A PLATFORM FOR RAPID DEPLOYMENT OF MOBILE ASSET MANAGEMENT SYSTEMS

Mr. Suen Lee and Prof. Yang Gao
Department of Geomatics Engineering
The University of Calgary
Calgary, Alberta, Canada T2N 1N4
sulee@ucalgary.ca; gao@geomatics.ucalgary.ca

KEY WORDS: GIS, Internet/Web, Management, Data Mining, Web based, Tracking

ABSTRACT:

In recent years, the convergence of location, information management and communication technologies have created an emerging market known as location-based service (LBS). LBS is a critical enabling technology using location as a filter to extract relevant information to provide value-added services. Mobile Asset Management System (MAMS) is one such service and has been rapidly gaining attention from corporations and individuals. A MAMS offers timely and relevant information necessary for informed decisions on efficient asset management, increasing productivity, profitability, safety and security. Despite there being a diverse array of potential applications, all MAMS share common elements such as data collection from remote assets, wireless communication for transmitting data from the assets to a central office back-end for storage, and software application to provide services to interested users. If these elements are recreated for every new MAMS, as is generally the case today, then significant time and resources will be wasted through duplication. Trying to tie the heterogeneous components into a MAMS has been a challenge for LBS developers. To overcome these obstacles, technologies that provide the common elements and fundamental functions have been investigated and developed at The University of Calgary. Heterogeneous components such as sensors, wireless networks and databases have been integrated into a single Development Platform which can become a foundation and is an innovative solution to numerous and diverse MAMS system development and for other LBS applications. By featuring an object-oriented, extensible and modular architecture, developers can choose the functions from the platform to use in their applications, extend or customize other functions and add their own specialized software applications when necessary. Technical details of the platform will be described along with field test results in applications to vehicle tracking for safety and security monitoring.

1. INTRODUCTION

1.1 Background

Advances in widespread, robust and inexpensive location and wireless communication technologies have resulted in an explosion of activities in the field of Location Based Services (LBS). Timely and relevant information enables informed decision-making and offers improvements for productivity, safety and security. One particular form of LBS, known as Mobile Asset Management System (MAMS) has garnered significant interest from corporations desiring more efficient methods in managing their asset fleets. Other promising applications include personal vehicle monitoring and security systems. To be an effective tool, a MAMS must be capable of aiding or performing operational decision-making. It must be able to differentiate between dissimilar assets, seamlessly handle spatial and non-spatial data. It must be capable of collecting and storing information from the field and process the data to and from numerous databases for data retrieval, editing, analysis, presentation and decision-making functions.

The impetus for LBS came out of the demands of the United State's Federal Communications Commission (FCC) for cellular operators to be able to provide the position of any cellular devices operating on their network to public emergency services, accurate to within 125 meters, as part of their Enhanced 911 program [Prasad, 2001][FCC, 2003]. Technologies developed for accurate positioning via cellular network or phone to provide emergency locations can also be used for new and innovative applications. Devices integrating wireless and positioning capabilities could be attached to vehicles and assets for monitoring. Mining operations utilize numerous and varied assets spread over a wide work area in their operations and would greatly benefit from the deployment of an effective MAMS into their operation. They could also be used by consumers to determine nearby points of interests or receive relevant information, using location-aware cellular phones or other wireless devices. For these reasons, along with expectations of rapid growth [Prasad, 2001], LBS has received significant attention from industry, users and providers.

1.2 Development Platform

In this paper, a location aware Development Platform in support of MAMS and other LBS applications will be presented. This Platform helps augment MAMS development by providing critical functionality required in a MAMS type applications, allowing developers to reuse these functionality and maximize their time and resources on the unique aspects of their applications. The intent is to have a platform that can become an engine for LBS development. While MAMS is the primary application for the Platform, it will remain flexible enough so that it could also be extended to other LBS activities in the future. This Platform will offer important low-level functions that developers can utilize as an important foundation for their enterprise MAMS applications and feature an open architecture, so that it can be utilized by third parties. A system developed with the Platform will also be presented along with field and simulated testing results.

2. JAVA, WIRELESS LOCATION AND COMMUNICATION TECHNOLOGY

The Platform is created using Java and leverages object-oriented principles to realize advantages in maintenance and ease of integration for third-party developers. It offers two-way communication and control capabilities in conjunction with a widespread cellular network and remote intelligent sensors. Users have access to assets through the Internet.

