第 XXI 部

Integrated Distributed Environment with Overlay Network

第 21 部
Integrated Distributed Environment with Overlay Network

第 1 章 Activities of IDEON WG in FY2007

1.1 Introduction

IDEON (Integrated Distributed Environment with Overlay Network) is a working group of researchers who try to approach realization of the integrated distributed environment (IDE) through construction of overlay networks (ON).

IDEON focuses on research, development and operation of overlay networks as an infrastructure to realize free and creative rendezvous, location and routing. IDEON members are encouraged to actually live on the infrastructure to verify their designs.

Research topics of IDEON include, but not limited to, the following:

- Application-layer multicast
- Operable distributed hash tables
- Self-sustained trust management for distributed autonomous systems

For the detailed plans of specific projects, please refer to the web pages of each project.

- `http://member.wide.ad.jp/wg/ideon/?en%2FProjects` (English)
- `http://member.wide.ad.jp/wg/ideon/?ja%2FProjects` (Japanese)

Activities in IDEON can be outlined as follows:

$\{understand \rightarrow build \rightarrow try\}* \rightarrow deploy \mid live$

where '$*$' denotes repetition and '$|$' denotes concurrency.

Currently, all our development projects are in *deploy* phase. For example, it is our goal that the number of users of *wija* (messaging platform we have developed) and its major plug-in *i*-WAT will reach over 20,000 people by the end of August 2008.

1.2 Summary of Activities

In this fiscal year, we have discussed on realization of Global OS and the programming language design for overlay networks in annual IDEON retreat. The discussion continued throughout the year, using the BoF slots of WIDE meetings and the camp.

We have reviewed the software design of *wija*, an instant messaging software as a prototype of Global OS Shell.

We have also investigated on ID resolution mechanism by DHT-DNS mounter.

1.3 Glossary

Table 1.1 shows the glossary for IDEON.

Table 1.1. Glossary for IDEON

Free	Having no restriction whatsoever as to with which peers one can communicate.
Creative	Being able to select a set of peers according to one's objectives, requirements, needs and contexts so that communication becomes most valuable for the participants.
Rendezvous	To identify such a peer.
Location	To locate such a peer in the overlay networks by the acquired identifier.
Routing	To deliver a message to such a peer on the acquired location.
Overlay Network	An application-specific virtual network of peers over the IP network to realize rendezvous, location and routing over an appropriate abstraction of entities.

第 2 章　IDEON Retreat 2007 Spring

2.1 Overview

IDEON WG members had an annual two-day meeting in April, 2007, this time at Shonan Fujisawa Campus, Keio University.

The objective of the meeting was to discuss on overlay network programming, in particular, how we realize Global Operating System with such a programming methodology.

We discussed on the concept of Global OS and how it can be applied to real problems, and tried out the initial implementation of GHC (Guarded Horn Clauses) that works on *wija*.

2.2 Global Operating System Concept

An operating system abstracts resources, and provides them to users.

Today, personal computers have ample resources, everyone carries portable computing/communicating devices, and RFID/micro computers are to embed intelligence everywhere in our living environment. If we think about an operating system which scales to the global level, it would be a super WIDE, super widely integrated distributed environment that self-organizes in autonomous, distributed and cooperative ways, which would overthrow conventional operating system designs.

CPU, memory, disk storage, network bandwidth, keyboards, displays, sensors, actuators, software, image, sound, documents, know-hows, automobiles and their seats, clothing, food, human beings and their talents, abilities and efforts are all resources which will be provided to users in need at the right time and in the right place to be used efficiently. A new information environment will be achieved.

This kind of information environment will enrich our lives, reduce energy consumption and form local-production-local-consumption economy as a new foundation for our lives, which is robust and helps us live in accordance with the natural environment of the 21st century.

2.3 Discussion

2.3.1 Day One

Question: What is a Global OS?

We thought in terms of benchmarks. The following is how we classified the application of Global OS.

1. Resource discovery and allocation (rendezvous)
   - Discovery and use of remote meeting space (n-party; human being → geological location)
   - Hitchhike (geological constraints → human)
2. Sensing and action
   - Wide-area fault analysis (collect data from a sensor grid)
   - Planting at deserts (employing youngsters with time but without money)
3. Sharing computing resources
   - Distributed storage
   - Distributed CPU[1]

We decided to hold a hacking convention at WIDE meeting/camp on hitchhikers problem.

- Automobiles can be simulated, but the system must work.
- Call for participation in May (at the WIDE meeting)
  - Easy problem: e.g.) Simple ALM → hacking convention at BoF
  - Problem at WIDE Camp: Hitchhikers problem

Benchmarks were to be actually challenged in the latter half of this fiscal year.

Keywords

- Multi-paradigm language

1　This quite literally turned into a 4 bit CPU simulator written in Overlay GHC later this year; the simulator lets multiple computers on an overlay network form a single CPU.

References

- K. Furukawa, F. Mizoguchi (Ed.) “並列論理型言語 GHC とその応用”
- Bruce Sterling “Maneki Neko”

2.3.2 Day Two

Practice

- Concurrent Logic Language GHC
- OverLog (partially)

2.4 Outcome of the Retreat

2.4.1 To-Do

We realized at the IDEON Retreat that we do not have means for freely programming the distributed system called Global OS.

Global OS needs to have means for directly programming distributed systems. However, we are not yet certain about the design of the necessary system for that. Therefore, we are going to build a prototype of a Global OS using the problem below as a benchmark.

2.4.2 Hitchhikers Problem

An unoccupied seat of a running automobile is a resource which can be shared by the humankind. Program an overlay network so that a pedestrian in need can discover the resource, and the driver can discover the pedestrian, which makes a hitchhike to the intended destination possible.

Keywords: mobility, geography, rendezvous, trust

Overlay GHC is an available language for programming.

2.4.3 Overlay GHC

- Specification/implementation of an extension of GHC to have control designed for overlay network programming, which can be used over the XMPP network of *wija*
- Bundled in the main distribution of *wija* since version 0.13.
- Latest version is available at the following URL: `http://www2.media-art-online.org/nightly/`

第 3 章 wija: an Open and Extensible Messaging Platform on the Internet

3.1 Introduction

3.1.1 Purpose of Development

Our purpose is to assure freedom of experimentation for researchers, software developers and anyone in pursuit of problems.

Openness, autonomy, extensibility, and resulted dependability/sustainability are the basic properties of the Internet, which we would like to enhance. Although IP provides global reachability as a foundation for freely proposing, testing and deploying new services, it is not an easy tool for creation to be used by general public. We are to provide an open platform on top of TCP/IP, or another Internet over IP, at a level closer to human beings and human relations, so that anyone can participate in creation of new communication on the Internet.

For this purpose, we have developed *wija* as a platform for experimenting with all possibilities of communication on the Internet. This statement is not an exaggeration. The basic semantics of IP is that a message is reachable to a destination specified by an identifier/locator. A communication protocol implementing this semantics should be able to cover all possibilities of the Internet. This is why we have chosen to use an instant messaging protocol in our development, which implements the basic semantics of the Internet at a human level.

3.1.2 Mission Statement

We have set forth the following mission statement:

M-1: Messaging Platform

We design *wija* to be a 1) cross-platform, 2) internationalized and 3) interoperable messaging software, 4) with means for extending its functionalities.

The need for interoperability would require *wija* to conform to available centralized messaging protocols. But centralization deprives users and developers of freedom of innovation. Centralization also have negative effects on dependability and sustainability, because when a server is down, there is nothing outsiders of the domain can do for recovering the services.

The design of *wija* should shift toward decentralization.

M-2: End-to-Endness

We design *wija* to be end-to-end oriented, so that problems in new applications are not to be solved by adding new features to servers, but by communication among clients.

In the end, it should eliminate all dependencies on servers for maximal autonomy, dependability and sustainability. This makes safe communication among clients particularly important.

M-3: Security

We design *wija* to be integrated with a foundation for secure communication, which must provide end-to-end public key cryptography. *wija* should provide easy-to-use human interface for handling keys and certificates, and encrypting/signing messages.

M-4: Freedom

We develop *wija* as a free software, without restrictions for further innovation by any parties.

3.1.3 Challenges

There are several challenges for completing the above missions:

C-1: Pathway to P2P (peer-to-peer)

One challenge is to set the pathway in the development for future elimination of dependencies on servers.

In addition to moving from centralized to more P2P-oriented protocols, we should also consider how we can eliminate dependencies on the web in distributing software and providing support for users and developers.

C-2: User Friendliness

Another challenge is to design an easy-to-use integration with the foundation for secure communication so that anyone can handle public key cryptography.

3.2 Background — Jabber/XMPP

3.2.1 History

Jabber/XMPP is a set of open protocols for instant messaging and presence sharing.

It was first proposed as *Jabber* protocol in 1998. The first version of an open source Jabber server, *jabberd*, was released in year 2000. JSF (Jabber Software Foundation) was established in 2001 to maintain the specifications and to support their enhancements. In 2004, the core protocols of Jabber were standardized by IETF[167] as Extensible Messaging and Presence Protocol (hence the name Jabber/XMPP).

