

Jupiter: Peer-to-Peer Networking Platform over Heterogeneous Networks

Norihiro ISHIKAWA, Takeshi KATO, Hiromitsu SUMINO
NTT DoCoMo Inc.
3-5 Hikarino-oka, Yokosuka, Kanagawa, Japan

Shingo MURAKAMI
Nippon Ericsson K.K.,
1-4-14 Koraku, Bunkyo-ku Tokyo, Japan

and

Johan HJELM
Ericsson Research,
Torshamnsgatan 23, Kista SE-16480, Stockholm, Sweden

ABSTRACT

Peer-to-peer has entered the public limelight over the last few years. Several research projects are underway on peer-to-peer technologies, but no definitive conclusion is currently available. Compared with traditional Internet technologies, peer-to-peer has the potential to realize highly scalable, extensible, and efficient distributed applications. This is because its basic functions realize resource discovery, resource sharing, and load balancing in a highly distributed manner. An easy prediction is the emergence of an environment in which many sensors, people, and many different kinds of objects exist, move, and communicate with one another. Peer-to-peer is one of the most important and suitable technologies for such networking since it supports discovery mechanisms, simple one-to-one communication between devices, free and extensible distribution of resources, and distributed search to handle the enormous number of resources. The purpose of this study is to explore a universal peer-to-peer network architecture that will allow various devices to communicate with one another across various networks. We have been designing architecture and protocols for realizing peer-to-peer networking among various devices. We are currently designing APIs that are available for various peer-to-peer applications and are implementing a prototype called "Jupiter" as a peer-to-peer networking platform over heterogeneous networks.

Keywords: Peer-to-peer, Distributed Network, Protocols.

1. INTRODUCTION

From its inception, the Internet has used conventional cooperative technologies such as the server-client approach to handle network resources and provide Internet services. This offers advantages in that Internet services can be regulated by just a few central servers, but some major concerns have been raised, such as overloaded central servers and high costs, and demand has arisen for direct communications between clients. As a result, peer-to-peer has become popular and has been extensively used in overlay networks that handle vast amounts of data daily, and balance the loads over a large number of servers. It has been used for applications such as distributed search [1], file sharing [2], distributed storage [3] and

groupware [4]. Additionally, a generalized platform for peer-to-peer applications has been proposed [5] and developed.

At the same time, various devices have recently been enhanced through the addition of communication abilities. Examples are home networks, wireless sensor networks and automotive telematics. In the near future, an environment where many sensors, persons and different kinds of objects exist, move, and communicate with one another will appear. In fact, peer-to-peer communication is one of the most important and suitable networking technologies for such communications since it effectively supports one-to-one communication between devices, the free and extensible distribution of resources, and the distributed search mechanisms needed to handle the enormous resources on the Internet. It will be used in many aspects of daily life.

As mentioned above, peer-to-peer communication is suitable for both the mature Internet and the emerging ubiquitous communication environment. Peer-to-peer communication has the ability to link heterogeneous network environments in a cross-sectional manner such as the Internet, ad-hoc networks, and home networks. It can realize seamless connection between various devices via various networks.

The principal goal of our research is to design a universal peer-to-peer architecture and a general peer-to-peer platform that enhances the communication capabilities of various devices via various networks.

The rest of this paper is organized as follows. Section 2 overviews our peer-to-peer architecture, and describes each key element of our architecture. Section 3 describes the proposed protocol. We report the current status of our prototype in Section 4. Finally, a conclusion is given in Section 5.

2. ARCHITECTURE

2.1. Overview

The proposed peer-to-peer architecture is shown in Fig. 1. Hybrid and pure peer-to-peer networks are integrated by several gateway nodes. All of the peer-to-peer nodes in a hybrid peer-to-peer network are managed by a control node. Each peer node reports its existence and its adjacent nodes to the control node. The control node thus can comprehend the topology of an entire hybrid peer-to-peer network. It accepts a request from a peer-to-peer node and provides related routing