2.1 Java

Java is a software development platform that was released in 1995 and offered the following benefits:

- The ability to write code once and have it run on multiple hardware and software platforms with minimal effort
- Designed from the ground up on object-oriented principals
- Better inherent security and safer memory management compared to the C/C++ programming languages.
- Allow full-fledged applications to run completely within an Internet browser
- Developing server-side applications for dynamic and interactive Internet content
- Usable on a wide range of consumer devices

Java has especially become popular in the creation, serving and presentation of content through the Internet and for Internet enterprise applications. Three Java technologies, namely Applets, Servlets and Java ServerPages, have features that are especially useful for the serving and presentation of asset data to users in a MAMS and are used in the Platform.

2.1.1 Java Applets

An Applet is a Java application designed specifically to work within the confines of an Internet browser. An Applet is used whenever complex or graphical information needs to be presented as it has all the capabilities of a Java application, such as a graphical user interface. However, as the Applet runs on the remote user's computer, it is a client-side process and is therefore barred by Java security practice from accessing resources on the server directly. To get access to these resources, some application must be available on the server that can respond to the Applet's request and provide it with the resources it needs. For this task, the Servlet is used.

2.1.2 Java Servlets and ServerPages

A Servlet is also a Java application, but one that resides on a server and listens for requests from the outside. Potential requests could be initiated by Applets. A Java Server Engine or a fully Java Servlet enabled web server is needed to properly operate Servlets. The Servlet is designed to primarily handle multiple short-lived tasks simultaneously so it utilizes child threads to handle new requests. Creating only a new child thread, which can share global memory with other child threads, is much faster and consumes less computer resources than previous server-side applications. Child threads also have local variables that are only visible within a thread context, so that data meant for one client does not go to another client and do not interfere with each other in any way.

Java ServerPages (JSP) is Java's answer for server-side scripting. A JSP file is simply a normal web page containing HTML code intermixed with Java source code and has an extension of JSP to identify it. The JSPs, like Servlets, are processed by a Java Server Engine and is executed when it is called by a user. After execution, the results are outputted as HTML code in place of the Java source code in the JSP and sent to the user for display. Because the JSP is a server-side application, it also has access to all the resources that are available on the server-side. The advantage that JSP has over other server-side scripting methods is that JSP is a true programming language and allows the full use of the extensive Java class libraries. Any new development in Java, even if it has nothing to do with Internet development, is automatically available to JSPs and Servlets. This greatly enhances the reuse of software and code. Another advantage is that JSPs are precompiled and this benefits response time and performance. JSPs are primarily designed to handle simple dynamic text content, such as querying databases and returning the results in a tabular form. Servlets can transfer any form of data to a requester, including complex class objects containing desired data. The class is serialized into a binary stream and sent to the user application, which casts the binary data back into the original class object and be directly used in the application.

2.2 Intelligent Sensors for Wireless Location and Asset Data Acquisition

Remote sensors are important tool in MAMS; not only do they provide the location of the mobile assets but they can also acquire important data about the assets. Unlike human data collectors, they are always at the asset and ready to acquire data at any time and condition. There are several such sensors available from different manufactures designed for mobile asset management and could be used by the Platform. However, in this paper, only the Asset-Link sensors from CSI Wireless will be discussed. Asset-Link sensors has been used with a prototype system, which will be discussed in Section 4, to combine cellular communication, GPS technologies and greater "intelligence" to create a fully integrated yet low-cost sensor that is highly suitable for mobile asset management.

Asset-Link sensors are an intelligent and fully autonomous location and data acquisition system developed specifically for mobile asset management applications. Asset-Link sensors utilize an onboard high-sensitivity GPS receiver chip to provide location data and data transmission through Aeris MicroBurst. Not only does the Asset-Link acquire data, it can also be pre-programmed to perform operations with the asset automatically or when directed by a command from the asset owner. Because of its small size, the Asset-Link sensors can be discretely hidden into vehicles, making it an effective alarm and monitoring system. Its low power consumption ensures that it does not detrimentally affect the performance or usability of a monitored asset, even if the asset is put into storage or is inactive for weeks or months. [CSI, 2003]

2.3 Wireless Communication

Mobile location and data acquisition sensors from different manufactures may use different wireless carriers in their products. The Platform is designed to handle current and future wireless technologies as they become available and allow users of the Platform to take advantage of the technologies with little effort. In the following, Aeris MicroBurst will be described since it has been integrated with the Asset-Link sensors.

