[O2-GI]

[Page F5.1 - F5.7]

Seventh International Conference
Remote Sensing for Marine and Coastal Environments

Miami, Florida 20-22 May 2002

MIAMI®

Seventh International Conference on Remote Sensing for Marine and Coastal Environments

Technical Program

20-22 May 2002

Miami, Florida, USA

Sixth International Airborne Remote Sensing Conference Exhibition

Session F: Information Sharing

F-1 BAYWATCH: Real-time Physical Measurements Program: Atchafalaya-Vermilion Bay Region, Louisiana, USA
N.D. Walker, S. Dartez, X.P. Zhang, A. Babin, A.S. Haag, R. Arnone, C. Hulbert, S. Myint, A. Maier, Q. Dortch, and N. Rabalais

F-3 Business-Government Alliance: Building New Relationships
E. Heldewier, M.L. Crane, and J. Gray

F-4 Database on Salinity Patterns in Florida Bay
N. Rishe, M. Chekmasov, M. Chekmasova, D. Hernandez, A. Roque, N. Terekhova, and A. Zhyzhkevych

F-5 GIS Inter-Protocol Bridge: GIS Vector and Raster Imagery Inter-Process Wrapper 7 Page
N. Rishe, S.-C. Chen, A. Mendoza, A. Selivonenko, O. Dyganova, and S. Graham Paper.

F-6 Applying TerraFly for Vulnerability Assessment of Mobile Home Communities
M. Gutierrez, N. Rishe, O. Dyganova, A. Selivonenko, G. Rocha, S. Graham, and R. Alvarez

F-7 Accessing Remote Sensing Information Through Standard Object Interfaces
S. Wong

F-10 Online Data Portals: Organizing Ocean Data for the Scientific Community
M. Holland

Session G: Sensors II

G-3 Initial Results From a Test of the NASA Experimental Advanced Airborne Research Lidar (EAARL) for the Study of Coral Reef Ecosystems
J.C. Brock and C.W. Wright

G-5 Laser Sensing Technologies in Studies of Marine and Coastal Environment
S. Babichenko, L. Poryvkina, and A. Dudelzak

G-6 Derivation of Terrain Models in Coastal Areas Using Airborne LIDAR Data
K. Zhang and D. Whitman

G-8 Gridding NASA ATM Coastal LIDAR Data
A. Nayegandhi and J.C. Brock

G-9 Detecting Ocean Water Using NASA ATM Solar Backscatter and Topographic LIDAR Data
A. Nayegandhi and J.C. Brock

GIS INTER-PROTOCOL BRIDGE: GIS VECTOR AND RASTER IMAGERY INTER-PROCESS WRAPPER*

Naphtali D. Rishe, Shu-Ching Chen, Alejandro Mendoza, Andriy Selivonenko, Oksana Dyganova, Scott Graham

High Performance Database Research Center
NASA Regional Applications Center
Florida International University
Miami, FL 33199, USA

ABSTRACT

The GIS inter-protocol wrapper is an image processing system that dynamically retrieves and merges remote sensing data obtained from the web's data sources. Processing of spatial imagery such as satellite, aerial, and topographic maps of the earth can be accessed online via Internet/Intranet protocols. The service and communication framework can be loaded into the client side via a web browser. The browser's Java run time environment obtains and hosts the components needed to execute the image processing services. The application gives the end-user the ability to transform the semantics of metadata to the desired imagery format in various resolutions, encoding processes such as JPEG or BMP, and ease of distribution through various protocols and/or computer media. Integrating the services with Internet browsers provides intuitive spatial and text interface to the data. Users need no special hardware, software, or knowledge to locate and browse imagery. Any PC, MAC, or UNIX workstation with a Java enabled web browser can access the system.

1.0 INTRODUCTION

The World Wide Web is an enormous repository and a popular medium for interacting with information. Web based Geographical Information Systems (GIS) are becoming an active area of research in the GIS community. Web-based GIS can make data retrieval and the analysis of GIS data available to users worldwide. Many GIS companies have become web-focused increasing the amount of spatial information that is available on line. The web is inherently a graphical environment and images of neighborhoods are recognizable and interesting to many around the world. Comprehensive information on and easy access to huge amounts of heterogeneous data are important prerequisites for GIS. The value of GIS stems from their ability to integrate great quantities of information about the environment and to provide a powerful repertoire of analytical tools to explore this data.

