Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 8 (5): 730-742, 2008
ISSN 1812-5654

A Hybrid Architecture for Implementing Efficient Geospatial Web Services: Integrating .Net Remoting and Web Services Technologies

Pouria Amirian and Ali Alesheikh
Faculty of Geodesy and Geomatics Engineering, K.N. Toosi University of Technology,
Vali-e-asr St., Mirdamad Gross, Tehran, Iran

Abstract: Open GIS Consortium (OGC) Geospatial Web services have been introduced to overcome spatial non-interoperability problem associated with most geospatial processing systems. Although OGC geospatial Web services provide interoperability among heterogeneous geospatial processing systems, in some cases they can not provide required performance and efficiency. This study proposes a hybrid architecture which can efficiently provide interoperability and high performance for transferring geospatial data. It is suggested that making use of Web services technologies for implementing OGC geospatial Web services would significantly facilitate sharing geospatial data in heterogeneous environments like Web. In addition, making use of a proprietary and platform-dependant technologies can provide best performance and efficiency in homogeneous environments like an internal network. In this context, design and development of an OGC geospatial Web service using hybrid architecture of Web services Technologies and .NET Remoting technology (as proprietary and .NET specific technology) is described. Based on our evaluations and practical tests, the hybrid architecture proved to be an efficient solution for development of geospatial Web services.

Key words: Interoperability, geospatial web service, web services technologies, distributed application, .NET platform, .NET remoting

INTRODUCTION

Majority of geospatial processing systems require some level of interoperability as a fundamental capability. Based on OGC Reference Model (Open GIS Consortium, 2003), spatial interoperability refers to capability to communicate, execute programs, or transfer geospatial data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units. Therefore, non-interoperability of geospatial processing systems prevents sharing of geospatial data and services among software applications. Spatial interoperability faces two main challenges; syntactic heterogeneity and semantic heterogeneity (Worboys and Duckham, 2004). Syntactic heterogeneity which is the result of differences in storage formats and software incompatibility is a technical issue and can be addressed by technical means. Semantic heterogeneity arises as a result of incompatibility in meanings of data. Addressing syntactic heterogeneity is the main concern of this study.

Syntactic heterogeneity of geospatial information systems can be categorized in data and access heterogeneity. In data heterogeneity geospatial processing systems use various internal proprietary data formats. To share geospatial data, converters and/or transfer formats must be developed, which is a resource and time consuming task. In addition, there are so many different standards for geospatial data that converting various data formats can itself become a barrier to interoperability.

Access heterogeneity restricts inter-process communication among various geospatial processing systems, since different vendors' geospatial processing systems use proprietary software access methods with proprietary software interfaces. In other words, interface definition languages, communication protocols, communication ports and even object transfer mechanisms, varies in each software development platform. So the software platform which has been used to develop the geospatial processing system imposes the use of specific and proprietary communication methods among various parts of the system. For this reason, different geospatial processing systems that have been developed by different software development platforms, cannot communicate and share services automatically and in an interoperable manner.

Corresponding Author: Pouria Amirian, Faculty of Geodesy and Geomatics Engineering, K.N. Toosi University of Technology, Vali-e-asr St., Mirdamad Gross, Tehran, Iran

In GIS community, OGC has introduced specific kind of online services, to overcome spatial non-interoperability problem. These services which are called OGC geospatial Web services (or Geospatial Web Services for short) have been developed with the goal of sharing geospatial data and services among heterogeneous geospatial processing systems. Web Feature Service (WFS), Web Map Service (WMS) and Web Coverage Service (WCS) are the most fundamental geospatial Web services which are introduced by OGC. At the same time, in Information Technology world, the best solution for providing interoperability among heterogeneous software systems in distributed and decentralized environments are Web services technologies (Volter *et al*., 2005).

Geospatial Web services and Web services differ in a way that Web services are composed of particular set of technologies and protocols but Geospatial Web services are comprised of defined set of interface implementation specifications which can be implemented with diverse technologies (Amirian, 2006).

Although Geospatial Web services provide interoperability among heterogeneous geospatial processing systems, in some cases they may not be the best tools for sharing geospatial data and services in network environments. In general, geospatial Web services are defined to be employed over the World Wide Web which uses HTTP (Hyper Text Transfer Protocol) to communicate between clients and servers. Plain and textual nature of HTTP decreases the total performance and efficiency of a system which has exploit its functionality and geospatial data using geospatial Web services. More accurately, making use of HTTP limits geospatial Web services to provide geospatial data (in feature level) in textual formats like Geography Markup Language (GML). HTTP is originally designed to be a transport protocol and lacks many of the functionalities which are built into binary and proprietary transfer protocols (Selly *et al*., 2006). As a result, geospatial Web services are not considered as fast and efficient solutions in homogeneous platforms (same operating system and software development platform). In homogenous environments using proprietary technologies provide resourceful and efficient solution to providing geospatial data and processing services.

Providing efficient and fast services for sharing geospatial data and services in an internal network is one of the main challenges in many organizations. National Cartographic Center of Iran (NCC) is a good example of such organizations. NCC is the main producer and maintainer of geospatial data in various scales in Iran. There are a lot of departments in the NCC which are in charge of producing and updating geospatial data (Remote Sensing Department, Photogrammetry Department, Geodesy department, Field Surveying Department and so forth). In addition there are some other departments which are in charge of making informational products from the produced or updated geospatial data (like producing digital terrain models from height point data which are produced using Photogrammetry approach). In other words, in National Mapping Agency (NMA) of Iran there are some departments which produce and update geospatial data. In addition there are two major classes of consumers for the mentioned geospatial data; there are a lot of NMA's departments which need geospatial datasets to accomplish their daily business (producing informational products) through NMA's internal network and there are quite a lot of organizations (such as municipalities, ministries, private sector, universities and disaster response communities) which demand those geospatial data via Internet and Web.