Aeris MicroBurst is a wireless data communication system that utilizes the analog portion of existing cellular networks, which gives excellent coverage. MicroBurst is considered an ideal solution for applications requiring widespread availability and reliability but only have small data bandwidth requirements. Many types of MAMS fall under this category of applications. MicroBurst utilizes the Forward Control Channels (FOCC) and the Reverse Control Channels (RECC) of an analog cellular network. Together, they enable two-way communication and control with assets.

FOCC messages are transmitted from the cellular switch to the MicroBurst-enabled devices. FOCC messages, otherwise known as Pages, are meant to trigger responses from the device or to get the device or asset to perform a desired operation. Each MicroBurst-enabled device can have up to 10 Mobile Identification Numbers (MIN) which can be used to trigger a different event or operation on the device. FOCC messages are initiated by the device's owner or interested users, who send the page in a MicroBurst specific format to Aeris's central hub. Aeris then sends out the page to the desired device. RECC messages are transmitted from the MicroBurst-enabled devices onboard assets back to the cellular switch. These RECC messages are then sent to the Aeris central hub, which finally relays the messages through the Internet back to the asset owners. Each message packet's data is encoded into a number string. To obtain data in real-time, the asset's owner must be continuously connected to the central hub via the Internet. Otherwise, Aeris will store the data until the owner connects with it. The maximum length of a number string is only 32 digits. However, by using each number efficiently, it is still possible to send useful amounts of data with a MicroBurst message packet. Multiple messages can also be used to increase the amount of unique data that can be sent from the asset. [Aeris.net, 2003]

3. PLATFORM ARCHITECTURE

The Platform is split into several components based on logical and physical boundaries. This follows object-oriented principles of grouping similar functionality together and also offers performance and security benefits.

3.1 Client-Server System Architecture

A common paradigm found in computing is the Client-Server (CS) architecture. The CS architecture utilizes servers that contain data resources and may perform some or all of the data processing function. The Client's primary purpose is to present data and processing results from the Server to users but may also process data depending on the CS design used. The CS design is split into additional subcategories, depending on the fatness or thinness of the client layer. The thin client is a lightweight application that can only display data and processing results that are retrieved from the server. Being lightweight, the client layer application requires less memory, storage and processing power, therefore enabling the client application to run on smaller and more mobile computing devices, like PDAs. On the other side of the spectrum, there is the fat or thick client where most processing functions are carried out at the client side and only the data is retrieved from the server. Between these two extremes, there are also hybrid clients where the processing workload is split between the server and client. The hybrid scheme was utilized in the Platform for its advantages in performance and control of remote access to the asset data [Peng and Tsou, pg. 13].

3.2 Platform Modules

The Platform features numerous useful functions for a MAMS application. These functions have been separated into data, application logic and presentation layers and placed into independent modules. These modules can be used fully, partially or customized depending on need. Much of the configuration for the modules can be done in configuration files or databases, thus allowing for significant customization without a need to change the code. While these modules are logically separated, they can physically operate on the same server or be distributed across different servers. This can realize improvements for scalability and performance while still working seamlessly with each other.

3.2.1 Communication Module

The Communication Module contains the functionality to make and maintain the connections to assets. The only configuration required if using Microburst-enabled Asset-Link sensors is to input settings regarding the IP address of the Aeris servers and account settings to login to Aeris's servers. The Communication Module connects to the servers at Aeris's central hub using the Internet with network socket connections. Two connections are made; one connection is to the Aeris Data (DS) Server; this server relays messages sent from a remote Microburst-enabled sensor back to the Office. The other connection is with the Aeris Page (AS) Server; the AS Server relays pages sent by users to the sensors.

