PUCC Basic Protocol

(Version 3.0 - March 22, 2012)

Peer-to-Peer Universal Computing Consortium (PUCC)

Intellectual Property Notice

Table of Content

1. Introduction

At the present day, various digital equipments are connected together using various communication protocols such as internet, mobile network and so on. In the field of personal area network and home network, various networks such as ECHONET, DLNA (UPnP) are developed and utilized. These technologies are highly effective for specific problem areas and have been getting the position as standard. However, focusing on each technology, there are also some problems for users, for example, captive of venders, no interoperability.

In these surroundings we expect that with the widespread use of mobile phones, digital cameras, printers and digital TVs, the technologies which enable these equipments to connect to global networks, to be seamlessly utilized from various locations, are strongly desired. The goals of PUCC are just such enabling technologies. For bring about breakthrough, PUCC aims to research and develop computing environment which provide interoperability of various networks and advanced services using the peer-to-peer communication technology without troublesome procedure of users.

This PUCC Basic Protocol specification provides the protocol stack structure for PUCC Basic Protocol and protocol specification for providing basic PUCC communication functions for PUCC applications.

2. Terminology

2.1. Definitions

2.1.1. PUCC architecture definitions

The following terms are defined in PUCC Architecture Specification.

- Control Node;
- Node;
- Node ID;
- Community;
- Community ID;
- Multicast Group;
- Multicast Group ID;
- Communication Mode;
- Communication Type.

2.1.2. Definitions in this document

- Message;
- Message ID;
- Application Protocol ID;
- User ID;
- Session ID;

2.2. Abbreviations

TCP	Transmission Control Protocol
UDP	User Datagram Protocol
IP	Internet Protocol
XML	eXtensible Markup Language
UUID	Universal Unique Identifier
FQDN	Fully Qualified Domain Name
P2P	Peer-to-Peer
WAP	Wireless Application Protocol
NAT	Network Address Translation
MIME	Multipurpose Internet Mail Extensions
TLS	Transport Layer Security
SSL	Secure Socket Layer

URL	Unique Resource Locator
URN	Uniform Resource Names
URI	Universal Resource Identifier
DTD	Document Type Definition
L2CAP	Logical Link Control and Adaptation Protocol
OBEX	Object Exchange Protocol
HTTP	Hypertext Transfer Protocol
BEEP	Blocks Extensible Exchange Protocol

3. References

- PUCC Architecture Specification Version 1.0

- T. Bray et al., “Extensible Markup Language (XML) 1.0 (Second Edition) ,” W3C Recommendation, October 2000.

- R. Fielding et al., “Hypertext Transfer Protocol -- HTTP/1.1,” RFC2616, June 1999.

- J. Postel, “Transmission Control Protocol,” RFC793, September 1981.

- J. Postel, “User Datagram Protocol,” RFC768, August 1980.

- J. Postel, “Internet Protocol,” RFC791, September 1981.

- The Bluetooth Special Interest Group, “Bluetooth”

- T. Berners-Lee et al., “Uniform Resource Identifiers (URI): Generic Syntax,” RFC2396, August 1998.

- Infrared Data Association, “OBEX(Object Exchange) Protocol”.

4. Goals and Requirements

4.1. Goals

The goals of this document are:

- To define general, scalable, extensible peer-to-peer protocols for various devices such as mobile phone, home appliance and digital equipments.
- To leverage existing standards where possible, especially existing and evolving Internet, home network, mobile network standards.

4.2. Requirements

4.2.1. Extensibility

Each file (parameter) for describing the communication message in peer-to-peer protocols should be defined independent of any transport protocol so that the defined peer-to-peer communication message could be encapsulated by other transport protocols easily.

4.2.2. Layered design

For providing a scalable and extensible application environment for each node and each protocol, layered design for the nodes themselves and the communication protocols between nodes is highly desirable. Protocol components should be defined independently.

4.2.3. Protocol functions

When designing peer-to-peer protocols, the following functions are typical of those that should be considered: Node discovery, node information exchange among nodes, and function negotiation among nodes.

4.2.4. Efficiency

Performance of peer-to-peer protocols should be optimized. To this end, various technologies such as route optimization, resource sharing, and resource caching should be considered when designing peer-to-peer protocols. Since wireless networks have specific characteristics such as latency, low bandwidth, and expensive data packets compared to wired networks, traffic on wireless networks should be minimized.

4.2.5. Co-existence with (Mobile) Internet

The peer-to-peer architecture should be realized over the existing (Mobile) Internet, home network, Internet. To this end, existing components of the (Mobile) Internet such as i-mode, WAP, firewall, and NAT. should be considered and utilized, when designing the Peer-to-Peer architecture. Since the current work is considered to be implemented based on the i-mode Mobile Internet, the i-mode gateway and NAT should be taken into account.

4.2.6. Convergence with existing standards

When designing the peer-to-peer protocols, the use of existing standards should be considered where applicable. Such standards may include but not be limited to: W3C standards such as SOAP, and IETF

standards such as HTTP, TCP, and BEEP.

5. Protocol Overview

5.1. Protocol Design

This specification defines PUCC Basic Protocol for exchanging PUCC application messages over the PUCC network.

The PUCC Basic Protocol is independent of IP transport protocol. Consequently the PUCC Basic Protocol can be implemented on TCP, Bluetooth and IEEE1394 and so on.

The PUCC Basic Protocol is designed as a layered structure (Figure 1). Each layer provides a set of services and/or functions to other services and applications through well-defined interfaces. Each layer of the protocol stack is accessible to the layers above.

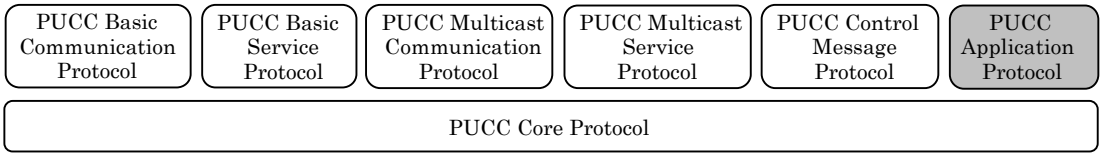


Figure 1. Protocol stack for PUCC Basic Protocol

The PUCC Basic Protocol is divided in two layers.

5.1.1. PUCC Core Protocol

This protocol provides the common peer-to-peer transport functions needed to establish or process a PUCC connection and/or session between two nodes on the PUCC network. The functions provided by this protocol include designate communication mode, provide communication type, source information, destination information, traces route information and so on.

5.1.2. Protocols over PUCC Core Protocol

On top of the PUCC Core protocol, several PUCC communication and application protocols are defined. The protocols are functional independent of each other and each protocol provides a specific PUCC function. The functions defined on this protocol layer include initialize a PUCC connection and/or session, release an established connection and/or session, keep an established connection and/or session alive, and transfer specific PUCC application message.

5.1.2.1. PUCC Basic Communication Protocol

The PUCC Basic Communication Protocol is defined to realize the establishment and release of a PUCC connection and/or session. Additionally, this protocol has the function of exchanging resource information of a node such as names of its adjacent nodes, joined multicast group, and running PUCC applications in the node.

5.1.2.2. PUCC Basic Service Protocol

In the hybrid PUCC architecture, the PUCC Basic Service Protocol is used between a PUCC node and a control node, to efficiently enhance communication between PUCC nodes.

5.1.2.3. PUCC Multicast Communication Protocol

The PUCC Multicast Communication Protocol is defined to construct a multicast distribution tree among multicast member nodes. Multicast messages are forwarded over it.

5.1.2.4. PUCC Multicast Service Protocol

In the hybrid PUCC architecture, the PUCC Multicast Communication Protocol is used between a PUCC node and a control node to efficiently enhance multicast communication between PUCC nodes.

5.1.2.5. PUCC Control Message Protocol

The PUCC Control Message Protocol is defined to provide ancillary functions such as notification of a message forwarding error, keep-alive of PUCC connection and/or session, first PUCC node discovery and searching a PUCC node.

5.1.2.6. PUCC Application Protocols

The PUCC Application Protocols are defined to provide specific functions for each application. These protocols are designed specifically for each application.

5.2. Operation mode

Three PUCC architectures are defined for the PUCC network.

(For details see the PUCC architecture specification)

5.2.1. Light PUCC architecture

This is a minimum architecture for the PUCC network. There are only PUCC nodes in the Light PUCC architecture. Messages are sent from one PUCC node to another directly. Passing messages via several intermediary PUCC nodes is optional. A PUCC node as a client need not support relay function. Routing information is discovered by broadcasting an inquiry message to the network.

In this architecture, a PUCC node needs to be accessible to PUCC Core Protocol, PUCC Basic Communication Protocol and PUCC Control Message Protocol. Those protocols are PUCC Basic Protocol – light profile.

5.2.2. Pure PUCC architecture

There are only PUCC nodes in the pure PUCC architecture. Messages are sent from one PUCC node to another directly or by passing them via several intermediary PUCC nodes. Routing information is discovered by broadcasting an inquiry message to the network.

In this architecture, a PUCC node needs to be accessible to PUCC Core Protocol, PUCC Basic Communication Protocol, PUCC Multicast Communication Protocol and PUCC Control Message Protocol include in PUCC Basic Protocol. To communicate with nodes of the light PUCC network, PUCC Basic Protocol – light profile is needed.

5.2.3. Hybrid PUCC architecture

The hybrid PUCC architecture consists of PUCC nodes and a control node. The control node provides a topology optimization function and security function as well responding to requests from PUCC nodes by returning routing

information. The gateway node collects topology information on the pure PUCC network and passes it to the control node. A PUCC node in a hybrid PUCC network reports its own existence and adjacent nodes to the control node and can efficiently communicate with other PUCC nodes by using the routing information provided by the control node.

In this architecture, a PUCC node needs to be accessible to PUCC Core Protocol, PUCC Basic Communication Protocol, PUCC Multicast Communication Protocol, PUCC Basic Service Protocol, PUCC Multicast Service Protocol, and PUCC Control Message Protocol. Those protocols are PUCC Basic Protocol. The hybrid PUCC network can be connected to the pure PUCC network using a gateway node. The gateway node has a function to combine the hybrid PUCC network and the pure PUCC network.

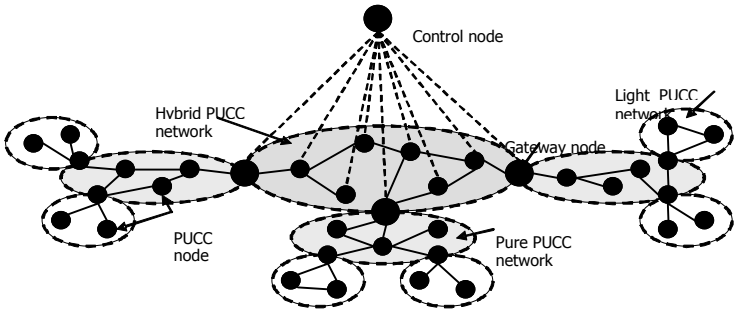


Figure 2. Structure of PUCC network

The following are protocols used between a PUCC node and a control node in the hybrid PUCC architecture, and between PUCC nodes in the light PUCC network.

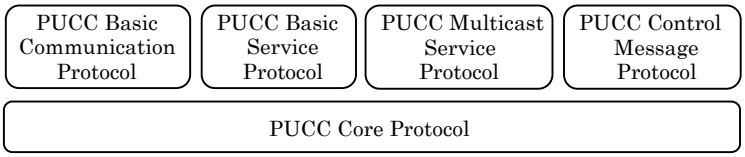


Figure 3. Protocols between a PUCC node and a control node

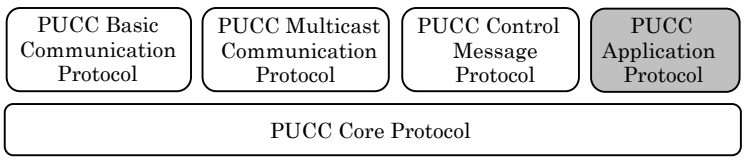


Figure 4. Protocols between PUCC nodes

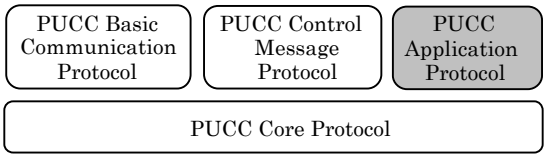


Figure 5. Protocols between PUCC nodes in the light PUCC network

5.3. Communication mode

Two communication modes are defined for PUCC communications. They are proactive communication mode and reactive communication mode. The two communication modes are according to different communication needs.

5.3.1. Proactive communication mode

In the proactive communication mode, a node initiates communication to other nodes without any request.

5.3.2. Reactive communication mode

In the reactive communication mode, a node initiates communication only when it receives a communication request from another node.

5.4. Implementation Design

The PUCC Basic Protocol has a framing mechanism that permits simultaneous and independent exchanges of messages between nodes. Messages are arbitrary MIME content. In this specification, all messages are structured using XML.

The messages are defined using the following XML envelope:

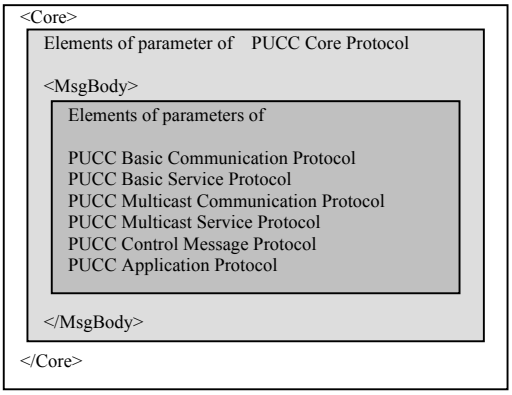


Figure 6. Message envelope using XML

Message blocks are distinguished from each other using the XML name space.

The concepts of PUCC Basic Protocol design include:

5.4.1. Framing

Message framing in the PUCC Basic Protocol is defined using the XML envelope. PUCC Basic Protocol messages are encapsulated using <Core> start tag and ended by </Core>, and so can be distinguished from other messages.

5.4.2. Encoding

The PUCC Basic Protocol messages are encoded in text/xml MIME content and use UTF-8 character code set for describing protocol messages.

5.4.3. Error Reporting

Use network communication failure, XML format error, and protocol format error for each layer. Define error description in PUCC Control Message Protocol for each protocol.

5.4.4. Asynchrony

PUCC communication messages are distinguished from each other by using Message ID.

5.4.5. Authentication

We use bidirectional Challenge-Response protocols (See Chapter 6.2 and Appendix K).

5.4.6. Message Signature

We use XML signature (See Appendix K).

5.4.7. Encryption

We use XML Encryption (See Appendix K).

5.4.8. Protocol Stack

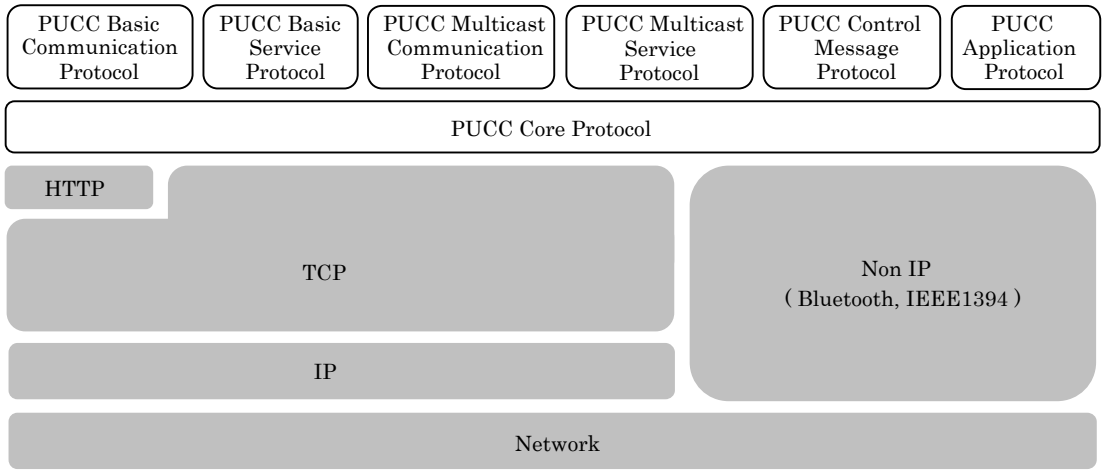


Figure 7. Protocol stack for PUCC Basic Protocol (including transport and network layer)

5.5. ID definition

The PUCC Basic Protocol refers to PUCC nodes, PUCC messages, communities, multicast groups, users, sessions and protocols using the different kind of IDs. The PUCC Basic Protocol uses 7 types of IDs to distinguish the entities of a PUCC network.

5.5.1. Node ID

The Node IDs refer to PUCC nodes. A Node ID must be unique in a PUCC network. For example, UUID can be used as Node ID.

5.5.2. Message ID

The Message IDs refer to PUCC messages. A Message ID must be eternally unique in a PUCC network. For example, we can generate unique Message IDs using "[node provided sequence number].[current time (ISO

8601)]@[node ID]".

5.5.3. Multicast Group ID

The Multicast Group IDs refer to PUCC multicast groups. A Multicast Group ID is a unique string that distinguishes multicast groups. URN is recommended for the notation of Multicast Group IDs.

5.5.4. Community ID

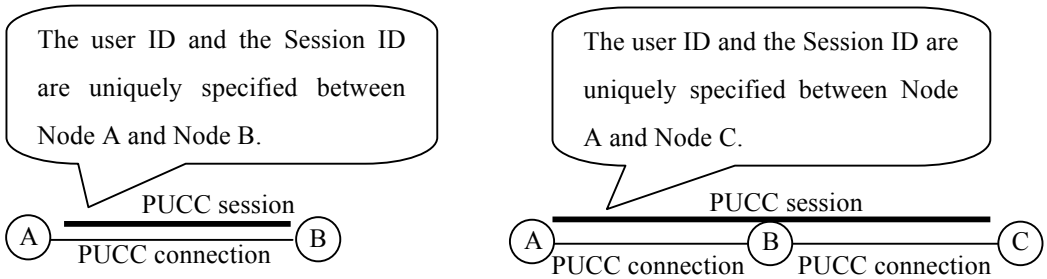
The Community IDs refer to PUCC communities. A Community ID is globally unique. URN is recommended for the notation of Community IDs.

5.5.5. User ID

The User IDs refer to the IDs that identify users who use the PUCC session. A User ID must be unique on the PUCC node.

5.5.6. Session ID

The Session IDs refer to the IDs that identify PUCC sessions. A PUCC session consists of a pair of user IDs; one which sets up the PUCC session and the other to which the PUCC session is set up. The Session ID shall be unique for the PUCC nodes between which the PUCC session is established. The Session ID is specified by either the Initiator node or the Responder node. In order to maintain the uniqueness of the Session ID, for example, the message ID of the Hello message may be used as the Session ID, which is then given by the Initiator node.



Single hop session

(The PUCC session is established between Node A and Node B.))

Multi hop session

(The PUCC session is established between Node A and Node C.))

Figure 8. PUCC connection and User ID/Session ID

5.5.7. Protocol ID

The Protocol IDs refer to PUCC Application Protocols. The Protocol ID is used for distinguish the PUCC Application Protocols. A Protocol ID is globally unique in the system. URI is recommended for the notation of Application IDs.

6. PUCC Basic Protocol

6.1. PUCC Core Protocol

The following is definition of fields (Parameter) in Core Protocol.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Core	Following XML fragment	-	-	1	Required	
		Xmlns	URI	1	Required	
ComType	"Unicast","Multicast", "Broadcast"	-	-	1	Required	
MsgID	Message ID (String)	-	-	1	Required	
ReplyID	Message ID (String)	-	-	1	MsgType dependent	
MsgType	"Request","Response"," Advertise"	-	-	1	Required	
CommunityID	Community ID (String)	-	-	1	Optional	
Source	Node ID (String)	-	-	1	Required	
Destination	Route,Target element	-	-	1	ComType dependent	
Route	Route,Target element	-	-	1	Optional	
		Node	Node ID(String)	1	Required	
Target	Node ID, Multicast GroupID (String)	-	-	1	Required	
TraceRoute	Route element	-	-	1	Optional	
Route	Route element	-	-	1	Required	
		Node	Node ID(String)	1	Required	
HopCount	Hop count (Integer)	-	-	1	Optional	
GatewayAction	"Hook","Hop"	-	-	1	Optional	
SessionID	Session ID(String)	-	-	1	Optional	
Signature	SignedInfo, CanonicalizationMethod, SignatureMethod, Reference, and SignatureValue are set in the element as specified by W3C XML Signature.	-		1	Optional	
		Xmlns	http://www.w3c.org/2000/09/xmlsig#	1	Required	
MsgBody	XML document	-	-	1	Optional	Either MsgBody or EncryptedData is required when encryption is not used.
		Protocol	URI	1	Required	
EncryptedData	EncryptionMethod, ds:KeyInfo, EncryptedKey, CipherData, and CipherValue are set in the element as specified by W3C XML Encryption.	-	-	1	Optional	When encryption is used
		Type	http://www.w3c.org/2001/04/xmlenc#Element	1	Required	
		Xmlns	http://www.w3c.org/2001/04/xmlenc#	1	Required	

Table 1: Fields of PUCC Core Protocol

6.1.1. Core element

The Core element is defined for encapsulating PUCC messages to distinguish them from other data. Every PUCC message should be structured under the Core element. This element has xmlns attributes which designate the namespace of the PUCC Core Protocol. Twelve elements are defined under Core element.

6.1.2. ComType element

The ComType element designates the communication type that the message uses.

Three communication types are defined in this specification: unicast, multicast, and broadcast. When the communication type is described as unicast, the message should be sent to one node indicated by node ID. When the communication type is described as broadcast, the message should be sent to all nodes except for a control node within the PUCC network. When the communication type is described as multicast, the message should be sent to all nodes within the given group. This field is essential.

6.1.3. MsgID element

MsgID element is unique and distinguishes the message from other messages. In this specification, the message ID value is defined as "[node provided sequence number].[current time(ISO 8601)]@[node ID]". This field is essential in each PUCC protocol message envelope.

Ex) 12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352

6.1.4. ReplyID element

The Reply ID element is to designate the MsgID of the message replied. This field is essential in response message (MsgType="Response"). In other cases, it is ignored.

6.1.5. MsgType element

The MsgType element is to designate which PUCC message type is used by the current message. In this specification, there are three PUCC message types: advertise, request, and response. This field is essential in each PUCC protocol message envelope.

6.1.5.1. Sequence of Messages

6.1.5.1.1. Advertise message

This is a message that demands no response. A sequence consists one advertise message named "Proactive Communication Mode".

6.1.5.1.2. Request message

This is a message that demands a response from the receiving nodes.

6.1.5.1.3. Response message

This is a message issued in response to a request message. A sequence consisting of a request message and a response message is named "Reactive Communication Mode".

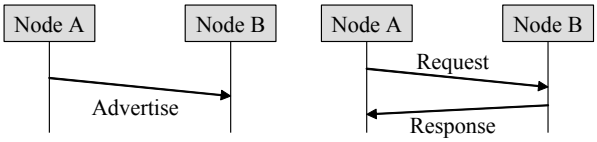


Figure 9. Sequences of Proactive (left) and Reactive (right)

6.1.5.2. Methods of Transmitting Response messages

There are two methods of transmitting response messages. The transmission method selected depends on the node's decision.

6.1.5.2.1. Direct response method

The direct response method is to transmit the response message directly to the source node with reference to Source element. Even if the request message passed through several nodes, the response message is transmitted directly to the source node.

6.1.5.2.2. Chained response method

The chained response method is to transmit the response message along the path traversed by the request message with reference to TraceRoute element.

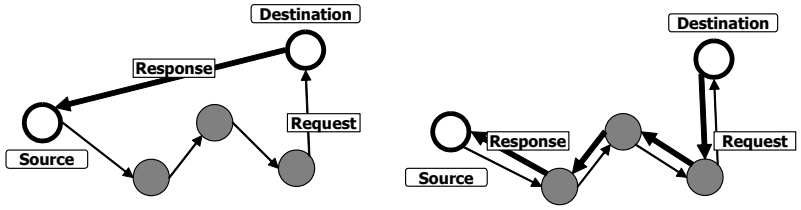


Figure 10. Direct Response (left) and Chained Response (right)

6.1.6. CommunityID element

The CommunityID element designates a community. All messages must contain this element. URN is recommended for the notation of CommunityID. If CommunityID element is empty or not specified, it is regarded as the default community. If a node receives a message with different community ID, the node discriminates the message.

Ex) urn:pucc:community:DoCoMo

6.1.7. Source element

The Source element designates the Node ID of the node that sent the message. This element is essential when a PUCC node sends any kind of message.

6.1.8. Destination element

The Destination element designates the destination of the message; the route taken to reach the destination is specified using the Node IDs of the nodes to be transited. When the communication type of the message is unicast, this field is essential. When the communication type is broadcast, this field should be dropped. When the communication type of the message is multicast, this field is essential. When the communication type is multicast,

this element has target element which has multicast group ID.

Destination element does not include Node ID of source node. It is designed by source node.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Destination	Route,Target element			1	ComType dependent	
Route	Route,Target element			1	Optional	
		Node	Node ID	1	Required	
Target	Node ID, Multicast GroupID (String)			1	Required	.

Table 2: Destination element

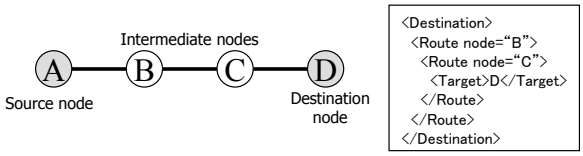


Figure 11. A sample of a route map and a notation of Destination element

6.1.9. TraceRoute element

The TraceRoute element contains the route passed by a message. When this element is used (the source node designates TraceRoute), every node on the message route appends its own node ID information to this element. TraceRoute includes the Node IDs of source node and destination node. This element is essential for the following cases.

1. A message belongs to Resource Information Exchange method or Diagnose method
2. ComType element = "Broadcast" and MsgType element = "Request"

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
TraceRoute	Route element	-	-	1	Optional	
Route	Route element	-	-	1	Required	
		Node	Node ID(String)	1	Required	

Table 3: TraceRoute element

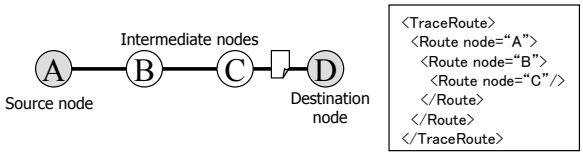


Figure 12. A sample of a route map and a notation of TraceRoute element

6.1.10. HopCount element

The HopCount element designates the maximum number of nodes the message should pass through. When the message reaches a node, the hop counter is decreased by one, and when the hop count reaches zero, the sent message should be destroyed. This field is essential in Broadcast message (ComType="Broadcast") but not "flooding". In other cases, it is ignored. If HopCount is not described in a Broadcast message, the value of HopCount is construed as infinity.

6.1.11. GatewayAction element

The GatewayAction element is defined for designating the behavior of a gateway node when it receives a broadcast message.

Value	Behavior
Hop (Default)	The gateway node forwards the broadcast message from a pure PUCC network to a hybrid PUCC network.
Hook	The gateway node stops forwarding the broadcast message from a pure PUCC network.

Table 4: Fields of PUCC Core Protocol

When a gateway node receives a broadcast message with GatewayAction = "Hook", if there is a corresponding service for the message in the control node, the gateway node sends a message to the control node as a proxy.

6.1.12. SessionID element

The SessionID element is an identifier to identify the transmission and reception of a PUCC message per user as a session between the PUCC nodes. The PUCC session can be uniquely identified by a pair of user IDs; one on the side that sets up the PUCC session and the other on the side that PUCC session is set up. To the Session ID element, the side that sets up the PUCC session, in other words, the side that transmits the Hello message, allocates and sets the value that indicates the pair mentioned above. The PUCC messages transmitted and received in the same PUCC session must set the same value in the SessionID element.

The following shows the relationships among the elements that constitute the PUCC session; With or without the user ID setting on the session setting up side and the side that the session is set up, and necessity of the Session ID element. When the user ID of the session setting up side or the user ID of the side that the session is set up is omitted, it is considered that the user ID indicating that 'the user does not exist' is set.

Table 5: Setting of the user ID and session ID

	User ID of the session setting up side	User ID of the side that the session is set up	SessionID element
1	Without setting	Without setting	Not necessary
2	With setting	With setting	Necessary
3	With setting	Without setting	Necessary

4	Without setting	With setting	Necessary

Between the user of the session setting up side and the user of the side that the session is setup, only one session can be setup simultaneously. Multiple setups of sessions are not allowed. When the user of the session setting up side transmits a Hello message to the user that a session has already been set up, the user of the side that the session is set up returns a HelloResponse(Failure) message by setting “SessionAlreadyEstablished” in the reason element (cause of failure), and multiple sessions are not setup.

The SessionID element is not set in the Lookfor message or LookforResponse message.

The user of the session setting up side and the user of the side that the session is set up do not perform relay of the multi-hop unicast, multicast or broadcast messages.

6.1.13. Signature

The Signature is set when an authentication code is set for the PUCC message. In the Signature, SignedInfo, CanonicalizationMethod, SignatureMethod, Reference and SignatureValue are set as specified by the W3C XML Signature.

6.1.14. MsgBody element

The MsgBody element is defined for encapsulating the upper PUCC protocols and application message fields to distinguish it from the PUCC Core Protocol. MsgBody element should be the last child of the PUCC Core Protocol and the protocol attribute include protocol ID to distinguish PUCC Application Protocols encapsulate by the MsgBody element.

6.1.15. EncryptedData element

The EncryptedData element encrypts and sets MsgBody element and other following elements when the PUCC message is encrypted. In the element specified by the W3C XML Signature, EncryptionMethod, ds:KeyInfo, EncryptedKey, CipherData and CipherValue are set.

6.2. PUCC Basic Communication Protocol

The PUCC Basic Communication Protocols provide the basic communication services to the nodes on the Peer-to-Peer network for communicating with each other. The PUCC Basic Communication Protocol includes

- Hello method

The Hello method is used to establish a PUCC connection between PUCC nodes and/or a PUCC session between PUCC users. The Hello method is a Reactive (Request/Response) type method and consists of a Hello message and a HelloResponse message.

- Bye method

The Bye method is used to inform other nodes before the node release a PUCC connection and/or a PUCC session to the network. The Bye method is a Proactive (Advertise) type method and consists of a Bye message.

- Resource Information Exchange method

The Resource Information Exchange method is used to exchange resource information on other nodes on the network. The Resource Information Exchange method combines Reactive and Proactive methods. The reactive type resource exchange method consists of a ResourceInformationRequest message and a ResourceInformationResponse Message. The proactive type resource exchange method consists of a ResourceInformationAdvertise message.

6.2.1. Hello method

6.2.1.1. Hello message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Hello	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
Mode	"Hybrid","Gateway"," ControlNode",	-	-	1	Optional	Default ="Pure"
InitiatorUserID	Initiator User ID (String)	-	-	1	Optional	
ResponderUserID	Reponder User ID (String)	-	-	1	Optional	
Authentication	for the first Hello -	-	-	1	Optional	
	for the second Hello HashAlgorithm, Response, Challenge element					
Response	Response Code (hexBinary)	-	-	1	Optional	for the second time only
Challenge	Challenge Code (hexBinary)	-	-	1	Optional	for the second time only
RequestedHashAlgorithm	Hash Algorithm that makes a request to the response of the Response Code (String)	-	-	1	Optional	for the second time only

RequestedSecurity	for second Hello CipherAlgorithm, HashAlgorithm, IterationCount, UpdateCount element.	-	-	1	Optional	for the second time only
CipherAlgorithm	Cipher Algorithm	-	-	1	Optional	for the second time only
HashAlgorithm	Hash Algorithm	-	-	1	Optional	for the second time only
IterationCount	Hash repeat counts	-	-	1	Optional	for the second time only
UpdateCount	Number of session key updates	-	-	1	Optional	for the second time only
RequestedCapability	ExtendedProtocol element	-	-	1	Optional	
ExtendedProtocol	"MulticastCommunication","MulticastService"	-	-	Multiple	Required	
		ID	Number(Integer)	1	Required	

Table 6: Fields of Hello message

The Hello message is mapped as per following table.