In this situation, making use of OGC Web services which are developed using Web services technologies provide the best solution to publish geospatial data to the organizations (external consumers) in an interoperable and cross-platform manner. Besides, using platform-dependant proprietary technologies provide the best performance to provide geospatial data to NMA's internal departments.

With respect to above description, this paper proposes a hybrid architecture which can efficiently provide interoperability and high performance for the both classes of consumers for NMA's geospatial data. It is suggested that making use of Web services technologies for implementing OGC Web services would significantly facilitate sharing geospatial data in heterogeneous environments like Web. In addition, making use of a proprietary and platform-dependant technologies can provide best performance and efficiency in homogeneous environments like an internal network of an organization. This paper describes the design and development of an OGC Web service using Web services Technologies and .NET Remoting technology (as proprietary and .NET specific technology) to achieve spatial interoperability over Web, while providing efficient and fast solution over internal network of NMA of Iran. The paper first assesses distributed applications (particularly distributed object technologies), .NET Remoting technology, Web services and Geospatial Web services concepts. In order to evaluate the proposed strategy and architecture, practical outcomes and performance evaluations for transferring some geospatial feature classes as well as various system configurations of the implemented OGC Web service are presented.

DISTRIBUTED ARCHITECTURE AND DISTRIBUTED COMPONENT TECHNOLOGIES

The architecture of a software application is the structure and configuration of its components. One of the most important characteristics of a software application is its ability to share data, information and processing services. In this context distributed architectures are often used to achieve high level interoperability in many types of systems, including GIS, databases and location-based services (Worboys and Duckham, 2004). Generally, by dividing an application into multiple layers (or tiers), it becomes possible for several computers to contribute in the processing of a single request, thus making application a distributed application. This distribution of logic typically slows down individual client requests (because of the additional overhead required for network communication), but it improves the scalability for the entire system. Distributed applications are implemented based on distributed architecture which enables them to make use of remote computers processing resources. More accurately, a distributed architecture is a collection of independent computers that appears to its users as a single coherent system (Tanenbaum and Van Steen, 2002).

There are several technologies and concepts for implementing distributed applications. In this context distributed component technologies provide an efficient software pattern and platform for implementing distributed applications (Hariri and Parashar, 2004).

The main concept behind all the existing distributed component technologies, is to ask objects which are running on remote machines to perform computational tasks on behalf of the objects residing in a local machine. Objects which are resides in local and remote machines are in different memory spaces and communicate via messages over a network through predefined communicational channels and protocols.

Out of the existing alternatives, Java-RMI (Remote Method Invocation) from Sun Microsystems, CORBA (Common Object Request Broker Architecture) from Object Management Group, DCOM (Distributed Component Object Model) and .NET Remoting from Microsoft and the Web services are the most popular distributed-component technologies among researchers and practitioners (Hariri and Parashar, 2004).

Web services are unique distributed component technology; since they are a defined set of open and non-proprietary eXtensible Markup Language (XML) vocabularies, they provide true interoperability among heterogeneous software platforms. But the unprecedented level of interoperability provided by Web services comes at a price; Web services are quite slower than the other distributed component technologies.

Interoperability is especially important to users of GIS, as geospatial analysis often relies on the integration of data from multiple resources. Considering unique characteristics of geospatial data and geoprocessing needs, high performance and communication speed is as important as interoperability.

In order to achieve high performance as well as interoperability, it is proposed that using hybrid architecture of Web services (as non-proprietary and platform-independent technologies) and an efficient and high performance distributed component technology (as proprietary and platform-dependent technology) can solve the mentioned issue. As mentioned before, many organizations (like national mapping agencies) need efficient and interoperable services for providing geospatial data to internal and external departments and organizations through internal and external communicational networks.

In this research Microsoft's .NET platform was used for implementing such services. In the mentioned platform, .NET Remoting is an infrastructure which enables inter-process communication.

.NET remoting: .NET Remoting was introduced as part of Microsoft's .NET platform. From the developer's perspective, .NET platform is a huge set of types (Classes, Structures, Namespaces, Interfaces and Delegates), which are fill in the place of the older Windows programming Application Programming Interfaces (API) such as the Windows 32-bit API and the Microsoft Foundation Classes, Web APIs like Active Server Pages and with respect to distributed component technologies it substitute DCOM. From component oriented view point, ordinary .NET components are replacing DCOM Components and .NET Web Service components are intended to be cross-platform and cross-language Web-based components (Wang and Qian, 2005).

.NET Remoting allows client applications to instantiate objects on remote computers and uses them like local objects. .NET Remoting is the ideal choice for intranet scenarios. It typically provides the greatest performance and flexibility, especially if remote objects have to be maintained their state or there is a need for creating peer-to-peer applications (MacDonald, 2003).

Using .NET Remoting in an enterprise application means hosting software components on separate computers and make them available through .NET Remoting infrastructure. Software components which are hosted on remote computers are called remote components. Remote components have to be hosted and activated using a component host. Component host typically is a service on Microsoft Windows which is responsible for listening to client requests. In .NET

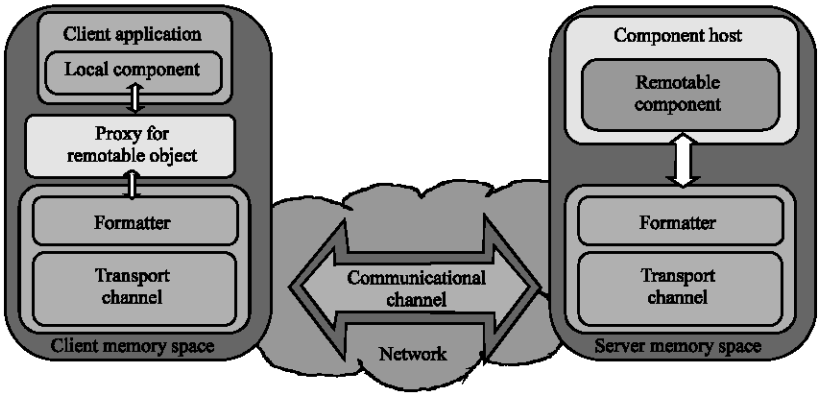

Fig. 1: .NET remoting architecture

remoting, a remote component can be hosted in one of several different types of applications. It can be hosted in a typical desktop application, dedicated Windows service or by Microsoft's main Web server software (IIS).