*Presented at the Seventh International Conference Remote Sensing for Marine and Coastal Environments, Miami, FL, 20-22 May, 2002

*This research was supported in part by NASA (under grants NAG5-9478, NAGW-4080, NAG5-5095, NAS5-97222, and NAG5-6830), NSF(CDA-9711582, IRI-9409661, HRD-9707076, and ANI-9876409), ONR (N00014-99-1-0952), and the Florida Space Grant Consortium.

2.0 SYSTEM ARCHITECTURE

The GIS bridge is a multi-threaded Java server providing access to a high number of simultaneous users via standard web browsers. The browser's capability to interpret java class byte code allows the server to execute natively on the client's computer. This framework allows embedding of complex java components on demand into the client-side. Internet capabilities allow the system to access numerous data sets without the need for any special GIS software (Rishe, 2001b)

The GIS inter-protocol wrapper bridge consists of a three-tier architecture.

Tier 1:
The *Client* is a graphical web browser or other hardware/software system that supports HTTP protocols. The application was built and tested with Netscape Navigator and Internet Explorer on Windows and UNIX platforms.

Tier 2:
The *Application Logic* is a web server application that responds to HTTP requests submitted by clients by interacting with the Tier 3 *Web Data Source* and applying logic to the results returned.

Tier 3:
The *Web Data Source* data base system contains all the imagery and meta-data required by the Application Logic tier.

