Protocol Reference Model of GLOBALBASE Architecture: The Autonomous Decentralized GIS

Hirohisa MORI joshua at globalbase.org

Edition 2007-11-04

Contents

Chapter 1

Overview

1.1 Abstract and Goal

The GLOBALBASE Architecture connects geographical and spatial information delivered from numerous servers in a seamless manner and provides a novel autonomous decentralized network-type GIS that allows information searchers (clients) to browse for information in a virtual space called "Earth" existing on the internet.

Since information providers can start up servers freely within this architecture without being restricted by a spatial information control center, the architecture has significantly higher freedom compared to conventional network-type GISs. We call the project that uses this architecture to actually accumulate spatial information of the entire world and create a virtual Earth on the Internet, the GLOBALBASE project.

In this Request for Comments (RFC), the functions necessary for building this architecture are discussed and a protocol stack designed to achieve such functions are shown in order to receive your opinions. Moreover, we would like to have a System (Well-Known) Port number (No. 794if possible) to be assigned for the XL protocol implemented on top of TCP/IP in the proposed protocol stack.

1.2 Human Requirements

Engineer or Researcher

1.3 System Requirements

1.4 Introduction

GLOBALBASE is not the first attempt to share geographical information. Various technologies, for instance Web GIS, already exist. All the existing technologies, however, are based on centralized architectures.

Fig.1.1shows the simplest quasi-centralized architecture, where all geographical information is collected and stored on one server. Fig.1.2shows a slightly developed architecture, where the basic geographical information is stored on different servers, and the central server handles only searching and overlapping of target maps. This architecture employs a so-called clearing house method.

Both methods have bottlenecks on the server side. Quantitative bottlenecks, such as CPU processing power and network communication speed, are solved as a matter of course as the technologies are improved. However, we believe that qualitative bottlenecks, which are impossible to solve simply by upgrading the hardware, also exist in these technologies.

Currently, there are many map sites found on the WWW. However, if one sends a request to the administrators of such sites saying "I am interested in stores in the Edo period. Can you place such information on your site?," then, immediately, s/he will probably get a reply along the lines of "Our site focuses only on current information and as far as we know there is no demand for ancient information. We are sorry, but we cannot support such information." This is a qualitative bottleneck. We think that

there may be many people interested in Japanese history, however, and if such information were available on maps, it might become rather popular.

Problems such as the one outlined above, and other similar problems, will not occur if it were possible for anyone to deliver geographical information of one's own liking on one's own homepage, in the same way as when creating a personal homepage.

"I" can deliver information on stores in the Edo period without constraint. Many people interested in history will look at this information and set forth on a romantic journey through history. They can look around many places of historic interest and take a break at a coffee shop near a historical site found on the GLOBALBASE Mapion when tired. Both the current coffee shops and ancient teahouses can be displayed in the same window.

One of the purposes of GLOBALBASE is to solve this qualitative bottleneck. Fig.1.3shows an overview of the architecture. Information providers have their own sites and homes, and deliver their own geographical information about their areas freely. Viewers of information enter search conditions according to their interests, and scroll around with the browser to obtain necessary information. There is no restraining bottleneck between the information providers and the viewers. Information will be linked like an amalgam and will develop just like a living organism. People will perform their activities in the GLOBALBASE framework just like they do in the real world. In this way, another Earth, parallel to our own, may be completed.

When we think about it, this is a natural world for the Internet. In the framework of the GIS architectures established so far, however, this could not have been achieved.

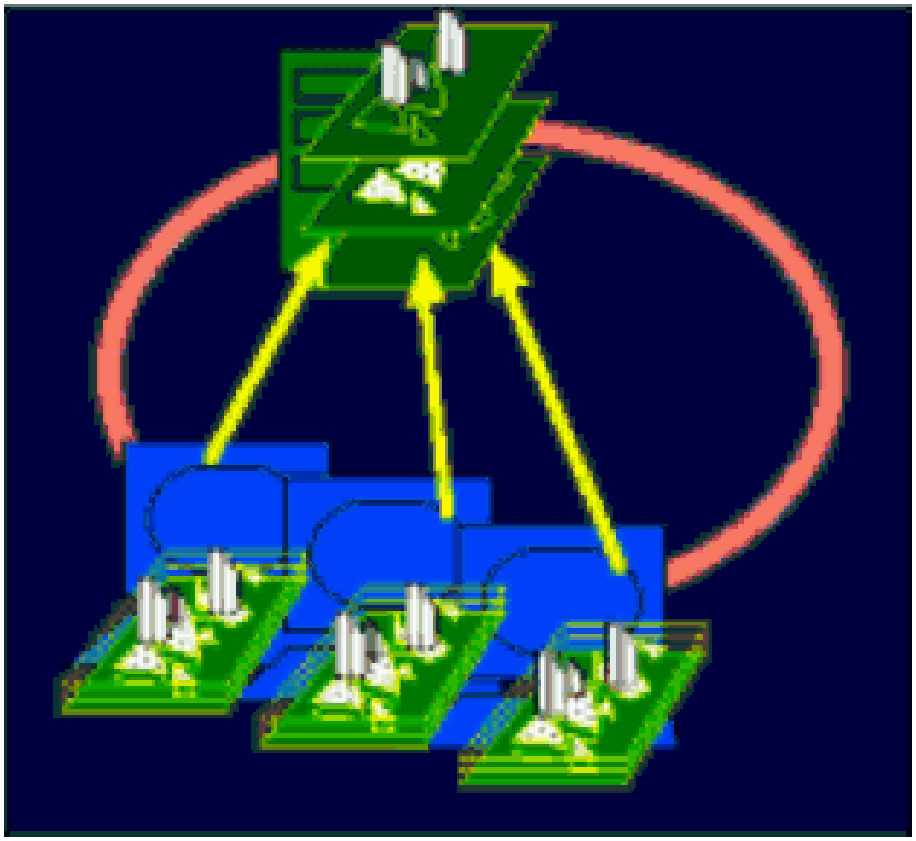

Figure 1.1: Centralized Type 1.

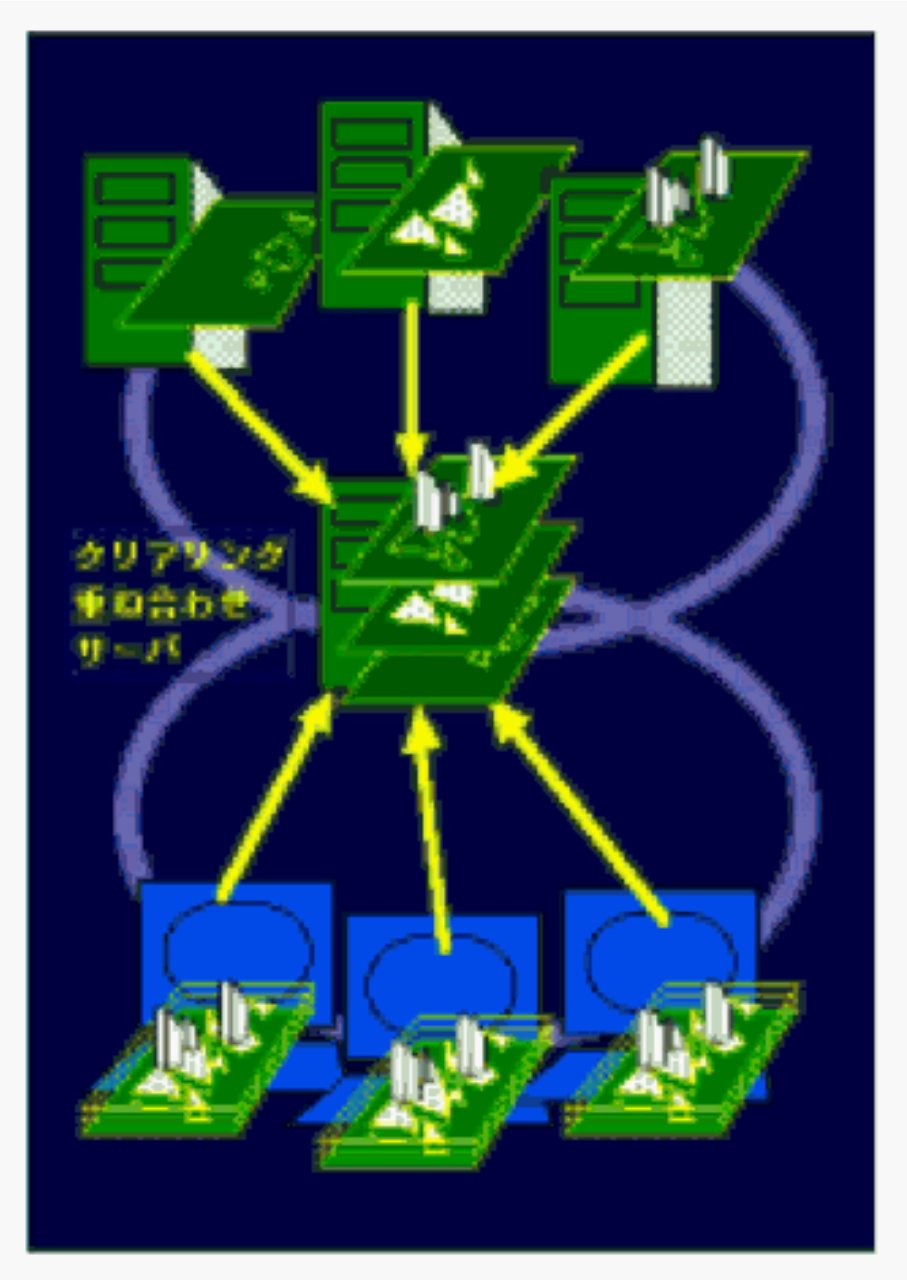

Figure 1.2: Centralized Type 2.

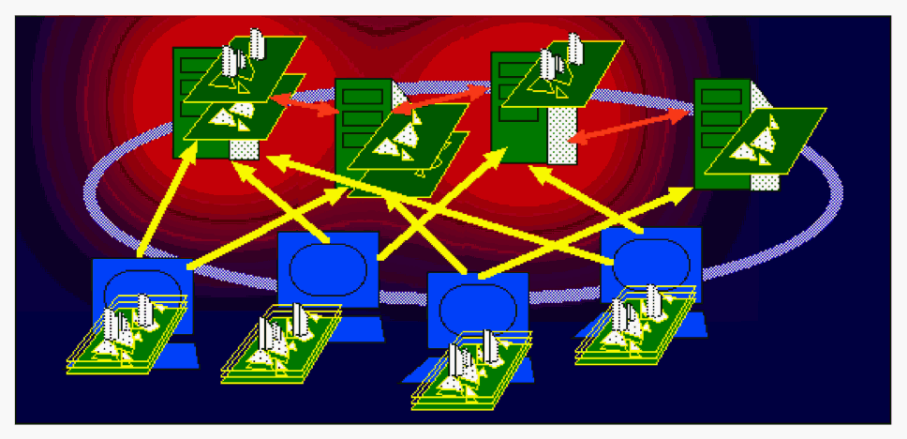

Figure 1.3: The Fully Autonomous Distributed Architecture Pursued by GLOBALBASE

Chapter 2

GLOBALBASE Functions

2.1 Abstract

2.2 Data Structure without Reference Coordinate System

In the process of decentralization, the first problem we encounter is the coordinate system. Current GISs have a single reference coordinate system, and all information must conform to it. This approach, however, leads to a qualitative bottleneck.

In GLOBALBASE, all coordinate systems are relative and can be defined individually by the user. In order for many people to be able to deliver geographical information freely and for such information to be shared to create one map of the Earth, GLOBALBASE must be able to handle maps created by normal drawing software as well as maps created by survey institutes in a consistent manner. To achieve this, being able to operate on user-defined coordinate systems is a necessity in order to deal with coordinate systems of different drawing software packages.

All the current GISs employ a single coordinate system that is used as reference. Such a coordinate system prohibits fusion of data from other fields that use different coordinate system concepts; this is one of the primary reason why the GIS technology has not achieved as great a degree of diffusion as expected. The common belief that GIS technology is difficult to use or requires expertise in geography, surveying etc., has given it a negative image.

By implementing user-definable coordinate systems, it becomes possible to capture any information within the same framework, from layouts drawn with drawing tools or architecture CAD software to layouts of rooms and show windows. However, since the only way to connect all this diversified information is through the use of distributed technologies, there must also be a method to integrate all the different fields in order to create one geospace.

2.3 Concept of Mappings Connecting Coordinate Systems

What will happen if each individual defines an arbitrary coordinate system? We no longer know the relationship between such individual, arbitrary coordinate systems. Fortunately, the technology called mapping presents a solution to this problem.

Let us assume that a person whose hobby is gardening draws a picture of the layout of his garden in order to present his new garden to other people. He then pastes it onto a Japanese map made available at the International Research Center for Japanese Studies at the location corresponding to his home.