Jabber/XMPP has been applied to many instant messaging systems including those developed by Apple, FedEx, Google, Oracle and Sun Microsystems.

3.2.2 Characteristics

The characteristics of Jabber/XMPP in comparison with other instant messaging systems are as follows:

1. *Openness*

   Specifications of Jabber/XMPP are open to public. Anyone can implement the protocols for free. Readers can find lists of clients and servers at [75].

2. *Autonomy*

   The network of Jabber/XMPP, as illustrated in Fig. 3.1, is close to that of SMTP[131] (Simple Mail Transfer Protocol); mail servers can be set up by anyone without permissions from any parties once a domain name is registered, and anyone can freely join the network of the global electronic mail system. Likewise, XMPP servers can be set up without any permissions, and anyone can freely join

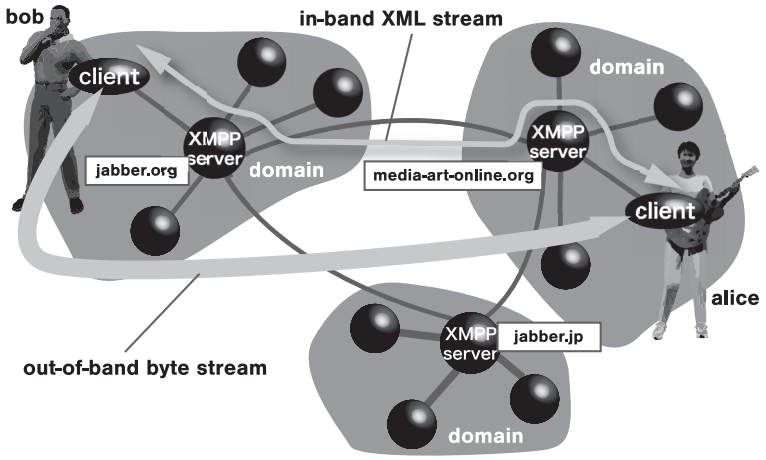


Fig. 3.1. Overview of XMPP communication

the global network of the instant messaging and presence sharing system.

3. *Extensibility*
   Jabber/XMPP is an extensible set of protocols whose data descriptions are based on XML[20]. New features can be incorporated without breaking the existing system. Extended protocols can be submitted to XSF (XMPP Standards Foundation) as an XEP (XMPP Extension Protocols) for standardization. An XEP is approved as a standard after feedbacks from implementers and reviews from XSF.

3.2.3 Communication Mechanism

In Jabber/XMPP, a peer is identified by a Jabber ID of the following form:

user@*domain*/*resource*

A *user* name defines a user in a domain. A *domain* name is effectively the host name of the server to which the user is connected with a client. A *resource* name distinguishes a communication session from others by the same user in the same domain.

Fig. 3.1 illustrates the communication mechanism of Jabber/XMPP.

Like SMTP, clients need to talk to a server in order to have their messages reach the clients on the other ends (*in-band XML stream*). It implies that Jabber/XMPP has such a weakness that communication becomes impossible when either of the servers in between is down. Therefore, it is our intention that it will be replaced by a more P2P oriented technology in the future to achieve the level of dependability and sustainability required for our purposes.

Jabber/XMPP also provides means for the clients to directly communicate (*out-of-band byte stream*), using protocols such as SOCKS5[91]. It is recommended that clients use out-of-band communication whenever a large data is involved. Still, in-band communication takes an important role in identifying the peers by their IP addresses and port numbers, which can dynamically change; because of this, Jabber/XMPP provides an excellent way to rendezvous for applications on the Internet.

3.3 Core Design

3.3.1 Primary Decisions

3.3.1.1 Development Language

We have chosen Java[160] as the development language for *wija*. We did it mainly for ease of implementing cross-platform and internationalized software. In addition, we hoped that popularity of the language would attract more developers.

A drawback of selecting Java for a client-side application is the necessity for users to prepare the runtime environment. Although this has never

been a problem for Mac OS X[9], on which Java 2 Standard Edition is pre-installed, and increasingly less problematic for Linux[95] platforms because of GCJ[46] (The GNU Compiler for Java) and GNU Classpath[47], this has always been a concern for Windows[107]. Section 3.6.2.1 discusses whether this decision has affected user distribution.

3.3.1.2 Communication Protocol

We have chosen Jabber/XMPP as the communication protocol to start with, and then shift toward a more P2P-oriented protocol for enhanced dependability and sustainability.

Many available instant messaging services today, such as MSN[108], AIM[7], Yahoo![186] and Skype[150] are proprietary. They may become interoperable with one another, but even then, freedom of innovation will not be ensured for general public.

We believe that Jabber/XMPP is the best available choice for the instant messaging protocol for our purposes because of its openness, autonomy and extensibility as described in section 3.2.2. It also provides a foundation for experimenting with P2P protocols using out-of-band byte streams.

3.3.1.3 Policy on Conforming to Standards

Conformity is important for assuring interoperability, and attracting potential users of existing messaging systems to use *wija*. But at the same time, *wija* should be inherently experimental.

Our policy is that the end-to-end principle (M-2) should take higher priority over conformity to existing protocols, unless the protocol is too popular in practice to ignore, in which case lack of interoperability is considered more harmful.

3.3.1.4 Cryptographic Framework

We have chosen OpenPGP[22] as the cryptographic framework because of its end-to-endness implied from the web of trust[165], and GnuPG[166] as its implementation because it is a free software. We have designed *wija* to be integrated with GnuPG (version 1.x), and to have a GUI front-end for handling all PGP operations.

It does not mean that *wija* ignores X.509[64]. Jabber/XMPP allows communication between a client and a server, and between a server and another server encrypted by TLS[35] (Transport Layer Security). Without supporting this, *wija* would lose compatibility with popular XMPP services as the one provided from Google, Google Talk[50].

By the above described policy, we have implemented TLS connection with XMPP servers using Java security framework, so that users can log into Google Talk using *wija* (a big advantage to us for increasing the number of potential users).

By the same policy, we have designed a direct end-to-end PGP public key exchange protocol over Jabber/XMPP as described in section 3.4.3.2, in addition to key exchange using public key servers.

3.3.1.5 Distribution License

We have chosen GNU GPL[48] version 3 as the license under which *wija* is distributed.

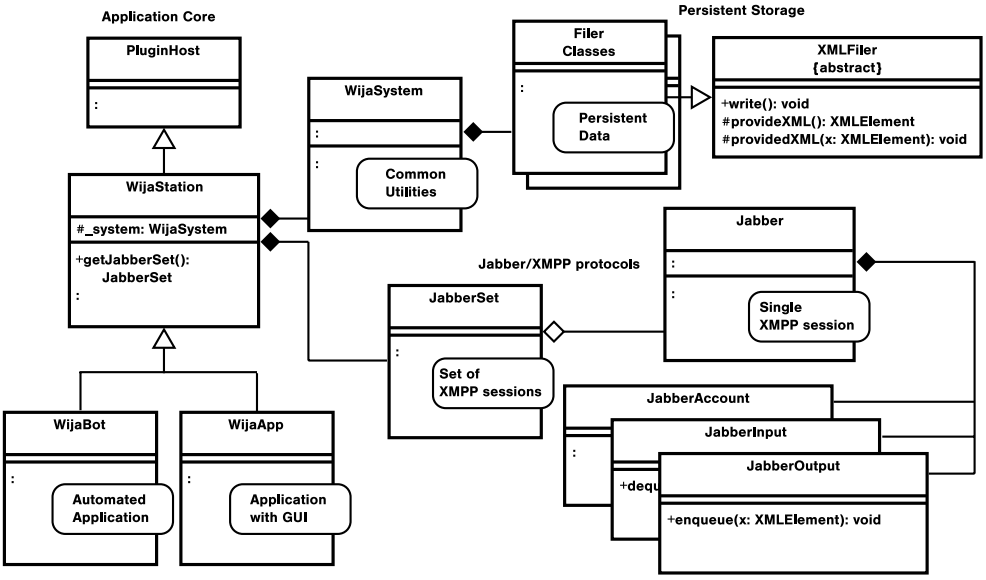
3.3.2 Core Components

Fig. 3.2 is a UML[127] (Unified Modeling Language) class diagram for core components of *wija*.

We have developed software *robot* version of *wija*, which we call *wijabot*. *wija* and *wijabot* share most features including hypertext sharing (section 3.4.2.1) and PGP public key exchange (section 3.4.3.2), but *wijabot* does not have direct human interface while running. *wijabot* is extensible with plug-ins with the same architecture (section 3.4.1.4) as *wija* to implement necessary automation to fit the purposes. *wijabot* has been used for providing services such as SOCKS5 proxy (section 3.4.2.2) which do not require human intervention.

Java classes of *wija* are designed in such a way that *wija* and *wijabot* can share most of the code.

