AGENT TECHNOLOGY AS A SOLUTION FOR NETWORK-ENABLED GIS

Saeid M. Kalantari [a] , Ali A. Alesheikh [b]
[a] Graduate student of master, Dept. of GIS Eng. sm_kalantary@yahoo.com
[b] Assistant Professor, Dept. of GIS Eng. alesheikh@kntu.ac.ir
[a,b] Faculty of Geodesy and Geomatics Eng., K.N. Toosi University of Technology, Vali_asr St, Tehran, Iran, P.C. 1996715433
Tel: +98 21 8789357, Fax: +98 21 877 9476

KEY WORDS: GIS, Mobile, Object, Distributed, Networks, Artificial_Intelligence

ABSTRACT:

With widespread availability of World Wide Web (www) and the acceptance of the intranet in various organizations, a new generation of GIS is burgeoning as Distributed GIS (DGIS). DGIS has open architecture distributed computing and high level mobility in mobile application and services. The Intranet involves in integrating a number of functions that are distributed physically or logically. Such distribution makes many challenges like inconsistent information, multiple information sources, decentralized control, conditional operation with failure and many other problems.
To interact with such systems and tackling mentioned problems in DGIS, new paradigm of software engineering and artificial intelligence combination has been introduced as agent technology. This paper introduces a new solution for distributed GIS using mobile agent technology. The concepts have been implemented in a case study and the results of test are evaluated scientifically.

1. INTRODUCTION

The network-based GIS introduced a new paradigm that enabled a broader variety of GIS applications and expanded GIS users. The new and innovative architecture delivered GIS for wide distribution across enterprises and organizations and made GIS accessible over the ubiquitous network.

1.1 Network-based GIS architectures

The development of network technology facilitated the applications of network-based GIS in two network levels: the Enterprise level and the Internet level. An enterprise network might be either the Local Area Network (LAN) or the Wide Area Network (WAN) and the Internet is also referred to as the World Wide Web (WWW). More specifically, GIS software exists in the following forms of network:

1.1.1 Host-Terminal network: This is the old model of networking in which a mainframe computer acts as the host and many terminals are used to access the data and GIS functions. Since every computation is calculated in the host and the terminals are only used for display and interactions, this model has very high performance requirements to the host. The major problem of this model is its slow response speed, high cost, and difficulty of development. Current networked GIS discarded this model (S. M. Kalantari, 2004).

1.1.2 Client-Server network: This model of network is widely exist within enterprises, in which some computers act as servers as well as others act as clients. The server computers usually have more power than the client and manage the centralized resources (figure 1). Different from the old host-terminal model, the client machines in this model also have some resources and computational power that might be used to relieve the load of servers. This characteristic of client-server network made it faster, more flexible and less costly than the host-terminal network. Actually, the client/server network is the major form of network in the enterprises (Yuan S., 2000).

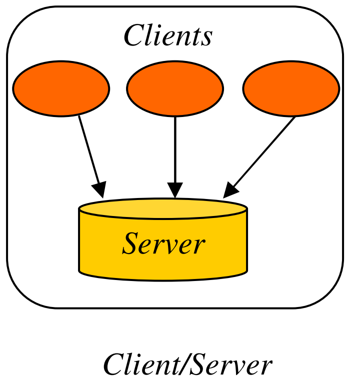


Figure 1: Client/Server architecture

1.1.3 Distributed architecture: In the third (Distributed) architecture (figure 2), GIServices are built upon a more advanced networking scheme. The significant difference is the adoption of distributed component technology, which can interact with heterogeneous systems without the constraints of traditional client/server relationships (McCarty B,1999). Under a distributed architecture, there is no difference between a client and a server. Every GIS node can act as a client or a server based on the task. A client is simply defined as the requester of a service. A server (likewise) is simply the machine that provides the service. This architecture permits dynamic linkages between data and software. In fact, the architecture is very similar to what is called "peer-to-peer" computing (P2P) that permits personal computers or workstations to communicate directly with one another, without a server (McCarty B,1999). The difference between P2P computing and the distributed architecture is that P2P can allow only one-to-one or one-to-many communication. A fully distributed GIService architecture has to permit many-to-many communications among computers; and here lies a current technical barrier to be overcome.