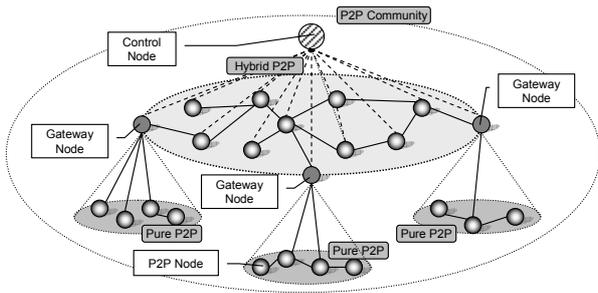


Figure 1. Peer-to-peer Architecture

information, topology optimization and security functions according to the state of the desired peer-to-peer network. The gateway node collects topology information of the pure peer-to-peer network and reports it to the control node. The control node thus can comprehend the topology of a pure peer-to-peer network as well. The gateway node also accepts a request from a peer-to-peer node in the pure peer-to-peer network and forwards it to the control node as a proxy node. This allows the gateway node to support seamless communication between a peer-to-peer node in a pure peer-to-peer network and a peer-to-peer node in a hybrid peer-to-peer network. All peer-to-peer nodes can thus efficiently communicate with each other by using the routing information provided by the control node.

While this proposed architecture has only three entities and is thus simple, it satisfies the main requirements for peer-to-peer networking. For an example, the hybrid peer-to-peer network is suitable for a large well-organized peer-to-peer network such as Grid computing over the Internet. On the other hand, the pure peer-to-peer network is well-adapted to ad-hoc type networks such as wireless LANs in hot spots and sensor networks.

In the next section, we describe the key elements of our architecture in more detail. These items in this architecture are defined as the following.

Peer-to-peer node: Peer-to-peer node is an independent, bidirectional communication entity. In our architecture, it can be a mobile device, a PDA, a personal computer, a server, a workstation, or any of a variety of devices.

Community: The term “community” means a logical collection of peer-to-peer nodes that have a common interest and obey a common set of policies. A community is identified by a community ID.

Pure peer-to-peer network: There are only peer-to-peer nodes in the pure peer-to-peer network, see Fig. 2 (a). The connection between peer-to-peer nodes is established on mutual trust. Each peer-to-peer node is an independent entity and can enter or depart the peer-to-peer network at its convenience. Messages are sent from one peer-to-peer node to another directly or via some intermediary peer-to-peer nodes. Routing information is discovered by broadcasting an inquiry message to the network.

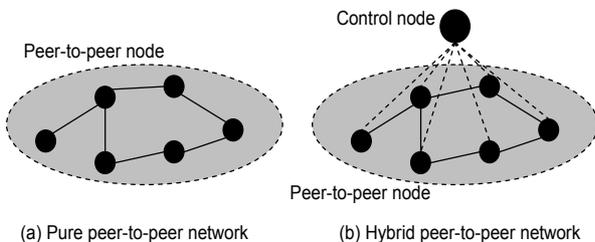


Figure 2. Pure Peer-to-peer and Hybrid Peer-to-peer

Although the pure peer-to-peer network is simple, it has some deficiencies as follows.

- Lack of efficient first peer discovery mechanism: When entering a peer-to-peer network, it's almost impossible to find the most appropriate node (e.g. the nearest node) to which it connects without having a bootstrap node that knows about other nodes in the peer-to-peer network.
- No efficient routing mechanism: Broadcast is a general method, when a node tries to find the route to a peer node to which it wants to send a message. However, broadcasting results in heavy traffic over the entire peer-to-peer network.
- Difficulty of introducing security framework: It is difficult to build an authentication infrastructure for the pure peer-to-peer network, especially certification mechanisms for authentication and access control, because it is difficult to introduce authorities for certification in the pure peer-to-peer network.

Based on the above consideration, we introduce the hybrid peer-to-peer network to offset the fundamental deficiencies of the pure peer-to-peer network. Its key point is the use of an administration entity for control purposes. Additional items include:

Control node: Control node is an administration entity that manages the peer-to-peer network. It provides several application independent functions such as name resolution, routing information provision, identifying the adjoining peer-to-peer node, network topology optimization, node authentication, and multicast group management.

Hybrid peer-to-peer network: Hybrid peer-to-peer network offers solutions to problems such as first peer discovery, inefficient routing, and insufficient security. Its architecture is shown in Fig. 2 (b). In our architecture, Control node provides the functions needed for the efficient route discovery, multicast group management, and security enhancement.