As a general rule, we have a class in the package of *wija* (*org.media_art_online.wija*) that



- *PluginHost* is an abstraction of a program that can be extended by plug-ins. A reference to the *PluginHost* object is passed to each plug-in when the plug-in starts.
- *wija* (implemented as a *WijaApp* object) and *wijabot* (implemented as a *WijaBot* object) share core components with respect to communication and storage.
- Since version 0.13, *wija* supports simultaneous multiple XMPP sessions with different Jabber IDs. All accesses to XMPP protocols need to go through *JabberSet* object that provides search functions for the particular XMPP session in question.

Fig. 3.2. Core components of *wija* in UML class diagram

implements a feature required in *wijabot* (e.g. *WijaStation*), and a separate class provides user interface to be used in *wija* (e.g. *WijaApp*). Compared to *wija*, only 3% of code was needed to implement *wijabot*.

3.4 Design for Missions

3.4.1 Design for M-1: Messaging Platform

3.4.1.1 Cross-platform Support

We have successfully made *wija* run on major operating system platforms such as Linux, Mac OS X and Windows by writing the code in a platform-independent manner; differences among platforms not concealed by Java are mostly abstracted within the core components. Fig. 3.3 shows a screenshot of *wija* and its plug-ins on Mac OS X.

There was a challenge in providing a cross-platform mechanism to link external plug-ins with *wija*. Our determination has been that all executable files including those of external plug-ins must be able to run on any platforms. Otherwise, we would need more efforts on supporting multiple platforms, and features such as secure P2P update (section 3.4.2.3) would have limited applications.

This linkage has been done by class loaders provided by the Java virtual machines. The mechanism requires the file name of the JAR (Java ARchive) file of a plug-in to be named as follows:

full-name-of-the-class.jar

For example, the plug-in file for *i*-WAT[106, 141] is named "`org.media_art_online.iwat.Iwat.jar`".

Upon start-up, *wija* searches for those files under the *plugins* directory inside the program directory. When it finds one, it tries to load the class specified by the file name from the JAR file, which will result in loading all available referred classes recursively.

An alternative is to store the name of the class as a property of the plug-in JAR file, but our design provides simpler build process and retrieval of the class at runtime.

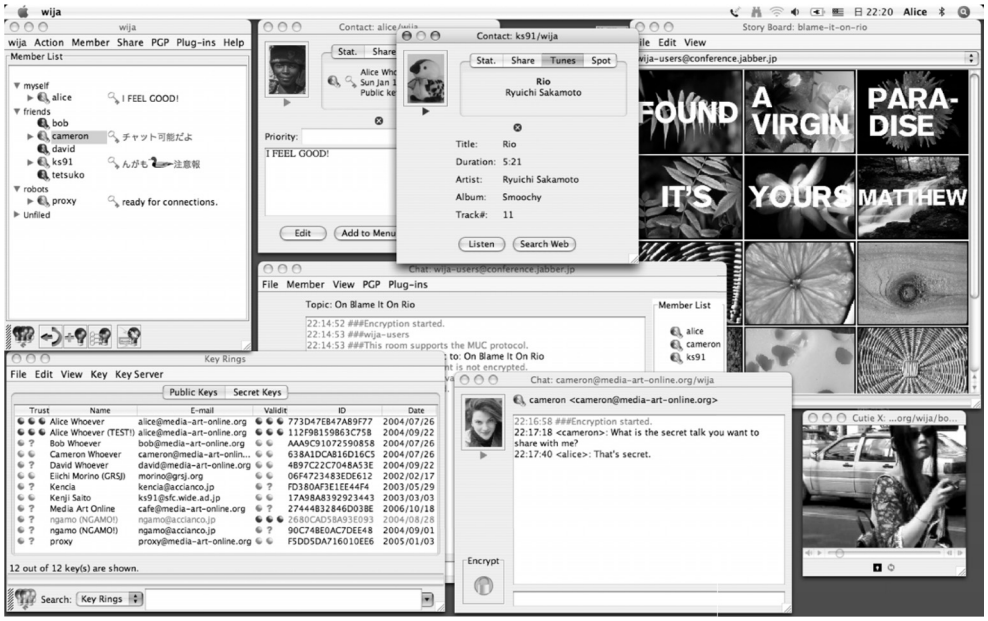


Fig. 3.3. Screenshot of *wija* and its plug-ins

3.4.1.2 Internationalization

wija utilizes localization support of Java for internationalization. Currently all messages to users are provided in two languages: English and Japanese. Developers can add properties files to support other languages.

3.4.1.3 Interoperability

Table 3.1 shows the list of XEPs implemented in *wija* to assure interoperability with other Jabber/XMPP messaging clients.

wija implements XEP-0115: *Entity Capabilities* so that a capability list is sent with presence from others, which is stored locally, and checked before *wija* or its plug-ins try to send any non-standard messages to the peer. In this way, *wija* can avoid obstructing communication when it is put among other clients in the instant messaging network.

Alternatively, we could use XEP-0030: *Service Discovery* to discover features supported by peers, but it would require more messages and code.

Table 3.1. XEPs Implemented in *wija*

XEP #	*Name*
XEP-0020	Feature Negotiation[110]
XEP-0027	Current Jabber OpenPGP Usage[118]
XEP-0030	Service Discovery[59]
XEP-0045	Multi-User Chat[137]
XEP-0047	In-Band Bytestreams[80]
XEP-0065	SOCKS5 Bytestreams[151]
XEP-0082	Jabber Date and Time Profiles[138]
XEP-0086	Error Condition Mappings[125]
XEP-0095	Stream Initiation[119]
XEP-0096	File Transfer[120]
XEP-0115	Entity Capabilities[60]
XEP-0153	vCard-Based Avatars[139]

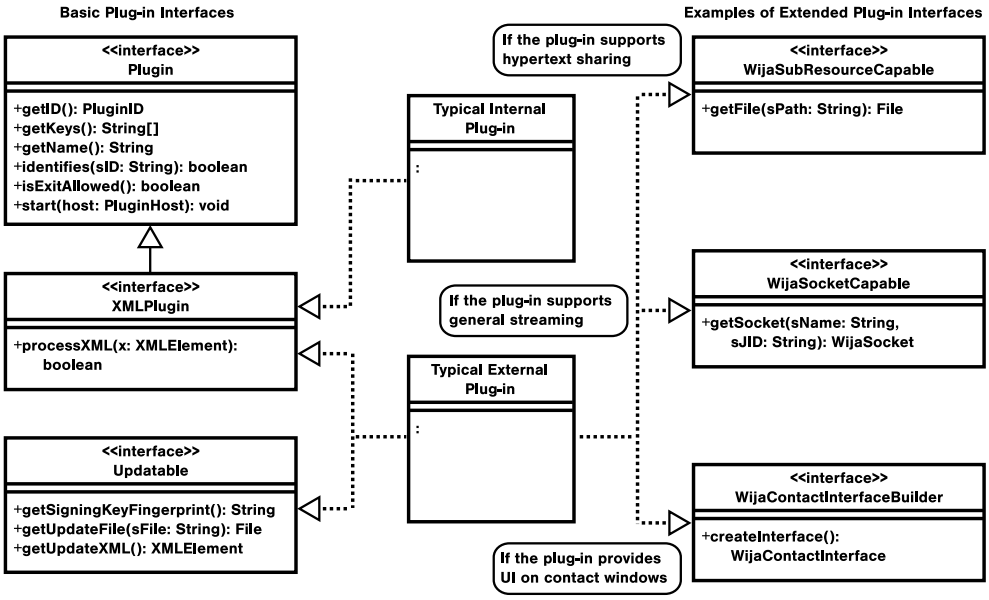
3.4.1.4 Means for Extension

wija allows new functionality to be added as a plug-in. Fig. 3.4 is a UML class diagram for plug-in APIs of *wija*.

Many XEPs, such as XEP-0045: *Multi-User Chat* and XEP-0047: *In-Band Bytestreams*, are implemented as *internal* plug-ins using the plug-in APIs, which has eased incremental development of *wija*. Those plug-ins are included in the executable file of *wija*, so that they must not implement *Updatable* that specifies external executable files to update. This is the reason why basic plug-in APIs have separated interfaces.

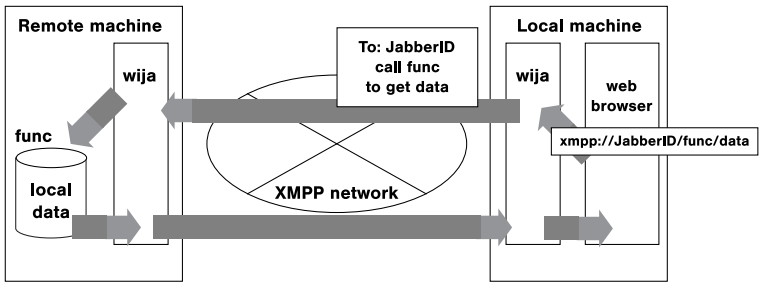
There are various extended plug-in interfaces that plug-ins may implement to provide certain types of functionalities.

