Network-enabling GIS: issues, models and a review

Zhu Xu,
Department of Land Surveying and Geoinformatics
The Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong
Email: zhu.xu@polyu.edu.hk

Y.C. Lee
Department of Geodesy and Geomatics Engineering
University of New Brunswick
Fredericton, N.B. E3B 6Y1 Canada
Email: yclee@unb.ca
and
National Laboratory for Information Engineering in Surveying, Mapping and Remote Sensing, China (LIESMARS)
Wuhan University, Wuhan, China

ABSTRACT

Networking GIS tools and applications vary greatly in both function and architecture. In this paper, we will provide a framework to study the problems of networking GIS. GIS resources consist of geodata and geoprocessing capabilities. From the viewpoint of geodata sharing, it is observed that decoupling geodata and geoprocessing is necessary given the difficulty of integrating autonomous GIS systems, and geodata catalogue service can be an alternative to tightly-coupled multiple GIS systems in certain circumstances. For the data-processing coupled case, we first examine various distributing possibilities of deploying geodata and geoprocessing capability in network environments. Then issues involved the integration of geodata and geoprocessing in different distributed computing architectures are then discussed. For the client-server architecture of distributed GIS, there is the problem of data model mismatch between GIS tools and spatial database based on general-purpose DBMS, which lacks support in fundamental spatial modeling concepts such as a map. The possible result of this data model mismatch is analyzed. For federated architecture, it is argued that most interoperable GIS under research assumes read-only access mode. Some validation of this assumption and its consequences are made in this paper. It is also identified that tools for geodata schema integration and geodata integration are needed, and we will review the status of research in these issues.

1. Introduction

With the development of computer networks and distributed computing technology, many computer-based applications are turning from the traditional stand-alone mode to the contemporary networked mode. By networking, computing resources distributed over a computer network is connected and integrated to allow resource sharing and the better deployment of resources. GIS is no exception. We use the term "network-enabling GIS" to refer to the practice of making GIS work in the network computing mode, and use the term "networked GIS" to refer to a GIS that benefits from network computing.

Many kinds of networking GIS tools and many to-some-degree networked GIS applications, with different functionality and architecture, are emerging and more are under research and development. The situation is on one hand quite confusing in the terms of objective, concept, and terminology. Yet on the other hand it leads to a perception that networking GIS is just to piece together a number of programs. Although there have been some serious studies into specific issues, we still lack a framework to consider the problem as a whole and to coordinate the various studies. In our opinion, fully networking GIS, although desirable, is extremely difficult. It is the objective of this paper to propose a framework for studying the difficult problems of networking GIS by identifying issues, suggesting models serving different levels of objectives, and reviewing the status of this technology. We will take a database perspective in this study since GIS is considered as a special database problem.

2. Issues of Networking GIS

Generally, there are two approaches to networking GIS. One is to make components of one GIS (a complete application system) work as a whole system in a network environment, which corresponds to the top-down design approach. The other is to make a number of independent GIS collaborate in a network environment, which corresponds to the bottom-up design approach. Usually, in the first case, the owner has control over the system in terms of whether to distribute, what to distribute, how to distribute, and so on. In the second case distribution of computing resources and decentralization of control is more a fact to accept than a result of intentional design. As we will see later, these two cases can be quite similar in the sense that when several GIS decide to collaborate they would more or less behave as one GIS. Therefore, our discussion begins with the case of networking one GIS and is then extended to the case of networking several GIS.

2.1 Basic Issues

Before networking, the computing resources of a GIS need first to be divided into well-defined components or partitions, which then are distributed over the network. There are two ways we can view this distribution. When viewed from the level of operating system, the hardware resources are distributed, including the CPU, main memory, secondary memory, peripherals, and so on. If we view it from the perspective of a particular distributed application, geodata and geoprocessing (the processing software) are the distributed resources.

2.1.1 Dividing geodata

Before dividing data we should have performed data modeling leading to a global conceptual schema. In the bottom-up data integration case, integration of data schema is always a prerequisite for integrating data instances. The integrated schema could be recorded in the global directory of a distributed database or just exist in the mind of the application developer.

Dividing of data is called fragmentation, which deals with both data schemas and data instances. In relational databases, tables can be fragmented horizontally and vertically. For an object-oriented data model, horizontal and vertical fragmentation can also be considered. The instances of a class can be stored at multiple sites, which is the case of horizontal distribution. Also the properties of a class can be scattered over multiple sites, which is the case of vertical distribution. However, vertical fragmentation of a class is much more complicated than that of a table due to class hierarchies. A third kind of fragmentation applicable to objects is path fragmentation, which is the clustering of all objects forming a composite object into partitions. In practice, a mixture of these fragmentation methods is applied.