Though the hybrid architecture is easily applicable to peer-to-peer networks over the Internet, it may not be applicable to wireless sensor networks and home networks, because it is impossible to have a control node to which all other nodes can connect. Consequently, we introduce gateway node, the third entity that interconnects pure peer-to-peer and hybrid peer-to-peer networks. Gateway node definition is interpreted as the following:

Gateway node: Gateway node is a connection entity that links a pure peer-to-peer network to a hybrid peer-to-peer network. It provides several proxy functions for nodes in the pure peer-to-peer network such as routing information provisioning, node authentication, and multicast group management.

2.2. Peer-to-peer Communication Model

In the peer-to-peer communication model, the role of each communicating entity is not always clearly distinguishable. In order to design an effective peer-to-peer communication protocol, the “role” concept is introduced. We have established the peer-to-peer communication model by looking into existing peer-to-peer applications. Figures 3 and 4 show some common peer-to-peer applications. As shown in Fig. 3, we found three kinds of nodes in peer-to-peer distributed search applications [1,2,3]. The first provides content information (e.g. its location) in response to search requests, the second requests the content search and the third relays the search request and their replies. Fig. 4 shows application layer multicast over the peer-to-peer network. This application also has three similar nodes. This suggests that peer-to-peer nodes can have three roles. The roles are (See Fig. 5):

Producer role: A node acts as a Producer when it provides application data or services.

Consumer role: A node acts as a Consumer when it asks for a service or consumes application data without any request or in response to its request.

Relay role: A node acts as a Relay when its current communication task is just to forward application data, service requests and their replies.

In peer-to-peer applications, a node may play any of these three defined roles and the peer-to-peer application dynamically determines which one is to be used.

Based on the roles described above, we define two peer-to-peer communication modes. (See Fig. 5)

Proactive communication mode: This mode represents the unsolicited transmission of information that does not require any specific response. It is generally used by a node to notify the others of its own existence or resources it holds. This communication mode consists of an advertise message only.

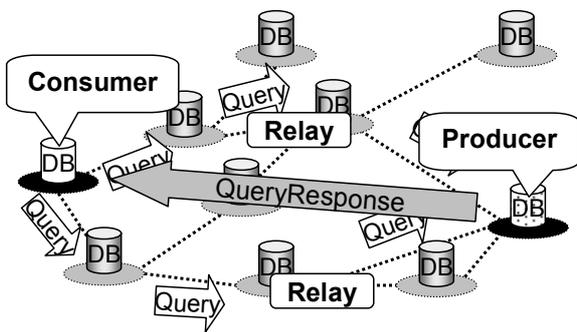


Figure 3. Peer-to-peer Distributed Search Application

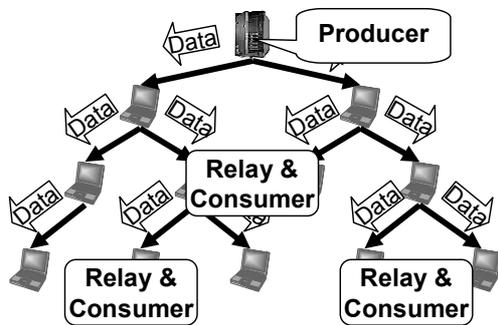


Figure 4. Peer-to-peer Application Layer Multicast

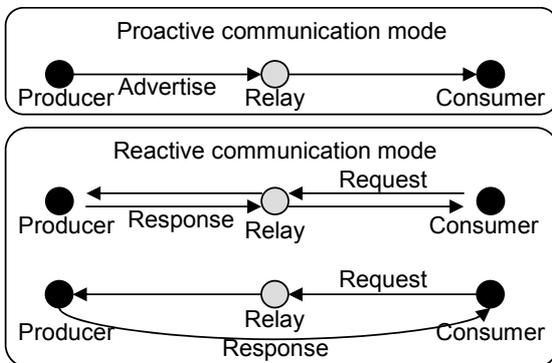


Figure 5. Roles and Communication Modes

Reactive communication mode: This mode represents the transmission of information that requires a response. It is generally used by a node which requests certain services or resources provided by other nodes. This communication mode consists of request and response messages.

As shown in Fig. 6, the peer-to-peer architecture also supports basic communication types such as unicast, broadcast, and multicast. Furthermore, we defined three unicast communications types: normal unicast, multi-hop unicast, and multi-destination unicast (See Fig. 7).