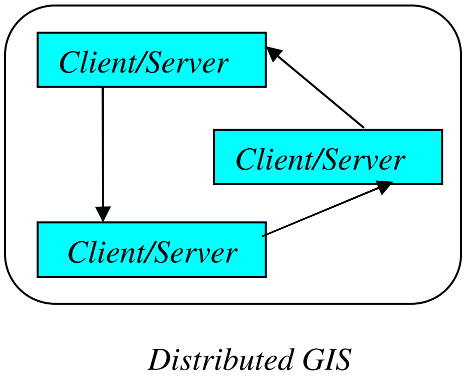


Figure 2: Distributed architecture

2. RESOURCE DISTRIBUTION POSSIBILITIES

According to the ideal conceptual model that support "geodata anywhere, geoprocessing anywhere", the geodata and geoprocessing functions could be distributed in different sites (Yuan S., 2000).The following gives the possibilities of distribution (P) of the geodata and geoprocessing tools in one location, no matter it is at a client site or at a server site. Let D represent geodata, and F indicate geoprocessing tools (Functions).

Using the following notations:

D: of geodata,

F: geoprocessing tools,

null: none of geodata or geoprocessing tools.

Then, we have:

$D = \{D, null\}$ and $F = \{F, null\}$

$P = D \cdot F = \{ (D , F) ,(null, F) , (D , null) ,(null, null) \}$ (1)

Which:

(null, null) Only exist at the client side, which means that neither geodata nor geoprocessing tools exist at the client side. The client in the typical client/server GIS model fit this case.

(D , F) Typically exist at the server side in an typical client/server GIS architecture, which means that all geodata and geoprocessing functions are provided by the server. Since the server has everything (see the (null, null) case). Standalone GIS systems also fit this case.

(null, F) The site has all geoprocessing components but none of geodata. The geodata are separated from the geoprocessing and distributed in other places.

(D , null) This may probably exist as a data center. Geoprocessing may be provided by other sites.

3. EXISTING SOLUTIONS

Based on the mentioned possibilities in distribution of resources and components, there are various ways which can be considered. Some of them have been implemented in various industries and businesses.

Due to previos section the (null,null) and (D,F) possibilities are equivalent ,because (null,null) manner only exist on client side and (D,F) manner only exist on server side. But the others two manner may exist on client or server side.

Two network strategies can support dynamic configuration of distributed GIServices (Tsou, M. and Buttenfield, B.P.,2003). The first strategy is object migration, where data or programs move from one node to another. The transferred data may or may not be deleted on task completion. In the second strategy, remote connection establishes a communication channel between two nodes, allowing data and services to be shared across the channel. Either can be applied to distribute geodata objects or GIS software components. Although the network technologies used in object migration and remote connection are quite different, the goal is the same, that is, utilizing available computing resources across networks. To distribute geodata objects dynamically (Figure 3), the objects can be shared or copied to a requesting node. To share objects, the network strategy used is remote connection to establish a link between distributed databases. The connection is established using SQL through Application Programming Interfaces (APIs), such as JDBC, ODBC or OLE DB. The second approach (object migration) utilizes an FTP protocol to actually move the data object and save it on the requesting node's local disk. Data migration may require both automated and manual procedures for download and format conversion. Similarly, either network strategy can distribute GIS software components dynamically (Figure 4). GIS operators may be invoked remotely, using Remote Procedure Calls. Other protocols are also possible, such as Internet Inter-ORB Protocol (IIOP) or Simple Object Access Protocol (SOAP). Technology frameworks that currently support this approach include DCOM, CORBA and Java RMI. It works as follows. A GIS application sends a request to local component services (that is, a service directory local to the node). The local service will use its client stub to build a connection with another node's (server) skeleton. The nodes temporarily adopt client and server roles to accomplish the connection. The server-side completes services then launch the required GIS component. This scenario assumes that the service is not locally available, and that the service directory can point to another node where the service actually resides.

