An Open Source GNSS Reference Server

Thomas Yan, *University of NSW*
Peter Mumford, *University of NSW*
Andrew Dempster, *University of NSW*
Chris Rizos, *University of NSW*
Manosh Fernando, *University of Technology, Sydney*
Nam Hoang, *University of Technology, Sydney*

BIOGRAPHY

Thomas Yan is a graduate of the School of Electrical Engineering & Telecommunications, The University of New South Wales (UNSW), Sydney, Australia. He joined the research group at the School of Surveying & SIS in 2003 and has been involved in setting up the state CORS network in New South Wales since. His research areas are in CORS network and GNSS data communications.

Peter Mumford is a Research Assistant in the Satellite Navigation and Positioning laboratory, within the School of Surveying & Spatial Information Systems, UNSW. Currently he is working on GNSS receiver design and INS/GNSS integration. Peter has an Engineering degree in Surveying, and a Science degree in Mathematics. His interests are in FPGA design, software, RF and electronic design.

Andrew Dempster is Director of Research in the School of Surveying and Spatial Information Systems at UNSW. He led the team that developed Australia's first GPS receiver in the late 80s and has been involved with satellite navigation ever since. His current research interests are GNSS receiver design, GNSS signal processing, and new location technologies.

Chris Rizos is a graduate of the School of Surveying, UNSW; obtaining a Bachelor of Surveying in 1975, and a Doctor of Philosophy in 1980. Chris is currently Professor and Head of School. Chris has been researching the technology and high precision applications of GPS since 1985, and has published over 200 journal and conference papers. He is a Fellow of the Australian Institute of Navigation and a Fellow of the International Association of Geodesy (IAG). He is currently the Vice President of the IAG and a member of the Governing Board of the International GNSS Service.

Manosh Fernando is an undergraduate student at the University of Technology, Sydney. He is doing a Bachelor of Engineering course, majoring in Electrical Engineering.

Nam Hoang is an undergraduate student at the University of Technology, Sydney. He is doing a Bachelor of Engineering course, majoring in Computer Systems Engineering.

ABSTRACT

Assisted-GNSS (A-GNSS) devices (clients) require a source for providing the assistance data that enable positioning in a wide variety of environments, particularly areas of weak signal strength. This paper introduces an Open Source GNSS Reference Server (OSGRS) and the "GRIP" interface protocol (GNSS Reference Interface Protocol). The OSGRS is an Open Source Java application that provides data for Assisted-GNSS clients [1].

The OSGRS and GRIP will support researchers in developing A-GNSS algorithms with minimal investment and it will facilitate commercial operators in getting a reference server up and running quickly to test their A-GNSS client before investing heavily in a commercial service. In this paper, the design and implementation of the OSGRS and GRIP is discussed along with how it is being used at the University of New South Wales.

INTRODUCTION

The OSGRS provides an alternative to commercial A-GPS reference data solutions. The commercial offerings typically employ proprietary protocols, making the task of a developing a client that supports a range of reference servers more complex. In addition, these protocols and commercial server solutions may not be readily available, particularly to non-commercial research organisations such as universities. This may make research in this area more difficult, and an organisation may need to develop their own reference server and protocol.

The OSGRS is an Open Source Java application that provides data for Assisted-GNSS clients. The OSGRS is cross-platform and provides client applications with current, relevant and specific assistance data. The client may be an Assisted-GNSS handset or an application that serves a network of A-GNSS handsets.

The OSGRS can be configured to connect to one or more sources of GNSS data (data sources) in order to cache it and serve it up to clients on request. The data is provided to the client in a format that is useful for A-GNSS satellite acquisition and calculating the location of handsets. The data source may be a local GNSS receiver or any other type of data streams such as an internet-based GNSS data server. In the OSGRS, the data sources are Java classes that implement a specific Java interface and receive data from a physical source. The OSGRS has support for a NovAtel OEM2/3 data source and a NovAtel OEM4 data source and it is expected that support for more data sources will be developed as the OSGRS is installed and used.