Let us suppose that geodata are stored in a database managed by a relational or an object DBMS. Vertical fragmentation of a geospatial feature could result in several geospatial features, each of which has at least one spatial property. Horizontal fragmentation of a geospatial feature will partition it into groups.

Geodata describes a continuous space and a spatial data model is more than a relational or an object oriented data model supporting geometric data types. For many applications, a *map* is an indispensable concept for geospatial modeling and geoprocessing just like *relation* in a relational model and *class* in an object-oriented model. A *Map* in the sense of data modeling is different from a relation or a class collection in at three aspects. First, a map has a boundary, meaning that an arbitrary horizontal fragmentation will result in the scattering of feature instances. If the map boundary is to be respected, geospatial features crossing a boundary will be segmented. Second, when topology is significant, as it is in many applications such as utility network management, a map assumes some degree of topological integrity. Third, a map when well defined in mathematical terms has an algebra that differs from the relation and object algebras.

If the *map* is a unit of dividing geodata, two basic forms of partitioning can be used: *geographic partitioning and thematic partitioning*. From a point-set topological perspective, a map is a closed orientable two-dimensional manifold, which assumes some degree of topological integrity. From a data model perspective, map is an important spatial modeling concept just like relation in the relation model and class (collection) in the object model. Geographic partitioning corresponds to horizontal fragmentation and thematic partitioning corresponds to vertical fragmentation. The result of these two partitioning processes is layered map sheets.

The current technical trend is to use industrial-strength DBMS to manage spatial data mainly for efficiency and to facilitate the integration of geodata and non-geospatial data. However, general-purpose DBMS does not support the *map* concept and its associated operations, such as map overlay or spatial join, and would ignore spatial data integrity constraints, such as topological integrity. These DBMS in general will support geospatial *feature table* or *feature class* but not *map*.

We will also distinguish data fragmentation from file fragmentation. The purpose of file fragmentation is to deal with large file sizes and it would not introduce data inconsistency from the viewpoint of application. Yet the purpose of data fragmentation is to distribute data so that different fragments are maintained at different sites, which has the potential to affect data integrity.

2.1.2 Dividing geoprocessing

The dividing of software functions into major pieces or modules is the study of software architecture. Similar to the dividing of data, there are two fundamental strategies: horizontal division and vertical division. Usually, horizontal division precedes vertical division. Layering (horizontal division) is a fundamental approach to deal with complexity in computing systems, in which layers represent different levels of abstraction are recognized. An example is the ISO/OSI seven-layer model for interconnecting computer networks. The functions of an application are also often divided into layers, which are often called tiers. When layers of client/server computing are physically distributed among independent computer systems of a computer network, we get the client/server architecture of distributed computing.

When the services of an application are replicated over the computer network, each participating system will need to cooperate with each, resulting in a vertical distribution or a peer-to-peer computing architecture. A system can be simultaneously vertically and horizontally fragmented. A typical case is the integration of existing autonomous database systems. This is the problem of multidatabase systems, where the DBMS of autonomous database systems have a peer-to-peer relationship with each other. In this case, we need to build a federated GIS or a multi-GIS system.

Within a layer, the processing function can further be divided horizontally into finer layers and vertically into modules. The modules are basically parallel and independent and do not talk to each other directly. Such a parallel relationship is different from the so-called peer-to-peer relationship, in which case programs having the same or similar functions are replicated at different sites and they cooperate to provide the functionality in a distributed environment. The relatively new three-tier client/server architecture is the result of further layering the client of two-tier architecture into user interface and application server that accommodates application logic. In principle, every layer can be replicated over the network, i.e. we can get a peer-to-peer architecture for every layer of distributed computing. However, it is on the one hand often not necessary to do so and on the other hand very difficult to make peers cooperate. To make peers cooperate, we often need to dynamically distribute tasks to peers and dynamically make them coordinate quite like the case of parallel processing. One example of peer-to-peer computing is a distributed DBMS, in which a global DBMS exists to coordinate component DBMS, managing global schema, global query processing, global transaction, and so on.