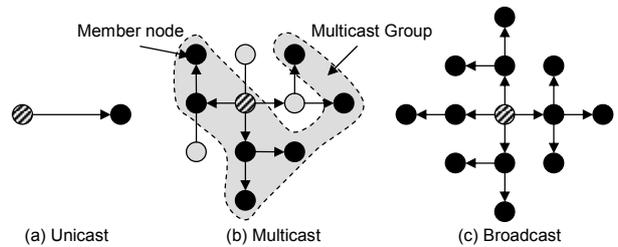


Figure 6. Peer-to-peer Communication Types

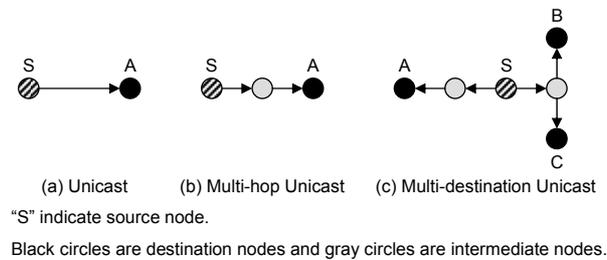


Figure 7. Unicast Communication Types

Figure 7. Unicast Communication Types

2.3. Node Naming

In peer-to-peer networks, each node should have a unique name. This name will be extensively used for various purposes such as node search, routing information to a node and cache table management. A naming system should satisfy the following requirements.

Uniqueness: Peer-to-peer node names should be unique within the scope of a community.

Manageability, Simplicity: Peer-to-peer names should be able to be autonomously generated in particular for pure peer-to-peer networks. Its naming structure should use simple for manageability and generality.

Scalability: The peer-to-peer naming system has to support extremely large number of nodes.

Anonymity, Privacy: Nobody should be able to infer any type of private information from node names.

Security: Node names should not be arrogated.

Independency: Node names should be independent of location, user, transport protocol, and application, and so on.

Considering the above requirements, we have defined a naming system that is based on UUID [6] where a node name is assigned by the node itself. UUID is, however, not human-readable, and an alias name system for human-readability may be required.

2.4. Peer-to-peer Message Routing

Message routing is one of the key mechanisms for realizing efficient and reliable peer-to-peer communication. In the

traditional Internet, routing is performed by a router according to a routing table it holds. However, since a peer-to-peer node freely enters and leaves a peer-to-peer network, the topology of a peer-to-peer network changes very frequently. Routing based on a stable routing table is hence inadequate and inefficient for a peer-to-peer network. Since peer-to-peer nodes communicate with each other across heterogeneous network environments, a transport layer independent routing mechanism is required. Furthermore, Autonomous decision on forwarding a message to the next node by each node itself is also desirable. In this architecture, two routing mechanisms are proposed. Peer nodes could select either of them automatically according to their current peer-to-peer environment. Both mechanisms are name-based routing.

Table 1 compares these mechanisms briefly. The first one is a routing mechanism that finds a source route toward a destination node by using broadcast. In this method, a node looks for the route to a destination node by broadcasting a request message. Good examples of this type of mechanism are ad-hoc network routing mechanisms such as Dynamic Source Routing (DSR) [7]. One advantage of this method is its simplicity because it does not need complicated routing protocols and a particular server. Obviously, it consumes a lot of network resources and incurs long delay. Hence, performance and efficiency could not be sufficiently ensured in large peer-to-peer network environments.

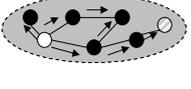
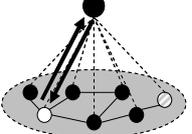
Method	Broadcast	Server
Figure		
Description	A node sends a search request message by broadcast, to find a source route toward a destination node.	A node requests the control node to provide routing information when it tries to send a message.
Advantage	Topology management and a control node are not needed.	It's possible to detect a split of network in real time.
Drawback	Broadcast request message spends a lot of network resource.	The control node has centralized load.

TABLE 1. Peer-to-Peer Routing Method

The second one is a routing mechanism that uses a special server called Control node. In this method, a node finds a source route toward a destination node by using a route discovery function in a control node. One advantage of this method is that it is easy to find the efficient route toward a destination node because the control node grasps the topology of the entire network. The drawback of this method is lack of scalability and fault tolerance, because a control node may be overloaded for a large peer-to-peer network, and the entire network is to be damaged due to the failure of a control node.