Upon starting *wija*, all classes of found plug-ins



- It is mandatory for any *wija* plug-ins to implement *XMLPlugin*, which can take XML messages not understood by the core components.
- It is highly recommended that an external plug-in implements *Updatable*, which is called when secure P2P update of the software (section 3.4.2.3) is performed.
- Depending on the functionalities of a plug-in, it may implement some of extended plug-in interfaces, such as *WijaSubResourceCapable* for hypertext sharing (section 3.4.2.1).

Fig. 3.4. Plug-in APIs of *wija* in UML class diagram



- *func* is either an internal or external plug-in that implements *WijaSubResourceCapable*.

Fig. 3.5. Hypertext transfer by *wija*

are loaded first, and then the initialization code of each plug-in is called. This allows a plug-in to access public methods and fields of other plug-ins, enabling developers to utilize existing extensions for their own extensions.

3.4.2 Design for M-2: End-to-Endness

3.4.2.1 Hypertext Sharing

Hypertext sharing provides an illusion of direct retrieval of data from a client at the other end, using a web browser via HTTP[17, 45] (Hypertext Transfer Protocol). This feature works even when both computers are inside their own private networks or within firewalls, using SOCKS5 proxy as described in the section to follow.

This is a basis for many plug-ins to conduct end-to-end communication with peers. We have been experimenting with the following form of URL:

xmpp://*JabberID*/*function*/*function-dependent*

to be fed to *wija* to initiate hypertext transfer.

Hypertext sharing is realized by running a pseudo-HTTP server locally inside *wija*, and having the web browsers of the user's choice access the server on the localhost, as illustrated in

Fig. 3.5. The server initiates streams for file transfers, via a proxy service if necessary, and all data transfers are performed at the back as in-band or SOCKS5 byte streams.

3.4.2.2 Proxy Discovery

XEP-0065: *SOCKS5 Bytestreams* defines the roles of SOCKS5 proxy to intermediate two Jabber/XMPP entities which cannot directly communicate with each other because they are inside private networks or firewalls. A SOCKS5 proxy is usually implemented as a service of an XMPP domain which users must predetermine to use, but the end-to-end principle (M-2) has urged us to design otherwise for *wija*.

All *wija* clients are designed to be capable of providing a proxy service. This feature can be turned on and off by the users.

When a direct SOCKS5 connection is found impossible, *wija* searches for an available proxy service in its buddy list. This allows users to set up a proxy service on some machine using an only regular distribution of *wija*[2], and make the service available to them by adding the entity onto their buddy lists.

There is a proxy service provided by us, whose Jabber ID is "proxy@media-art-online.org", but every *wija* user has liberty to set up their own proxies, or to become one themselves.

3.4.2.3 Secure P2P Update

Since version 0.12, *wija* allow users to update it or its plug-ins by directly downloading the new software from the computers of their buddies if they use newer versions. The validity of the new software is automatically verified since they are digitally signed by the system's public key, which is imported the first time *wija* is started. The system's public key is stored in the internal file directory within the executable file of *wija*.

As Fig. 3.6 shows, the buddy from whom the software is downloaded can be selected by the user if there are multiple candidates.

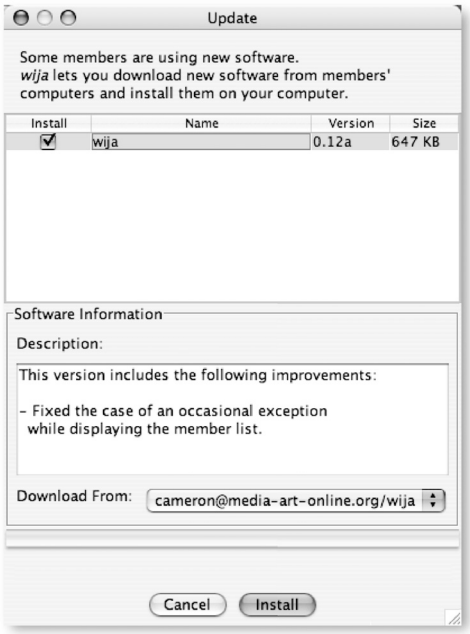
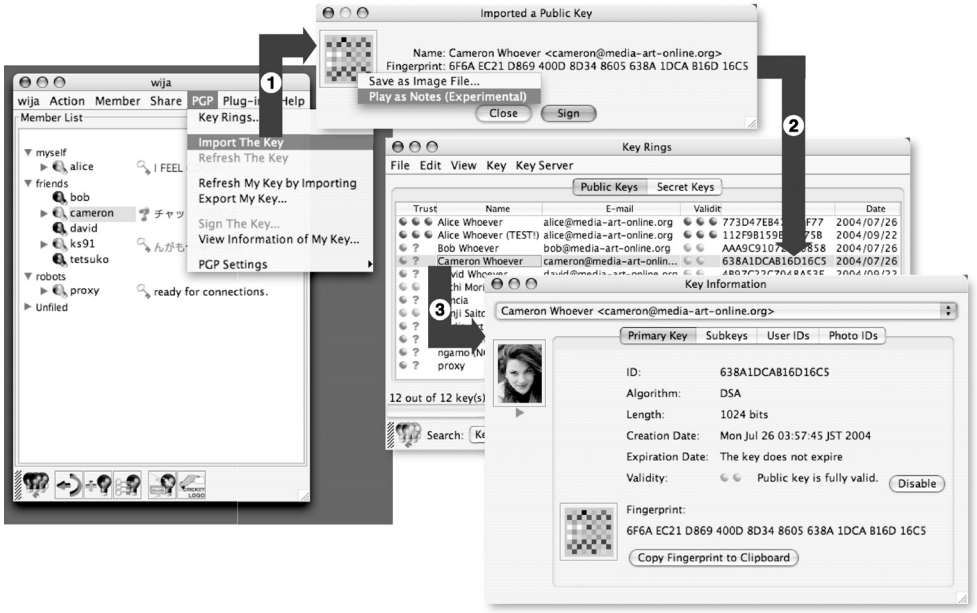


Fig. 3.6. Update window

Secure P2P update is performed using hypertext sharing. The update feature is implemented as an internal plug-in of *wija* that implements *WijaSubResourceCapable*. *wija* and external plug-ins register with the update plug-in so that when the update module receive a request containing the identifier of those modules it passes the message for requesting files to them. The actual transfer is performed by the hypertext transfer engine of *wija*.

It is important to log occurrences of this update for measurement purposes to find out the actual user distribution of *wija*. When the *wija*'s executable JAR file has been received from a peer, *wija* records the Jabber ID of the peer. When someone else receives the executable from the client, it sends "increment" message to the Jabber ID. Upon receiving the message, *wija* records the incremented number to its local storage. The message is forwarded to their parents if they have ones. Finally, those who have downloaded the new version directly from the web site (or the developer himself for that matter) has the approximate number of peers who are using the new version.

2 Although usually *wijabot* (section 3.3.2) is used for the purpose of setting up a SOCKS5 proxy.



1. A user can import public keys of others transferred via XMPP.
2. The imported keys are signed after checking their fingerprints. The keys appear in the visual representation of the key ring.
3. By double-clicking on the key, the detail information of the key is displayed.

Fig. 3.7. Integration with GnuPG

It is possible to record the identities of those who have downloaded the new versions, but *wija* does not do so for privacy protection reasons.

3.4.3 Design for M-3: Security

3.4.3.1 Integration with GnuPG

For providing cryptographic foundation, *wija* is designed to cover most functions of *gpg* command (version 1.4.x) in graphical ways, as well as the features defined in XEP-0027: *Current Jabber OpenPGP Usage*.

Supported GnuPG features include signatures to files and verification of them, encrypting files and decrypting them, importing and exporting public keys, searching in local key rings and in key servers, generation of key pairs, signature to public keys, generation of revocation certificates, changing passphrases, deletion of public or secret keys, browsing information on keys including photo IDs, adding and deleting user IDs and photo IDs, and sending public keys to and receiving them from key servers.

3.4.3.2 Public Key Exchange

By the end-to-end principle (M-2), *wija* is designed to have a replacement feature for the function of key servers; public key exchange can be made through Jabber/XMPP in-band XML streams.

Fig. 3.7 shows how this feature is integrated. *wija* tracks bindings between public keys and Jabber IDs, which are initiated when users generate a new key pair. Or if *wija* finds just one key in user's secret key ring, a binding is created automatically. Users can import public keys of others by selecting peers on their buddy lists.

Managing a web of trust is hard, especially for novices of using PGP. *wija* provides some experimental ways to represent key fingerprints as illustrated in Fig. 3.7, as color and sound patterns, to ease the burden of fingerprint checking.

3.4.3.3 Encapsulation of GnuPG Features

The functionalities of GnuPG are accessed from *wija* by calling *gpg* command through *exec()* method of the Java runtime module.

The functionalities are encapsulated as a Java class named *org.media_art_online.gnupg.GnuPG*, so that plug-in writers need not to be bothered by the detail of using GnuPG. The package *org.media_art_online.gnupg* provides Java classes to abstract GnuPG data structures such as keys, user IDs and photo IDs.

3.4.4 Design for M-4: Freedom

This mission is about providing freedom of innovation to developers, which we have been challenging through practices described in section 3.5.