Element name	Element Value	Attribute name (if it has)	Attribute Value	Occurrence	Status	Description
Core	Following XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast"	-	-	1	Required	
MsgID	Message ID (String)	-	-	1	Required	
ReplyID	Message ID (String)	-	-	1	unused	
MsgType	"Request"	-	-	1	Required	5
CommunityID	Community ID (String)	-	-	1	Optional	
Source	Node ID (String)	-	-	1	Required	
Destination	Route,Target element	-	-	1	Required	
Route	Route,Target element	-	-	1	Optional	
		Node	Node ID(String)	1	Required	
Target	Node ID(String)	-	-	1	Required	
TraceRoute	Route element	-	-	1	Optional	
Route	Route element	-	-	1	Optional	
		Node	Node ID(String)	1	Required	
HopCount	Hop count (Integer)	-	-	-	unused	
GatewayAction	"Hook","Hop"	-	-	-	unused	
SessionID	Session ID(String)	-	-	1	Optional	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI	1	Required	

Table 7: Mapping of Hello message to PUCC Core Protocol

The following is a sample of Hello message without user ID and session ID.

```
<Core xmlns=" Namespace of PUCC 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 PUCC Basic Communication Protocol" >
    <Hello xmlns=" Namespace of PUCC Basic Communication Protocol" >
      <RequestedCapability>
        <ExtendedProtocol ID=" 1" >MulticastCommunication</ExtendedProtocol>
      </RequestedCapability>
    </Hello>
  </MsgBody>
</Core>
```

Figure 13. Sample of Hello message without user ID and session ID

The following is a sample of Hello message with session ID and user ID.

```
<Core xmlns=" Namespace of PUCC 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>
  <SessionID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</SessionID>
  <MsgBody protocol=" Namespace of PUCC Basic Communication Protocol" >
    <Hello xmlns=" Namespace of PUCC Basic Communication Protocol" >
      <InitiatorUserID>urn:pucc:user:A</InitiatorUserID>
      <ResponderUserID>urn:pucc:user:B</ResponderUserID>
      <RequestedCapability>
        <ExtendedProtocol ID=" 1" >MulticastCommunication</ExtendedProtocol>
      </RequestedCapability>
    </Hello>
  </MsgBody>
</Core>
```

Figure 14. Sample of Hello message with user ID and session ID

The following is a sample of Hello message with session ID and user ID used to establish a multihop session.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <MsgType>Request</MsgType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <Destination>
    <Route Node="874542ab-a5c6-4305-8745-ababcdcdefef">
      <Target>874542ab-a5c6-4305-8745-f98e56f12547</Target>
    </Route>
  </Destination>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <ComType>Unicast</ComType>
  <SessionID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</SessionID>
  <MsgBody protocol=" Namespace of PUCC Basic Communication Protocol" >
    <Hello xmlns=" Namespace of PUCC Basic Communication Protocol" >
      <InitiatorUserID>urn:pucc:user:A</InitiatorUserID>
```

```
      <ResponderUserID>urn:pucc:user:B</ResponderUserID>
      <RequestedCapability>
        <ExtendedProtocol ID=" 1" >MulticastCommunication</ExtendedProtocol>
      </RequestedCapability>
    </Hello>
  </MsgBody>
</Core>
```

Figure 15. **Sample of Hello message with session ID and user ID used to establish a multihop session**

6.2.1.1.1. Hello element

The Hello element is used to initialize a PUCC connection from one node to another node and/or a PUCC session from one user to another user on the PUCC network. Four elements are defined as fields (Parameters) in the Hello message.

6.2.1.1.2. Mode element

The Mode element designates the operation mode of behavior of the node. Every node operates in one of the following modes. (For more information, read PUCC Architecture Specification.) This element is optional. If this element is omitted, the Mode element value is “pure”.

Mode	Description
Pure	Operation of a normal node without control node management.
Hybrid	Operation of a normal node with control node management.
Gateway	Operation of a gateway node.
ControlNode	Operation of a control node

Table 8: Mode of PUCC communication

6.2.1.1.3. InitiatorUserID element

The InitiatorUserID element sets the user ID of the session setting up side when per user session setup is performed between the PUCC nodes. This element is optional. If this element is omitted, that indicates that a user does not exist on the node. When this element is set, the Session ID element must be set in the Core element.

6.2.1.1.4. ResponderUserID element

The ResponderUserID element is used to set the use ID of the session setting up side when per user session setup is performed between the PUCC nodes. This element is optional. If this element is omitted, that indicates that a user does not exist on the node. When this element is set, the Session ID element must be set in the Core element.

6.2.1.1.5. Authentication element

The Authentication element designates that the user authentication is required. This element is optional. If this element is omitted, user authentication is not executed.

For the first Hello message, the Authentication element has no content. For the second Hello message, the Authentication element has Response element, Challenge element and HashAlgorithm element. The Response element designates the response code. The Challenge element designates the challenge code.

6.2.1.1.6. RequestedSecurity element

The RequestedSecurity element sets the value of the parameter required for PUCC session message encryption and message authentication. This parameter is optional.

6.2.1.1.7. RequestedCapability element

The RequestedCapability element is used to negotiate node capability. A node sends a Hello message with capability requirement using the RequestedCapability element. In the present situation, we use it to negotiate the installed protocols such as PUCC Multicast Communication Protocol. When the mode of the node is Pure, the PUCC Basic Communication Protocol （"BasicCommunication”） and PUCC Control Message Protocol （"ControlMessage”） specified by the PUCC Basic Protocol - light profile shall be out of the scope of negotiation. When the mode of the node is other than Pure, the PUCC Basic Communication Protocol"BasicCommunication”） and PUCC Control Message Protocol（"ControlMessage”） specified by the PUCC Basic Protocol - light profile, and the PUCC Basic Service Protocol（"BasicService”）specified in this specification shall be out of the scope of negotiation. The RequestedCapability element has an ExtendedProtocol element. The ExtendedProtocol element designates the name of the protocol desired by the node. This element can be extended in the future. The following show values of ExtendedProtocol element and an example of RequestedCapability element.

Element name	Element Value	Attribute name (if it has)	Attribute Value	Occurrence	Status	Description
RequestedCapability	XML fragment	-	-	1	Optional	
ExtendedProtocol	“MulticastCommunication”,”MulticastService”	-	-	Multiple	Required	
		ID	Number(Integer)	1	Required	

Table 9: RequestedCapability element

```
<RequestedCapability>
  <ExtendedProtocol ID=” 1” >MulticastCommunication</ExtendedProtocol>
  <ExtendedProtocol ID=” 2” >MulticastService</ExtendedProtocol>
</RequestedCapability>
```

Figure 16. Example of RequestedCapability element

6.2.1.2. HelloResponse message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
HelloResponse	Following XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
Mode	“Hybrid”,“Gateway” ,“ControlNode”	-	-	1	Optional	Default =”Pure”
Result	“Success”,”Failure”, "Continue"	-	-	1	Required	
Reason	String	-	-	1	Depend on result	
Authentication	for the first HelloResponse HashAlgorithm, Challenge element	-	-	1	Optional	

		for the second HelloResponse Response element					
	Challenge	Challenge Code (hexBinary)	-	-	1	Optional	for the first time only
	Requested HashAlgorithm	)Hash Algorithm that makes a request to the response of the Response Code (String)	-	-	1	Optional	for the first time only
	Response	Response Code (hexBinary)	-	-	1	Optional	for the second time only
NegotiatedSecurity		for the second HelloResponse CipherAlgorithm, HashAlgorithm, IterationCount, UpdateCount element.	-	-	1	Optional	for the second time only
	CipherAlgorithm	Cipher Algorithm	-	-	1	Optional	for the second time only
	HashAlgorithm	Hash Algorithm	-	-	1	Optional	for the second time only
	IterationCount	Hash repeat counts	-	-	1	Optional	for the second time only
	UpdateCount	Number of session key updates	-	-	1	Optional	for the second time only
NegotiatedCapability		XML fragment	-	-	1	Optional	
	ExtendedProtocol	"MulticastCommunication","MulticastService"	-	-	Multiple	Required	
			ID	Number(Integer)	1	Required	

Table 10: Fields of HelloResponse

The HelloResponse message is mapped to the PUCC Core Protocol as follows.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast"	-	-	1	Required	
MsgID	Message ID (String)	-	-	1	Required	
ReplyID	MsgID (String)	-	-	1	Required	
MsgType	"Response"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Route,Target element	-	-	1	Required	
Route	Route,Target element	-	-	1	Optional	
		Node	Node ID(String)	1	Required	
Target	Node ID(String)	-	-	1	Required	
TraceRoute	Route element	-	-	1	Optional	
Route	Route element		-	1	Optional	
		Node	Node ID(String)	1	Required	
HopCount	Hop count (Integer)	-	-	-	unused	
GatewayAction	"Hook","Hop"	-	-	-	unused	

	SessionID	Session ID(String)	-	-	1	Optional	
	MsgBody	XML fragment	-	-	1	Required	
			Protocol	URI	1	Required	

Table 11: Mapping of HelloResponse message to Core Protocol

The following is a sample of HelloResponse message.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <MsgType>Response</MsgType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <ReplyID>54321.2002-12-20T16:17:32Z@968749ab-f9bb-2647-7459-f66e56f11234</ReplyID>
  <Destination>
    <Target>968749ab-f9bb-2647-7459-f66e56f11234</Target>
  </Destination>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <ComType>Unicast</ComType>
  <MsgBody protocol=" Namespace of PUCC Basic Communication Protocol" >
    <HelloResponse xmlns=" Namespace of PUCC Basic Communication Protocol" >
      <Result>Success</Result>
      <NegotiatedCapability>
        <ExtendedProtocol ID=" 1" >MulticastCommunication</ExtendedProtocol>
      </NegotiatedCapability>
    </HelloResponse>
  </MsgBody>
</Core>
```

Figure 17. Sample of HelloResponse message

The following is a sample of HelloResponse message used to establish a multihop session.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <MsgType>Response</MsgType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <ReplyID>54321.2002-12-20T16:17:32Z@968749ab-f9bb-2647-7459-f66e56f11234</ReplyID>
  <Destination>
    <Route Node="874542ab-a5c6-4305-8745-ababcdcdefef">
      <Target>968749ab-f9bb-2647-7459-f66e56f11234</Target>
    </Route>
  </Destination>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <ComType>Unicast</ComType>
  <SessionID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</SessionID>
  <MsgBody protocol=" Namespace of PUCC Basic Communication Protocol" >
    <HelloResponse xmlns=" Namespace of PUCC Basic Communication Protocol" >
      <Result>Success</Result>
      <NegotiatedCapability>
        <ExtendedProtocol ID=" 1" >MulticastCommunication</ExtendedProtocol>
      </NegotiatedCapability>
    </HelloResponse>
  </MsgBody>
</Core>
```

Figure 18. Sample of HelloResponse message with Session ID used to establish a multihop session

6.2.1.2.1. HelloResponse element

This element is used to designate the response to the hello message. The content of this element describes the ”Success” or “Failure” response code and diagnostic description. If a node has no capability to connect to other nodes, the node sends HelloResponse message with “Failure”.

6.2.1.2.2. Mode element

The Mode element designates the operation mode of behavior of the node. Every node operates in one of the following modes. (For more information, read PUCC Architecture Specification.) This element is optional. If this element is omitted, the Mode element value is “Pure”.

6.2.1.2.3. Result element

The Result element designates the “Success” or “Failure” or “Continue” of a PUCC connection or a PUCC session. This element is essential.

6.2.1.2.4. Reason element

The Reason element is used to notify about reason of failure. This element takes following values. If the Result element value is “Failure”, Reason element is required. Don’t set this element, if it is not so.

Value	Description
NotAuthenticated	Authentication failure
ConnectionCapabilityExceeded	The node has no capability to connect to other nodes.
IncompatibleMode	The node acts according to the incompatible operation mode of behavior.
UserAccessDenied	User access was denied because authentication failed
SessionAlreadyEstablished	Session has already been established between the same users

Table 12: Values of Reason element

6.2.1.2.5. Authentication element

The Authentication element designates that the user authentication is required. This element is optional. If this element is omitted, user authentication is not executed.

For the first HelloResponse message, the Authentication element has a Challenge element and a RequestedHashAlgorithm element. The Challenge element designates the challenge code. For the second Hello message, the Authentication element has a Response element. The Response element designates the response code.

6.2.1.2.6. NegotiatedSecurity element

The NegotioatedSecurity element is used to set the determined value for the parameter required for message encryption and message authentication of the PUCC session. This parameter is optional.

6.2.1.2.7. NegotiatedCapability element

The NegotiatedCapability element is used to negotiate node capability. A node sends a Hello message with capability requirement using the RequestedCapability element to other node. The other node sends HelloResponse message with its own capability using the NegotiatedCapability element. In the present situation, we use it to negotiate installed protocols such as PUCC Multicast Communication Protocol. In case the mode of the node is Pure, the PUCC Basic Communication Protocol （"BasicCommunication”） and PUCC Control Message Protocol

（"ControlMessage”）specified by the PUCC Basic Protocol - light profile are outside of the scope of negotiation. In case the mode of the node is Pure, the PUCC Basic Communication Protocol"BasicCommunication”）and PUCC Control Message Protocol（"ControlMessage”）specified by the PUCC Basic Protocol - light profile, and the PUCC Basic Service Protocol（"BasicService”）specified by this specification are outside of the scope of negotiation. The NegotiatedCapability element has an ExtendedProtocol element. The ExtendedProtocol element equals one of the RequestedCapability elements. This element can be extended in the future.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
NegotiatedCapability	XML fragment	-	-	1	Optional	
ExtendedProtocol	“MulticastCommunication”,”MulticastService”	-	-	Multiple	Required	
		ID	number (Integer)	1	Required	

Table 13: NegotiatedCapability element

```
<NegotiatedCapability>
  <ExtendedProtocol ID=” 1” >MulticastCommunication</ExtendedProtocol>
  <ExtendedProtocol ID=” 2” >MulticastService</ExtendedProtocol>
</NegotiatedCapability>
```

Figure 19. Example of NegotiatedCapability element

6.2.1.3. Sequence of Hello method

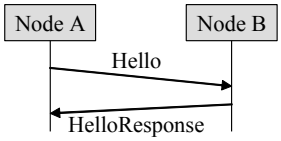


Figure 20. Sequence of Hello method

A relay node in a multihop session does not necessarily need to have a session with the nodes at both ends between which the session is established. The relay node forwards a message from a node which is not performing Hello.

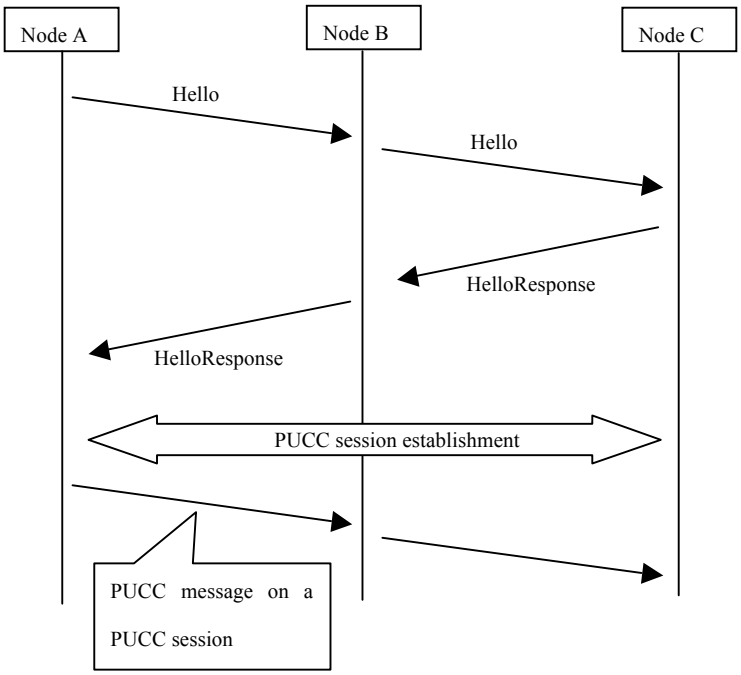


Figure 21. Sequence of Hello method used to establish a multihop session

Even when multiple multihop sessions are established between the same nodes, they are considered different ones if they go through different relay nodes.

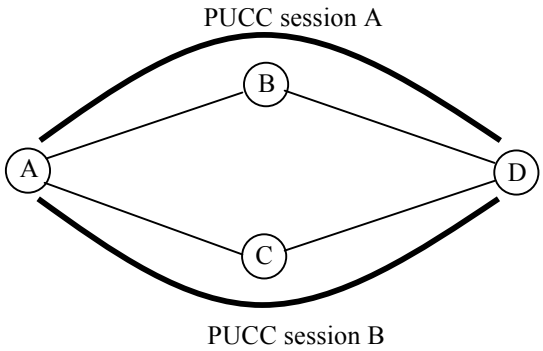


Figure 22. Multihop session between Node A and Node B with different relay nodes

The PUCC connection and the PUCC session are independent from each other.

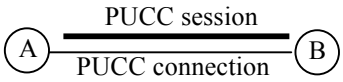


Figure 23. Sample of PUCC connection and PUCC Session

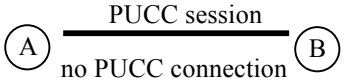


Figure 24. Sample of PUCC connection and PUCC session

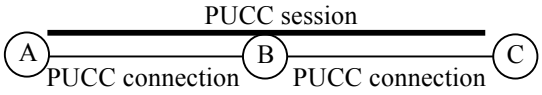


Figure 25. Sample of PUCC connection and multihop session

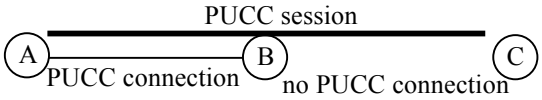


Figure 26. Sample of PUCC connection and multihop session

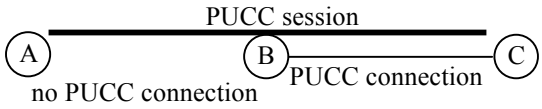


Figure 27. Sample of PUCC connection and multihop session

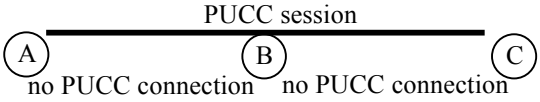


Figure 28. Sample of PUCC connection and multihop session

6.2.2. Bye method

6.2.2.1. Bye message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Bye	-	-	-	1	Required	
		xmlns	URI	1	Required	

Table 14: Fields of Bye message

The Bye message is mapped as per the following table.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Core	Following XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	-	unused	
MsgType	"Advertise"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Route,Target element	-	-	1	Required	
Route	Route,Target element	-	-	1	Optional	
		Node	Node ID(String)	1	Required	
Target	Node ID(String)	-	-	1	Required	
TraceRoute	Route element	-	-	1	Optional	
Route	Route element	-	-	1	Optional	
		Node	Node ID(String)	1	Required	
HopCount	Hop count (Integer)	-	-	-	unused	
GatewayAction	"Hook","Hop"	-	-	-	unused	
SessionID	Session ID(String)	-	-	1	Optional	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI		Required	

Table 15: Mapping of Bye message to PUCC Core Protocol

The following is a sample of Bye message.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <MsgType>Advertise</MsgType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <Destination>
    <Target>457284ab-f9bb-4305-9900-f98e56f12355</Target>
  </Destination>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <ComType>Unicast</ComType>
  <MsgBody protocol=" Namespace of PUCC Basic Communication Protocol" >
    <Bye xmlns=" Namespace of PUCC Basic Communication Protocol" />
  </MsgBody>
</Core>
```

Figure 29. Sample of Bye message

The following is a sample of Bye message used to release a multichip session

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <MsgType>Advertise</MsgType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <Destination>
    <Route Node="874542ab-a5c6-4305-8745-ababcdcdefef">
      <Target>457284ab-f9bb-4305-9900-f98e56f12355</Target>
    </Route>
  </Destination>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <ComType>Unicast</ComType>
  <SessionID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</SessionID>
  <MsgBody protocol=" Namespace of PUCC Basic Communication Protocol" >
    <Bye xmlns=" Namespace of PUCC Basic Communication Protocol" />
  </MsgBody>
</Core>
```

Figure 30. Sample of Bye message used to release a multihop session

6.2.2.1.1. Bye element

The Bye element is used to release a PUCC connection from one node to another node or a PUCC session from one user to another user on the PUCC network. This element has no content.

6.2.2.2. Sequence of Bye method

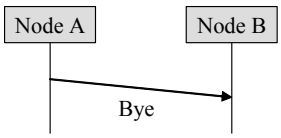


Figure 31. Sequence of Bye method

The multichip session is released when transmission of a message fails after the relay node exits.

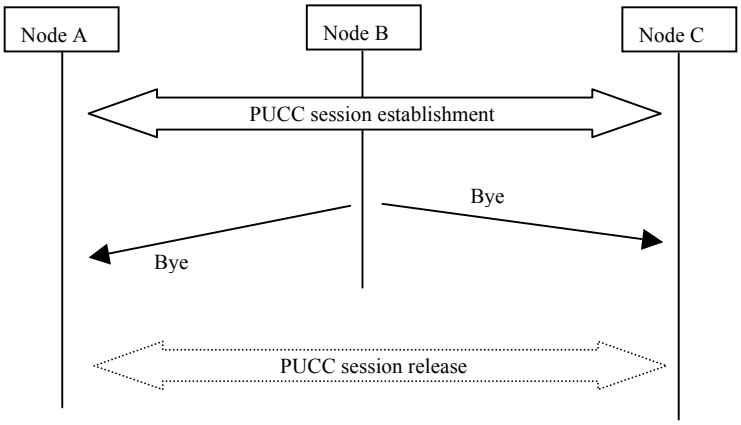


Figure 32. Release of multichip session due to the exit of relay node

6.2.3. Resource Information Exchange method

6.2.3.1. ResourceInfomationRequest message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
ResourceInformationRequest	-	-	-	1	Required	
		xmlns	URI	1	Required	

Table 16: Fields of ResourceInformationRequest message

The ResourceInformationRequest message is mapped following table.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Core	Following XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast","Broadcast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	-	unused	
MsgType	"Request"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Route,Target element	-	-	1	ComType Dependent	
Route	Route,Target element	-	-	1	Optional	
		Node	Node ID(String)	1	Required	
Target	Node ID (String)	-	-	1	Required	
TraceRoute	XML fragment	-	-	1	Required	
HopCount	Hop count(Integer)	-	-	1	Optional	
GatewayAction	"Hook","Hop"	-	-	1	Optional	
SessionID	Session ID(String)	-	-	1	Optional	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI	1	Required	

Table 17: Mapping of ResourceInformationRequest to PUCC Core Protocol

The following is a sample of ResourceInformationRequest message.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Broadcast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <MsgType>Request</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f1235</Source>
  <HopCount>10</HopCount>
  <MsgBody protocol=" Namespace of PUCC Basic Communication Protocol" >
    <ResourceInformationRequest xmlns=" Namespace of PUCC Basic Communication Protocol" />
  </MsgBody>
</Core>
```

Figure 33. Sample of ResouceInformationRequest

6.2.3.1.1 ResourceInfomationRequest element

The ResourceInformationRequest element is used to request the resource information of a node. This element has no content..

6.2.3.1. ResourceInfomationResponse message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
ResourceInformationResponse	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ResourceData	XML fragment	-	-	1	Required	

Table 18:: Fields of ResourceInformationResponse

The ResourceInformationResponse message is mapped following table.

Element name	Element Value	Attribute name (if it has)	Attribute Value	Occurrence	Status	Description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	1	Required	
MsgType	"Response"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Route,Target element	-	-	1	Required	
Route	Route,Target element	-	-	1	Optional	
		Node	Node ID(String)	1	Required	
Target	Node ID, Multicast Group ID(String)	-	-	1	Required	
TraceRoute	XML fragment	-	-	1	Required	
Route	-	-	-	1	Optional	
		Node	Node ID(String)	1	Required	
HopCount	Hop count (Integer)	-	-	-	unused	
GatewayAction	"Hook","Hop"	-	-	-	unused	
SessionID	Session ID(String)	-	-	1	Optional	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI	1	Required	

Table 19: Mapping of ResourceInformationResponse message to PUCC Core Protocol

The following is a sample of ResourceInformationResponse message.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <ReplyID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687</ReplyID>
  <MsgType>Response</MsgType>
  <Source>nd2</Source>
  <Destination>
    <Target>nd1</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Basic Communication Protocol" >
    <ResourceInformationResponse xmlns=" Namespace of PUCC Basic Communication Protocol" >
      <ResourceData>
```

```
      (For details see 6.2.4 Resource Information.)

      </ ResourceData>
    </ResourceInformationResponse>
  </MsgBody>
</Core>
```

Figure 34. Samples of ResourceInformationResponse

6.2.3.1.2. ResourceInformationResponse element

This element designates the response to the ResourceInformationRequest message. The content of this element describes the resource information on the requested node.

6.2.3.1.3. ResourceData element

The ResourceData element is used to describe the resource information of a node. This element has 9 elements. (For details see 6.2.4 Resource Information.)

6.2.3.2. ResourceInformationAdvertise message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
ResourceInformationAdvertise	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ResourceData	XML fragment	-	-	Multiple	Required	

Table 20: Fields of ResourceInformationAdvertise

The ResourceInformationAdvertise message is mapped as following table.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast","Broadcast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	-	unused	
MsgType	"Advertise"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Route, Target element	-	-	1	ComType Dependent	
Route	Route , Target Element	-	-	1	Optional	
		Node	Node ID(String)	1	Required	
Target	Node ID (String)	-	-	1	Required	
TraceRoute	Route element	-	-	1	Required	
Route	Route element	-	-	1	Optional	
		Node	Node ID(String)	1	Required	
HopCount	Hop Count(Integer)	-	-	1	Optional	
GatewayAction	"Hook","Hop"	-	-	1	Optional	
SessionID	Session ID(String)	-	-	1	Optional	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI	1	Required	

Table 21: Mapping of ResourceInformationAdvertise to PUCC Core Protocol

The following is a sample of ResourceInformationAdvertise message.

```
<Core xmlns= “Namespace of PUCC Core Protocol “>
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687</MsgID>
  <MsgType>Advertise</MsgType>
  <Source>968985ab-e6aa-5842-1234-f98e56f15687</Source>
  <Destination>654742bb-11aa-2586-1123-b00e56f58974</Destination>
  <MsgBody protocol= “Namespace of PUCC Basic Communication Protocol “>
    <ResourceInformationAdvertise xmlns= “Namespace of PUCC Basic Communication Protocol “>
      <ResourceData>

      (For details see 6.2.4 Resource Information.)

      </ ResourceData>
    </ResourceInformationAdvertise>
  </MsgBody>
</Core>
```

Figure 35. Sample of ResourceInformationAdvertise

6.2.3.3.1. ResourceInformationAdvertise element

This element designates the advertise message sent to other nodes. The contents of this element describe the resource information on the node.

6.2.3.3.2. ResourceData element

The ResourceData element is used to describe resource information of a node. This element has 9 elements. (For details see 6.2.4 Resource Information.). In the case, Gateway notifies ControlNode the topology of PUCC network, occurrence number is arbitrary number of 0 or more. Occurrence number is 1 in other cases.

6.2.3.4. Sequence of Resource exchange method

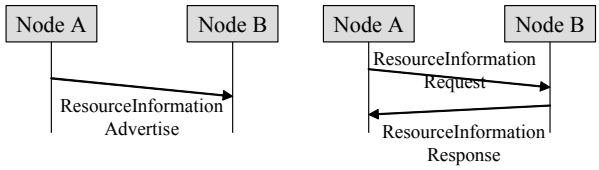


Figure 36. Sequence of Resource exchange method

6.2.4. Resource Information

6.2.4.3. Items of Resource Information

Information of each node in PUCC network (Network Metadata) is described using XML. The node requests another node’s information, and the response is made using XML. The above process flow is named “Resource Information Exchange”. The following is a definition of the fields in Resource Information.

Items	Description
OwnNodeID	Own Node ID
Mode	Operation Mode (Hybrid, Pure, Gateway, ControlNode) In the case of Gateway, the mode setup value shall be changed according to the mode of the node to which the ResourceInformation is conveied.

	• Set “Gateway” when the mode of the conveid to node is ControlNode • Set “Hybrid” when the mode of the conveid to node is Hybrid。 • Set “Pure” when the mode of the conveid to node is Pure
OwnedApplication	The Functions (application) provided by the node (e.g.: search, instant message)
ConnectionCapability	The number of the maximum connection.
ConnectedNode	The Node ID of the connecting nodes. (e.g.: node B, node C)
JoiningGroup	The Group ID of the joining multicast group.
RoutingGroup	The Group ID of the routing multicast group.
MulticastRouting	Adjacent node list of a multicast
TransportAddress@protocol	A transport layer protocol of a node.
TransportAddress@type	A transport layer address of a node.

Table 22: Items of Resource Information

6.2.3.1. Schema Definition

The following is a definition of the fields (Parameter) in Resource Information.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
ResourceData	XML Fragment	-	-	Multiple	Required	
OwnNodeID	Node ID	-	-	1	Required	
Mode	Hybrid, Pure, Gateway, ControlNode	-	-	1	Optional	Default =”Pure”
OwnedApplication	Application element	-	-	1	Optional	
Application	URI (String)	-	-	Multiple	Required	
ConnectionCapability	Maximum number of connection	-	-	1	Required	
ConnectedNode	XML fragment	-	-	1	Required	
Node	TransportAddress element, ConnectionCapability element, RTT element	-	-	Multiple	Optional	
		ID	Node ID(String)	1	Required	
		parent	“true”, “false”	1	Required	
Transport Address	Transport layer address (String)	-	-	Multiple	Required	
		protocol	classification of protocol(String) such as ”TCP”, ”SOAP”, ”HTTP”, ”BEEP”, ”Bluetooth” , “OBEX”, “IEEE1394”	1	Required	
		type	classification of address(String) such as ”IPv4”, ”IPv6”, ”URL”,”Bluetooth”, “IEEE1394”	1	Required	
ConnectionCapability	Maximum number of connection (Integer)	-	-	1	Optional	
RTT	RTT by Ping. Time scale is millisecond. (Integer)	-	-	1	Optional	
JoiningGroup	XML fragment	-	-	1	Optional	
GroupID	Multicast Group ID(String)	-	-	Multiple	Optional	
RoutingGroup	XML fragment	-	-	1	Optional	
GroupID	Multicast Group ID(String)	-	-	Multiple	Optional	
MulticastRouting	XML fragment	-	-	1	Optional	
Group	NodeID element	-	-	Multiple	Optional	

				ID	Multicast Group ID (String)	1	Required	
				joinDirection	Node ID(String)	1	Required	
		NodeID	Node ID (String)	-	-	Multiple	Optional	
	TransportAddress		Transport layer address (String)	-	-	Multiple	Required	
				protocol	classification of protocol(String) such as "TCP", "SOAP", "HTTP", "BEEP", "Bluetooth" , "OBEX", "IEEE1394"	1	Required	
				type	classification of address(String) such as "IPv4", "IPv6", "URL","Bluetooth", "IEEE1394"	1	Required	

Table 23: Structure of ResourceData Element

6.2.3.1.1. ResourceData element

The ResourceData element is used to describe resource information of a node. This element has 9 elements.

6.2.3.1.2. OwnNodeID element

The OwnNodeID element is used to designate node ID.

6.2.3.1.3. Mode element

The Mode element is used to designate operation mode such as hybrid, pure, gateway and control node. If Mode element is empty or not specified, it is regarded as the "Pure".

6.2.3.1.4. OwnedApplication element

The OwnedApplication element is used to describe names of applications running in a node. This element has multiple Application elements according to applications running in the node. The content of the Application element is an application identifier described using URI. This element is optional.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
OwnedApplication	XML fragment	-	-	1	Optional	
Application	URI (String)	-	-	Multiple	Required	

Table 24: The OwnedApplication element

6.2.3.1.5. ConnectionCapability element

The ConnectionCapability element is used to designate the maximum number of connections. A node can specify a number of connections using this element. This element is essential.

6.2.3.1.6. ConnectedNode element

The ConnectedNode element is used to describe a list of nodes connected to a node. This element is essential. This element has Node elements which have TransportAddress elements ConnectionCapability elements, RTT elements and ID attribute.

The ConnectionCapability element is used to designate the maximum number of connections of the connected nodes.

The RTT element is used to designate the round trip time on connections with the connected nodes.

The TransportAddress element has two attribute: protocol and type. The protocol attribute is used to designate the class of the transport protocol used such as TCP, and SOAP. The type element is used to designate the class of transport address used such as IPv4 and URL.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
ConnectedNode	XML fragment	-	-	1	Required	
Node	TransportAddrtess element, ConnectionCapability element, RTT element	-	-	Multiple	Optional	
		ID	Node ID (String)	1	Required	
		parent	“true”,”false”	1	Required	Default value is “false”.
Transport Address	Transport layer address (String)	-	-	Multiple	Required	
		protocol	classification of protocol(String) such as ”TCP”, ”SOAP”, ”HTTP”, ”BEEP”, ”Bluetooth” , “OBEX”, “IEEE1394”	1	Required	
		type	classification of address(String) such as ”IPv4”, ”IPv6”, ”URL”,”Bluetooth”, “IEEE1394”	1	Required	
Connection Capability	Maximum number of connection (Integer)	-	-	1	Optional	
RTT	RTT by Ping. Time scale is millisecond. (Integer)	-	-	1	Optional	

Table 25: The ConnectedNode element

6.2.3.1.7. JoiningGroup element

The JoiningGroup element is used to designate the multicast group IDs of the multicast groups to which the node is joined. This element has multiple GroupID elements. A GroupID element describes the group ID of a multicast group to which the node is joined. This element is optional. If this element is empty or not specified, PUCC Multicast Communication Protocol is not supported. If GroupID is not specified, the node does not join any multicast group.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
JoiningGroup	XML fragment	-	-	1	Optional	
GroupID	Multicast Group ID(String)	-	-	Multiple	Optional	

Table 26: The JoiningGroup element

6.2.3.1.8. RoutingGroup element

The RoutingGroup element is used to designate the multicast group IDs of the multicast groups which the node participates in routing for. This element has multiple GroupID elements. A GroupID element describes the group ID of a multicast group which the node participates in routing for. This element is optional. If this element is empty or not specified, PUCC Multicast Communication Protocol is not supported. If GroupID is not specified, the node does not participate in any multicast routing.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
RoutingGroup	XML fragment	-	-	1	Optional	
GroupID	Multicast Group ID(String)	-	-	Multiple	Optional	

Table 27: The RoutingGroup element

6.2.3.1.9. MulticastRouting element

The MulticastRouting element is used to describe a list of nodes that a correspondent node has to send a multicast message to. This element has multiple Group elements. A Group element has ID attribute which describes a multicast group ID, and joinDirection attribute which is used for reconstruction of multicast tree. A Group element has NodeID element which designate routing nodes. This element is optional. If this element is empty or not specified, PUCC Multicast Communication Protocol is not supported. If Group is not specified, the node does not join any multicast group and participate in any multicast routing.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
MulticastRouting	XML fragment	-	-	1	Optional	
Group	XML fragment	-	-	Multiple	Optional	
		ID	Multicast Group ID(String)	1	Required	
		joinDirection	Node ID (String)	Multiple	Required	
NodeID	Node ID (String)	-	-	Multiple	Optional	

Table 28: The MulticastRouting element

6.2.3.1.10. TransportAddress element

The TransportAddress element is used to designate the transport layer addresses of the node. This element has two attributes: protocol and type. The protocol attribute is used to designate the class of the transport protocol used such as TCP, and SOAP. The type element is used to designate the class of transport address used such as IPv4 and URL. The contents of the TransportAddress element are the transport addresses of the node. This element is essential.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
TransportAddress	Transport layer	-	-	Multiple	Required	

	address (String)	protocol	classification of protocol(String) such as ”TCP”, ”SOAP”, ”HTTP”, ”BEEP”, ”Bluetooth” , “OBEX”, “IEEE1394”	1	Required	
		type	classification of address(String) such as ”IPv4”, ”IPv6”, ”URL”,”Bluetooth”, “IEEE1394”	1	Required	

Table 29: The TransportAddress element

