A Peer-To-Peer Intermediary for Building Enterprise-Class Web Services

Omar Hasan
Graduate Student
oh23@drexel.edu

Bruce Char, PhD.
Professor and Advisor
bchar@mcs.drexel.edu

Department of Mathematics and Computer Science
College of Arts and Sciences
Drexel University

Abstract

Web Services is an emerging technology for making the functionality of an application available over the Internet. Web Services provide a standard means for remote software programs to interact with each other without human intervention. Web Services have several benefits over similar technologies. These benefits include: seamless cross-platform interoperability, automatic discovery and invocation of new services, and the ability to use Web Services as software components to develop new software. Web Services with all their wonderful benefits, however, lack quality-of-service features such as security, reliability and manageability. These features are essential for enterprise-class applications. One recently introduced solution to this problem is that of a Web Service intermediary. An intermediary is placed between the Web Service provider and the client, which takes care of all the quality-of-service requirements. Although, this is a suitable solution, its current implementations have some undesirable constraints. These constraints include the requirement of additional coding to interact with the intermediary and the need for a central server in the intermediary architecture. A central server is not desirable because it limits the scalability of the Web Service and centralizes control in a single authority. We present a peer-to-peer Web Service intermediary, which is free from these problems. Due to the peer-to-peer architecture, it does not require a central server. The solution also does not require any additional coding in the Web Service provider or the client, which makes it readily deployable. Our solution is convenient and practical for building enterprise-class Web Services.

Problem

Web Services is an emerging technology that enables application-to-application interaction over the Internet. Web Services technology would be very useful to enterprises if it did not lack quality-of-service features such as security, reliability and manageability.

Web Service intermediary is a suitable solution to this problem. However, its current implementations have undesirable constraints.

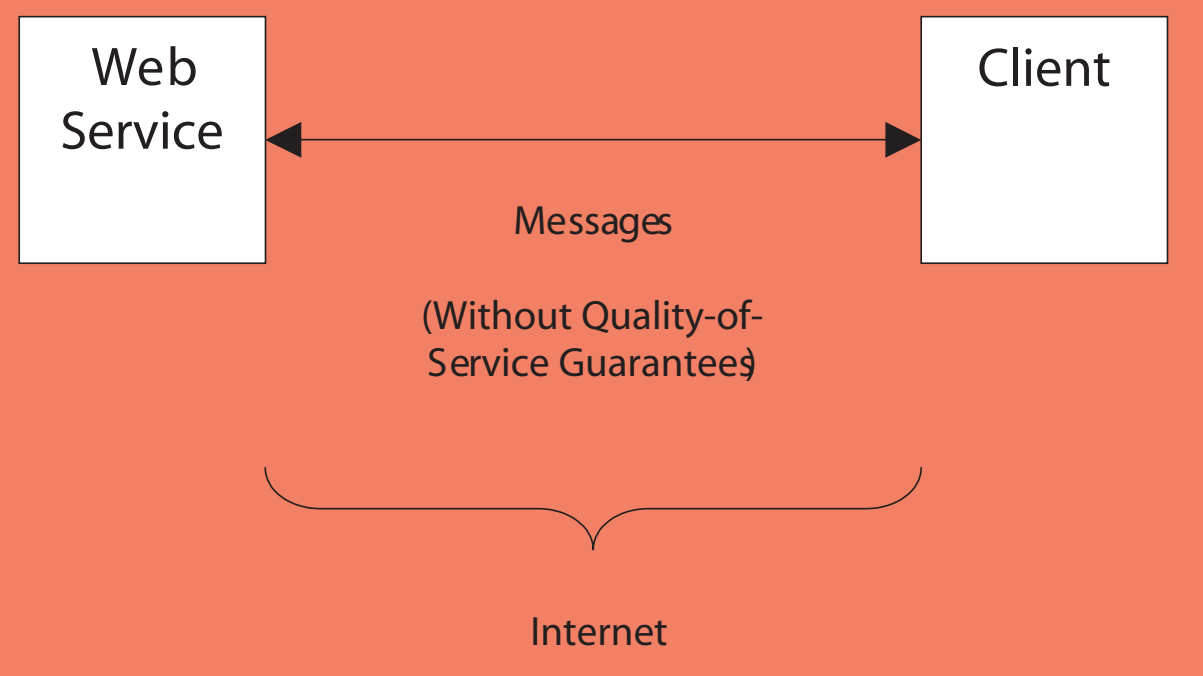

Figure: Problem

Our Solution

We have designed a peer-to-peer intermediary that is paced between the Web Service and the client. It processes the information stream and adds quality-of-service features to it. Our implementation does not have the problems associated with other existing implementations of the Web Service intermediary solution.