The pluggable architecture has made *wija* extensible by allowing anyone to add new functionality, but there is another reason for the design: retractability. Because *wija* is experimental in its nature, some features may not be found socially fit. If a feature is seen that way, in light of copyright laws, for example, the plug-in is safely discarded from the system without affecting the entire platform of *wija*, so that other experiments can go on.

3.5 Practice

3.5.1 Development Environment

3.5.1.1 Build Toolkit

wija's developer team has been using Perforce[128] for concurrent version management, and Apache Ant[8] for building software.

We have configured a machine to perform nightly builds so that any files added locally and missing from the Perforce depot or any discrepancies among modules, especially ones between *wija* and its plug-ins, are detected the next morning.

3.5.1.2 Documentation Toolkit

Documentation on the web are expressed in what we call *OmniDocument* format which we have developed as an extension of RD for Ruby language. By using the format, not only it is easier to write, but also the resulted web pages are ensured to be displayable by the built-in browsing functionality of Java runtime.

The help pages of *wija* have been generated using the toolkit.

3.5.2 Public and Developer Relations

3.5.2.1 Publicities

We have been advertising *wija* via a number of media.

We have set up an official web site of *wija* for general users, and a Wiki site[182] for developers has also been made open to public.

wija is among the lists of Jabber/XMPP clients at the JSF web site and an entry in Wikipedia[185]. It has its own entry in Wikipedia at the following URL:

- `http://en.wikipedia.org/wiki/wija`

We have introduced *wija* in special articles in the January 2006 issue of the monthly JavaWorld[140] and in the October 2006 issue of the quarterly UNIX magazine[142], both in Japan.

3.5.2.2 Communities

As for more community oriented activities, we have set up community pages for *wija* at SNS (Social Networking Service) sites *mixi*[39] and *GREE*[51]. There is a *wija* developers mailing list for discussions.

Since February 2007, we have been using SourceForge.net for development management and software distribution at the following project page:

- `http://sourceforge.net/projects/wija/`

3.5.3 Public Releases of wija

3.5.3.1 Releases

wija has been available at the following URL:

- `http://www.media-art-online.org/wija/`

Table 3.2 is the chronological list of releases of *wija*. Statistical analysis on the publicity of the recent versions is found in section 3.6.2.

We have also been distributing *wija* from SourceForge.net since version 0.12.

3.5.3.2 Lessons Learned

These public releases have been made as we

Table 3.2. Chronological list of *wija* releases

Date	*Version*	*Description*
Jun 14, 2004	version 0.03	Pre-public release version.
Jun 14, 2004	version 0.04	Initial public release.
Jun 21, 2004	version 0.05	Improved data exchange.
Jul 15, 2004	version 0.06	Many improvements for usability.
Sep 29, 2004	version 0.07	Key rings, photo IDs and Spot-lite.
Jan 3, 2005	version 0.08	Hypertext sharing and Tunes.
Feb 2, 2005	version 0.09	Many improvements for usability.
Apr 5, 2005	version 0.10	Variance over time (*i*-WAT) and PaNIC storyboard.
Dec 6, 2005	version 0.11	GnuPG 1.4.x support and new barter currencies.
Jan 3, 2007	version 0.12	MSN/Google account support, automatic update, etc.
Sep 8, 2007	version 0.13	Simultaneous multiple sessions, Overlay GHC (initial release), etc.

learn by the feedbacks from users.

Supports for key rings management and photo IDs (version 0.07), for example, resulted from our observation that users had difficulty in handling public keys.

Variance over time feature of *i*-WAT and PaNIC storyboard (version 0.10), new barter currencies (version 0.11), MSN/Google account support (version 0.12), and simultaneous multiple sessions (version 0.13) resulted from explicit requests from users that are now managed on our SourceForge.net project pages.

3.6 Results and Evaluation

3.6.1 Effects on People

3.6.1.1 Effects to Research Activities

Most existing plug-ins of *wija* have been required from researchers' needs, and were quickly implemented on *wija* to help their research activities.

So far, *wija* and plug-ins have helped one Ph.D. degree (*i*-WAT), 3 masters degrees (Tunes, POCOMZ and related proto-typing) and one bachelor's degree (PaNIC). Spot-*lite* and Cutie X were also proposed at first as undergraduate projects.

We believe that for the researchers, *wija* has been expanding their universe of what they can do.

3.6.1.2 Relations with Users

Because of *i*-WAT, *wija* has already been in use by many of the WAT System community members as the reference platform of the currency exchange.

Many of such users are not specialized in computing, which makes them perfect for returning feedbacks on the usability of *wija*. Many improvements as described in section 3.5.3.2 owe feedbacks from them.

3.6.2 Statistics

3.6.2.1 Web Server Access Logs

Fig. 3.8 shows the frequencies of successful downloads of *wija* during the period between December 2005 and January 20, 2007[3].

The number of downloads in January 2007 shows that transition to the succeeding version has been successful. However, there remain mysterious downloads of prior versions of *wija*, especially in June and August 2006. Further investigating the server record has shown that many downloads were initiated from the same IP addresses in a very short periods, suggesting that those cases are accesses from some kind of software robots.

Fig. 3.9 shows the number of successful downloads of *wija* version 0.11 for different platforms.

3 The data for December 2006 has been mostly lost because of an accident resulted from replacing the web server.

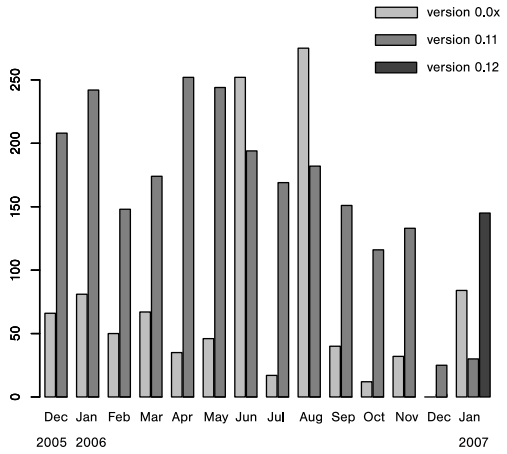
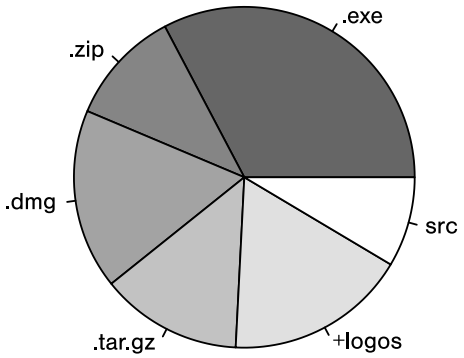


Fig. 3.8. Successful downloads of *wija* Dec/2005 ∼ Jan/2007



wija011.exe	781
wija011.zip	261
wija011.dmg	408
wija011.tar.gz	321
wija011x-logos0yy.{exe|zip}	412
wija011src.tar.gz	204
Total	2,387

Fig. 3.9. Successful downloads of *wija* version 0.11 (∼ Jan/20/2007)

After a little over one year from its release, *wija* version 0.11 had 2,387 successful downloads. 44% (.exe + .zip) of them were for Windows platform, followed by 17% each for distribution of LOGOS and for Mac OS X. About 9% were downloads of the source code.

Choice of Java as development language did not seem to affect the user distribution as much as we feared.

3.6.2.2 XMPP Server Records

Some information can be collected from the XMPP server we have been operating.

The number of XMPP users at media-art-online.org, which we believe can approximate the number of *wija* users, is 335 as of January 22, 2007.

3.6.2.3 P2P Update Log

The number from the XMPP server can be misleading because users can easily start and stop using *wija*. Substantial number of users recorded on the server may not be using *wija* currently.

To obtain more accurate measurement, we have been investigating a way to use the logging of secure P2P update. Soon after the release of *wija* version 0.12 on January 3, 2007, we have built a minor update called version 0.12a, and have been distributing the version only through the secure P2P update.

We discovered that 47 users have successfully downloaded and used version 0.12a before another minor update on April 26, 2007. We believe that it approximates the size of the user cluster of *wija* around us.

3.6.3 Evaluation

3.6.3.1 Evaluation of Challenges

Our view is that although challenges remain in many ways, our efforts so far have been successful as follows:

C-1: Pathway to P2P

In many development items, we have successfully eliminated necessity of XMPP servers supporting certain features.

Elimination of necessity of web servers is beginning. Distribution of the software is partially done in a P2P way.

C-2: User Friendliness

In large part, this remains to be a challenge. However, GnuPG integration seems to be appreciated by users; to our best knowledge, *wija* is the only software written in Java (thus naturally cross-platform) that provides GUI frontend for GnuPG. Some novel approaches to enhance usability have been experimented

such as direct end-to-end exchange of public keys and audio/visual representation of fingerprints. As a casual observation, many non-technical users are signing their presences and using encrypted messages, which seems to be indicating that the design has been successful.

3.6.3.2 Evaluation of Missions

Our view is that except the above remaining challenges, our missions have been complete as described below.