2.5. Network Topology Management

Though entire topology information is monitored and optimized by a control node, this method increases control node processing load for a large-scale peer-to-peer network. It is desirable for each peer node to provide autonomous topology optimization function that can adjust its local topology according to topology information around itself. In our architecture, each node aggregates evaluation values (e.g. hop count, bandwidth, or connection cost etc.) about the connections with its neighbor peer nodes. Then, such

information is broadcasted to all of the nodes within the 2-hops range. Each node collects local topology information around itself, and analyzes whether its current neighbor nodes are the best ones by comparing the current evaluation values with the new average evaluation values with nodes within the 2-hops range when it changes its connection with another one. Then, a connection adjustment will be executed if a local optimal node is found. We believe that this mechanism will optimize the peer-to-peer network topology autonomously and peer-to-peer communication performance will be improved.

2.6. Peer-to-Peer Multicast Communication

In a peer-to-peer network, multicast communication can be used by various applications such as groupware applications. The requirements placed on peer-to-peer multicast communication are as follows.

Optimization of multicast distribution tree: Peer-to-peer multicast should be able to optimize the distribution tree even in dynamic network environments where nodes frequently join and leave.

Dynamic management of members in a multicast group: Peer-to-peer multicast should efficiently manage members of a multicast group, because such members may change very frequently.

Efficient support for multiple multicast senders: Recent standardization in IETF focuses on single-source multicast, where it is assumed that there exists a single multicast sender. However, in a peer-to-peer network, a node can be both a multicast sender and receiver simultaneously due to its inherent symmetric nature of the peer-to-peer communication. Peer-to-peer multicast should efficiently support the case where each member in the multicast group sends and receives multicast messages simultaneously.

Fault tolerance: Peer-to-peer multicast should have mechanisms to recover from distribution tree splitting, due to the failure or the leave of a peer node.

Considering the above requirements, we adopt the bi-directional shared tree algorithm as an algorithm for constructing a multicast distribution tree and forwarding a multicast message along the tree. Figure 8 shows the example of a multicast distribution tree. Our algorithm consists of the following four steps.

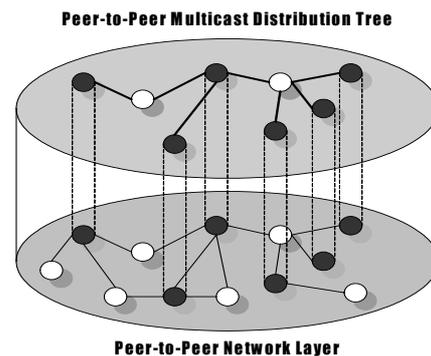


Figure 8. Peer-to-peer Multicast Distribution Tree

Step1: Finding a member node

When a peer node joins a multicast group, it must find a member node of the multicast group. In a pure peer-to-peer network, a node looks for a member node of the multicast group, using broadcast. In a hybrid peer-to-peer network, a node asks a control node about the nearest member node of the multicast group. In both cases, a peer node finds one or

more member nodes (if any) with the paths towards those nodes, as the responses to the query.

Step2: Joining the multicast group

When a peer node finds one or more member nodes, it selects one of them (e.g. the nearest member node). A peer node sends a join message to the selected member node and joins the multicast group. At the same time, a multicast routing table is generated at each node along the path toward the member node.

Step3: Forwarding a multicast message

Each member node can send a multicast message. When a member node sends a multicast message, it sends the message to all adjacent member nodes. When a member node receives a multicast message, it forwards the message to remaining adjacent member nodes. A multicast message is thus forwarded toward the member nodes along the multicast distribution tree.

Step4: Leaving the multicast group

Each member node may leave the multicast group at any time. When a member node leaves the multicast group, it sends a leave message to all adjacent member nodes. When the member node has only one neighbor member node (i.e. leaf node), a multicast routing table is discarded at each node along the path toward the member node at the same time.

3. PROTOCOL DESIGN

3.1. Protocol Overview

The peer-to-peer protocols are designed to realize the proposed peer-to-peer architecture. The protocols described herein are designed, considering the following requirements.

Extensibility: Peer-to-peer protocols should be layered, generic, and have extensibility so that they can support various peer-to-peer applications.

Utilization of existing technology: Peer-to-peer protocols should leverage existing technologies such as XML, and support existing network infrastructures such as the Internet to simplify implementation and deployment.

Independence of transport protocols: Peer-to-peer protocols should be independent of transport protocols for realizing peer-to-peer applications over heterogeneous network environments (e.g. the Internet, home networks and ad-hoc networks).