```
<ResourceData>
  <OwnNodeID>968742ab-f9bb-4305-9900-f98e56f12352</OwnNodeID>
  <Mode>Pure</Mode>
  <OwnedApplication>
    <Application>http://www.xxx.co.jp/PUCC_search_app</Application>
    <Application>http://www.xxx.co.jp/PUCC_instantmsg_app</Application>
  </OwnedApplication>
  <ConnectionCapability>5</ConnectionCapability>
  <ConnectedNode>
    <Node ID=” 145742a8-63cc-7569-5468-f98e55f12478” >
      <TransportAddress protocol=” TCP”  type=” IPv4” >10.74.126.155<TransportAddress>
    </Node>
    <Node ID= “654742bb-11aa-2586-1123-b00e56f58974 “>
      <TransportAddress protocol=” TCP”  type=” IPv4” >10.74.126.156<TransportAddress>
    </Node>
  </ConnectedNode>
  <JoiningGroup>
    <GroupID>urn:pucc:multicastgroup:DoCoMo</GroupID>
  </JoiningGroup>
  <RoutingGroup>
    <GroupID>urn:pucc:multicastgroup:XXX</GroupID>
  </RoutingGroup>
  <MulticastRouting>
    <Group ID=” urn:pucc:multicastgroup:DoCoMo”  joinDirection=” 654742bb-11aa-2586-1123-b00e56f58974” >
      <NodeID>654742bb-11aa-2586-1123-b00e56f58974</NodeID>
    </Group>
    <Group ID=” urn:pucc:multicastgroup:XXX”  joinDirection=” 654742bb-11aa-2586-1123-b00e56f58972” >
      <NodeID>654742bb-11aa-2586-1123-b00e56f58979</NodeID>
    </Group>
  </MulticastRouting>
  <TransportAddress protocol=” TCP”  type=IPv4” >192.168.0.56</TransportAddress>
</ ResourceData>
```

Figure 37. A sample of ResourceData

The DTD of node resource information is described in APPENDIX E.

6.3. PUCC Basic Service Protocol

The PUCC Basic Service Protocol defines the communication between a node and a control node. The node communicates with the control node using this protocol. The PUCC Basic Service Protocol includes:

- Service Provide method

The Service Provide method is used to receive services from a control node such as routing information offering service, name resolution service and so on. This method consists of a ServiceRequest message, a ServiceResponse message, and ServiceAdvertise message.

6.3.1. Service Provide method

6.3.1.1. ServiceRequest message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
ServiceRequest	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
Service	Source element, Destination element, NodeID element	-	-	1	Required	
		type	Type of Service	1	Required	
Source	Node ID(String)	-	-	1	Type attribute dependent	
Destination	Node ID(String)	-	-	1	Type attribute dependent	
NodeID	Node ID (String)	-	-	1	Type attribute dependent	
UserID	User ID (String)	-	-	1	Type attribute dependent	

Table 30: Fields of ServiceRequest message

The ServiceRequest message is mapped as following table.

Element name	Element Value	Attribute name (if it has)	Attribute Value	Occurrence	Status	Description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	-	unused	
MsgType	"Request"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Target element	-	-	1	Required	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
Target	Node ID(String)	-	-	1	Required	
TraceRoute	Route element	-	-	-	unused	
Route	Route element	-	-	-	unused	
		Node	Node ID(String)	-	unused	
HopCount	Hop count (Integer)	-	-	-	unused	
GatewayAction	"Hook","Hop"	-	-	-	unused	
SessionID	Session ID(String)	-	-	-	unused	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI	1	Required	

Table 31: Mapping of ServiceRequest message to PUCC Core Protocol

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <MsgType>Request</MsgType>
  <Source>nd1</Source>
  <Destination>
    <Target>nd2</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Basic Service Protocol" >
    <ServiceRequest xmlns=" Namespace of PUCC Basic Service Protocol" >
      <Service type=" Route" >
        <NodeID>975870cd-f9bb-4305-1235-e65e29f6029</NodeID>
      </Service>
    </ServiceRequest>
  </MsgBody>
</Core>
```

Figure 38. A sample of ServiceRequest message

6.3.1.1.1. ServiceRequest element

The ServiceRequest element is used to designate a control node service. This element has type attribute which designates the type of service such as providing an adjacent node and providing route information to destination node.

6.3.1.1.2. Service element

The Service element has four elements: Source, Destination, NodeID and UserID. When value of the type attribute is “AdjacentNode”, the Service element has no element. When value of the type attribute is “Route”, the Service element has NodeID element. When value of the type attribute is “NameResolution”, the Service element has NodeID element or UserID element. When value of the type element is “Cost”, the Service element has the Source element and the Destination element. (For details see 6.3.2 Service Type)

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Service	XML fragment	-	-	1	Required	
		type	“AdjacentNode”,” Route”,”NameResolution”,”Cost”	1	Required	
Source	Node ID (String)	-	-	1	Type attribute dependent	
Destination	Node ID (String)	-	-	1	Type attribute dependent	
NodeID	Node ID (String)	-	-	1	Type attribute dependent	
UserID	User ID (String)	-	-	1	Type attribute dependent	

Table 32: Content of Service element

type attribute	AdjacentNode	Route	NameResolution	Cost
Source	x	x	x	o
Destination	x	x	x	o

NodeID	x	o	o	x
UserID	x	x	o	x

Table 33: usage of the elements

6.3.1.2. ServiceResponse message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
ServiceResponse	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
Service	XML fragment or "Failure"	-	-	1	Required	
		type	Type of Service	1	Required	

Table 34: Fields of ServiceResponse message

The ServiceRequest message is mapped as following table.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	1	Required	
MsgType	"Response"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Target element	-	-	1	Required	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
Target	Node ID(String)	-	-	1	Required	
TraceRoute	Route element	-	-	1	unused	
Route	Route element	-	-	1	unused	
		Node	Node ID(String)	1	unused	
HopCount	Hop count (Integer)	-	-	-	unused	
GatewayAction	"Hook","Hop"	-	-	-	unused	
SessionID	Session ID(String)	-	-	-	unused	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI	1	Required	

Table 35: Mapping of ServiceResponse message to PUCC Core Protocol

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <ReplyID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687</ReplyID>
  <MsgType>Response</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>968985ab-e6aa-5842-1234-f98e56f15687</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Basic Service Protocol" >
    <ServiceResponse xmlns=" Namespace of PUCC Basic Service Protocol" >
      <Service type=" AdjacentNode" >
```

```
        <Node ID=" 975870cd-f9bb-4305-1235-e65e29f6029" >
          <TransportAddress protocol=" TCP"  type=" IPv4" >1074.126.155</TransportAddress>
        </Node>
      </Service>
    </ServiceResponse>
  </MsgBody>
</Core>
```

Figure 39. A sample of ServiceResponse message

6.3.1.2.1. ServiceResponse element

The ServiceResponse element is used to provide control node services for PUCC nodes. This element has type attribute which designates the type of service such as providing an adjacent node and providing route information to destination node.

6.3.1.2.2. Service element

This element has three types: Node, Destination, and HopCount element. When the value of the type attribute is "AdjacentNode", the Service element has the Destination element. When the value of the type attribute is "Route" or "NameResolution", the Service element has the Node element. When the value of the type element is "Cost", the Service element has the HopCount element. (For details see 6.3.2 Service Type)

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Service	XML fragment or "Failure"	-	-	1	Required	
		type	"AdjacentNode","Route","NameResolution","Cost"	1	Required	
Node	TransportAddress element	-	-	Multiple	Type attribute dependent	
		ID	Node ID(String)	1	required	
Transport Address	Transport layer address(String)	-	-	1	Required	
		protocol	classification of protocol(String) such as "TCP", "SOAP", "HTTP", "BEEP", "Bluetooth" , "OBEX", "IEEE1394"	1	Required	
		type	classification of address(String) such as "IPv4", "IPv6", "URL","Bluetooth", "IEEE1394"	1	Required	
Destination	Route, Target element	-	-	1	Type attribute dependent	
Route	Route, Target element	-	-	Multiple	Optional	
		Node	Node ID(String)	1	Required	
Target	Node ID(String)	-	-	Multiple	Required	
HopCount	Hop count (Integer)	-	-	1	Type attribute dependent	

Table 36: Content of Service element

A) Node element

The Node element is used when the type attribute of Service element is "AdjacentNode" or "NameResolution". The Node element has multiple TransportAddress elements which describe the node's transport layer addresses. The TransportAddress element has protocol attribute and type attribute. The protocol attribute designates the classification of the transport protocol used such as TCP, SOAP, and HTTP. The type attribute designates the classification of transport address used such as IPv4, IPv6, and URL.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Node	TransportAddress element	-	-	1	Required	
		ID	Node ID(String)	1	required	
Transport Address	Transport layer address (String)	-	-	1	Required	
		protocol	classification of protocol(String) such as "TCP", "SOAP", "HTTP", "BEEP", "Bluetooth", "OBEX", "IEEE1394"	1	Required	
		type	classification of address(String) such as "IPv4", "IPv6", "URL","Bluetooth", "IEEE1394"	1	Required	

Table 37: Content of Node element

B) Destination element

The Destination element is used when the type attribute of Service element is "Route". The structure of this element is the same as that of the Destination element of PUCC Core Protocol. (For details see 6.1.8 Destination element)

C) HopCount element

The HopCount element is used when the type attribute of Service element is "Cost". This element designates the cost between source and destination. This element is the same as HopCount element of PUCC Core Protocol. (For details see 6.1.10 HopCount element)

6.3.1.3. ServiceAdvertise message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
ServiceAdvertise	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
Service	ConnectNode element, DisconnectNode element	-	-	1	Required	
		type	"MeshImprovement"	1	Required	

ConnectNode		Node element	-	-	1	Optional	
	Node	TransportAddress element	-	-	1	Required	
			ID	Node ID(String)	1	required	
		Transp orAdd ress	Transport layer address (String)	-	1	Required	
				protocol	classification of protocol(String) such as ”TCP”, ”SOAP”, ”HTTP”, ”BEEP”, ”Bluetooth” , “OBEX”, “IEEE1394”	1	Required
				type	classification of address(String) such as ”IPv4”, ”IPv6”, ”URL”,”Bluetooth”, “IEEE1394”	1	Required
DisconnectNode		Node element	-	-	1	Optional	
	Node	TransportAddress element	-	-	1	Required	
			ID	Node ID(String)	1	required	
		Transp ortAd dress	Transport layer address (String)	-	1	Required	
				protocol	classification of protocol(String) such as ”TCP”, ”SOAP”, ”HTTP”, ”BEEP”, ”Bluetooth” , “OBEX”, “IEEE1394”	1	Required
				type	classification of address(String) such as ”IPv4”, ”IPv6”, ”URL”,”Bluetooth”, “IEEE1394”	1	Required

Table 38: Fields of ServiceAdvertise message

Element name			Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Core			XML fragment	-	-	1	Required	
				xmlns	URI	1	Required	
	ComType		“Unicast”	-	-	1	Required	
	MsgID		Message ID(Sting)	-	-	1	Required	
	ReplyID		-	-	-	-	unused	
	MsgType		“Advertise”	-	-	1	Required	
	CommunityID		Community ID(String)	-	-	1	Optional	
	Source		Node ID(String)	-	-	1	Required	
	Destination		Route, Target element	-	-	1	Required	
		Route	-	-	-	-	unused	
				Node	Node ID(String)	-	unused	
		Target	Node ID, Multicast Group ID (Sting)	-	-	1	Required	
	TraceRoute		Route element	-	-	-	unused	
		Route	Route, Target element	-	-	-	unused	
				Node	Node ID(String)	-	unused	
	HopCount		Hop count (Integer)	-	-	-	unused	
	GatewayAction		“Hook”,”Hop”	-	-	-	unused	

	SessionID	Session ID(String)	-	-	-	unused	
	MsgBody	XML fragment	-	-	1	Required	
			protocol	URI	1	Required	

Table 39: Mapping of ServiceAdvertise message to PUCC Core Protocol

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <MsgType>Advertise</MsgType>
  <Source>nd2</Source>
  <Destination>
    <Target>nd1</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Basic Service Protocol" >
    <ServiceAdvertise xmlns=" Namespace of PUCC Basic Service Protocol" >
      <Service type=" MeshImplovement" >
        <ConnectNode>
          <Node ID=" 975870cd-f9bb-4305-1235-e65e29f6029" >
            <TransportAddress protocol=" TCP"  type=" IPv4" >1074.126.155</TransportAddress>
          </Node>
        </ConnectNode>
        <DisconnectNode>
          <Node ID="  968742ab-f9bb-4305-9900-f98e56f12352" >
            <TransportAddress protocol=" TCP"  type=" IPv4" >1074.126.100</TransportAddress>
          </Node>
        </DisconnectNode>
      </Service>
    </ServiceAdvertise>
  </MsgBody>
</Core>
```

Figure 40. A sample of ServiceAdvertise message

6.3.1.3.1. ServiceAdvertise element

The ServiceAdvertise element is used to provide services from a control node to PUCC nodes. This element has Service element which describes the service data.

6.3.1.3.2. Service element

The Service element has type attribute which designates type of service such as mesh improvement. This element has 2 kinds of element, “ConnectNode” and “DisconnectNode”.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Service	XML fragment	-	-	1	Required	
		type	“MeshImprovement”	1	Required	
ConnectNode	XML fragment	-	-	1	Optional	
Node	TransportAddress element	-	-	Multiple	Required	
		ID	Node ID(String)	1	required	
Transp	Transport layer	-	-	Multiple	Required	

		ortAddress	address (String)	protocol	classification of protocol(String) such as ”TCP”, ”SOAP”, ”HTTP”, ”BEEP”, ”Bluetooth” , “OBEX”, “IEEE1394”	1	Required	
				type	classification of address(String) such as ”IPv4”, ”IPv6”, ”URL”,”Bluetooth”, “IEEE1394”	1	Required	
	DisconnectNode		XML fragment	-	-	1	Optional	
		Node	TransportAddress element	-	-	Multiple	Required	
				ID	Node ID(String)	1	required	
		TransportAddress	Transport layer address (String)	-	-	Multiple	Required	
				protocol	classification of protocol(String) such as ”TCP”, ”SOAP”, ”HTTP”, ”BEEP”, ”Bluetooth” , “OBEX”, “IEEE1394”	1	Required	
				type	classification of address(String) such as ”IPv4”, ”IPv6”, ”URL”,”Bluetooth”, “IEEE1394”	1	Required	

Table 40: Content of Service element

A) ConnectNode element

The ConnectNode element is used to designate the nodes to which the node is connected. This element has Node element, which is the same as the Node element of the ServiceResponse message.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
ConnectNode	XML fragment	-	-	1	Optional	
Node	TransportAddress element	-	-	Multiple	Required	
		ID	Node ID(String)	1	Required	
Transport Address	Transport layer address (String)	-	-	Multiple	Required	
		protocol	classification of protocol(String) such as ”TCP”, ”SOAP”, ”HTTP”, ”BEEP”, ”Bluetooth” , “OBEX”, “IEEE1394”	1	Required	
		type	classification of address(String) such as ”IPv4”, ”IPv6”, ”URL”,”Bluetooth”, “IEEE1394”	1	Required	

Table 41: Content of ConnectNode element

B) DisconnectNode element

The DisconnectNode element is used to designate nodes which the node has to disconnect to. This element has Node element which is same as the Node element of the ServiceResponse message.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
DisconnectNode	XML fragment	-	-	1	Optional	
Node	TransportAddress element	-	-	Multiple	Required	
		ID	Node ID(String)	1	Required	
Transport Address	Transport layer address (String)	-	-	Multiple	Required	
		protocol	classification of protocol(String) such as ”TCP”, ”SOAP”, ”HTTP”, ”BEEP”, ”Bluetooth” , “OBEX”, “IEEE1394”	1	Required	
		type	classification of address(String) such as ”IPv4”, ”IPv6”, ”URL”,”Bluetooth”, “IEEE1394”	1	Required	

Table 42: Content of DisconnectNode element

6.3.2. Service Type

A control node provides 5 kinds of service for PUCC nodes in the PUCC network managed by the control node. The services of the control node can be extended in the future.

Service Name	mode	Description
AdjacentNode	Reactive (Request/Response)	Provides the best adjacent node to connect to for a participating node.
Route	Reactive	Provides a path information between given source node and destination node.
NameResolution	Reactive	Provides a transport layer address of a specific node using the Node ID.
Cost	Reactive	Provides the cost between given source node and destination node.
MeshImprovement	Proactive	Improves the topology of a PUCC network.

Table 43: The services of the control node

6.3.2.1. AdjacentNode service

The AdjacentNode service is used to indicate the PUCC node to which a new participating node is connected. The control node can construct a topology of PUCC network according to its policy using this service. Additionally the control node can recover the split of a PUCC network. When the PUCC network is disconnected, a control node can notify isolated PUCC nodes of new adjacent nodes to which they connect.

The PUCC node sends a ServiceRequest message with a condition that the type attribute of the Service element = “AdjacentNode”. The control node replies a ServiceResponse message with a condition that the type attribute of the Service element =”AdjacentNode” and using Node element. This service involves a ServiceRequest message and a

ServiceResponse message.

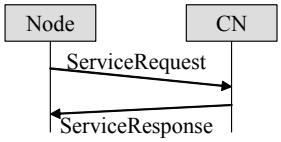


Figure 41. Sequence of AdjacentNode service

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <MsgType>Request</MsgType>
  <Source>nd1</Source>
  <Destination>
    <Target>nd2</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Basic Service Protocol" >
    <ServiceRequest xmlns=" Namespace of PUCC Basic Service Protocol" >
      <Service type=" AdjacentNode" />
    </ServiceRequest>
  </MsgBody>
</Core>
```

Figure 42. A sample of ServiceRequest message for the AdjacentNode service

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <ReplyID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687</ReplyID>
  <MsgType>Response</MsgType>
  <Source>nd2</Source>
  <Destination>
    <Target>nd1</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Basic Service Protocol" >
    <ServiceResponse xmlns=" Namespace of PUCC Basic Service Protocol" >
      <Service type=" AdjacentNode" >
        <Node ID=" 975870cd-f9bb-4305-1235-e65e29f6029" >
          <TransportAddress protocol=" TCP"  type=" IPv4" >107.4.126.155</TransportAddress>
        </Node>
      <Service>
    </ServiceResponse>
  </MsgBody>
</Core>
```

Figure 43. A sample of ServiceResponse message for the AdjacentNode service

6.3.2.2. Route service

The Route service is used to provide path information between the source node (a PUCC node) and the destination node specified by the source node. The control node can provide path information to PUCC nodes using this service. It allows PUCC nodes to efficiently communicate with distant nodes.

The PUCC node sends a ServiceRequest message with the condition that the type attribute of the Service element

= “Route”; the destination node is designated by using a NodeID element. The control node replies with a ServiceResponse message with the condition that the type attribute of the Service element =”Route”; the path information is designated using a Destination element. This service involves a ServiceRequest message and a ServiceResponse message.

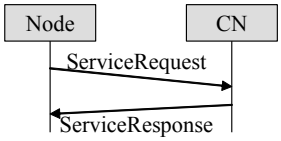


Figure 44. Sequence of AdjacentNode service

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <MsgType>Request</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>975870cd-f9bb-4305-1235-e65e29f6029</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Basic Service Protocol" >
    <ServiceRequest xmlns=" Namespace of PUCC Basic Service Protocol" >
      <Service type=" Route" >
        <NodeID>968985ab-e6aa-5842-1234-f98e56f15687</NodeID>
      </Service>
    </ServiceRequest>
  </MsgBody>
</Core>
```

Figure 45. A sample of ServiceRequest message for the Route service

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <ReplyID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687</ReplyID>
  <MsgType>Response</MsgType>
  <Source>975870cd-f9bb-4305-1235-e65e29f6029</Source>
  <Destination>
    <Target>968742ab-f9bb-4305-9900-f98e56f12352</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Basic Service Protocol" >
    <ServiceRequest xmlns=" Namespace of PUCC Basic Service Protocol" >
      <Service type=" Route" >
        <Destination>
          <Route Node=" 115869aa-88bb-4355-8935-e33e29f6029" >
            <Target>968985ab-e6aa-5842-1234-f98e56f15687</Target>
          </Route>
        <Destination>
      </Service>
    </ServiceRequest>
  </MsgBody>
</Core>
```

Figure 46. A sample of ServiceResponse message for the Route service

6.3.2.3. NameResolution service

The NameResolution service is used to provide transport layer addresses of a node whose node ID is specified by a PUCC node. The control node can provide the transport layer addresses of the node to PUCC nodes using this service. It allows the PUCC nodes to know transport layer address of nodes.

The PUCC node sends a ServiceRequest message with the condition that the type attribute of the Service element = “NameResolution” and designates Node ID using a NodeID element. The control node replies a ServiceResponse message with the condition that the type attribute of the Service element =”NameResolution” and designate the transport layer addresses using Node element. This service involves a ServiceRequest message and a ServiceResponse message.

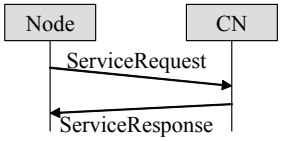


Figure 47. Sequence of AdjacentNode service

```
<Core xmlns=” Namespace of PUCC Core Protocol” >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <MsgType>Request</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>975870cd-f9bb-4305-1235-e65e29f6029</Target>
  </Destination>
  <MsgBody protocol=” Namespace of PUCC Basic Service Protocol” >
    <ServiceRequest xmlns=” Namespace of PUCC Basic Service Protocol” >
      <Service type=” NameResolution” >
        <NodeID>968985ab-e6aa-5842-1234-f98e56f15687</NodeID>
      </Service>
    </ServiceRequest>
  </MsgBody>
</Core>
```

Figure 48. A sample of ServiceRequest message for the NameResolution service

```
<Core xmlns=” Namespace of PUCC Core Protocol” >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <ReplyID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687</ReplyID>
  <MsgType>Response</MsgType>
  <Source>975870cd-f9bb-4305-1235-e65e29f6029</Source>
  <Destination>
    <Target>968742ab-f9bb-4305-9900-f98e56f12352</Target>
  </Destination>
  <MsgBody protocol=” Namespace of PUCC Basic Service Protocol” >
    <ServiceResponse xmlns=” Namespace of PUCC Basic Service Protocol” >
      <Service type=” NameResolution” >
        <Node ID=” 968985ab-e6aa-5842-1234-f98e56f15687” >
          <TransportAddress protocol=” TCP”  type=” IPv4” >10.74.126.155</TransportAddress>
        </Node>
```

```
      </Service>
    </ServiceResponse>
  </MsgBody>
</Core>
```

Figure 49. A sample of ServiceResponse message for the NameResolution service

6.3.2.4. Cost service

The Cost service is used to provide cost information between the source node (a PUCC node) and the destination node designated by the source node. The control node can provide the cost information of a node to node connection for PUCC nodes using this service. It allows the PUCC nodes to discover the cost of a path between the source node and the destination node.

The PUCC node sends a ServiceRequest message with the condition that the type attribute of the Service element = “Cost”; Node ID is designated by a Source element and a Destination element. The control node replies with a ServiceResponse message with the condition that the type attribute of the Service element =”Cost”; the cost of the path is designated using HopCount element. This service involves a ServiceRequest message and a ServiceResponse message.

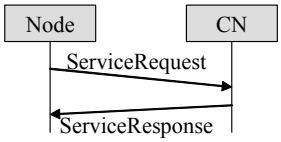


Figure 50. Sequence of AdjacentNode service

```
<Core xmlns=” Namespace of PUCC Core Protocol” >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <MsgType>Request</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>975870cd-f9bb-4305-1235-e65e29f6029</Target>
  </Destination>
  <MsgBody protocol=” Namespace of PUCC Basic Service Protocol” >
    <ServiceRequest xmlns=” Namespace of PUCC Basic Service Protocol” >
      <Service type=” Cost” >
        <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
        <Destination>968985ab-e6aa-5842-1234-f98e56f15687</Destination>
      </Service>
    </ServiceRequest>
  </MsgBody>
</Core>
```

Figure 51. A sample of ServiceRequest message for the Cost service

```
<Core xmlns=” Namespace of PUCC Core Protocol” >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <ReplyID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687</ReplyID>
  <MsgType>Response</MsgType>
```

```
  <Source>975870cd-f9bb-4305-1235-e65e29f6029</Source>
  <Destination>
    <Target>968742ab-f9bb-4305-9900-f98e56f12352</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Basic Service Protocol" >
    <ServiceResponse xmlns=" Namespace of PUCC Basic Service Protocol" >
      <Service type=" Cost" >
        <HopCount>5</HopCount>
      </Service>
    </ServiceResponse>
  </MsgBody>
</Core>
```

Figure 52. A sample of ServiceResponse message for the Cost service

6.3.2.5. MeshImprovement service

The MeshImprovement service is used to make a PUCC node connect to or disconnect from specified nodes. The control node can improve the topology of a PUCC network by using this service.

The control node sends a ServiceAdvertise message with the condition that the type attribute of the Service element = "MeshImprovement"; Node list is designated using a ConnectNode element and a DisconnectNode element. This service is carried by a ServiceAdvertise message.

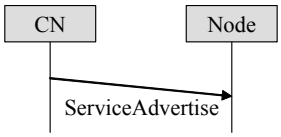


Figure 53. Sequence of AdjacentNode service

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <MsgType>Advertise</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>975870cd-f9bb-4305-1235-e65e29f6029</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Basic Service Protocol" >
    <ServiceAdvertise xmlns=" Namespace of PUCC Basic Service Protocol" >
      <Service type=" MeshImprovement" >
        <ConnectNode>
          <Node ID=" 968985ab-e6aa-5842-1234-f98e56f15687" >
            <TransportAddress protocol=" TCP"  type=" IPv4" >192.168.0.25</TransportAddress>
          </Node>
        </ConnectNode>
        <DisconnectNode>
          <Node ID=" 687453ab-e6aa-4582-9842-f22e56f14667" >
            <TransportAddress protocol=" TCP"  type=" IPv4" >192.168.0.10</TransportAddress>
          </Node>
        </DisconnectNode>
      </Service>
    </ServiceAdvertise>
  </MsgBody>
</Core>
```

Figure 54. A sample of ServiceAdvertise message for the MeshImprovement service

6.4. PUCC Multicast Communication Protocol

The PUCC Multicast Communication Protocols provide multicast communication services to the nodes on the PUCC network. The PUCC Multicast Communication Protocols include:

- Join method

The Join method is used to by a PUCC node to join a certain multicast group. The Join method is a Reactive (Request/Response) type method and involves a Join message and a JoinResponse message.

- Leave method

The Leave method is used by a PUCC node to leave a joined multicast group. The Leave method is a Proactive (Advertise) type method and involves a Leave message.

6.4.2. Join method

6.4.2.1. Join message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Join	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
GroupID	Multicast Group ID(String)	-	-	1	Required	
JoinSource	Node ID(String)	-	-	1	Required	
JoinDestination	Route element, Target element	-	-	1	Required	
Route	Route element, Target element	-	-	Multiple	Optional	
		Node	Node ID (String)	1	Required	
Target	Node ID (String)	-	-	1	Required	

Table 44: Fields of Join message

6.4.2.1.1. Join element

The Join element is used to designate the multicast group to be joined. This element has GroupID element which describes a multicast group ID. This element is essential.

6.4.2.1.2. GroupID element

The GroupID element is used to designate a multicast group ID of a multicast group which a PUCC node tries to join. This element has string data of a multicast group ID. This element is essential.

6.4.2.1.3. JoinSource element

This element is used to designate a node which sends the Join message. This element is essential.

6.4.2.1.4. JoinDestination element

The JoinDestiantion element is used to designate a route to a multicast routing node. The message is propagated according to this element. This element is essential.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	

ComType			"Unicast"	-	-	1	Required	
MsgID			Message ID(String)	-	-	1	Required	
ReplyID			-	-	-	-	unused	
MsgType			"Request"	-	-	1	Required	
CommunityID			Community ID(String)	-	-	1	Optional	
Source			Node ID	-	-	1	Required	
Destination			Target element	-	-	1	Required	
	Route		-	-	-	-	unused	
				Node	Node ID(String)	-	unused	
		Target	Node ID (String)	-	-	1	Required	
TraceRoute			-	-	-	-	unused	
	Route		-	-	-	-	unused	
				Node	Node ID(String)	-	unused	
HopCount			Hop count (Integer)	-	-	-	unused	
GatewayAction			"Hook","Hop"	-	-	-	unused	
SessionID			Session ID(String)	-	-	1	Optional	
MsgBody			XML fragment	-	-	1	Required	
				protocol	URI	1	Required	

Table 45: Mapping of Join message to PUCC Core Protocol

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <MsgType>Request</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>975870cd-f9bb-4305-1235-e65e29f6029</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Multicast Communication Protocol" >
    <Join xmlns=" Namespace of PUCC Multicast Communication Protocol" >
      <GroupID>urn:pucc:multicastgroup:DoCoMo</GroupID>
      <JoinSource>968742ab-f9bb-4305-9900-f98e56f12352</JoinSource>
      <JoinDestination>
        <Route Node=" 975870cd-f9bb-4305-1235-e65e29f6029" >
          <Target>854620cd-f9bb-1124-1235-e65e29f6987</Target>
        </Route>
      </JoinDestinatoin>
    </Join>
  </MsgBody>
</Core>
```

Figure 55. A sample of Join message

6.4.3.1. JoinResponse message

Element name		Element Value	Attribute name (if it has)	Attribute Value	Occurrence	Status	Description
JoinResponse		XML fragment	-	-	1	Required	
			xmlns	URI	1	Required	
	Result	"Success","Failure"	-	-	1	Required	
	Reason	"WrongGroupID", "NotAuthenticated"	-	-	1	Result element dependent	
	GroupID	Multicast Group ID(String)	-	-	1	Required	
			joinDirection	Node ID(String)	1	Required	
	JoinResponseSource	Node ID (String)	-	-	1	Required	

	JoinResponse Destination		Route element, Target element	-	-	1	Required	
		Route	Route element, Target element	Node	Node ID(String)	Multiple	Optional	
		Target	Node ID (String)	-	-	1	Required	

Table 46: Fields of JoinResponse message

6.4.3.1.1. JoinResponse element

The JoinResponse element is used to designate the multicast group to be joined. This element has GroupID element which describe a multicast group ID. This element is essential.