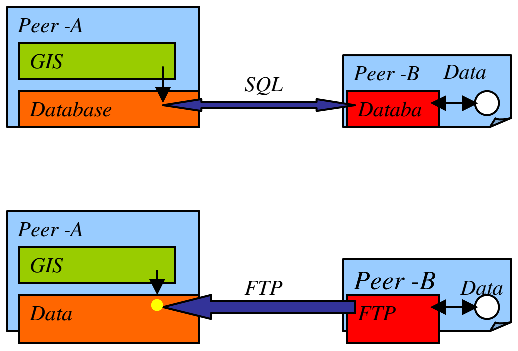


Figure 3: Two types of network connection for GIS software components

The second network strategy actually moves the software modules (the services) from one node to another. The migration process uses a HTTP protocol to transfer the required GIS procedure dynamically into the targeted GIS application. The downloaded GIS component is stored inside a container, which binds with the local GIS application. This approach is currently supported by several technology frameworks, such as Java applets in a Virtual Machine, or Active X containers. It has not been widely integrated into GIS environments as yet.
In the first strategy it is hard to move geodata through the network .Various challenges may occur in this process. Geodata are valuable resources in decision making so security issues is influenced in this strategy. Another important problem is network loading which is failed in this way. Uploading and downloading high volume geodata in a network fail the network traffic control. Uploading geodata across the network and then downloading it is done bye applets or FTP protocol. In each manner we faced with thick client strategy. Recent development in object oriented programming make it possible to produce software components and send them to the client before the running in the client machine, such as Java classes, ActiveX components and plug-ins. This comes out the thick client GIS. Java applet, Netscape plug-ins and Microsoft ActiveX component technology were involved in the structure. In the thick client architecture, the client machine does most processing work locally.

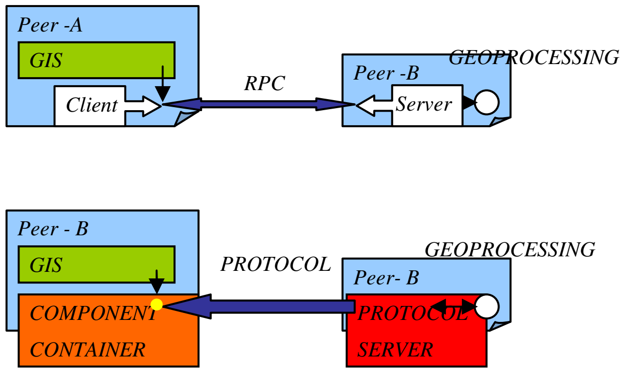


Figure 4: Two types of network connection for Geoprocessing

By combining RPC and sharing geodata objects, thin client strategy is accessible. In the thin client strategy the client only have user interface to communicate with the server and display the results.
In this manner several interactions between client and server is inevitable especially in the case of GIS processing because of high memory load and CPU usage, increasing in the number of the interactions may cause heavy traffic in the network.
Both thin and thick client systems have some advantages and drawbacks, but they are not the best solution in terms of taking advantage of network resources.
Combination between shared data and movable software opens a new paradigm in distributed GIS.
As it is mentioned in the previous section geoprocessing part of distributed GIS is weak and this solution may solve many problems such as security, efficiency and network load.
Such movable software across the network are entitled as agent. Agent technology is a new paradigm in computer science that next section will discuss about it widely.

4. AGENT TECHNOLOGY

Agent concept began by an idea of non-human agencies. It's is realized by constructing robots. Nowadays it is implemented as soft robot, living and doing its task within computer world.
An agent is a computer system suited in some environment and that is capable of autonomous action in this environment in order to meets its design objectives. There are several characteristics for an agent, some of them are ideal characteristics and are far from reality. But some characteristics like mobility, communication ability, reactivity and inferential capability can enhance GIS applications in various fields.