The Communication Module is also responsible for maintaining a continuous connection to the Aeris servers. These servers send regular "pings" to any connection that is made to it, to ensure that the connection is still valid. If no acknowledgement message is sent back, then the connection will be terminated. As well, other messages that are initiated by the Aeris server usually require an acknowledgment. The Communication Module has the necessary logic to recognize the message type and to return a suitable acknowledgement back to the servers to ensure the connection is not terminated. For messages that are sent from a sensor onboard an asset, the raw data is extracted and passed onto the Data Module for further processing.

There are a large number of messages that are used on the Aeris MicroBurst service but all share a common format and binary encoding method. Each message has a header and limiter plus the actual message contents. The message contents include the metadata, such as the message length, message identification number to distinguish its message type and finally, the message body itself. As the messages share a similar structure, then it is natural to implement the different Aeris message types as objects in the Platform by implementing Java classes to represent each message type.

3.2.2 Data Module

The Data Module is responsible for serving data from geospatial sources, such as vector data in ESRI Shape form or raster image data, to local or remote users through the Internet. It features a highly optimized code base and supports simultaneous multi-user access. This module is also responsible for receiving the decoded raw asset data from the Communication Module, to store into databases and serve the asset data to users. It is designed primarily to serve requests from the Web Client, though any application that understands the common protocol and the module's data structure can connect to the Data Module and receive data. It can be configured without need to modify the source code, enabling rapid setup of new MAMS applications and projects. The Data Module is expected to work in a one-to-many environment. That is to say, in an environment where multiple users are expected to access the module simultaneously. The Java Servlet technology fits this requirement well, given its seamless handling of multi-user access. It also makes it possible for only one instance of the Data Module to handle all the clients, thus minimizing its memory footprint.

The Data Module works in conjunction with the Communication Module. The Communication Module obtains from the received Aeris message, the raw number string and hands it to the Data Module. The Data Module then decodes the number string into meaningful information, based on the configuration set in the remote sensor and stores the information into relational databases. There is a separate Database component within the Data Module that is used to handle the database management aspects. This is necessary as the database is among the most variable aspect in a MAMS as any changes in assets, or collected asset information would have to be reflected in the database. Communication between the Module and the databases are done through a standard public interface and queries to help abstract the structure and format of the actual database. This use of encapsulation helps minimizes the effect that any changes to the asset data collection process would have to the whole Data Module. The Database component is where the number string decoding occurs and where the necessary queries and update functions used to interact with the actual databases are held. The database product currently used is Microsoft Access, but all transactions with databases involve the standard ODBC protocol, which will allow different database software packages to be used in place of Microsoft Access if necessary.

When providing vector data, the Data Module first opens the ESRI Shape file to decode into an array of separate vector features. This array is then broken apart into a **M**ulti-**L**ayer **S**torage **S**cheme (MLSS), which is a key optimization for reducing the search time for a request. This storage scheme works by having several levels that cover the entire vector map area; each level contains geospatial features of varying size, with no feature being duplicated between the levels. Furthermore, each level partitions the map area into a certain cell size, with lower levels having smaller cell dimensions and a greater number of cells as compared to higher levels.

The topmost level has only one cell that contains the entire map. As each vector feature is decoded from the ESRI Shape file, its boundary coordinates is then added into the smallest possible cell that they can fit into. By sorting the vector features into different cells, only a fraction of the entire vector data set needs to be searched for the majority of client requests and thereby greatly improve the responsiveness of the system. The MLSS is fully customizable; the number of levels, the number of cell partitions in the horizontal and vertical can be adjusted to best match the vector feature density characteristics of any particular dataset. As the Data Module is built upon a Servlet foundation, the vector data is stored into global memory, so that threads created to service requests from multiple users share the vector data and thereby reduce memory usage. The vector data is persistent in memory, even after client threads have been destroyed once requests have been serviced. This ensures that the time and processor consuming procedure of decoding the ESRI Shape file only occurs once during the Data Module's initialization phase.

The Data Module also provides raster data, most commonly with MapPoint Web Service. MapPoint provides global spatial data and extensive street maps for North America, Europe and many cities in South America. This large amount of spatial information from one source is essential for providing services to companies with assets spread over entire regions or countries.

Requests for data are received by the Data Module through the Servlet communication method. All forms of user request follow a standard protocol, which indicate the type of request and other associated information required by the server to perform the request. The server then performs the request and then packages the requested data into a single message and sends it back to the user. After receiving a request, the Servlet performs the operations necessary to accomplish the request and sends the results back to the requestor [Lee and Gao, 2002].

