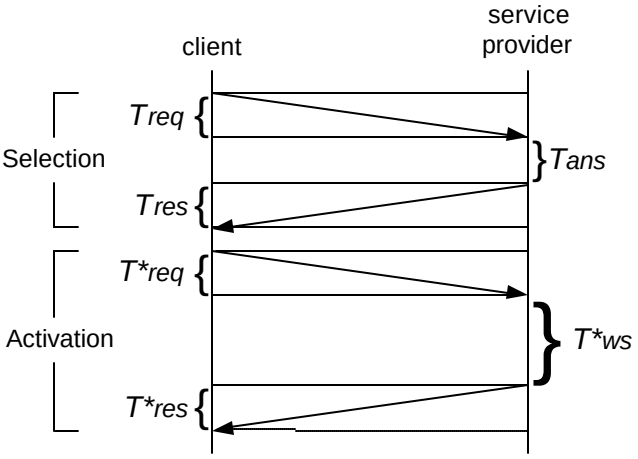

Figure 10 – Basic terminology

4.1 RPC Model

On average, the total time for activating a web service without using the RPC model would be:

$$Tavg = \frac{\sum_{i=1}^{n} (Treqi + Tresi + Twsi)}{n}$$

and the total average communication time would be

$$\frac{\sum_{i=1}^{n} (Treqi + Tresi)}{n}$$

where n is the number of service providers offering similar web services for the client to choose from.

Let $Ta = Treqi + Tresi + Tansi$

Ta therefore represents the total time for a client to query service provider i for information.

The total activation time of the selected web service is represented by *Tactivation*, and is:

*Tactivation = T*req + T*res + T*ws*

We express the duration of the processing time of a sequentially activated RPC model as follows:

$$Trpc = \sum_{i=1}^{n} (Treqi + Tresi + Tansi)$$

If the RPC messaging is performed in parallel, we can express it as follows:

Trpc1 = Treq1+Tres1+Tans1
Trpc2 = Treq2+Tres2+Tans2
Trpc3 = Treq3+Tres3+Tans3
.
.
.
Trpcn = Treqn+Tresn+Tansn

Equation 1

$Trpc = max\ (Trpc1, Trpc2, Trpc3 \ldots Trpcn)$

And the total activation time of the RPC model is therefore:

Trpc_total_activation = Trpc + Tactivation

We can formulate a general rule for activating the RPC model with respect to response time, and state that whenever Equation 1 is observed, the client is encouraged to use the RPC model, as it gains a better overall response time compared with an average usage of a randomly selected web service.

Equation 2

$Trpc\_total\_activation < Tavg$

Equation 2 is more likely to occur if high variation in the activation time of web services (*Twsi*) is observed. In such a case, it is more likely that Trpc_total_activation will be less than *Tavg.* Furthermore, we argue that since the client may not be concerned only with response time, it is likely that it would prefer to use the RPC model even if it has a longer total response time, to a certain extent. We denote this argument in Equation 3, where a client still prefers to use the model as long as it is not longer than a specific time amount.

Equation 3

$Trpc\_total\_activation < Tavg + d$

Where d denotes an extra amount of time.

An example of this would be when a client is more interested in guaranteeing the reliability of the service or is concerned with the service pricing, rather than just the response time. In such cases it will be willing to "pay" for a longer process.

4.2 Mobile Agent Model

We define Tmi,j as the migration time between node i and node j in a network. n represents the total number of participating nodes (i.e. the service provider nodes and the client. We count the client as the first node). We investigate a model in which the agents only visit service providers in the first level of the service providers network. When only a single agent is launched, the total migration and query time of the mobile agent model is:

Equation 4

$$Tma = Tmn,1 + \sum_{i=1}^{n-1} (Tmi, i+1) + \sum_{i=2}^{n} Tansi$$

And the total activation time of the Mobile Agent model is:

Tma_total_activation = Tma + Tactivation

The rule for activation of the mobile agent model follows the same guidelines as the RPC model:

Equation 5

$Tma\_total\_activation < Tavg + d$

A general form for multiple agents launched can be described as follows:[1]

M1, M2, M3 … MN denote the total individual migration and query times for N agents.
a denotes the number of agents participating.

$$M1 = T(n/a)*1,(n+1) + T(n+1),1 + \sum_{i=1}^{n/a-1} (Ti, i+1) + \sum_{i=2}^{n} Tansi$$

$$M2 = T(n/a)*2,(n+1) + T(n+1),1 + \sum_{i=(n/a)+1}^{(n/a)*2-1} (Ti, i+1) + \sum_{i=2}^{n} Tansi$$

$$M3 = T(n/a)*3,(n+1) + T(n+1),1 + \sum_{i=(n/a)*2+1}^{(n/a)*3-1} (Ti, i+1) + \sum_{i=2}^{n} Tansi$$

•
•

$$MN = T(n),(n+1) + T(n+1),1 + \sum_{i=(n/a)*(a-1)+1}^{n} (Ti, i+1) + \sum_{i=2}^{n} Tansi$$

The maximal migration and query time for the Mobile Agent model that uses multiple amounts of agents would then be:

[1] For simplicity we assume that service providers are equally divided between agents

Equation 6

$Tma = Max\ (M1, M2, M3 \ldots MN)$

In this case we measure the migration and query time of the slowest agent.

4.3 Circulating Mobile Agents model

Based on Equation 4, which depicts the total migration and query time of a single mobile agent in a network, we can present the average time for a system to be notified by one of its N circulating agents, as follows:

Equation 7

$$Tcirculating = \frac{Tmn,1 + \sum_{i=1}^{n-1}(Tmi, i+1) + \sum_{i=2}^{n} Tansi}{2N}$$

Where we assume a uniform distribution of circulating agents and similar processing times of service providers for client's information queries.

The nominator in Equation 7 presents the total time it takes for a circulating agent to start circulating agent and to arrive back to the system and informs on the information collected. This is divided by *N* Circulating Agents to denote the maximal time it takes single agent to arrive back. Finally, this is further divided by 2 to denote the average time for an agent to arrive back. In other words, every agent arrives back in Tma / N, and on average in $(Tma / N) / 2$.

This can be analyzed also from a client point of view. In the circulating model, a client that wishes to employ the circulating agents, first needs to initiate the beginning of their circulation in the network and only then after a certain amount of time request for information. In such scenario, (Tma / N) presents a top boundary for the time a client receives the information. Assuming that a client request for information is uniformly distributed, an average time for a client to receive the system information would be $(Tma / N) / 2$.

The total activation time of the circulating model from a client point of view would be:

$Tcirculating\_total\_activation = Tcirculating + Tactivation$

The activation rule to be considered for circulating agents would then be:

$Tcirculating\_total\_activation < Tavg + d$

5. Implementation of Prototype

The architectural overview of the system is depicted in figure 11. The main functionality is implemented in the WSAdvisor components; client applications interact with this functionality either directly - when requesting web service activation recommendations, or indirectly – when updating information on new possible web services in the repository. WSAdvisor components query that repository to obtain information on web services and create itinerary for communicating with the service providers. An agent server is used to launch new agents with itineraries to service providers' destinations. A communication utility object – VMSBridge is used to facilitate decoupled communication between the main WSAdvisor components and other third party applications. VMSBridge also serve as a link between different platforms. Many applications today utilize the .Net platform for their web services and applications, while mobile agents are usually java based. In such environments a utility in the form of VMSBridge is needed to transfer information between the two virtual machines.

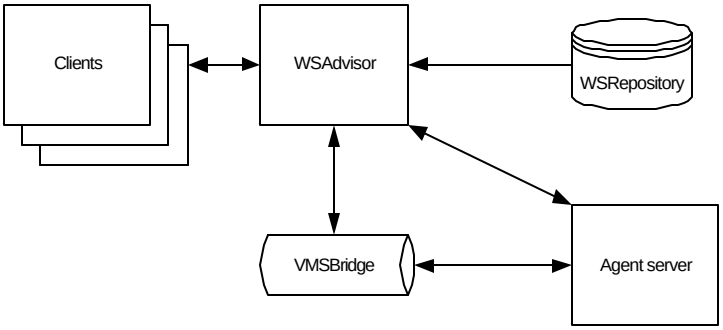

Figure 11 – high-level overview

To facilitate an integration of the three models and develop a system, which is scalable, generic and flexible, we have pursued an initial design that is depicted in figure 12.

