Proceedings of the 6th AGILE
April 24th-26th, 2003 – Lyon, France

PROGRESSIVE VECTOR DATA TRANSMISSION

Haiyang Han, Vincent Tao, Huayi Wu

Geospatial Information and Communication Technology Lab (GeoICT lab), Department of Earth and Space Science & Engineering, York University, 4700 Keele Street, Toronto, Ontario, Canada M3J1P3

Email: {hhan,tao,hwu}@yorku.ca

1. INTRODUCTION

In recent years, World Wide Web (WWW) has increasingly become an indispensable platform for remote data access and sharing. Web-based Geographic Information Systems (GIS) have more and more been recognized and applied as an effective way of publishing and sharing spatial data with others [1]. Nevertheless, a dilemma is that its promising future is not equivalent to its current market application. One of the key technical problems is its low data loading and updating efficiency. Although current Internet bandwidth has been expanded many times compared to years ago, it is still unable to meet the increasing requirements of quickly distributing and processing large data sets on the web. Recent research work has proved that efficient spatial data compression and progressive data transmission, especially in the form of raster and DEM [4], are very useful in quickly transmitting and displaying spatial data over constrained communication channels.

Although significant progress has been achieved in progressive image and DEM transmission and compression [5, 12, 13, 14], fewer breakthroughs have been made in the vector domain [7,8,9]. Different from pixel-based image, according to Open GIS specification [6], vector data are composed of geometric objects in the form of points, curves and surfaces. It cannot be simply sampled and compressed as the image data.

In the remainder of this paper, we fist discuss the concept, principles and challenges behind the implementation of progressive vector data transmission. From the system implementation's point of view, a couple of fundamental principles are brought up. Following those principles, an overall architecture is given to demonstrate the entire workflow. To implement the progressive vector transmission, some critical technical issues have to be addressed such as appropriate spatial index, line simplification and multiple-representation and encoding algorithms, definition and interpretation of data request and response communication protocol, data consistency, interoperability and openness, client- and server-sides buffer management and presentation issues and so on. Finally, a system implementation interface is demonstrated to explain the feasibility and effectiveness. A conclusion is given to point out the strengths and weaknesses of the implementation as well as the future work.

2. CHALLENGES IN IMPLEMENTING PROGRESSIVE VECTOR TRANSMISSION

Conventionally, a user cannot perform any map-related operations in a browser until all data requested have been downloaded and displayed on the client side. One solution to the problem is the progressive transmission of vector data brought up by Bertolotto [2, 3]. The concept of progressive vector transmission contains two-side meanings. On one hand, because a vector layer is regarded as a set of geometric objects, when a user requests a

layer within a given map extent, all requested data on the server side will be sent to the client piece by piece. Each piece can be a packet of one or more geometric objects. Following the “first send, first display” principle, it allows a piece of data to be transmitted first and then instantly be displayed on the client side as soon as it arrives. Then when the next piece of data arrives, it will join the preserved data and display again. A user does not have to wait until all required data are downloaded. On the other hand, within a geometric object, each is organized into multiple representations on the server side. A request from a user contains metadata information such as the current display scale and screen resolution, which helps the server decide which vertices should be firstly transmitted to the client. As a result, a simplified version of a geometric object will eventually show up with an acceptable quality and resolution on the client side. A further drilling-in will transmit more vertices to enrich the skeleton of a geometric object.

Technically speaking, the time length of progressive lossless vector transmission will be relatively longer than that of downloading the entire raw data sets because of added encoding indexes. But it had its own advantages. It adds tremendous values in increasing system flexibility, interactivity and efficiency. Instead of waiting for the whole data to be downloaded slowly and then displayed, a user can visually know the transmission process when data is been loading and displaying piece by piece. In addition, a user has a control over the data transmission process. If a coarser version at a higher level has already met his requirements, a user can just use this coarser version to conduct some operations. It is unnecessary to ask for finer increments in this situation. If a user is not satisfied with the coarser data, by drilling into the map, finer details will be fetched from the server and then merged into the preserved coarser data to offset the resolution flaw, in the meantime, a user can also perform some operations using available data. Such a transmission can be interrupted at any time during the process.

