

MapAsia 2003

Promoting Distributed GIServices Using Mobile Agent Technology

Authors:

Saeid M. Kalantari (Contact Person)

M.Sc. Student, Dept. of GIS Eng.

Email: sm_kalantary@yahoo.com

Ali A. Alesheikh (Presenting Person)

Assistant Professor, Dept. of GIS Eng.

Email: alesheikh@kntu.ac.ir



Ali A. Alesheikh is assistant professor at the K.N.Toosi University of Technology, Tehran, Iran. Dr. Alesheikh got his Ph.D. in Geomatics Engineering from the University of Calgary, AB Canada in 1998, specializing in Geospatial Information Systems. In 1993, he got his Masters in Digital Photogrammetry from the University of New Brunswick, N.B. Canada. The areas of his interests are: Online GIS, Location Based Services, Uncertainty in GIS, and visualization.

Ahmad Talebzadeh

Application and GIS director of Iranian Remote Sensing Center

Email: talebzadeh_a@yahoo.com

Address: Faculty of Geodesy and Geomatics Eng., K.N. Toosi University of Technology
Vali Asr St., Vanak Sq., Tehran, Iran,
Post Box: 15875-15433
Fax: +98 21 878 6213
Tel: +98 21 877 9473-5

Promoting Distributed GIServices Using Mobile Agent Technology

Saeid M. Kalantari

M.Sc. Student, Dept. of GIS Eng.
Email: sm_kalantary@yahoo.com

Ali A. Alesheikh

Assistant Professor, Dept. of GIS Eng.
Email: alesheikh@kntu.ac.ir

Ahmad Talebzadeh

Application and GIS director of Iranian Remote Sensing Center
Email: talebzadeh_a@yahoo.com

Address: Faculty of Geodesy and Geomatics Eng., K.N. Toosi University of Technology
Vali Asr St., Vanak Sq., Tehran, Iran,
Post Box: 15875-15433
Fax: +98 21 878 6213
Tel: +98 21 877 9473-5

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. Distributed Systems

A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages. This definition leads

to the following characteristics of distributed systems: concurrency of components, and independent failures of components. Three examples of distributed systems are given:

- The Internet;
- An intranet, which is a portion of the Internet managed by an organization;
- Mobile and ubiquitous computing.

The sharing of resources is a main motivation for constructing distributed systems. Resources may be managed by servers and accessed by clients, or they may be encapsulated as objects and accessed by other client objects. The Web is discussed as an example of resource sharing.

Networks of computers are everywhere. The Internet is one, as are the many networks of which it is composed. Mobile phone networks, corporate networks, factory networks, campus networks, home networks, in-car networks, all of these, both separately and in combination, share the essential characteristics that make them relevant subjects for study under the heading of distributed systems. The characteristics of networked computers that impact system designers and implementers are discussed. The main concepts and techniques that have been developed to help in the tasks of designing and implementing systems are elaborated in this paper.

A distributed system is defined as one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages [1]. This simple definition covers the entire range of systems in which networked computers can usefully be deployed.

Computers that are connected by a network may be spatially separated by any distance. They may be on separate continents, in the same building or the same room.

There are various challenges in these systems that must be respected like heterogeneity, openness, security, scalability, failure handling, concurrency, transparency. Here are some of the problems that the designers of distributed systems face:

- Widely varying modes of use: The component parts of systems are subject to wide variations in workload - for example, some web pages are accessed several million times a day. Some parts of a system may be disconnected, or poorly connected some of the time - for example when mobile computers are included in a system. Some applications have special requirements for high communication bandwidth and low latency - for example, shortest path.
- Wide range of system environments: A distributed system must accommodate heterogeneous hardware, operating systems and networks. The networks may differ widely in performance - wireless networks operate at a fraction of the speed of local networks. Systems of widely differing scales - ranging from tens of computers to millions of computers - must be supported.
- Internal problems: Non-synchronized clocks, conflicting data updates, many modes of hardware and software failure involving the individual components of a system.
- External threats: Attacks on data integrity and secrecy, denial of service[1]