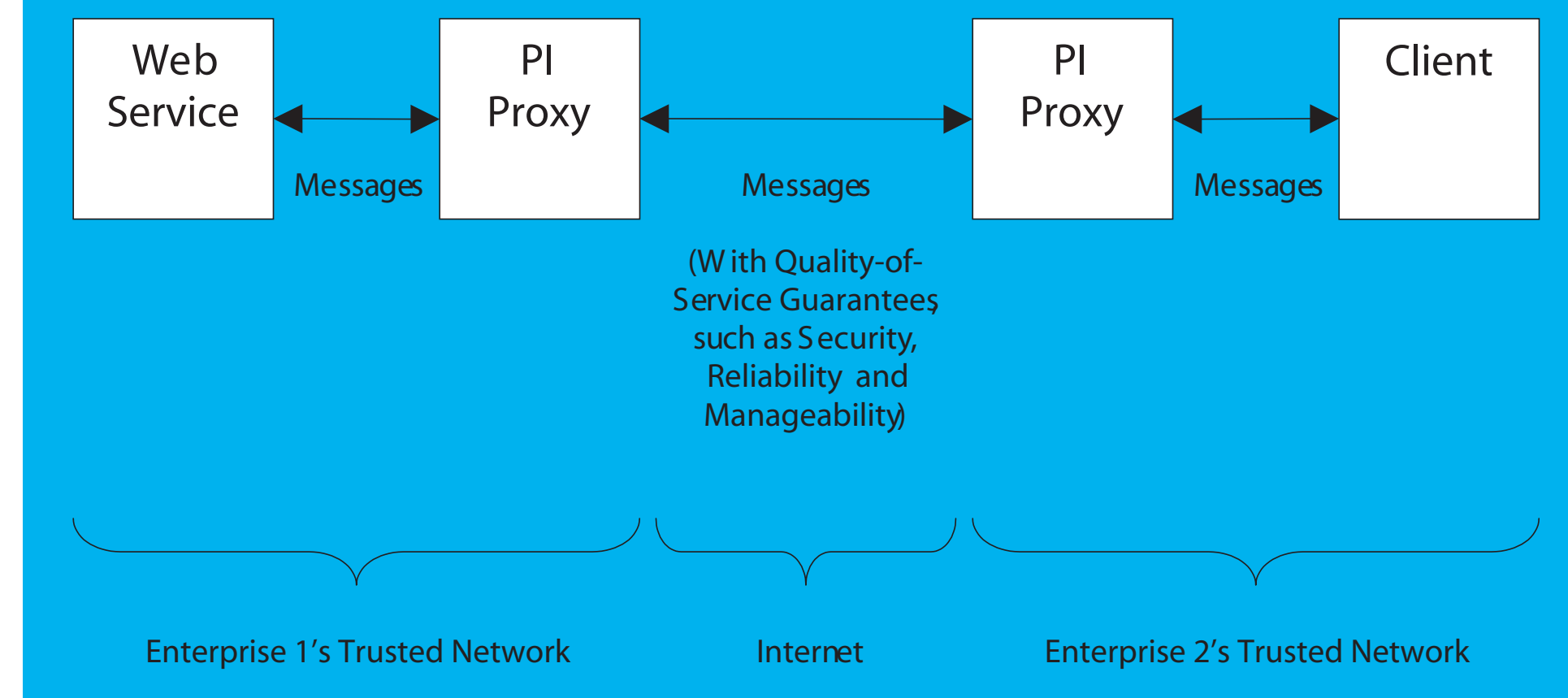
Web Service
PI Proxy
PI Proxy
Client
Messages
Messages
Messages
(With Quality-of-Service Guarantees, such as Security, Reliability and Manageability)
Enterprise 1's Trusted Network
Internet
Enterprise 2's Trusted Network

Figure: Our Solution

What are Web Services?