With .NET Remoting, the communication between local and remote components is accomplished through a proxy layer. This approach is conceptually similar to the way other distributed object technologies communicate. With this type of communication, local component communicates directly to proxy layer which mimics the remote component's interface; that is when the local component calls a method on proxy layer, it calls the remote component behind the scene, waits for response and then returns the appropriate information to local component (Fig. 1). In .NET Remoting, when communicating with remote component, proxy layer does not actually perform all the required tasks for remote communication. Instead, it communicates with a formatter object and transport channels to perform the remote communication. The formatter object packages the client request or server response in an appropriate format then communicates with a transport channel which transmits the packages using the appropriate protocol. Figure 1, shown how a method invocation is packaged on the client side and then unpacked on the server side. In this case, the information flows from the client to the server, but the return value flows back from server to the client along the same communicational path.

By default, .NET Remoting provides two formatters (binary and SOAP formatters) and two channels: HTTP and Transmission Control Protocol (TCP) channels. Binary formatter packages the request and response to a compact, proprietary .NET format. As a result, this formatter offers best performance but can be used only by .NET applications. SOAP (Simple Object Access Protocol) formatter serializes the request and response to SOAP messages, which is a cross-platform XML-based plain-text format. SOAP format requires larger message size, therefore it can reduce overall performance but it provides cross-platform communication with a remote .NET component (Rammer, 2005).

There are also two predefined transport channels; TCP channel and HTTP channel. As the name implies, TCP channel uses the TCP protocol and it is ideal for internal networks. At the other hand, HTTP channel uses the HTTP protocol which is ideal for scenarios in which there is a need for communicating over the Web.

.NET Remoting is not the only way of implementing distributed applications in .NET platform. XML Web services (which are the concrete implementation of Web services technologies) are among the most widely hyped features of Microsoft .NET platform. Next section introduces Web services technologies briefly.

Web services technologies: Web services technologies are the implementation of a conceptual architecture, which is called Service Oriented Architecture (SOA) (Marks and Werrell, 2003). SOA is frequently characterized as a style that supports loose coupling, business alignment and Web-based services, which permits extensibility and interoperability independent of the underlying technology (Murakami *et al*., 2007). SOA provide a set of guidelines, principles and techniques in which business processes, information and enterprise assets can be effectively (re)organized and (re)deployed to support and enable strategic plans and productivity levels that are required by competitive business environments (Papazoglou and Van del Heuvel, 2006).

In SOA, the central elements are services. In many respects, a service is the natural evolution of the component, just as the component was the natural evolution of the object. Services are autonomous, platform-independent computational elements that can be described, published, discovered, orchestrated and

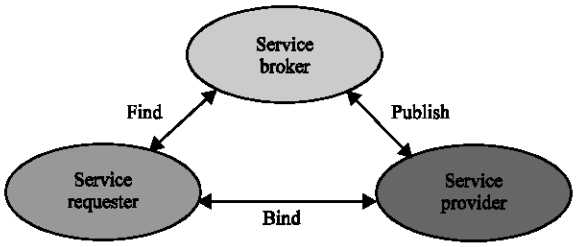

Fig. 2: Major components of service oriented architecture

programmed using standard protocols for the purpose of building networks of collaborating applications distributed within and across organizational boundaries (Papazoglou and Van del Heuvel, 2005).

Service-orientation is the correct way to build maintainable, robust and secure applications (Lowy, 2005). As shows in Fig. 2, SOA consists of three primary roles and three primary tasks (Huhns and Singh, 2005). Service provider, service requester and service broker are distributed computational nodes in the network environment. Service provider publishes its own service with service broker. Service requester uses the service broker to find desirable services and then binds to a service provider to invoke the service (Fig. 2).

SOA architecture allows obtaining a loose coupling between its processing components (service requester and service provider), because, it uses simple generic and application-independent connectors and messages defined by an XML schema (Fallside and Walmsley, 2004).

The actual implementation of SOA using open, standard and widely used protocols and technologies is called Web services (Marks and Werrell, 2003). Web Services are the basic components of distributed service-oriented systems. The World Wide Web Consortium (W3C) defines Web Services as a software system designed to support machine-to-machine interaction over the Internet (Stal, 2002; Booth *et al*., 2004; W3C, 2006).

Any Web service has an interface described in a machine-processable format. Other systems and services interact with the Web service in a manner described by its description using messages. Messages are conveyed typically using HTTP with an XML serialization, in conjunction with other Web-related standards, but any other communication protocol can be used as well (Booth *et al*., 2004).

Web services are based on open standards, so they provide interoperability in decentralized and distributed environments like Web. These new technologies can be developed by using any software platform, operating system, programming language and object model.

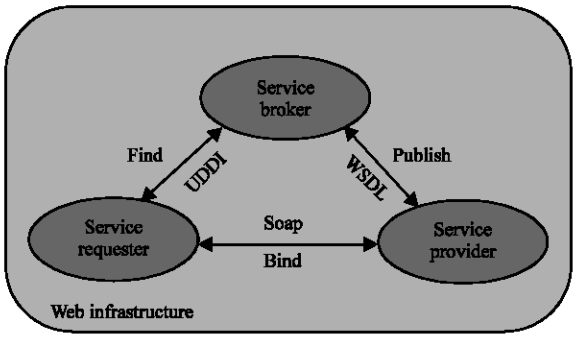

Fig. 3: Mapping between SOA and Web services technologies

Web services are implemented using a collection of standards and technologies. SOAP, Web Services Description Language (WSDL) and Universal Description Discovery and Integration (UDDI) form the core technologies for implementing Web services (Gailey, 2004).