M-1: Messaging Platform

wija is a platform that runs on multiple OS platforms. It is internationalized in such a way that every message to user is provided in both English and Japanese, and it provides means for incorporating additional languages. It has been helping innovations as a tool for researchers and students pursuing their new applications of the Internet.

M-2: End-to-Endness

Most features, except vCard sharing (including avatars) that is too popular among Jabber/XMPP clients to design otherwise, do not require specific functionality of XMPP servers.

We have also eliminated the need for public key servers.

M-3: Security

wija is integrated with GnuPG, and provides means for signing and encrypting messages, as well as safely updating the software by downloading them from peers.

M-4: Freedom

wija has been distributed from our site under GNU GPL version 3, and third-party sites are also redistributing *wija*.

Secure P2P update uses the system's public key to verify the new software, which might sound as if we are the only ones who can develop *wija* and external plug-ins. But the key is stored within the original executable file a user trusted to download by other means. Therefore, this does not restrict freedom of developers distributing modified versions of *wija* with their own system's public keys, and allowing users to maintain the line of software by the secure P2P update.

3.7 Related Work

There are a lot of messaging clients which support Jabber/XMPP, but we only refer to Psi[81] and Adium X[1], for their cross-platform support and extensibility, respectively.

There are surprisingly small number, if not none at all, of projects that make use of extensibility of Jabber/XMPP protocols other than *wija*.

3.7.1 Psi

Psi is a popular Jabber/XMPP messaging client which supports multiple platforms. Although Psi is based on the same set of protocols, the policy is quite different from that of *wija*, as quoted below.

"The goal of the Psi project is to create a powerful, yet easy-to-use Jabber/XMPP client that tries to strictly adhere to the XMPP drafts and Jabber JEPs[4]. This means that in most cases, Psi will not implement a feature unless there is an accepted standard for it in the Jabber community."

3.7.2 Adium X

Adium X is a popular messaging client for Mac OS X that supports multiple protocols through use of libgaim library.

Like *wija*, Adium X has its plug-in architecture, and many of the essential features of the software are provided by plug-ins.

To our best knowledge, Adium X is not known as a platform for extension of what we can do over the Internet, except Gizmo, an IP telephony plug-in.

4 XEPs were formerly known as JEPs, standing for Jabber Extension Proposals.

3.8 Future Work

We intend to further brush up the design of *wija* so that it will be more usable and graphically pleasing. We also intend to support more protocols (via gateways). We believe that we need those improvements to attract more attentions from researchers and general public, so that *wija* will become a more valuable tool for research and for living.

Meanwhile, we intend to develop further on the P2P aspects of *wija*. We intend to put more efforts on Overlay GHC as a tool for research on declarative programming of overlay networking. The insights we will acquire from the experience will be used upon the design of *wija* to replace XMPP with P2P protocols (we will probably make those protocols switchable).

3.9 Conclusions

We have developed *wija*, an open and extensible messaging platform on the Internet to assure freedom of experimentation for researchers, software developers and anyone in pursuit of problems, in the form of a Jabber/XMPP client.

Although this work is still in progress, we believe that we have achieved many aspects of the original mission statement. In particular, we have avoided dependencies on servers in many scenes of our development; public key exchange, proxy discovery, and so on, not to mention the level of ease we have achieved in handling public key cryptography.

In the hope that this software project will be an asset for advancement of software science and networking, we will continue our development.

Copyright Notice

Copyright © WIDE Project (2007). All Rights Reserved.

第 4 章　ID Resolution Mechanism by DHT-DNS Mounter

4.1 背景

「全世界の個品 ID」のようなフラットで莫大な規模の ID 空間を管理するために、Chord[157] などの分散ハッシュテーブル（DHT）を利用するのは自然な考え方である。実際、特定の業者の個品 ID を管理するためだけであれば、大きな問題はないように考えられる。しかし、システムが普及するにつれ、単一の DHT で多くの業者の多様な商品カテゴリを取り扱うことを想定すると、DHT の一様性が問題となる可能性がある。

商品カテゴリには、商品のライフサイクルが短く商品 ID の数が膨大になるカテゴリ（食品など）と、逆にライフサイクルが長く商品 ID の数が限定的なカテゴリ（車などの耐久消費財）、あるいは要求の季節変動の激しい商品（お正月用品や学習用具）などを含む。ここで、商品カテゴリの性質が ID 空間に求める性質、とくにコストへのの許容条件が異なる場合、単一の DHT システムではそれらを吸収できない。

その上、単独の DHT で全個品の情報を取り扱う場合、その DHT が動作しなかった場合の責任を誰が取るのか、コスト負担はどのようにするのか。一様な DHT に全ての ID 解決を任せると、責任分界点を明確化できない。

そこで、物品の ID と、それぞれに対応する全世界に分散した情報との柔軟な紐付けを目指して、小規模から円滑に拡張でき、個品に対する現場レベルの情報を管理可能なトレーサビリティシステム [190] を提案している。とくに、その中で用いる DHT と DNS を併用するハイブリッド ID 解決機構 [37] は、タグ ID から関連情報を持つ DB の URL を解決するための ID 解決機構である。これは、DHT を用いた商品カテゴリ毎の個品 ID 解決サービスと、異なる個品 ID 空間を DNS の名前空間に結合するための DHT-DNS マウンタと呼ぶ機構から成る。

本研究では、トレーサビリティシステムに対する応用を主眼とし、複数の商品カテゴリに由来する異なる性質の個品 ID 空間を統合でき、かつ、個品 ID

を取り扱うのに十分な規模拡張性を持つ ID 解決機構を構築し、評価する。さらに、DHT-DNS マウンタにより ID 解決機構を結合することで、多様な商品カテゴリそれぞれにあわせて適切な性質の個品 ID 空間を提供できることを示す。

4.2 多様性および規模拡張性を実現する ID 解決機構の設計

4.2.1 要件と環境の定義

個品追跡トレーサビリティに用いる ID 解決機構の要件としては、以下の 3 つが存在する。

DNS を基本プロトコルとする：主として ONS[40] が DNS ベースで構築されている、という理由から、クライアントからアクセスするためのプロトコルは DNS であることが望ましい。DNS クライアント（リゾルバ）はほぼ全てのインターネットノードで利用できるため、クライアント側は最小限の改変のみで個品 ID 解決機構が利用できる。

複数の異なる特性の ID 空間を統合する：個品を追跡するための ID 空間は、複数の商品カテゴリで必要とされている。ここで、個々の商品カテゴリそれぞれに独立した個品 ID 空間を構築すると、利用者は商品カテゴリ毎に利用する個品 ID 解決手段を選択しなければならない。一方、個々の商品カテゴリはそれぞれ商品のライフサイクルや流通量、個品 ID 解決がもたらす付加価値とそれに伴い負担できるコストなどが異なり、単独の一様な ID 空間によって ID 解決手段を提供できるものではない。そこで、商品カテゴリ毎に異なる特性の個品 ID 空間を用意し、これを一つのグローバルな商品 ID 空間に統合できる構成が求められる。

規模拡張性を持つ：個品追跡を行うためには、ある商品カテゴリに属する個品 ID 全てを含む ID を格納した個品 ID 空間を作り、また維持できなければならない。一方で、トレーサビリティシステムに当初から多大な投資は期待できないことから、小規模から円滑に拡張できる必要がある。また、季節性商品は 1 年の中で特定の時期に売買が集中するため、要求に応じた規模のサービスを動的に実現できる必要がある。

なお、本システムが想定する動作環境は次の通り。本研究では、個品 ID 解決サービスを構成する各 DHT ノードは名前サービスのための専用ノードであり、一般に P2P 的な技術で想定されるような、クライアント PC によるものは想定しない。従って、各 DHT ノードは、少数のデータセンタに集約された、十分に管理されたノードであるとする。その結果、ノードの故障発生頻度は通常のサーバと同程度（MTBF は 1 年以上）であり、またノード間のネットワーク遅延は平均で 3 ミリ秒以下に抑えられるとする。なお、ネットワーク遅延の仮定値は報告者の環境から、東京都内に本拠があると思われる主要 ISP や企業などの WWW サーバまでの遅延を測定した値（0.2〜2.5 ミリ秒）に基づく。

4.2.2 システム構成

報告者らが提案するトレーサビリティシステム（以後 **LOTR**：**Linked Object TRaceability** と呼ぶ）は、以下の要素から成る。図 4.1 にその概要を示す。