As shown in Fig. 9, the proposed peer-to-peer protocols are currently defined over TCP/IP and Bluetooth [8] and will be defined over IEEE 1394 [9]. The proposed protocols consist of the following protocols.

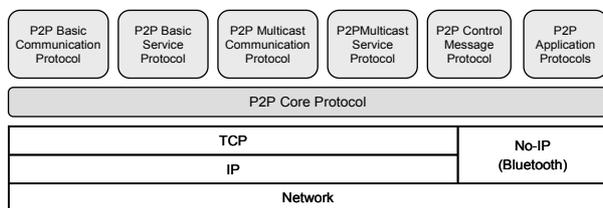


Figure 9. Protocol Stack

3.2. P2P Core Protocol

The P2P Core Protocol processes peer-to-peer messages according to the peer-to-peer communication model. Three

message types are defined in Section 2 to realize reactive and proactive communication modes. Request and response messages are defined for the reactive communication mode while an advertise message is defined for the proactive communication mode. Additionally, we define three communication types namely, unicast, multicast and broadcast. A unicast message is sent to the destination node either directly or using multi-hop unicast or multi-destination unicast. When a node receives a multicast message from a multicast member node, it forwards the received message according to the mechanism described in Section 2.6. A node sends a broadcast message to all adjacent nodes. The forwarding of a broadcast message is controlled by its hop count. The naming and the name-based message routing mechanisms of the P2P Core Protocol are independent of transport protocols.

Other protocols are defined over this protocol. Functions of these protocols are briefly described below.

3.3. P2P Basic Communication Protocol

The P2P Basic Communication Protocol establishes and releases peer-to-peer sessions. In our peer-to-peer architecture, all communications are based on a peer-to-peer session between pair adjacent peer-to-peer nodes. This protocol also has the function of exchanging node resource information such as names of its adjacent nodes, joined multicast groups, and supported peer-to-peer applications. This protocol consists of *Hello* method, *Bye* method, and *Resource Information Exchange* method. *Hello* method is used to establish a peer-to-peer session. *Bye* method is used to release a peer-to-peer session. *Resource Information Exchange* method is used to exchange node resource information between peer-to-peer nodes.

3.4. P2P Multicast Communication Protocol

The P2P Multicast Communication Protocol constructs a multicast distribution tree for multicast member nodes and forwards multicast messages along it. A node finds a member node of a multicast group, and sends a *Join* message to it, and a multicast routing table is generated at each node along the path toward it at the same time. When a node wants to send a multicast message, it sends the message towards the adjacent member nodes based on the multicast routing table. The multicast messages are forwarded along the multicast distribution tree using the bi-directional shared tree mechanism described in Section 2.6. When the node leaves the multicast group, it sends a *Leave* message to the adjacent member nodes.

3.5. P2P Basic Service Protocol

In hybrid peer-to-peer network environments, the P2P Basic Service Protocol is used between a peer-to-peer node and a control node, to provide the functions for first peer discovery, source route discovery toward a destination node and network topology optimization.

3.6. P2P Multicast Service Protocol

The P2P Multicast Service Protocol is used between a peer-to-peer node and a control node, and provides the functions for member nodes discovery of a multicast group and source route discovery toward member nodes in hybrid peer-to-peer network environments. This protocol will improve the performance of multicast group communication.

3.7. P2P Control Message Protocol

The P2P Control Message Protocol provides ancillary functions such as notification of message forwarding error, keep-alive for

peer-to-peer sessions, and first peer-to-peer node discovery in pure peer-to-peer network environments. For example, an *ErrorReport* message is used to notify the source node of the forwarding error of a message. A *Diagnose* message is used to measure RTT (Round Trip Time) between peer-to-peer nodes. A *Lookfor* message is used to find the first peer node to which a node should connect in pure peer-to-peer network environments.