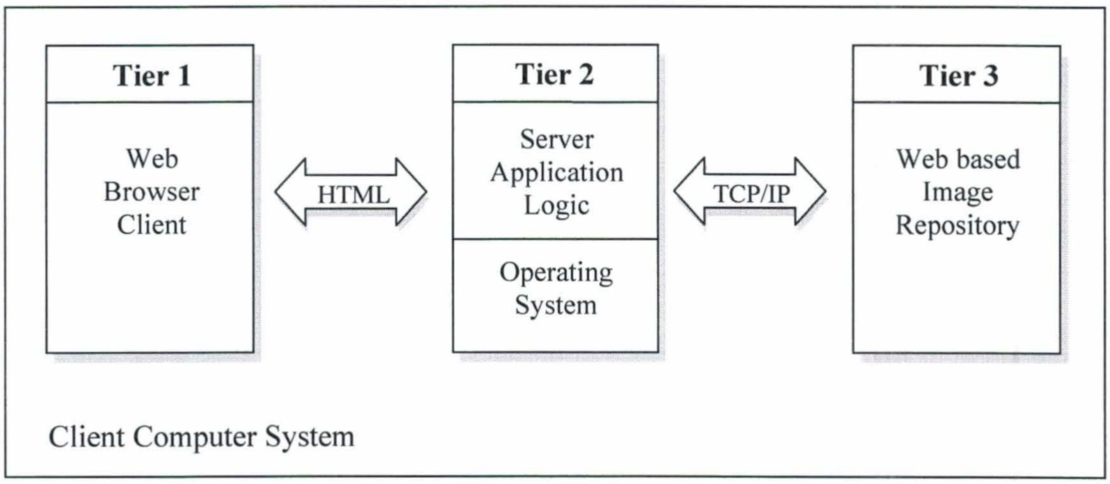

Figure 1: System Architecture

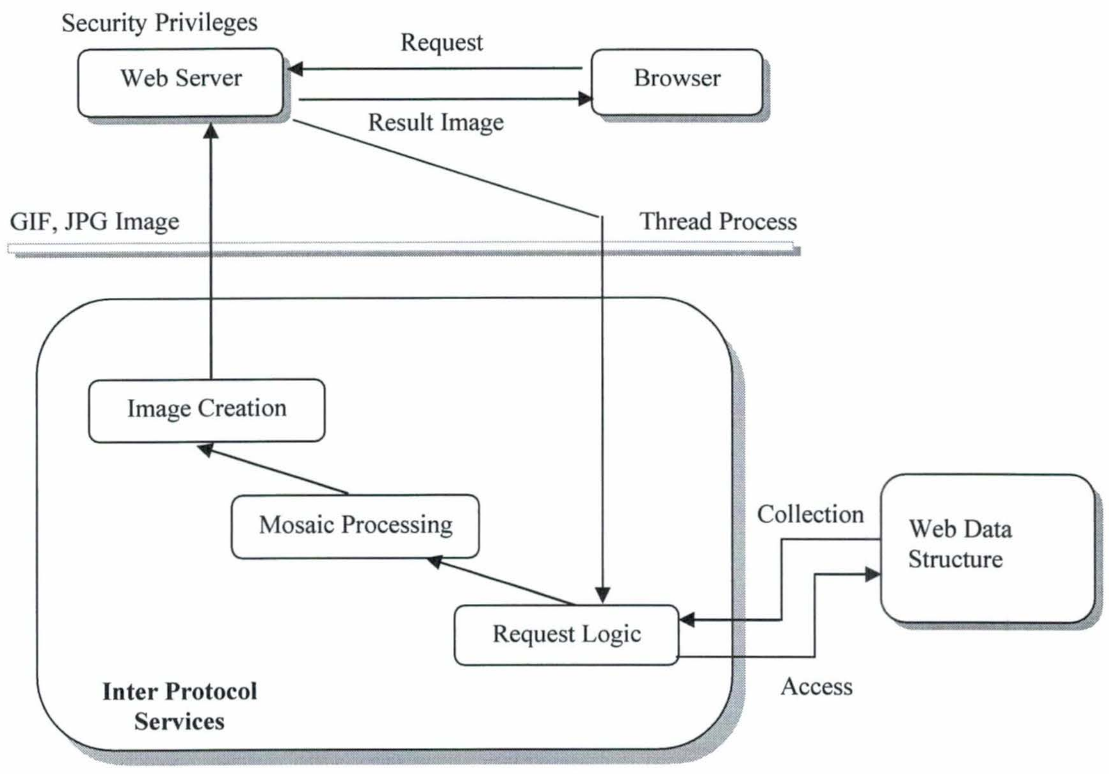

Figure2: System Logic

3.0 GENERAL APPROACH

The application is a wrapper that translates queries and meta-data from one data model to another from different heterogeneous data sources. This functionality is provided around each individual meta-data repository. The wrapper converts queries into one or more queries understandable by the underlying data repository and transforms the result into a format understood by the client. The application provides a simplified interface to encapsulate diverse data sources so that they all present a common interface (Rishe, 2001a) (Chen, 2000).

The application provides the client with the ability to dynamically retrieve and merge remote sensing data. Internet connectivity allows the system to access numerous data sets without the need of any special GIS software. Designed for users of all levels and unlike other geographic system, the inner-wrapper runs via standard Web browsers, with no need to download software or data. The system is a Java applet that executes within the runtime environment of the browser once the user reaches the website. The browser's java enabled environment relieves the application from problems relating to platform dependency, thus providing a greater flexibility within multiple environments. The execution process begins by checking that the client's browser contains the latest version of the application. If it does not, the web site provides the browser with a compressed distribution package containing the java byte code needed in order to execute the server services. The site prompts the user with a security certificate informing them of additional security privileges required. The application needs to be given security privileges that are normally restricted by the applet "sandbox" security model such as: network resources to communicate with the web data sources and access to the clients' local disk in order to retrieve and save images. If the user does not accept the security certificate the application cannot establish its services and the process is aborted. The system also checks the client's browser setting. The browser's setting should be set to allow the execution of Java applets. If the java virtual machine is disabled, the system will prompt the user to enable it.

When the security privileges are granted, the site starts dispensing the java byte code package to the client's browser. Once the dispensing process has been completed the client is informed that the application needs additional information to optimize the available services. The application prompts the user to provide the port number where the server will be executing and the name of the proxy if one is required. Asking the user for a port provides the user more leeway in setting up the server. If binding of the port entered by the user fails, the application will prompt the user for another by suggesting the next available port from a scan of ports automatically performed by the application. The application is capable of scanning ports on the client's computer until it finds one that is available and informing the user of the port scan outcome. Once the startup procedure has completed, the application is ready to accept incoming connections.