The relationship between the layout of the garden and the Japanese map only needs to describe where the four corners of the garden are situated on the Japanese archipelago. A mapping is a list of such correspondences between the coordinates of the four points in the "garden" coordinate system and the corresponding coordinates in the "Japanese archipelago" coordinate system. By providing this mapping, the garden layout is connected to the Japanese archipelago, i.e., users can now find the garden by zooming in on the map of the Japanese archipelago.

A mapping plays the role of a bi-directional link; please refer to Fig.2.1Coordinate systems are indicated by rhombs and mappings defining the relationships between coordinate systems are indicated by

ovals. Mappings and coordinate systems can be placed on any server, because the mappings work as links beyond the confines of the servers.

Currently, mapping processing must be manually edited by information providers, but in the future, mapping editors and drawing editors supporting GLOBALBASE will become available. When that happens, it will become possible to paste one's own "garden" onto a map of the Japanese archipelago on another server just by dragging and dropping a control point in the drawing software.

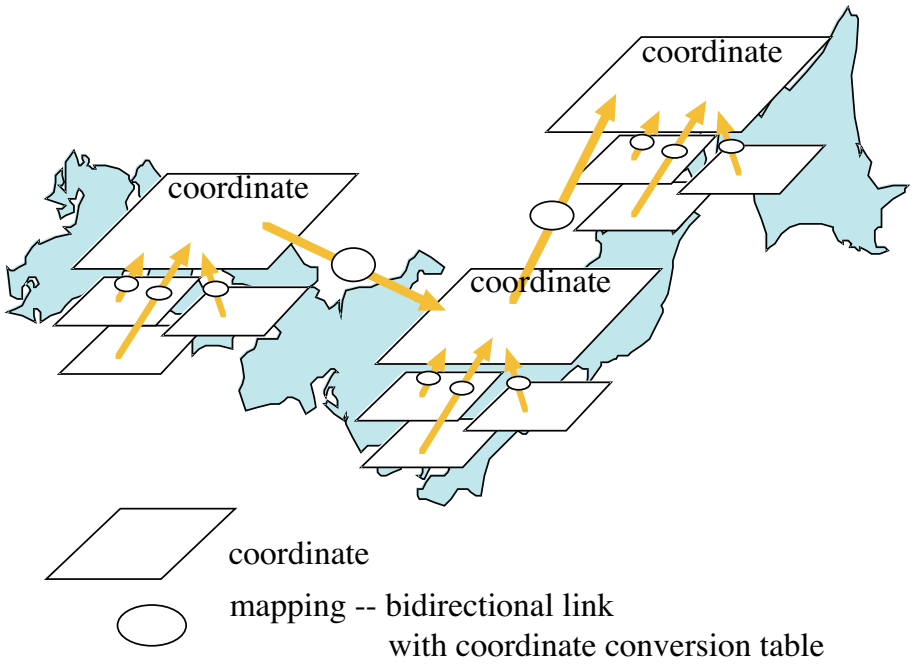

Figure 2.1: Coordinate Systems and Mappings

2.4 Distributed Search Engine

When an infinite number of information providers define one coordinate system after another, the maps become cluttered and we can no longer know what is where. That is, if one zooms to a location, all sorts of information may be displayed all at once. Obviously, from the point of view of users who search for some specific information, it is necessary that such information can be displayed in an orderly manner so that only the necessary information is obtained.

This problem is similar to the problem that occurred in the WWW in its early stage. One decade ago, a list of all WWW sites in the entire world was found at the NCSA site. It was possible to view all the WWW information in the world in one day by accessing those sites sequentially. At that time, there were less than 10Japanese sites in existence.

However, after that time, the Internet has been commercialized and the number of servers has exploded.

Finding necessary information quickly evolved into an insurmountable task. For this reason, much research has gone into efficient search engines that can help solving these issues.

Similar problems may occur to GLOBALBASE as well. If too much geographical information flows into the browser, it may cease to be usable. Search engines for GLOBALBASE, naturally, become necessary. Search engines for WWW, however, are of a centralized search type. In the GLOBALBASE architecture, they will be decentralized.

It is almost impossible to decentralize the WWW search engines. Fortunately, in GLOBALBASE, all types of information have coordinate systems, albeit relative, as well as geographical locality. Using these features, it is possible to create an infinite number of small search engines that gather local information. This new technology is called local search engines and works as follows.

A browser connects to a local search engine in the area it is currently displaying. When the user enters certain search conditions, such as a time period, the browser searches for geographical information of the surrounding areas. The local search engine, then, returns all geographical information that matches the conditions from a slightly wider area than the displayed area. The browser caches the information and displays it as new information when the user searching for information moves to the left or right. When the displayed area changes, the browser searches for a new local search engine corresponding to the new displayed area and searches for information again. The browser repeats this process regularly, thus moving from one local search engine to another.

Fig.2.2shows an example of states of local search engines. Each server collects geographical information in its vicinity by tracking mappings regularly. In this example, it can be seen that server A never overlaps with server C, and server C never overlaps with server A. Since a server only accumulates local information in this manner, each local search engine does not need to have a large capacity, and can be very simple.

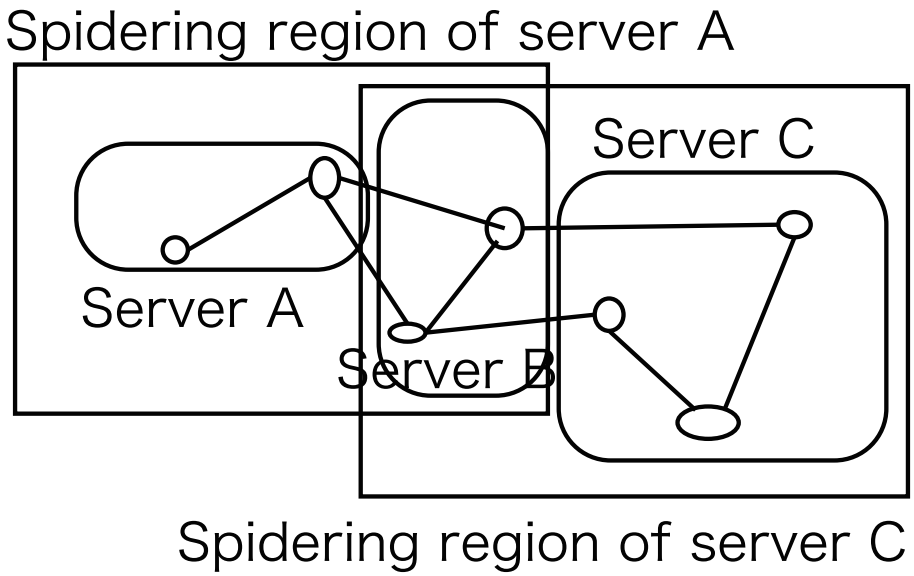

Figure 2.2: Local Search Engines

2.5 Searching for Mapping Paths Auto-Configured Routing Protocol

A browser may collect and store necessary geographical information, but all of this information is based on relative coordinate systems. This means that information on different servers cannot be superimposed as is. It is necessary to check the physical relationship between two or more coordinate systems.

Furthermore, it is not guaranteed that two coordinate systems among the searched geographical information are connected with one mapping. This is because mappings created by information providers are arbitrary. The coordinate system of the garden mentioned before and the coordinate system of the Japanese archipelago are connected with a mapping. Now suppose that there is also a map of parks in the Kanto area connected to the Japanese archipelago, and the garden map and the map of parks in Kanto area are not directly connected, but it is required to overlap these two maps only.

In this case, it is necessary to convert the coordinate system of the garden to the coordinate system of the Japanese archipelago, and further convert this to the coordinate system of the map of parks in Kanto area. In order to do so, it is first necessary to search for and link between the arbitrarily specified mappings.

We found a hint for the solution to this search problem in an unexpected field, namely network routing technology. Let us try to apply the Internet framework to a network of coordinate systems connected by mappings. Coordinate systems can be considered to correspond to hosts on the Internet and mappings correspond to networks; then GLOBALBASE can be considered as the Internet as is. On the Internet,

two arbitrary hosts can exchange packets by specifying the IP address of the destination, no matter how they are connected. IP addresses are Internet addresses that can be used for routing. This implies that it is possible to know how two coordinate systems are connected via mappings if addresses that can be used for routing can be assigned to the coordinate systems.

It is, however, not practical to make the information providers themselves assign addresses for which IP routing is possible to each coordinate system. For this reason, a technology to automatically assign addresses that can be routed to coordinate systems is required. This technology is called the ACRP (Auto-Configurated Routing Protocol) and was originally invented by the leader of this project as a protocol for control of Local Area Networks used in intelligent buildings. It is now applied as a technology for sharing information in the GLOBALBASE architecture. It is safe to say that GLOBALBASE could not have been made possible if we had not had this technology.

2.6 Browsing Method

The last technology to be explained here is the method for overlapping maps and changing the display of geographical information according to the instruction by information searchers, based on geographical information and mapping information thus gathered. As shown in Fig.2.3, collected coordinate systems are overlapped by converting mappings. The area indicated by the vertical cube is the area displayed in the browser. If the user moves this display area to the left or right, some coordinate systems go outside the cube and others come inside the cube. Those outside the cube are excluded from overlapping, while those coming into the cube are included in the overlapping by searching through the associated mappings. By repeating this process, the browser is able to connect maps seamlessly, and is eventually able to scroll across the entire Earth.

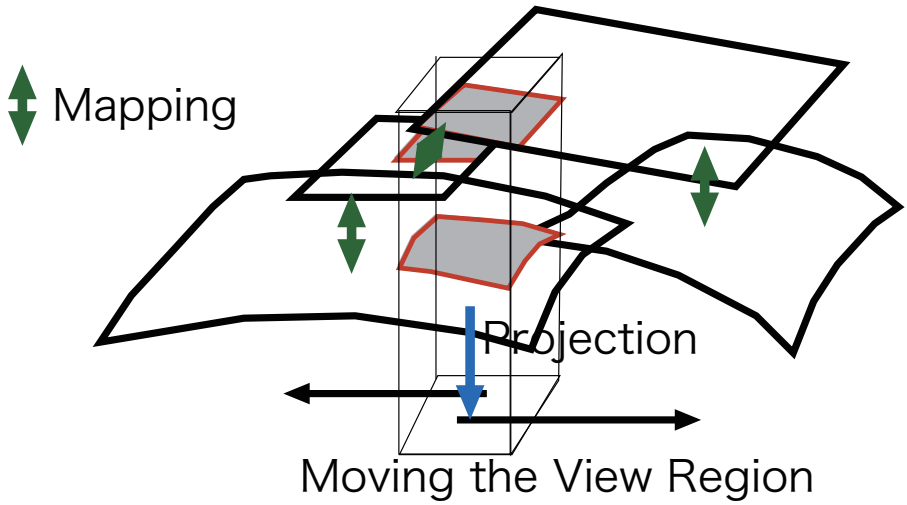

Figure 2.3: Movement of browser view area

2.7 Source Code

The architecture with the functions discussed above is implemented in the GLOBALBASE protocol, which is explained in the following. Please refer to http://www.globalbase.org/, where you can download the source code. The areas of the source code where the functions of the protocol are implemented are indicated in the explanation of the protocol given here. It is thus possible to understand the dynamics of the protocol by referring to the source code in a formal manner.

First, the configuration of the source code is briefly explained. The source code archive downloaded from the download site is named according to the format gbs.X.XX.XX.tar.gz. When you decompress this file, a directory called gbs.X.XX.XX is created. The files under gbs.X.XX.XX/env contain the source code. In the following explanation, this location is assumed to be the working directory and the location of

the source code is indicated as a relative path from the working directory. The src/arch directory contains code dependent on the machine architecture and OS. Under this directory, there are subdirectories named according to the relevant operating system, such as src/arch/unix and src/arch/mac. Directories under src other than src/arch contain code independent of the machine architecture.

Chapter 3

Protocol Layering

3.1 Abstract

This chapter explains the protocol stack that is used for the actual implementation of the functions discussed in Chapter 2,i.e., the GLOBALBASE protocol reference model (Fig.3.1(a) (b)). The GLOBALBASE protocol can run on any reliable, bi-directional byte-stream type communication protocol. A typical example is TCP/IP.

Two major protocols are placed on top of this bi-directional stream-type protocol: the XL (XML LISP) protocol, which allows calling functions implemented on a remote host, and the GLOBALBASE protocol, which implements the functionality that characterizes GLOBALBASE, as discussed in the previous section (hereinafter abbreviated as GB protocol). The GB protocol is implemented by utilizing functions provided by the XL protocol.

The XL protocol implemented at the initial stages of the GLOBALBASE project was written directly in LISP. A system was implemented where a LISP S-expression sent to a remote host is evaluated on the remote host side and the result is returned. XL is an extension of this system; an environment was created in which it is possible to interpret and execute elements of XML as S-expressions and thus interpret tags as function symbols.

It may be possible to construct this XL functionality with existing protocols such as HTTP. For example, it may be possible to implement the XL functionality based on the HTTP protocol by running cgi scripts instead of employing the method of evaluating S-expressions.