6.4.3.1.2. Result element

The Result element is used to notify the "Success" or "Failure" of joining the multicast group. This element is essential.

6.4.3.1.3. Reason element

The Reason element is used to notify about reason of failure. This element takes following values. If the Result element value is "Failure", Reason element is required. Don't set this element, if it is not so.

Value	Description
WrongGroupID	The designated multicast group doesn't exist.
NotAuthenticated	Authentication failure

Table 47: Values of Reason element

6.4.3.1.4. GroupID element

The GroupID element is used to designate the multicast group ID of the multicast group that the PUCC node wants to join. This element has string data which represents the multicast group ID. The joinDirection attribute is used to designate direction of joining. This attribute is used for reconstruction of a multicast distribution tree. This element is essential.

6.4.3.1.5. JoinResponseSource element

This element is used to designate a node which sends the JoinResponse message. This element is essential.

6.4.3.1.6. JoinResponseDestination element

The JoinResponseDestination element is used to designate a route to a new multicast group node. The message is propagated according to this element. This element is essential.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	1	Required	
MsgType	"Response"	-	-	1	Required	

	CommunityID	Community ID(String)	-	-	1	Optional	
	Source	Node ID(String)	-	-	1	Required	
	Destination	Target element	-	-	1	Required	
	Route	-	-	-	-	unused	
			Node	Node ID (String)	-	unused	
	Target	Node ID (String)	-	-	1	Required	
	TraceRoute	XML fragment	-	-	-	unused	
	Route	-	-	-	-	unused	
			Node	-	-	unused	
	HopCount	Hop count (Integer)	-	-	-	unused	
	GatewayAction	“Hook”,”Hop”	-	-	-	unused	
	SessionID	Session ID(String)	-	-	1	Optional	
	MsgBody	XML fragment	-	-	1	Required	
			xmlns	URI	1	Required	

Table 48: Mapping of JoinResponse message to PUCC Core Protocol

```
<Core xmlns=” Namespace of PUCC Core Protocol” >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <ReplyID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687</ReplyID>
  <MsgType>Request</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>975870cd-f9bb-4305-1235-e65e29f6029</Target>
  </Destination>
  <MsgBody protocol=” Namespace of PUCC Multicast Communication Protocol” >
    <JoinResponse xmlns=” Namespace of PUCC Multicast Communication Protocol” >
      <Result>Success</Result>
      <GroupID
joinDirection=” 968742ab-f9bb-4305-9900-f98e56f12352” >urn:pucc:multicastgroup:DoCoMo</GroupID>
      <JoinResponseSource>968742ab-f9bb-4305-9900-f98e56f12352</JoinResponseSource>
      <JoinResponseDestination>
        <Route Node=” 975870cd-f9bb-4305-1235-e65e29f6029” >
          <Target>854620cd-f9bb-1124-1235-e65e29f6987</Target>
        </Route>
      </JoinResponseDestination>
    </JoinResponse>
  </MsgBody>
</Core>
```

Figure 56. A sample of JoinResponse message

6.4.3.2. Sequence of Join method

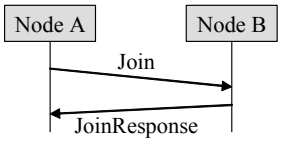


Figure 57. Sequence of Join method

6.4.4. Leave method

6.4.4.1. Leave message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Leave	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
GroupID	Multicast Group ID(String)	-	-	1	Required	

Table 49: Fields of Leave message

6.4.4.1.1. Leave element

The Leave element is used to indicate the intention to leave a multicast group. This element has GroupID element which describes a multicast group ID. This element is essential.

6.4.4.1.2. GroupID element

The GroupID element is used to designate the multicast group ID of the multicast group that the PUCC node wants to leave. This element has string data which represents the multicast group ID. This element is essential.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	-	unused	
MsgType	"Advertise"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Target element	-	-	1	Required	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
Target	Node ID (String)	-	-	1	Required	
TraceRoute	-	-	-	-	unused	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
HopCount	Hop count (Integer)	-	-	-	unused	
GatewayAction	"Hook","Hop"	-	-	-	unused	
SessionID	Session ID(String)	-	-	1	Optional	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI	1	Required	

Table 50: Mapping of Leave message to PUCC Core Protocol

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <MsgType>Advertise</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>975870cd-f9bb-4305-1235-e65e29f6029</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Multicast Communication Protocol" >
    <Leave xmlns=" Namespace of PUCC Multicast Communication Protocol" >
```

```
      <GroupID>urn:pucc:multicastgroup:DoCoMo</GroupID>
    </Leave>
  </MsgBody>
</Core>
```

Figure 58. A sample of Leave message

6.4.4.2. Sequence of Leave method

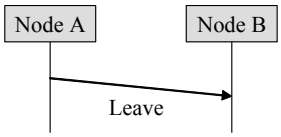


Figure 59. Sequence of Leave method

6.5. PUCC Multicast Service Protocol

The PUCC Multicast Service Protocol defines communication between a node and a control node regarding multicast communication. The node communicates with the control node using this protocol to receive multicast-related services. The PUCC Multicast Service Protocol includes:

- JoinDeclare method

The JoinDeclare method is used by a PUCC node to declare its intention to a control node to join a certain multicast group. The JoinDeclare method is a Reactive (Request/Response) type method and involves a JoinDeclare message and a JoinDeclareResponse message.

- LeaveDeclare method

The LeaveDeclare method is used by a PUCC node to declare to a control node its intention to leave a joined multicast group. The LeaveDeclare method is a Proactive (Advertise) type method and involves a LeaveDeclare message.

- MulticastServiceProvide method

The MulticastServiceProvide method is used to receive multicast services from a control node. The MulticastServiceProvide method is a Proactive (Advertise) type method and involves a MulticastServiceRequest message and a MulticastServiceResponse.

6.5.1 JoinDeclare method

6.5.1.1. JoinDeclare message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
JoinDeclare	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
GroupID	Multicast Group ID(String)	-	-	1	Required	

Table 51: Fields of JoinDeclare message

6.5.1.1.1. JoinDeclare element

The JoinDeclare element is used to declare to a control node the node's intention to join a multicast group. This element has GroupID element which describes a multicast group ID. This element is essential.

6.5.1.1.2. GroupID element

The GroupID element is used to designate the multicast group ID of the multicast group that the PUCC node wants to join. This element has string data that presents multicast group ID. This element is essential.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	

ComType			"Unicast"	-	-	1	Required	
MsgID			Message ID(String)	-	-	1	Required	
ReplyID			Message ID(String)	-	-	-	unused	
MsgType			"Request"	-	-	1	Required	
CommunityID			Community ID(String)	-	-	1	Optional	
Source			Node ID(String)	-	-	1	Required	
Destination			Route, Target element	-	-	1	Required	
	Route		Route, Target element	-	-	-	unused	
				Node	Node ID (String)	-	unused	
		Target	Node ID (String)	-	-	1	Required	
TraceRoute			Route element	-	-	-	unused	
	Route		-	-	-	-	unused	
				Node	Node ID(String)	-	unused	
HopCount			Hop count (Integer)	-	-	-	unused	
GatewayAction			"Hook","Hop"	-	-	-	unused	
SessionID			Session ID(String)	-	-	-	unused	
MsgBody			XML fragment	-	-	1	Required	
				protocol	URI	1	Required	

Table 52: Mapping of JoinDeclare message to PUCC Core Protocol

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <MsgType>Request</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>975870cd-f9bb-4305-1235-e65e29f6029</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Multicast Service Protocol" >
    <JoinDeclare xmlns="  Namespace of PUCC Multicast Service Protocol" >
      <GroupID>urn:pucc:multicastgroup:DoCoMo</GroupID>
    </JoinDeclare>
  </MsgBody>
</Core>
```

Figure 60. A sample of JoinDeclare message

6.5.1.2. JoinDeclareResponse message

Element name			Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
JoinDeclareResponse			XML fragment	-	-	1	Required	
				xmlns	URI	1	Required	
	Result		"Success","Failure"	-	-	1	Required	
	Reason		String	-	-	1	Result element dependent	
	GroupID		Multicast Group ID(String)	-	-	1	Required	
	JoinDestination		Route, Target element	-	-	1	Required	
		Route	Route, Target element	-	-	1	Required	
				Node	Node ID (String)	1	Required	
		Target	Node ID (String)	-	-	1	Required	

Table 53: Fields of JoinDeclareResponse message

6.5.1.2.1. JoinDeclareResponse element

The JoinDeclareResponse element is used by a node to declare to a control node its intention to join a multicast group. This element has Result element, Reason element, GroupID element, and JoinDestination element..

6.5.1.2.2. Result element

The Result element is used to notify the ”Success” or “Failure” of joining a multicast group.

6.5.1.2.3. Reason element

The Reason element is used to notify about reason of failure. This element takes following values.

Value	Description
WrongGroupID	The designated multicast group doesn't exist.
NotAuthenticated	Authentication failure

Table 54: Values of Reason element

6.5.1.2.4. GroupID element

The GroupID element is used to designate the multicast group ID of the multicast group that a PUCC node wants to join. This element has string data that represents multicast group ID.

6.5.1.2.5. JoinDestination element

The JoinDestiantion element is used to designate a route to a new multicast routing node. The message is propagated according to this element.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	“Unicast”	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	1	Required	
MsgType	“Response”	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Target element	-	-	1	Required	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
Target	Node ID (String)	-	-	1	Required	
TraceRoute	-	-	-	-	unused	
Route	-	-	-	-	unused	
		Node	NodeID(String)	-	unused	
HopCount	Hop count (Integer)	-	-	-	unused	
GatewayAction	“Hook”,”Hop”	-	-	-	unused	
SessionID	Session ID(String)	-	-	-	unused	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI	1	Required	

Table 55: Mapping of JoinDeclareResponse message to PUCC Core Protocol

```
<Core xmlns=” Namespace of PUCC Core Protocol” >
  <ComType>Unicast</ComType>
```

```
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <ReplyID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687</ReplyID>
  <MsgType>Request</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>975870cd-f9bb-4305-1235-e65e29f6029</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Multicast Service Protocol" >
    <JoinDeclareResponse xmlns=" Namespace of PUCC Multicast Service Protocol" >
      <Result>Success</Result>
      <GroupID>urn:pucc:multicastgroup:DoCoMo</GroupID>
      <JoinDestination>
        <Route Node=" 975870cd-f9bb-4305-1235-e65e29f6029" >
          <Target>854620cd-f9bb-1124-1235-e65e29f6987</Target>
        </Route>
      </JoinDestination>
    </JoinDeclareResponse>
  </MsgBody>
</Core>
```

Figure 61. A sample of JoinDeclareResponse message

6.5.1.3. Sequence of JoinDeclare method

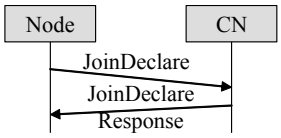


Figure 62. Sequence of JoinDeclare method

6.5.2. LeaveDeclare method

6.5.2.1. LeaveDeclare message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
LeaveDeclare	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
GroupID	Multicast Group ID(String)	-	-	1	Required	

Table 56: Fields of LeaveDeclare message

6.5.2.1.1. LeaveDeclare element

The LeaveDeclare element is used by a node to declare to a control node its intention to leave a multicast group. This element has GroupID element which describes a multicast group ID.

6.5.2.1.2. GroupID element

The GroupID element is used to designate a multicast group ID of a multicast group which a PUCC node tries to leave. This element has string data of a multicast group ID.

Element name	Element Value	Attribute name	Attribute Value	Occurr	Status	Description

		(if present)		ence		
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	-	unused	
MsgType	"Advertise"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Target element	-	-	1	Required	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
Target	Node ID (String)	-	-	1	Required	
TraceRoute	-	-	-	-	unused	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
HopCount	Hop count (Integer)	-	-	-	unused	
GatewayAction	"Hook","Hop"	-	-	-	unused	
SessionID	Session ID(String)	-	-	-	unused	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI	1	Required	

Table 57: Mapping of LeaveDeclare message to PUCC Core Protocol

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <MsgType>Advertise</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>975870cd-f9bb-4305-1235-e65e29f6029</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Multicast Service Protocol" >
    <LeaveDeclare xmlns=" Namespace of PUCC Multicast Service Protocol" >
      <GroupID>urn:pucc:multicastgroup:DoCoMo</GroupID>
    </LeaveDeclare>
  </MsgBody>
</Core>
```

Figure 63. A sample of LeaveDeclare message

6.5.2.2. Sequence of LeaveDeclare method

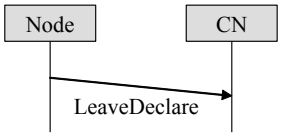


Figure 64. A Sequence of LeaveDeclare

6.5.3. MulticastServiceProvide method

6.5.3.1. MulticastServiceRequest message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
MulticastServiceRequest	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
GroupID	Multicast Group ID(String)	-	-	1	Type attribute dependent	
Service	-	-	-	1	Required	
		type	Type of Service	1	Required	

Table 58: Fields of MulticastServiceRequest message

6.5.3.1.1. MulticastServiceRequest element

The MulticastServiceRequest element is used to describe a multicast-related service of a control node. This element has a service element which designates type of service such as providing a multicast routing node, providing multicast group information and so on.

6.5.3.1.2. Service element

The Service element has no element and one type attribute. The type attribute is used to designate service type such as RoutingNode service and GroupList service. (For details see 6.5.4 Multicast Service Type)

Following is the mapping of MulticastServiceRequest message to the PUCC Core Protocol.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	-	unused	
MsgType	"Request"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Route, Target element	-	-	1	Required	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
Target	Node ID (String)	-	-	Multiple	Required	
TraceRoute	-	-	-	-	unused	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
HopCount	Hop count (Integer)	-	-	-	unused	
GatewayAction	"Hook","Hop"	-	-	-	unused	
SessionID	Session ID(String)	-	-	-	unused	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI	1	Required	

Table 59: Mapping of MulticastServiceRequest message to PUCC Core Protocol

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
```

```
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <MsgType>Request</MsgType>
  <Source>nd1</Source>
  <Destination>
    <Target>nd2</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Multicast Service Protocol" >
    <MulticastServiceRequest xmlns=" Namespace of PUCC Multicast Service Protocol" >
      <GroupID>urn:pucc:multicastgroup:DoCoMo</GroupID>
      <Service type=" RoutingNode" />
    </MulticastServiceRequest>
  </MsgBody>
</Core>
```

Figure 65. A sample of MulticastServiceRequest message

6.5.3.2. MulticastServiceResponse message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
MulticastServiceResponse	XML fragment or "Failure"	-	-	1	Required	
		xmlns	URI	1	Required	
GroupID	Multicast Group ID(String)	-	-	1	type attribute dependent	
Service	XML fragment	-	-	1	Required	
		type	Type of Service	1	Required	

Table 60: Fields of MulticastServiceResponse message

6.5.3.2.1. MulticastServiceResponse element

The MulticastServiceResponse element is used to provide a multicast-related service for PUCC nodes. This element has a service element that designates the type of service such as providing a multicast routing node and providing multicast group information.

6.5.3.2.2. GroupID element

The GroupID element is used to designate a multicast group ID of a multicast group which a PUCC node is joining. This element has string data of a multicast group ID. When type attribute of Service element is "RoutingNode", this element is essential. When type attribute of Service element is "GroupList", this element is unused and omitted.

6.5.3.2.3. Service element

The Service element has JoinDestination element or GroupList element or string of "Failure". And it has one type attribute. The type attribute is used to designate service type such as RoutingNode service and GroupList service. (For details see 6.5.4 Multicast Service Type)

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Service	XML fragment or "Failure"	-	-	1	Required	
		type	Type of Service	1	Required	
JoinDestination	Route element, Target element	-	-	1	type attribute dependent	
Route	Route element, Target element	Node	Node ID(String)	Multiple	Optional	
Target	Node ID (String)	-	-	1	Required	

GroupList		GroupInformation element	-	-	1	type attribute dependent	
	GroupInformation	GroupID element, Description element, Number element, Application element	-	-	Multiple	Required	
			ID	Sequential number (Integer)	1	Required	
		Group ID	Multicast Group ID(String)	-	-	1	Required	
		Description	Detail of group(String)	-	-	1	Optional	
		Number	Number of group (Integer)	-	-	1	Optional	
		Application	Used application (URI)	-	-	Multiple	Optional	

Table 61: Fields of Service element for MulticastServiceResponse

A) JoinDestination element

When type attribute of Service element is "RoutingNode", the JoinDestiantion element is used to designate a route to a multicast routing node. The message is propagated according to this element.

B) GroupList element

When type attribute of Service element is "GroupList", the GroupList element is used to designate information of available multicast groups. This element has GroupInformation elements.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
GroupList	XML fragment	-	-	1	type attribute dependent	
GroupInformation	XML fragment	-	-	Multiple	Required	
		ID	Sequential number (Integer)	1	Required	

Table 62: Fields of GroupList element for MulticastServiceResponse

The GroupInformation element is used to describe information of an available multicast group using a GroupID element, a Description element, a Number element and an Application element.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
GroupInformation	XML fragment	-	-	Multiple	Required	
		ID	Sequential number (Integer)	1	Required	
GroupID	Multicast Group ID(String)	-	-	1	Required	
Description	Detail of group(String)	-	-	1	Optional	
Number	Number of group (Integer)	-	-	1	Optional	
Application	Used application (URI)	-	-	Multiple	Optional	

Table 63: Fields of GroupInformation element for MulticastServiceResponse

The GroupID element is used to designate group ID. The Description element is used to provide details of a multicast group. The Number element is used to designate the number of nodes in a multicast group. The

Application element is used to designate information of applications used by a multicast group.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	1	Required	
MsgType	"Response"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Target element	-	-	1	Required	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
Target	Node ID (String)	-	-	Multiple	Required	
TraceRoute	-	-	-	-	unused	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
HopCount	Hop count (Integer)	-	-	-	unused	
GatewayAction	"Hook","Hop"	-	-	-	unused	
SessionID	Session ID(String)	-	-	-	unused	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI	1	Required	

Table 64: Mapping of MulticastServiceRequest message to PUCC Core Protocol

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <ReplyID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687</ReplyID>
  <MsgType>Response</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>968985ab-e6aa-5842-1234-f98e56f15687</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Multicast Service Protocol" >
    <MulticastServiceResponse xmlns=" Namespace of PUCC Multicast Service Protocol" >
      <GroupID>urn:pucc:multicastgroup:DoCoMo</GroupID>
      <Service type=" RoutingNode" >
        <JoinDestination>
          <Route Node=" 975870cd-f9bb-4305-1235-e65e29f6029" >
            <Target>854620cd-f9bb-1124-1235-e65e29f6987</Target>
          </Route>
        </JoinDestination>
      </Service>
    </MulticastServiceResponse>
  </MsgBody>
</Core>
```

Figure 66. A sample of MulticastServiceResponse message

6.5.3.3. MulticastServiceAdvertise message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
MulticastServiceAdvertise	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
GroupID	Multicast Group ID(String)	-	-	1	type attribute dependent	
Service	XML fragment	-	-	1	Required	
		type	Type of Service	1	Required	

Table 65: Fields of MulticastServiceAdvertise message

6.5.3.3.1. MulticastServiceAdvertise element

The MulticastServiceAdvertise element is used to describe a multicast-related service of a control node. This element has a GroupID element which designates the multicast group ID and a Service element which designates the type of service.

6.5.3.3.2. GroupID element

The GroupID element is used to designate a multicast group ID of a multicast group which a PUCC node is joining. This element has string data of a multicast group ID.

6.5.3.3.3. Service element

The Service element has a NewRouting element and a StaleRouting element. And it has one type attribute. The type attribute is used to designate TreeImprovement service. (For details to 6.5.4 Multicast Service Type)

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Service	XML fragment	-	-	1	Required	
		type	"TreeImprovement"	1	Required	
NewRouting	JoinDestination element	-	-	1	Optional	
JoinDestination	Route, Target element	-	-	1	Required	
Route	Route, Target element	-	-	1	Optional	
		Node	Node ID(String)	1	Required	
Target	Node ID(String)	-	-	1	Required	
StaleRouting	LeaveDestination element	-	-	1	Optional	
LeaveDestination	Target element	-	-	1	Required	
Target	Node ID(String)	-	-	1	Required	

Table 66: Fields of Service element for MulticastServiceAdvertise

A) NewRouting element

The NewRouting element is used to designate a route to a new multicast routing node to join. This element has a JoinDestination element.

B) StaleRouting element

The StaleRouting element is used to designate a route to the multicast routing node to leave. This element has

LeaveDestination element.

The following table shows the mapping of MulticastServiceAdvertise to PUCC Core Protocol.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	-	unused	
MsgType	"Advertise"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Target element	-	-	1	Required	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
Target	Node ID (String)	-	-	1	Required	
TraceRoute	-	-	-	-	unused	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
HopCount	Hop count (Integer)	-	-	-	unused	
GatewayAction	"Hook","Hop"	-	-	-	unused	
SessionID	Session ID(String)	-	-	-	unused	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI	1	Required	

Table 67: Mapping of MulticastServiceAdvertise message to PUCC Core Protocol

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <MsgType>Advertise</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>975870cd-f9bb-4305-1235-e65e29f6029</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Multicast Service Protocol" >
    <MulticastServiceAdvertise xmlns=" Namespace of PUCC Multicast Service Protocol" >
      <GroupID>urn:pucc:multicastgroup:DoCoMo</GroupID>
      <Service type=" TreeImplovement" >
        <NewRouting>
          <JoinDestination>
            <Route Node=" 975870cd-f9bb-4305-1235-e65e29f6029" >
              <Target>854620cd-f9bb-1124-1235-e65e29f6987</Target>
            </Route>
          </JoinDestination>
        </NewRouting>
        <StaleRouting>
          <LeaveDestination>
            <Target>854620cd-f9bb-1124-1235-e65e29f6988</Target>
          </LeaveDestination>
        </StaleRouting>
      </Service>
```

```
    </MulticastServiceAdvertise>
  </MsgBody>
</Core>
```

Figure 67. A sample of MulticastServiceAdvertise message

6.5.4. Multicast Service Type

A control node provides 3 multicast-related services for PUCC nodes managed by the control node. The multicast-related services of the control node can be extended in the future as needed.

Service Name	mode	Description
RoutingNode	Reactive (Request/Response)	Informs requesting node of the nearest adjacent routing node to connect to.
TreeImprovement	Proactive(Advertise)	Improves a multicast tree.
GroupList	Reactive(Request/Response)	Provides information of available multicast groups.

Table 68: The multicast-related services of the control node

6.5.4.1. RoutingNode Service

The RoutingNode service is used to indicate the first multicast routing node to which a new node should join.

The PUCC node sends a MulticastServiceRequest message with the condition that the type attribute of the Service element = “RoutingNode”. The control node replies with a MulticastServiceResponse message with the condition that the type attribute of the Service element = “RoutingNode”; it notifies a node of joining of a multicast routing node using JoinDestination element. This service involves a MulticastServiceRequest message and a MulticastServiceResponse message.

The sequence of RoutingNode service is as follows.

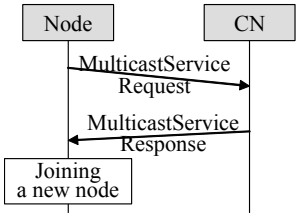


Figure 68. Sequence of RoutingNode service

Following is samples of message for the AdjacentMember service.

```
<Core xmlns=” Namespace of PUCC Core Protocol” >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <MsgType>Request</MsgType>
  <Source>nd1</Source>
  <Destination>
    <Target>nd2</Target>
  </Destination>
  <MsgBody protocol=” Namespace of PUCC Multicast Service Protocol” >
    <MulticastServiceRequest xmlns=” Namespace of PUCC Multicast Service Protocol” >
      <GroupID>urn:pucc:multicastgroup:DoCoMo</GroupID>
      <Service type=” RoutingNode” />
    </MulticastServiceRequest>
```

```
  </MsgBody>
</Core>
```

Figure 69. A sample of MulticastServiceRequest message for the AdjacentMember service

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687 </MsgID>
  <ReplyID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</ReplyID>
  <MsgType>Response</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>968985ab-e6aa-5842-1234-f98e56f15687</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Multicast Service Protocol" >
    <MulticastServiceResponse xmlns=" Namespace of PUCC Multicast Service Protocol" >
      <GroupID>urn: pucc:multicastgroup:DoCoMo</GroupID>
      <Service type=" RoutingNode" >
        <JoinDestination>
          <Route Node=" 975870cd-f9bb-4305-1235-e65e29f6029" >
            <Target>854620cd-f9bb-1124-1235-e65e29f6987</Target>
          </Route>
        </JoinDestination>
      </Service>
    </MulticastServiceResponse>
  </MsgBody>
</Core>
```

Figure 70. A sample of MulticastServiceResponse message for the AdjacentMember service

6.5.4.2. TreeImprovement Service

The TreeImprovement service is used to notify multicast routing nodes to join a new multicast routing node or to leave the stale multicast routing node. The control node can use it to improve a multicast tree.

The control node sends a MulticastServiceAdvertise message with the condition that the type attribute of the Service element = “TreeImprovement” and designate the node list using a NewRouting element and a StaleRouting element. This service involves a MulticastServiceAdvertise message.

The sequence of the TreeImprovement service is as follows.

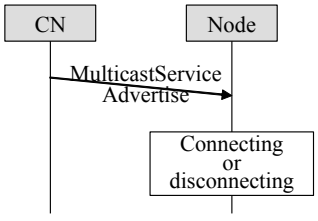


Figure 71. Sequence of TreeImprovemnt service

Following is samples of messages for TreeImprovement service.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
```

```
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <MsgType>Advertise</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>975870cd-f9bb-4305-1235-e65e29f6029</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Multicast Service Protocol" >
    <MulticastServiceAdvertise xmlns=" Namespace of PUCC Multicast Service Protocol" >
      <GroupID>urn:pucc:multicastgroup:DoCoMo</GroupID>
      <Service type=" TreeImplovement" >
        <NewRouting>
          <JoinDestination>
            <Route Node=" 975870cd-f9bb-4305-1235-e65e29f6029" >
              <Target>854620cd-f9bb-1124-1235-e65e29f6987</Target>
            </Route>
          </JoinDestination>
        </NewRouting>
        <StaleRouting>
          <LeaveDestination>
            <Target>854620cd-f9bb-1124-1235-e65e29f6988</Target>
          </LeaveDestination>
        </StaleRouting>
      </Service>
    </MulticastServiceAdvertise>
  </MsgBody>
</Core>
```

Figure 72. A sample of MulticastServiceAdvertise message for the TreeImprovement service

6.5.4.3. GroupList Service

The GroupList service is used to provide information of available multicast groups to the PUCC nodes. The control node can provide the information of available multicast groups using this service. The PUCC nodes can discover the available multicast groups using this service.

The PUCC node sends a MulticastServiceRequest message with the condition that the type attribute of the Service element = “GroupList”. The control node replies with a MulticastServiceResponse message with the condition that the type attribute of the Service element =”GroupList”, and provides a list of available multicast groups using GroupList element. This service involves a MulticastServiceRequest message and a MulticastServiceResponse message.

The sequence of GroupList service is as follows.

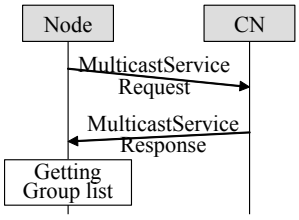


Figure 73. Sequence of GroupList service

The following is a sample of the message for GroupList service.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <MsgType>Request</MsgType>
  <Source>nd1</Source>
  <Destination>
    <Target>nd2</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Multicast Service Protocol" >
    <MulticastServiceRequest xmlns=" Namespace of PUCC Multicast Service Protocol" >
      <Service type=" GroupList" />
    </MulticastServiceRequest>
  </MsgBody>
</Core>
```

Figure 74. A sample of MulticastServiceRequest message for the GroupList service

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687 </MsgID>
  <ReplyID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</ReplyID>
  <MsgType>Response</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>968985ab-e6aa-5842-1234-f98e56f15687</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Multicast Service Protocol" >
    <MulticastServiceResponse xmlns=" Namespace of PUCC Multicast Service Protocol" >
      <Service type=" GroupList" >
        <GroupList>
          <GroupInformation ID=" 1" >
            <GroupID>urn:pucc:multicastgroup:OnlineGame</Group>
            <Description>This group exists for online game.</Description>
            <Number>145</Number>
            <Application>http://www.xxx.co.jp/PUCC_appli/game</Application>
          </GroupInformation>
          <GroupInformation ID=" 2" >
            <GroupID>urn:pucc:multicastgroup:Chat</Group>
            <Description>Chat group interested in XML</Description>
            <Number>90</Number>
            <Application>http://www.xxx.co.jp/PUCC_appli/chat</Application>
          </GroupInformation>
        </GroupList>
      </Service>
    </MulticastServiceResponse>
  </MsgBody>
</Core>
```

Figure 75. A sample of MulticastServiceResponse message for the GroupList service

6.6. PUCC Control Message Protocol

The PUCC Control Message Protocol provide ancillary functions such as notification of message forwarding error, keep-alive of PUCC connection or session, diagnosis of condition of a node and first PUCC node discovery. The PUCC Control Message Protocols include;

- Error Report method

The Error Report method is used to notify a source node of the forwarding error of a message The Error Report method is a Proactive (Advertise) type method and involves an Error Report message.

- Diagnose method

The Diagnose method is used to measure RTT (Round Trip Time) between PUCC nodes, keep a PUCC connection or session alive and search a route to the PUCC node satisfied the condition of a node. The Diagnose method is a Reactive (Request/Response) type method and involves a Diagnose message and a DiagnoseResponse message.

- Lookfor method

The Lookfor method is used to find the first PUCC node to which a PUCC node can connect in the pure PUCC network. When a PUCC node receives a Lookfor message, if the community ID of the message matches the community ID of a community that it is participating in, the PUCC node responds with a LookforResponse message. The Lookfor method is a Reactive (Request/Response) type method and involves a Lookfor message and a LookforResponse message.

6.6.1. Error Report method

6.6.1.1. ErrorReport message

The ErrorReport message has the following structure.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
ErrorReport	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ErrorMsgID	Message ID of an error message (String)	-	-	1	Required	
Error	String	-	-	1	Required	

Table 69: Fields of ErrorReport message

6.6.1.1.1. ErrorReport element

The ErrorReport element is used to designate an error in message forwarding. This element has an ErrorMsgID element and an Error element. This element is essential.

6.6.1.1.2. ErrorMsgID element

The ErrorMsgID element is used to designate the Message ID of an error message. This element has a string representing the Message ID. This element is essential.

6.6.1.1.3. Error element

The Error element is used to provide a description of an error. The following table shows error descriptions. This element is essential.

No.	Error	Description
1	StaleRoutingInformation	This error indicates that message destination is wrong. (The message forwarding is successful.)
2	DestinationUnreachable	This error designates failure of message forwarding.
3	ParameterProblem	Format of PUCC Core Protocol is wrong.
4	SourceQuench	Message abandoned due to congestion.
5	NodeFailure	This error designates failure of message forwarding to adjacent nodes.
6	UnsupportedProtocol	Correspondent protocol is unsupported.