Clearly, the progressive vector transmission introduces an innovative and efficient way in dramatically improving vector data downloading performance. Nevertheless, the implementation of this technology is still not sophisticated in the sense that inter-disciplinary knowledge, which ranges from computer graphics, network communication, spatial data structure and model, to cartographic generalization and multiple representations, are required in this process. Each discipline has its own understanding and requirements regarding a specific problem. Theoretically, it is nothing wrong with that. But the realistic challenge in implementing the progressive vector transmission system lies in that some requirements are contradictory when we try to combine all those disciplines together. From the implementation’s point of view, there are always some trade offs in software development. This paper will address some critical design and implementation principles in the progressive vector transmission. Briefly, those points include:

- Application oriented implementation
- Data consistency control
- Data storage and multiple representations
- Real time generalization and compression
- Appropriate algorithms and accuracy control
- Efficient spatial index mechanism
- Buffer management and fine-tuning
- Interoperability and Openness

3. OVERALL ARCHITECTURE OF PROGRESSIVE VECTOR TRANSMISSION

The emerging technique of progressive vector transmission aims to break network bottleneck of spatial data loading and processing, to improve performance, the system should be based on multiple tier client/server architecture as shown in Figure 1. Client side

is for data request, load, display and operations; web server handles network connection and firewall as fundamental communication basis; application server receives processes and responds all data requests from both streaming and non-streaming spatial databases. Additionally, author is a pre-processing tool for streaming data preparation.

When a wired or wireless mapping client is launched and connected to the server, the client, namely the first tier, which can be written either in Java or in other computer languages, will be dynamically loaded from the remote server and reside inside the mobile devices. A user can perform display or querying operations, and communicates with its back end via this graphic interface by sending a sequence of compact data loading requests to the server.

The second tier can be Tom Cat or other web servers, which support HTTP and firewall functions. The third tier or application server relies on the web server and handles all application logic: namely, it receives the data request from the client, and then retrieves the raw database to acquire geometric objects. Upon bandwidth and request protocol, a required entire or partial feature is computed and encoded in a certain order in the memory on the fly. All required geometric objects lines up and wait for transmission according to their loading priorities on the server side. Before transmission, one or more geometric objects are bundled into a packet. One after one, all packets are progressively transmitted to the client side and display on the screen. The client will confirm the arrival of each packet. If one packet fails to reach or some data are dropped, the client will report the server and ask for resending.

If a coarser version of a geometric feature has already preserved on the client side regardless of its incomplete in either spatial or frequency domain, only additional or incremental parts are sent to the client. The newly introduced vertices, if containing more details of a curve or a surface in a division, will be inserted into the coarser version of the same geometry to generate a finer version and redraw it on the screen. If remaining divisions of a feature is fetched to the client, the existing parts will merge with the coming ones to fully or partially restore the original feature. Once all spatial divisions of a feature are merged on the client side, neither logical nor physical spatial divisions exist in this feature. As a result, when a user drills into the map, initially, only the coarsest version exactly within the display map extent shows up. Under a certain control, both the data precision and waiting time can be acceptable. If a user expects a finer map, continuous drilling-in will bring more details from the server to the preserved data on the client side. At the lowest level, all raw data will be losslessly displayed. When a user continuously zooms back to the top level from the lowest level, the coarsest version will show up again. Therefore, the display level and data details of the whole raw data set on the client side are adaptive according to the screen resolution and the scale level. In addition, pool management mechanism will monitor user-side memory consumption and maintain it under an acceptable level to avoid overhead crush