There are, however, many GLOBALBASE processes that must maintain state transition information of the remote host or can be executed more efficiently if they maintain the state of the remote host. Such functions can be achieved by using cookies in HTTP, but that would be inefficient. This is because HTTP is basically a stateless-type communication system centered around individual documents, and implementing cookies with the explicit aim of achieving stream-type communication is inefficient. Since TCP/IP itself is a stream-type communication protocol, it is more efficient to implement a system that can have states directly in it.

XL is a character-string-based protocol. Basically, character strings conforming to the XML syntax are transmitted via a stream-type protocol. Various functions can be executed when the remote host interprets such character strings using LISP. With this method, however, it is not possible to transfer binary data. Thus, in order to transfer binary data efficiently, an approach is taken where binary data is inserted as part of the character strings, marked by escape codes.

The implementation of the XL layer is included in the src/xl directory of the source code directory tree. Moreover, the implementation of the GB layer is included in the src/gbs directory.

3.2 XL Layer

3.2.1 Overview of XL Layer

The XL layer is an interpreter that interprets and executes LISP or XML. It is unique in that it can define and interpret documents created in the XL (XML LISP) language. XL is a novel language implementing a mechanism to interpret XML elements as LISP S-expressions. The interpreter can only interpret LISP,

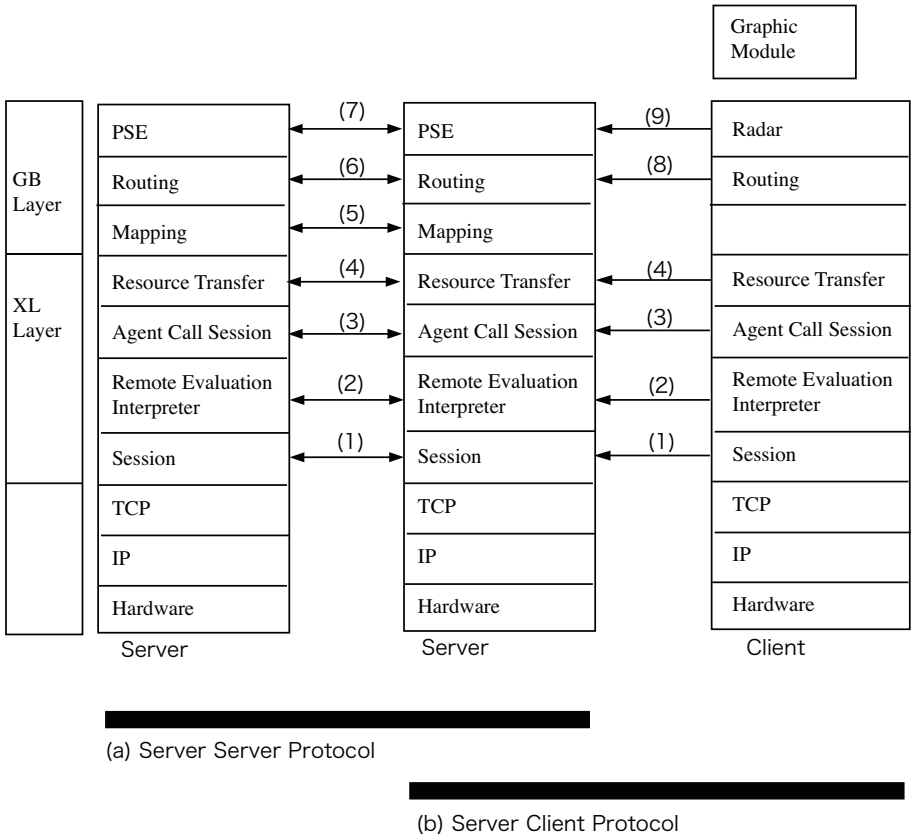

Figure 3.1: GLOBALBASE Protocol Reference Model

so XL was developed in order to address the diffusion of XML in recent years. This language system contains elements of both XML and LISP syntaxes, but the XML elements and LISP S-expressions have the same internal structure of representation.

The XL layer plays the role of evaluating expressions that come in from an upper or lower layer and returns the results of the evaluation. When an expression originating from an upper layer is evaluated, the result is returned to the upper layer without changing the internal representation. An expression from a lower layer is formatted as a character string conforming to the XL language syntax. The character string is parsed, converted to the internal representation, and evaluated. The result is subsequently converted back to a character string and returned to the lower layer.

In the XL layer, a function for sending an expression to the XL layer on a remote host is implemented as well. This function converts the internal representation of the expression to a character string and sends it to a remote host via the lower layers. With this protocol, the internal state of the XL layer of the remote host changes as a consequence of these executions. This internal status can be read from an upper layer as necessary; as a result, a full-featured communication protocol is achieved.

In order to exchange character strings conforming to the XL language syntax with a lower layer, the XL protocol is implemented on top of a reliable stream-type communication protocol. It is thus a prerequisite that the ability to invoke functions of a stream-type communication protocol, such as open, close, write and read, is supported. The way a stream is opened is not necessarily fixed; active open can be used on the client side and passive open can be used on the server side. Moreover, when a stream is passed from one process to another process, the stream is already opened for the receiving process to which the stream was passed.

The typical example of a reliable stream-type communication protocol is TCP/IP, but various other applications can be considered in order to support the XL layer. For example, in a network environment where HTTP proxies are used, the XloHTTP protocol implemented on top of HTTP using cookie functions is useful. If it is desired to encrypt communication, a protocol that supports encryption of streams can be inserted here as well. Moreover, it is possible to combine several protocols for the communication.

Each part of the source code of the XL layer explained in subsequent sections is placed under the src/xl/lib directory. The code for the lexical analyzer, parser etc. of the XL layer is placed under the src/xl/lib/lisp directory, and the code for the interpreter, remote evaluation, call agents etc. is placed under the src/xl/lib/std directory.

The following sections explain the structure of the XL layer, starting from the lower layers.

3.2.2 Stream Control (Reference Model (1))

The role of this layer in the XL protocol handles, but is not limited to, relaying the data from the XL interpreter into an existing lower-layer stream-type communication protocol such as TCP/IP. The XLoHTTP protocol, which implements streaming via HTTP, is implemented in the stream control layer along with an encryption protocol. We are planning to prepare a separate RFC for each protocol; please refer to these.

In the source code tree, the implementation of the streaming services is placed under the src/stream directory and the src/arch/$MACHINE/std/stream directory. The former directory contains code that is independent of the OS and machine architecture, while the latter contains platform-dependent code. $MACHINE should be replaced with the appropriate architecture name to obtain the entire directory name.

3.2.3 XL Language Concept

In the XL layer, expressions coming from a lower layer are transported via a network, i.e., a heterogeneous environment. To accommodate this, they are encoded as character strings conforming to the syntax of the XL language. When the character strings enter the XL layer from a lower layer, they are then converted to internal expression by a lexical analyzer.

One of the major characteristics of the XL language is that it has two modes: the XML mode, where input character strings are interpreted according to the XML syntax, and the LISP mode, where they are interpreted according to the LISP syntax. It would be simpler and more elegant to unify the language system to use either XML or LISP. Nonetheless, both of them are used in order to create a processing system that allows parsing XML, which is becoming the dominant standard for information exchange on the Internet, while at the same time using the LISP syntax for the sake of its efficiency.

Switching between these two modes are performed in the following manner.

1. The LISP mode is selected if ' ' (' is received while in the XML mode.
2. The LISP mode is selected if an end tag corresponding to the tag that triggered the XML mode is received.
3. The XML mode is selected if the XML start tag is received while in the LISP mode.
4. The XML mode is selected if ')' corresponding to ' (' that triggered the current LISP mode is received.

It is possible to switch between the LISP mode and XML mode by the changes above.

The XML elements and LISP S-expressions have the same internal meaning. A tag of an XML element can therefore be interpreted as a function name (symbol) of a LISP S-expression. For example, <greeting> Hello World </greeting> in the XML language format is equivalent to (greeting "Hello" "World") in the LISP language format. On the other hand, LISP has also been expanded so that it is possible to attach attributes to symbols. These attributes are comparable with attributes of XML. Thus, the following two descriptions have the same meaning.

```
<greeting type="morning"> Hello World </greeting>

( [greeting type="morning"] "Hello" "World")
```

[greeting type="hello"] is a symbol with an attribute.

The detailed syntax of the language will be specified in another RFC.

3.2.4 XL Language Syntax Digest

This section explains the minimum syntax of the XL language in order to describe the protocols implemented in the layers above the XL layer. Firstly, the basic types of the XL language are as follows.

```
i) symbol (tag)

SYMBOL ::= . (escape characters <>()'",/ )

ii) string

STRING ::= .*

iii) signed integer

INTEGER_DEC ::= [1-9][0-9]*
INTEGER_OCT ::= 0[0-7]*
INTEGER_HEX ::= 0x([0-9]|[a-f]|[A-F])*
INTEGER_1 ::= INTEGER_DEC|INTEGER_OCT|INTEGER_HEX
INTEGER ::= (+|-)INTEGER_1 | INTEGER_1

iv) floating point number

MANTISSA_1 ::= [0-9]*\.[0-9]*
MANTISSA_2 ::= [0-9]*
EXPONENTIAL ::= (+|-)[0-9]*

FLOATING ::= (MANTISSA_1 | MANTISSA_2) (E|G|e|g) EXPONENTIAL

v) raw data

RAW ::= #BYTE].....

This is the escape sequence for the raw binary data transimssion
embedded in the string of the stream.
```

A decimal number is embedded in BYTE and as many bytes of raw data as specified by this number is placed instead of "...." Other than the basic types, there are two compound types, LIST and NULL, which combine the basic types above. These types are defined by the following syntax. In particular, the LIST type that starts with a SYMBOL is called the ELEMENT format.

Next, the syntax structure is as follows.

(sp1 is regarded as a number of contiguous white-space, line feed and tab characters. sp is regarded as sp1 or a 0length character string.)

```
/* XML syntax */

xml-elements ::= xml-element
    | xml-element sp xml-elements
    ;

xml-element ::= start-tag sp1 xml-data sp1 end-tag
    | empty-tag
    ;

xml-data-list ::= sp
    | xml-data-list sp1 xml-data
```

```
    ;

xml-data ::= xml-element
    | list-type
    | '^' lisp-element
    | xml-string
    ;

xml-string ::= STRING

start-tag ::= '<' SYMBOL sp attributes '>'
    | '<' SYMBOL '>'
empty-tag ::= '<' SYMBOL '/' '>';
end-tag ::= '<' '/' SYMBOL '>';

/* LISP syntax */

lists ::= sp
    | list-type sp lists
    ;

lisp-elements ::= sp
    | lisp-elements sp1 lisp-element
    ;

lisp-element ::= lisp-terminal-list
    | '(' sp ')' sp /* NULL type */
    | list-type sp /* LIST type */
    | '\'' lisp-element /* = (quote lisp-element) */
    ;

list-type ::= '(' sp lisp-elements sp ')'
    ;

lisp-terminal-list ::= sp
    | lisp-terminal
    | lisp-terminal sp1 lisp-terminal-list
    ;

lisp-terminal ::= '"' STRING '"'
    | SYMBOL
    | '[' sp SYMBOL attributes sp ']'
    | INTEGER
    | FLOATING
    | RAW
    ;

/* shared syntax */

attributes ::= attribute
    | attribute sp1 attributes
    ;

attribute ::= SYMBOL sp '=' sp '"' STRING '"'
    ;
```

The NULL and LIST types are defined as compound types based on lisp-elements using the syntax above. If lisp-terminal, the first element of the LIST type, is a symbol, that lisp-element is regarded as the same as an xml-element with the same symbol name. Thus the XL language as a whole is structured as follows.

```
/* start terminator */

XL-language ::= lists | xml-elements;
```

Finally, the character codes received from the streams are discussed. Basically, the character codes that can be transmitted via the streams must conform to open standards. The XL language not only functions as a description language for a package of complete programs of finite lengths, but can also be used directly as a communication protocol where it waits in an infinite loop for lines of various unrelated expressions to be processed. Thus, it must tolerate various changes in transmission status. For example, it must be possible to change encoding or insert binary data in the middle of a character string.

For this reason, the following escape codes are introduced.

1. It is possible to change character encoding if lt;code representing a character>;" is received while in the XML mode.
2. If "#<number of bytes>]" is received while in the LISP mode, the subsequent number of bytes indicated by the argument "<number of bytes>" is interpreted as binary data.

Flexible communication can be achieved by introducing the escape codes above.

The source code for the XL language lexical analyzer is found in src/xl/lib/lisp/xllisp_lex.c and the source code for the parser is found in src/xl/lib/lisp/xllisp_parser.c.

3.2.5 XL Interpreter

The XL layer contains a list of symbols to which values and functions are bound, or, in LISP terms, an aggregation of environments. An expression passed to the XL layer from a lower or upper layer is evaluated within one of these environments. An evaluation expression passed from a lower layer is evaluated by an interpreter, which is a thread that takes care of the evaluation of the evaluation expression. An expression passed from an upper layer is evaluated by an internal function implemented in the XL layer such as eval. This section describes how the expressions from lower layers are handled – in other words, the interpreter structure. The evaluation method of expressions passed from upper layers is described in the next section and onward.