In the above discussions, we have considered only static allocation of processing, in which each process stay at a computer and responds to processing requests by returning resulting data. When the partitioning of functions become dynamic, other computing architecture emerges, such as mobile code and mobile agent. Presently, an alternative to "move data" is to "move code" instead. In some cases, it is more efficient to send a program that travels through different sites and do some processing at each site. This is the idea of mobile code. Java applet and ActiveX control can be used for this purpose. Today mobile code has been mostly used for enhancing user interaction within browsers but other applications are possible.

Layering of GIS has not been studied extensively. GIS is traditionally decomposed into components or sub-systems without paying much attention to the layering relation between them. The OGIS Service Architecture identifies dozens of GI-related services, many of which are for networking geodata instead of networking GIS and are not necessarily components of a GIS. We emphasize the layering and parallel relationship among components and suggest the following modular components of GIS:

a. Spatial database that stores and manages spatial data
b. Spatial application: spatial transformation, measurement framework conversion, basic geodata input, and so on.
c. Data presentation (map generalization) and user interface

It has been emphasized that the interface between layers be standardized to provide syntactic interoperability. GIS efficiency and effectiveness of integration depends on whether the GIS components share a common data model otherwise *data model mismatch* will happen. A solution to this data model mismatch problem is to use read-only spatial databases. We first make spatial data topologically clean before we load them into a spatial database. Then the spatial database provides read-only service to clients. Consider the case that the application needs a spatial data model that supports map algebra and combinatory topology while the spatial database supports only geometric data types like those defined in the OGIS Simple Feature Geodata model. When the application obtains the data over a study area from a spatial database, it is the responsibility of the application to build topology and perform very costly map-related transformations such as overlay. After performing spatial analysis, the results cannot be stored in the spatial database because it does not support the sophisticated spatial data model. That means that the result has to be stored locally, defying the purpose of data sharing.

2.1.3 Providing distribution transparency
When data and services are scattered over the computer network, it is desirable that such distribution is transparent to users. Providing transparency to users means that these physically distributed resources need to be logically combined or integrated so that to the users all resources seem to be local and they form a whole system instead of separated components. To provide data distribution transparency, data needs to be integrated; to provide services distribution, services need to be integrated, which is called application integration here to distinguish it from system integration at the level of distributed computing platform.

Data integration is a problem that also exists in non-networked environment. When data describing overlapping the universe of discourse need to be used together to serve an application, they need to be integrated. Here data integration refers to the process of integrating fragments of data about the real world so that the resulting data (set) gives a more comprehensive description of the real world. Data integration is highly related to semantic data sharing, since the major impediment to data integration is currently the semantic heterogeneity of data between source context and the target application context.

The other problem is application integration and in our case GIS application integration. The current technical trend is component-based software development, which means developing application by assembling interoperable components, particularly designed software modules facilitating assembling or plug-and-play. It is import for these components to fit well in a pre-defined framework with standard interfaces. For GIS, the OGIS is an industry-wide effort of such standardization, whose aim is for components from different vendors to be interoperable. However, in our opinion, GIS integration is more than assembling components. It also includes the practice if federating independent peer-to-peer GIS systems, which is much similar to database integration. When a top-down design approach is taken, the problem of sub-system integration becomes relative easy with the sacrifice of autonomy. On the contrary, the integration of existing autonomic systems can raise much technical difficulty and usually full integration transparency cannot be achieved without compromising autonomy.

2.2. Complicating factors

2.2.1 Autonomy
Autonomy refers to the freedom of a participating system in making decisions on all aspects of the system, such as selection of platform and software, the design and implementation of application, and so on. Autonomy can cause problems in integrating independent participating systems. We argue that the unit of autonomy should be system and not component. Components within a system have no autonomy and they must be designed and implemented according to a pre-defined framework.

2.2.2 heterogeneities
Heterogeneity refers to the differences existing in all levels of participating systems. There is platform related heterogeneity (network, hardware, operating system, and application development tools such as distributed computing platform), and there is application related heterogeneity (domain conceptual reference model and implementation options such as data structure differences, data schema differences in the case of a distributed database system). In the case of distributed applications, we assume that heterogeneities at lower levels have been resolved by distributed computing platforms. Of concern to us are then only the heterogeneities specific to, in our case, GIS applications.

We will first consider data heterogeneity, which happen at three levels: semantic heterogeneity, data model heterogeneity, and syntactic heterogeneity. Data integration is traditionally manual in the form of data preparation. When considering data integration in the context of networking GIS, however, the key is automation. For example, we need a mechanism to detect possible contextual differences and resolve them without user interaction. The context mediator architecture is a proposal for this purpose. Yet, it is widely accepted that not all semantic heterogeneities can be automatically detected. In this case, manual detection is needed and methods of resolution need to be specified in some way to support real-time data integration. We propose the schema transformation approach for this purpose.