The OSGRS provides its packaged assistance data to A-GNSS clients using the GRIP protocol. The GRIP protocol is an XML schema based protocol that uses HTTP transport. The client can request a list of A-GNSS data types that it requires and the OSGRS will return the appropriate data.

The client can include an approximate location in the request and it will receive the A-GNSS assistance data for satellites in view of that location. Otherwise it will receive the requested assistance data for all satellites that the OSGRS has information for.

The OSGRS complies with HTTP 1.1 which allows the client to maintain a persistent connection to the server. On request, the OSGRS provides all of the data types specified and/or appropriate errors for unavailable data. The protocol of the messages on the HTTP connection is request-response based with an XML payload as defined by GRIP.

OPEN SOURCE

This software has been implemented as Open Source in order to harness the collective wisdom of diverse users of Assisted-GNSS applications. This will enable developers to modify and improve the OSGRS offering to ensure that it is both functionally correct and users with different models of GNSS receiver will be supported. It will also enable GRIP to be enhanced and become a standard interface for providing A-GNSS data.

GRIP PROTOCOL

GRIP is the protocol used to make requests and receive responses from the OSGRS. GRIP defines the structure of the HTTP POST request as well as the structure of the XML document in the body of the request. The MIME type of the request and response body is 'application/xml'[8].

GRIP differs greatly from proprietary formats in that its messages are in XML format. Requests and responses of most proprietary protocols are in binary format. A key advantage to using XML is that it is easy to read. There are also many libraries which can be used parse XML data.

An XML schema describes the structure and content of an XML document. The type of XML schema language used in the GRIP protocol is XML Schema. XML Schema has been specified by the World Wide Web Consortium (W3C) [9].

Having a schema define an XML document is advantageous for a number of reasons. It clearly shows a user the format a GRIP request or response must follow. Sanity checking for various attributes can also be incorporated into the schemas. With the availability of schema validation systems, requests and response can be easily verified against schemas.

GRIP requests and responses are defined by a set of XML Schema files. The main files are GNSSRequest, GNSSResponse and GNSSErrorResponse. Each new version of the protocol is accompanied by a set of XML Schemas. XML Schemas are extensible so GRIP can therefore be extended to support other GNSS's such as Galileo, GLONASS, etc.

Versioning of the protocol is controlled by the namespace of the schemas. Each version of the protocol is assigned its own namespace i.e. the namespace of version 1.5 is http://www.gmat.unsw.edu.au/snap/grip/1.5. This lets the server know which version of the protocol a client is using.

GRIP has been fully specified for GPS reference data. The assistance data types are the navigation model or ephemeris, almanac, ionosphere model, UTC model, reference time, real time integrity, acquisition assistance and DGNSS corrections (RTCM type 1)[5]. The navigation model, almanac, ionosphere model and UTC model are specified as hex strings in the GRIP document which are similar to their raw formats in the GPS ICD. It may be desirable to specify new elements or extend present ones to represent these assistance types as a list of integers representing individual parameters.

There are two categories for assistance data in GRIP, all satellites and satellites in view. Table 1 shows which

assistance types are part of each category. Any assistance data types that are requested as 'satellites in view', will list the satellites that are in view of the latitude and longitude specified in the request or be data that is relevant to a position such as acquisition assistance or DGNSS data. Assistance data that is requested as 'all satellites' lists the entire collection of data stored for a particular data type in the OSGRS or other reference server implementing the GRIP protocol. Ideally the reference server should list data which covers the entire GNSS fleet.

Assistance data	Request for All Satellites supported	Request for satellites in View supported
Navigation Model	Yes	Yes
UTC	Yes	No
RTI	Yes	Yes
Ionosphere Model	Yes	No
Acquisition Assistance	No	Yes
Almanac	Yes	Yes
Reference Time	Yes	Yes
DGNSS	No	Yes

Table 1 GRIP assistance data categories

At the time of writing, the OSGRS supports all of assistance types specified in GRIP with the exception of acquisition assistance and DGNSS.