The application is a multi-threaded java server running within the environment of the web browser. The application is a wrapper capable of dispensing images from repositories on the Internet. The application can interact directly with the user through the browser interface or can programmatically interact with other applications requiring image processing. When the server receives a request it creates a thread-process for each incoming request. This thread handles the imaging requirements of the client. The thread gets a handle of the port the client is mapped to and establishes a communication channel with the client. Once the communication channel has been established the client can communicate via HTTP protocol and request the needed services. At the same time, the server continues to intercept any incoming requests and creates an individual thread per request handle. The HTTP request contains commands that the application logic understands in order to initiate the services. There are several commands required in order for the image services to proceed. The server provides several services and each service requires a fixed number of parameters. Though the order of the parameters does not matter, the number of parameters must be correct in order for the services to begin. Once the application has understood the request, the dispensing process begins. The client specifies the coordinates, imagery type, and dimensions. The imagery is based on the Universal Transverse Mercator (UTM) projection using the North American Datum (NAD) ellipsoid. UTM is a projection system that divides the earth into 60 wedges shaped zones number 1 thru 60 beginning with the International Date Line. Each zone is 6 degrees wide and goes from the equator to the poles. UTM grid coordinates are specified as zone number, then meters from the equator and from the zone meridian (UTM grid units can also be in inches, feet, meters or kilometers). The system supports several resolutions depending on the data being utilized. The system can modify available imagery of power of 2 from 1/1024 meter per pixel through 16384 meters per pixel range. The user specifies the coordinates for the data that needs to be retrieved using UTM coordinates and can select from satellite imagery, maps, and aerial photography.

The application's data fetching logic contacts the data repositories' web site to attain the needed imagery. The application fetches images from repositories that dispense their images in small sizes such as 200X200 pixels. These sizes are ideal since they contain a small amount of data and their transfer rate over the network is fairly fast even when high-speed connections are not available. The latency of this size of image and the bandwidth requirements do not hinder system resources on the client's computer. The application provides the user with the ability to create images of any size. These images are created from smaller images or tiles. Once the user specifies the size of the image, the application logic begins to build the image by mosaic the number of images required to build the one requested by the client. Since the source images are small the application spawns multiples threads whose jobs are to retrieve each individual tile. This process allows the application to request a large number of images at the same time and concurrently wait for them to be received into the client's computer. Since these images are obtained from the web there are problems that can arise from the data source provider. The application must be able to track the images being retrieved and also guarantee that the images obtained have all the correct data. The application has a monitoring mechanism that allows it to track and check each individual image. If the image requested did not transfer correctly, it resends the request to the data provider requesting the image to be fetched. If the data context of the image is not corrupted, the application temporarily buffers the image in order to mosaic it with the other incoming images. The application also has a fail check mechanism that allows it to deal with any unusual delays encounter through the fetching process. The application waits a fixed amount of time for the request to finalize. Since the fetching of images has variable time requirements the application must be able to handle the wide variety of time discrepancies encounter throughout the process. If the data provider does not deliver the data within this time the application resends the request several times. If no response is received, the application creates a black tile for the particular missing image. This informs the user visually of the fact that an error was encounter during the transmission and that the outcome of the requested image is not correctly formatted.

When the request logic determines that the incoming images have passed all of the preventive measures, they are then transferred to the image assembly module. This modules creates a mosaic from the dynamically retrieved imagery acquired from the web's data repositories. During image acquisition the system aggregates the images and process the images to obtain the final Ad-hoc image request. The construction of the images requires an efficient memory management algorithm to dispense multiple images concurrently. Therefore, memory optimization is of high priority. The web browser-hosting environment natively provides a limited amount of memory to applications. The application must be efficient in the use of memory; otherwise the maximum size of the image it can produce is limited to the physical memory available on the client's computer. The application keeps track of the images being used and once the images are no longer needed the resources are given back to the system. When the tiling of the images is concluded and the required image has been produced the application then needs to convert the image to one of the encoding format it provides such as JPEG or BMP.