System heterogeneities originally includes network, hardware, operating system, application development tools such as middleware, developing language, DBMS (data model, data access, and transaction management). We will discuss GIS-specific heterogeneities in detail in the next section, including syntactic heterogeneity, structural heterogeneity, and semantic heterogeneity. Heterogeneity is closely related to autonomy. Generally, the more autonomy to be protected, the more heterogeneity need to be resolved. It is difficulty for homogeneity to be consistently achieved even in a top-down design approach.

2.2.3 Providing heterogeneity transparency --- interoperability

The term interoperability has been widely used to refer to quiet different concepts. We use it to refer to the provision of transparency over heterogeneity. We believe that when heterogeneities are resolved then mediated applications become interoperable. The primary strategy to deal with heterogeneity is through standardization.

2.2.4 granularity of resource sharing

By granularity we mean the basic sharing unit of resources. For data, the possible sharing unit can be file, data units managed by DBMS (tables/views in the case of relational database, and objects in the case of object-oriented database) or persistency-enabled systems. For processing capability, the possible sharing units can be services in the form of commands and SQL queries and procedures (methods in the case object-oriented program). Also there is a correlation between data sharing granularity and processing capability sharing granularity. When granularity interacts with other factors of distributed systems, we can get different levels of functions that the resulting distributed application can provide. Table 1 shows the interaction of granularity and transparency.

Table 1: Interaction of granularity and transparency

Data granularity	Processing granularity	Transparency	Example distributed application
File	Command	No	FTP service
File	Command	Yes	Distributed file system
Data item	Process	Yes	Distributed database system
Data item	Process	No	Some multidatabase system
Object	Method	Yes	Application based on distributed object technology

3. Architectural models

3.1 Centralized GIS

A GIS is controlled by a single operating system.

3.2 Centralized GIS supported by file server

A network layer is added to connect individual computer systems. This layer may introduce network heterogeneity. In principle, it is the responsibility of the operating system to shield this heterogeneity from applications. The benefit is that we obtain data sharing through the network file system provided by network operating system.

3.3 Distributed (two-tier) client/server GIS

A GIS is split into two functional layers, namely the client application and the database server. This split has three effects. First, we obtain both data and processing sharing. One database now serves multiple clients, which may be for different applications. Second, it can benefit from the power of a general purpose DBMS, which provides high effectiveness and efficiency over proprietary spatial data file systems. Third, the communication between clients and database servers may introduce heterogeneity. Fourth, as processing is distributed, it is desirable that the system provides distribution transparency. Distributed computing is facilitated by the middleware layer, which is connectivity software that consists of a set of enabling services that allow multiple processes running on one or more machines to interact across a network. Middleware is essential to migrating mainframe applications to client/server applications and to providing for communication across heterogeneous platforms.

3.4 Three-tier client/server GIS

In this model, the GIS client is further divided into two parts: user interface and application server. For enterprise-wide information management, the three-tier architecture provides better scalability and maintainability and also more efficient resource deployment. In this architecture, a Web-based application co-exists with conventional two-tier application. The web layer provides worldwide accessibility, a consistent user interface and a standard connection method between user interface and application servers. The middleware layer has evolved to provide for interoperability in support of the move to client/server architectures through componentization and object-orientation. Although interoperability is one objective of a middleware, middleware itself may introduce heterogeneity if the component systems adopt different middlewares.

3.5 Multi-tier client/server GIS

This is a generalization of the three-tier model, in which the processing part of an application system is further divided into horizontal layers, thus allowing more than one application server in a multi-tier GIS. The purpose is still to achieve better scalability and maintainability.

3.6 Distributed (peer-to-peer) GIS

All the previous models have a centralized database. In fact, the application system is horizontally divided. In this model, the spatial database (spatial data and the associated management processing) is distributed. This introduces the problems of data fragmentation in the design phase and then data integration in processing. A global database layer is needed if fragmentation transparency is desired to provide global directory management, global query processing, global transaction management, and so on. The lack of such a global layer will burden the client or application server with accessing multiple databases and integrating data retrieved, which could be even more involved as each database may evolve. Also the component/peer databases may be heterogeneous in terms of data model, query language, and transaction management model. There are approaches to deal with this heterogeneity. One is to define a standard interface between databases that covers all aspects of DBMS. ODBC and JDBC are two *de facto* industry standards. The other is approach is the so-called gateway technique, which is vendor-dependent.