Heterogeneity of distributed systems is an outstanding challenge among the others. All parts of distributed system like platform, data and application challenge with this problem. For solving platform problem the virtual machine approach provides a way of making code executable on any hardware: the compiler for a particular language generates code for a virtual machine instead of a particular hardware. For example, the Java compiler produces code for the Java virtual machine, which needs to be implemented once for each type of hardware to enable Java programs to run. However, the Java solution is not generally applicable to programs written in other languages. But, other problem like application and data heterogeneity must be considered in application developing platform.

2. Distributed GIS

Distributed GIS is a subset of distributed system so whole coordination and requirements of it must be considered. In addition because of special data type and processing tools in this kind of distributed system, other GIS specifications are necessary. The ideal model of distributed GIS is “Geodata anywhere and Geoprocessing anywhere” [2]. Geodata and Geoprocessing do not have to be in the same side of the network. Four important part of

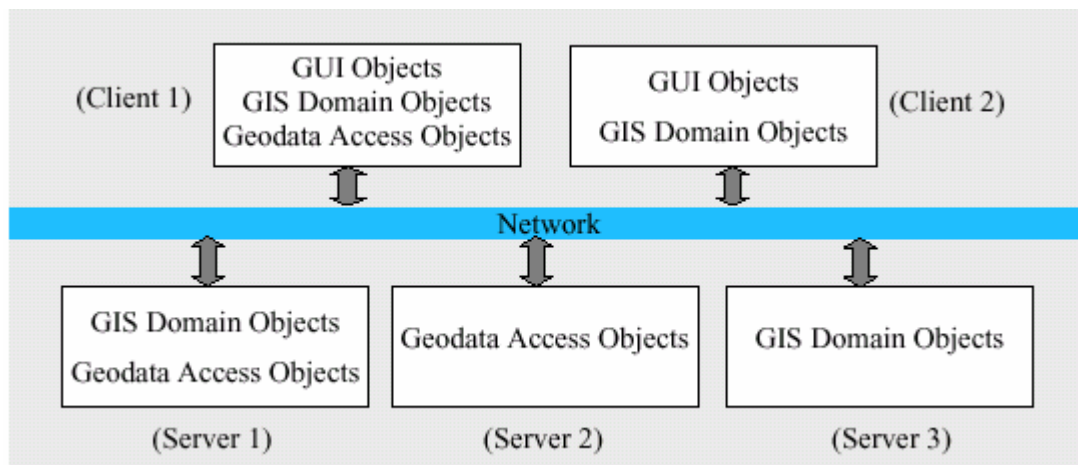


Figure 1: Geodata and geoprocessing distribution in the network [2].

conceptual architecture of distributed GIS are client, network, geodata and geoprocessing. Network is located in the centre of this architecture and other parts connect via network (see figure 1).

Almost all of current distributed GIS products are still only for geodata publishing purpose. Geoprocessing tool keeps as a weak point [2]. Although there are some distributed GIS focused on geoprocessing tools but all of them challenge with considerable problem. Most of them implement geoprocessing tools near the data.

Assuming the possibility of the geodata and geoprocessing tools in one location be presented by P, no matter it is at a client site or at a server site. If D represent geodata, and F indicate geoprocessing tools (Functions), we have

D: geodata, F: geoprocessing null: none of geodata or geoprocessing tools.

Then, we have:

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

So

$P = D \times F = \{(D, F), (F, \text{null}), (D, \text{null}), (\text{null}, \text{null})\}$

In 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. In this case, the client is a thin client.

(D, F): Typically exist at the server side in a typical client/server GIS architecture, which means that all geodata and geoprocessing functions are provided by the server. Since the server has everything, its clients should be thin clients (see the *(null, null)* case). Standalone GIS systems fit this case.