Web Services is an emerging technology, which can make the functions of a computer program available on the Internet. This enables any other computer program that has access to the Internet to use these functions.

Until recently, the Internet was used by humans to access information mostly on web pages. The user can read this information but that's all, it cannot be processed for gaining further advantage from it.

Let's look at some examples, which would establish this limitation and the benefits of Web Services.

Examples

1. Personalized Route Planning

Imagine that there is a Web Service that provides real-time traffic conditions of Pennsylvania roads. A computer program on a user's computer can connect to this Web Service, download the needed information and then plan the best route for the user to take from his house to his workplace. If this information were present only in the form of a web page, a program could not have processed it. The user would have had to sift through the web page to find the relevant information and then plan the route by himself.

2. E-Business

Web Services are of particular importance for business applications. An example is an e-commerce web site using a Web Service that processes users' credit card information. The same web site could also be using another Web Service provided by the postal service to determine the shipping costs.

Architecture

The Web Service and its client are two computer programs that can exchange information over the Internet without any human involvement.

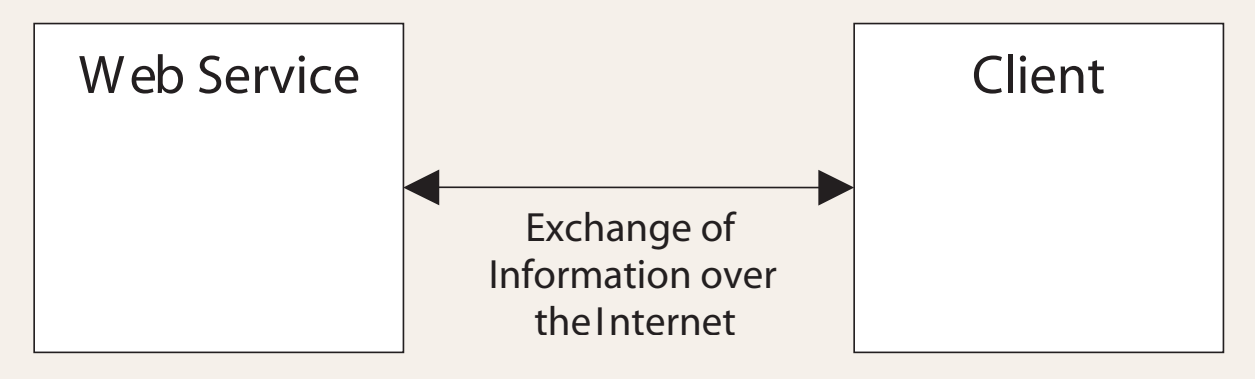

Figure: Web Services Architecture

Wait a second, Web Services sound very similar to some older technologies, such as CORBA and JAVA RMI! So how are Web Services any better?

The key difference between older technologies and Web Services is that they were not suitable for the Internet. Web Services technology is specially designed for use over the Internet.

Some other benefits of Web Services are as follows:

§ Web Services allow dissimilar applications to interoperate, irrespective of their development and operating platforms. For example an application written in COBOL running on a UNIX machine can use the Web Services technology to talk to an application written in Visual Basic running on Windows XP.

§ Web Services protocols are text-based which make them firewall-friendly. Firewalls are strict on binary protocols used by CORBA, JAVA RMI etc. because binaries can be malicious.

§ A Web Service can be used as a software module to build new software. It is also possible to chain together several Web Services. For example a Web Service that processes credit card transactions could itself be using a Web Service that provides currency conversion.