3.7 Distributed Multi-GIS

In this model, several independent distributed GIS (each of which can be of the client/server model or peer-to-peer model) need to share data and processing. Distributed Multi-GIS is similar to distributed peer-to-peer GIS in the sense that there is more than one database in the system. The essential difference lies in autonomy. In Multi-GIS, each participating system has much more autonomy than the component database in a distributed peer-to-peer system. There may or may not be a global conceptual schema. If there is one, we call it a federated GIS in accordance with the naming practice of other applications including database. The problem to be solved is GIS integration. If there is no tight GIS integration, we need some way to facilitate accessing multi-GIS.

4. Networking geodata vs. networking GIS

We have analyzed the issues, architectural models of networking GIS and also a review of them. We have also indicated the difficulties we will face in various cases of networking GIS. Many applications of distributed database and multidatabase system are driven by actual need.

Traditionally, more attention has been paid to sharing of data as geospatial data is very costly and sharing of them alone is relatively easier than sharing both geospatial data and geospatial services. This is due to the lack of both fundamental support of distributed computing technology and common agreement on spatial information theory and processing model.

A distinct characteristic of GIS is that the production of geodata can and is often separated from the use of geodata. In conventional database applications that serve business management, business activity involves both application and production of business data. Such a case is often not true for GIS applications. It is quiet often that geodata is collected by designated professionals and delivered to geodata users, who seldom produce or update geodata by themselves. For these cases, data sharing does not have to be tightly coupled with data services sharing. This makes networking geodata an important alternative to networking GIS. This separation has two effects: geodata distribution and semantic difference. The unavailability of geodata had been a major impediment to GIS applications. This situation is changing after decades of geodata digitalization and collection. The users are now more concerned with the accessibility of geodata. Technically, interconnected computer networks provide a revolutionarily convenient way to access and distribute geodata. Therefore sharing geodata through computer network (LAN/WAN, Internet/Intranet, Wired/Wireless), called networking geodata here, is an alternative to networking GIS. One way of networking geodata, preferred by many geodata producers and enterprises that consume large amount of geodata, is to build (digital) geodata library. Another way, preferred by departmental usage, is to employ (geodata) file servers.

Unlike conventional database system for business applications, the application or the problems determines the conceptual data model and the implementation of database. When geodata, whose data schema is designed for serving many applications, is obtained externally, the data has to be *prepared* for this application before being ingested by the database. This is a general problem faced by many GIS developers. This also explains why semantic geodata translation is so important to GIS.

5. A review of technologies

5.1 Supporting technologies.

5.1.1 Communication
Although communication happens at all levels of a networked system, we are more concerned here with communication at the application level. When application logic is distributed, we need a channel for them to “talk” to each other so that they can cooperate to complete a task. Network API provides the fundamental support for inter-process communications. Although network Application Programming Interface (API) is flexible enough to build any networked application, they are low-level in terms of abstraction. Application developers need to deal with too much detail of communication and deal with heterogeneity among computers. Therefore higher-level communication mechanisms are designed to facilitate application development including RPC of DEC for communication between distributed procedural process and RMI of Corba, Java and COM/DCOM communication between distributed objects and distributed event notification for distributed event-based system.

Moreover, proprietary protocols and mechanisms need to be designed and employed for communication and coordination among component applications and therefore interoperability among applications from different vendors is not facilitated. DCP (Distributed Computing Platforms) are developed for facilitating cooperation and interoperability at the system level. It aims to provide a neutral platform that not only makes geographical distribution transparent but also hides the heterogeneity among applications due to differences in hardware, operating system, developing language, binary coding practice and therefore protects autonomy. Older procedure-oriented DCP, for example the old version of OSF DCE (Distributed Computing Environment) is not desirable in the sense of detail transparency and evolvement maintenance and now serves as infrastructure for building higher-level DCP such as CORBA and DCOM. Object orientation is important to both distributed and non-distributed computing platforms. Current DCP are all object oriented or OO alike. OODCP can also be called distributed component/object technology. The essence of DCP is for software components, which are to cooperate with others, to be encapsulated and expose only its interfaces to other components. The key technique of DCP is the mechanism of finding the desired serving components, for instance the ORB (Object Request Broker) of CORBA (Common Object Request Broker Architecture) and DCOM (Distributed Component Object Model).