In the case of an error occurring during an attempt to process a client request, a response called the 'GNSSErrorResponse' is sent to the client. This response indicates that processing of the request was unsuccessful and provides information about why an error has occurred. In most cases the cause of an error is an XML request that has failed to validate against the XML Schemas. An internal error in the reference server may also cause an error response.

OSGRS DESIGN

The OSGRS has been designed using object oriented analysis and design (OOAD) techniques and is fully documented in UML. The design documentation is available on the web site [1].

The OSGRS design document documents the internal design of the OSGRS. UML diagrams are used in the document to illustrate key classes and processes. Any changes made to the design, such as in implementation, must be reflected in the design document. A new version of the document should accompany every new release of the OSGRS.

The initial high level design was created after the requirements were initially specified. The high level design changed to accommodate the addition of new requirements and features. The design also changed slightly throughout the implementation stage. This section of the paper will discuss the final high level design.

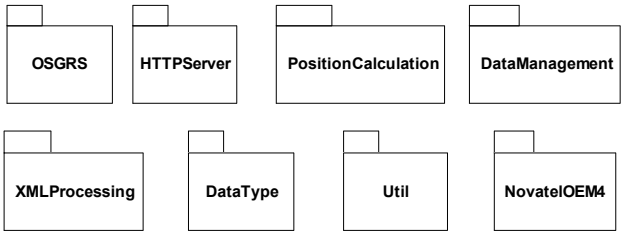

Figure 1 Package diagram of the OSGRS

A brief description of the classes in packages:

OSGRS: The main class of OSGRS. Contains the 'main' function of OSGRS.

HTTPServer: The classes associated with the HTTP server aspect of OSGRS.

DataManagement: The classes responsible for the management of data sources and the caching of assistance data.

XMLProcessing: The classes associated with processing XML data. This includes request validation, request parsing and response generation.

PositionCalculation: The classes associated with calculating user and satellite position.

DataType: Contains the data models for each of the assistance data types handled by OSGRS.

Util: Utility classes.

Upon start-up the main classes which exist for the lifetime of the execution are initialised. These include the HTTPServer class and DataSourceManager. The HTTP server class handles client connections, client requests and sends responses to clients. The DataSourceManager class is responsible for initialising data sources and holding the data cache.

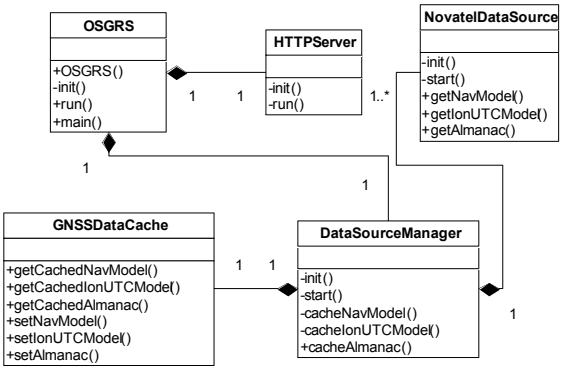

Figure 2 Class diagram of classes loaded at start up

The DataSoureManager was designed to communicate with all data sources in a generic fashion. For this purpose an interface class known as 'DataSource' was designed. This allows any data source that implements the 'DataSource' interface correctly, to be compatible with the DataSourceManager.

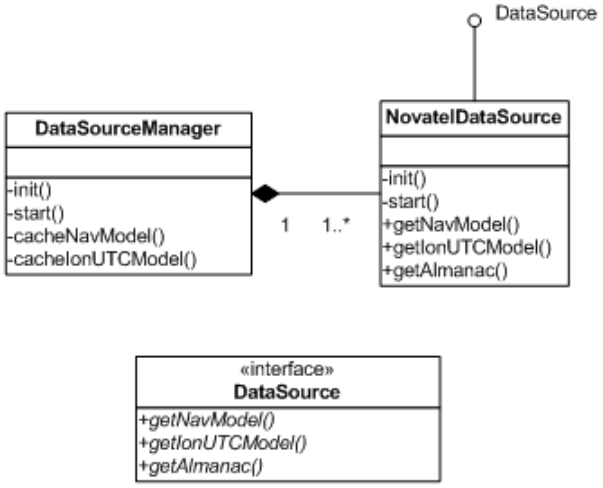

Figure 3 Data source implementing the DataSource interface

A 'DataSource' acts as an interface between the OSGRS and a source of GNSS data. A class that implements the DataSource interface shall provide assistance data upon request. Recievers like the NovAtel provide GNSS reference data in the form of a log feed. To be able to provide instantaneous GNSS reference data to the data cache, it is necessary to cache this data.