An interpreter can only refer to one environment, through a link that was assigned to it when it was created, as well as the parent environment of that environment; it is not aware of all the environments in the layer. There are several interpreters and they can operate in parallel. Actually, an elegant approach is to implement the interpreters in parallel threads.

An interpreter is always connected to one stream of a lower layer, and analyzes and evaluates character strings streamed in via this stream. Assume that there are two hosts, called A and B, and a stream is opened between them via TCP/IP or a similar protocol. An interpreter is created on both hosts at the same time as the stream is opened. Not only the interpreter connected to the stream, but also other interpreters and threads can perform writing operations to the stream from both hosts. Reading operations, however, are only allowed for the interpreters directly assigned to the stream.

If an expression is written to the stream from host A, the interpreter of host B receives and evaluates it. The evaluation result is returned to the stream by the interpreter of host B. The interpreter of host A then receives the result. The evaluation result is also expressed as one expression. It has the following structure.

```
<Result> [line number] [result] </Result>
```

The interpreter of host A evaluates this "Result" function (element). The Result function is a standard built-in function; the interpreter is not allowed to change the operation result. When the Result

function is evaluated, it is identified which entity in host A that sent the original expression via the stream, and the result is passed to the sender.

The evaluation result of an expression is not always returned to the stream. For example, if the evaluation result of the Result function is empty and the result had to be returned to the stream, hosts A and B would end up throwing the expression back and forth between each other.

In our implementation, the streams that can be connected to interpreters are not limited to network lines; file streams can also be connected. With this mechanism, it is possible to evaluate XML and LISP scripts in files using the interpreter and lexical analyzer.

If you wish to view the source code of the interpreter, it is found in src/xl/lib/std/interpreter.c.

3.2.6 XL Layer Call Sequences

If it is desired to interact with the XL layer from an upper layer or obtain information of this layer, an expression is sent to the XL layer by invoking a certain internal function provided by the XL layer. The function can be invoked according to the following three procedures.

1. Eval procedure: a procedure for simple evaluation only
2. Remote procedure: a procedure corresponding to the opening procedure of a socket, i.e., stream (Reference Model (
3. Agent call procedure: an agent is defined as a process providing various services of remote hosts. This procedure is used for calling agents on remote hosts in an efficient manner, and has more characteristics and sophistication than 2(Reference Model (

The eval procedure is the simplest procedure for interacting with the XL layer from an upper layer. An upper layer invokes the eval function with an environment and an expression as arguments, and the function returns the evaluation result. The valid environments are limited to those on the same host as the function was invoked from; i.e., it is not allowed to perform evaluations of expressions outside the same host. The eval function is explained below.

```
eval
Agent:    all
Arguments:
    1. Environment
    2. Expression
Return value:
    evaluation result
```

Procedures 2and 3basically open a stream and perform evaluation of an expression in an environment on a remote host. Procedure 3,in particular, is more complicated than the others and utilizes the implementation of procedures 1and Therefore, in a protocol stack where the internal XL layer is divided into greater detail, it would be positioned in an upper layer compared to procedures 1and Procedures 2and 3will be explained in the following sections.

The source code that implements each procedure is placed in different directories. The source code for eval is found in src/xl/lisp/eval.c, the source code for the remote procedure in src/xl/std/remote.c, and the source code for the agent call procedure in src/xl/std/session.c.

3.2.7 Remote Evaluation Procedure (Reference Model (2))

This procedure opens a stream in a layer below the XL layer. There are basically two ways of opening a stream, active open and passive open. A data communication function and a function that closes a stream and interpreter are also provided in this procedure.

First of all, a procedure for opening the XL layer, open, is provided. A parameter indicating whether to open the stream in either passive or active mode can be specified for the open procedure. Upon receiving an active open request, the XL layer sends a request to open a stream in the active mode to the specified address and port. When the stream is opened, an interpreter is assigned to the stream. This

completes the active open processing of the XL layer. The following parameters are required for active open.

```
open active
Agent:    all
Arguments:
    1. Address of the target host
    2. Port number of the target host
    3. Environment used by the interpreter on the local
       host for evaluation
Return value:
    Interpreter ID
```

Conversely, upon receiving a passive open request, the XL layer sends a request to open a stream in the passive mode to a certain port. After that, the XL layer waits to receive a request to open a stream from another host. If it receives a stream open request, it checks the permissions related to the connection. If it is judged that the connection is possible, it opens the stream and assigns an interpreter to it. This completes the passive open processing of the XL layer. At least the following parameters are required for passive open.

```
open passive
Agent:    all
Arguments:
    1. Port number for awaiting connection
    2. Information required for checking permissions of
       the target host
    3. Environment used by the interpreter on the local
       host for evaluation
Return value:
    None
```

Once a stream is opened, the XL layer returns an ID that identifies the stream and interpreter to an upper layer. The upper layer then sends an expression to the target host and obtains the evaluation result using this ID, by means of a function provided by the XL layer that allows this operation. On the passively opened side, the passive open function does not return an ID directly. It is thus all right to implement some functionality that generates an event in an upper layer when the opening actually succeeds. Information can be transferred from the XL layer to an upper layer via a return value of a function, evaluation result and various other ways. The currently available methods are explained in section 3.2.8.

The function that sends an expression from an upper layer to the stream in question has the following parameters.

```
remote
Agent:    all
Arguments:
    1. Interpreter/stream ID
    2. Expression
Return value:
    Evaluation result
```

The function that closes a stream and terminates all the related processing has the following parameters.

```
close
Agent:    all
Arguments:
    1. Interpreter/stream ID
Return value:
    None
```

This function closes the stream specified by the stream/interpreter ID, discards the interpreter and releases/invalidates the resources reserved by the stream/interpreter. If a stream is closed on one host, the peer host can no longer send or receive an evaluation expression; the stream is closed and the interpreter discarded.

It may be a good idea to implement an appropriate action to be taken when a stream is closed only on one host, such as the XL layer generating an event to an upper layer. It is also necessary to take measures to prevent IDs from being used easily once invalidated, because if an interpreter ID is invalidated and immediately used for another interpreter, an upper layer may refer to this ID by mistake.

3.2.8 Remote Agent Call Procedure (Reference Model (3))

In this procedure, the concept of agent is introduced. An agent is defined as a process that provides various services based on the set of XL primitive functions. For example, in case of GLOBALBASE, it is possible to define an agent equipped with GLOBALBASE functions. This allows implementing various functions on one server and building a system where such functions can coexist.

Another important role of this layer is to manage timeout for recovery from failures. In case of reliable communication, there is no way of knowing in the stream layer if a remote host goes down. Basically, there is no means to distinguish whether the host went down or the network is slow. If it is carelessly judged that the host is down when it is not actually down, there is a risk that processing on the remote host side fails. The fact that the host went down is only known when the host recovers and it is realized that the state is different from the state in the previous communication. If a remote host never responds, it is not possible to judge whether or not the remote host is down forever.

One of the means to solve this problem is to set appropriate timeouts judging from the function of an application. If a remote host does not respond for more than a specified time, the stream to the host and associated processing are paused by the timeout. This timeout setting depends on the application, however. This means that it cannot be implemented in the stream layer. Thus it is appropriate to implement it as a function of this agent.

This section describes how to call an agent first and then explains how an agent handles timeout processing.

A client can call various agents on the server. An agent called by a client sends various expressions from the client and the state of the agent's environment changes depending on the order.

There are two problems involved in performing various types of processing by connecting to an agent on a remote host. One is whether or not it is possible to select an agent suited for the purpose. Another problem is that part of the agent's state data must be shared between the host and client. The consistency of this data may not be maintained between the agent and the client, which is situated at a remote location, if the line is disconnected. These problems can be solved by the remote agent call procedure.

Firstly, a function for activating an agent with the remote call procedure is defined. This expression is the one defined for an agent called xlsv (xl server). At least one xlsv agent is activated for a server supporting the XL protocol and this xlsv agent opens an XL port. If a client connects to this open port, it is also connected to the xlsv agent at the same time. The following function must have been defined for this xlsv agent.

```
(SetAgent agent-name mode)
Agent:    xlsv
Arguments:
    STRING agent-name
    STRING mode
Attributes:
```

```
    version (client version)
Return value:
    NULL
ERROR
    invalid agent
    permission denied
```

"SetAgent" activates an agent specified by "agent-name" in the mode specified by "mode." An agent is activated if permitted by a table of correspondences between the agent-name read by xlsv at the activation and the actually activated process name and argument. Agents that are not registered in this table cannot be activated (permission denied/undefined agent).

The mode can be user, server, root etc., and is passed as part of the arguments to the agent invocation.

The activated agent inherits the stream that was opened between the client and xlsv as is. The expression sent after the SetAgent instruction is sent to this agent.

In many cases, an agent performs processing on some resources identified by URLs. An agent that is suited to the individual type of file should be used; for this reason, xlsv has a table of correspondences between filename prefixes and agents. The following function is used to refer to this correspondence table.

```
(GetPrefix method prefix)
Arguments:
    STRING method (Get/Set method)
    STRING prefix (URL prefix)
Attributes:
Return value:
    ELEMENT list of the agent
```

The client can know the required agent with this function.

By using the remote call procedure and SetAgent function outlined above, it is possible to activate as many agents as one wants on one server.

By taking advantage of the aforementioned remote call procedure, it is basically possible to generate an agent on a remote host and perform communication. However, as mentioned earlier, it is necessary to implement timeout management for recovery from failure in a reliable communication system here. Basically, the following two types of timeout must be implemented.

1. Timeout when there is no communication from a remote host for a fixed period of time
2. Maximum operation time of an agent

For 1,several timeouts must be implemented for individual phases. First of all, the following timeouts should be implemented for a client.

- Timeout of response to active open (including permission processing etc.)
- Timeout of response to SetAgent

is implemented at the opening of a stream as well, but must be set taking permission processing specific to the XL layer into account as well.

Next, the following timeout should be set for the server.

- Timeout when there is no request from a client

  Moreover, the following timeout should be implemented for both the client and server.

- Timeout of response to transmission of an evaluation expression

The timeouts above have been given default values, but it may be a good idea to implement a protocol that measures each response time and dynamically changes the default value to the optimal value corresponding to a specific network.

Note that it may happen that a stream may be disconnected due to one of the timeouts above, even though there is no problem in the system. In such cases, it may be possible to perform fault recovery processing, but the problem can usually be solved more simply by setting the necessary conditions and sending the same evaluation expression again. When directly accessing the remote call procedure, it is necessary to implement unique fault recovery processing, however. The remote agent call procedure has a function to perform fault recovery processing as well.

In the remote agent call procedure, the client side assigns a session for each state of an agent in a remote host. It connects to the same agent if the session ID, connection destination server and activated agent are the same. It then generates an environment for this session to this agent and sets the necessary state. With this procedure, the same environment is never used for different sessions even if the agent is the same.

If a remote agent times out or goes down due to failure, the client tries to activate the agent again. If it is activated, the state before the failure is set up for the environment. If it turned out to be impossible to recover from failure after three attempts, it is necessary to generate an error.

One session is opened by invoking open_session, its environment is set by invoking remote_env, and the expression is sent by invoking remote_session. The session is closed by invoking close_session. How these functions are invoked is summarized below.

```
open_session
Agent:    all
Arguments:
    none
Return value:
    session ID
```

This function opens a remote agent session and assigns a session ID to it.

```
remote_session
Agent:    all
Arguments:
    1. session ID
    2. Activated agent (can be omitted)
    3. Destination URL
    4. Get/set operation (can be omitted)
    5. user
    6. mode
    7. Expression
Return value:
    Evaluation result of the expression returned by the
    remote agent
```

If the activated agent is omitted, the client makes an inquiry to the remote server for a list of agents that can be activated. It selects the operation specified by the "operation" argument and an agent corresponding to the extension of the resource at the URL. If an agent is explicitly specified, it connects to that agent. "user" and "mode" are used as arguments of SetAgent. When the agent is successfully activated, the expression is sent and the evaluation result is returned as the return value of remote_session.

```
close_session
Agent:    all
Arguments:
    1. session ID
Return value:
    None
```

This function closes the session indicated by the session ID.

3.2.9 Some Delayed Controls

When evaluating expressions, it is efficient to implement a processing (a certain type of continuation) that delays the evaluation until the evaluation result actually becomes necessary. If the evaluation is made on a remote host, i.e., the remote evaluation and remote agent call procedures are used, an evaluation expression is transmitted to the remote host when the evaluation is requested, At the same time, it is possible to implement a processing that returns a pseudo-return value without waiting for the real result. The pseudo-return value from the remote host is delayed until the point when the real result from the remote host actually becomes necessary. This is called delayed rendezvous.

In case a client passes two or more expressions to a remote agent and the first expression goes into waiting state for some reason, the agent may evaluate the second expression before the first. In this case, the sequence in which the results are returned is changed, but the client can still recognize the correspondence between the called expressions and the returned results using the line numbers attached to the result elements of the response.