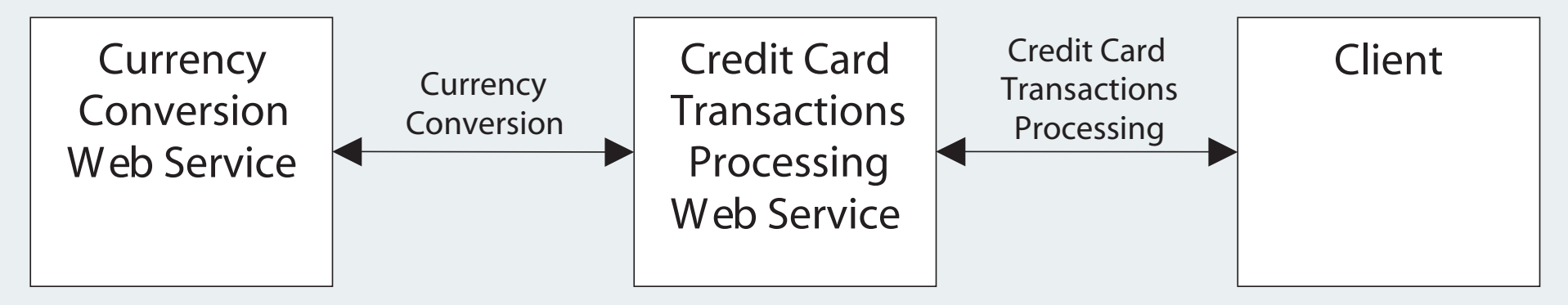
Currency Conversion Web Service
Currency Conversion
Credit Card Transactions Processing Web Service
Credit Card Transactions Processing
Client

Figure: A chain of Web Services

§ Clients can automatically discover and use new Web Services.