The GNSSDataCache is used to store all the assistance data that is received from the data sources . The periods at which each data source is polled for assistance data can be set up via a configuration file. Any data that is added to the data cache, is checked to be acceptable in terms of its 'age'. A series of threads monitor the assistance data stored in the cache on their staleness. Any stale data is discarded.

The first step is the validation of a client's request. For this task the Xerces XML parser's validation feature is used [7]. If any errors encountered while validating the request against the XML schema, a GNSSErrorResponse listing the errors is generated and sent to the client.

If the request is acceptable, it is parsed. The required assistance data types are then retrieved from the data cache. If 'satellites in view' data is requested, the satellites in view are generated from the clients supplied position. The assistance data types are then sent to GNSSResponse writer and the response is sent to the client. The request handling is shown in Figure 4.

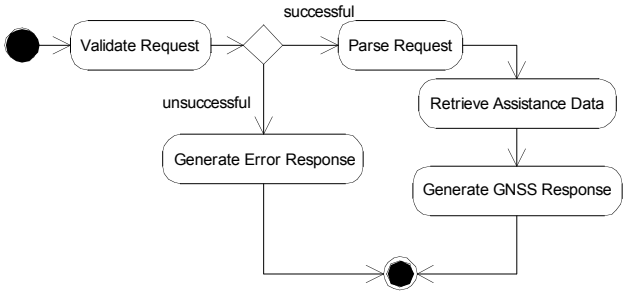

Figure 4 Activity diagram for handling requests

OSGRS IMPLEMENTATION

The OSGRS has been written in the Java programming language. The OSGRS was developed and tested on Microsoft Windows XP using Java 2 Platform Standard Edition 5.0 (J2SE 5.0). The Eclipse IDE was used to write, compile and build the code.

Java is a structured object oriented language, developed by Sun Microsystems. Java is platform independent, with runtime environments being available for most platforms. Java syntax is similar to that of the C programming language. Most Java applications run in an interpreted fashion although it can also be compiled directly into machine code.

The main reason for choosing Java as the programming language was portability. Java applications can be run on various other platforms with little to no modification. Memory management is almost autonomous and requires little management from the developer. This makes Java programs much less prone to memory leaks in comparison to languages such as C++. The standard classes included with Java are quite powerful which minimises linking to external libraries.

The OSGRS is a multi-threaded program. Some of the main threads which run for the entire execution include the listener threads of the NovAtel receivers (implemented inside the NovAtel data source), the threads

responsible for retrieving assistance data from data sources, the cache monitoring threads and the threads related with the Jetty web server component [6]. The program has been designed to accept connections from multiple clients. A thread is generated for each request the OSGRS receives.

The OSGRS links to external libraries on two occasions. The Jetty web server library provides handling of HTTP connections, accepting requests, sending responses etc. The Xerces XML parser library is used in the validation of XML requests against an XML schema.

The OSGRS uses a central configuration file, to set up logging directories, the listener port, refresh times for data types, etc. A configuration file is also used for each NovAtel receiver that will be used by the OSGRS.

OSGRS TESTING

During development, unit testing was performed manually with a set of test classes.

Due to time constraints no automated test harnesses exists for the OSGRS at the present time. This is an important area for future development of the OSGRS. This will allow developers to have more confidence in their code thereby giving more reassurance to any party using the OSGRS. Automated testing should be done on a class and component level.

At present, the output of the OSGRS is verified manually by comparing it to the output of the GPS receiver used as the data source. An automated test suite which performs this test automatically has been planned for future development.