(F, null): 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.

Some products of geodata and geoprocessing are located in a same side in server (thin client). So when a client sends a request to server because of high CPU usage of geoprocessing tools and also high memory load of geodata (for example shortest path) other clients must be waiting until server finish the process. This causes high traffic in the network. In another was if geodata sent to geoprocessing side high volume of data must be uploaded in the network and finally we have high traffic in such networks. This kind of connection can be seen in applets. Although after downloading the client can interact with system and data easily and quickly but network traffic during the period of downloading is failed.

There is another manner in this kind of systems which sending geoprocessing tools to geodata side. But in this manner a geoprocessing tool must be ready and move it in a manner of run software through network to geodata side. This can be done by new paradigm of computer science entitled as agent technology.

3. 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”[3]. 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.

3.1 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

- *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.
- *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.
- *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[4].

4. Design of agent-based GIS

To design and develop an Agent-Based GIServices (AGIS) one should make a plan in a long run. Since the technology is developing so fast nowadays, adding new

functionalities to existing system should be possible and easy; system maintenance should be simple and fast as well. These can be achieved by software engineering and object-oriented technology. Since Unified Modeling Language (UML) provides a standard notation for modeling and design, UML must be used

4.1 What is object-oriented?

Barroca et al. (2000) describe object-oriented as follows: “A software system whose basic structuring is around things rather than around actions is said to be object-oriented”[5]. The reason is that although businesses change frequently, it has been observed that the things, which are manipulated by computer systems, stay quite stable, while the ways in which they are used change rapidly. This leads to structuring software around things that are manipulated instead of around actions that manipulate them. For example, we may develop one unit of software for spatial analysis, another unit of software for network analysis etc. No matter how application will change, these units are still needed; only the way of how they are coupled will change.

4.2 Distributed object technology

Distributed object computing extends an object-oriented programming system by allowing objects to be distributed across a heterogeneous network. Each of these distributed object components inter-operates as a unified whole. Currently, the three most popular industry standards for fulfilling distributed computing tasks are Microsoft's Distributed Component Object Model (DCOM), Object Management Group's (OMG's) Common Object Request Broker Architecture (CORBA) , Sun's Java Remote Method Invocation (Java RMI) and ObjectSpace Co. Voyager[6].

4.3 UML

In this paper, Unified Modeling Language (UML) is used for AGIS modeling and design. The reason for it is that UML dominates recently object-oriented techniques. UML is the result of the merging of several object-oriented methods. It defines a number of diagrams to describe a system, and what these diagrams mean.

4.3.1. Agent-Oriented Design Process with UML

We can divide the traditional software development process in two fundamental steps: the analysis phase in which system requirements are to be captured and the design phase where the identified system functionalities are implemented. In other words, while in the system analysis phase we focus on “what”, in the implementation phase we focus on “how”. Dealing with the design of agent based software, we use a slight different approach: the AODPU (Agent-Oriented Design Process with UML)[7]. In this approach there are three steps:

a) Identification of the agents.

A functional description of the system is provided through a hierarchical series of use-case diagrams. The first diagram (we could consider it as some kind of ‘context’ diagram) will only represent one usecase (the system), some actors in the environment and any external entity interacting with the system. Other use-case diagrams will give more details on the system. By representing agents as use cases, external entities and

environment as actors and interactions among agents with relationships we can fully describe our system from a functional, external point of view.

b) Definition of the agents' structure.

In the usecase diagrams some agents are identified and their roles are described. At this point of the process, a specification of the structure of each agent can be provided through a class diagram in which the methods of each class correspond to the subtasks that each agent is able to perform. Each agent can play his own role in the system organization using his own methods.

c) Description of the behaviors.

We can describe the scenarios relative to the usecase diagrams using some sequence diagrams: by this way we can also detail the agents' behavior taking into account the time variable that is one of the key factors in real-time services. At the end of this process all the requirements are fixed (for this iteration) and the implementation can start.