SOAP is a lightweight, XML-based protocol for exchanging information in decentralized, distributed environments. SOAP is used for messaging among various SOA's components in a Web services platform. SOAP is platform independent and also it can be used with virtually any Network Transport protocols.

WSDL is XML-based specification for describing the capabilities of a service in a standard and extensible manner. Technically, WSDL defines the software interface of Web service in platform independent approach.

UDDI is a set of specifications and APIs for registering, finding and discovering services.

As shown in Fig. 3, these layers establish an explicit mapping between elements of SOA as a conceptual and technology independent architecture and Web services as specific collection of standards, protocols and technologies.

In this case, Web service provider publishes its own service description using WSDL, then Web service requester take advantage of search API's of UDDI to find appropriate Web services and finally, Web service requester binds to the service provider using SOAP.

From middleware point of view, Web service technologies are one of distributed component technologies. But the goal of Web services goes beyond those of classical distributed component technologies such as .NET Remoting and CORBA: Web services aim at standardized support for higher level interactions such as service and process flow orchestration, enterprise application integration and provision of middleware of middleware (Vinoski, 2003). Instead of building

applications that result in collections of objects or components that are firmly integrated and understood just in development time the service approach of Web services platform is much more dynamic and is able to find, retrieve and invoke a distributed service dynamically. Another key difference is that with Web Services the industry is solving problems using technologies and specifications that are being developed in an open way, via partnerships and consortia such as the W3C and the Organization for the Advancement of Structured Information Standards (OASIS) and using standards and technologies that are the basis of the Internet. Next section will briefly explains the major differences between .NET Remoting and Web services technologies.

Web services vs. .NET remoting: Implementing SOA using Web services can be taught as adaptation of distributed component technology with Web infrastructure. Some of the important advantages of using Web services as the technology platform for implementing SOA are derived from the way in which the World Wide Web achieved its tremendous success; in particular, the fact that a simple markup language (XML) can provide a powerful interoperability solution and the fact that a lightweight document transfer protocol (HTTP) can provide an effective, universal data transfer mechanism.

The following items describe major differences between .NET Remoting and Web services technologies (or more accurately XML Web services in .NET platform): Components published via Web services are more restricted than components exposed over .NET Remoting. Web services are designed around the stateless exchange of messages. In .NET Remoting it is possible to expose stateful components as well.

Since .NET Remoting Infrastructure can use binary formatter and TCP channel, communication with .NET Remoting is faster than Web service communication.

Web services support open standards which target cross-platform use. Any client that can parse XML message and connected over an HTTP channel can use a Web service, even if the client is written in Java and hosted on Linux it can consume a Web service which is written in C# and hosted on Windows.

Web services are designed for use between organizations. They can use a discovery mechanism or a UDDI registry that advertises services to interested parties over the Web. With .NET Remoting there is no clearly defined registry service or specification.

Web services are firewall-friendly. This means since most Web services communicate through HTTP channel, there is no need for an administrator to open additional ports on firewall. In contrast, with .NET Remoting any port can be utilized for inter-process communication, doing so could leave a major hole in security.

Unlike Web services .NET Remoting can be used for implementing peer-to-peer applications in which individual clients communicate back and forth and there is no central server.

.NET Remoting is more suitable as a high-speed solution for binary communication between proprietary .NET components usually over internal networks. Although Web services can not match the communication speed and stateful communication scheme of .NET Remoting, they can be used in cross-platform scenarios elegantly.

GEOSPATIAL WEB SERVICES

In general, geospatial Web services are particular kind of online services which deal with geospatial information. Geospatial Web services can do the following (Lake *et al*., 2004):

- Provide access to geospatial information
- Perform geospatial analysis
- Return messages that contain geospatial information, which can be delivered as image, text, numeric data or geographic features

In this context, OGC has defined a comprehensive framework of geospatial Web services which is known as OGC Web service framework. OGC Web service framework consists of interface implementation specification and encodings which are openly available to be implemented by developers. The interface implementation specifications are software technology-neutral details about various operations of each OGC Web service. The encodings provide the standard glue among different parts of geospatial Web services. Each service of this framework can be implemented using various software technologies and systems. The most fundamental services and encodings of the OGC Web service framework are WMS, WFS, Geography Markup Language (GML) and Common Query Language (CQL).

WMS intends to share and publish geospatial data in plain image format. WFS is the main geospatial Web service for publishing and requesting vector geospatial data in GML format. WMS and WFS and other geospatial Web services provide standard access method to geospatial data thus provide interoperability in GIS community.

According to GML specification (Open GIS Consortium, 2005a), GML is an XML grammar written in

XML Schema for modeling, transporting and storing geospatial information. GML provides data interoperability among heterogeneous geospatial processing systems. CQL is XML-based format for defining queries on geospatial data. This language provides standard grammar for making query statements, independent of any software system.

The OGC Web services differ from the Web services. Following items explain the differences between them in detail (Amirian, 2006):

In the OGC Web service framework, HTTP is defined as the sole distributed computing environment. In contrast, Web services can be implemented virtually with any standard protocols such as HTTP, FTP and TCP.