The implementation of the mechanism above is optional, but the protocol must be programmed assuming that this option is implemented.

The implementation of delayed controls can be found in src/xl/lib/std/remote.c, src/xl/lib/std/session.c and src/xl/lib/std/interpreter.c .

3.2.10 XL Layer Agents

The basic agents defined for the XL layer are xlsv and standard. xlsv is an agent that performs passive open of a port for XL and waits for the connection. SetAgent and the functions necessary for activating the agents are defined here.

standard is an agent that implements the basic expressions of the XL layer. Get and Set expressions are especially important. They correspond to Reference Model (

```
(Get path)
Agent:    standard
Arguments:
    STRING path
Return value:
    ELEMENT / LIST

(Set path data)
Agent:    standard
Arguments:
    STRING path
    LIST data
Return value:
    ELEMENT / LIST
```

The Get expression returns the resource corresponding to the path given by the "path" argument. A path is given as an absolute path from the data root directory of the XL server. Moreover, a combination of a server name (including domain name) and port number corresponds to a URL. For example, if "path" is /foo/data.crd, the server name is hoge.domain.jp and the port number is 8080,the URL becomes xlp://hoge.domain.jp:8080/foo/data.crd.

The Set expression is an instruction that writes the information indicated by "data" to the resource specified by "path." Basically, this instruction rewrites the status of the server; it is currently necessary to choose not to implement or authenticate it.

Currently, the main source code of xlsv is in src/xl/server and standard is in src/xl/sh; please refer to these files.

3.3 GB Layer

3.3.1 Target Data Structure

Overview of Data Structure

The GB layer actually manages the data structure of GLOBALBASE discussed in Chapter 2and implements mapping path routing and distributed search engines. To provide an overview of these features, the data managed by resources with URLs is first defined. Such resources can basically be retrieved by the standard agent because they are equipped with URLs. Each resource, however, contains not only information created by the geographical information provider at the server but also information added automatically by the server processing itself to manage mappings etc. Here, this added information is called management information. Some of the management information cannot be directly retrieved by the standard agent and the method of retrieval varies; the file can simply be downloaded as is or accessed via various operations. For this reason, unique Get/Set operations are prepared for gbstd (an agent of the GB layer that will be explained later) so that detailed information can be specified via attributes. This section explains the types of information specified by information provider and loading information attached by the server for each resource type.

The basic resources include coordinate system, mapping, raster data and vector data resources. Collections of raster data and vector data are called objects. Moreover, each resource is equipped with metadata. When a resource is described in the XML format, the head part contains metadata indicated by the meta tags, while the rest of the content is called the data main body.

Whenever resources exist on the GLOBALBASE server, the server generates various types of management information from the above-mentioned metadata. The server never refers to the data main body of resources other than mapping resources. The data main body is displayed and referred to as the target of geographical information processing by a client.

Metadata mainly contains definition data of each resource, such as basic declarations of bibliographical information, storage position of data main body, type and size of coordinate system, resolution etc. All contents outside the meta tags are considered to belong to the data main body. If the data main body is specified as a separate file, this file is considered the data main body.

Since the data main body is basically not referenced by the server, it can be upgraded to advanced data formats without being limited by the version of the server. The data main body of mapping resources, however, must be referenced by the server, as it is required for position calculation on the server for creating index information. How this part can be upgraded to a new method is an issue to be resolved in the future.

Fundamental Metadata Structure

Every resource has meta elements immediately below the root element. The file element is the only type of meta element that is essential. The file element indicates the position of the data main body.

The file attribute type=”xl” indicates that the data main body is in the same XML file following at this meta element. In this case, the file is an empty element. If ”type” is a value other than ”xl,” the type attribute indicates the format of the data main body, the file element is not empty, and a URL to the data main body is specified as the data.

<file type=”xl”/> ... The data main body is inside the file in which the meta tag is defined.

<file type=”pdb”/> url </file> ... The data main body is in a file specified by the URL.

A bib meta element is not essential, but can be specified at most once. A bib element defines bibliographical data of a resource. Namespaces are available within the bib element. The namespace specified by xlp://isjhp1.nichibun.ac.jp:8080/gb_metadata (the common name is gb) is the reference namespace of the GLOBALBASE server. Date etc. is referenced as an index, for example:

```
<bib xmlns:gb="xlp://isjhp1.nichibun.ac.jp:8080/gb_metadata">
<gb:title type="text" data="Toyo University, Itakura Campus"/>
<gb:creator type="text"
  data="Toyo University, Itakura Campus, Fujita Laboratory"/>
<gb:content.period type="W3C-DTF" data="1990-01-01 /"/>
<gb:issue.period type="W3C-DTF" data="2003-08-25"/>
```

```
<gb:property type="gb-prop" data="base"/>
</bib>
```

The following tags are defined as tags of the gb namespace.

- gb:title: Title of a resource
- gb:creator: Creator of the resource
- gb:content.period: Period of the content of the resource
- gb:issue.period: Period when the resource was created
- gb:property: Classification of the resource specified by GLOBALBASE

These tags are the minimum requirements for creating indexes for resources in the PSE layer. If they are missing, the PSE layer does not register resources for the indexes, as explained later. It is possible to define many namespaces for other bibliographical information to use them together with the gb namespace, but they are not essential.

Other meta elements are unique to each resource, which are explained in the following sections.

Raster Data Resource

One example of raster data is shown below.

```
<?xl version="0.1" encoding="UTF-8"?>
<raster>
 <meta>
  <file type="cr">
   Itakura_Newtown.cr
  </file>
  <scale>
   (/ 1 5000.0 )
  </scale>
  <scan-resolution>
   200.0dpi
  </scan-resolution>
 </meta>
</raster>
```

The root tag of raster data is either raster or luster. The meta elements unique to resources of raster data include scale and scan-resolution. If a map of reduced scale is scanned as raster data, the scale is registered for the scale element and the resolution set in the scanner at scanning is registered for the scan-resolution element.

The data main body is not defined currently.

Vector Data Resource

Two examples of vector data are shown below. The first example shows how to specify a binary vector format, while the latter is an example where shape data is embedded in this description.

```
<?xml version="1.0" encoding="EUC-JP"?>
<vector>
 <meta>
  <file type="pdb">
   gaiku.pp
  </file>
```

```
 </meta>
 <set-scheme scheme="kokudo-digital-map-2500"
  onmap="xlp://isjhp1.nichibun.ac.jp:8080/kokudo/style/kdm2500.onmap.xl"
  onmap.type="text/xl"
  card="xlp://isjhp1.nichibun.ac.jp:8080/kokudo/style/kdm2500.card.xl"
  card.type="text/xl"/>
 <set-visible-resolution>
   0.200000dot/m
 </set-visible-resolution>
</vector>

<?xl version="0.1" encoding="EUC-JP"?>
<vector>
 <meta>
  <bib xmlns:gb="xlp://isjhp1.nichibun.ac.jp:9100/gb_metadata">
   <gb:title data="Annual City Event Sketchbook" type="text"/>
   <gb:subtitle data="Events" type="text"/>
   <gb:creator data="International Research Center for Japanese Studies"
        type="text"/>
   <gb:content.period data="1928" type="W3C-DTF"/>
   <gb:issue.period data="2002-03-31" type="W3C-DTF"/>
   <gb:property data="plot" type="gb-prop"/>
  </bib>
  <file type="xl"/>
 </meta>
 <set-scheme card.type="text/xl"
  card="xlp://tois1.nichibun.ac.jp:8080/tois/style/Gyouji-plot.card.xl"
  onmap.type="text/xl"
  onmap="xlp://tois1.nichibun.ac.jp:8080/tois/style/Gyouji-plot.onmap.xl"
  scheme="Gyouji-plot"/>
 <mark code="1280" img="xlp://tois1.nichibun.ac.jp:8080/image/gyouji1.gif">
  <point>-114841.6m -19315.5m</point>
 </mark>
 <information code="1280" name="1280" scheme="Gyouji-plot">
  <item-list>
   <item>
    <category>January 5</category>
    <name>Inari Ohyama Festival</name>
    <url>http://tois1.nichibun.ac.jp/database/html/gyouji/gyouji_4.html</url>
   </item>
  </item-list>
 </information>
</vector>
```

In both examples, a style file is specified to display attribute information related to the shape data by means of a set-scheme element. The set-scheme element has attributes only and contains no data. The attributes have the following meaning.

card: Style file used when displaying information in another window card.type: Syntax of the style file above, MIME encoding onmap: Style file for displaying information on a map onmap.type: Syntax of the style file above, MIME encoding scheme: Name specified for the entire set of the styles above

It is assumed that the attribute information related to shape data can be expressed in the XML format regardless of the format, and thus a corresponding indication and display style must be specified. Currently, only text/xl is supported for card.type and onmap.type, but we plan to support text/xsl in the future as well. text/xl assumes that the XML data along with the attribute information represents an XL evaluation expression, evaluates this expression within the appropriate environment after the style file is executed, and then displays the result.

Mapping Resource

A mapping resource describes information required to overlap two resources. For example, it might indicate the position relationship of two coordinate systems or the positions of vector and raster objects within a coordinate system. For this reason, it consists of elements that specify two resources to be overlapped and elements to specify the position relationship. The resources are specified by src and dest elements within the metadata. The definition of the position relationship is placed in the data main body.

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<map>
 <meta>
  <file type="xl" />
  <src> itakura_nt.crd </src>
  <dest> /kokudo/coord/09.crd </dest>
  <dp> 130 </dp>
 </meta>
 <point-map> "1" (list 0m 0m) (list 0m 0m) </point-map>
</map>
```

The meta elements unique to mapping resources include src, dest and dp. They specify the mapping source (src), mapping destination (dest) and mapping depth (dp).

Coordinate System Resource

```
<?xml version="1.0" encoding="EUC-JP"?>
<coordinate>
 <meta>
  <bib xmlns:gb="xlp://isjhp1.nichibun.ac.jp:8080/gb_metadata">
   <gb:title type="text" data="Toyo University, Itakura Campus"/>
   <gb:creator type="text"
     data="Toyo University, Itakura Campus, Fujita Laboratory"/>
   <gb:content.period type="W3C-DTF" data="1990-01-01 /"/>
   <gb:issue.period type="W3C-DTF" data="2003-08-25"/>
   <gb:property type="gb-prop" data="base"/>
  </bib>
  <file type="e2d"/>
  <mr>
   ((0km 0km )
    (1.3km 0.62km ))
  </mr>
  <v>
   <resolution>
    2dot/km
   </resolution>
  </v>
 </meta>
</coordinate>
```

The special elements unique to coordinate system resources are mr (minimum rectangle) and v (visible). "mr" is the effective range of the coordinate system. The rectangular range is given by the pair of points for which the coordinate values along each dimension are minimum and maximum, respectively.

The "v" element specifies the condition where the coordinate is visible on the browser display. Currently, only one format is supported for this condition, providing the resolution of the display using the following format.

```
<v><resolution> minimum-resolution maximum-resolution </resolution></v>
```

or

```
<v><resolution> minimum-resolution </resolution></v>
```

When the coordinate is within the browser's display range, it will be visible if the current resolution of the browser is between these parameters. If maximum-resolution is omitted, maximum-resolution is set to 2000times larger than the value of minimum-resolution.

Another method of indicating whether to display a coordinate is by judging the distance between the coordinate and the view-point of the browser. This method will be considered when 3DGLOBALBASE is developed.

The "file" element in the coordinate system resource gives many types of coordinate systems. We assume that the characteristics of the coordinate system is given by the topological and geometric structure. Currently, the topologies of Euclidean space topology (euclides) and globe surface topology (globe-sur) are supported. Geometric structure is then induced by introducing a distance measure into the topology space. The Pythagorean distance is supported for the Euclidean and, globe topologies and ellipsoid distance is supported for the globe surface topology. The globe and ellipsoid distance measures require radius and long and short radius values, respectively.

According to mathematical theory (Manifold theory), it is enough to support Euclidean topology. For the sake of user convenience, however, we implemented support for the globe surface topology as well, as it provides an easy framework for describing the surface of the Earth.

In the future, it will be possible to define many non-Euclidean geometric structures, for instance Minkovskian geometry, embedded in euclides, and we will eventually be able to describe the whole universe within the GLOBALBASE framework.

The format of the "file" element is

```
<file="[topology];[geometry];[parameters];[axis]"/>
```

and we provide certain macro descriptions, such as "e2d" for often-used parameter values. "e2d" stands for Euclidean 2Dspace, and specifying this macro is equivalent to specifying the following parameter values.