3.8. Example of P2P Protocol Message

All P2P protocols are defined in XML syntax. Figure 10 shows an example of a "hello" message sent by a peer node in hybrid peer-to-peer network environments. Information on the P2P Core protocol such as message type, message ID, destination node ID and source node ID is declared inside the <Core> element. Information on the P2P Basic Communication protocol is declared inside the <MsgBody> element. For the compliance with XML and independency among P2P protocols, each P2P protocol has its own namespace.

```
<Core xmlns="Namespace of P2P Core Protocol">
  <MsgType>Request</MsgType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <Destination>
    <Target>874542ab-a5c6-4305-8745-f98e56f12547</Target>
  </Destination>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <ComType>Unicast</ComType>
  <MsgBody protocol="Namespace of P2P Communication Protocol">
    <Hello xmlns="Namespace of P2P Communication Protocol">
      <Mode>hybrid</Mode>
    </Hello>
  </MsgBody>
</Core>
```

Figure 10. An Example of P2P Protocol Message

3.9. Example of Protocol Sequence

Fig. 11 shows a basic sequence when a peer-to-peer node participates in a peer-to-peer network and exchange resource information with another node in pure peer-to-peer network environments. At first, Node A sends a *Lookfor* message using network specific broadcast or multicast mechanisms (e.g. IP multicast) and receives the corresponding *LookforResponse* messages from certain nodes. Then, Node A sends a *Hello* message to one of discovered nodes (Node B) to participate in a peer-to-peer network. When a *HelloResponse* Message from

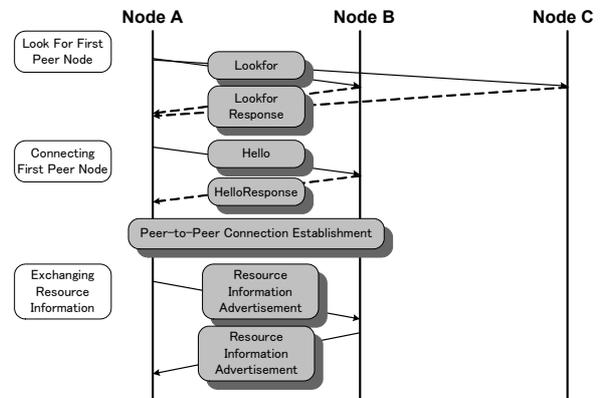


Figure 11. An Example of Sequence

Node B is received, a peer-to-peer session is established between Node A and Node B. Resource information will be exchanged using *Resource Information Advertisement* messages in the next step.

4. PROTOTYPE IMPLEMENTATION

4.1. Jupiter: Peer-to-Peer Networking Platform

We have currently implemented a prototype called *Jupiter*. *Jupiter* stands for "Jisedai Ubiquitous Peer-to-peer networking InfrasTructuRE". "Jisedai" means "next-generation" in Japanese. *Jupiter* is developed in Java (J2SE 1.3.1) [10] on Microsoft Windows 2000 and Red Hat Linux 7.2. This prototype provides the basic functions of the peer-to-peer protocols and some utility tools. Jupiter APIs have been provided for peer-to-peer application developers. Fig. 12 shows the three-tiered API design in a peer-to-peer node.

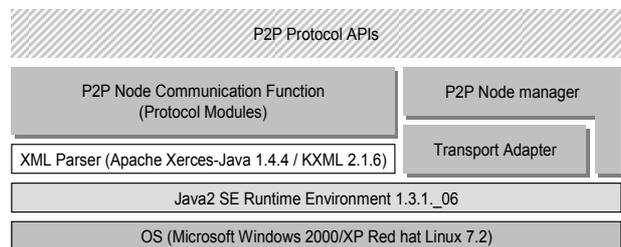


Figure 12. Jupiter Software Architecture

4.2. Jupiter API Design

The Jupiter APIs are designed based on the following requirements.

Generality: The main goal of defining the APIs is to provide the interfaces to the Jupiter platform for developing peer-to-peer applications. The APIs should include all necessary functions for peer-to-peer application development, such as generating, sending and receiving their own application messages, being notified of error occurrence, obtaining current platform conditions and so on.

Multiple Applications Supports: The APIs must be designed to accommodate multiple peer-to-peer applications but no application must be permitted to access or control the kernel operations (e.g., the operation to modify the state of the platform).

Platform Independence: The APIs should not control any of the operational events of the platform, such as starting or stopping peer-to-peer applications, configuring platform parameters for peer-to-peer applications or what transport mechanism is used under the platform. This is because they have nothing to do with running peer-to-peer applications and it could prevent platform developers from freely creating their own platforms on their target devices.

Although the APIs in Fig.12 appear to fall into just two categories (i.e. P2P Communication service API and P2P Core API) with the layered protocol stacks, there are actually several sub-divided packages in the Java2 SDK environment. The package has one root package "Jupiter", which is divided into nine sub-packages based on functionality. Figure 13 shows the tree structure of the packages.