5.1.2 Openness and Component-based software development
Component-Based Software Development (CBSD) focuses on building large software systems by integrating existing software components. By enhancing the flexibility and maintainability of systems, this approach can potentially be used to reduce software development costs, assemble systems rapidly, and reduce the spiraling maintenance burden associated with the support and upgrade of large systems. Adapting preexisting components to a system requires techniques such as API, wrapping, bridging, or mediating, as well as an increased understanding of architectural interactions and components' properties.

5.1.3 The Web
The Web was originally designed for information publication. It is now often viewed as a huge worldwide information repository. Later, the Web technology was extended to accommodate computing services by plug-ins, applets in the Web browser, and server applets at the Web-server. The Web has therefore become a web of services. There are now many “Web-based” applications. From the viewpoint of distributed computing, we consider the Web as a worldwide user interface, which provides potentially universal access to worldwide information and services.

5.1.4 Advances of database technology
Database systems are undergoing great changes from the relational to the object-relational and object-oriented, from proprietary to open, and from homogenous to heterogeneous. These changes have significant impacts on GIS. Distributed computing in database systems has been studied for a long time and there are many mature commercial systems available. However, these distributed systems have been mostly homogenous until the ODBC technique came into being and later JDBC. Two trends of database technology are multidatabase and object-orientation. The research and development of multidatabase system are in response to the demand on integrating existing and emerging databases that are inherently heterogeneous and distributed. The heterogeneity includes data models, data structures, query languages, data schemas, and so on. It is well known that relational databases are not well suited for many non-traditional applications including GIS. The current trend is to modify the relational model with concepts from the object oriented DBMS.

5.2 Network-enabling GIS
Many GIS tools were developed without the coordination among vendors, making applications developed using tools from different vendors difficult to communicate with each other except by exchanging data in some intermediate format. Also, GIS applications were developed with the proprietary languages.

The development of Internet makes the physical connection of GIS systems a fact. The physically connected GIS need some kind of mechanism to communicate, which can be based on the Web, proprietary protocols between GIS tools, or GIS applications and DCP. These various communicating mechanisms differ in capability and flexibility.

The object orientation development of database technology makes it more suitable for GIS applications. GIS tools based on object-oriented database or object-relational database is the current trend. Major DBMS vendors are providing spatial data managing tools. Examples are the Spatial Option of Oracle [Oracle, 1999] and Spatial Blader of Informix [Informix, 1999].

The data transfer mechanism of network GIS is still open. The point here is the unit of transfer and the data type for transfer. The possible candidates are a map or an image, a layer, a set of (related) spatial objects (as the result of a query), a spatial object, and part of a spatial object. Actually, there is a relationship between the unit of data transfer and the sophistication of network GIS. It can be said that the smaller the unit of transfer, the more sophisticated is the network GIS. ISO211 tends to adopt XML [W3C, 1998; Zaslavsky, 2000] for transfer, and XML is likely to become the standard for exchanging objects over the Internet. There are standardization efforts to develop mechanisms for transferring spatial objects, such the WKS (Well-Known Structure) and GML of OpenGIS and OSF of SAIF.

5.3 Network-enabled Geodata sharing

By data sharing systems, we refer to GIS related systems that make geodata sharable to many users but no processing capability can be shared. According to how data sharing happens, they can be further classified into three classes, distributed GIS, spatial data catalog service, and web map server.

5.3.1 Distributed GIS

At a primitive level, network file systems are often used for data sharing among users of a working group. Several cases of such GIS can be identified, which vary in the level of sophistication.

Geodata file servers and geodata file clients are actually file servers that can contain or logically contain great amount of spatial data. Clients of spatial data access them through network file system protocols. It should be noted that the coupling of server and clients happens at the system level and not at the application level. The essence here is to use computer network for data transfer instead of tapes or CD.

Geodata maintaining servers and geodata clients: this server is similar to geodata file server but it provides “write-locking”. The server deals with concurrent writable accessing and long transaction. An example is the Arc/Info Librarian.

Feature level spatial data server: this is actually what can be called a multi-user geodata file system. Such geodata server provides feature level transaction management. That means a number of users can access the same spatial data in a writable mode yet the system guarantees that one feature would not be modified by multi-users at the same time. Arc/Storm is an example of such a geodata data server tool.