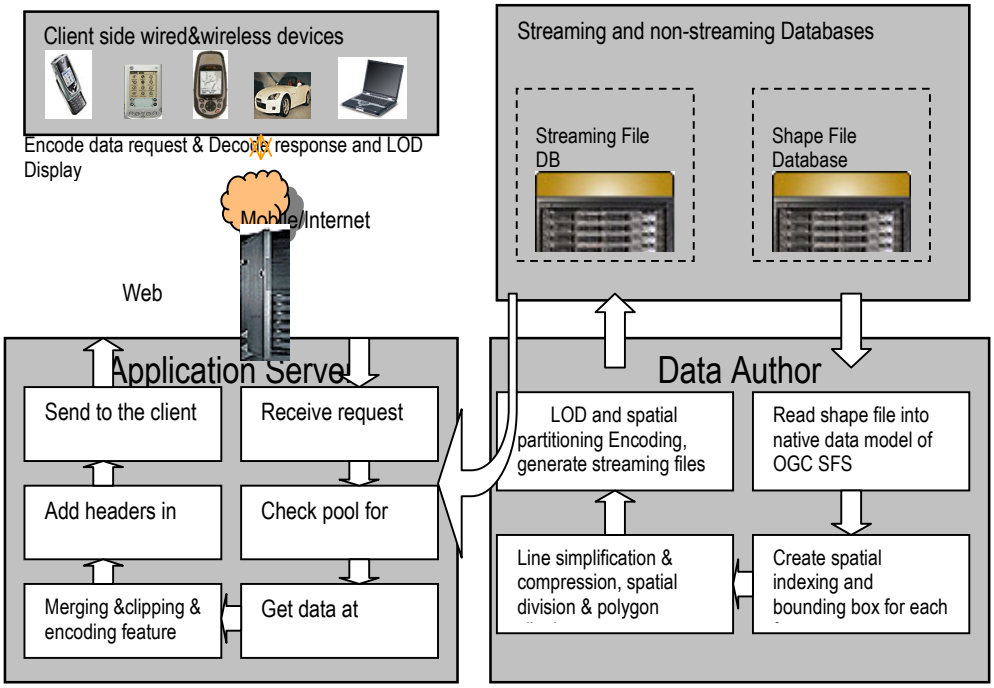

Fig. 1 System architecture

In order to be compatible with different vector formats, the built-in vector data model on bother client- and server- sides should be an implementation of Open GIS simple feature specification (SIF), which enables the system to be scalable and handle complex geometric objects. Therefore, no matter what kinds of external data formats are, all are read and imported into internal OGC SIF data model prior to encoding.

Database side could be either file- or relational database-based. In addition to the three-tier, an authoring tool is also developed to pre-compute raw data sets. An author will read and encode raw data according to certain criteria and then save encoded data into databases. After the data is published on the server with certain configuration, user is able to access and load data either progressively or not.

Because the computational tasks on the server side are relatively heavy, a server cluster, cache management and other hardware and software solutions are recommended to optimise data access and processing performance.

Although a multi-tier architecture for wired and wireless system has become very common today, in terms of the system implementation of progressive vector transmission, a handful of techniques are still very critical in achieving system's high performance. Those implementation techniques include, accuracy control, data encoding algorithms, data request and response protocols, client- and server- side buffer management as well as OGC interoperability. Following sections will address those issues.

4. PULL VS. PUSH DATA TRANSMISSION MODELS

Push and Pull are two mostly used spatial data loading models over Internet. Basically, Internet GIS architecture can be partitioned into two tiers: client side and server side. Dependent to the “thin” or “thick” of client- and server-side functionality, either Push or Pull models is applied to access, download, process and operate data from the server. Using Pull model, the client will compute spatial objects in which extent it needs and then ask the server to send exactly specified spatial objects it need. To large degree, server behaves like a slave, following client’s request without storing personalized information. On the country, Pull model behaves very differently in Internet GIS architecture. Upon user’s map loading request, the client will only sent the map display extent to the server. The serve will first examine user’s request and then overlap this display extent with existing data sets in the databases to decide which spatial objects should be sent to the client. In addition, in order to record client’s operation, the server also needs a session for each client to record personalized information including the current status of each client. Generally, thin client/thick server uses the Push model, while fat client/thin server uses Pull model.

In terms of progressive vector transmission, the situation is very different from current data loading mechanism. “Progressive transmission” does not only mean that only spatial data at a certain LOD level within exact areas of interest is queried and loaded from the server, it also means the requested data on the server side is wrapped into many small packets, which are fetched to the client side independently. Because each data packet is independently sent to the client for display and analysis, it is of importance to know which part of a spatial object should be sent and how to reunite this part with others, and also how many parts have already existed on the client side or remained on the server side.