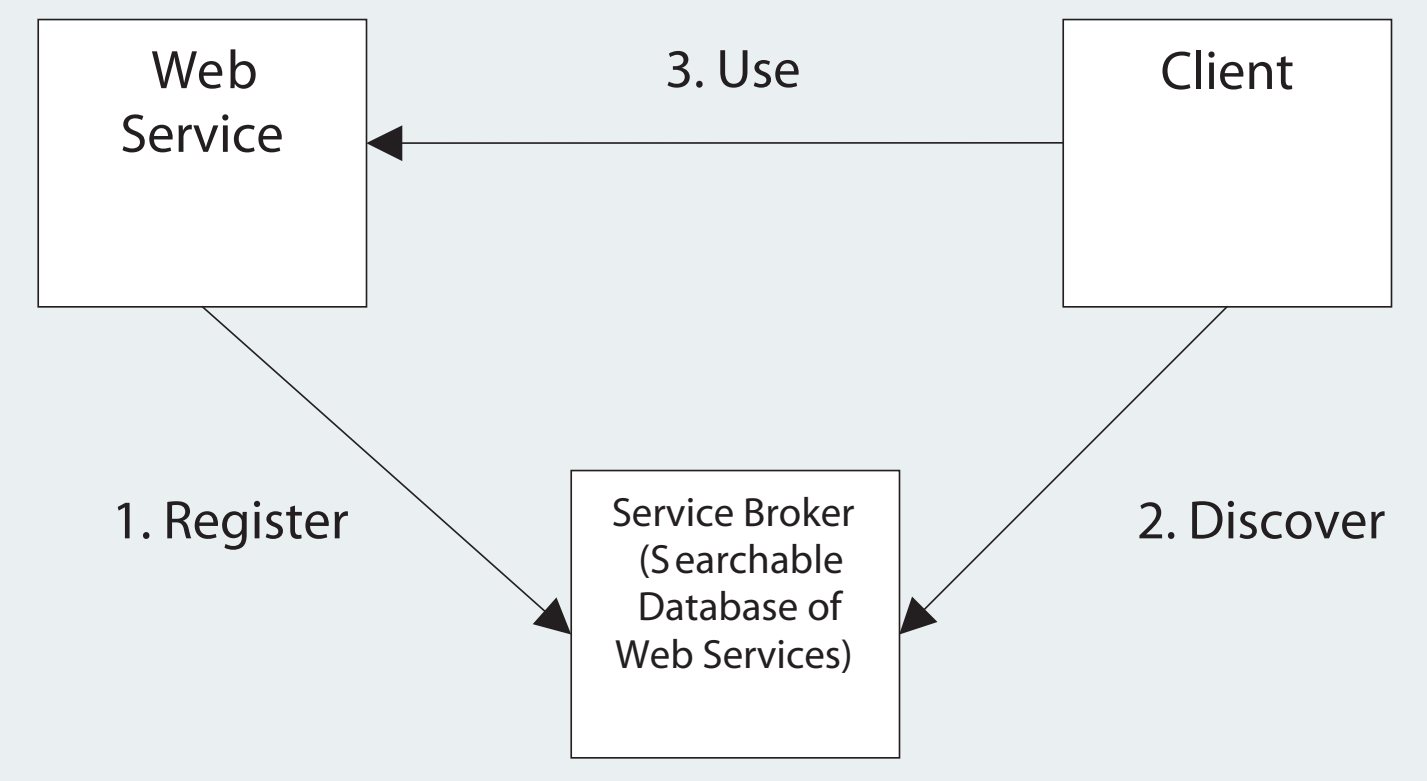

Figure: Clients can automatically discover and use new Web Services

Ok, so Web Services are wonderful, what's the problem then?

Web Services technology is currently in its initial stages. It lacks quality-of-service features such as security, reliability and manageability. These features are essential in applications used by enterprises.

A Web Service that is able to meet an enterprise's quality-of-service requirements can be termed as an enterprise-class Web Service.

Quality-of-service in Web Services refers to their non-functional properties, such as security, reliability and manageability.

Security includes:

§ Authentication of clients to prevent unauthorized access.
§ Authorization / access control to enforce a different level of access for each user.
§ Maintaining confidentiality of in-transit information.
§ Non-repudiation to guarantee that a completed transaction cannot be denied.

Reliability includes:

§ Guaranteed delivery of messages.
§ Ordered delivery of messages.
§ Exactly-once delivery of a message.

Manageability includes:

§ Monitoring of the Web Service for exceptions and performance.
§ Logging monitoring information for permanent record.
§ Easy and effective client management.

Don't people already know about this problem? Haven't they thought of solutions?

The Web Services community is well aware of these problems. Some solutions have been proposed but each of them has its drawbacks.

One solution, which does not have any serious drawbacks, is that of a Web Service Intermediary.

A Web Service intermediary is a computational element placed between the Web Service and the client.

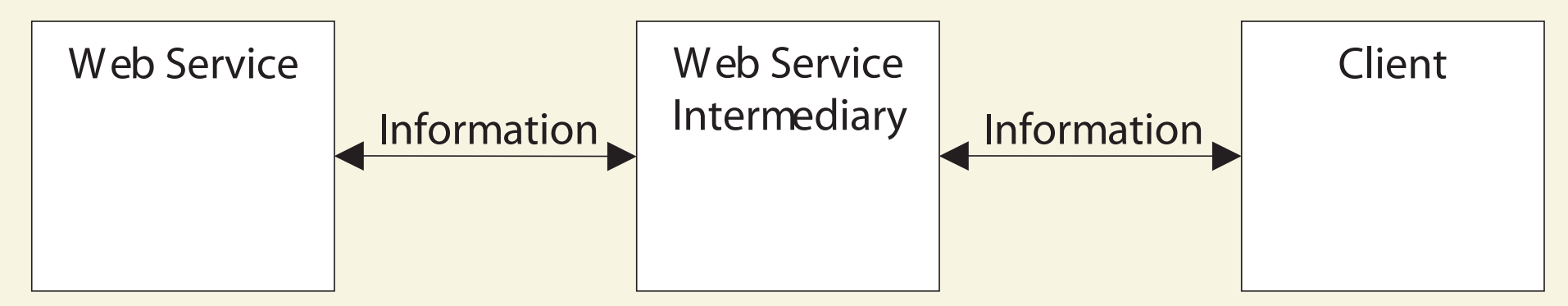

Figure: Web Service intermediary

The Web Service intermediary can process the information stream to enhance its quality. It can take care of the quality-of-service requirements by providing security, reliability and manageability services.

Web Service intermediary sounds like a good solution. Problem solved?