```
euclides;pythagoras;comp
```

The "axis" argument gives the direction of the coordinate axis, " math " specifies the mathematical coordinate system and " comp " indicates that the computer coordinate system is used.

3.3.2 GB Layer Agents

In the GB layer, two essential agents are defined: gbstd (GB standard agent) and gbpmd (GB plate metadata database). gbstd handles the xl agent functionality in the XL layer as well as the mapping and routing functions of the GB layer, which are explained below. gbpmd implements the PSE functionality and distributes/accumulates metadata of maps among the geographical information. The functions of each agent are explained in the following sections.

It is allowed to define auxiliary agents other than these two agents in the actual implementation. For example, in the current implementation, the gbstd agent only provides a function to receive requests from a network at the front end in most cases; in the current mechanism, such requests are always passed on to gbmp (GB management process), an agent operating in the background. The gbmp agent not only processes requests of the gbstd agent, but also caches various types of information in memory, executes polling processing etc.

Since such auxiliary processing is not directly seen from the network, there is no restriction on the implementation. For example, it is possible to consider embedding the functions directly in the gbstd agent.

The main implementation of the gbstd and gbpmd agents exists in src/gbs/std and src/gbs/pmd, respectively. In addition to these two agents, the gbmp agent always exists on the server, but cannot be seen from the client side. The main implementation of this agent resides in src/gbs/mp. In reality, the system is implemented in such a way that in many cases, various types of processing requests issued to the gbstd agent, in particular, are passed on to and executed by the gbmp agent.

3.3.3 Polling Interval Control

The GB layer presupposes implementation of the XL layer and GLOBALBASE servers communicate with each other based on this layer. One of the basic communication forms is polling. For example, if there are mappings between resources existing on different servers, a server with a mapping performs polling to let each of the servers with resources know that its mapping is active. Moreover, if a resource is registered in a certain server, the server polls the PSEs (Partial Search Engine) in the vicinity of the server. As a result of this polling, the PSEs attempt to obtain the bibliographical information of the resource and place it in the indexes. If the polling yields no result, the resource is deleted from the indexes.

A problem in such polling approaches is that polling is performed very often on servers with many mappings, and, depending on the amount, the server may even go down. For this reason, it is necessary to implement a mechanism that calculates and controls the polling interval and the load on the GLOBALBASE server.

Basically, the polling function is activated by sending a polling expression to the target agent of the polling destination. The polling interval requested by the polling source must be set in one of the arguments of this expression. The polling destination performs processing related to the polling and, at the same time, sets the time limit where the resource to be polled is deleted from the indexes to the time obtained by multiplying the polling interval specified in the argument by four and adding the current time.

The polling destination calculates a polling interval that does not apply load on the server. This value is encoded and returned as the return value of the polling expression. Thus, an expression of this type will have the following general format, although it may differ slightly depending on the type of polling.

```
(Polling url interval)
Agent:    all agents using polling
Arguments:
    url: Target resource
    interval: Polling interval indicated by the polling
            source
Return value:
    Integer (the resulting polling interval in units of
            seconds)
```

The polling source calculates the actual polling interval from the returned polling interval value. If there are several polling destinations in the polling source and they are processed at one go, the maximum value among the polling intervals returned from individual polling destinations is set to the target polling interval. However, if the next polling interval is set to the target value in case the previously set polling interval value was too small compared to the target value, the resource targeted by the polling may be deleted from the indexes due to a timeout on the polling destination side.

For this reason, the target value is not specified at once but temporarily set to twice the previous polling interval. In this case, the resource is not deleted from the indexes because the polling destination does not time out until after four times the polling interval. The polling interval is gradually made larger in this manner and set to the target value when it just exceeds the target value.

If, on the other hand, the target value is shorter than the previous polling interval, no problem occurs even if the target value is changed in one go.

Depending on the target, various methods of calculating the polling interval that take the load on the polling destination into consideration, can be considered. If the processing related to polling can

be completed instantly, the number of resources to be polled can be checked constantly and the polling interval can be set to this number times some constant. If the processing related to polling takes some time, as for example in the case of bibliographical data registration processing in a PSE (especially the calculation of the position of the resource to be registered takes time), it can be considered to employ a method to set the polling interval to the average registration processing time multiplied by the number of registered resources times a constant.

Alternatively, it is possible to employ a method where the ratio between the registration processing and idling is constantly monitored and the polling time is fed back and controlled so that this rate is kept constant.

3.3.4 Mapping Layer (Reference Model (5))

The meta/src and meta/dest elements in the mapping description specify a sort of bi-directional link that connects two resources. An operation for tracing this link from one resource to the other in layers above the mapping layer is implemented; the actual overlapping of coordinate systems is achieved by taking advantage of this operation.

The main task of the mapping layer is to maintain and operate on these bi-directional links. The information of the src and dest elements and the URL of the mapping resource are registered at the resources at both ends as management information. Through this management information, upper layers can see the adjacent resources of any resource via mapping resources.

The function interface of the mapping layer is implemented in the gbstd agent. The expression related to the mapping layer is as follows.

```
(Set url data)
Agent:    gbstd
Arguments:
    STRING url
    LIST data
Attributes:
    STRING mode
Return value:
    INTEGER
```

This expression stores information of a mapping in the management information of the resource specified by the first URL argument. The URL of the mapping resource, the URLs of src and dest, the update time and the polling interval of the polling source are specified in "data." The return value is the polling interval obtained by the polling destination. The polling interval is based on the GLOBALBASE polling interval control method.

The format of the "data" argument list is as follows.

```
<record>
<src> src-url </src>
<dest> dest-url </dest>
<map> map-url </map>
<dp> dp </dp>
<mod> mod </mod>
</record>
```

"map-url" is a mapping file, i.e., the URL of the mapping resource that issues this expression. "src-url" and "dest-url" are the URLs of the mapping source and mapping destination, respectively. "dp" refers to depth and indicates the order of overlapping of coordinate systems or objects that are overlapped with this mapping resource; this value is ignored currently. "mod" indicates the time when the mapping file was updated, in XL time (number of seconds from New Year's day of

This management information can be retrieved by the Get operation. Note that it is necessary to specify either mode="src-map," mode="dest-map" or mode="object" as attribute when invoking this function.

```
(Get url)
Agent:    gbstd
Arguments:
    STRING url
Attributes:
    STRING mode
Return value:
    LIST
```

If mode="src-map" is specified, a list of mappings for which "src" is equal to the resource specified by the "url" argument is returned. If mode="dest-map" is specified, a list of mappings for which "src" is a coordinate system resource among mappings whose "dest" resource is equal to the resource specified by the "url" argument is returned. Finally, if mode="object" is specified, a list containing only mappings for which "dest" is equal to the resource specified by the "url" argument and the resource of "src" is either vector and raster resources is returned.

A server to which a mapping resource is registered must check the mapping at a fixed interval and issue a Set operation to the URLs of the src and dest so that the management information above can be maintained. On the other hand, servers positioned at src and dest must set timeout limits for the management information and perform operations to delete timed out management information. These polling operations are performed using the GLOBALBASE polling interval control method.

The implementation of the mapping layer is found in src/gbs/mp/mapping.c.

3.3.5 Routing Layer (Reference Models (6) and (8))

The role of the routing layer is to provide a service that returns a list of mappings (called mapping path) that traces a path between two given arbitrary coordinate systems (coordinate system resources).

GLOBALBASE can be regarded as a collection of overlay networks superimposed on the Internet, where mappings and relationships between resources can be seen as vertices and other resources can be seen as nodes. On a slightly higher level of abstraction, since object resources are always connected to coordinate systems, which are in turn connected by mappings, GLOBALBASE can, in fact, be considered a set of overlay networks consisting of only coordinate system resources and mappings.

In this way, an algorithm that searches for a mapping path between two arbitrary coordinate systems is equivalent to an algorithm that assigns route selectable addresses to all coordinate systems and performs routing among these addresses.

In GLOBALBASE, ACRP (Auto-Configurated Routing Protocol) is employed for this algorithm. ACRP is an algorithm that extends RIP (Routing Information Protocol) to automatically deliver path selectable information and assign a path selectable address to each node (coordinate system resource), which eliminates the trouble of having to manually assign route selectable addresses from information providers of GLOBALBASE. The basic algorithm of ACRP has been published in a paper [2], and the detailed algorithm used for GLOBALBASE will be introduced as a separate RFC. This RFC only describes how to determine the basic information and the protocol for exchanging information between servers as well as between server and client.

A route selectable address assigned to an individual coordinate resource is called a coordinate system ID. This ID is an address of a variable length with a subnet address value from 0to It can for example be expressed as (cid 3345This address is the same as (cid 03345or (cid 003345which have additional leading zeros. When the number of coordinate systems increase and the range of assigned addresses is extended, the old addresses are interpreted as being equivalent to the new addresses with additional zeros.

One of the major features of ACRP is that a unique priority must be specified for each coordinate system resource. Fortunately, in GLOBALBASE, a URL unique to each resource has already been assigned; this character string can be specified as a priority. Moreover, it is desirable to assign higher order to older priorities, so that the older information has higher stability. Hence, the following character

string is defined as the priority, so that the smaller the character string when compared alphabetically, the higher the priority.

time, octal 11-digitformat]: [domain name of url]: [port-no of url]: [path name of url]

The XL time above is the number of seconds elapsed since 0o'clock midnight on January 1,If it should happen that the XL time cannot be expressed in octal 11-digitformat, the first character can be set to 1and the number of digits increased.

The route selection table of ACRP is stored in the management information of each coordinate system resource. This table is called a mapping path table in GLOBALBASE, and has the following entries.

```
<?xl encoding=".." version="1.0"?>
<mpt>
<cid pri="...."> id0 id1 </cid>
<regulation> interval </regulation>
<level> 1
<entry> id0-entry "priority" sum
<dir> hops "map" "crd" </dir>
.....
</entry>
....
</level>
<level> 0
....
</level>
</mpt>
```

The cid element above indicates an ID assigned to a coordinate system. The pair of integers, i.e., id0 and id1, which can take values from 0to 127,are concatenated to form the coordinate system ID. As the number of coordinate systems increase, this list of integers may become longer, i.e., 3integers, 4integers and so on. The pri attribute of the cid tag indicates the priority of the coordinate system in this overlay network that has the highest priority. This value must be consistent in all coordinate systems existing in a connected network. Thus, it is possible to check whether or not routing is possible in advance by looking at this value.

"regulation" indicates the regulation interval based on the ID assignment algorithm ACRP. The unit is seconds and "interval=0" signifies that the address and table are determined.

The level element is a routing table at the level corresponding to id0 and id1. The number of levels naturally increases as the number of concatenated IDs increases. An entry element corresponding to the ID is prepared for each level. "id0-entry" indicates the ID and "priority" indicates the priority of this entry (see ACRP [2]). "sum" is the number of coordinate systems belonging to the address indicated by this entry and "dir" indicates the routing destination for reaching the address of this entry. If there are two or more systems, the two routing destinations with the least number of hops are stored. From this coordinate system, it is possible to route to "crd" via "map."

If the "id0-entry" data of the "entry" element is equal to id0, it means the resource itself; thus hops=0, "map" = "" and "crd" = "".

An example of a routing table of a coordinate system is shown below.

```
([?xl eoncoding="EUC-JP" version="1.0"])
(mpt
([cid
pri="000334750032:tois1.nichibun.ac.jp:8080:/tois/gyouji/gyouji11.crd"]
31 56)
(regulation 0)
(level 1
(entry 127
"000334750573:isjhp1.nichibun.ac.jp:8080:/nichibunken/uno/thysen/4new-kuro.crd"
1
```

```
(dir 2
"xlp://isjhp1.nichibun.ac.jp:8080/nichibunken/uno/uno1/uno1.no6.map"
"xlp://isjhp1.nichibun.ac.jp:8080/nichibunken/uno/uno1/uno1.crd"))
(entry 126
"000334750537:isjhp1.nichibun.ac.jp:8080:/nichibunken/uno/thysen/1old-kuro.crd"
1
(dir 2
"xlp://isjhp1.nichibun.ac.jp:8080/nichibunken/uno/uno1/uno1.no6.map"
"xlp://isjhp1.nichibun.ac.jp:8080/nichibunken/uno/uno1/uno1.crd"))
(entry 125
"000334750512:isjhp1.nichibun.ac.jp:8080:/nichibunken/isjhp/Heian/6/base.crd"
1
(dir 1
"xlp://isjhp1.nichibun.ac.jp:8080/nichibunken/isjhp/Heian/6/base.map"
"xlp://isjhp1.nichibun.ac.jp:8080/nichibunken/isjhp/Heian/6/base.crd"))
(entry 124
"000345677213:isjhp1.nichibun.ac.jp:8080:/lib/maizo/iseki/h2_107_fig15-1.crd"
1
(dir 2 "xlp://isjhp1.nichibun.ac.jp:8080/lib/maizo/iseki/no6.map"
"xlp://isjhp1.nichibun.ac.jp:8080/lib/maizo/iseki/no6.crd"))
(entry 123
"000334750674:isjhp1.nichibun.ac.jp:8080:/lib/ku/1637/road.x.crd" 1
(dir 1 "xlp://isjhp1.nichibun.ac.jp:8080/lib/ku/1637/road.x.no6.map"
"xlp://isjhp1.nichibun.ac.jp:8080/lib/ku/1637/road.x.crd"))

......