Configuration of the Data Module can be done through two methods. The primary method is through adding parameters into a standard XML based configuration file used for each Servlet. The second supported method is to have the XML file provide the location of a separate configuration file, which would be parsed by the Data Module. These configuration methods provide the Data Module with important parameters such as the location of the vector data, the ODBC name of associated databases and the coordinates of the desired map extent. In this way, different projects can be created using the same template Data Module and without the need to modify the source code or to write any additional code.

3.2.3 Application Module

The Application Module consists of a Server component and also the Web Client which helps users access data and tools. Whenever a user connects to the Application Module through a browser, the Web Client, which is the other component in this module, is downloaded and proceeds to run on the user's computer. By centralizing the storage and distribution of the Web Client, benefits can be realized through reduced support and administrative costs. Administrators need only to apply updates and bug fixes to the Application Module code base and users can be confident that they are using the latest version of the Web Client whenever they connect to the Module.

The basis of the Web Client is the Java Applet, which allows for the creation of a software application that offers a graphical user interface that would easy for users to grasp. It also enables the application to run on multiple platforms without any need to modify the source code. The Web Client includes a Toolbar component, a Statusbar component and a Map component plus a Parent Shell and is based on a Model-View-Controller scheme. The Toolbar is the Controller in the Web Client as it is responsible for receiving inputs from users and controlling the application based on the inputs. The Parent Shell acts as the Model, containing the majority of the Web Client's logic. It also links all the other components together, passing controls and notifications to the View and for communicating with Platform server-side modules. The Statusbar component is used to display text information passed to it from either the Map or Toolbar components. The Map component is used to display the geospatial data via a graphical presentation. Both the

Statusbar and Map components are part of the View, as they help present information and processing results to the user.

For the Web Client, all of the user accessible tools appear to be part of the Toolbar, at least as seen from the user's point of view. However, the implementations of these functions are actually in the Parent Shell. The Toolbar is only used to activate the proper functions in the Parent Map. The tools that have been created include typical GIS-type functions for map viewing and manipulation as well as for viewing and querying spatial and non-spatial information. These functions interface with the Data Module and retrieve data from it. Another category of functions are used for asset control, which sends user-issued commands to the Relay Server. The type of control functions available vary depending on the asset and the configuration of the onboard remote sensor but typically include positioning, tracking and safety and security related controls. The current configuration of the remote sensor and asset combination is part of the information stored into the database dedicated to assets. This database is queried by the Web Client to determine which command functions, if any, to show to a user when a certain asset is selected. In addition to asset information that can be queried using the Web Client, spatial information regarding the current address of assets or nearby streets can also be queried using the geo-referencing tools available in the Web Client. The nearest vector feature to a mouse click is determined using a nearest vector feature search algorithm. Each vector feature has metadata that includes the necessary identification numbers that allow the Parent Shell to request extended information regarding the feature.

While the Web Client can run on numerous platforms, there are certain requirements that must be met in order for a computer to properly run the Web Client. The primary requirement is that the client computer must have a recent Java Virtual Machine (JVM) available. A JVM is necessary in order to run the Java based Web Client and to provide the built-in Java classes that are used by the application, so that the downloaded application can be as small as possible. The JVM is also necessary to run other Java applications. Fortunately, JVMs are offered by numerous providers for multiple platforms and operating systems. The version developed by Sun is one of the most popular and featured-filled, which is not surprising since Java was developed by Sun. It is freely available for download from Sun's website and is known as the Java Run-Time Environment (JRE).

When a user first accesses the Application Module, the Web Client is downloaded onto the user's computer, compiled and executed within the Internet browser environment. Once the initialization processes are complete, the Web Client is in a blank state with no data or parameters stored in the Web Client's memory. It must then request an initial set of parameters from the Application Module back at the Office server. This first set of parameters provides important details regarding the map and data that would be needed for the rest of the client session. It provides the access addresses for the Data and Relay Server modules, the coordinate extent of the selected map area as well as information regarding other map data sources that are available from the Data Modules.