Although, Web Service intermediary is a suitable solution, its current implementations have some undesirable constraints. Let's take a look at an implementation by a company called Grand Central (http://www.grandcentral.com).

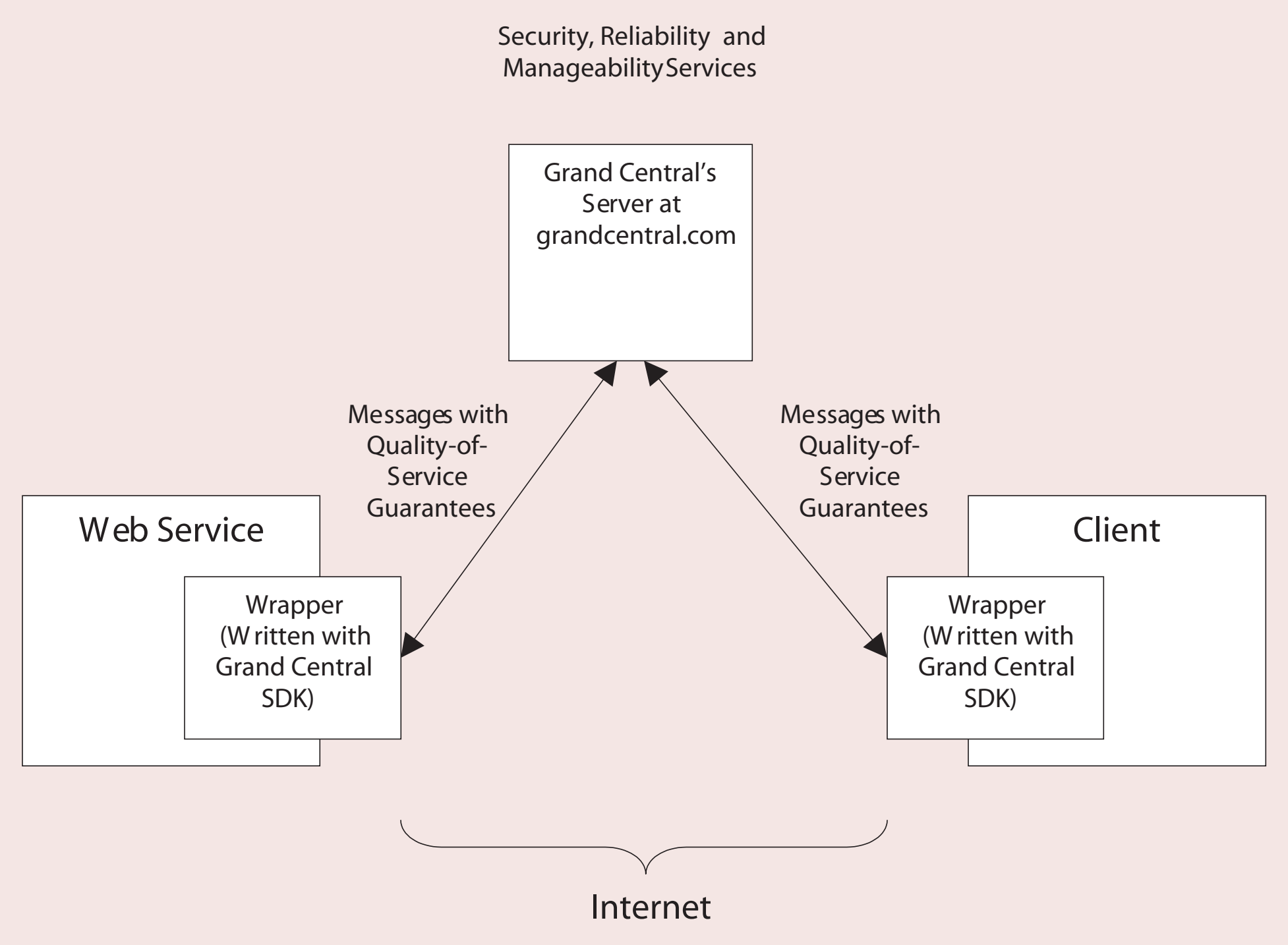
Security, Reliability and Manageability Services
Grand Central's Server at grandcentral.com
Messages with Quality-of-Service Guarantees
Messages with Quality-of-Service Guarantees
Web Service
Wrapper (Written with Grand Central SDK)
Client
Wrapper (Written with Grand Central SDK)
Internet

Figure: Grand Central's Web Service intermediary

Grand Central's Web Service intermediary comprises of a central server and a wrapper each at the Web Service and the client. All information must pass through the central server.