Certainly, server can use Push model to record the status of each client. When a user requests a map for the first time, the server will examine how many spatial objects are within the requested display extent and send those objects at a certain LOD level within the areas of interest. Because the user will display the returned map continuously, more requests will arrive at server side. In order to push each packet to the client side, the server has to maintain a session to record changes on client side. Push model takes advantages of server-side powerful multi-task processing capabilities to relieve client. But shortcoming is also obvious. Progressive vector transmission needs to maintain spatial index on both client and server sides. Allocating data dispatching tasks to the server will consume tremendous computing resources in session maintenance. In addition, close coupling between client and server in Push model will also significantly increase system complexity.

An alternative to above problems is Pull model used in this paper. There are some reasons for the choice. First, to enable vector data transmission progressively, spatial index has to be maintained on both sides; therefore, it is convenient and efficient to let the client itself to decide which spatial objects it needs and how those objects should be wrapped and transmitted. Second, Pull model can be easier to know exactly what the client needs and to detect computing environmental changes. Such changes include data request abortion, network bandwidth and real-time transmission rates, error detection as well as lost packet recovery. Finally, Pull model makes server’s job easier and simplifies system architecture, while takes full advantages of client side’s computing resources with the consideration of increasing PC performance.

5. PROGRESSIVE VECTOR TRANSMISSION ALGORITHMS

Efficient progressive vector transmission requires the combination of inter disciplinary knowledge. Because factors such as data quality, transmission rate, topological changes, band width and so on, all have to be taken into consideration, some trade offs are unavoidable and also necessary in this process.

5.1 Accuracy Control

A map is drawn according to a given scale. When zooming from one scale to another scale, data consistency has to be taken into consideration. At present, most of wired/wireless GIS oriented applications mainly focus on visualization and simple querying. Therefore, for simplicity, the display resolution is controlled by the physical distance, which a pixel represents at the current screen size and scale level.

5.2 Level-of-detail (LOD) Algorithm

A seemingly countless number of line simplification algorithms for LOD display have been developed over the past three decades. This paper has no intention to discuss those algorithms in detail. The Douglas and Peucker algorithm is chosen in this implementation due to its most frequent use and acceptable results [Kreveld, 2000]. In addition, because the algorithm implementation is encapsulated in a separate mathematical library, it is easy to update or replace the current algorithm whenever a new algorithm is proved to be more suitable than the current one.

On the client side, when quickly drilling-in and drilling-out in a map, even all data have been preserved on the client side, at the top level, the system should only display the coarsest level in order to speed up drawing and avoid unnecessary details. To reach this objective, a scale level structure is also maintained at the LineString level on the client side when vertices are progressively received. The detailed algorithm of multiple scale representation will be presented in the full paper.

5.3 Spatial Index

For simplicity, grid index is used for fast data retrieval. Based on the size of raw data set, a number of grids are created. Each grid contains a number of geometry. When the system is launched, the grid index, which contains the encoding metadata of its geometric objects, will be sent to the client first. Then depend upon the bandwidth, display extent and other factors, one or more grids are packed and sent to the sever side for data at a certain level.

6. DATA REQUEST AND RESPONSE PROTOCOL

Because each time, only increments of a geometric object are sent to the client side, therefore, in client/server architecture, the server and the client have to talk with each other. As far as the server is concerned, before progressively transmitting increments to a client, it has to know which geometry is needed and how many vertices of a geometric object have already been preserved on the client side. Without such information, it becomes impossible for the server to fetch exactly the required vertices at the required level. Likewise, when a client receives a data packet from the server, the client needs to interpret the content of the data and how to merge the newly introduced increments into the preserved data and then to display them on the screen. The two-way talk requires the definition of data request and

response communication protocol. Figure 2 demonstrates the data request and response workflow in a sequence diagram, which is depicted using Unified Modeling Language (UML).