When a vector data message is received from the Data Module, the Parent Shell passes the message to the Map component. The contents of a vector data message includes the coordinate extent of the new visible area, the vector feature type (polygons, polylines, etc.), metadata regarding each distinct vector feature's class as well as any assets that are within the visible area. The Map decodes the packed vector and asset data into an array of coordinates, converting from coordinate space to the display space using the coordinate extent of the visible area. Each vector feature is drawn onto the display using Java provided tools for drawing polygons and polylines. Information about assets can be provided in graphical form, by indicating the location of assets on the map, or in text and tabular form using JSPs to query the Data Module's database for current or historical information.

A unique ability offered by the Platform is the ability for users to remotely check on and control assets in real-time. This is possible through the integration of the Aeris MicroBurst service with CSI Asset-Link sensors. Users can issue a specific command through the Web Client to the Communication module. It passes the message onto the Aeris's AS server. Aeris then proceeds to deliver the command by paging the onboard Asset-Link sensor which then performs the desired operation and reports back the success or failure of the operation as well as any results that may have occur. An Asset-Link sensor can be preprogrammed to perform a series of operations automatically based on conditions or triggered manually through a page. Operations of use in real-world applications include requests for current positions, unlocking doors, arming and unarming alarm systems, enabling or disabling engine ignition, speed limiters and continuous tracking. The configured operation and their numbered command strings are stored into the sensor database as the configuration can be unique for different sensors and/or asset combinations. For security reasons, the actual command strings and sensor identification numbers are not shown to users.

4. TESTING RESULTS

A prototype system, named as iVCAMS3 (Internet-based Vehicle Control And Monitoring System for Safety and Security), has been developed using the development platform at The University of Calgary to monitor and control vehicles for safety and security purposes. The main screen can be seen in the following figure.

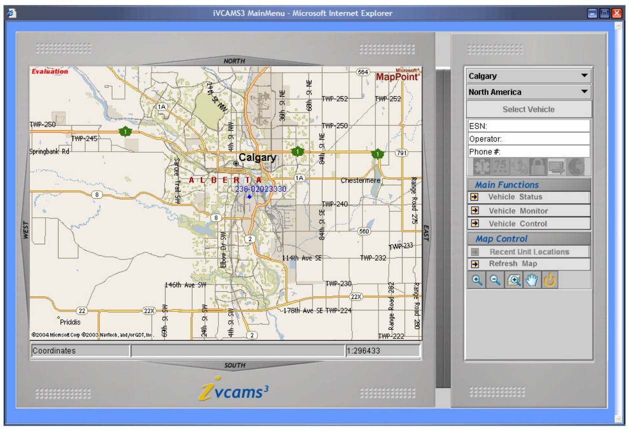

Figure 1. iVCAMS3 main screen

4.1 Field Testing

Field-testing was conducted to ensure that iVCAMS3 could successfully receive messages from Asset-Link equipped assets using the MicroBurst network and send commands back. Asset-Link sensors were placed onto cars in several cities throughout North America. During the tests, pages were issued to the

Asset-Link sensors for their locations. Their locations were sent back to the server and instantly updated into iVCAMS3 and plotted onto the map. The accuracy of the location data was confirmed with the drivers of the cars and was high. Other commands, such as remote unlocking of doors, were also tested and were successfully received by the sensors and the correct operations performed.

The total time necessary to complete all steps in paging an asset were also monitored during the course of tests. The times for 29 pages are shown in Figure 2. These pages were sent to two assets; one asset traveling between Calgary and Edmonton while the second asset was in Fort Lauderdale, Florida. The average time for roundtrip of a page was found to be 9.25s, with a maximum time of 16s, which was reached by only one of the pages. The standard deviation was two seconds.