The encoding module takes the newly created native java image object and applies one of the encoding methods provided by the application. The user chooses the encoding format and the application dynamically loads the require image encoder. All the encoders provided by the application have a handle to the outputstream of the client's socket. This stream is use to deliver the final compressed image. The output stream can be directed to the client's local hard drive if the client whishes to do so or it can be redirected to a common TCP/IP port. During the compression phase the application can concurrently deliver the data to the client even though the image has not been totally compressed. The application buffers a small portion of data and applies the compression to it. This approach allows the clients to start receiving the data stream quickly without having to wait for the entire image to be compressed.

The system might need to transfer extremely large amount of data, which may cause significant network delay. To address this issue, progressive transmission of spatial data over the web has been applied. This is particularly useful when the client is using a slow network connection or when the datasets are very large. The system can provide data in raster datasets. Subsampling of the pixels of the image is sent to the client so the client can start working without having to wait for the whole image to be downloaded. The full version of the original images is progressively treated by adding new pixels to the transmitted image on the client side.

Once the entire image has been fully compressed, the application releases all the system resources and closes and terminates the open socket. Terminating the socket to the client guarantees the client will not continue to wait unnecessarily for any more data to be delivered. Once all the dispensing process has been fully completed, the thread handling the request is terminated. The termination of the thread releases all the resources allocated to the system. At the same time the application can continue executing other incoming requests. The application keeps an open socket to the client while the distribution process is in progress. Once all the housekeeping has been executed the application closes the socket. The client's socket is kept open and delivery of data continues until the full image has been delivered.

4.0 SECURITY ISSUES

At first glance, embedding of Java components into a browser seems straightforward and simple. However, if the components to be embedded violate the Java "sandbox" permission model, additional settings in the web browser's environment allows the applet to acquire the necessary system resources. Along with the use of an applet, come issues relating to security and the digital signing of the applet to make a secure I/O operation.

The permission model for trust-based security supports a useful set of parameterized and non-parameterized permissions that can be individually granted or denied for a particular zone. The Internet Explorer security model can be configured to three predefined permission sets, called High, Medium, and Low. These represent the most restrictive to the least restrictive set of permission, respectively.

The system is deployed as an applet to attain purely distributed functionality. As such the system faces many permissions restrictions that need to be granted according to Internet Explorer and Netscape security model. The standard Java sandbox permissions are as follows:

- Thread access in the current execution context.
- Network connections to the applet host.
- Creation of top-level windows with warning banners.
- Reflection to classes from the same loader.
- Access to base system properties.

The system is a multi-threaded application that makes remote network connections with not only the applet host but other computers as well. Therefore, the system violates the first two sets of permissions that correspond to the Java sandbox structure. To circumvent these security constraints the distribution and packaging mechanism contains digital signatures requesting the needed permissions in order for the application to function correctly.

6.0 CONCLUSION

A distributed GIS component approach was proposed. The capability of Java-enabled browsers to execute Web services was demonstrated. The proposed approach of lightweight Java applet GIS components running in a regular Internet browser brings a new perspective of client-server interaction to the GIS area, allowing better exploitation of available CPU and network resources. The lightweight Java component GIS Web services model provides a variety of GIS computation features to generic PCs of any GIS user.

7.0 REFERENCES

S.C. Chen, N. Rishe, X. Wang, and A. Weiss. "A High-Performance Web-Based System Design for Spatial Data Access," In *Eighth Symposium of ACM GIS*, Washington D.C., pp. 33-38, November 10-11, 2000.

N. Rishe, T. Berk, O. Dyganova, A. Selivonenko, S. Graham, and D. Mendez. "Crawling Distributed Operating System within Multiple Heterogeneous Operating Systems of Internet-connected Hosts and Devices." In *Proceedings of the 5th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2001),* Orlando, FL., Vol I, pp. 71-76. July 22-25, 2001a.

N. Rishe, O. Dyganova, A. Selivonenko, M. Chekmasov, and A. Mendoza. "MedFerret: Client-Based Semantic Query Integrator." In *Proceedings of the 5th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2001),* Orlando, FL., Vol I, pp. 66-70. July 22-25, 2001b.