Table 70: Error items

The following shows the mapping of ErrorReport message to PUCC Core Protocol.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	-	unused	
MsgType	"Advertise"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	XML fragment	-	-	1	Required	
Route	Route, Target element	-	-	Multiple	Optional	
		Node	Node ID (Sting)	1	Required	
Target	Node ID (String)	-	-	Multiple	Required	
TraceRoute	XML fragment	-	-	1	Optional	
Route	Route element	-	-	Multiple	Optional	
		Node	Node ID (Sting)	1	Required	
HopCount	Hop count (Integer)	-	-	-	unused	
GatewayAction	"Hook","Hop"	-	-	-	unused	
SessionID	Session ID(String)	-	-	1	Optional	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI	1	Required	

Table 71: Mapping of ErrorReport message to Core Protocol

The following shows a sample of the ErrorReport message.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687 </MsgID>
  <MsgType>Advertise</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>968985ab-e6aa-5842-1234-f98e56f15687</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Control Message Protocol" >
    <ErrorReport xmlns=" Namespace of PUCC Control Message Protocol" >
      <ErrorMsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</ErrorMsgID>
      <Error>DestinationUnreachable</Error>
    </ErrorReport>
```

```
  </MsgBody>
</Core>
```

Figure 76. A sample of ErrorReport message

6.6.1.2. Sequence of ErrorReport method

The ErrorReport method is a proactive communication mode and involves just an advertise type message.

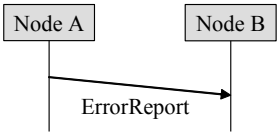


Figure 77. Sequence of ErrorReport service

6.6.2. Diagnose method

6.6.2.1. Diagnose message

The Diagnose message has the following structure.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Diagnose	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
DiagnoseData	Sting	-	-	1	Optional	
DiagnoseDestination	ID (String)	-	-	1	Optional	
		type	NodeID, MulticastGroupID, MulticastRoutingGroupID, ApplicationID	1	Required	

Table 72: Fields of Diagnose message

6.6.2.1.1. Diagnose element

The Diagnose element is used to request a response from other PUCC node. This element has DiagnoseData and DiagnoseDestination element. This element is essential.

6.6.2.1.2. DiagnoseData element

The DiagnoseData element is used to set an arbitrary message to diagnose. If RTT (Round Trip Time) between PUCC nodes is measured, this element has the number of the milliseconds from Jan. 1st, 1970. This element is optional.

6.6.2.1.3. DiagnoseDestination element

The DiagnoseDestination element is used to designate a search condition of a node. This element has type attribute. The following table shows condition descriptions. This element is optional.

No.	Condition	Description
1	NodeID	Search a node which has correspondent node ID
2	MulticastGroupID	Search a node which joins correspondent multicast group
3	MulticastRoutingGroupID	Search a node which participates in routing of correspondent multicast group
4	ApplicationID	Search a node which has correspondent application

Table 73: Condition items

The following shows the mapping of Diagnose message to PUCC Core Protocol.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast","Broadcast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	-	unused	
MsgType	"Request"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Route, Target element	-	-	1	Required	
Route	Route, Target element	-	-	1	Optional	
		Node	Node ID (Sting)	1	Required	
Target	Node ID (String)	-	-	1	Required	
TraceRoute	Route element	-	-	1	Required	
Route	Route element	-	-	Multiple	Optional	
		Node	Node ID (Sting)	1	Required	
HopCount	Hop count (Integer)	-	-	1	Optional	
GatewayAction	"Hook","Hop"	-	-	-	Optional	
SessionID	Session ID(String)	-	-	1	Optional	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI	1	Required	

Table 74: Mapping of Diagnose message to Core Protocol

The following shows a sample Diagnose message.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Broadcast</ComType>
  <MsgID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687 </MsgID>
  <MsgType>Request</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <MsgBody protocol=" Namespace of PUCC Control Message Protocol" >
    <Diagnose xmlns=" Namespace of PUCC Control Mesage Protocol" >
      <DiagnoseData>104353920000</DiagnoseData>
      <DiagnoseDestination type=" NodeID" >968985ab-e6aa-5842-1234-f98e56f15687</DiagnoseDestination>
    </Diagnose>
  </MsgBody>
</Core>
```

Figure 78. A sample of Diagnose message

6.6.2.2. DiagnoseResponse message

The DiagnoseResponse message has the following structure.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
DiagnoseResponse	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
DiagnoseData	Sting	-	-	1	Optional	

Table 75: Fields of Diagnose message

6.6.2.2.1. DiagnoseResponse element

The DiagnoseResponse element is used to respond tor Diagnose message. This element has a DiagnoseData element. This element is essential.

6.6.2.2.2. DiagnoseData element

The DiagnoseData element is used to designate echo of DiagnoseData element of Diagnose message. This element is optional. If DiagnoseData element in the Diagnose message is specified, this element is required.

The following shows the mapping of DiagnoseResponse message to PUCC Core Protocol.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	1	Required	
MsgType	"Response"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Route, Target element	-	-	1	Required	
Route	Route, Target element	-	-	Multiple	Optional	
		Node	Node ID (Sting)	1	Required	
Target	Node ID (String)	-	-	1	Required	
TraceRoute	XML fragment	-	-	1	Required	
Route	Route, Target element	-	-	Multiple	Optional	
		Node	Node ID (Sting)	1	Required	
HopCount	Hop count (Integer)	-	-	-	unused	
GatewayAction	"Hook","Hop"	-	-	-	unused	
SessionID	Session ID(String)	-	-	1	Optional	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI	1	Required	

Table 76: Mapping of DiagnoseResponse message to Core Protocol

The following shows a sample DiagnoseResponse message.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687 </MsgID>
  <ReplyID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</ReplyID>
  <MsgType>Response</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>968985ab-e6aa-5842-1234-f98e56f15687</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Control Message Protocol" >
    <DiagnoseResponse xmlns=" Namespace of PUCC Control Message Protocol" >
      <DiagnoseData>104353920000</DiagnoseData>
    </DiagnoseResponse>
  </MsgBody>
```

```
</Core>
```

Figure 79. A sample of DiagnoseResponse message

6.6.2.3. Sequence of Diagnose method

The Diagnose method is a reactive communication mode and involves a request message and a response message.

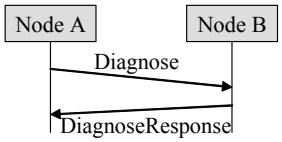


Figure 80. Sequence of Diagnose method

6.6.3. Lookfor method

6.6.3.1. Lookfor message

The Lookfor message has the following structure.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Lookfor	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
		type	”gateway”,”controlnode”	1	Optional	Default =”node”
		count	The number of LookforResponse message.(Integer)	1	Optional	
Condition	Original search condition of application.(String)	-	-	1	Optional	
		protocol	URI	1	Required	

Table 77: Fields of Lookfor message

6.6.3.1.1. Lookfor element

The Lookfor element is used to locate PUCC nodes in a network. This element has type and count attributes and Condition element. The type attribute is optional and used to designate the kind of searching node to locate. If the type attribute is omitted, the value is regarded as “node”. The count attribute is optional and used to designate the request of the number of LookforResponse message from the PUCC network. If the count attribute is omitted, the value is regarded as infinite. This element is essential.

6.6.3.1.2. Condition element

When a node receives a lookfor message with Condition element, the node notifies application that namespace of Condition element specified. This element is used to search a node satisfied conditions that application specifies.

The following shows the mapping of Lookfor message to PUCC Core Protocol. This element is optional.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Core	XML fragment	-	-	1	Required	

		xmlns	URI	1	Required	
ComType	“Broadcast”	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	-	unused	
MsgType	“Request”	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	-	-	-	-	unused	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
Target	-	-	-	-	unused	
TraceRoute	-	-	-	-	unused	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
HopCount	Hop count (Integer)	-	-	-	unused	
GatewayAction	“Hook”,”Hop”	-	-	-	unused	
SessionID	Session ID(String)	-	-	-	unused	
MsgBody	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	

Table 78: Mapping of Lookfor message to Core Protocol

The following shows a sample Lookfor message.

```
<Core xmlns=” Namespace of PUCC Core Protocol” >
  <ComType>Broadcast</ComType>
  <MsgID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687 </MsgID>
  <MsgType>Request</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <MsgBody protocol=” Namespace of PUCC Control Message Protocol” >
    <Lookfor xmlns=” Namespace of PUCC Control Message Protocol” />
  </MsgBody>
</Core>
```

Figure 81. A sample of Lookfor message

6.6.3.2. LookforResponse message

The LookforResponse message has the following structure.

Table 79: Fields of LookforResponse message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
LookforResponse	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
		type	”gateway”,”controlnode”	1	Optional	Default =”node”
Description	String	-	-	1	Optional	
DeviceList	XML fragment	-	-	1	Optional	
Device	XML fragment	-	-	multiple	Optional	
		type	Device ID(URI)	1	Required	
		id	Globally unique ID of the device(String)	1	Optional	
		name	Short user-friendly name of the device (String)	1	Optional	
		private	This attribute shall not be set when the device is defined by PUCC. In case the device is defined by a metadata creator, the name of the vendor etc. who performed extension shall be set. (string)	1	Optional	

6.6.3.2.1. LookforResponse element

The LookforResponse element is used to respond to Lookfor message. This element has type attribute and Description element and DeviceList element including Device element. The type attribute is optional and used to designate the kind of searched node to locate. If this attribute is omitted, the type attribute value is regarded as “node”. This element is essential.

6.6.3.2.2. Description element

Description element is used by the node or application that responded to the search for setting up an arbitrary message it wants to notify the source of the search. This element is optional.

6.6.3.2.3. DeviceList element

DeviceList element is used by the node or application that responded to the search for setting up a list of devices that exist on a node. This element is optional.

6.6.3.2.4. Device element

Device element is used for setting up static metadata for a device. This element is Optional. This element has type and id and name attributes. The type attribute designates the device type. The type attribute is essential. The id attribute designates the globally unique id for the device and should be < 32 characters. The id attribute is optional. The name attribute designates the short user-friendly name of the device and should be < 64 characters. The name attribute is optional. The private attribute is optional.

The structure of Device element is defined by PUCC Device Metadata Template specification. If the device has no metadata, only Specification element is set in the structure of Device element.

The following shows the mapping of LookforResponse message to PUCC Core Protocol.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	1	Required	
MsgType	"Response"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Route, Target element	-	-	1	Required	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
Target	Node ID (String)	-	-	1	Required	
TraceRoute	-	-	-	-	unused	
Route	-		-	-	unused	
		Node	Node ID(String)	-	unused	
HopCount	Hop count (Integer)	-	-	-	unused	
GatewayAction	"Hook","Hop"	-	-	-	unused	
SessionID	Session ID(String)	-	-	-	unused	
MsgBody	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	

Table 80: Mapping of LookforResponse message to Core Protocol

The following show a sample LookforResponse message.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687 </MsgID>
  <MsgType>Request</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>968985ab-e6aa-5842-1234-f98e56f15687</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Control Message Protocol" >
    <LookforResponse xmlns=" Namespace of PUCC Control Message Protocol" />
  </MsgBody>
</Core>
```

Figure 82. A sample of LookforResponse message

6.6.3.3. Sequence of Lookfor method

The Lookfor method is a reactive communication mode and involves a request message and a response message.

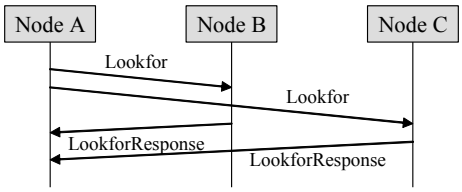


Figure 83. Sequence of Lookfor method

7. Transport protocol bindings

This section describes the transport bindings. We defined all PUCC protocols independent of the underlying layer. In this specification, TCP/UDP binding, Bluetooth binding, IEEE1394 binding, OBEX binding, and HTTP binding are defined.

7.1. TCP Transport

7.1.1. Frame

This section describes the transport binding of the PUCC Protocols over TCP/IP. All PUCC messages are encapsulated by the following frame when the PUCC Protocol is used over TCP/IP.

	Description	Status
Header	There are 2 kinds of header. A) Frame of normal message **P2PFRM<SP>Connectiontype<SP>FrameNo(- SeqNo/WholePacketCounts) <SP>Size<SP>DestNodeID<SP>SrcNodeID (<SP>CommunityID) <CR><LF>** B) Frame in response to a frame error **FRMERR<CR><LF>**	Required
	Connectiontype The Connectiontype parameter is used to designate a connection type. **0: keep-alive** (TCP connection not disconnected after this frame.) **1: single** (Disconnect TCP connection after this frame.)	Required
	FrameNo The FrameNo parameter is sequential number used to distinguish frames. This parameter is set at the default value at 0 and final value is 2147483647. If a PUCC message is divided, "-SeqNo/WholePacketCounts" is given to the end of FrameNo. The frame is specified by changing only the part of "- SeqNo/WholePacketCounts". The position of a frame is specified by "-SeqNo/WholePacketCounts".	Required
	Size The Size parameter is used to designate size of payload (without header and trailer) by the octet number.	Required
	DestNodeID The DestNodeID parameter is used to designate destination node ID.	Required
	SrcNodeID The SrcNodeID parameter is used to designate source node ID.	Required
	CommunityID The CommunityID parameter is used to distinguish which community the message belongs to.	Optional
Payload	A) Frame of normal message **(MIME entity header) <CR><LF> [PUCC message]** The MIME entity header is optional. The Payload has one PUCC message. B) Frame to response to a frame error **[Header of P2PFRM]** The Payload has one header of P2PFRM which is regarded as error.	Required
Trailer	**FRMEND<CR><LF>**	Required

Table 81: The format of the frame for binding of PUCC Protocols over TCP/IP

The following shows a sample P2PFRM frame.

```
P2PFRM 0 0-1/3 702 965841a8-aa56-8746-b456c458450 113541a8-bb56-8746-b456c457581 urn:pucc:community:DoCoMo

<Core xmlns=" Namespace of PUCC Core Protocol" >
 //omitted
</Core>
FRMEND
```

Figure 84. A sample of P2PFRM frame over TCP/IP

The following shows a sample FRMERR frame.

```
FRMERR
P2PFRM 0 0-1/3 702 965841a8-aa56-8746-b456c458450 113541a8-bb56-8746-b456c457581 urn:pucc:community:DoCoMo
FRMEND
```

Figure 85. A sample of FRMERR frame over TCP/IP

7.1.2. Connection Type

We defined two connection types for the transport binding of the PUCC Protocols over TCP/IP.

1. **Persistent connection**

   PUCC nodes which connect to each other keep the TCP connection active and send PUCC messages using it while the PUCC connection is active. When PUCC connection is released, TCP connection is released at once.

2. **Separate connection**

   PUCC node sends a PUCC message using one TCP connection and closes it after sending.

When PUCC nodes establish a PUCC connection, they decide the connection type. The PUCC nodes negotiate connection type using the connectiontype parameter of a frame. The PUCC node sends a Hello message with the value of connectiontype, the other PUCC node send a HelloResponse message with the value of connectiontype. Requesting a keep-alive connection is regard as a higher level request than a single connection.

For example, if a PUCC node receives a Hello message with a request for a keep-alive connection, the PUCC node can select either connection type, keep-alive or single. Conversely, if a PUCC node receives a Hello message with a request for a single connection, the PUCC node can chose only a single connection. The following table shows cases of negotiation on connection type.

Case	Value of connectiontype in the Hello message.	Value of connectiontype in the HelloResponse message.	Determined connection type
1	0 : keep-alive	0 : keep-alive	0 : keep-alive
2	0 : keep-alive	1 : single	1 : single
3	1 : single	1 : single	1 : single

Table 82: Cases of negotiation on connection type

According to the above negotiation, TCP binding handler works as follows.

Case	Value of connectiontype in the Hello message.	Value of connectiontype in the HelloResponse message.	Destination of a HelloResponse message

1	0 : keep-alive	0 : keep-alive	Source address and port of Hello message (Use same socket)
		1 : single	Source address of Hello message and well-known port. (Open another socket)
2	1 : single	1 : single	Source address of Hello message and well-known port. (Open another socket)

Table 83: Selection of port number

The TCP binding handler closes a TCP connection upon receiving a Bye message.

If PUCC connection establishment is unsuccessful, the PUCC node returns a HelloResponse message with a request for a single connection.

7.2. UDP Transport

7.2.1. Frame

This section describes the transport binding of the PUCC Protocols over UDP/IP. The Lookfor message and LookforResponse of PUCC Control Message Protocol are encapsulated with the following frame when the PUCC Protocol is used over UDP/IP. The message frame is almost the same as that of TCP/IP. Differences are the Connectiontype parameter and the DestNodeID parameter. The Connectiontype is only "1" because UDP is used. The DestNodeID is specified as "*" when Lookfor message is sent using IP Multicast.

	Description	Status
Header	There are 2 kinds of header. A) Frame of normal message **P2PFRM<SP>Connectiontype<SP>FrameNo(-SeqNo/WholePacketCounts)<SP>Size <SP>DestNodeID<SP>SrcNodeID (<SP>CommunityID) <CR><LF>** B) Frame to response to a frame error **FRMERR<CR><LF>**	Required
	Connectiontype **1: single** (Using UDP)	Required
	FrameNo The FrameNo parameter is a sequential number used to distinguish frames. This parameter is set at the default value at 0 and the final value is 2147483647. If a PUCC message is divided, "-SeqNo/WholePacketCounts" is given to the end of FrameNo. The frame is specified by changing only the part of "- SeqNo/WholePacketCounts". The position of a frame is specified by "-SeqNo/WholePacketCounts".	Required
	Size The Size parameter is used to designate size of payload (without header and trailer) by the octet number.	Required
	DestNodeID The DestNodeID parameter is used to designate destination node ID. If IP Multicast is used, "*" is specified.	Required
	SrcNodeID The SrcNodeID parameter is used to designate source node ID.	Required
	CommunityID The CommunityID parameter is used to distinguish which community the message belongs to.	Optional
Payload	C) Frame of normal message **(MIME entity header) <CR><LF> [PUCC message]** The MIME entity header is optional. The Payload has one PUCC message. D) Frame to response to a frame error **[Header of P2PFRM]** The Payload has one header of P2PFRM which is regarded as error.	Required
Trailer	**FRMEND<CR><LF>**	Required

Table 84: The format of the frame for binding of PUCC Protocols over UDP/IP

The following shows a sample of the P2PFRM frame over UDP/IP.

```
P2PFRM 0 0-1/3 702 * 113541a8-bb56-8746-b456c457581 urn:pucc:community:DoCoMo

<Core xmlns=" Namespace of PUCC Core Protocol" >
 //omitted
  <MsgBody protocol=" Namespace of PUCC Control Message Protocol" >
    <Lookfor xmlns=" Namespace of PUCC Control Message Protocol" />
  </MsgBody>
</Core>
FRMEND
```

Figure 86. A sample of P2PFRM frame over UDP/IP

7.3. Bluetooth Transport

7.3.1. L2CAP

Bluetooth transport uses the L2CAP connection channel to send and receive PUCC messages. L2CAP packet has a 3 part structure and payload of L2CAP packet is bound to PUCC Protocols.

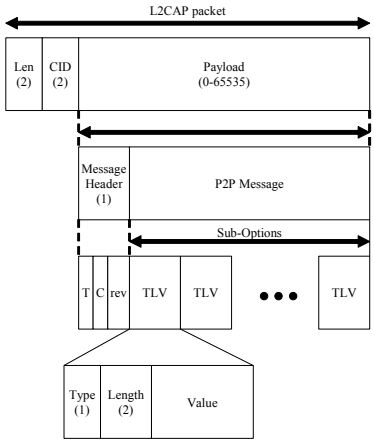


Figure 87. Transport binding of PUCC protocols over Bluetooth

The following shows the detailed structure of PUCC message over L2CAP.

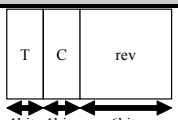
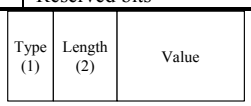
	Description	Status
Header		Required
	Frame Type(T) : 1bit This bit must be set to 1.	Required
	Connection Type(C) : 1 bit This bit indicates whether Bluetooth connection (relation of Master-Slave) must be kept or not, If C is 0, the connection might be kept if possible. If C is 1, the connection must be closed. The detailed behavior of this bit is identical to that described in Section 7.1.1.2.	Required
	Reserved (RSV) : 6bit Reserved bits	Required
Sub-Option	 Sub-Option has TLV (Type-Length-Value) structure. Type and Length fields are statically 1 byte and 2 bytes long, respectively. The Value field length is specified in Length field in units of bytes. There are currently three types defined.	Required
	0x01 (TYPE_XML_DATA) A PUCC message ranging <Core> tag to </Core> tag in bytes of UTF-8	Required
	0x02 (TYPE_SRC_NODEID) A source node's Node ID in bytes of UTF-8.	Required
	0x03 (TYPE_DST_NODEID) A Destination node's Node ID in bytes of UTF-8	Optional

Table 85: Message Format for Bluetooth L2CAP Transport

7.3.2. SPP

Bluetooth transport uses the SPP connection channel to send and receive PUCC messages.

7.3.2.1. Frame

The following shows the detailed structure of PUCC message over SPP.

	Description	Status
Header	There are 2 kinds of header. A) Frame of normal message **P2PFRM<SP>Connectiontype<SP>FrameNo(- SeqNo/WholePacketCounts) <SP>Size<SP>DestNodeID<SP>SrcNodeID (<SP>CommunityID) <CR><LF>** B) Frame in response to a frame error **FRMERR<CR><LF>**	Required
	Connectiontype The Connectiontype parameter is used to designate a connection type. **0: keep-alive** (connection not disconnected after this frame.)	Required
	FrameNo The FrameNo parameter is sequential number used to distinguish frames. This parameter is set at the default value at 0 and final value is 2147483647. If a PUCC message is divided, "-SeqNo/WholePacketCounts" is given to the end of FrameNo. The frame is specified by changing only the part of "- SeqNo/WholePacketCounts". The position of a frame is specified by "-SeqNo/WholePacketCounts".	Required
	Size The Size parameter is used to designate size of payload (without header and trailer) by the octet number.	Required
	DestNodeID The DestNodeID parameter is used to designate destination node ID. If destination node ID is not fixed, "*" is specified.	Required
	SrcNodeID The SrcNodeID parameter is used to designate source node ID.	Required
	CommunityID The CommunityID parameter is used to distinguish which community the message belongs to.	Optional
Payload	A) Frame of normal message **(MIME entity header) <CR><LF> [PUCC message]** The MIME entity header is optional. The Payload has one PUCC message. B) Frame to response to a frame error **[Header of P2PFRM]** The Payload has one header of P2PFRM which is regarded as error.	Required
Trailer	**FRMEND<CR><LF>**	Required

Table 86: Message Format for Bluetooth SPP Transport

7.3.2.2. Connection Type

When PUCC protocol is used over Bluetooth SPP Transport, ConnectionType value is 0 (keep-alive) .

7.3.2.3. Relationship between PUCC Session and Virtual Serial Connection

The side which establishes the PUCC session plays a role of Device A in Bluetooth SPP (sets up a virtual serial connection). The side to which the PUCC session is established plays a role of Device B in Bluetooth SPP (waits for the virtual serial connection to be made to it). The PUCC session is set up on a virtual serial connection (an

RFCOMM session on an L2CAP channel).

7.3.2.4. Virtual Serial Communication Parameters

This document does not specify any setting values for the following parameters used for virtual serial communication. It is assumed that these values are already determined based on the agreement between the nodes.

- Communication speed (baud rate)
- Number of data bits
- Number of stop bits
- Parity (no parity/odd number/even number/mark/space)
- Flow control (with/without flow control, input/output)

7.3.2.5. Paring

This document does not specify any paring method. It is assumed that the BD Address (Bluetooth Device Address) and the path key of the communications counterpart are already known between the nodes).

7.4. IEEE1394 Transport

This section describes the transport binding of the PUCC Protocols over IEEE1394. All PUCC messages are encapsulated by the following frame when the PUCC Protocol is used over IEEE1394.

7.4.1. Frame

The message frame is the same as TCP/IP.

7.4.2. Communication Method

There are two communication methods in IEEE1394. First is an Asynchronous communication. In the Asynchronous communication, it is guaranteed to transmit the packet to the other node surely. However, it isn't guaranteed to delay a transmission. Second is an Isochronous communication. This communication suits a transmission of video or voice data. In the Isochronous communication, it is guaranteed to complete the data transmitting every 125[μ sec]. PUCC Protocols over IEEE1394 basically treats only a control message. A binary data like multimedia contents etc. is not treated. Therefore, the message is sent and received by using IEEE1394 Asynchronous Transmission.

7.4.3. Configuration ROM

A specification of Configuration Rom uses a specification that defined IP over 1394.

7.4.4. Address Resolution

In the address of IEEE1394, There is two ID, IEEE1394 node ID (16bit variable) and EUI-64 address (64bit fixed). IEEE1394 node ID is changed by bus reset. However, EUI-64 address isn’t changed by bus reset, because of equipment unique. Therefore, the transport address of PUCC node uses EUI-64 address. In the IEEE1394 network, IEEE1394 node ID is used. So, each node has a corresponding table of EUI-64 address and IEEE1394 node ID. And, each node renews this corresponding table at each bus reset.

7.5. OBEX Transport

7.5.1. Frame

This section describes the transport binding of the PUCC Protocols over OBEX. All PUCC messages are encapsulated by the following frame when the PUCC Protocol is used over OBEX.

Table 1: The format of the frame for binding of PUCC Protocols over OBEX

	Description	Status
Header	There are 2 kinds of header. A) Frame of normal message **P2PFRM<SP>Connectiontype<SP>FrameNo(- SeqNo/WholePacketCounts) <SP>Size<SP>DestNodeID<SP>SrcNodeID (<SP>CommunityID) <CR><LF>** B) Frame in response to a frame error **FRMERR<CR><LF>**	Required
	Connectiontype The Connectiontype parameter is used to designate a connection type. **0: keep-alive** (Transport connection not disconnected after this frame. When PUCC protocols are over OBEX, this value must be set.) **1: single** (Disconnect transport connection after this frame.)	Required
	FrameNo The FrameNo parameter is sequential number used to distinguish frames. This parameter is set at the default value at 0 and final value is 2147483647. If a message is divided, "-SeqNo/WholePacketCounts" is given to the end of FrameNo. The frame is specified by changing only the part of "- SeqNo/WholePacketCounts". The position of a frame is specified by "-SeqNo/WholePacketCounts".	Required
	Size The Size parameter is used to designate size of payload (without header and trailer) by the octet number.	Required
	DestNodeID The DestNodeID parameter is used to designate destination node ID. If destination node ID is not fixed, "*" is specified.	Required
	SrcNodeID The SrcNodeID parameter is used to designate source node ID.	Required
	CommunityID The CommunityID parameter is used to distinguish which community the message belongs to.	Optional
Payload	A) Frame of normal message **(MIME entity header) <CR><LF> [PUCC message]** The MIME entity header is optional. The Payload has one PUCC message. B) Frame to response to a frame error **[Header of P2PFRM]** The Payload has one header of P2PFRM which is regarded as error.	Required
Trailer	**FRMEND<CR><LF>**	Required

The following shows a sample P2PFRM frame.

```
P2PFRM 0 0-1/3 702 965841a8-aa56-8746-b456c458450 113541a8-bb56-8746-b456c457581 urn:pucc:community:X

<Core xmlns=" Namespace of PUCC Core Protocol" >
 //omitted
</Core>
FRMEND
```

Figure 88 A sample of P2PFRM frame

The following shows a sample FRMERR frame.

```
FRMERR
P2PFRM 0 0-1/3 702 965841a8-aa56-8746-b456c458450 113541a8-bb56-8746-b456c457581 urn:pucc:community:X
FRMEND
```

Figure 89 A sample of FRMERR frame

7.5.2. Connection Type

When PUCC protocol is used over OBEX, ConnectionType value is 0 (keep-alive) .

7.5.3. Transport layer address

The transport layer address of the OBEX binding uses the URL format. The URL of the OBEX server shall be known to the OBEX client in advance with the TransportAddress element etc. of the Resource Information When IrDA is used in the lower layer of OBEX, the value obex:/irclient shall be used as the URL of the OBEX client and the value obex:/irserver shall be used as the URL of the OBEX server.

7.5.4. OBEX Header

The following shows the OBEX headers and values to be set that are used by the OBEX transport.

Table 90 OBEX Header

OBEX Header name	Description	Value when conveying PUCC protocol messages
Name	Object name	P2PFRM
Type	Object type	text/x-p2pfrm
Time	Object's UTC time of last modification in ISO8601 format.	Transmission time of message
Body	Object	A fragment of P2PFRM frame or FRMERR frame.
End of Body	Last chunk of the object	The last fragment of P2PFRM frame or FRMERR frame.

7.5.5. OBEX role

The OBEX is a client-server model protocol consisting of the OBEX client and OBEX server.

In the OBEX transport binding, one node becomes OBEX client and the other node becomes OBEX server.

The OBEX client establishes a session with the OBEX server by the CONNECT operation.

The OBEX client periodically performs the GET operation and polls if there is a message from the

OBEX server to the OBEX client. If there is a message from the OBEX client to the OBEX server, the message is transmitted by the OBEX PUT operation. Because the OBEX GET operation cannot be performed during the OBEX PUT operation, the OBEX GET operation shall be performed after the completion of the OBEX PUT operation.
The OBEX server constantly waits for the GET operation from the OBEX client.
In case there is a message from the OBEX server to the OBEX client, the message is set to the response to the OBEX GET operation and returned. In case there is no message to the OBEX client, an empty message shall be returned to the OBEX client. In case the PUT operation is performed from the OBEX client, the message shall be received and the result shall be returned.
Release of the OBEX session can be performed either from the OBEX client or the OBEX server.

Values for the cycle of polling shall not be specified in this specification.

The sequences are shown below:

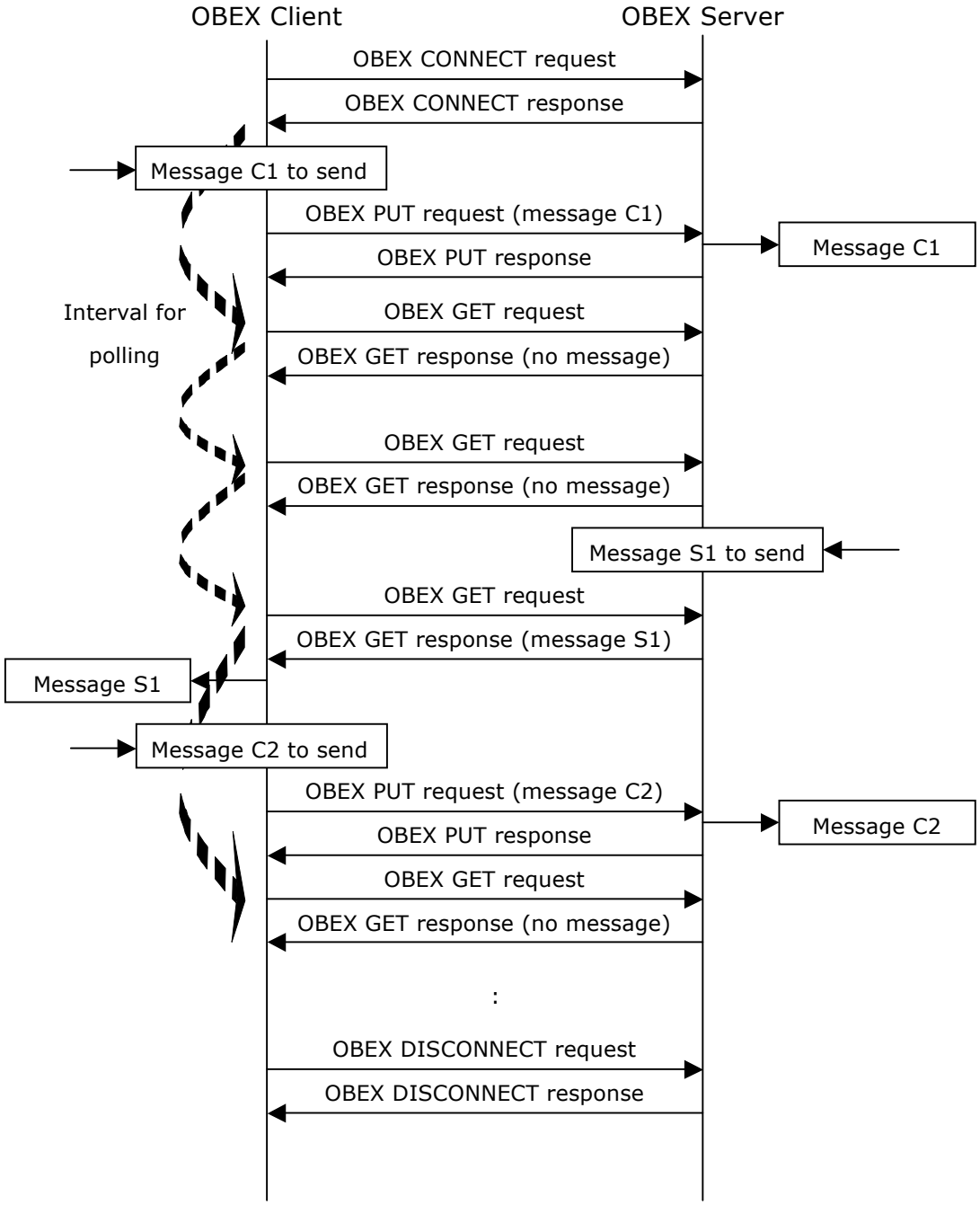


Figure 91 A sample of OBEX protocol sequence

7.6. HTTP Transport

7.6.1. Frame

This section describes the transport binding of the PUCC Protocols over HTTP. All PUCC messages are encapsulated by the following frame when the PUCC Protocol is used over HTTP.