4.4 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 extended analysis. We need an interface to invoke the system, GIS system as an object and finally a mobile object for carrying GIS object(see figure 2).

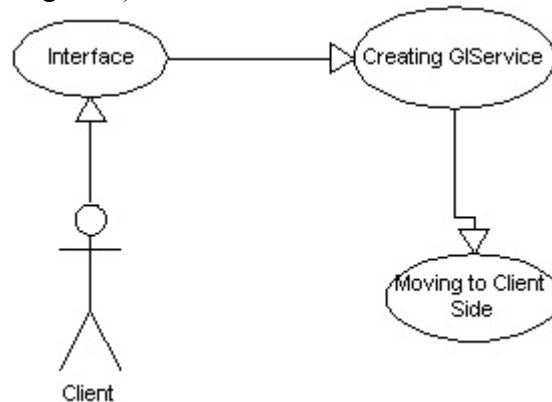


Figure 2: System usecase diagram

4.5. Finding objects, variables and methods and creating classes

After usecase diagram and identification of scenarios, another fundamental step is defining class for class diagram. Classes are created using their instance (objects) and their variable and methods. For receiving such goal the simple way is to consider the names and verbs in system functionality description. This way is recommended by many authors. Table 1 illustrated this analysis.

Table 1 : Main components of system

<i>Noun/Verb</i>	<i>Description</i>	<i>Class/Variable/Method</i>
System	System is that thing we are discussing about it	Map
Client	Client is person who wants to work with system	Client
GIS	GIS is a system that include GIS functionality	TelAgent
Agent	Agent is an object that carries some functionality in the network.	Run
Interface	It is responsible for serve system main functionality	DataLoad
MapDisp	It is frame that implement system functionality	MapDisp

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.

4.6 .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(see figure 3,5).

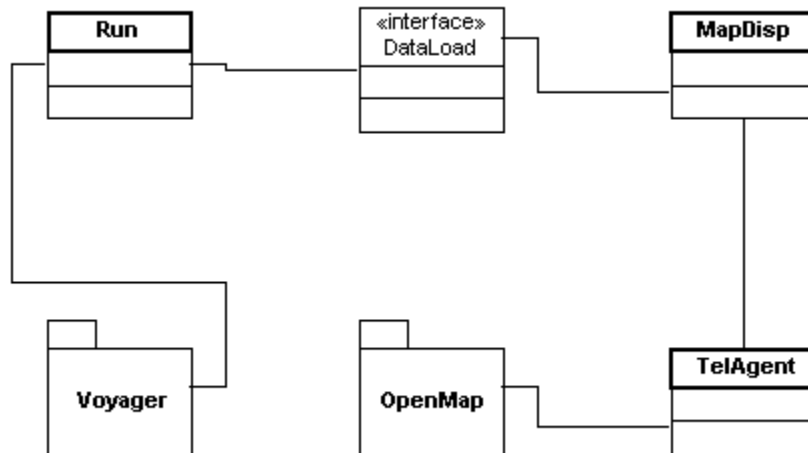


Figure 3: Map Class digrams

4.7. 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(see figure 4).