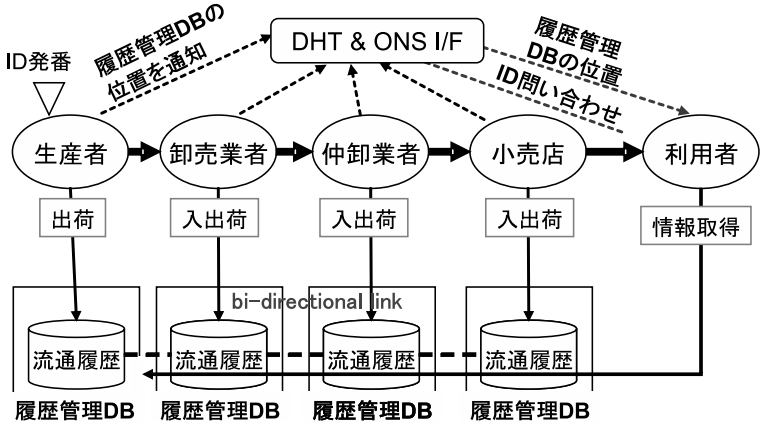


図 4.1.　LOTR の全体像

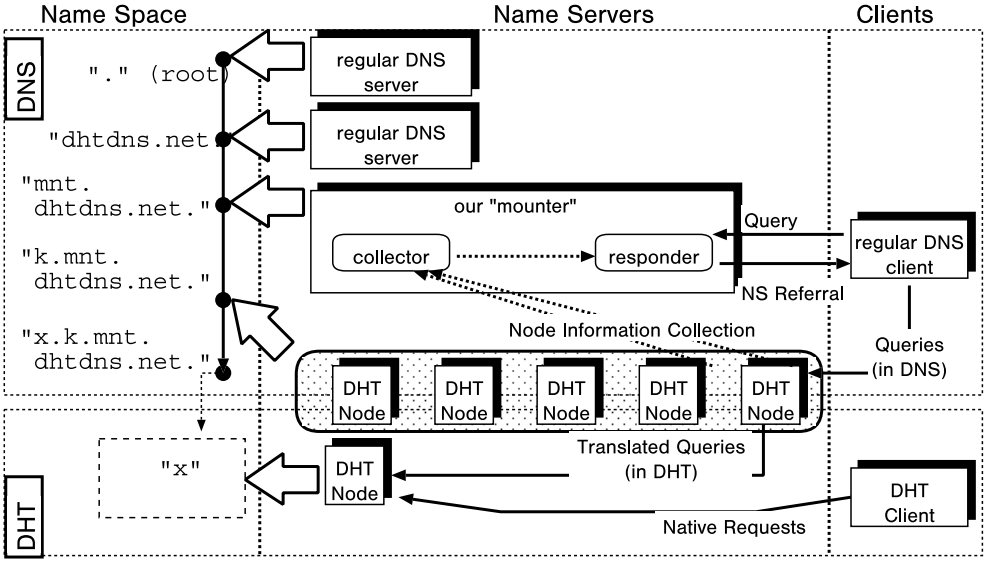


図 4.2.　ID 解決に関連する構成

履歴管理 DB：実体となる情報を管理するデータベース。**LOTR** における履歴管理 DB は流通経路の前後の「管理ドメイン」（生産・流通・卸・小売など、ある業者により物品の出入りが管理される領域）が持つ履歴管理 DB へのリンクを保持する機能が存在する。詳細は別文献 [202] 参照。原則として一つの管理ドメインに対して代表する履歴管理 DB が一つ存在する。

ID-DB 解決：以下の 3 つから成る、個品 ID からその ID に関連するデータベースの位置を発見するためのサービス。

ONS 互換 DNS：システムの入口として、ONS と類似した DNS サービスを持つ。これは、タグ ID から会社名レベル、あるいは商品カテゴリレベルの ID 解決の権限を、特定の DHT-DNS マウンタに移譲する。システム全体を通して一つ存在する。

DHT-DNS マウンタ：DNS の名前空間木に射影された商品 ID のうち、「個品 ID」あるいは「商品カテゴリ ID + 個品 ID」の部位全てを、1 つまたは複数の DHT ノードに、DNS により権限を委譲（NS RR[116]）する。なお、どちらの構成になるかは、DNS の名前空間設計によって決まる（詳細は本稿の範囲外になるため省略）。商品 ID の生産者あるいは商品カテゴリに対して一つ存在する。

個品 ID 解決サービス（DHT）：個品 ID あるいは商品カテゴリ ID + 個品 ID と、その ID に対応するトレース DB の URL の対応付けを DHT の原理によって行う。各々の DHT ノードは DNS の問い合わせを受け付け、これを DHT の問い合わせに翻訳して解決し、DNS の応答（NAPTR RR[105]）を返す。

ONS 互換 DNS、DHT-DNS マウンタ、個品 ID 解決サービスの 3 つの構成により、規模拡張可能かつ商品カテゴリ毎に独立した性質を持たせられる個品レベル ID 解決が可能となる。

ID 解決に関連する構成例を図 4.2 に示す。図中左側が論理的な名前空間、中央がそれを構成するサーバ、右側が利用するクライアントである。また、図中点線で囲まれた上側が DNS、下側が DHT により構成される。図中の DHT ノードは全て、DNS における特定の名前空間（この例の場合は `k.mnt.dhtdns.net`）に対する名前サーバとして振る舞う。多数の DHT ノードはそれぞれに DNS client からの負荷を分散するために、DHT-DNS マウンタ（図中 **mounter**）は名前権限委譲のプロセスで、これら DHT ノードをラウンドロビンで利用する。

DHT-DNS マウンタ内には **collector** と **responder** の二つの機能が存在する。**collector** は DHT ノードの情報を動的に収集し、**responder** は DNS クライアントからの問い合わせに対して収集した DHT ノード情報を応答する。例えば、`x` を個品 ID としたとき、`x.k.mnt.dhtdns.net.` の問い合わせがあった場合は、`k.mnt.dhtdns.net.` に対して権限を持つサーバとして収集した DHT ノード情報をクライアン

トに返す。このとき、DNS 名前空間におけるサーバの権限を示す NS レコードにはドメイン名しか記入できず、**collector** は IP アドレスしか収集していないため、それら IP アドレスから A/AAAA レコードを合成し、そのレコードの名前を NS レコードに格納する。

あるタグ ID に関連する情報を持つ管理ドメイン上の履歴管理 DB は、そのタグ ID を DNS 名前空間に射影した際に権限を持つ DNS 名前サーバを検索し、これが DHT ノードだった場合、自身の URL を登録する。各 DHT ノードには DHT に格納されたエントリを DNS レコードに変換するルールが存在する。そのルールに沿った形式で URL を登録することで、DHT により登録したデータを DNS から閲覧可能になる。なお、本 DHT の実装では、DHT との通信プロトコルに OpenDHT[135] などと同等の XMLRPC と、独自プロトコルの二種が利用できる。

4.3 評価

以下、(1) 問い合わせ量の総量を考えた際のシステム全体を通じてのボトルネックとなる **responder** のサービス構成の推定、(2) 保持すべきデータの総量に対する DHT ノード数およびその規模における問い合わせにかかる時間の推定、の 2 つにより **LOTR** を評価する。

4.3.1 ボトルネック

LOTR においては、処理可能な単位時間あたりの要求量を決定づける構成要素は、DHT-DNS マウンタである。個品 ID 解決サービス側の DHT ノードの数は要求量などに応じて随時増設可能だが、DHT-DNS マウンタの数には実質的な上限が存在する。これは、DNS では実用上メッセージを 1 つの分割されない UDP データグラムに収めることが求められていることによる。

DHT-DNS マウンタは、DHT ノードの情報を取得する機能（**collector**）と、DNS 問い合わせに対してノード情報を NS RR の形で返答する機能（**responder**）から成る。DHT-DNS マウンタ自体の性能は、**responder** 側の性能で一意に定まる。**collector** が十分な速度で動作しない場合は、同じ DHT ノードの情報を多数のクライアントに応答することで回避可能だからである。当然、その場合は一部の DHT ノードに問い合わせ負荷が集中する可能性がある。この問題は、十分な数の候補を並列に集めることで回避可能であると考え、本報告書では考慮しない。

responder に対する負荷は、DNS の再帰リゾルバサーバによるものである。再帰リゾルバサーバは各個にキャッシュを持つため、**responder** で指定された TTL（キャッシュ生存時間）の間はキャッシュレコードにより応答が行われる。従って、TTL の調整により **responder** の実際の負荷は調整可能である。

4.3.1.1 responder の性能

表 4.1 に、試作した DHT-DNS マウンタの **responder** の性能（1 秒間あたりの問い合わせ応答数）を示す。試作した DHT-DNS マウンタは 2 つあり、一つは Python で書かれている多機能版、もう一つは C 言語で書かれている高速版である。表 4.1 は、**responder** の性能限界を見積るため、現状の実装における DNS 問い合わせ応答の速度（qps: query per second）を ISC BIND に付属の queryperf により 10 秒間計測したものの 5 回の平均である。計測は、Dell PowerEdge 1750（CPU: Intel Xeon 3.20 GHz × 2、Memory: 512 MB）上で行った。なお、queryperf における並列化（オプション -q）の影響は **responder** がシングルスレッドのため無視できる程度であった。

表 4.1. DHT-DNS マウンタ **responder** の性能

実装	qps 平均
多機能版	1980
高性能版	7970

4.3.1.2 必要なサービス構成の推定

仮に、1 つのコンビニエンスストア（CVS）の POS にて読み込まれる商品の種類の数は約 4000 種[5]、日本国内の CVS 店舗数を 4 万店[6] とする。それぞれの店舗が独立した DNS キャッシュサーバを用いて個品 ID 解決をすると、最大で 1 億 6000 万の (キャッシュサーバ, 問い合わせ内容) の組み合わせが考えられる。