The following section describes the functionality of the major classes in the design.

AdvisorImpl, Advisor - The *Advisor* set of classes is implemented as a proxy to be used by the client to initiate requests for information gathering on particular web services. Several modes of operations are available for the client, including synchronous, asynchronous and optimised (cached). The default and recommended mode is optimised and Asynchronous operation, in which the client is advised on recommended web services Asynchronously. The user also specifies the ranking criteria for the service providers' information. The implementation supports different numbers of client applications.

Chain and Operations - The chain class manages the flow of operations in a generic way. Operations that adhere to the interface IOperation can be added during design time or dynamically and are controlled by the chain class. We identified four types of operations, which are represented by the Operation classes: Information, Cache, Itinerary and Activator. These four basic operations identify the basic steps in the process of preparing the system to query information from the service providers.

VMSBridge - VMSBridge offers a generic and decoupled way to bridge between different environments such as Java and .Net virtual machines. The idea behind this functionality is to enable maximum flexibility of the framework to work with different third party applications. For example .Net based client applications and Java baesd agent toolkits.

WSManager, AgentManager - The manager classes control the activity of communicating with the service providers.
This activity is pursued either with an RPC or mobile agents approach.

Ranking - Upon receiving information collected from the service providers, it is sent to be ranked according to the user request. The ranking follows a multi-dimensional shortest distance approach, as described in [3]. Weights are calculated in reference to the user specifications.

DAO - A layer that communicates with a repository of web services descriptions that are dynamically added and controlled by the client.