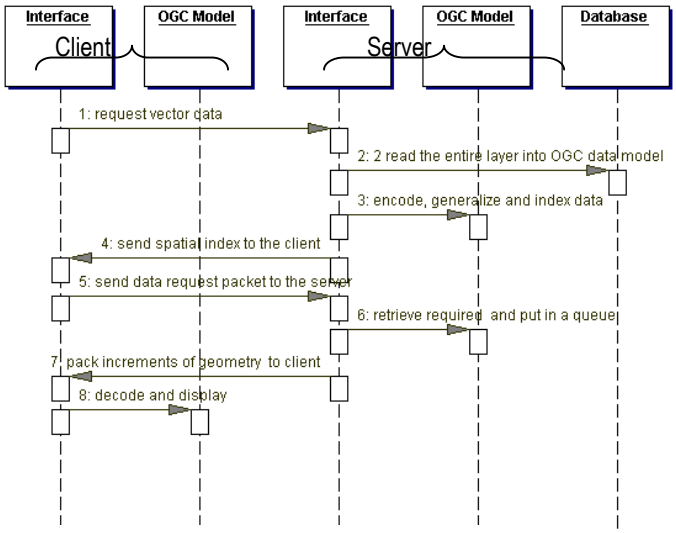

Fig. 2 data request and respon

7. BUFFER MANAGEMENT AND PRESENTATION ISSUES

Due to the contradiction between the limited memory and fast data transmission under a constrained bandwidth on the client side, buffer management becomes extremely important in achieving high performance. Likewise, on the serve side, quick processing and response to client's request relies on efficient cache management.

8. EXAMPLES

In terms of progressive vector transmission, the original vector data used in tests is the ESRI shape file. The figure as below is the configuration interface for pre-computing original data sets in the authoring tool. By defining the number of levels, scale value at each level and screen size, data distribution and compression ratio at each level can be under control.

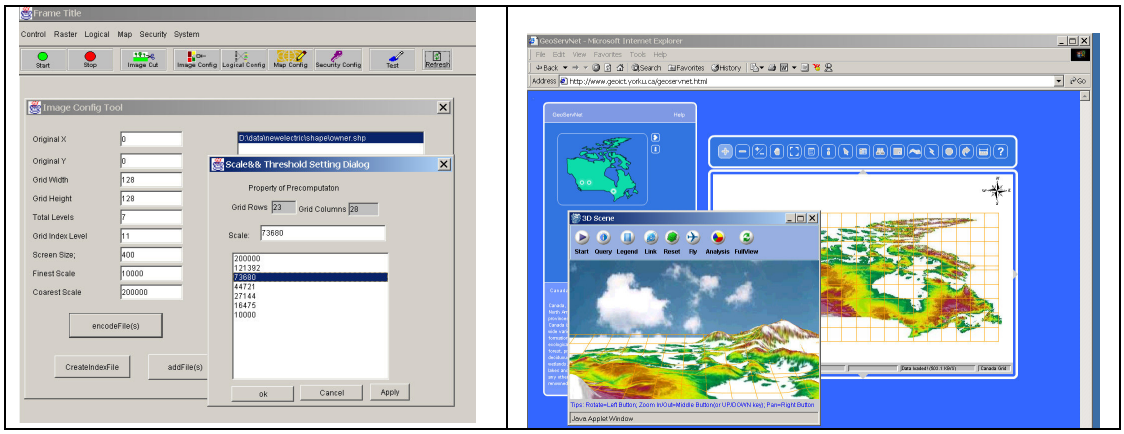

Fig. 3 Streaming based system implementation

9. CONCLUSIONS AND FUTURE WORK

9.1 Conclusions

With the popularity of Internet and web services, offering on demand geospatial data and geoprocessing services have become a very promising tendency in GeoICT field. Over past years, history has proved that innovation in information and communication technologies (ICT) often brings novel ideas into GIS field and promotes the development of GIS. To some extent, GIS is really a small field and its technology development is limited to the availability and application and mainstream information and communication technology. For instance, the emergence of object-relational DBMS and spatial DBMS developed by IT mainstream is becoming a popular tool of managing large-scale spatial database in the GIS field. Another obvious example is that GIS industry uses Internet and borrows the multiple tier architecture of Internet related applications for its own network-centric GIS development. All these prove that GIS industry needs to be more active to borrow and apply ICT in its won domain.