OGC Web services do not necessarily use the usual Web services core standards, including SOAP and WSDL. In other words, in Web services technology, the main messaging protocol is SOAP and this protocol can be considered as the main cause of achieving interoperability among various applications which are running on heterogeneous platforms. In OGC Web service framework, SOAP is not the main messaging protocol. In addition in most geospatial Web services, creation and publication of WSDL document has not been defined yet. OGC Web services have particular interface for binding that might bring about interface coupling problem. In accordance with OGC Web service framework specifications, each geospatial Web services have to publish its capabilities through a so called capabilities document. This document (which is an XML document) provides human and machine-readable information about the geospatial data and operations supported by a specific instance of a geospatial Web service. But this document is not comprehensive enough to play a same role as WSDL document. In other words, capabilities document cannot offer enough information to enable developers and thus software applications to consume a geospatial Web service programmatically and automatically. While according to Newcomer and Lomow (2005), ideally the service requester should be able to use a service exclusively based on the published service contract.

In OGC Web service Framework, Common Query Language (CQL) is used to specify which geospatial data have to be sent back to the requester. This language can cause interoperability problems when considering the various scenarios in which there is a need for geospatial Web services to communicate with other types of Web services (non-geospatial). This language has unconventional structure when compared with other standard query languages such as Structured Query Language (SQL) and XQuery. The structure of CQL just increases the software development costs and also use of this language has the potential of causing interface binding problem. Interface binding problem is the main barrier in front of making truly loosely coupled services.

In a nutshell, OGC's Web services and Web services are compatible with each other, but they are not technically implemented in the same way.

As mentioned before, OGC Web services can be implemented using existing software development technologies. It is suggested that using Web services technologies as enabling infrastructure for implementing OGC Web services can significantly facilitate sharing geospatial data as well as access to processing services from multiple resources in and out of geospatial community. But Web services can not match the communication speed and stateful communication scheme of .NET Remoting, thus they are not suitable for internal networks. As a result proposed hybrid architecture of Web services technologies and .NET Remoting can provide two types of communication schemes for internal (intranet) and external (Internet) consumers of geospatial Web service.

HYBRID ARCHITECTURE FOR IMPLEMENTING EFFICIENT AND INTEROPERABLE GEOSPATIAL WEB SERVICES

In order to evaluate the hybrid architecture (of Web service and .NET remoting infrastructure) to serve as a proof-of-concept, to guide its evolution and to evaluate and to define technical solutions, a prototype system was implemented. This section describes the architecture and capabilities of the implemented system.

In this research WFS (Basic WFS) is selected to be implemented using hybrid architecture. Three main operations have been defined for basic WFS (Open GIS Consortium, 2005b):

GetCapabilities: The purpose of this operation is to obtain service metadata, which is a machine readable (and also human-readable) description of the required technical information for consuming WFS. The most important part of the service metadata (or capabilities document) indicates which feature types the WFS can provide and what operations are supported for each feature type.

DescribeFeatureType: WFS describes the structure or schema of any feature type it can provide using DescribeFeatureType operation. This structure will be used for retrieving geospatial data in GetFeature Operations.

GetFeature: This operation is used for retrieving geospatial data. Making use of CQL, spatial and non-spatial criteria can be introduced to retrieve the necessary GML data from WFS.

The above operations provide the software interface of the WFS system. Internal details of the functional units and software components as well as communications are transparent to consumer applications; the consumer application just communicate with the WFS system through the operations and defined set of parameters which are specified in WFS implementation specification. Software components, communication among them and physical location of each component are specified in logical and physical architecture of the WFS system.

Physical architecture is quite different from a logical architecture. The physical architecture is about the number of machines or network hops involved in running the application. Rather, a logical architecture is all about separating different types of functionality in software components (Lhotka, 2006).

Traditionally, logical architecture of software applications consists of three tiers; presentation and user interface tier, business logic tier and data management tier (Shalloway and Trott, 2004). With the advent of new technologies and software design patterns the traditional logical architecture is rarely efficient for the modern software applications. In this research the WFS designed in four logical tiers (Fig. 4).

As the name implies, the presentation and user interface tier (Fig. 4) provides the end user an appropriate and friendly user interface which hides details of local and remote computational tasks from user. In the case of WFS, this tier is responsible for gathering the user inputs, validating the user inputs, composing CQL statements based on the user inputs to make requests, validating requests against proper schemas, sending validated requests to WFS server and displaying the returned geospatial data. The business logic tier includes all business rules for the WFS system. For the implemented WFS these rules consist of translating requests to DBMS specific query language statements and dispatching them to the next tier.

Data access tier interacts with the data management tier to retrieve, update and remove information. The data access tier does not actually manage or store the data; it merely provides an interface between the business logic and the database. Logically, defining data access as a separate tier enforces a separation between the business logic and how application interacts with a database (or any other data source).

The forth tier (data management tier) handles the physical retrieval, update and deletion of data. This is different from the data access tier, which requests the retrieval, update and deletion of data. Often these operations are implemented within the context of a full fledged database management system.

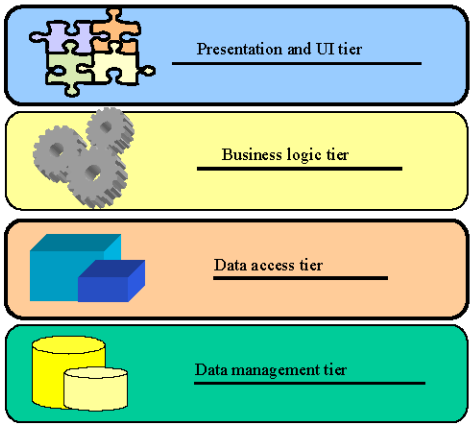

Fig. 4: Four tiers logical architecture of the WFS

In order to integrate .NET Remoting and Web services technologies (which are called XML Web services in the realm of Microsoft .NET) a hybrid architecture were designed (Fig. 5). In this hybrid architecture, there are two types of clients; internal clients and external clients. Internal clients utilized internal software system which is the proprietary software that manages some part of the daily task of running business. For internal software system a .NET Remoting-based solution provides the best possible communication speed. For exposing functionality of the internal software system to third-parties (external clients), XML Web service-based solution provides best application integration possibilities. Since XML Web services are platform independent, third-party clients can bind to the published service and gain the same functionality as the internal clients.