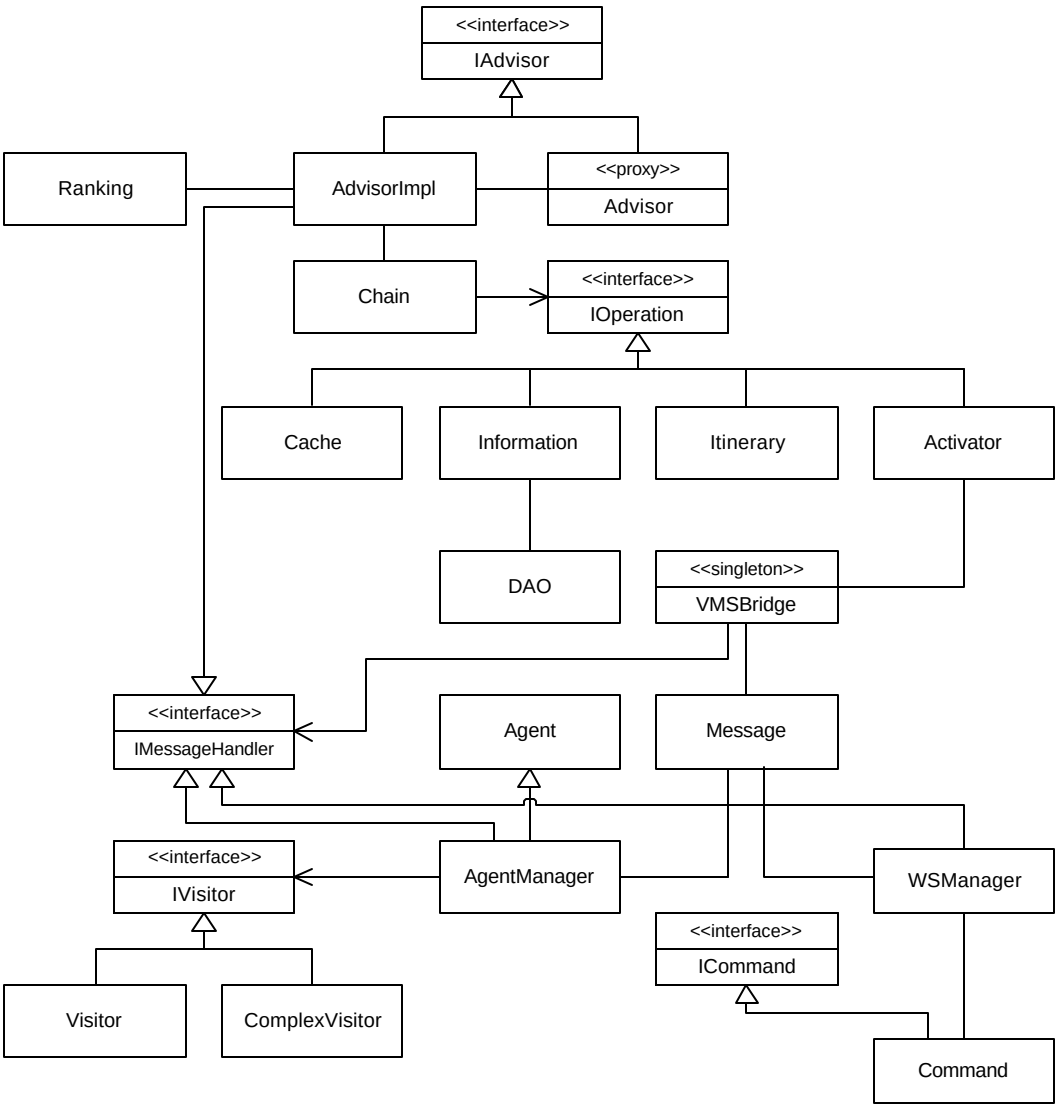

Figure 12 – system class diagram

6. Conclusions and Future Work

We are currently in the process of implementing the WSAdvisor system, which selects related service provider information for the activating client. We aim to analyze results drawn from the activation of the RPC, Mobile Agent and Circulating models and learn more about their performance characteristics in large-scale distributed environments such as the Internet. The next implementation step involves the development of a hybrid model that integrates the existing models into one.

7. References

[1] Yoshikawa C., Chun B., Eastham P., Vahdat A., Anderson T. Culler D. Using Smart Clients to Build Scalable Services *Proceedings of the {USENIX} 1997 Annual Technical Conference*

[2] Anderson T., Dahlin M., Neefe J., Patterson D., Roselli D., Wang R. Serverless Network File Systems *Proceedings of the 15th ACM Symposium on Operating Systems Principles*, pp. 109-126 December 1995

[3] Krishnaswamy, S., Pin, E, P., Ho, J., Gunawan, W., An XML Specification Language to Support A Virtual Marketplace of Data Mining E-Services**,** *2002, Proceedings of the Workshop on Data Semantics in Web Information Systems (DASWIS 2002)* held in conjunction with Third InternationaL Conference on Web Information Systems Engineering (WISE 2002), Singapore, December, pp. 194-206, IEEE Press

[4] World Wide Web Consortium (2000/2001): Web Services, eXtended Markup Language (XML),
Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL)
Available at: 4.1 http://www.w3.org/TR/ws-gloss/, 4.2 http://www.w3.org, 4.3 http://www.w3.org/TR

[5] Huhns M. N.
Software Agents: The Future of Web Services
Agent Technology Workshops 2002, LNAI 2592, pp. 1-18, 2003

[6] Hendler J., McGuinness D. L.
DARPA Agent Markup Language
IEEE Intelligent Systems, 15(6):72 73, 2001

[7] Dean M., Connolly D., van Harmelen F., Hendler J., Horrocks I., McGuinness D. L., Patel-Schneider P. F. and Stein L. A.
OWL Web Ontology Language
In Bradshaw J., editor, *Software Agents*.
MIT Press, Cambridge, 1997

[8] DAML-S: Semantic Markup for Web Srevices
Submitted for publication in *The Emerging Semantic Web.*