Table 2: The format of the frame for binding of PUCC Protocols over HTTP

	Description	Status
Header	There are 5 kinds of header. A) Frame of normal message P2PFRM<SP>Connectiontype<SP>FrameNo(- SeqNo/WholePacketCounts) <SP>Size<SP>DestNodeID<SP>SrcNodeID (<SP>CommunityID) <CR><LF>	Required
	Connectiontype The Connectiontype parameter is used to designate a connection type. 0: keep-alive (Transport connection not disconnected after this frame.) 1: single (Disconnect transport connection after this frame. When PUCC protocols are over HTTP, this value must be set.)	Required
	FrameNo The FrameNo parameter is sequential number used to distinguish frames. This parameter is set at the default value at 0 and final value is 2147483647. If a message is divided, "-SeqNo/WholePacketCounts" is given to the end of FrameNo. The frame is specified by changing only the part of "- SeqNo/WholePacketCounts". The position of a frame is specified by "-SeqNo/WholePacketCounts". iAppli(Doja Platform) limits the maximum length of each HTTP body. When the length exceeds the maximum length, it conducts segment and reassembly of P2PFRM frames	Required
	Size The Size parameter is used to designate size of payload (without header and trailer) by the octet number.	Required
	DestNodeID The DestNodeID parameter is used to designate destination node ID. If destination node ID is not fixed, "*" is specified.	Required
	SrcNodeID The SrcNodeID parameter is used to designate source node ID.	Required
	CommunityID The CommunityID parameter is used to distinguish which community the message belongs to.	Optional
	B) Frame in response to a frame error FRMERR<CR><LF> C) Frame in response to a normal message. ACKFRM＜CR><LF> D) Frame for polling by HTTP POST request. POLFRM＜CR><LF> E) Frame representing no message in response to a polling frame. NOMSGFRM＜CR><LF>	
Payload	A) Frame of normal message (MIME entity header) <CR><LF> [PUCC message] The MIME entity header is optional. The Payload has one PUCC message. B) Frame to response to a frame error [Header of P2PFRM] The Payload has one header of P2PFRM which is regarded as error. C) Frame in response to a normal message. Payload is not defined and not used. D) Frame for polling.	Optional

	Payload is not defined and not used. E) Frame representing no message left in response to a polling frame. Payload is not defined and not used.	
Trailer	A) Frame of normal message **FRMEND<CR><LF>** B) Frame to response to a frame error **FRMEND<CR><LF>** C) F rame in response to a normal message. Trailer is not defined and not used. D) Frame for polling. Trailer is not defined and not used. E) Frame representing no message left in response to a polling frame. Trailer is not defined and not used.	Optional

The following shows a sample P2PFRM frame.

```
P2PFRM 0 0-1/3 702 965841a8-aa56-8746-b456c458450 113541a8-bb56-8746-b456c457581 urn:pucc:community:X

<Core xmlns=" Namespace of PUCC Core Protocol" >
 //omitted
</Core>
FRMEND
```

Figure 92 A sample of P2PFRM frame

The following shows a sample FRMERR frame.

```
FRMERR
P2PFRM 0 0-1/3 702 965841a8-aa56-8746-b456c458450 113541a8-bb56-8746-b456c457581 urn:pucc:community:X
FRMEND
```

Figure 93 A sample of FRMERR frame

The following shows a sample ACKFRM frame.

```
ACKFRM
```

Figure 94 A sample of ACKFRM frame

The following shows a sample POLFRM frame.

```
POLFRM
```

Figure 95 A sample of POLFRM frame

The following shows a sample NOMSGFRM frame.

NOMSGFRM

Figure 96 A sample of NOMSGFRM frame

7.6.2. Connection Type

When PUCC protocol is used over HTTP, ConnectionType value is 1 (single).

This does not mean that the persistent connection specified in HTTP/1.1 should not be used. The ConnectionType of the HTTP binding does not set the Connection type of transport layer (TCP) for HTTP, but it sets the connection type of the transport layer (HTTP) of the PUCC protocol. Because the HTTP does not have the concept of connection type(it is both the session layer protocol and stateless protocol), 1: single shall be set to the ConnectionType.

Shall not be involved in the setup control of HTTP's Connection header or Keep-Alive header.

7.6.3. Mapping of PUCC connection with HTTP session

Because HTTP is a protocol of the client-server model, one PUCC node performs the HTTP client role and the other PUCC node performs the HTTP server role, and carry out communication. When the HTTP binding is specified, there are two possible cases; the case where both of the PUCC nodes can execute both the HTTP client role and HTTP server role simultaneously, and the case where one of the PUCC nodes can only execute either the HTTP client role or the HTTP server role. In the former case, one PUCC connection is realized by two HTTP sessions per PUCC protocol message transmission direction. In the latter case, one PUCC connection is realized by one HTTP session.

A) The case where both of the PUCC nodes support both the HTTP client role and HTTP server role

In this case, the P2PFRM frame encapsulating the PUCC protocol message is transmitted/received asynchronously over the independent HTTP session per transmission direction. The P2PFRM frame encapsulating the PUCC protocol message is transmitted/received per transmission direction of PUCC protocol message by setting in the body part of the HTTP POST request message. However, if the P2PFRM frame storing the PUCC response message can be set in the body part of the HTTP POST response message, that is also allowed. The ACKFRM frame shall be set for other HTTP POST response messages.

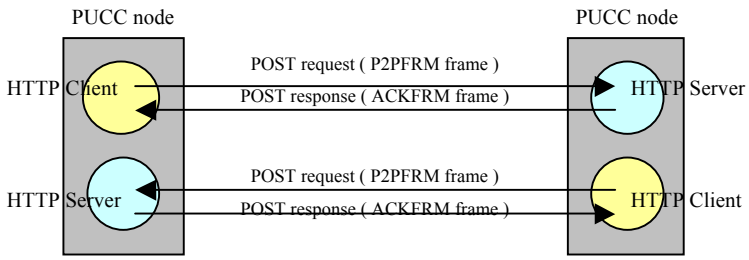


Figure 97 In case PUCC nodes support both HTTP client and HTTP server role

The following table shows the mapping between the PUCC protocol message and HTTP message for this case.

Table 98 Mapping of PUCC protocol message to HTTP message

MsgType of the PUCC protocol message	send / receive	HTTP message
"Request"	Send	HTTP POST request
	Receive	HTTP POST request
"Response"	Send	HTTP POST request or HTTP POST response
	Receive	HTTP POST request or HTTP POST response
"Advertise"	Send	HTTP POST request
	Receive	HTTP POST request

The following shows the sequences.

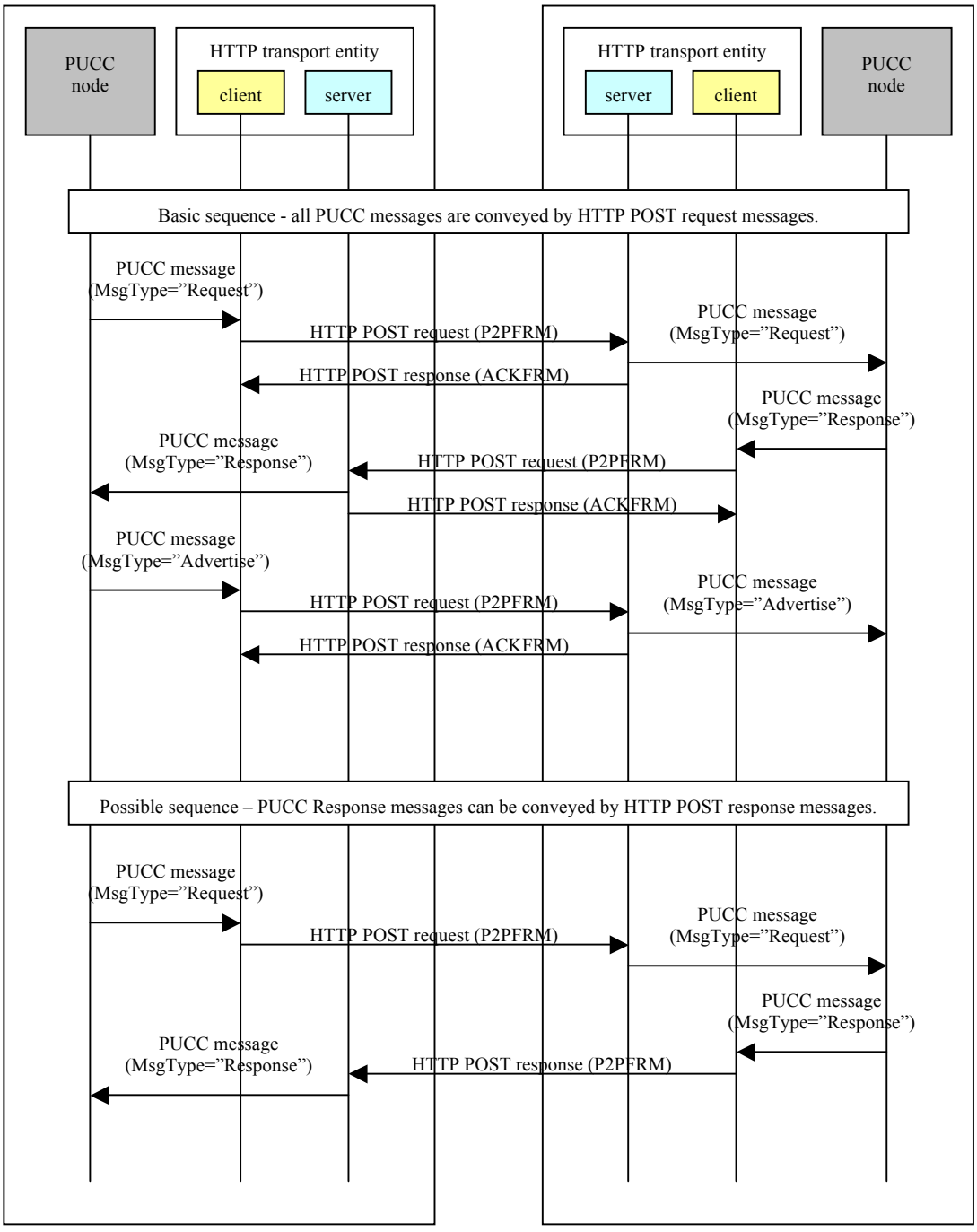


Figure 99 Sequence of sample PUCC protocol message exchange over HTTP between PUCC nodes which support both HTTP client role and HTTP server role

B) The case where one PUCC node only supports either the HTTP client role or HTTP server role

In this case , the P2PFRM frame encapsulating the PUCC protocol message is transmitted/received over one HTTP session synchronously. The P2PFRM frame encapsulating the PUCC protocol message is set in the body part of the HTTP POST request message in one direction, and in the other direction, it is set in the body part of the HTTP POST response message.

Because the HTTP message cannot be pushed from the HTTP server role to the HTTP client role, if the PUCC protocol message with the Msgtype “Request” or “Advertise” is transmitted from the node with HTTP server role to the node with HTTP client role, the smart pull (polling) shall be performed periodically from the HTTP client role to the HTTP server role.

In the body part of the HTTP POST request message for the polling from the PUCC mode with the HTTP client role to the PUCC node with the HTTP server role, the POLFRM frame indicating polling is set.

As a result of the polling, if there is a PUCC protocol message from the PUCC node with the HTTP server role to the PUCC node with the HTTP client role, the P2PFRM frame shall be set in the body part of the HTTP POST response message. If not, NOMSGFRM frame which indicates there is no PUCC protocol message shall be set in the body part of the HTTP POST response message.

Cycle time values for polling shall not be specified in this specification.

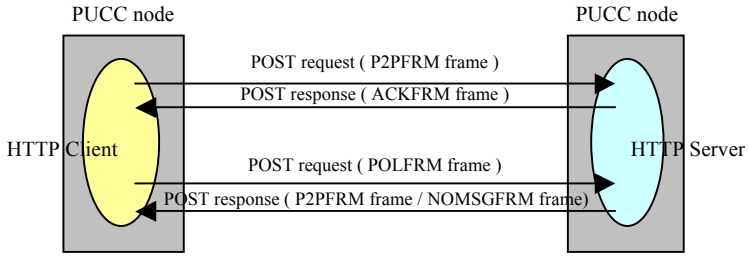


Figure 100 In case PUCC nodes support either HTTP client or HTTP server role

The following table shows the mapping between the PUCC protocol message and the HTTP message for this case.

Table 101 Mapping of PUCC protocol message to HTTP message

MsgType of the PUCC protocol message	send / receive	HTTP message
“Request”	Send	HTTP POST request
	Receive	HTTP POST response (by polling)
“Response”	Send	HTTP POST request
	Receive	HTTP POST response (by polling or not)

“Advertise”	Send	HTTP POST request
	Receive	HTTP POST response (by polling)

The following shows the sequences.

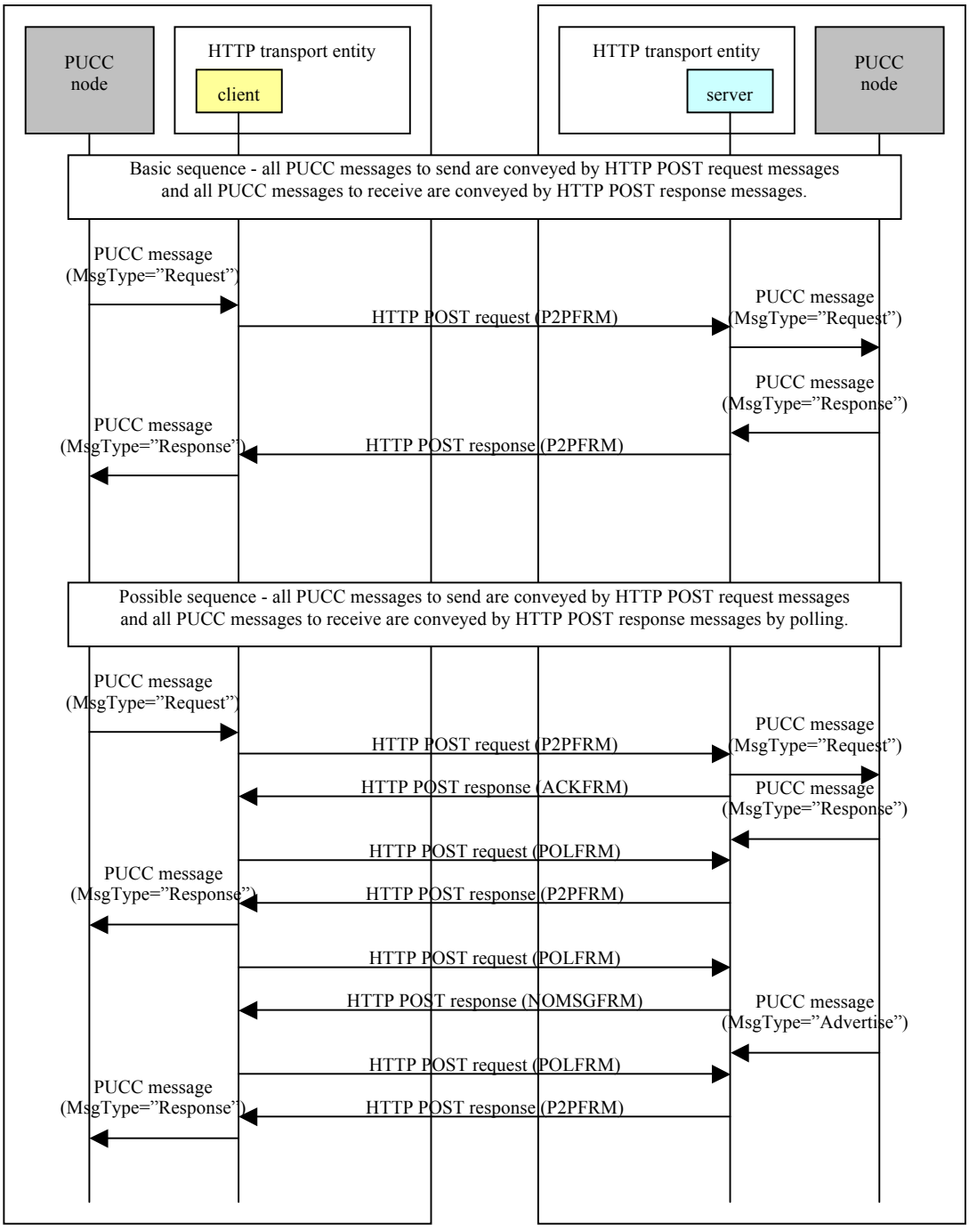


Figure 102 Sequence of sample PUCC protocol message exchange over HTTP between PUCC node which supports HTTP client role and PUCC node which supports HTTP server role

7.6.4. Transport layer address

The URL format shall be used for the transport layer address of the HTTP binding. URL for the HTTP server role shall not be specified. The URL of the HTTP server role shall be known to the PUCC node with the HTTP client role in advance by the TransportAddress element etc. of the Resource Information.

The PUCC node with the HTTP client role sets, in a fixed manner, a special URL ”http:/httpclient” in the TransportAddress element of the Resource Information. The node which has the Resource Information with this URL indicates that it behaves as the HTTP client role.

7.6.5. HTTP version

The HTTP version supported by this specification shall not be specified.

7.6.6. HTTP status code

It is only when the status code of the HTTP POST response is set to ”200 OK” , the frame specified in this specification shall be set to the HTTP POST response. Specification of the HTTP POST response message in the case other status codes are set shall not be specified.

7.6.7. HTTP header

a) Content-Type

”text/x-p2pfrm” shall be set in the Content-Type header of the frame specified in this specification.

Note however, the frame can form the MIME multipart content with other arbitrary contents. In this case, the multipart content type such as multipart/formdata shall be set in the Content-Type header of the multipart content, and the ”text/x-p2pfrm” shall be set in the Content-Type header of the frame part.

b) Content-Length

Frame length shall be set in the Content-Length header of the frame specified in this specification.

Note however, the frame can form the MIME multipart content with other arbitrary contents. In this case, the total length of the multipart content shall be set in the Content-Length header of the multipart content, and frame length shall be set in the Content-Length header of the frame part.

c) Content-Encoding

”chunked” must not be set in the Content-Encoding header of the frame specified in this specification.

d) Host

Server’s FQDN(and a port number if a port other than the Well-Known port is specified)shall be set in the Host header of the HTTP POST request of HTTP/1.1.

Appendix A: Version History

Document number	Date	Note
PUCC Basic Protocol	30 Sep, 2007	Version 1.0
PUCC Basic Protocol	30 Nov, 2009	Version 2.0
PUCC Basic Protocol	22 March, 2012	Version 3.0

Appendix B: First Peer Discovery

To connect to a PUCC network, a PUCC node must first locate the first (nearest) PUCC node to which it then connects. There are two ways of making this connection.

A) Case of Pure PUCC architecture

In the pure PUCC architecture, the joining node sends a Lookfor message around the physical network. As shown in Fig. A-1, when an existing PUCC node receives the Lookfor message, it replies to the joining node with a LookforResponse message. The joining node connects to the PUCC network according to LookforResponse messages received. The Lookfor message is a special connectionless message. For PUCC over IP network, the Lookfor message is realized as an IP multicast message. The LookforResponse message is also a connectionless message. In a PUCC over IP network, the LookforResponse message is realized as an IP multicast message or an IP unicast message.

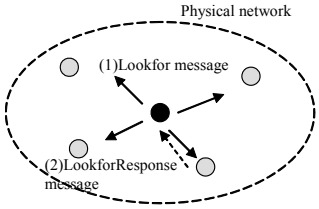


Figure 103. First peer discovery in pure PUCC architecture

The sequence of first peer discovery in the pure PUCC architecture is shown in Fig A-2.

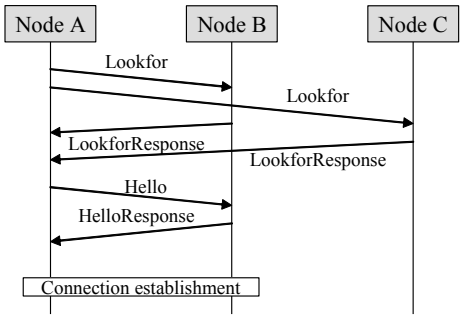


Figure 104. Sequence of the first peer discovery in pure PUCC architecture

B) Case of Hybrid PUCC network

In the hybrid PUCC architecture, the joining PUCC node sends a ServiceRequest message to the control node. As shown in Fig. A-3, when the control node receives the ServiceRequest message, the control node replies to the joining node with a ServiceResponse message which contains information about the first PUCC node. The participating node connects to the PUCC network according to the ServiceResponse message received.

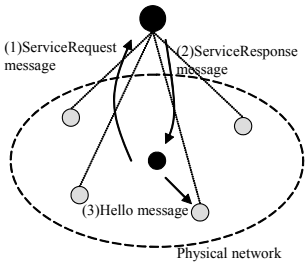


Figure 105. First peer discovery in hybrid PUCC architecture

The sequence of first peer discovery in a hybrid PUCC architecture is shown in Fig. A-4.

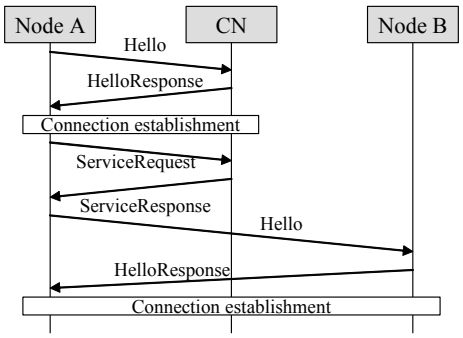


Figure 106. Sequence of the first peer discovery in hybrid PUCC architecture

Appendix C: NAT Transmission

This section describes consideration about how to make a PUCC node in LAN participate in a PUCC network in WAN using an UPnP-enabled IP router.

A) What's UPnP?

UPnP is a mechanism that makes various devices connected to IP network, recognize one another and interconnect seamlessly without complicated settings by users.

The most common function of an UPnP-enabled IP router is automatic NAT setting function cooperated with UPnP-enabled applications. The windows messenger bundled with Windows XP is one of the famous UPnP-enabled applications.

UPnP-enabled devices and UPnP-enabled applications consist of device, service and control point.

- **Device**

"Device" means each UPnP-enabled device and UPnP-enabled application. It is a container of service components. Multiple devices can compose one device. (Combo device)

- **Service**

"Service" means functions provided by UPnP-enabled devices and applications. The way to apply and control

the services is called "Action".

For instance, a clock device provides a timer service. The timer service has a time setting action and a time acquisition action and so on.

- **Control Point**

  It is a controller to detect devices and use services.

  It can be embedded in UPnP-enabled devices and applications, and also can independently exist.

Information of device, service and action is defined using a “Device description document” based on XML. The device has HTTP server and responses a request to get a device description document from the control point.

Following protocol is used for communications between the control point and devices.

- **HTTP/HTTPU/HTTPMU**

All protocol used in UPnP system is based on HTTP. HTTPU is HTTP using UDP. HTTPMU is HTTP using UDP multicast.

- **SSDP (Simple Service Discovery Protocol)**

This protocol is used for control point to detect services.

The control point sends a request message (SSDP search request) to search devices and services using HTTPMU.

The devices always listens UDP port. When it receives an appropriate request message, it replies a response message to the control point using HTTPU.

Conversely, when the device connects to a network, it sends a SSDP presence announce using HTTPMU to notify the control point of its own services.

- **GENA (Generic Event Notification Architecture)**

GENA is defined as function to send and receive a notification message using HTTP over TCP/IP and multicast UDP. It is a mechanism to notify the control point of service state change by the devices.

- **SOAP (Simple Object Access Protocol)**

The control point sends a message to devices to use services provided by the devices using SOAP. The device uses SOAP to send a response to the control point, too.

B) IGD (InternetGatewayDevice)

The specification of UPnP-enabled router is provided in “InternetGatewayDevice (IGD) V1.0”
The specification is available in UPnP forum(http://www.upnp.org/).

IGD provides "Layer3Forwarding Service. And IDG has one or more "WANDevice" and one or more “LAN Device". These devices respectively provide services and "WANDevice" has multiple "WANConnectionDevices".

IDG is not for user or device authentication mechanism and access control mechanism.

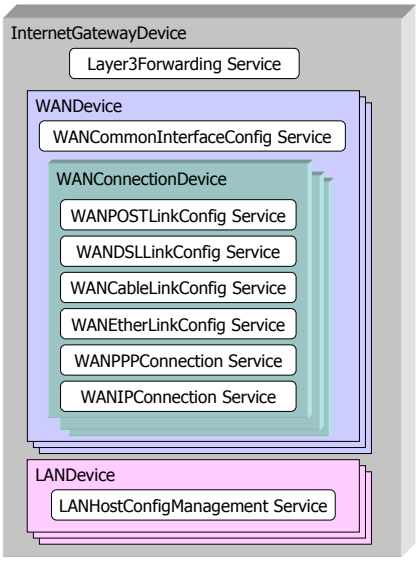


Figure 107. Devices and services of InternetGatewayDevice

C) Application of UPnP to PUCC Platform (Pure and Hybrid mixed PUCC network)

Fig.B-2 and Fig.B-3 show a solution to connect PUCC network in home LAN to PUCC network in Internet using UPnP-enable router.

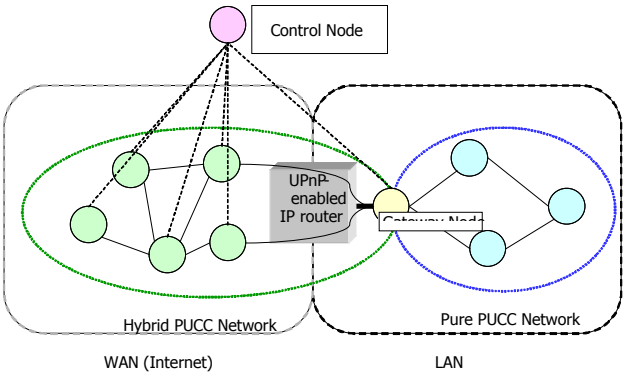


Figure 108. Hybrid and Pure PUCC network with UPnP-enable router

In this case, Gateway node has UPnP functions and control point to acquire global IP address from UPnP-enabled router and set up NAT of it.

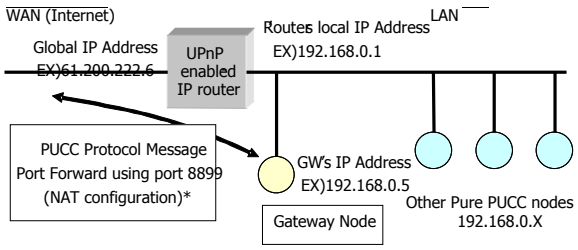


Figure 109. Relationship between PUCC node and UPnP-enabled router

* Port forward is a mechanism that forwards external communication to a internal particular computer. In this case, the router forwards communication using port 8899 of global IP address to a gateway node in LAN.

When the gateway node starts, it finds IGD (InternetGatewayDevice) using SSDP and acquires global IP address of the router and sets up the port forwarding. In that example, the gateway node communicates using local IP address "192.168.0.5", has the global IP address of the router as resource information. Fig.B-4 shows example of resource information that the gateway node sends to the control node.

```
<ResourceData>
    <OwnNodeID>gateway</OwnNodeID>
    <OwnedApplication></OwnedApplication>
    <ConnectionCapability>10</ConnectionCapability>
    <ConnectedNode></ConnectedNode>
    <JoiningGroup></JoiningGroup>
    <RoutingGroup></RoutingGroup>
    <MulticastRouting></MulticastRouting>
    <TransportAddress protocol="TCP" type="IPv4">61.200.222.6</TransportAddress>
</ResourceData>
```

Global IP

Figure 110. Example of resource information which the gateway node sends to control node

- The gateway node uses global IP address of the router as its own IP address of hybrid PUCC side.
- The gateway notifies hybrid PUCC nodes and a control node of global IP address of hybrid PUCC side transport address. And it notifies pure PUCC nodes of local IP address of pure PUCC side transport address.

D) Application of UPnP to PUCC Platform (Pure PUCC network)

When internal and external PUCC network are pure or light PUCC networks, an internal particular pure PUCC node has to have multiple IP address (Global and local). The gateway-like pure or light PUCC node has to control both local and global IP addresses and choose them according to target nodes. In current PUCC Basic Protocol specification, a node can not distinguish between an external PUCC node and an internal PUCC node. It is needed to add such as information of netmask to distinguish the networks.

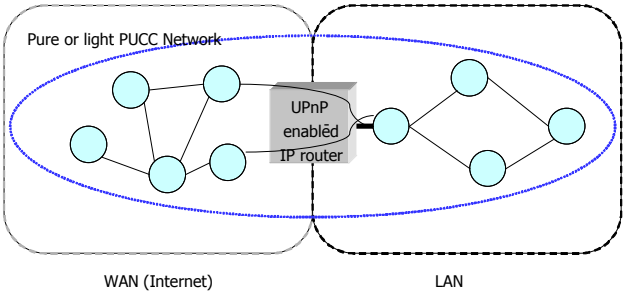


Figure 111. Pure or light PUCC network using an UPnP-enabled router

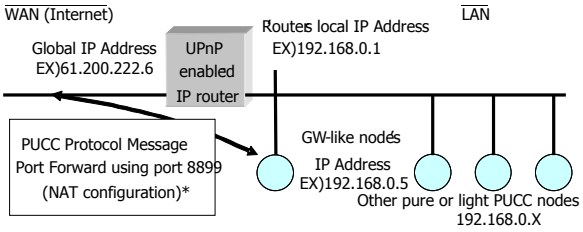


Figure 112. Relationship between a pure or light PUCC node and an UPnP-enabled router

The gateway-like pure or light PUCC node works as a control point of UPnP and acquires global IP address of the router and sets up the port forwarding.

Appendix D: Use cases

A) Distributed Metadata Search Application

The PUCC network is more scalable and offers better management than the client-server model. The PUCC network can be used for locating distributed data in the ubiquitous communication environment. Distributed Metadata Search Applications can be developed with PUCC Protocols.

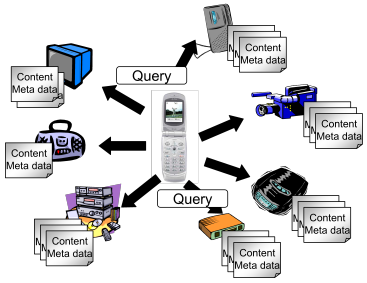


Figure 113. Distributed Metadata Search in the ubiquitous communication environment

B) Mobile Push Service

The PUCC network is adaptable for ad-hoc communication. The PUCC network can be used for advertising push service based on the user's location on the street.

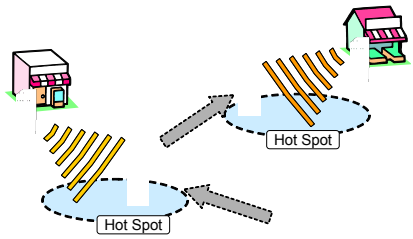


Figure 114. Advertising Push Service on the street

C) Small Personal Network Platform

The PUCC network can construct a confidential network. The PUCC network can be used as a Home appliance network and a personal area network and so on.