Especially, over the last decade, more and more GIS people have realized the importance of high performance computing and contribute tremendous efforts in geocomputation fields. The most significant achievements focus on distributed geoprocessing computing and parallel geoprocessing field. Internet computing as a part of high performance computing has attracted more and more attention with the popularity of Internet. Nevertheless, the development of network bandwidth is also behind human's expectations. To solve the contradiction between the bandwidth constraints and human's needs, streaming has emerged as a very popular technology to remotely access and process data. Although this hot technology has been applied widely in the information and communication technology fields, most of GIS people have not realized the promising future and influence of streaming on spatial web services.

Using of streaming for progressive spatial data compression and transmission can be further divided into three topics, namely vector streaming, image streaming and terrain streaming. The emergence of JPEG in IT industry, to large degree, has also defined the streaming transmission standard for GIS industry. Some researches have focused on the progressive terrain transmission and commercial systems are also available on line to interactively visualize and model our real world under a limited bandwidth. Come to

progressive vector transmission, neither IT industry nor GIS people find an efficient way to deal with vector transmission over the Internet. The reason is partly because the vector data size is traditionally considered much smaller than the sizes of image and terrain. Nevertheless, the availability of progressive image and terrain transmission technologies has significantly changed the situation. Vector data transmission over the Internet is emerging as the primary bottleneck of affecting performance. Therefore, this paper is aiming to come up with a solution to solve the problem.

None of techniques applied in this paper is really new in the IT and GIS field, but the novelty of the proposed solution lies in that it couples a number of available technologies together and brings up an algorithm from the system implementation's point of view.

Specifically speaking, first, spatial index is superimposed over the vector data to improve query efficiency. Although tremendous researches have discussed on the best algorithm performance from different points of view, to be simplistic and practical, the paper just picks up the most common grid index as the proposed algorithm basis. Nevertheless, the spatial indexing can only speed up client and server sides querying efficiency, it is unable to reduce the physical data transmission amount over Internet. Basically, two strategies help data reduction. The first one is the spatial partitioning, which loads only points of a spatial object within display extent. The second is the frequency partitioning, which generalizes the raw data for level of detail representation. Both of two methods can significantly reduce the data transmission amount over Internet. The proposed algorithm aims to combine those two techniques for progressive vector transmission. The main challenges of the proposed algorithm are how to index the spatial data and how to conduct spatial object clipping on the server side and merging on the client side along with high performance.

Other than the proposed algorithm, from system implementation's point of view, multiple tier architecture is indispensable. In addition, many additional components have to be developed to support streaming transmission. Those components include buffer management on both client and server side, error detection and recovery mechanism, vector network transmission protocol, compressed LOD vector data storage structure, display control and multi threading management as well as others.

Furthermore, the paper describes the system design and implementation. To prove the effectiveness and efficiency of proposed solution, different types of vector data are loaded for testing and performance evaluation. The solution is also proved to be very innovative in practical project application such as Canada Disaster Management Information System.

Overall, progressive vector streaming technology is a very promising field in both IT and GIS fields. It is no doubt that this technology will gain more and more attention in the near future. The contributions of this research are in following sides. First, it reviews all related background research in streaming fields including media, still image, terrain and vector. Second, an innovative spatial and frequency fusion algorithm is proposed for high performance vector transmission. Third, a system implementation is given to demonstrate the effectiveness and efficiency of the proposed algorithm. In addition, a series of system implementation related issues are discussed in detail. Finally, a performance evaluation is conducted to give an overview of streaming against non-streaming methods.

9.2 Future Work

This research is only a preliminary start in this field. There is still plenty of room for further improvements in the future. It can be summarized as follows.

First, data consistency is not a fundamental issue in this research, partly due to the fact that it is still an ongoing research and the solution is not mature enough in the cartography field. For example, when different vector layers are simplified separately, topology and consistency among layers will be changed and thereby will cause the delivery of misleading map information. Trade-off in this research is that temporary consistency distortion can be gradually offset with follow up refinement at a more detailed scale level. But future work should take the consistency into consideration.