In the case of format-open GIS, data of different formats can be shared between GIS. Keep in mind that in this case of network GIS, one GIS can only read data of another GIS and nothing more. Actually the two involved GIS do not talk to each other directly. What actually happens is the sharing of data without worrying about format differences. GeoMedia is a data-open GIS, which can directly read data in native formats of Arc/Info, MicroStation, ArcView and Oracle Spatial Data Option, and so on in an online mode. We call this case friend GIS, meaning that the data structure of one GIS product is open and ready for reading to other GIS. However, it should be mentioned that although in this case data can be read, possibly online, the external data user could not write or modify these data.

In the case of distributed GIS system, there is no system coupling between different GIS systems. Actually, although GIS are connected to each other, they are isolated logically since no direct communication can happen between these systems.

True distributed GIS have not emerged yet in the form of commercial products. A current approach is to use data-open GIS middleware (not a GIS by itself) to provide mediation between clients and heterogeneous geodata sources. There are many prototype middleware that provide access to heterogeneous geodata sources. Examples are the OGDI (Open Geospatial Datastore Interface) [Clement et al., 1999], which is an application programming interface sitting between a GIS application and various geospatial data products, providing a uniform method to access these data products. The OGDI effort developed an intermediate data model and a protocol (GLTP, geographic library transfer protocol) for communication between clients and servers. For each kind of geodata source, a ‘driver’ is developed to read data from it. There are other similar projects that provide data-open GIS middleware.

It is worth noting that the interaction between ArcExporer and ArcInfo or an ESRI spatial data server is similar to the interaction between an OGDI client and OGDI server in that they both use their proprietary protocols respectively for communication. In the OGDI architecture, a driver is needed to read a proprietary geodata format. The difference is also significant in that in the OGDI case spatial data needs to be converted into the intermediate format before the client can view the data while in the case ArcExporer-ArcView/ArcInfo no conversion is needed since ArcExporer is built to read SHAPE and other ESRI file formats.

The OpenMap [Cranston et al., 1999] is another example of data-open GIS tool suites. It also defines the interface between clients and logical data servers (application server). The difference from OGDI is that it is based on CORBA instead of TCP/IP. The interface between logical data server and the real database is a dedicated middleware called Specialist that provides access to the spatial database.

The justification of data-open GIS is that most spatial data are acquired and maintained by designated task groups and in most cases data users may want to have real-time access to these data while they are neither allowed to nor interested in modifying these data. The obvious disadvantage of data open (only) GIS is that their users cannot actively make use of the processing capability of the data server even if they are online and ready to serve others. This is the objective of service open GIS.

OGIS is an industry-wide effort toward interoperable GIS, which would benefit GIS in two ways: facilitates interoperability among component-based GIS tools (from different vendors) and facilitates integration of geodata and geoprocessing. OGIS has two levels: the abstract level and the implementation level. The abstract specification is platform-independent and the implementation specification is targeted to specific DCP. Parallel to the database design problem, the abstract specification is similar to a conceptual schema and the implementation specification is the DBMS-dependent schema definition. In the process of implementing abstract specification, the meta model of each DCP will impose certain restrictions. The relationship between OGIS-compliant GIS based on different DCP is similar to heterogeneous databases.

Although OGIS is standardizing the interface between GIS components, developing application system is not a trivial.matter. [Camara et al., 1999] reported that different OpenGIS implementations varied significantly. When legacy data storage is to be integrated, it is necessary to develop a wrapper for it. When multiple data sources are to be accessed, many problems may rise. One of them is that although they all support the OGIS specification, they may vary in access methods: one may support SQL with geometry data types, another does not support geometry types, and yet another may not support SQL at all. If we want to transparently access multiple data servers, the server integration problem needs to be tackled as well. The current strategy is to use wrappers, GIS middleware, and mediators.

In addition, OpenGIS is now defined over three main distributed computing platforms, CORBA, COM/OLE and SQL. On the one hand, the interoperability among these different DCP needs to be addressed. On the other hand, new DCP may emerge and existing DCP may evolve. Java is now considered to be very promising not only as a programming language but also as a platform of its own. Sybase is redefining its database architecture using Java technology [Sybase, 2000]. As another example, although SQL is prominent in today’s databases, it is doubtful that SQL can serve as a future data manipulation language. New standardization efforts, such as JDO (Java Data Objects) and OQL (Object Query Language), could be more widely accepted in the future.