Figure 115. Personal Area Network

Appendix E: DTD for PUCC Basic Communication Protocol

A) DTD for Hello message

```
<?xml version="1.0"?>
<!DOCTYPE Core [
<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, Destination, TraceRoute?, SessionID?, MsgBody)>
<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">
<!ELEMENT ComType (#PCDATA)>
<!ELEMENT MsgID (#PCDATA)>
<!ELEMENT MsgType (#PCDATA)>
<!ELEMENT CommunityID (#PCDATA)>
<!ELEMENT Source (#PCDATA)>
<!ELEMENT Target (#PCDATA)>
<!ENTITY % Route.class "(Route? | Target)" >
<!ELEMENT Route %Route.class;>
<!ATTLIST Route Node CDATA #REQUIRED>
<!ELEMENT Destination %Route.class;>
<!ELEMENT TraceRoute %Route.class;>
<!ELEMENT SessionID (#PCDATA)>
<!ELEMENT MsgBody (Hello)>
<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/basic_com">
<!ELEMENT Hello (Mode?, InitiatorUserID?, ResponderUserID?, Authentication?, RequestedSecurity?, RequestedCapability?)>
<!ATTLIST Hello xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/basic_com”
<!-- Mode element is mandatory when Hello message is sent to a control node or a multicast control node
```

or a gateway node. However, Mode element is optional when Hello message is sent to a hybrid node

or a pure node. -->

```
<!ELEMENT Mode (#PCDATA)>
<!ELEMENT InitiatorUserID (#PCDATA)>
<!ELEMENT ResponderUserID (#PCDATA)>
<!ELEMENT Authentication (Response?, Challenge?, RequestedHashAlgorithm?)>
<!ELEMENT Response (#PCDATA)>
<!ELEMENT Challenge (#PCDATA)>
<!ELEMENT RequestedHashAlgorithm (#PCDATA)>
<!ELEMENT RequestedSecurity (CipherAlgorithm?, HashAlgorithm?, IterationCount?, UpdateCount?)>
<!ELEMENT CipherAlgorithm (#PCDATA)>
<!ELEMENT HashAlgorithm (#PCDATA)>
<!ELEMENT IterationCount (#PCDATA)>
<!ELEMENT UpdateCount (#PCDATA)>
<!ELEMENT RequestedCapability (ExtendedProtocol+)>
<!ELEMENT ExtendedProtocol (#PCDATA)>
<!ATTLIST ExtendedProtocol ID NMTOKEN #REQUIRED>
]>
```

B) DTD for HelloResponse message

```
<?xml version="1.0"?>
<!DOCTYPE Core [
<!ELEMENT Core (ComType, MsgID, ReplyID, MsgType, CommunityID?, Source, Destination, TraceRoute?, SessionID?,
MsgBody)>
<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">
<!ELEMENT ComType (#PCDATA)>
<!ELEMENT MsgID (#PCDATA)>
<!ELEMENT ReplyID (#PCDATA)>
<!ELEMENT MsgType (#PCDATA)>
<!ELEMENT CommunityID (#PCDATA)>
<!ELEMENT Source (#PCDATA)>
<!ELEMENT Target (#PCDATA)>
<!ENTITY % Route.class "(Route? | Target)" >
<!ELEMENT Route %Route.class;>
<!ATTLIST Route Node CDATA #REQUIRED>
```

<!ELEMENT Destination %Route.class;>

<!ELEMENT TraceRoute %Route.class;>

<!ELEMENT SessionID (#PCDATA)>

<!ELEMENT MsgBody (HelloResponse)>

<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/basic_com">

<!ELEMENT HelloResponse (Mode?, Result, Reason?, Authentication?, NegotiatedSecurity?, NegotiatedCapability?)>

<!ATTLIST HelloResponse CDATA #FIXED "http://www.pucc.jp/2007/09/basic_com">

<!ELEMENT Mode (#PCDATA)>

<!-- Reason element is required when Result value (#PCDATA) is Failure.

However, Reason element should be absent when Result value (#PCDATA) is Success. -->

<!ELEMENT Result (#PCDATA)>

<!ELEMENT Reason (#PCDATA)>

<!ELEMENT Authentication (Challenge?, RequestedHashAlgorithm?, Response?)>

<!ELEMENT Challenge (#PCDATA)>

<!ELEMENT RequestedHashAlgorithm (#PCDATA)>

<!ELEMENT Response (#PCDATA)>

<!ELEMENT NegotiatedSecurity (CipherAlgorithm?, HashAlgorithm?, IterationCount?, UpdateCount?)>

<!ELEMENT CipherAlgorithm (#PCDATA)>

<!ELEMENT HashAlgorithm (#PCDATA)>

<!ELEMENT IterationCount (#PCDATA)>

<!ELEMENT UpdateCount (#PCDATA)>

<!ELEMENT NegotiatedCapability (ExtendedProtocol+)>

<!ELEMENT ExtendedProtocol (#PCDATA)>

<!ATTLIST ExtendedProtocol ID NMTOKEN #REQUIRED>

]>

C) DTD for Bye message

<?xml version="1.0"?>

<!DOCTYPE Core [

<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, Destination, TraceRoute?, SessionID?, MsgBody)>

<!ATTLIST Core xmlns CDATA #FIXED " http://www.pucc.jp/2007/09/core">

<!ELEMENT ComType (#PCDATA)>

<!ELEMENT MsgID (#PCDATA)>

<!ELEMENT MsgType (#PCDATA)>

<!ELEMENT CommunityID (#PCDATA)>

<!ELEMENT Source (#PCDATA)>

<!ELEMENT Target (#PCDATA)>

<!ENTITY % Route.class "(Route? | Target)" >

<!ELEMENT Route %Route.class;>

<!ATTLIST Route Node CDATA #REQUIRED>

<!ELEMENT Destination %Route.class;>

<!ELEMENT TraceRoute %Route.class;>

<!ELEMENT SessionID (#PCDATA)>

<!ELEMENT MsgBody (Bye)>

<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/basic_com">

<!ELEMENT Bye EMPTY>

<!ATTLIST Bye xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/basic_com">

]>

D) DTD for ResourceInformationRequest message

<?xml version="1.0" ?>

<!DOCTYPE Core [

<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, Destination?, TraceRoute, HopCount?, GatewayAction?, SessionID?, Signature?, MsgBody)>

<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">

<!ELEMENT ComType (#PCDATA)>

<!ELEMENT MsgID (#PCDATA)>

<!ELEMENT MsgType (#PCDATA)>

<!ELEMENT CommunityID (#PCDATA)>

<!ELEMENT Source (#PCDATA)>

<!ELEMENT Target (#PCDATA)>

<!ENTITY % Route.class "(Route | Target)*" >

<!ELEMENT Route %Route.class;>

<!ATTLIST Route Node CDATA #REQUIRED>

<!ELEMENT Destination %Route.class;>

<!-- Destination element is required when ComType (#PCDATA) is Unicast or Multicast.

However, Destination element should be absent when ComType value (#PCDATA) is Broadcast. -->

<!ELEMENT TraceRoute %Route.class;>

<!-- Route element is required within Destination element if the distance from Source node to Target node is
more than 1 hop. However, Route element should be absent within Destination element if the distance
from Source node to Target node is only 1 hop.
Route element is always required within TraceRoute element. -->
<!ELEMENT HopCount (#PCDATA)>
<!-- HopCount element is required when ComType value (#PCDATA) is Broadcast and hop count is finite.
HopCount element should be absent when ComType value (#PCDATA) is Broadcast and hop count is infinite.
HopCount element always should be absent when ComType value (#PCDATA) is Unicast or Multicast. -->
<!ELEMENT GatewayAction (#PCDATA)>
<!ELEMENT SessionID (#PCDATA)>
<!ELEMENT Signature (SignedInfo?, SignedValue?)>
<!ATTLIST Signature xmlns CDATA #FIXED “http://www.w3c.org/2000/09/xmldsig#”>
<!ELEMENT SignedInfo (CanonicalizationMethod?, SignatureMethod?, Referance?)>
<!ATTLIST CanonicalizationMethod Algorithm CDATA #FIXED “http://www.w3c.org/TR/2001/REC-xml-c14n-20010315”>
<!ATTLIST SignatureMethod Algorithm CDATA #REQUIRED>
<!ATTLIST Referance URI CDATA #FIXED “xpointer(/Core)”>
<!ELEMENT SignedValue (#PCDATA)>
<!ELEMENT MsgBody (ResourceInformationRequest)>
<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/basic_com">
<!ELEMENT ResourceInformationRequest EMPTY>
<!ATTLIST ResourceInformationRequest xmlns CDATA #FIXED “http://www.pucc.jp/2007/09/basic_com”>
]>

E) DTD for ResourceInformationResponse message

<?xml version="1.0" ?>
<!DOCTYPE Core [
<!ELEMENT Core (ComType, MsgID, ReplyID, MsgType, CommunityID?, Source, Destination, TraceRoute, SessionID?, Signature?, MsgBody)>
<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">
<!ELEMENT ComType (#PCDATA)>
<!ELEMENT MsgID (#PCDATA)>
<!ELEMENT ReplyID (#PCDATA)>
<!ELEMENT MsgType (#PCDATA)>
<!ELEMENT CommunityID (#PCDATA)>

<!ELEMENT Source (#PCDATA)>

<!ELEMENT Target (#PCDATA)>

<!ENTITY % Route.class "(Route? | Target)" >

<!ELEMENT Route %Route.class;>

<!ATTLIST Route Node CDATA #REQUIRED>

<!ELEMENT Destination %Route.class;>

<!ELEMENT TraceRoute %Route.class;>

<!-- Route element is required within Destination element if the distance from Source node to Target node is

more than 1 hop. However, Route element should be absent within Destination element if the distance

from Source node to Target node is only 1 hop.

Route element is always required within TraceRoute element. -->

<!ELEMENT SessionID (#PCDATA)>

<!ELEMENT Signature (SignedInfo?, SignedValue?)>

<!ATTLIST Signature xmlns CDATA #FIXED “http://www.w3c.org/2000/09/xmldsig#”>

<!ELEMENT SignedInfo (CanonicalizationMethod?, SignatureMethod?, Referance?)>

<!ATTLIST CanonicalizationMethod Algorithm CDATA #FIXED “http://www.w3c.org/TR/2001/REC-xml-c14n-20010315”>

<!ATTLIST SignatureMethod Algorithm CDATA #REQUIRED>

<!ATTLIST Referance URI CDATA #FIXED “xpointer(/Core)”>

<!ELEMENT SignedValue (#PCDATA)>

<!ELEMENT MsgBody (ResourceInformationResponse)>

<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/basic_com">

<!ELEMENT ResourceInformationResponse (ResourceData+)>

<!ATTLIST ResourceInformationResponse xmlns

CDATA #FIXED "http://www.pucc.jp/2007/09/basic_com”>

<!ELEMENT ResourceData (OwnNodeID, Mode?, OwnedApplication?, ConnectionCapability, ConnectedNode, JoiningGroup?,

RoutingGroup?, MulticastRouting?, TransportAddress+)>

<!ELEMENT OwnNodeID (#PCDATA)>

<!ELEMENT Mode (Pure | Hybrid | Gateway | ControlNode) #REQUIRED >

<!ELEMENT OwnedApplication (Application*)>

<!ELEMENT Application (#PCDATA)>

<!ELEMENT ConnectionCapability (#PCDATA)>

<!ELEMENT ConnectedNode (Node*)>

<!ELEMENT Node (TransportAddress+, ConnectionCapability?, RTT?)>

<!ATTLIST Node ID CDATA #REQUIRED>

<!ELEMENT JoiningGroup (GroupID*)>

<!ELEMENT RoutingGroup (GroupID*)>

<!ELEMENT GroupID (#PCDATA)>

<!ELEMENT MulticastRouting (Group*)>

<!ELEMENT Group (NodeID*)>

<!ATTLIST Group ID CDATA #REQUIRED>

<!ELEMENT NodeID (#PCDATA)>

<!ELEMENT TransportAddress (#PCDATA)>

<!ATTLIST TransportAddress protocol CDATA #REQUIRED>

<!ATTLIST TransportAddress type CDATA #REQUIRED>

<!ELEMENT RTT (#PCDATA)>

]>

F) DTD for ResourceInformationAdvertise message

<?xml version="1.0" ?>

<!DOCTYPE Core [

<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, Destination?, TraceRoute, HopCount?, GatewayAction?, SessionID?, Signature?, MsgBody)>

<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">

<!ELEMENT ComType (#PCDATA)>

<!ELEMENT MsgID (#PCDATA)>

<!ELEMENT MsgType (#PCDATA)>

<!ELEMENT CommunityID (#PCDATA)>

<!ELEMENT Source (#PCDATA)>

<!ELEMENT Target (#PCDATA)>

<!ENTITY % Route.class "(Route? | Target)" >

<!ELEMENT Route %Route.class;>

<!ATTLIST Route Node CDATA #REQUIRED>

<!ELEMENT Destination %Route.class;>

<!-- Destination element is required when ComType (#PCDATA) is Unicast or Multicast.

However, Destination element should be absent when ComType value (#PCDATA) is Broadcast. -->

<!ELEMENT TraceRoute %Route.class;>

<!-- Route element is required within Destination element if the distance from Source node to Target node is

more than 1 hop. However, Route element should be absent within Destination element if the distance

```
        from Source node to Target node is only 1 hop.
        Route element is always required within TraceRoute element. -->
<!ELEMENT HopCount (#PCDATA)>
<!-- HopCount element is required when ComType value (#PCDATA) is Broadcast and hop count is finite.
        HopCount element should be absent when ComType value (#PCDATA) is Broadcast and hop count is infinite.
        HopCount element always should be absent when ComType value (#PCDATA) is Unicast or Multicast. -->
<!ELEMENT GatewayAction (#PCDATA)>
<!ELEMENT SessionID (#PCDATA)>
<!ELEMENT Signature (SignedInfo?, SignedValue?)>
<!ATTLIST Signature xmlns CDATA #FIXED “http://www.w3c.org/2000/09/xmldsig#”>
<!ELEMENT SignedInfo (CanonicalizationMethod?, SignatureMethod?, Referance?)>
<!ATTLIST CanonicalizationMethod Algorithm CDATA #FIXED “http://www.w3c.org/TR/2001/REC-xml-c14n-20010315”>
<!ATTLIST SignatureMethod Algorithm CDATA #REQUIRED>
<!ATTLIST Referance URI CDATA #FIXED “xpointer(/Core)”>
<!ELEMENT SignedValue (#PCDATA)>
<!ELEMENT MsgBody (ResourceInformationAdvertise)>
<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/basic_com">
<!ELEMENT ResourceInformationAdvertise (ResourceData*)>
<!ATTLIST ResourceInformationAdvertise xmlns
  CDATA #FIXED “http://www.pucc.jp/2007/09/basic_com”>
<!ELEMENT ResourceData (OwnNodeID, Mode?, OwnedApplication?, ConnectionCapability, ConnectedNode, JoiningGroup?,
RoutingGroup?, MulticastRouting?, TransportAddress+)>
<!ELEMENT OwnNodeID (#PCDATA)>
<!ELEMENT Mode (Pure | Hybrid | Gateway | ControlNode) #REQUIRED >
<!ELEMENT OwnedApplication (Application*)>
<!ELEMENT Application (#PCDATA)>
<!ELEMENT ConnectionCapability (#PCDATA)>
<!ELEMENT ConnectedNode (Node*)>
<!ELEMENT Node (TransportAddress+, ConnectionCapability?, RTT?)>
<!ATTLIST Node ID CDATA #REQUIRED>
<!ELEMENT JoiningGroup (GroupID*)>
<!ELEMENT RoutingGroup (GroupID*)>
<!ELEMENT GroupID (#PCDATA)>
<!ELEMENT MulticastRouting (Group*)>
```

<!ELEMENT Group (NodeID*)>

<!ATTLIST Group ID CDATA #REQUIRED>

<!ELEMENT NodeID (#PCDATA)>

<!ELEMENT TransportAddress (#PCDATA)>

<!ATTLIST TransportAddress protocol CDATA #REQUIRED>

<!ATTLIST TransportAddress type CDATA #REQUIRED>

<!ELEMENT RTT (#PCDATA)>

]>

Appendix F: DTD for PUCC Basic Service Protocol

A) DTD for ServiceRequest message

<?xml version="1.0" ?>

<!DOCTYPE Core [

<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, Destination, MsgBody)>

<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">

<!ELEMENT ComType (#PCDATA)>

<!ELEMENT MsgID (#PCDATA)>

<!ELEMENT MsgType (#PCDATA)>

<!ELEMENT CommunityID (#PCDATA)>

<!ELEMENT Source (#PCDATA)>

<!ELEMENT Target (#PCDATA)>

<!ELEMENT Destination (#PCDATA | Target)*>

<!-- The content of Destination element is not only Target element but also #PCDATA

because Destination element within Service element is #PCDATA. -->

<!ELEMENT MsgBody (ServiceRequest)>

<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/basic_srv">

<!ELEMENT ServiceRequest (Service)>

<!ATTLIST ServiceRequest xmlns CDATA #FIXED “http://www.pucc.jp/2007/09/basic_srv”>

<!ELEMENT Service (Source?, Destination?, NodeID?, UserID?)>

<!-- NodeID element is required when type attribute value of Service element is Route or NameResolution.

Source element and Destination element is required when type attribute value of Service element is Cost.

No element is required when type attribute value of Service element is AdjacentNode. -->

<!ATTLIST Service type (AdjacentNode | Route | NameResolution | Cost) #REQUIRED>

<!ELEMENT NodeID (#PCDATA)>

<!ELEMENT UserID (#PCDATA)>

<!-- Source element and Destination element is defined above at Core parameters. -->

]>

B) DTD for ServiceResponse message

<?xml version="1.0" ?>

<!DOCTYPE Core [

<!ELEMENT Core (ComType, MsgID, ReplyID, MsgType, CommunityID?, Source, Destination, MsgBody)>

<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">

<!ELEMENT ComType (#PCDATA)>

<!ELEMENT MsgID (#PCDATA)>

<!ELEMENT ReplyID (#PCDATA)>

<!ELEMENT MsgType (#PCDATA)>

<!ELEMENT CommunityID (#PCDATA)>

<!ELEMENT Source (#PCDATA)>

<!ELEMENT Target (#PCDATA)>

<!ENTITY % Route.class "(Route? | Target)" >

<!ELEMENT Route %Route.class;>

<!ATTLIST Route Node CDATA #REQUIRED>

<!ELEMENT Destination %Route.class;>

<!-- The content of Destination element is not only Target element but also Route element,

because Destination element within Service element may include Route element. -->

<!ELEMENT MsgBody (ServiceResponse)>

<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/basic_srv">

<!ELEMENT ServiceResponse (Service)>

<!ELEMENT ServiceResponse xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/basic_srv">

<!ELEMENT Service (#PCDATA | Node | Destination | HopCount)>

<!-- Node element is required when type attribute value of Service element is RoutingNode or NameResolution.

Destination element is required when type attribute value of Service element is Route.

HopCount element is required when type attribute value of Service element is Cost. -->

<!-- A content of Service element is #PCDATA, that is "Failure", when no service is provided.

Therefore, as Service content is a mixed content, element type declaration must be described as

"(#PCDATA | XXX)*".

Therefore, "+" mark at Node element should be absent. -->

<!ATTLIST Service type (AdjacentNode | Route | NameResolution | Cost) #REQUIRED>

<!ELEMENT Node (TransportAddress+)>

<!ATTLIST Node ID CDATA #REQUIRED>

<!ELEMENT TransportAddress (#PCDATA)>

<!ATTLIST TransportAddress protocol CDATA #REQUIRED>

<!ATTLIST TransportAddress type CDATA #REQUIRED>

<!ELEMENT HopCount (#PCDATA)>

]>

C) DTD for ServiceAdvertise message

<?xml version="1.0" ?>

<!DOCTYPE Core [

<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, Destination, MsgBody)>

<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">

<!ELEMENT ComType (#PCDATA)>

<!ELEMENT MsgID (#PCDATA)>

<!ELEMENT MsgType (#PCDATA)>

<!ELEMENT CommunityID (#PCDATA)>

<!ELEMENT Source (#PCDATA)>

<!ELEMENT Target (#PCDATA)>

<!ELEMENT Destination (Target)>

<!ELEMENT MsgBody (ServiceAdvertise)>

<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/basic_srv">

<!ELEMENT ServiceAdvertise (Service)>

<!ATTLIST ServiceAdvertise xmlns

CDATA #FIXED "http://www.pucc.jp/2007/09/basic_srv">

<!ELEMENT Service (ConnectNode?, DisconnectNode?)>

<!ATTLIST Service type (MeshImprovement) #REQUIRED>

<!ELEMENT ConnectNode (Node+)>

<!ELEMENT DisconnectNode (Node+)>

<!ELEMENT Node (TransportAddress+)>

<!ATTLIST Node ID CDATA #REQUIRED>

<!ELEMENT TransportAddress (#PCDATA)>

<!ATTLIST TransportAddress protocol CDATA #REQUIRED>

<!ATTLIST TransportAddress type CDATA #REQUIRED>

]>

Appendix G: DTD for PUCC Multicast Communication Protocol

A) DTD for Join message

<?xml version="1.0" ?>

<!DOCTYPE Core [

<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, Destination, TraceRoute?, SessionID?, Signature?, MsgBody)>

<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">

<!ELEMENT ComType (#PCDATA)>

<!ELEMENT MsgID (#PCDATA)>

<!ELEMENT MsgType (#PCDATA)>

<!ELEMENT CommunityID (#PCDATA)>

<!ELEMENT Source (#PCDATA)>

<!ELEMENT Target (#PCDATA)>

<!ENTITY % Route.class "(Route | Target)*" >

<!ELEMENT Route %Route.class;>

<!ATTLIST Route Node CDATA #REQUIRED>

<!ELEMENT Destination %Route.class;>

<!ELEMENT TraceRoute %Route.class;>

<!-- Route element is required within Destination element if the distance from Source node to Target node is more than 1 hop. However, Route element should be absent within Destination element if the distance from Source node to Target node is only 1 hop.
Route element is always required within TraceRoute element. -->

<!ELEMENT SessionID (#PCDATA)>

<!ELEMENT Signature (SignedInfo?, SignedValue?)>

<!ATTLIST Signature xmlns CDATA #FIXED “http://www.w3c.org/2000/09/xmldsig#”>

<!ELEMENT SignedInfo (CanonicalizationMethod?, SignatureMethod?, Referance?)>

<!ATTLIST CanonicalizationMethod Algorithm CDATA #FIXED “http://www.w3c.org/TR/2001/REC-xml-c14n-20010315”>

<!ATTLIST SignatureMethod Algorithm CDATA #REQUIRED>

<!ATTLIST Referance URI CDATA #FIXED “xpointer(/Core)”>

<!ELEMENT SignedValue (#PCDATA)>

<!ELEMENT MsgBody (Join)>

<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/multi_com">

<!ELEMENT Join (GroupID, JoinSource, JoinDestination)>

<!ATTLIST Join xmlns CDATA #FIXED “http://www.pucc.jp/2007/09/multi_com”>

<!ELEMENT GroupID (#PCDATA)>

<!ELEMENT JoinSource (#PCDATA)>

<!ELEMENT JoinDestination %Route.class;>

]>

B) DTD for JoinResponse message

<?xml version="1.0" ?>

<!DOCTYPE Core [

<!ELEMENT Core (ComType, MsgID, ReplyID, MsgType, CommunityID?, Source, Destination, TraceRoute?, SessionID?, Signature?, MsgBody)>

<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">

<!ELEMENT ComType (#PCDATA)>

<!ELEMENT MsgID (#PCDATA)>

<!ELEMENT ReplyID (#PCDATA)>

<!ELEMENT MsgType (#PCDATA)>

<!ELEMENT CommunityID (#PCDATA)>

<!ELEMENT Source (#PCDATA)>

<!ELEMENT Target (#PCDATA)>

<!ENTITY % Route.class "(Route | Target)*" >

<!ELEMENT Route %Route.class;>

<!ATTLIST Route Node CDATA #REQUIRED>

<!ELEMENT Destination %Route.class;>

<!ELEMENT TraceRoute %Route.class;>

<!-- Route element is required within Destination element if the distance from Source node to Target node is
more than 1 hop. However, Route element should be absent within Destination element if the distance
from Source node to Target node is only 1 hop.
Route element is always required within TraceRoute element. -->

<!ELEMENT SessionID (#PCDATA)>

<!ELEMENT Signature (SignedInfo?, SignedValue?)>

<!ATTLIST Signature xmlns CDATA #FIXED “http://www.w3c.org/2000/09/xmldsig#”>

<!ELEMENT SignedInfo (CanonicalizationMethod?, SignatureMethod?, Referance?)>

<!ATTLIST CanonicalizationMethod Algorithm CDATA #FIXED “http://www.w3c.org/TR/2001/REC-xml-c14n-20010315”>

<!ATTLIST SignatureMethod Algorithm CDATA #REQUIRED>

<!ATTLIST Referance URI CDATA #FIXED “xpointer(/Core)”>

<!ELEMENT SignedValue (#PCDATA)>

<!ELEMENT MsgBody (JoinResponse)>

<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/multi_com">

<!ELEMENT JoinResponse (Result, Reason?, GroupID?, JoinResponseSource?, JoinResponseDestination?)>

<!-- Reason element is required when Result value (#PCDATA) is Failure.

However, Reason element should be absent when Result value (#PCDATA) is Success. -->

<!ATTLIST JoinResponse xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/multi_com">

<!ELEMENT Result (#PCDATA)>

<!ELEMENT Reason (#PCDATA)>

<!ELEMENT GroupID (#PCDATA)>

<!ELEMENT JoinResponseSource (#PCDATA)>

<!ELEMENT JoinResponseDestination %Route.class;>

]>

C) DTD for Leave message

<?xml version="1.0" ?>

<!DOCTYPE Core [

<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, Destination, TraceRoute?, SessionID?, Signature?, MsgBody)>

<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">

<!ELEMENT ComType (#PCDATA)>

<!ELEMENT MsgID (#PCDATA)>

<!ELEMENT MsgType (#PCDATA)>

<!ELEMENT CommunityID (#PCDATA)>

<!ELEMENT Source (#PCDATA)>

<!ELEMENT Target (#PCDATA)>

<!ENTITY % Route.class "(Route | Target)*" >

<!ELEMENT Route %Route.class;>

<!ATTLIST Route Node CDATA #REQUIRED>

<!ELEMENT Destination %Route.class;>

<!ELEMENT TraceRoute %Route.class;>

<!-- Route element is required within Destination element if the distance from Source node to Target node is more than 1 hop. However, Route element should be absent within Destination element if the distance from Source node to Target node is only 1 hop.
Route element is always required within TraceRoute element. --><!ELEMENT MsgBody (Leave)>

<!ELEMENT SessionID (#PCDATA)>

<!ELEMENT Signature (SignedInfo?, SignedValue?)>

<!ATTLIST Signature xmlns CDATA #FIXED “http://www.w3c.org/2000/09/xmldsig#”>

<!ELEMENT SignedInfo (CanonicalizationMethod?, SignatureMethod?, Referance?)>

<!ATTLIST CanonicalizationMethod Algorithm CDATA #FIXED “http://www.w3c.org/TR/2001/REC-xml-c14n-20010315”>

<!ATTLIST SignatureMethod Algorithm CDATA #REQUIRED>

<!ATTLIST Referance URI CDATA #FIXED “xpointer(/Core)”>

<!ELEMENT SignedValue (#PCDATA)>

<!ATTLIST MsgBody xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/multi_com">

<!ELEMENT Leave (GroupID)>

<!ATTLIST Leave xmlns CDATA #FIXED “http://www.pucc.jp/2007/09/multi_com”>

<!ELEMENT GroupID (#PCDATA)>

]>

Appendix H: DTD for PUCC Multicast Service Protocol

A) DTD for JoinDeclare message

<?xml version="1.0" ?>

<!DOCTYPE Core [

<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, Destination, MsgBody)>

<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">

<!ELEMENT ComType (#PCDATA)>

<!ELEMENT MsgID (#PCDATA)>

<!ELEMENT MsgType (#PCDATA)>

<!ELEMENT CommunityID (#PCDATA)>

<!ELEMENT Source (#PCDATA)>

<!ELEMENT Target (#PCDATA)>

<!ELEMENT Destination (Target)>

<!ELEMENT MsgBody (JoinDeclare)>

<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/multi_srv">

<!ELEMENT JoinDeclare (GroupID)>

<!ATTLIST JoinDeclare xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/multi_srv">

<!ELEMENT GroupID (#PCDATA)>

]>

B) DTD for JoinDeclareResponse message

<?xml version="1.0" ?>

<!DOCTYPE Core [

<!ELEMENT Core (ComType, MsgID, ReplyID, MsgType, CommunityID?, Source, Destination, MsgBody)>

<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">

<!ELEMENT ComType (#PCDATA)>

<!ELEMENT MsgID (#PCDATA)>

<!ELEMENT ReplyID (#PCDATA)>

<!ELEMENT MsgType (#PCDATA)>

<!ELEMENT CommunityID (#PCDATA)>

<!ELEMENT Source (#PCDATA)>

<!ELEMENT Target (#PCDATA)>

<!ELEMENT Destination (Target)>

<!ELEMENT MsgBody (JoinDeclareResponse)>

<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/multi_srv">

<!ELEMENT JoinDeclareResponse (Result, Reason?, GroupID?, JoinDestination?)>

<!-- Reason element is required when Result value (#PCDATA) is Failure.

However, Reason element should be absent when Result value (#PCDATA) is Success. -->

<!-- AdjacentMember element is required when control node can provide adjacent member. However,

AdjacentMember element should be absent when control node can provide no adjacent member. -->

<!ATTLIST JoinDeclareResponse xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/multi_srv">

<!ELEMENT Result (#PCDATA)>

<!ELEMENT Reason (#PCDATA)>

<!ELEMENT GroupID (#PCDATA)>

<!ELEMENT JoinDestination %Route.class;>

]>

C) DTD for LeaveDeclare message

<?xml version="1.0" ?>

<!DOCTYPE Core [

<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, Destination, MsgBody)>

<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">

<!ELEMENT ComType (#PCDATA)>

<!ELEMENT MsgID (#PCDATA)>

<!ELEMENT MsgType (#PCDATA)>

<!ELEMENT CommunityID (#PCDATA)>

<!ELEMENT Source (#PCDATA)>

<!ELEMENT Target (#PCDATA)>

<!ELEMENT Destination (Target)>

<!ELEMENT MsgBody (LeaveDeclare)>

<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/multi_srv">

<!ELEMENT LeaveDeclare (GroupID)>

<!ATTLIST LeaveDeclare xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/multi_srv">

<!ELEMENT GroupID (#PCDATA)>

]>

D) DTD for MulticastServiceRequest message

<?xml version="1.0" ?>

<!DOCTYPE Core [

<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, Destination, MsgBody)>

<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">

<!ELEMENT ComType (#PCDATA)>

<!ELEMENT MsgID (#PCDATA)>

<!ELEMENT MsgType (#PCDATA)>

<!ELEMENT CommunityID (#PCDATA)>

<!ELEMENT Source (#PCDATA)>

<!ELEMENT Target (#PCDATA)>

<!ELEMENT Destination (Target)>

<!ELEMENT MsgBody (MulticastServiceRequest)>

<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/multi_srv">

<!ELEMENT MulticastServiceRequest (GroupID?, Service)>

<!-- GroupID element is required when type attribute value of Service element is AdjacentMember.

GroupID element should be absent when type attribute value of Service element is GroupList. -->

<!ATTLIST MulticastServiceRequest xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/multi_srv">

<!ELEMENT GroupID (#PCDATA)>

<!ELEMENT Service EMPTY>

<!ATTLIST Service type (“RoutingNode” | “GroupList”) #REQUIRED>

]>

E) DTD for MulticastServiceResponse message

<?xml version="1.0" ?>

<!DOCTYPE Core [

<!ELEMENT Core (ComType, MsgID, ReplyID, MsgType, CommunityID?, Source, Destination, MsgBody)>

<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">

<!ELEMENT ComType (#PCDATA)>

<!ELEMENT MsgID (#PCDATA)>

<!ELEMENT ReplyID (#PCDATA)>

<!ELEMENT MsgType (#PCDATA)>

<!ELEMENT CommunityID (#PCDATA)>

<!ELEMENT Source (#PCDATA)>

<!ELEMENT Target (#PCDATA)>

<!ELEMENT Destination (Target)>

<!ELEMENT MsgBody (MulticastServiceResponse)>

<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/multi_srv">

<!ELEMENT MulticastServiceResponse (GroupID?, Service)>

<!-- GroupID element is required when type attribute value of Service element is AdjacentMember.

GroupID element should be absent when type attribute value of Service element is GroupList. -->

<!ATTLIST MulticastServiceResponse xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/multi_srv">

<!ELEMENT GroupID (#PCDATA)>

<!ELEMENT Service (#PCDATA | JoinDestination | GroupList)?>

<!-- NodeID element is required when type attribute value of Service element is AdjacentMember.

GroupList element is required when type attribute value of Service element is GroupList. -->

<!-- A content of Service element is #PCDATA, that is "Failure", when no service is provided.