There is a test client supplied with the OSGRS which allows the user to select the assistance data types required. The XML request and the response are shown in a text window.

OSGRS PACKAGE

The OSGRS, associated programs and documents will be available for download from the projects website. The main files will include the OSGRS source code, the OSGRSClient source code, the GRIP document, GRIP XML Schemas and the OSGRS design document.

The OSGRSClient software is a GUI OSGRS client which allows a user to create GRIP requests and send them to a server running the OSGRS software. Once the client receives the GRIP response, the client formats the XML data and displays it to the user. The OSGRSClient will serve as a reference implementation of an OSGRS client. The source code for the OSGRSClient has no licence associated with it.

If a user is to run the OSGRS, they have to meet a few requirements. Firstly, there must be a Java compiler and virtual machine available for their target platform. They must also have their data sources set up accordingly. If they intend to use a receiver which is does not have a data source, they must write their own.

The OSGRS is supplied as set of source files and must be compiled and built by the user. The user must modify the supplied configuration file/s according to their set-up and preferences.

CURRENT USE

At UNSW the OSGRS is currently operating to support and enhance research in A-GNSS field. The source data comes from a NovAtel OEM3 that is part of the UNSW Continuously Operating Reference Station (CORS) system. A GPS clock demonstrator project is currently being designed and will be using GRIP to communicate with the server in synchronising its time precisely to the GPS constellation.

LICENSING AND SUPPORT MODEL

The OSGRS is licensed using version 2 of the GNU General Public Licence (GPL) [2]. The GPL is a free software licence originally written by Richard Stallman of the Free Software Foundation for use in the GNU project. The GPL is a popular licence for free and open source software.

GPL was chosen as the licence for a number of reasons. The main reason is because while it allows an individual or organisation to make modifications and enhancements to the OSGRS, they must apply the conditions of the GPL to their derived work [2] known as ‘copyleft’ [3]. This encourages developers to share their enhancements with OSGRS community. Developers are free charge a fee for their derived work, but must provide the source code to their derived work.

Conditions 11 and 12 of the GPL [2] indicate there is no warranty provided by the owners of the program. This ensures that nobody is held liable for any issues arising from the use of OSGRS. An individual or group may wish to provide a warranty to a derivative of OSGRS for a fee. An individual or group is also allowed to charge a fee for the installation and support of an OSGRS.

There are two instances where the OSGRS links to libraries which are not covered by the GPL. These two libraries are the Jetty web server and Xerces XML parser,

both of which are covered by the Apache 2.0 licence. This is also free open source licence but it is incompatible with the GPL [4]. For this reason, there is a special exception on the two classes where these libraries are accessed.

The master copy, documentation and releases of OSGRS and GRIP protocol are controlled by the School of Surveying & SIS at the University of NSW. Any group or individual interested in making improvements to the OSGRS are encouraged to document their changes and submit these to the School. These changes will be reviewed and if deemed acceptable will be put into testing. After testing is completed, the changes will be available on a future release of OSGRS.

FUTURE

At present, the only GNSS that the OSGRS supports is GPS. As more GNSSs become operational, the OSGRS and GRIP will be expanded to accommodate these systems and their receivers. The GRIP protocol has already been specified for the Galileo GNSS.

REFERENCES

[1] The OSGRS Web site
http://sourceforge.net/projects/osgrs

[2] GNU General Public License,
http://www.gnu.org/licenses/gpl.html

[3] Copy Left, http://www.gnu.org/copyleft/copyleft.html

[4] GPL compatibility,
http://www.gnu.org/licenses/license-list.html

[5] RTCM Standard for Differential GNSS (Global Navigation Satellite Systems) Service, Version 2.3, RTCM Paper 136-2001/SC104-STD.

[6] Jetty http://www.mortbay.org/

[7] Xerces 2 http://xerces.apache.org/xerces2-j/

[8] RFC 3023 http://www.rfc-editor.org/rfc/rfc3023.txt

[9] W3C