Jupiter prepares tiered APIs for each functional unit to satisfy different level developers' requirements.

The Jupiter package consists of three parts. The first one is a core component of Jupiter, which consists of *Jupiter.protocol* and *Jupiter.core*. Peer-to-peer application developers can specify their own application protocols based on these core APIs. The second part is a collection of protocol related packages. Each sub-package corresponds to the related protocol (e.g. P2P Basic Communication Protocol). Developers can use those APIs based on the requirements of their own peer-to-peer applications. The third part is the sub-packages for peer-to-peer application management. Those APIs provide developers with management functions such as invocation and termination of a peer-to-peer application, and error handling. The sub-packages consist of *Jupiter.application* and *Jupiter.exception*.

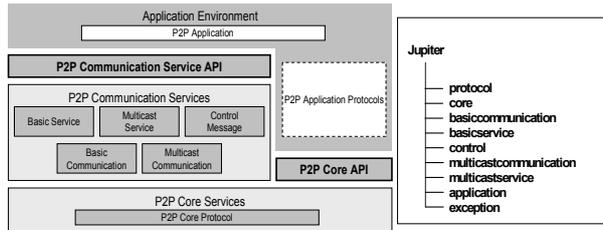


Figure 13. APIs and Its Structure

4.3. Peer-to-peer Application Developments

Based on Jupiter APIs described in Section 4.2, Instant message application [11] and Content search application [12] are developed over this platform. Instant message application is a message exchange system in which a user on a peer node can exchange messages with one or a group of users on other peer nodes without any server interference. Content search application is a distributed content search system. In this system, metadata of multimedia content is defined under a predefined metadata schema using RDF. Such metadata information is distributed over the peer-to-peer network. Each user on a peer node can find its favorite content by submitting a metadata query over the peer-to-peer network. From our experience with this system, there is a possibility that peer-to-peer metadata search systems provide more flexible and powerful search functionalities with lower maintenance cost, comparing with existing server-based keyword search systems.

5. CONCLUSION

We presented a peer-to-peer networking architecture where peer-to-peer protocols allow a variety of devices to communicate with each other. Seamless interconnection between various devices across various networks such as the Internet, home networks and wireless sensor networks could be achieved using our peer-to-peer architecture and protocols. We also described the current status of our prototype implementation.

Peer-to-peer security and privacy are important topics that we will investigate in the near future. In future communication environments, since a lot of devices will freely communicate with one another, peer-to-peer security and privacy are serious problems that must be solved. We are considering the incorporation of peer-to-peer security and privacy into our peer-to-peer architecture. Additionally, we will continue to develop new peer-to-peer applications and will evaluate generality and performance of our peer-to-peer architecture and protocols in more depth.

5. REFERENCES

- [1] The Gnutella protocol specification v4.0, <http://dss.clip2.com/GnutellaProtocol04.pdf>
- [2] KaZza, <http://www.kazaa.com/us/index.htm>
- [3] Clarke I., Sandberg O., Wiley B., and Hong T. W. "Freenet: A distributed anonymous information storage and retrieval system", In Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability, Berkeley, California, June 2000.
- [4] Groove, <http://www.groove.net/home/>
- [5] Sun Microsystems, "Project JXTA", <http://www.jxta.org/>
- [6] Paul J. Leach, Rich Salz, "UUIDs and GUIDs", IETF Internet Draft, draft-leach-uuids-guids-01.txt (Expired), Feb. 1998.
- [7] David B. Johnson, David A. Maltz, and Yih-Chun Hu, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)", IETF Internet Draft, draft-ietf-manet-dsr-10.txt (Working in Progress), July 2004.
- [8] Bluetooth <http://www.bluetooth.com/>
- [9] IEEE 1394 <http://www.1394ta.org/>
- [10] Java J2SE 1.3.1 <http://java.sun.com/j2se/1.3/>
- [11] Hiromitsu Sumino, et al, "Design and Implementation of a Multicast Instant Messaging System on the Mobile P2P Network", First International Conference on Mobile Computing and Ubiquitous Networking (ICMU), Yokosuka, Japan. 8-9, January, 2004
- [12] Hiromitsu Sumino, et al, "A Mobile Multimedia Content Search using RDF", MDM2003 4th International Conference on Mobile Data Management, Melbourne, Australia. 21-24, January, 2003.