Both Web services and .NET Remoting use the same components to retrieve geospatial data and expose them to interested clients (Fig. 5). Doing so, it is ensured that both types of clients will utilize the same version of the components of business logic and data access tiers. Also using the same backbone component reduces the deployment and update costs.

The mentioned four tier logical architecture have been developed using Microsoft .NET 2.0 framework and SQL Server 2005 DBMS. .NET windows forms were used to implement the client side application for both internal and third-party clients (user interface and presentation tier). Windows forms provide much more flexibility and capability to use the client machine resources when compared with browser based applications. In addition,

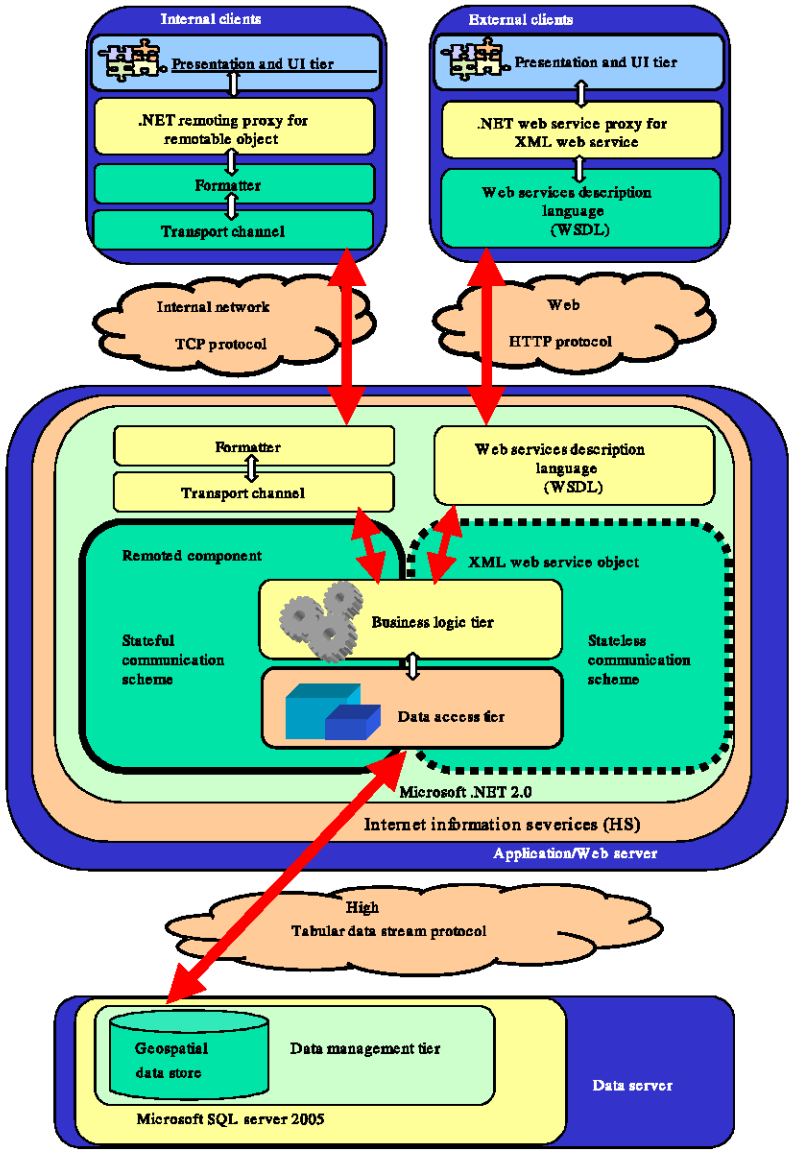

Fig. 5: Physical hybrid distributed architecture of the implemented WFS

using multi threading features of this kind of client ensures that client application remains responsive while transmitting requests and responses. .NET Remoting was used in communication among WFS server components and internal software system. At the other hand, Web services infrastructure was utilized in all interactions between third-party client side application and WFS server.

In client side application, responses to WFS-specific operations (client's requests) specify which feature types and attributes can be requested (after getting capabilities document and GML schema of interested feature types of WFS server via GetCapabilities and DescribeFeatureType operations, respectively). Then CQL statements are created by client using various logical and comparison operators which are provided as a part of user interface. The created CQL then is sent to the WFS server and the requested geospatial data is sent back to client side application (Fig. 6).

In order to allow user to visualize the GML data (returned from WFS server), a basic GML viewer was developed as part of client side application (Fig. 7). The GML viewer just represents the retrieved GML data and provides basic functionalities such as Zoom, Pan and Identify as well as saving data as GML 3.1 and SVG formats. In addition to the developed client side

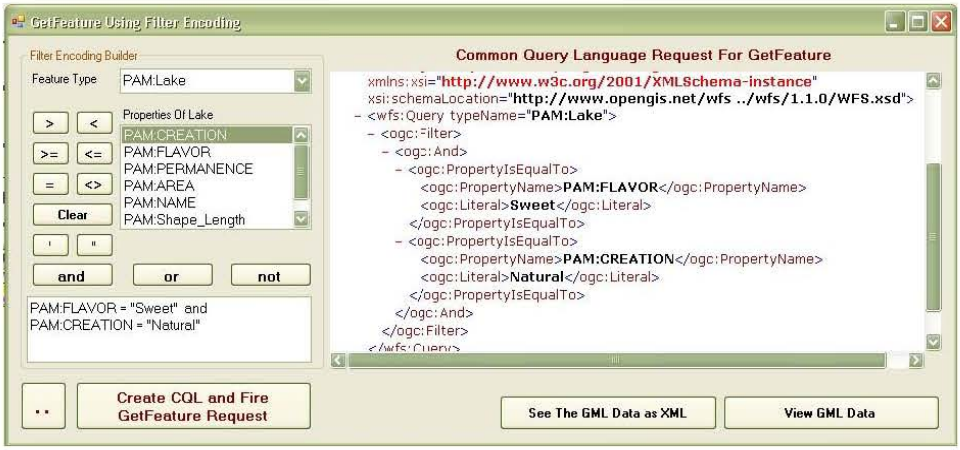

Fig. 6: User interface for introducing CQL statements and retrieve GML data from WFS server

Fig. 7: GML Viewer which developed using GDI+, as part of client side application that shows railroads, provinces and Lakes of I.R. Iran

application, any Web browser or application can consume the implemented WFS using WFS specifications.