4.1 Mobile agent

Mobile agents are a class of agents whose predominant feature is the ability to transport between nodes on a network or between nodes across networks. They are the basis upon which true distributed information management agents can be built.
Mobile (or transportable) agents are a direct extension of the client/server technology. In the client/server paradigm, communicating entities have fixed and well-defined roles; a server offers a set of services and a client makes use of those services. This model also implies a strict sense of dependency; clients are dependent upon servers to provide the services that they require. The communication mechanism that takes place between a client and a server is through a message passing protocol since network communication is assumed. However, message passing has been criticized as being too low level, requiring programmers to determine network addresses and synchronization points themselves.
However, a fundamental problem exists with client/server architectures when considering distributed information system. If the server does not provide the exact service that the client requires, for example the server only provides low-level services, and then the client must make a series of remote calls to obtain the end service that it requires. This may result in an overall latency increase and in intermediate information being transmitted across the network which is wasteful and inefficient, especially for high volume of spatial data. Moreover, if servers attempt to address this problem by introducing more specialized services, then, as the number of clients grow, the amount of services required per server becomes unfeasible to support.
The mobile agent paradigm attempts to address the issues that are raised by the client/server and paradigms. Typical characteristics of mobile agents are their ability to migrate at will, autonomy in their actions, a peer-to-peer personality and a processing and network independence from their original location.
Mobility is a desirable characteristic in agents for a number of reasons:

4.1.1 Efficiency: If an agent can move across networks to the location where resources reside, then network traffic can be reduced since the agent can preprocess data and decide which the most important information to transfer is. This is a crucial aspect when considering users who connect through a low bandwidth link.

4.1.2 Persistence: Once a mobile agent is launched, it should not be reliant on the system that launched it and should not be affected if that node fails. The concept of an agent moving between network nodes gives it the ability to `survive' and to reach as many resources as possible. This is useful for mobile computer users due to the fact that they can log on, launch an agent, log off and check later on its progress.

4.1.3 Peer-to-peer communication: A failure of the client/server paradigm is the inability of servers to communicate. Mobile agents are considered to be peer entities and, as such, can adopt whichever stance is most appropriate to their current needs. For example, when a mobile agent is interrogating a resource it takes the role of a client. However, when another mobile agent wishes to query it, then it becomes a server. This allows for great flexibility in dealing with network entities and distributed resources(Buehler, Kurt and McKee, Lance, 1998).

5. DESIGNING AGENT-BASED GISERVICES USING UML

Designing software based on UML specification includes several steps. First of all, system analysis and problem definition must be respected. Our specification system is a prototype for integrating simple GIS software in a mobile agent framework. The problem, which thesis is challenged with, is to transform typical GIS software as an Agent for moving to remote address.

5.1 Analysis

Object-Oriented analysis emphasize on finding and describing the objects or concepts. Our system is a simple prototype so it does not need to an extended analysis. We need an interface to invoke system, GIS system as an object, and finally a mobile object for carrying GIS object (figure 5).

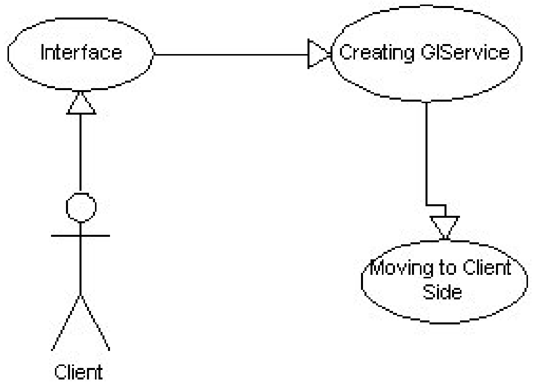


Figure 5: Usecase diagram of the system

5.2 Finding objects, variables and methods

After usecase diagram (figure 5) and identification of scenarios, another fundamental step is defining of class for the class diagram. Classes are created using their instance (objects) and their variables and methods. For receiving such goal the simple way is to consider name and verb in system functionality description
Using founded objects and their methods and variables now classes can be created for each object. By this way DataLoad class (as a interface), MapDisp class, TelAgent class and Run Class are composed.

5.3 Class diagram

On of the important aspect of object-oriented system is placing objects and classes in a hierarchical form. There are various relations in UML that programming languages like java have not special method for visualizing it. In our system we have two relations. First association which is simple relation between our own created classes and also aggregation relation which our own classes have this relation with imported classes (figure 6).