(entry 56 "000345676725:isjhp1.nichibun.ac.jp:8080:/kokudo/coord/no6.crd" 1
(dir 0 "" ""))

......

(entry 5
"000334750635:isjhp1.nichibun.ac.jp:8080:/nichibunken/uno/thysen/furuold-all.crd"
1
(dir 2
"xlp://isjhp1.nichibun.ac.jp:8080/nichibunken/uno/uno1/uno1.no6.map"
"xlp://isjhp1.nichibun.ac.jp:8080/nichibunken/uno/uno1/uno1.crd"))
(entry 4
"000512015314:tois1.nichibun.ac.jp:8080:/tois2/gyouji/gyouji3.crd" 1
(dir 2 "xlp://tois1.nichibun.ac.jp:8080/kokudo/coord/no6.map"
"xlp://tois1.nichibun.ac.jp:8080/kokudo/coord/no6.crd"))
(entry 3
"000334750555:isjhp1.nichibun.ac.jp:8080:/nichibunken/uno/thysen/3new-all.crd"
1
(dir 2
"xlp://isjhp1.nichibun.ac.jp:8080/nichibunken/uno/uno1/uno1.no6.map"
"xlp://isjhp1.nichibun.ac.jp:8080/nichibunken/uno/uno1/uno1.crd"))
(entry 2
"000334750521:isjhp1.nichibun.ac.jp:8080:/nichibunken/isjhp/Heian/9/base.crd"
1
(dir 1
"xlp://isjhp1.nichibun.ac.jp:8080/nichibunken/isjhp/Heian/9/base.map"
"xlp://isjhp1.nichibun.ac.jp:8080/nichibunken/isjhp/Heian/9/base.crd"))
(entry 1
"000345677377:isjhp1.nichibun.ac.jp:8080:/lib/maizo/iseki/h2_203_fig96-3.crd"
1
```

```
(dir 2 "xlp://isjhp1.nichibun.ac.jp:8080/lib/maizo/iseki/no6.map"
"xlp://isjhp1.nichibun.ac.jp:8080/lib/maizo/iseki/no6.crd"))
(entry 0
"000412325271:tois1.nichibun.ac.jp:8080:/tois2/gyouji/gyouji6.crd" 1
(dir 2 "xlp://tois1.nichibun.ac.jp:8080/kokudo/coord/no6.map"
"xlp://tois1.nichibun.ac.jp:8080/kokudo/coord/no6.crd")))

(level 0
(entry 125
"000346676734:isjhp2.nichibun.ac.jp:8080:/kokudo/coord/06.crd" 10
(dir 1 "xlp://isjhp1.nichibun.ac.jp:8080/kokudo/coord/no6.map"
"xlp://isjhp2.nichibun.ac.jp:8080/kokudo/coord/06.crd"))
(entry 124
"000346676726:isjhp2.nichibun.ac.jp:8080:/kokudo/coord/08.crd" 11
(dir 3 "xlp://isjhp1.nichibun.ac.jp:8080/kokudo/coord/no6.map"
"xlp://isjhp2.nichibun.ac.jp:8080/kokudo/coord/06.crd"))
(entry 122
"000345677331:isjhp1.nichibun.ac.jp:8080:/lib/maizo/iseki/h2_203_fig96-1.crd"
1
(dir 2 "xlp://isjhp1.nichibun.ac.jp:8080/lib/maizo/iseki/no6.map"
"xlp://isjhp1.nichibun.ac.jp:8080/lib/maizo/iseki/no6.crd"))

.......

(entry 39 "000346676705:isjhp2.nichibun.ac.jp:8080:/kokudo/coord/15.crd" 4
(dir 3 "xlp://isjhp1.nichibun.ac.jp:8080/kokudo/coord/no6.map"
"xlp://isjhp2.nichibun.ac.jp:8080/kokudo/coord/06.crd"))
(entry 33
"000421014012:isjhp1.nichibun.ac.jp:8080:/lib/maizo/iseki/h2_203_fig96-4.crd"
1
(dir 2 "xlp://isjhp1.nichibun.ac.jp:8080/lib/maizo/iseki/no6.map"
"xlp://isjhp1.nichibun.ac.jp:8080/lib/maizo/iseki/no6.crd"))
(entry 31
"000334750035:tois1.nichibun.ac.jp:8080:/tois/gyouji/gyouji9.crd" 128
(dir 0 "" ""))
(entry 30
"000345677206:isjhp1.nichibun.ac.jp:8080:/lib/maizo/iseki/h2_108_fig18-1.crd"
1
(dir 2 "xlp://isjhp1.nichibun.ac.jp:8080/lib/maizo/iseki/no6.map"
"xlp://isjhp1.nichibun.ac.jp:8080/lib/maizo/iseki/no6.crd"))
(entry 27 "000346676713:isjhp2.nichibun.ac.jp:8080:/kokudo/coord/13.crd" 2
(dir 3 "xlp://isjhp1.nichibun.ac.jp:8080/kokudo/coord/no6.map"
"xlp://isjhp2.nichibun.ac.jp:8080/kokudo/coord/06.crd"))
(entry 25
"000345677314:isjhp1.nichibun.ac.jp:8080:/lib/maizo/iseki/h2_106_fig09-2.crd"
1
(dir 2 "xlp://isjhp1.nichibun.ac.jp:8080/lib/maizo/iseki/no6.map"
"xlp://isjhp1.nichibun.ac.jp:8080/lib/maizo/iseki/no6.crd"))

.......

(entry 3
"000345677214:isjhp1.nichibun.ac.jp:8080:/lib/maizo/iseki/h2_130_fig75-2.crd"
1
(dir 2 "xlp://isjhp1.nichibun.ac.jp:8080/lib/maizo/iseki/no6.map"
"xlp://isjhp1.nichibun.ac.jp:8080/lib/maizo/iseki/no6.crd"))
(entry 0
```

```
"000334750032:tois1.nichibun.ac.jp:8080:/tois/gyouji/gyouji11.crd" 1
(dir 2 "xlp://tois1.nichibun.ac.jp:8080/kokudo/coord/no6.map"
"xlp://tois1.nichibun.ac.jp:8080/kokudo/coord/no6.crd"))))
```

This mapping table is exchanged regularly between adjacent coordinate resources. In RIP, the routing table is exchanged among nodes via broadcast, but in GLOBALBASE, this exchange is achieved by obtaining a route selection table from the adjacent coordinate resource. This approach is taken both because our protocol does not have any broadcast mechanism, as well as in order to prevent a routing table from being sent automatically from an irrelevant resource that is not in the mapping list of the management information. Moreover, in RIP, in order to know a node adjacent to a certain node, it is necessary to scan the interface for that purpose only, which is wasteful processing, whereas GLOBALBASE allows going to the surrounding coordinate system resources directly, because mapping lists are prepared for all resources by lower layers.

The mapping path table of a coordinate system resource is obtained by providing the attribute mode="mpt" (mapping path table) to the Get expression as shown below.

```
([Get mode="mpt"] url)
Agent:    gbstd
Arguments:
    STRING url
Attributes:
    STRING mode
Return value:
    LIST (mapping path table)
```

There are cases where updates of a certain coordinate system and/or mapping should be reflected in the mapping path table as soon as possible. The following expression has been prepared for such cases.

```
(MPtrigger url)
Agent:    gbstd
Arguments:
    STRING url
Attributes:
    None
Return value:
    NULL
```

By issuing this expression to a target URL, the mapping path table is forcefully rebuilt.

Lastly, expression formats for obtaining a mapping path between two arbitrary coordinate system resources, are shown below.

```
(MProuting url1 url2)
Agent:    gbstd
Arguments:
    STRING url1
    STRING url2
Attributes:
    None
Return value:
    LIST (mapping path)

(MProuting url1 cid)
Agent:    gbstd
```

```
Arguments:
    STRING url1
    ELEMENT cid
Attributes:
    None
Return value:
    LIST (mapping path)
```

The first expression format takes two resources specified as URLs as input arguments, while the latter expression format obtains the mapping path between resources given by "url1" and "cid." "cid" is passed to the Get function using the following format.

```
(cid subaddr1 ...)
Agent:    all
Arguments:
    INTEGER subaddr1 ...
Attributes:
    STRING body
Return value:
    None
```

The address is indicated in place of "subaddr1." It is not necessary to specify "body," but doing so indicates the highest priority in the overlay network to which this resource belongs. If the body value is different for two resources, these two resources belong to different independent overlay networks and no mapping path exists between them.

The format of the mapping path returned by the operation above, is as follows.

```
(mapping coordinate mapping coordinate ....)
```

As can be seen, this is a list where URLs of mapping resources and URLs of coordinate system resources are repeated alternately. The mapping resource at the start of the list is adjacent to url1 and the last coordinate system resource is always the coordinate system specified by url2 or cid.

See src/gbs/lib/mp/routing.c and src/gbs/lib/mp/acrp.c for the source code implementing the routing algorithm on the server side, and src/gbs/lib/view/routing.c for the source code implementing the routing algorithm on the client side.

3.3.6 Object Coverage Size Indexing

When displaying a coordinate system resource accessed by a client, it is necessary to obtain information about object resources linked to this coordinate system. Accessing an object resource without knowing its position within the coordinate system is very inefficient, as it would then be necessary to obtain information of all resources at every access. For this reason, a mechanism that calculates the position of each object resource within the coordinate system on the server side in advance and stores them in the management information as indexes is introduced. These indexes can be obtained together with mapping information via the (Get mode="object") operation. The format is specified by the mapping layer.

The server regularly checks coordinate system resources on it to see whether or not any mapped objects have been changed. If an object has been changed, the server obtains the smallest rectangle encompassing the object, called the minimum rectangle, converts it via the associated mapping to a corresponding rectangle in the coordinate system, and registers this information as an index.

3.3.7 Partial Search Engine (PSE) (Reference Models (7) and (9))

Overview

This layer provides a service that determines whether or not coordinate system resources that match with the search criteria of a given client exist within the currently displayed range. This search engine

is a collection of small search engines called lumps, each of which searches within a limited range. It is thus not a big server that collects information in a centralized manner; rather, it is a distributed search engine, where a client obtains information by moving from one lump to another.

There are two types of lumps, each type collecting a separate type of data. The first type of lump collects bibliographical information assigned to each resource and creates an index of this information; it is called a plate metadata database (PMD). The other type of lump searches through the contents of each resource for the purpose of finding the feature name such as place names and building names on a map; it is called a feature metadata database (FMD). Currently, only the PMD type is implemented.

Lump Generation

In most cases, one lump exists on each server. However, several lumps can exist on one server or several servers can share one lump, depending on the range in which information is collected. Each lump is connected to one coordinate system resource residing on the server on which it exists. Basically, it has the same URL as the coordinate resource.

In normal search engines, resources do not register themselves in a search engine on their own. In case of PSE, however, each resource registers itself within appropriate search engines. For this reason, the PSE can be updated immediately whenever a new resource is loaded into a server or a resource is deleted from a server.

In order to implement this mechanism, each coordinate system resource traces the overlay network and stores the nearest five lumps in its management information by itself. This is called lump information. The coordinate system resource then registers its bibliographical information to these five lumps. The lump information has the following format.