In the developed system, objects and components in business logic and data access tiers work together to prepare an appropriate response messages. More accurately, user supplied parameters are parsed by business objects to determine which methods have to be executed on the remote component. In the case of GetFeature operation, user supplied CQL statements are translated to appropriate XQuery statements. Then XQuery statements are delivered to objects and components in data access tier to be sent to SQL Server 2005 DBMS.

In the last tier of the architecture, geospatial data were stored as GML 3.1 in the back end database. For retrieving geospatial data, XQuery statements which were sent by data access components are executed and result are sent back to the data access component. Data access components dispatch retrieved geospatial data to business logic components. Afterwards, geospatial data

Table 1: Performance evaluation for three configuration of .NET Remoting and XML Web service for transmitting 500KB, 1MB and 5MB geospatial data

	.NET Remoting			
XML Web service	First configuration	Second configuration	Third configuration	Distributed technology
ASP.NET runtime[1]	Windows application	IIS	Windows application	HOST
	SOAP	Binary	Binary	Formatter
	HTTP	HTTP	TCP	Channel
Average times in milliseconds for 2000 calls				Size of geospatial data transferred using WFS589
	127	25	12	500 KB
1445	344	72	34	1 MB
5487	1297	216	145	5 MB

[1]:ASP.NET runtime which is a core part of .NET platform is in charge of hosting components and serializing and deserializing Web service requests and responses

are prepared to be valid against WFS specifications. Finally, prepared GML data are sent back to the client using Web services infrastructure (using SOAP) for third-party consumers or binary protocol of .NET Remoting for internal clients.

In implemented physical architecture of WFS, two server machines were employed. Data access and business logic components resided on the Web/Application server machine (which utilized IIS as Web server application and component host for providing WFS functionality through .NET Remoting and XML Web service) and SQL Server 2005 installed on data server machine. Web/Application server and data server machines connected through a high speed Local Area Network (LAN). As a result, communication between data access components and DBMS were performed using well defined and DBMS specific binary protocol (In the case of SQL Server 2005 this protocol is called Tabular Data Stream) to optimize the performance. Also internal and third party clients are installed on two separate client machines which are connected to Web/Application server through a LAN (Fig. 5).

In order to evaluate the performance of the hybrid architecture, four GetFeature Operations were performed by internal and third-party clients. Geospatial data which were requested by these operations occupied nearly 500 KB, 1 MB and 5 MB of disk space in plain GML 3.1 format. The average time computed for 2000 GetFeature operations (request and response) are shown in Table 1 (times are in milliseconds). Using various channels and formatters, three configurations of .NET Remoting were used and the mentioned evaluation was performed for each configuration (Table 1).

From the results, it can be observed that the time taken for transferring geospatial data was the least while using TCP channel and Binary formatter (Table 1).

The average time for transferring geospatial data using Web services found high. In addition to using verbose SOAP messages and textual transfer protocol (HTTP), this could be the result of another issue. Stateless nature of Web service infrastructure imposes using single call objects. The single call objects are automatically created with every method call invocation (every request) and live for only the duration of the execution time of that method. With .NET Remoting it is possible (as we were utilized this approach) to make use of singleton objects. This type of object retains state and is shared between all clients. No matter how many clients are connected and referenced the remote component, there is only one remote object instance. Since there is no need for retaining state between requests and responses of WFS, singleton object (which was used in all .NET Remoting configurations of this research) performs faster; because there is no need to create the remote object for each new client request and terminate it as it finishes its task.

As an additional point, as the size of transferred geospatial data increased, performance of Web service degrades considerably (Table 1).

CONCLUSION AND FUTURE WORKS

In this study the design and implementation of a distributed geospatial Web service (basic WFS) using a combination of .NET Remoting and Web Services technologies based on the proposed hybrid architecture was illustrated. Following items describe outcomes of this research:

Since Web services technologies are the foundation of direct and open application-to-application communication, functionality of the implemented WFS can be simply added to any geospatial or non-geospatial software systems which are running on heterogeneous platforms. In other words, developing geospatial Web service using Web services technologies provide interoperability among geospatial and non-geospatial processing systems. Any type of client on any software platform which can send HTTP requests and parse the retrieved XML document (GML data) can use the implemented WFS.

Logical designing of WFS using four tiers logical architecture, results in a software system which is flexible to be implemented in various physical architectures. So the WFS can be configured into an appropriate physical architecture that will depend on our performance, scalability, fault-tolerance and security requirements.

As the .NET framework-specific distributed technology, .NET Remoting is not designed to provide interoperability or crossing trust boundaries to third-party clients. At the other hand, .NET Remoting provides faster communication speed over internal networks. As a result in some situation (such as access to geospatial data from internal network of a national mapping agency) it is a natural choice to make use of compact binary protocols of .NET Remoting to achieve the fastest possible communication speed.