ここで、**responder** によるサービスの平均待ち時間を $h = \frac{1}{qps}$ と置く。また、クライアントは **responder** が返す TTL 毎に 1 回問い合わせを発行

5 http://www.posbank.jp/によると、商品点数 3800（2007 年 5 月時点）

6 http://www.d2.dion.ne.jp/%7Ehmurata/conveni/tenpotable.html インターネットタウンページに基づく調査結果（2000 年）

表 4.2. 目標とする呼損率 $B < 0.01$ を維持するための要求組み合わせ数 c の上限と DHT-DNS マウンタ数 S の下限

	多機能版	高性能版
$c <$	1,728,001	6,955,682
$S \geq$	93	23

する、つまり、CVS 側では連続的に問い合わせが発生しており、キャッシュ生存時間が切れると必ず問い合わせが発生すると仮定する。**responer** がシングルスレッドであることから、これを $M/M/1$ の即時式待ち行列とみなすと、Erlang の損失式

$$B = \frac{\lambda h}{\lambda h + 1}$$

により呼損率 B が求められる。

目標とする呼損率 B を 0.01 以下とする。これは、Jung らの測定 [79] における DNS の再送パケットの比率である 20%〜24%に比べて十分低い値として選択した。また、RFC1537[16] で推奨されている一般的な TTL 値が 86400（1 日）であることと、DHT ノードが DNS と同等な水準で管理がされているという仮定から、TTL を 86400 とおく。

この条件下で、目標とする呼損率 B を上まわらない範囲で 1 つの DHT-DNS マウンタが対応できる要求の組み合わせ数 c と、日本の CVS 全体（1 億 6000 万組み合わせ）に対応するために必要な DHT-DNS マウンタの数 S を、表 4.2 に示す。

DNS のルートネームサーバ同等の並列度 13 を DHT-DNS マウンタがサービスする各ドメインに適用することを考えても、全ての製品を 1 つのドメインで支えることは困難であることが予想できる。一方で、商品 ID には構造が存在し、DHT-DNS マウンタあるいは DNS はこの構造に沿ってそれぞれの商品の名前空間を分離できる。従って、これを利用することで 2（高性能版：$2 > 23/13$）ないし 8（多機能版：$8 > 93/13$）以上に名前空間を分割することにより、現状の DHT-DNS マウンタにより構成する名前サービスで、仮定した要求量に応えられるといえる。

4.3.2 DHT 規模と問い合わせにかかる時間の推定

本研究では、節 4.2.1 に示したように、100 億個の電子タグ ID を検索するとき検索に利用できる時間を平均 0.3 秒とし、この条件を満たす構成の実現可能性を検討する。

検討は、必要なノード数の推定、そのノード数における DHT 中の再帰問い合わせ段数の推定を行い、最終的に 100 億個の ID を検索するための必要時間の推定値を算出する。

4.3.2.1 必要ノード数の推定

必要ノード数はノードローカルに存在するデータテーブルの適切なサイズを求め、テーブルの総量をこれで割ることによって行う。

ノードローカルなテーブルの管理には、Berkeley DB の実装を用いた。節 4.3.1.1 と同等の Linux OS 環境において大量のデータを格納したテーブルを生成したところ、10^6 規模までのデータは問題なく取り扱えるが、10^7 規模のデータのテーブルにおいては、データの書きこみ性能が劣化した。表 4.3 に、本研究で用いているテーブル管理ルーチンを経由してデータの書きこみを行うテストプログラムの実行結果を示す。

ここで、ノード 1 台あたり平均 10^6 個の ID を管理するとして、10^{10} 個の ID を管理するには 10^4 台のノードが必要であるといえる。

表 4.3. テーブル読み書き時間の平均（ミリ秒）

テーブル規模	書き込み	読み出し
10^5	0.15	0.13
10^6	0.55	0.12
10^7	3.22	0.44

4.3.2.2 再帰段数の推定

文献 [157] では、試作した DHT で採用している基本アルゴリズム（Chord）の再帰問い合わせの深さは、$(0.5 \log_2 N)$ であるとされている。ここで単純かつ理想的な二分木探索が行われているとすると、10^4 台のノードにおいては 6〜7 段程度（$0.5 \log_2 10^4 \approx 6.64$）の再帰が行われると予測できる。しかし、実装したアルゴリズムはテーブル空間上の近傍情報の処理に最適化が施されている。また、確実に対象ノードに到達するために安全な経路を取り余分なホップ数をかけている部分もある。これらの理由により、理想的な二分木探索とは異なる可能性がある。そこで、計測が容易で計算機の性能などの条件に左右されない「再帰問い合わせ段数」の平均値のモデルを

$$r = a \log_2 N + b \qquad (1)$$

表 4.4. パラメータ a、b の具体値

a	b
0.500	2.22

とし、パラメータ a と b を実測により求めることとした。

DHT 単独の規模拡張性計測において、DHT 自身が式 (1) をモデルとし規模拡張可能なことを確認する。

計測には、Pentium III 1 GHz 主記憶 512 MB の計算機が 208 台存在する StarBED クラスタ A を利用した。StarBED は、NICT 北陸リサーチセンターに存在する、PC の実機を用いた分散エミュレーション環境であり、大規模実験制御機能 [115] が提供されている。

これらの物理ノードに、OS として Linux 2.4.27 を PXE Boot によりロードし、DHT 環境を実現した。また、実際にはより多数の DHT ノード環境を模擬するため、1 物理ノードあたり 20 DHT ノードを動作させた。この物理ノード数を n とおく。

この n 台の物理ノードとは別に、もう 1 つの物理ノードを専用に確保して、DHT の起動時に他のノードへの連絡を行うプロセス（seeding node）を 1 つ起動している。これは起動後は通常の DHT ノードとして振舞うため、合計 DHT ノード数 N は $N = 20n + 1$ となる。

実験から得られた式 (1) に対応する回帰分析結果のパラメータを表 4.4 に示す。なお、Finger の構成方法は Chord と同等であり、$a = 0.5$ であるが、実装上経路を最適化しない部分から定数 b が発生し、約 9 段の再帰段数となっている。

4.3.2.3 100 億個 ID の問い合わせ時間の推定

前出の規模拡張性の検討結果と、節 4.3.2.1 の検討から実際の再帰段数 r および問い合わせ時間 Q を推測し、その結果を表 4.5 に示す。なお、ここでの誤差は最大で約 4.7%であり、2 つのケースを除いた全てのパラメータが誤差 2.5%未満で実測値に一致した。

これらから、本研究で開発した DHT の実装を用いた、10000 ノード DHT の問い合わせ再帰問い合わせ段数の平均値は、**8.9 程度**であると推測できる。

1 つの問い合わせが応答されるまでの DHT 内の処理にかかる時間 Q は次式で表わせる。

$$Q = (t_n + t_r) \times h + t_l$$

表 4.5. $N = 10000$ における予測再帰段数 r および問い合わせ応答予測時間 Q

r	Q
8.86	292

（ただし、t_n：ノード間のネットワーク遅延、t_r：ノード再帰問い合わせ応答時間、h：再帰問い合わせ段数、t_l：最終段テーブル検索時間）

1 段の再帰にかかる平均的な時間は、DHT 1 ノードの再帰問い合わせ応答（メッセージ処理および再帰経路のルックアップ）にかかる時間と同等である。実測で 30 ミリ秒程度という数値を得ている。節 4.2.1 で述べたとおり、DHT ノード間の遅延を 3 ミリ秒以下に抑えたとする。なお、t_l は表 4.3 から、10^6 のテーブル規模であれば 1 ミリ秒未満なので無視できる。

これらの実測値および仮定から、およその Q（ミリ秒）が 292 程度であることが推定できる。

4.4 結論と今後の課題

本研究では、個品追跡可能なトレーサビリティシステムを実現する基礎として、DHT と DNS を併用したハイブリッド ID 解決機構についての検証を行った。100 億（10^{10}）個の商品 ID を検索するために、10000 ノードを想定し、これらの間で対象となるデータの探索を行うことでどの程度の時間がかかるかを推定した。約 2500 ノードまでの規模の実験と、文献に基づくモデルを適用し、10000 ノードの再帰段数を推測した上、実測した問い合わせ応答の一段あたりの処理時間などから、具体的な数値を求めた。10000 ノードから構成される DHT は、平均して 6 段〜9 段の再帰問い合わせを行う。その結果平均 179〜275 ms 程度の時間を ID 解決で消費することを推定した。

第 5 章 Conclusions

In this report, the following topics we have investigated this fiscal year have been described in detail:

1. Annual IDEON retreat.
2. Software design of *wija*.

3. ID resolution mechanism by DHT-DNS mounter.

Since all our development projects are in *deploy* phase now, we will continue our efforts for refining our methodologies through actual usage of our technologies in our lives.