The ultimate goal is to develop distributed heterogeneous autonomous GIS (Multi-GIS). Significant progress has been made in GIS networking. The various kinds of network GIS discussed in the previous sections tackled physical distribution and different level of heterogeneity. However, as we can see, the problem of cooperation of servers from different vendors has not yet been resolved, and none of them have touched the semantic level of heterogeneity. In fact, the semantic level interoperability has just been addressed by researcher and is far from being practical [Xu et al., 1999].

5.3.2 Spatial catalog services

Spatial catalog services provide the capabilities for organizing and managing spatial metadata. Catalog services simplify the process of data discovery in a large spatial data store, facilitate data transfer, and help data fusion. Strictly speaking, spatial data catalog service is not different from other data catalog services. The success of these systems depends on the power of searching available spatial data and the interoperability of different catalog services [Cranston et al., 1999]. The DIAL [Di et al., 1999] is an example of such systems. OGC also identifies catalog service as an important GIS service and is putting effort on the standardization of catalog services.

5.3.3 Web map server

This kind of network GIS distributes spatial data to users in the form of maps and in the image data type through the Web. There are now several commercial tools including ArcView IMS, MapObject IMS from ESRI, Map IMS from MapInfo and GeoMedia Web Map from Intergraph. The OGC has issued a specification for Web Mapping. There are also many application Web mapping systems in use such as the CenterMap of Hong Kong [CenterMap, 1999].

It is interesting to compare the web mapping systems with the data-open GIS. In the former case, the client is just a browser. The distributed spatial data is in the form of image maps while in the latter case, spatial data is usually transferred from the server to the client. The former has the advantages that any user connected to the Internet can access the Web mapping server and no specialized client program is needed. However, it has the disadvantage that every query leads to regeneration of the map and therefore is likely to overload the server. The data-open GIS mode is different. The client program has to be specifically developed although it can be a Web browser with plug-ins. Usually, the client program has the ability to display, zoom, answer some query and possibly do other light geospatial analysis. It is therefore likely to reduce network flow and the load of server.

6. Conclusion

In this paper, we have given a perspective on network-enabling GIS. We have assumed that geodata and its associated management processing is the core of GIS. Our viewpoint is from the software architecture by first decomposing a GIS, distribute the resultant components over a network, and then assemble and integrate these components. In doing so, we have exposed many problems hidden in stand-alone GIS. These problems include data distribution, map-boundary mismatches, and topological continuity. As interoperability is a requirement for integrating executable software components, component-based software development helps to support interoperability among components. The OGIS effort helps interoperability among GIS components while DBMS-based spatial data management helps the integration of geospatial information with the rest of enterprise information. While truly distributed GIS system involving highly autonomous component is still a long way to go, one-vendor solution will still be favored by applications that require optimal performance.

7. References

Camara G, R. Thome and U. Freitas, (1999) Interoperability in Practice: Problems in Semantic conversion from current technology to OpenGIS. In Interoperating Geographic Information Systems, Proceedings of Second International Conference, INTEROP'99, Zurich, Switzerland, March10-12, 1999.

CenterMap Corp. (1999) http://www.centermap.com

Clement G, C. Larouche, and P. Morin (1999) Interoperating geographic information systems using open geospatial datastore interface (OGDI). In: M. Goodchild and M. Egenhofer (eds) Interoperating geographic information systems. Kluwer academic publishers, 1999.

Cranston C.B., F. Brabec and G.R. Hjaltason (1999) Adding an interoperable server interface to a spatial database: Implementation experiences with OpenMap. In M. Goodchild and M. Egenhofer (eds) Interoperating Geographic Information Systems, Proceedings of Second International Conference, INTEROP'99, Zurich, Switzerland, March10-12, 1999.

Di L.P., R. Suresh, K. and Doan (1999) DIAL: A web-based interoperable scientific data distribution system. In: Interoperating geographic information systems. Kluwer academic publishers, 1999

Informix Corp. (1999) http://www.informix.com

Oracle Corp. (1999) http://www.oracle.com

Sybase Corp. (2000) http://www.sybase.com

W3C (1998). Extensible Markup Language (XML) 1.0. W3C Recommendation, 10-February-1998. http://www.w3.org/TR/1998/REC-xml-19980210

Xu Z.,Y.C. Lee, and Y.Q. Chen (1999) Schema transformation for semantic geodata translation, XIXth Congress of the ISPRS, Amsterdam, The Netherlands.

Zaslavsky I., R. Marciano, and A. Gupta (2000) XML-based Spatial Data Mediation Infrastructure for Global Interoperability, 4th Global Spatial Data Infrastructure Conference, Cape Town, South Africa, 13-15 March 2000.