Using the proposed hybrid architecture ensures achieving interoperability in heterogeneous environments while providing efficient way of exchanging geospatial data in internal and homogenous networks. In this case using TCP channel and Binary formatter provides the highest communication speed and Web services offers lowest one. As the performed evaluation indicates, using HTTP channel and SOAP formatter in .NET remoting, provides faster communication speed over Web services but this configuration of .NET Remoting cannot provide cross-platform communication. At the other hand using .NET Remoting over Web (in .NET-to-.NET applications communication) is only achievable through use of HTTP channel and SOAP formatter.

By considering high volume of geospatial data and faster communication speed provided by .NET Remoting, modifying geospatial data through the use of transactional WFS can be efficiently performed if .NET Remoting is used rather than HTTP-based solutions. Also based on the evaluation in this research, as the size of geospatial data increased, the communication speed of Web services (or in general, HTTP-based solutions) degrades significantly.

Based on our evaluations and practical tests, the implemented WFS using hybrid architecture proved to be an efficient solution for development of geospatial Web services. Integrating pure distributed component technologies (.NET Remoting) with Web services technologies is new in GIS world. Therefore more research and experiments are needed to test these technologies. In addition to underlying software technologies, geospatial data is rich in data types, huge in data volume and complex in semantics. So dynamic discovery, orchestration and invocation of geospatial Web services has their own challenges which should be considered carefully. Design and development of other geospatial Web services using Hybrid Architecture and improvement of the developed WFS by considering the unique characteristics of geospatial data will be the future works of the authors.

REFERENCES

Amirian, P., 2006. Design and development of a distributed geospatial Web services using XML and .NET technologies. M.Sc. Thesis, K.N. Toosi University of Technology, Tehran, Iran.

Booth, D., H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris and D. Orchard, 2004. Web Services Architecture, W3C Working Group. http://www.w3.org/TR/ws-arch (accessed 19-05-2007). eXtensible Markup Language, World Wide Web Consortium, XML: eXtensible Mark-up Language. http://www.w3.org/xml, (2000).

Fallside, D.C. and P. Walmsley, 2004. XML Schema Part 0: Primer 2nd Edn. W3C Recommendation. http://www.w3.org/TR/xmlschema-0 (accessed 05-05-2007).

Gailey, J.H., 2004. Understanding Web Services Specifications and the WSE. Washington, USA, Microsoft Press.

Hariri, S. and M. Parashar, 2004. Tools and Environments for Parallel and Distributed Computing. John Wiley and Sons, Inc., New Jersey, USA.

Huhns, M. and M.P. Singh, 2005. Service-oriented computing: Key concepts and principles. IEEE J. Internet Comput., 9: 75-81.

Lake, R., D. Burggraf, M. Trinic and L. Rae, 2004. Geography Markup Language. John Wiley and Sons, Chichester, England.

Lhotka, R., 2006. Expert C No. 2005. Business Objects. 2nd Edn. A Press Publishing, California, USA.

Lowy, J., 2005. Programming WFC Services. O'Reilly Media Inc., Orielly, California, USA.

MacDonald, M., 2003. Peer To Peer with VB. NET. A Press Publishing, California, USA.

Marks, E. and M. Werrell, 2003. Executive's Guide to Web Services. John Wiley and Sons, Inc., New Jersey, USA.

Murakami, E., A.M. Saraiva, L.J. Ribeiro, C.E. Cugnasca, A.R. Hirakawa and P.L. Correa, 2007. An infrastructure for the development of distributed service-oriented information systems for precision agriculture. Comput. Elect. Agric., (CEA), 58 (1): 37-48.

Newcomer, E. and G. Lomow, 2005. Understanding SOA with Web Services. Addison Wesley Inc., Maryland, USA.

Open GIS Consortuim, 2003. The OpenGIS Reference Model. http://portal.opengeospatial. org/files, visited on October 2006.

Open GIS Consortium, 2005a. Geography Markup Language Specification 3.1 (2005a). http://portal. opengeospatial.org/files/?artifact_id=4700 (accessed 05-12-2006).

Opern GIS Consortium, 2005b. Open GIS Web Feature Service implementation specification 2.1. https:// portal.opengeospatial.org/files/ (accessed 05-12-2006).

Papazoglou, M.P. and W.J. Van Den Heuvel, 2005. Web services management: A survey. IEEE J. Internet Comput., 9 (6): 58-64.

Papazoglou, M.P. and W.J. Van Den Heuvel, 2006. Service-oriented design and development methodology. Int. J. Web Eng. Technol., (IJWET 2006), 2 (4): 412-442.

Rammer, I., 2005. Advanced .NET Remoting. 2nd Edn. A Press Publishing, California, USA.

Selly, D., A. Troelsen and T. Barnaby, 2006. Expert ASP .NET 2.0 Advanced Application Design. A Press Publishing California, USA.

Shalloway, A. and J. Trott, 2004. Design Patterns Explained A New Perspective on Object-Oriented Design. 2nd Edn. Addison Wesley Inc., Maryland, USA.

Stal, M., 2002. Web services: Beyond component-based computing. J. Commun. ACM., 45 (10): 71-76.

Tanenbaum, A. and M. Van Steen, 2002. Distributed Systems: Principles and Paradigms. Prentice Hall Inc., New Jersey, USA .

Vinoski, S., 2003. Integration with web services. IEEE. J. Internet Comput., 7 (6): 75-77.

Volter, M., M. Kricher and U. Zdun, 2005. Remoting Patterns: Fundamental of Enterprise, Internet and Real-time Distributed Object Middleware. John Wiley and Sons Inc., New Jersey, USA.

W3C, 2006. The World Wide Web Consortium. Web Services Activity Statement. http://www.w3.org/ 2002/ws/Activity (accessed 05-05-2007).

Wang, A. and K. Qian, 2005. Component Oriented Programming. John Wiley and Sons Inc., New Jersey, USA.

Worboys, M. and M. Duckham, 2004. GIS a Computing Perspective. CRC Press, Florida, USA.