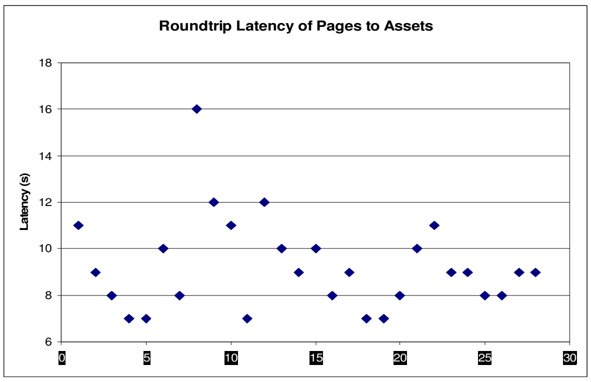

Figure 2. Page latencies

4.2 System Testing

To test the theoretical limits on the system, in terms of number of assets that can be supported, simulation testing was performed using several different computer systems acting as the Server. Simulation enabled precise control of variables and made it possible to quantify the limits of the Platform. In its development tools, Aeris provides software that simulates the Aeris AS and DS servers. This enables developers to create applications that are fully compatible with the Aeris protocol before using it with the real Aeris system. The simulated server is customizable and can be modified to send messages at any defined rate. In this way, we can simulate a different number of units by varying the frequency that messages are sent out by the simulated Aeris server. We can then determine the limitations of the system in terms of the number of units and assets that it can support, by monitoring the CPU and network bandwidth usage using PerfMon. These monitored attributes are displayed in Figure 3. The server tested consisted of an Intel Pentium III 933MHz system with 512MB of RAM and running the Microsoft Windows 2000 operating system.

With an assumption that an asset will send a message once every 30s, 64 messages per second approximates to a fleet of about 1900 assets and 131 messages per second represents about 3900 assets. From the results of Figure 3, the CPU usage for even 3900 assets is not taxing the server, as it only ranges from 20% to 25%. For 1900 assets, the CPU usage is lower as would be expected, ranging from 7% to 15%. As for network bandwidth usage, the simulation results shows that this should not be a major concern as the usage is 15KB/s and 29KB/s for 1900 and 3900 assets respectively and well within the limits of even home broadband Internet connections.

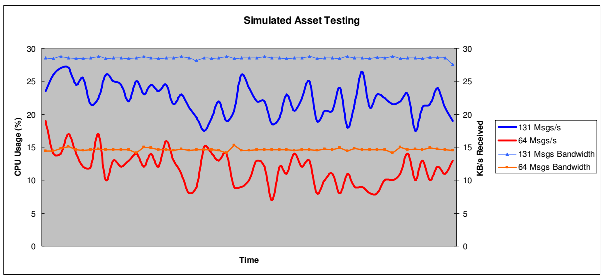

Figure 3. Simulated asset testing

5. CONCLUSION

A fully complete platform has been developed and is capable of addressing all major concerns in MAMS, including asset data acquisition, transmission and handling and analysis as well as unique capabilities for user access to asset data. It reduces the resource, time and cost needed to deploy a MAMS, thereby enhancing its benefits. It features a modular design separating the important functions of a MAMS and enables developers the flexibility to choose whether to use all or only certain capabilities of the Platform.

This Platform was used as a basis for a real-world MAMS known as iVCAMS3. This system was built based on the requirements of industrial partners and field testing was conducted to ensure that the asset could send messages and locations back to the Office and that users could issue commands to the asset. Testing was completed successfully with vehicles with Asset-Links installed in multiple cities in Canada and the USA. It was possible to monitor vehicles when they were stationary and in motion. Roundtrip latency of messages was around 10s, making it possible to have real-time monitoring and control capabilities with assets.

ACKNOWLEDGMENTS

The research was supported by a NCE GEOIDE grant.

REFERENCES

Aeris.net, 2003. “MicroBurst Technical Details”, http://www.aeris.net/aeris_web/products_microburst_technical.html (accessed 1 Feb. 2004)

CSI Wireless, 2003. “Asset Management Products”, http://www.csi-wireless.com/products/documents/NewAssetLink.pdf (accessed 1 Feb. 2004)

FCC, 2003. FCC: Enhanced 911. http://www.fcc.gov/911/enhanced/ (accessed 14 Oct. 2003).

Lee, S. and Y. Gao. “*Mobile Asset Tracking and Management Over the Web*”. Proceedings of Geodesy for Geotechnical and Structural Engineering. May 21-24, 2002, Berlin, Germany.

Peng, Z. and M. Tsou. *Internet GIS – Distributed Geographic Information Services for the Internet and Wireless Networks*. John Wiley & Sons, New Jersey, 2003.

Prasad, M., 2001. "Location Based Services", http://gisdevelopment.net/technology/lbs/techlbs003pf.htm (accessed 1 Feb. 2004)