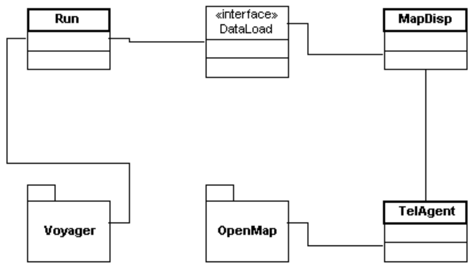


Figure 6: Class diagram of the system

5.4 Description of behavior

Now we can describe the scenarios relative to the use-case diagram using sequence diagram .by this way behavior of agent can be described (figure 7).

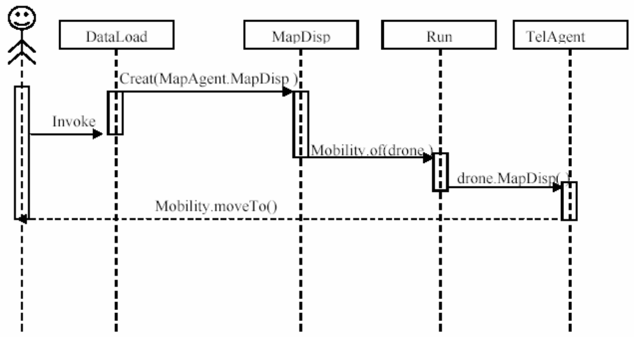
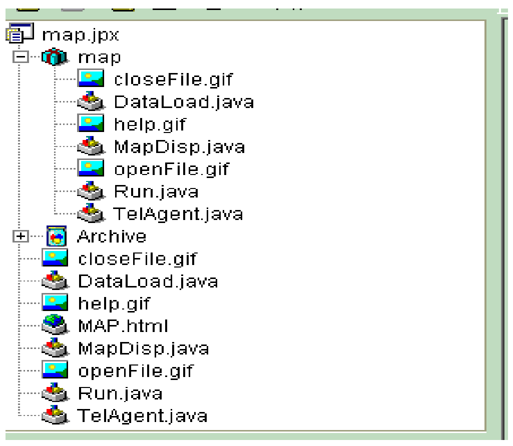


Figure 7: Sequential diagram of the system

6. DEVELOPING PROTOTYPE

The application has been developed in Jbuilder environment. Jbuilder is a visual environment for debugging JDK applications .for Java application we used JDK 1.3.0_02. Jbuilder facilities make code debugging simpler than JDK own environment (figure 8).

As it is described in class diagram we should use two packages for supporting mobility and GIS applications .we used voyager as a package for supporting distributed object technology and openmap for supporting GIS functionality.



```
map.jpx
  map
    closeFile.gif
    DataLoad.java
    help.gif
    MapDisp.java
    openFile.gif
    Run.java
    TelAgent.java
  Archive
  closeFile.gif
  DataLoad.java
  help.gif
  MAP.html
  MapDisp.java
  openFile.gif
  Run.java
  TelAgent.java
```

Figure 8: Java base Map packages

After invoking the client, as it is described in sequential diagram TelAgent will run on client machine as GIS software

7. CONCLUSIONS

We have described the requirement of Distributed GIS application in a network environment and presented a solution based upon an agent metaphor. We have characterized the agents and introduced mobile agent in distributed environment. Future description of this system can be seen in two points of view. Firstly as telecommunication view which system can be develop for wireless application of GIS like location-based services. In another view we add intelligent characteristics of agent to mobile agent system for decreasing client work within distributed GIServices.

8. REFRENCES

Buehler, Kurt and McKee, Lance, 1998, The OpenGIS Guide, Introduction to Interoperable Geoprocessing and the OpenGIS Specification, third edition, Open GIS Consortium Technical Committee

McCarty B.,(1999) Java distributed objects ,Techmedia publication, India

Tsou, M. and Buttenfield, B.P. (2003) An Agent-based, Dynamic Architechture Distributed Geographic Information Services.

S. M. Kalantari, (2004), developing a Distributed Geoprocessing Service model, M.Sc. Thesis, K.N.Toosi University of Technology, Iran.