Second, the research does not put emphasis on progressive vector compression. Raw vector data stores plenty of redundant points. In some cases, more than 70% coordinate pairs can be eliminated without significantly lowering its accuracy. Furthermore, using chain coding, run-length coding and other compression techniques, the physical storage space can be further reduced. At this time, industry compression standard for vector data is not available. But its coming should be a matter of time. The vector compression will be a very promising field in the coming future. It will also be my next research focus.

Third, in this paper, the LOD simplification is conducted beforehand instead of real time. It is true that such a strategy saves plenty of computational resources and makes the on-demand loading and transmission more efficient. But it also causes another data consistency problem. Since all vector data are pre-processed and encoded in self-contained data format in advance and enterprise GIS software does not support direct editing for those streaming data format, each updating of vector data will cause pre-processing to generate data streaming format. The solution to this problem is to generalize vector data in real time. Each time, upon user's request, the server is able to load raw data sets and generate the streaming data for transmission. The procedure is time consuming, but parallel and distributed computing algorithms as well as other techniques could improve the performance.

Fourth, at present, no vector data compression and transmission industry standards are available. With the increasing popularity of spatial web services, it is becoming really necessary to develop such standards. Currently, OGS has developed various XML/GML based data exchange standards. It does improve spatial data interoperability. But the low efficiency of XML has largely limited its practical application. One solution is to employ progressive spatial data transmission into XML/GML to improve performance. To reach the objective, OGC needs to implement new XML/GML based data transfer schema by adding some display and transfer indexing metadata in the existing standards. To be simplistic and practical, the first step could be implementing progressive vector transmission algorithms on the server side. Each time, upon user's request and its scale level, vector data are transmitted to the client side in different representations. For example, for the first time, user is able to view a coarsest version. But continuously scaling down will bring a refinement version to the display screen. There is no correlation between LOD representations. Data redundancy exists in this process. But the advantage of this initial step is allowing LOD representation on the client side without changing existing OGC standards. Implementation of streaming transmission standards could follow the success of the proposed initial step.

Fifth, integral spatial data services will be one of future tendencies. When progressive vector, image and terrain transmission techniques become more mature, we can foresee the necessity of integrating all three types of spatial data for progressive transmission. Therefore, how to coordinate the transmission pace for such integral services will be a new challenge.

10. REFERENCES

[1] Tao, C.V. 2001. Towards Location based GIServices, *Journal of Geospatial Engineering,* Vol. 3, No. 2, pp. 1-10.

[2] Bertolotto, M. and M. J. Egenhofer. 2001. Progressive Transmission of Vector Map Data over the World Wide Web. *GeoInformatica* 5:4,345-373.

[3] Bertolotto, M. and M. J. Egenhofer. 1999. Progressive Vector Transmission. *7th ACM Symposium on Advances in Geographic Information Systems*, Kansas City, Mo.

[4] Pajarola, R. 1998. Large Scale Terrain Visualization Using the Restricted Quadtree Triangulation. *IEEE. Visualization'98*.

[5] Saha, S. 2000. Image Compression – from DCT to Wavelets: A Review. The ACM Crossroads Student magazine.

[6] Open GIS Consortium. www.opengis.org

[7] GeoMedia Web map. www.intergraph.com/software/geo_map/geo_web.asp

[8] MapXtreme. www.mapinfo.com

[9] MapGuide. www.mapguide.com

[10] Paiva, J. 1998. Topological Equivalence and Similarity in Multi-representation Geographic Databases. *PhD Thesis*. University of Maine.

[11] Kreveld, M. and J. Nievergelt editors. 2000. *Algorithmic Foundations of Geographic Information Systems*. Springer, Berlin.

[12] Jpeg2000 specification. www.jpeg.org

[13] TerraExplorer. Skyline Inc. http://www.skylinesoft.com

[14] Sports-3D. GeoNova Inc. http://www.sports-3d.com/