This implementation has the following constraints:

§ Additional coding is required to develop a wrapper that is needed to communicate with the central server.
§ The central server architecture limits the scalability of the Web Service.
§ If the server ceases to function, the whole communication would fail.
§ An enterprise has to rely on a third-party for its important business functions.

Other existing implementations have similar drawbacks.

There is always another problem! So is this the problem that your research addresses? What is your solution?

Our research addresses the above-mentioned problems in the existing implementations of the Web Service intermediary solution. The goal is to design a Web Service intermediary that:

§ Does not require additional coding.
§ Does not limit the scalability of the Web Service.
§ Does not centralize all control in a single authority.
§ Is readily deployable.

With these objectives in mind, we have designed a peer-to-peer Web Service intermediary. We call it PI (Peer-to-peer Intermediary) for short.

Architecture and Operation

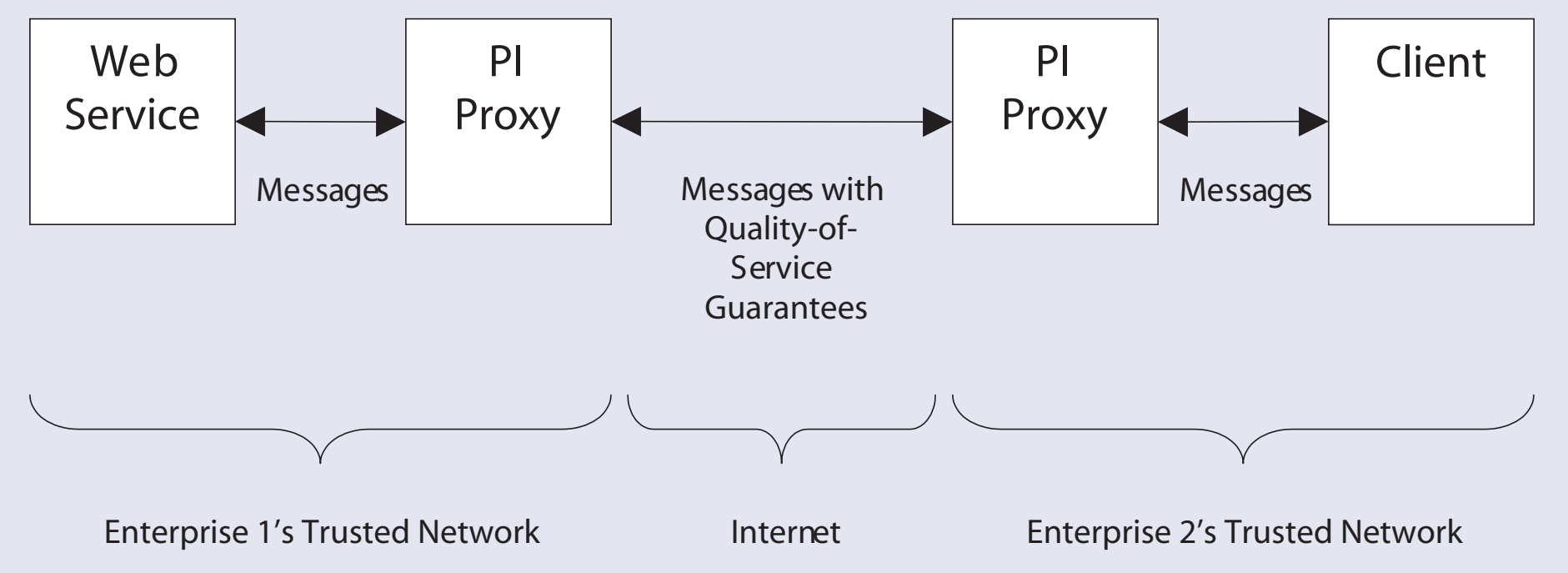
Web Service
PI Proxy
PI Proxy
Client
Messages
Messages with Quality-of-Service Guarantees
Messages
Enterprise 1's Trusted Network
Internet
Enterprise 2's Trusted Network

Figure: Our peer-to-peer Web Service intermediary

PI comprises of two peer proxies. A PI proxy is installed on each communicating party's trusted network. The Web Service and the client communicate only with the proxy installed on their own trusted network.