```
<?xl .... ?>
<lump-info>
[
 <option>
[ <lump> "lump-path" </lump>]
[ <destroy> time </destroy>]
[ <launch> time </laumch>]
[ <fade> time interval </fade>]
 </option>
]
  <entry> "lump-crd" "crd" pri hops hops-max </entry>
[ <entry> "lump-crd" "crd" pri hops hops-max </entry>
  <entry> "lump-crd" "crd" pri hops hops-max </entry>
  <entry> "lump-crd" "crd" pri hops hops-max </entry>
  <entry> "lump-crd" "crd" pri hops hops-max </entry>
]
</lump-info>
[] can be omitted.
```

The entry element indicates the five lumps (lump-crd) closest to this coordinate system and the coordinate system (crd) adjacent to it in the direction where each of the lumps exists. "pri" is the priority of each of the lumps (the time when the lump was generated), "hops" is the number of hops required to reach the lump, and "hops-max" is the maximum number of hops. Lumps farther than "hops-max" cannot be seen from this coordinate system.

"hops-max" is set by the lump itself and is set to a large value when the capacity of the lump is large and a large amount of information can be stored. The older the time of generation, the higher the value of "pri" becomes.

The lump element within the option tags indicates that a lump exists for the coordinate system and the contents of the element provides the path to the lump.

The destroy element is specified when it is judged that there are many lumps in the vicinity of the coordinate system that the lump is attached to and indicates the time at which this lump is destroyed.

If it is judged that the density of surrounding lumps has become lower by this time, the destroy element is deleted.

The launch element, on the contrary, is specified when the density of surrounding lumps is too low in the vicinity of a coordinate system without any lump attached, and it is necessary to create new lumps. The element indicates the time at which a new lump is launched. The times specified in the "destroy" and "launch" elements are set with some margins, rather than immediately after the judgment.

The fade element indicates the time interval at which to acquire the surrounding conditions again after a lump was launched or destroyed, or the surrounding conditions have changed and the contents of the entry element have changed. The value of this time interval gradually becomes larger, and the checking caused by "fade" is ended when the value reaches a set value.

The following shows two examples of lump information. i. An example of a coordinate system without a lump

```
([?xl encoding="EUC-JP" version="0.1"])
(lump-info
(entry
"xlp://vkyoto.sd.docomo-kansai.co.jp:8080/v-kyoto/kyoto-map/kyoto.crd"
"xlp://isjhp2.nichibun.ac.jp:8080/kokudo/coord/06.crd" 83138304 3 9)
(entry "xlp://isjhp2.nichibun.ac.jp:8080/kokudo20000/coord/05.crd"
"xlp://isjhp2.nichibun.ac.jp:8080/kokudo/coord/06.crd" 77024699 3 9)
(entry "xlp://tois1.nichibun.ac.jp:8080/tois/gyouji/gyouji11.crd"
"xlp://tois1.nichibun.ac.jp:8080/kokudo/coord/no6.crd" 57921589 2 9)
(entry "xlp://gbs.kyoto-archives.gr.jp:8080/kyoto2500/kyoto2500.crd"
"xlp://isjhp2.nichibun.ac.jp:8080/kokudo/coord/06.crd" 87731039 2 9)
(entry
"xlp://isjhp1.nichibun.ac.jp:8080/nichibunken/isjhp/Heian/4/base.crd"
"xlp://isjhp1.nichibun.ac.jp:8080/nichibunken/isjhp/Heian/4/base.crd"
57921893 1 9))

ii. An example of a coordinate system with a lump

([?xl encoding="EUC-JP" version="0.1"])
(lump-info
(option
(lump "/work/lump/0/0/0/246/"))
(entry
"xlp://vkyoto.sd.docomo-kansai.co.jp:8080/v-kyoto/kyoto-map/kyoto.crd"
"xlp://isjhp1.nichibun.ac.jp:8080/kokudo/coord/no6.crd" 83138304 4 9)
(entry "xlp://isjhp2.nichibun.ac.jp:8080/kokudo20000/coord/05.crd"
"xlp://isjhp1.nichibun.ac.jp:8080/kokudo/coord/no6.crd" 77024699 4 9)
(entry "xlp://tois1.nichibun.ac.jp:8080/tois/gyouji/gyouji11.crd"
"xlp://isjhp1.nichibun.ac.jp:8080/kokudo/coord/no6.crd" 57921589 3 9)
(entry "xlp://gbs.kyoto-archives.gr.jp:8080/kyoto2500/kyoto2500.crd"
"xlp://isjhp1.nichibun.ac.jp:8080/kokudo/coord/no6.crd" 87731039 3 9)
(entry
"xlp://isjhp1.nichibun.ac.jp:8080/nichibunken/isjhp/Heian/4/base.crd" ""
57921893 0 9))
```

This format is the same as for some of the routing information; the only difference is the inclusion of the number of hops and the path to the lump. A coordinate system resource obtains and merges surrounding lump information regularly, placing higher priority on the direction with the minimum number of hops to reach the same lump. This algorithm is thus an implementation of the distributed Dijkstra shortest path algorithm. If the coordinate system resource finds information of more than six lumps, the information is deleted and not stored as management information.

A coordinate system resource with a lump, naturally, knows the positions of four lumps in addition to that of its own lump. If the positions of these four lumps are too close to the position of its own

lump, it destroys its own lump. Conversely, if a coordinate system resource without a lump is positioned too far away from the nearest surrounding lumps, it generates a lump attached to itself. There are various algorithms for judging the limit of what should be regarded as too far or too close. In the current implementation, a lump is generated if no lump exists in the vicinity on the server on which the coordinate system resource resides. With this algorithm, however, too many lumps may be generated if there are many servers with few resources each, which can be a problem.

See src/gbs/mp/lump.c for the source code dealing with lump management.

Metadata Registering

When a resource is updated, it looks up lumps in the vicinity. This operation is not performed if no bibliographical information is attached, however, since there is no information to be registered.

A coordinate system resource can have information of up to five lumps stored in its own lump information; it registers its bibliographical information to these lumps. In case of other types of resources, a resource refers to the lump information of the coordinate system resource to which it is mapped. It then sends a trigger request to the lump found, i.e., sends the following expression to the server on which the lump exists.

```
(MPTrigger path url)
Agent: gbpmd
Arguments:
    STRING path
    STRING url
Attributes:
Return value:
    INTEGER interval
```

"path" is the path to a lump. "url" is the URL of the resource that is requesting the registration. The triggered lump registers the target "url" in its queue, accesses resources in order with ([Get type="meta"]..) and creates index information from the bibliographical information contained within the metadata obtained. The return value is the appropriate registration polling interval obtained by calculating CPU load at registration. The resource that requested the registration repeats transmitting the MPTrigger expression at this registration interval, even if its data is not updated.

Lumps remember when each of the indexed bibliographical data items was polled. If a given item has not been polled for after four times the polling interval has elapsed, the information is regarded as lost and deleted.

Although it may depend on the specific implementation, there are the following general phases involved in the registration operation of a lump.

a) Checking format of bibliographical information

A lump acquires the metadata of the URL to be registered and checks that the obtained metadata has the correct format. If the format is wrong or the metadata itself is not found, the subsequent registration operation is not performed.

b) Coordinate conversion

A lump searches for the mapping path between the URL to be registered and the coordinate system to which it is attached. It converts the minimum rectangle and valid resolution range of the resource to be registered to the corresponding coordinate values of the minimum rectangle within the coordinate system to which the lump is attached, which is then registered as the minimum rectangle of the resource.

c) Registration of indexes

Once the processes above are finished for a resource, indexes are assigned to all the metadata and registered. It is important to create indexes for the minimum rectangle and valid resolution range in such a way that they can be searched for quickly and efficiently.

See the source code of gbpmd in src/gbs/pmd for the implementation of registration of data to a lump.

Searching Query

The following expression is used to search for resources registered in a lump.

```
(PMDquery db rect reso-min reso-max filter)
Agent: gbpmd.get
Arguments:
    STRING db
    LIST rect
    FLOATING/INTEGER reso-min
    FLOATING/INTEGER reso-max
    LIST filter
Attributes:
    format long or short (short if omitted)
Return value:
    LIST Search result list
```

The features of this search query is that the geographical range (rect) and resolution (reso-min/reso-max) of the search are specified. They correspond to the range displayed in a browser and its resolution. "rect" is a list of two points, with each point being a list of two numerical values, specifying the coordinates of the points at which the x and y coordinate values of a rectangle become the minimum and maximum, respectively. That is, a rectangle is specified using the following format.

```
((0.0km 1.5km) (100km 20km))
```

When target resources is narrowed down in this way, the bibliographical information matching the data specified by a filter is acquired. The format of the filter is described as a combination of qualifier and URL functions with AND, OR and NOT operators, for instance as follows.

```
(OR (AND
    ([qualifier cond="part"]
        "xlp://isjhp1.nichibun.ac.jp:8080/gb_metadata"
        0
        "property"
        ()
        "base")
    ([qualifier cond="boundary"]
        "xlp://isjhp1.nichibun.ac.jp:8080/gb_metadata"
        0
        "content.period"
        "W3C-DTF"
        "2003-10-27 / 2003-10-27"))
    (URL "xlp://isjhp2.nichibun.ac.jp:8080/world/00.crd"))))
```

The structure of each function is defined as follows.

```
(AND ...)
Agent: gbpmd.get
Arguments:
    gbpmd.get Type specified within an agent
Attributes:
Return value:
    gbpmd.get Specified within an agent

(OR ...)
Agent: gbpmd.get
```

```
Arguments:
    gbpmd.get Type specified within an agent
Attributes:
Return value:
    gbpmd.get Specified within an agent

(NOT query)
Agent: gbpmd.get
Arguments:
    gbpmd.get Type specified within an agent
Attributes:
    cond part,boundary, match (match if omitted)
Return value:
    gbpmd.get Specified within an agent

(qualifier namespace inheritance name type data)
Agent: gbpmd.get
Arguments:
    STRING namespace
    INTEGER inheritance
    STRING name
    STRING type
    STRING data
Attributes:
    cond part, boundary, match (match if omitted)
Return value:
    gbpmd.get Specified within an agent

(URL url)
Agent: gbpmd.get
Arguments:
    STRING url
Attributes:
    cond part, boundary, match (match if omitted)
Return value:
    gbpmd.get Specified within an agent
```

There are three types of result resources lists: a list consisting only of URLs (format="short"), a list containing ranges and flags indicating which conditions are matched (format="middle") in addition to the above, and a list containing all data including biographical information (format="long").

Chapter 4

Security Consideration

4.1 Abstract

4.2 Technical Security – Address Storm

This architecture has just been created and, to be honest, nothing has been established with respect to the security.

There are two types of security that should be built into the architecture. The first type of security is purely technical, i.e., protection of the XL protocol so that it is not cracked and server data is not stolen or tampered with. The other type of security relates to human factor of management of the system (s) and contents of GLOBALBASE, such that image or map data that threatens human rights or gives people unpleasant feelings can be prevented from being published.

A frequently asked question regarding the technical security is that of ACRP. Since it automatically assigns addresses to resources according to the surrounding conditions, it can be feared that a large number of resources with addresses that overlap with resources provided by other information providers may be generated intentionally to force the address arbitration protocol to reassign the overlapping addresses and drive the network into congestion. Such an attack will be called an address storm here.

A situation similar to an address storm may also be generated unintentionally. If two big overlay networks are integrated into one overlay network, their addresses will inevitably overlap. In this case, it may be necessary to take a measure to assign addresses of different values to the most significant digit of each of the overlay networks to prevent the address storm. When we experimented with the actual implementation, however, it appears that there is little effect, because in actuality it is possible to search for mapping paths even if some of the addresses are not assigned.

To generate an address storm intentionally, it is necessary to simulate or create an overlay network of the same size as the current overlay network. Moreover, since addresses are changed dynamically to prevent duplication, it is necessary to track the changes and create several overlay networks of the same scale, one after another. This is not an easy task.

This, however, does not mean that there is no possibility of occurrence of address storms. It will be necessary to investigate the possibility of occurrence of address storms through various simulations.

4.3 Management Security

Next, security against delivery of unpleasant information is considered. With the current GLOBALBASE architecture, anyone can launch a server and provide geographical information as far as one has the technical capabilities, and map and overlap any geographical information one wishes to provide with geographical information provided by other people.

The original purpose of GLOBALBASE is to share geographical information in this way. It is not impossible, for example, to create software that automatically places signs at 1km intervals all over the virtual world, thus abusing GLOBALBASE. Fortunately, doing so requires considerable technical capabilities and, even if one succeeds, there is no effect to be gained. Such signs will be removed by people who think they are unpleasant. For this reason, we have not put any effort into implementing a mechanism to regulate such behaviors in the protocol.

However, as GLOBALBASE is diffused and becomes popular in the future, there will most likely be people who will want to disrupt or take over the virtual Earth. If such things happen, then, we who joined the GLOBALBASE project can discuss how to prevent them.

There will be many technologies available for preventing such problems, such as employment of a license system for linking and delivering resources. The essential issue that remains is how we define unpleasant information and who should regulate it. There is no way that such a line can be drawn on a global scale. Such a line can only be determined by people who joined in the actual project management of this virtual space called GLOBALBASE.

This problem is similar to a problem that occurs on the actual Earth. Originally, the collection of lands of the Earth did not belong to anyone. Humans appeared to live there and started to own lands for living; basically, a piece of land belonged to the person who cultivated it first. However, as the population grew and many people required lands, conflicts over ownership of lands started to occur. As a means to prevent such conflicts, laws and agreements were made to make people's behaviors toward each other more civil. Such laws and agreements, however, are neither perfect nor universally obeyed; we see many images of ugly, dismal conflicts broadcast on the TV news every day.

GLOBALBASE is currently in a phase equivalent to the initial period of the Earth. We are the first people who colonized here. We have a space too big to handle by ourselves now. It is an homogeneous, bleak space where there is no culture or religion. We expect that people with various cultures will start to live here sooner or later. As we have feelings, it is natural that there will be conflicts and other problems. We hope to take measures to reconcile such problems in a democratic manner and avoid making the same mistakes as were made on the actual Earth. The dream we pursue is to create an Earth that provides more freedom than the actual Earth – if such a thing is possible.

Chapter 5

References

5.1 Abstract

5.2 References

- [1] Hirohisa Mori "GLOBALBASE PROJECT" Can be downloaded from http://www.globalbase.org/
- [2] Hirohisa Mori, Ken Sakamura "A New Network Layer Protocol with Routing Addresses and Tables Auto-Configuration Mechanism" in proc. of ICDCS, IEEE, 2000,pp. 84-96.

Bibliography

History

1. **Date:** 2007-11-04
   Generatting this manual。(Edition 2007-11-04)
2. **Date:** 2007-08-01
   Author: Hirohisa MORI **TargetVersion:** ver.A.
   Starting to write this manual.