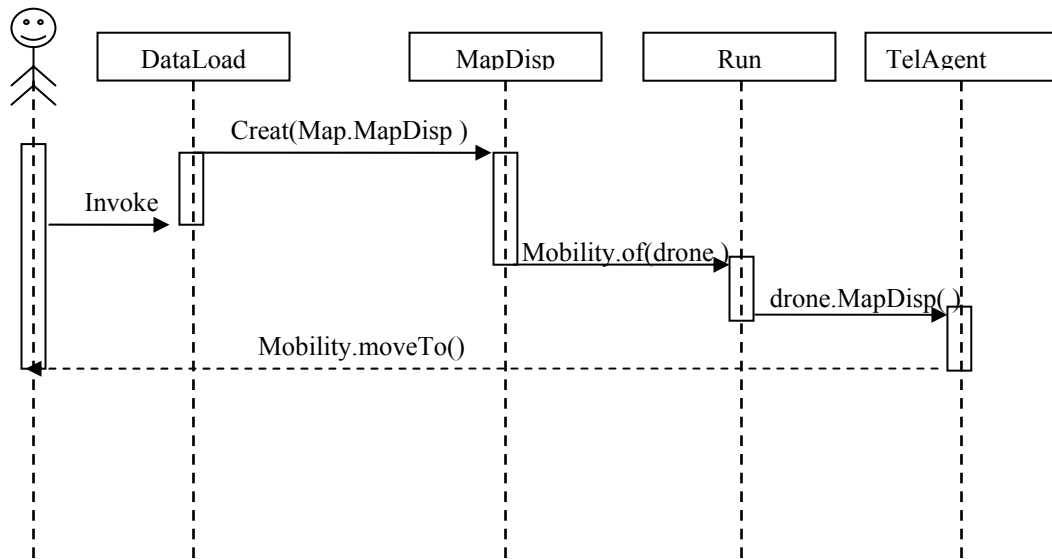


Figure 4: Map sequential diagram

6.5 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. 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.

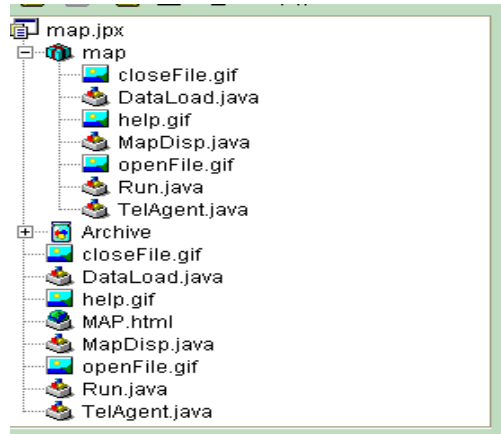


Figure 5: Map packages

After invoking the client, as it is described in sequential diagram TelAgent will run on client machine as GIS software(see figure 6).

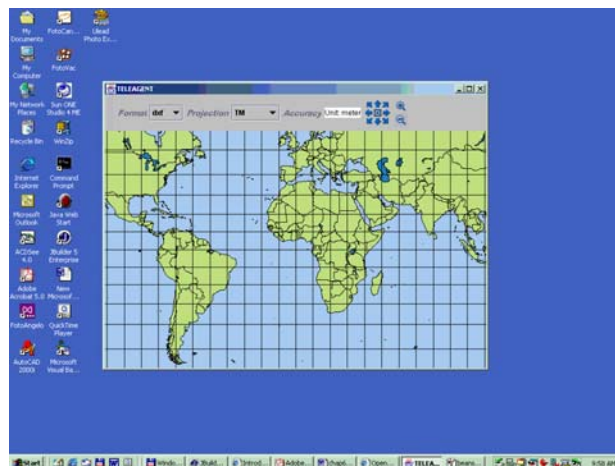


Figure 6: TelAgent after invoking, in client side

6.6 Conclusion

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.

References

- [1] Coulouris G., (2001), Distributed Systems, Addison Wesley ,England

- [2] Yuan S., (2000), Developing a Distributed Geoprocessing Service model, M.Sc. Thesis, University of Calgary, Canada

- [3] Jennings, N.R. and Wooldridge, M. (2001), Applications of intelligent agent, Queen Mary and Westfield college, University of London

- [4] Dale J.,(1997) ,A mobile agent architecture for distributed Information management, PhD thesis, University of Southampton

- [5] Zhao Y, (2002),Design and development of prototype Airport Noise Information System, MSc thesis, ITC, Netherland

- [6] McCarty B.,(1999) Java distributed objects ,Techmedia publication, India

- [7] Cossentino M., Designing agent-based systems with UML