It is presumed that this internal communication already meets the enterprise's quality-of-service requirements. This is generally true because the proxy runs either on the same machine as the Web Service/client, or on a machine connected by a reliable Local Area Network (LAN).

When a proxy receives a message from the application inside the trusted network, it forwards it over the Internet to the proxy of the destination application. That proxy on receiving this message then delivers it to the destination application residing on its own trusted network.

The proxies interact in a peer-to-peer manner to exchange messages. The proxies cooperate with each other to implement the required quality-of-service in their communication.

Since, the two parties are allowed to select their own quality-of-service requirements, the stricter of the two are applied after an automatic negotiation.

Services Provided

PI is designed to accommodate all the quality-of-service requirements that we mentioned earlier. How each of these requirements is met is described below:

Security

§ Authentication is provided by using digital certificates and a trusted certificate authority such as VeriSign (http://www.verisign.com).
§ The Web Service manager can select the functions that are accessible to a particular client. The proxy restricts access of a client to functions that are marked inaccessible. This is done by simply checking the identity of the client and its access list before allowing it access.
§ Confidentiality is provided by public-key encryption of confidential information.
§ Non-repudiation is guaranteed by requiring the communicating parties to digitally sign their messages.

Reliability

§ Guaranteed delivery is provided by queuing messages if they are temporarily undeliverable. The proxy repeatedly keeps trying to deliver an undelivered message for a set amount of time.
§ Ordered delivery is attained by using sequence numbers and a standard FIFO ordering algorithm
§ Each message is uniquely numbered, therefore it is delivered exactly-once even if duplicate copies are received.

Manageability

§ The proxy interface displays all ongoing activity for monitoring purposes.
§ Monitoring information is logged in text files.
§ A separate profile is maintained for each client (client management). This profile contains the client's location, digital certificate, access list etc

All these services are customizable both at the Web Service and the client end. It is also worthy to note that these internal workings of the proxies are invisible to the Web Service and the client.
□

PI solves the problem

PI meets our goals, because:

§ PI does not require additional coding. The Web Service/client only need to install the PI proxy, configure it and start using it. The only difference is that instead of communicating directly, the Web Service and the client each communicate with their own proxy. This quality is due to the PI design in which it is presumed that the communication on the trusted network already has the required quality-of-service.

§ PI does not limit the scalability of the Web Service. A Web Service can have several PI proxies serving it to distribute load. This is opposed to a central server, which forms a bottleneck in the communication.

§ Due to its peer-to-peer design, PI does not centralize control in a single authority. Administrators of both the Web Service and the client are each free to select their own quality-of-service requirements. The stricter of the two are applied after automatic negotiation between the PI proxies.

§ Since PI does not require additional coding, it is readily deployable.

Do you also plan to implement this solution as a product?

An implementation of PI is currently under development. Sun Microsystems' Java/J2EE is being used as the development platform. Java was chosen because it provides a comprehensive Web Services Development Pack (WSDP) and also because applications written in Java are platform independent.

Conclusion

Ok, I now understand your work. So Web Services are a wonderful new technology to make remote computer programs interact with each other. But this technology has the problem that it can't provide features such as security, reliability and manageability, which are important to enterprises. Web Service intermediary is a good solution but all its current implementations are problematic. You have designed a new implementation of the Web Service intermediary solution, which is problem-free. It does not require additional coding and due to its peer-to-peer design, it does not limit the scalability of the Web Service or centralize all control in a single authority. Because of your work, now enterprises can benefit from Web Services without worrying about quality-of-service.