Therefore, as Service content is a mixed content, element type declaration must be described as

"(#PCDATA | XXX)*".

Therefore, "+" mark at NodeID element should be absent. -->

<!ATTLIST Service type (RoutingNode | GroupList) #REQUIRED>

<!ELEMENT JoinDestination %Route.class;>

<!ELEMENT GroupList (GroupInformation+)>

<!ELEMENT GroupInformation (GroupID, Description?, Number?, Application*)>

<!ATTLIST GroupInformation ID NMTOKEN #REQUIRED>

<!ELEMENT Description (#PCDATA)>

<!ELEMENT Number (#PCDATA)>

<!ELEMENT Application (#PCDATA)>

]>

F) DTD for MulticastServiceAdvertise message

<?xml version="1.0" ?>

<!DOCTYPE Core [

<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, Destination, MsgBody)>

<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">

<!ELEMENT ComType (#PCDATA)>

<!ELEMENT MsgID (#PCDATA)>

<!ELEMENT MsgType (#PCDATA)>

<!ELEMENT CommunityID (#PCDATA)>

<!ELEMENT Source (#PCDATA)>

<!ELEMENT Target (#PCDATA)>

<!ELEMENT Destination (Target)>

<!ELEMENT MsgBody (MulticastServiceAdvertise)>

<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/multi_srv">

<!ELEMENT MulticastServiceAdvertise (GroupID, Service)>

<!ATTLIST MulticastServiceAdvertise xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/multi_srv">

<!ELEMENT GroupID (#PCDATA)>

<!ELEMENT Service (NewRouting?, StaleRouting?)>

<!ATTLIST Service type (TreeImprovement) #REQUIRED>

<!ELEMENT NewRouting (JoinDirection)>

<!ELEMENT JoinDirection %Route.class;>

<!ELEMENT StaleRouting (LeaveDirection)>

<!ELEMENT LeaveDirection (#PCDATA)>

]>

Appendix I: DTD for PUCC Control Message Protocol

A) DTD for ErrorReport message

```
<?xml version="1.0" ?>
<!DOCTYPE Core [
<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, Destination, TraceRoute?, SessionID?, MsgBody)>
<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">
<!ELEMENT ComType (#PCDATA)>
<!ELEMENT MsgID (#PCDATA)>
<!ELEMENT MsgType (#PCDATA)>
<!ELEMENT CommunityID (#PCDATA)>
<!ELEMENT Source (#PCDATA)>
<!ELEMENT Target (#PCDATA)>
<!ENTITY % Route.class "(Route | Target)*" >
<!ELEMENT Route %Route.class;>
<!ATTLIST Route Node CDATA #REQUIRED>
<!ELEMENT Destination %Route.class;>
<!ELEMENT TraceRoute %Route.class;>
<!-- Route element is required within Destination element if the distance from Source node to Target node is
     more than 1 hop. However, Route element should be absent within Destination element if the distance
     from Source node to Target node is only 1 hop.
     Route element is always required within TraceRoute element. -->
<!ELEMENT SessionID (#PCDATA)>
<!ELEMENT Signature (SignedInfo?, SignedValue?)>
<!ATTLIST Signature xmlns CDATA #FIXED “http://www.w3c.org/2000/09/xmldsig#”>
<!ELEMENT SignedInfo (CanonicalizationMethod?, SignatureMethod?, Referance?)>
<!ATTLIST CanonicalizationMethod Algorithm CDATA #FIXED “http://www.w3c.org/TR/2001/REC-xml-c14n-20010315”>
<!ATTLIST SignatureMethod Algorithm CDATA #REQUIRED>
<!ATTLIST Referance URI CDATA #FIXED “xpointer(/Core)”>
<!ELEMENT SignedValue (#PCDATA)>
<!ELEMENT MsgBody (ErrorReport)>
<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/ctrl_msg">
<!ELEMENT ErrorReport (ErrorMsgID, Error)>
<!ATTLIST ErrorReport xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/ctrl_msg">
<!ELEMENT ErrorMsgID (#PCDATA)>
```

<!ELEMENT Error (#PCDATA)>

B) DTD for Diagnose message

<?xml version="1.0" ?>

<!DOCTYPE Core [

<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, Destination?, TraceRoute, HopCount?, GatewayAction?, SessionID?, Signature?, MsgBody)>

<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">

<!ELEMENT ComType (#PCDATA)>

<!ELEMENT MsgID (#PCDATA)>

<!ELEMENT MsgType (#PCDATA)>

<!ELEMENT CommunityID (#PCDATA)>

<!ELEMENT Source (#PCDATA)>

<!ELEMENT Target (#PCDATA)>

<!ENTITY % Route.class "(Route | Target)*" >

<!ELEMENT Route %Route.class;>

<!ATTLIST Route Node CDATA #REQUIRED>

<!ELEMENT Destination %Route.class;>

<!-- Destination element is required when ComType (#PCDATA) is Unicast or Multicast.

However, Destination element should be absent when ComType value (#PCDATA) is Broadcast. -->

<!ELEMENT TraceRoute %Route.class;>

<!-- Route element is required within Destination element if the distance from Source node to Target node is

more than 1 hop. However, Route element should be absent within Destination element if the distance

from Source node to Target node is only 1 hop.

Route element is always required within TraceRoute element. -->

<!ELEMENT HopCount (#PCDATA)>

<!-- HopCount element is required when ComType value (#PCDATA) is Broadcast and hop count is finite.

HopCount element should be absent when ComType value (#PCDATA) is Broadcast and hop count is infinite.

HopCount element always should be absent when ComType value (#PCDATA) is Unicast or Multicast. -->

<!ELEMENT GatewayAction (#PCDATA)>

<!ELEMENT SessionID (#PCDATA)>

<!ELEMENT Signature (SignedInfo?, SignedValue?)>

<!ATTLIST Signature xmlns CDATA #FIXED “http://www.w3c.org/2000/09/xmldsig#”>

<!ELEMENT SignedInfo (CanonicalizationMethod?, SignatureMethod?, Referance?)>

<!ATTLIST CanonicalizationMethod Algorithm CDATA #FIXED “http://www.w3c.org/TR/2001/REC-xml-c14n-20010315”>

<!ATTLIST SignatureMethod Algorithm CDATA #REQUIRED>

<!ATTLIST Referance URI CDATA #FIXED “xpointer(/Core)”>

<!ELEMENT SignedValue (#PCDATA)>

<!ELEMENT MsgBody (Diagnose)>

<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/ctrl_msg">

<!ELEMENT Diagnose (DiagnoseData?, DiagnoseDestination?)>

<!ATTLIST Diagnose xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/ctrl_msg">

<!ELEMENT DiagnoseData (#PCDATA)>

<!ELEMENT DiagnoseDestination (#PCDATA)>

<!ATTLIST DiagnoseDestination type

(”NodeID” | ”MulticastGroupID” | ”MulticastRoutingGroupID” | ”ApplicationID”) #REQUIRED>

]>

C) DTD for DiagnoseResponse message

<?xml version="1.0" ?>

<!DOCTYPE Core [

<!ELEMENT Core (ComType, MsgID, ReplyID, MsgType, CommunityID?, Source, Destination, TraceRoute, SessionID?, Signature?, MsgBody)>

<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">

<!ELEMENT ComType (#PCDATA)>

<!ELEMENT MsgID (#PCDATA)>

<!ELEMENT ReplyID (#PCDATA)>

<!ELEMENT MsgType (#PCDATA)>

<!ELEMENT CommunityID (#PCDATA)>

<!ELEMENT Source (#PCDATA)>

<!ELEMENT Target (#PCDATA)>

<!ENTITY % Route.class "(Route | Target)*" >

<!ELEMENT Route %Route.class;>

<!ATTLIST Route Node CDATA #REQUIRED>

<!ELEMENT Destination %Route.class;>

<!ELEMENT TraceRoute %Route.class;>

<!-- Route element is required within Destination element if the distance from Source node to Target node is

more than 1 hop. However, Route element should be absent within Destination element if the distance

from Source node to Target node is only 1 hop.

Route element is always required within TraceRoute element. -->

<!ELEMENT SessionID (#PCDATA)>

<!ELEMENT Signature (SignedInfo?, SignedValue?)>

<!ATTLIST Signature xmlns CDATA #FIXED “http://www.w3c.org/2000/09/xmldsig#”>

<!ELEMENT SignedInfo (CanonicalizationMethod?, SignatureMethod?, Referance?)>

<!ATTLIST CanonicalizationMethod Algorithm CDATA #FIXED “http://www.w3c.org/TR/2001/REC-xml-c14n-20010315”>

<!ATTLIST SignatureMethod Algorithm CDATA #REQUIRED>

<!ATTLIST Referance URI CDATA #FIXED “xpointer(/Core)”>

<!ELEMENT SignedValue (#PCDATA)>

<!ELEMENT MsgBody (DiagnoseResponse)>

<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/ctrl_msg">

<!ELEMENT DiagnoseResponse (DiagnoseData)>

<!ATTLIST DiagnoseResponse xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/ctrl_msg">

<!ELEMENT DiagnoseData (#PCDATA)>

]>

D) DTD for Lookfor message

<?xml version="1.0" ?>

<!DOCTYPE Core [

<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, MsgBody)>

<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">

<!ELEMENT ComType (#PCDATA)>

<!ELEMENT MsgID (#PCDATA)>

<!ELEMENT MsgType (#PCDATA)>

<!ELEMENT CommunityID (#PCDATA)>

<!ELEMENT Source (#PCDATA)>

<!ELEMENT MsgBody (Lookfor)>

<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/ctrl_msg">

<!ELEMENT Lookfor (Condition?)>

<!ATTLIST Lookfor xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/ctrl_msg">

<!ATTLIST Lookfor type (node | gateway | controlnode) “node”>

<!ATTLIST Lookfor count CDATA #IMPLIED>

<!ELEMENT Condition (#PCDATA)>

<!ATTLIST Condition protocol CDATA #REQUIRED>

]>

E) DTD for LookforResponse message

<?xml version="1.0" ?>

<!DOCTYPE Core [

<!ELEMENT Core (ComType, MsgID, ReplyID, MsgType, CommunityID?, Source, Destination, MsgBody)>

<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">

<!ELEMENT ComType (#PCDATA)>

<!ELEMENT MsgID (#PCDATA)>

<!ELEMENT ReplyID (#PCDATA)>

<!ELEMENT MsgType (#PCDATA)>

<!ELEMENT CommunityID (#PCDATA)>

<!ELEMENT Source (#PCDATA)>

<!ELEMENT Target (#PCDATA)>

<!ELEMENT Destination (Target)>

<!ELEMENT MsgBody (LookforResponse)>

<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/ctrl_msg">

<!ELEMENT LookforResponse (Description?, DeviceList?)>

<!ATTLIST LookforResponse xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/ctrl_msg">

<!ATTLIST LookforResponse type (node | gateway | controlnode) “node”>

<!ELEMENT Description (#PCDATA)>

<!ELEMENT DeviceList (Device+)>

<!ELEMENT Device (Specification?, StateVariableList?, ServiceList?, EventConditionList?, PrimitiveDeviceList?)>

<!ELEMENT Specification (URLBase?, Manufacturer?, ManufacturerURL?, ManufactureDate?, ModelDescription?, ModelName?, ModelNumber?, ModelURL?, SerialNumber?, UDN?, UPC?, IconList?)>

<!ELEMENT URLBase (#PCDATA)>

<!ELEMENT Manufacturer (#PCDATA)>

<!ELEMENT ManufacturerURL (#PCDATA)>

<!ELEMENT ManufactureDate (#PCDATA)>

<!ELEMENT ModelDescription (#PCDATA)>

<!ELEMENT ModelName (#PCDATA)>

<!ELEMENT ModelNumber (#PCDATA)>

<!ELEMENT ModelURL (#PCDATA)>

<!ELEMENT SerialNumber (#PCDATA)>

<!ELEMENT UDN (#PCDATA)>

<!ELEMENT UPC (#PCDATA)>

<!ELEMENT IconList (Icon+)>

<!ELEMENT Icon (Mimetype?, Width?, Height?, Depth?, Url?)>

<!ELEMENT Mimetype (#PCDATA)>

<!ELEMENT Width (#PCDATA)>

<!ELEMENT Height (#PCDATA)>

<!ELEMENT Depth (#PCDATA)>

<!ELEMENT Url (#PCDATA)>

<!ATTLIST Device type CDATA #REQUIRED>

<!ATTLIST Device id CDATA #IMPLIED>

<!ATTLIST Device name CDATA #IMPLIED>

<!ATTLIST Device private CDATA #IMPLIED>

]>

Appendix J: Namespace Definitions

Namespaces are defined with URI.

A) PUCC Core Protocol

http://www.pucc.jp/2012/03//core

xmlns attribute of Core element uses it.

B) PUCC Basic Communication Protocol

http://www.pucc.jp/2012/03//basic_com

xmlns attribute of MsgBody element uses it in PUCC Basic Communication Protocol.

C) PUCC Multicast Communication Protocol

http://www.pucc.jp/2012/03//multi_com

xmlns attribute of MsgBody element uses it in PUCC Multicast Communication Protocol.

D) PUCC Basic Service Protocol

http://www.pucc.jp/2012/03//basic_srv

xmlns attribute of MsgBody element uses it in PUCC Basic Service Protocol.

E) PUCC Multicast Service Protocol

http://www.pucc.jp/2012/03//multi_srv

xmlns attribute of MsgBody element uses it in PUCC Multicast Service Protocol.

F) PUCC Control Message Protocol

http://www.pucc.jp/2012/03//ctrl_msg

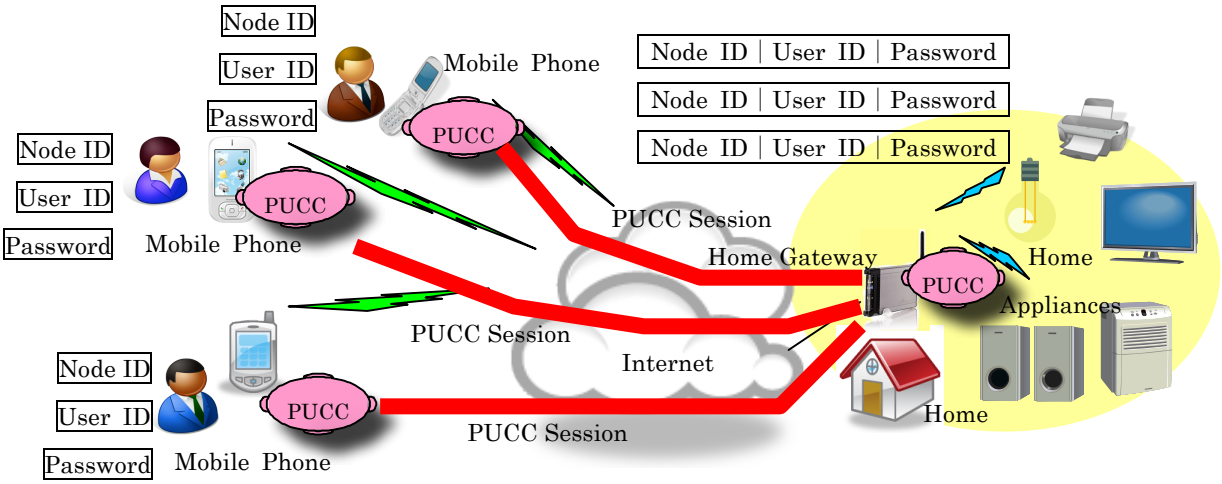
xmlns attribute of MsgBody element uses it in PUCC Control Message Protocol.

Appendix K: User Authentication, Message Encryption and Message Authentication.

This Appendix K specifies the user authentication method used when a PUCC session is set up and the message encryption and message authentication methods used in a PUCC session.

A) Use case

Users remotely control their home appliances from their mobile phones via the home gateway.
The home gateway allows only the users of the authenticated mobile phones to access their home appliances.



PUCC protocols enable remote control of the user's home appliances from the mobile phone. HTTP or TCP is used as a lower layer protocol. Users are connected to the home gateway via the internet from their mobile phones.

Only one PUCC node is operated in a mobile phone. The user of the PUCC node in the mobile phone shall be only one and unique.

Only one home gateway is installed per home. Only one PUCC node is operated in the home gateway. The application on the PUCC node in the home gateway controls home appliances.

The PUCC node on the home gateway authenticates all users of the PUCC nodes in all family members' mobile phones. For such authentication, the node ID of the PUCC node, the user ID and the password of each of the family members' mobile phones get registered in the home gateway in advance.

The PUCC node in the mobile phone initiates a PUCC session(with the user ID designated for the source user) toward the PUCC node in the home gateway. At this timing, authentication is performed. When the authentication is successful, a PUCC session is established. When the authentication is not successful, no PUCC session is established.)

Remote control of the user's home appliances is performed with the PUCC Device Discovery and Service Invocation Protocol upon encryption and signing of the PUCC message on the established PUCC session.

The PUCC node of the mobile phone terminates the PUCC session when remote control of the home appliances is completed. The PUCC node in the home gateway releases the PUCC session upon receipt of PUCC session release request or keep-alive monitoring.

B) Trust Model

(1) Trustworthyness

①One PUCC node shall be installed per mobile phone or per home getaway. A globally unique node ID shall be given to the PUCC node. How to assign and allocate the node ID is an implementation matter.

②A user ID unique to the PUCC node shall be allocated to the user of the PUCC node of the mobile phone. How to allocate the user ID is an implementation matter.

③))The user shall input the user ID and the password to the PUCC node of the mobile phone before a session is established. The user ID and the password input by the user shall not either be stored in the nonvolatile memory of the mobile phone (not be stored even in the memory for some applications) or be accessed or seen from anywhere other than the PUCC node. The length (the minimum number of characters) and the lifetime of the password are application dependent.

④The PUCC node of the mobile phone shall have acquired in some way the node ID of the PUCC node of the home gateway.

⑤A combination of the node ID, the user ID and the password of each of the family members' mobile phones shall have been registered in the home gateway in advance. The registered information mentioned above shall not be accessed or seen from anywhere other than the PUCC node in the home gateway.

⑥This specification document does not assume any of such cases where the mobile phone is lost or stolen, the home gateway is stolen, or the information stored in the mobile phone and the home gateway is improperly accessed. These cases are outside the scope of this specification document.

(2) Protection

①Protection against unauthorized access and spoofing

Passwords shall be protected from being decoded in the network from the mobile phone to the home gateway so as not to allow any mobile phone user who is not a member of the family to control home appliances of the family.

- Protect information which could be used by a third party to pretend to be a mobile phone user of the family.
- Protect information which could be used by a third party to pretend to be a mobile phone user of the family and to start authentication and eventually perform it successful.
- Prevent any third party from obtaining enough information to enable a brute force attack off-line and crack the password.

②Protection against spoofing

Protect home appliances from being controlled by a third party pretending the home gateway.

- Protect information which could be used to allow a third party to pretend to be the home gateway
- Protect information which could be used to allow a third party to pretend to be the home gateway and start authentication and eventually perform it successful
- Prevent any third party from obtaining enough information to enable a brute force attack off-line and crack the password.

③Protection of privacy

Home appliance control details and results and data on home appliances shall be protected from being accessed by an unauthorized third party in the network from the mobile phone to the home gateway

④Protection against malicious attacks

- Protect home appliances from any third party trying to take over their control in the network from the mobile phone to the home gateway.
- Protect PUCC protocol messages from being tampered in the network from the mobile phone to the home gateway.

C) User Authentication Mechanism

(1) Principles

①The authentication method shall be based on a challenge-response mechanism. The two PUCC nodes which establish a PUCC session between them shall have shared in advance the user ID, the node ID and the password. The password shall be long enough to be resistant to brute force attacks.

②The authentication shall be bidirectional. A unilateral authentication can be vulnerable to spoofing by a fraudulent party since what is required for authentication is just to return an arbitrary challenge.

③When the party who initiates communication is A and the other party is B, A and B represents a pair of node IDs and user IDs. A and B shall never have the same node ID. The user IDs of A and B may either be the same or different. The node ID shall be globally unique while the user ID shall be unique in the node. The pair of a node ID and a user ID, therefore, is globally unique.

④The party A shall be authenticated first so as to verify the identity of the party who initiates communication and prevent any of its replay attack. A sends a request for authentication which sets its node ID and user ID to B.

⑤The party B, which conducts authentication, generates a random number (a challenge) and sends it to the party A, which is going to be authenticated.)

⑥The authenticating party B generates a hash value based on the random number it sends to A and the password it holds. The algorithm used for hash value generation shall be stronger than SHA-256.

⑦The authenticated party A generates a hash value based on the random number it received and the password it holds and returns the hash value (a response) to B. The algorithm used for hash value generation shall be stronger than SHA-256.

⑧The authenticating party B compares the hash value it received with the hash value it generated. If the two values are identical, B concludes that A is successfully authenticated. If the two values are different, B concludes that A fails to be successfully authenticated. B sends the result of the authentication to A.

⑨Next, the party B undergoes authentication. The authenticating party A generates a hash number and sends it to B, which is going to be authenticated. This challenge is sent together with the response in ⑦.

⑩The authenticating party A generates a hash value based on the random number it sends to B and the password it holds. The algorithm used for hash value generation shall be stronger than SHA-256.

⑪)The authenticated party B generates a hash value based on the random number it received and the password it holds and returns the hash value (a response) to A. This response is sent together with the result of the authentication conducted in ⑧. The algorithm used for hash value generation shall be stronger than SHA-256.

⑫The authenticating party A compares the hash value it received with the hash value it generated. If the two values are identical, A concludes that B is successfully authenticated. If the two values are different, A concludes that B fails to be successfully authenticated. When the authentication result is found successful, the bidirectional authentication is completed and a PUCC session is established. When the authentication result is unsuccessful, a report is sent to B telling the authentication is unsuccessful. In the latter case, the communication is terminated without any PUCC session being established.

(2) Realization method

①The Hello method is repeated twice to complete user authentication. The first round of the Hello method starts with sending a request for authentication and ends with the phase in which the challenge of the authenticating party is received. The second round of the Hello method starts with sending both the response and the challenge of the authenticated party and ends with the phase in which the authentication result and the challenge of the authenticating party are received. The first and second rounds of the Hello method are linked to each other with Session IDs of the PUCC Core Protocol parameter.

②In the HelloResponse message reporting the result of the first Hello method, the parameter “Continue” is set as the Result parameter to indicate authentication is ongoing, When the authenticating side finds the result of the second Hello method is successful, “Success” is set as the Result parameter in the HelloResponse message. When the authenticating side finds the result of the second Hello method is unsuccessful, “Failure” is set as the Result parameter and “NotAuthenticated” as the Reason parameter in the HelloResponse message. When the authenticated side detects the success of the authentication, message encryption and signature are conducted in the established PUCC session to start a secure communication. When the authenticated side detects the failure of the authentication, the Bye message is immediately sent to release the PUCC session.

③The parameters necessary for message encryption and authentication of the PUCC session are negotiated in the second Hello method. The requested parameters are set in the second Hello message. Out of the requested parameters, only the parameters that can be supported by the responding side are set in the second HelloResopnse message.

The following is a sample of the first Hello message with session ID and user ID for user authentication.

```
<Core xmlns=" Namespace of PUCC 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>
  <SessionID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</SessionID>
  <MsgBody protocol=" Namespace of PUCC Basic Communication Protocol" >
    <Hello xmlns=" Namespace of PUCC Basic Communication Protocol" >
      <InitiatorUserID>urn:pucc:user:A</InitiatorUserID>
      <ResponderUserID>urn:pucc:user:B</ResponderUserID>
      <Authentication/>
    </Hello>
  </MsgBody>
</Core>
```

Figure 116. Sample of the first Hello message with user ID and session ID for user authentication

The following is a sample of the first Hello message with session ID and user ID.

```
<Core xmlns=" Namespace of PUCC 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>
  <SessionID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</SessionID>
  <MsgBody protocol=" Namespace of PUCC Basic Communication Protocol" >
    <Hello xmlns=" Namespace of PUCC Basic Communication Protocol" >
```

```
      <InitiatorUserID>urn:pucc:user:A</InitiatorUserID>
      <ResponderUserID>urn:pucc:user:B</ResponderUserID>
      <Authentication>
        <Response>784f8ba630c542hc</Response>
        <Challenge>39076daa78cb42ba</Challenge>
        <RequestedHashAlgorithm>SHA-256</RequestedHashAlgorithm>
      </Authentication>
    </Hello>
  </MsgBody>
</Core>
```

Figure 117. Sample of the second Hello message with user ID and session ID for user authentication

The following is a sample of the first HelloResponse message for user authentication.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <MsgType>Response</MsgType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <ReplyID>54321.2002-12-20T16:17:32Z@968749ab-f9bb-2647-7459-f66e56f11234</ReplyID>
  <Destination>
    <Target>968749ab-f9bb-2647-7459-f66e56f11234</Target>
  </Destination>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <ComType>Unicast</ComType>
  <SessionID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</SessionID>
  <MsgBody protocol=" Namespace of PUCC Basic Communication Protocol" >
    <HelloResponse xmlns=" Namespace of PUCC Basic Communication Protocol" >
      <Result>Continue</Result>
      <Authentication>
        <Challenge>348bff097c54a3109</Challenge>
        <RequestedHashAlgorithm>SHA-256</RequestedHashAlgorithm>
      </Authentication>
    </HelloResponse>
  </MsgBody>
</Core>
```

Figure 118. Sample of the first HelloResponse message for user authentication

The following is a sample of the second HelloResponse message when user authentication is successful.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <MsgType>Response</MsgType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <ReplyID>54321.2002-12-20T16:17:32Z@968749ab-f9bb-2647-7459-f66e56f11234</ReplyID>
  <Destination>
    <Target>968749ab-f9bb-2647-7459-f66e56f11234</Target>
  </Destination>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <ComType>Unicast</ComType>
  <SessionID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</SessionID>
  <MsgBody protocol=" Namespace of PUCC Basic Communication Protocol" >
    <HelloResponse xmlns=" Namespace of PUCC Basic Communication Protocol" >
      <Result>Success</Result>
      <Authentication>
        <Response>348bff097c54a3109</Response>
```

```
      </Authentication>
    </HelloResponse>
  </MsgBody>
</Core>
```

Figure 119. Sample of the second HelloResponse message when user authentication is successful

The following is a sample of the second HelloResponse message when user authentication.is failed.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <MsgType>Response</MsgType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <ReplyID>54321.2002-12-20T16:17:32Z@968749ab-f9bb-2647-7459-f66e56f11234</ReplyID>
  <Destination>
    <Target>968749ab-f9bb-2647-7459-f66e56f11234</Target>
  </Destination>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <ComType>Unicast</ComType>
  <SessionID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</SessionID>
  <MsgBody protocol=" Namespace of PUCC Basic Communication Protocol" >
    <HelloResponse xmlns=" Namespace of PUCC Basic Communication Protocol" >
      <Result>Failure</Result>
      <Reason>NotAuthenticated</Reason>
    </HelloResponse>
  </MsgBody>
</Core>
```

Figure 120. Sample of the second HelloResponse message when user authentication is failed

The following is a sample of sequences of Hello method.

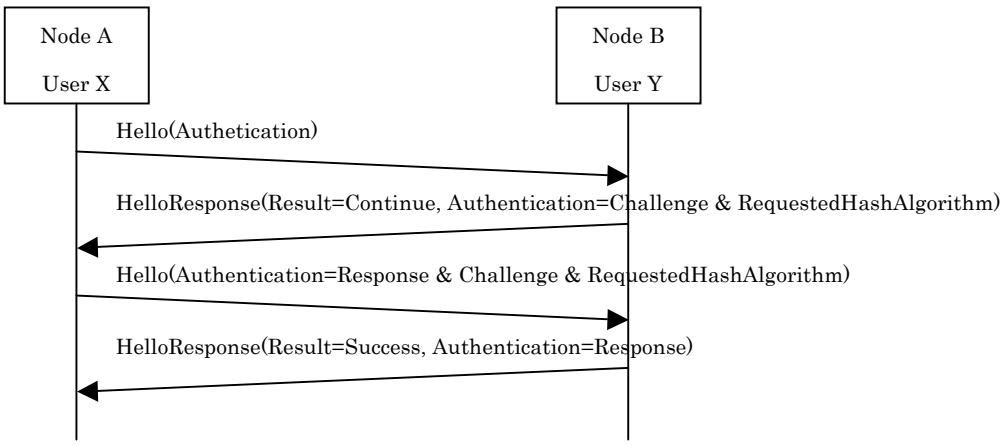


Figure 121. Sequence of Hello method for user authentication when user authentication is successful

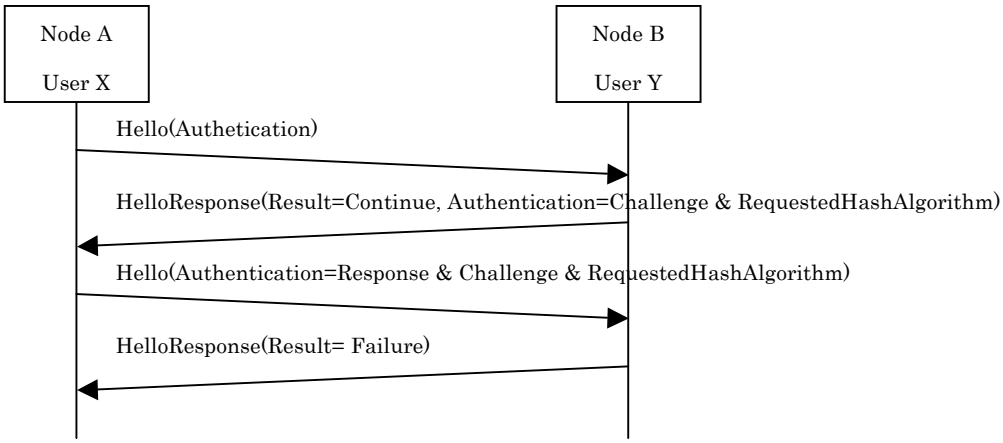


Figure 7. Sequence of Hello method for user authentication when initiator authentication is failed

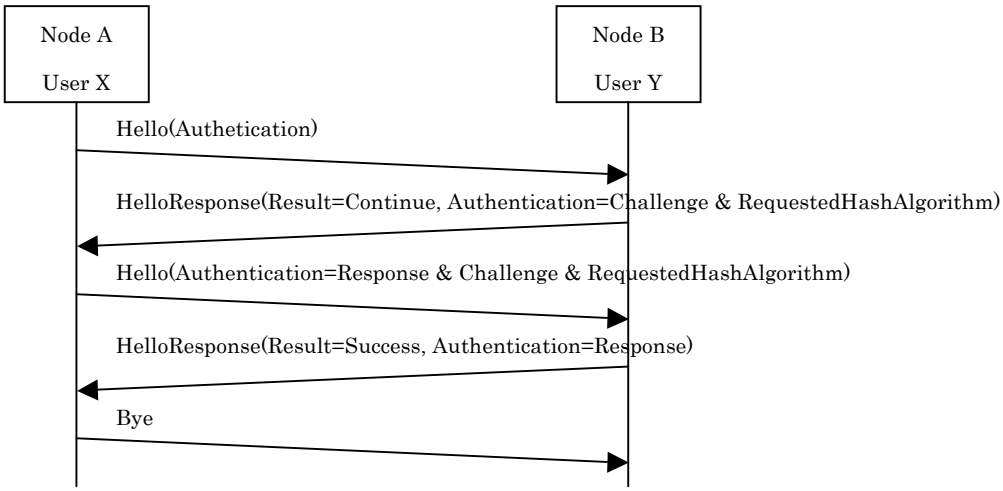


Figure 8. Sequence of Hello method for user authentication when responder authentication is failed

D) Message Encryption Mechanism

(1) Principles

①)A public key shall be used for message encryption and decryption in consideration of a situation where any PKI (Public key Infrastructure) cannot be used.

②)))A Password-Based Encryption (PBE) method as specified in PKCS#5 (RFC2898) shall be used to generate public keys. A public key (PBE key) shall be generated based on the pass word and the count of unidirectional hash function iterations.

③ A session key shall be used for encryption and decryption of actual data to be processed. A session key shall be generated by a pseudo-random number generator. A session key shall be prepared for each direction of PUCC message transmission between the two communicating nodes.

④))The session key shall be encrypted with the public key (PBE key) and be sent from one node to the other node it is communicating to together with the encrypted data. The other node decrypts the session key with the public key (PBE key) and then decrypts the encrypted data sent together with the session key.

⑤ Encryption shall be applied to the entire MsgBody element in the PUCC message. After the MsgBody element is encrypted, it shall be deleted from the PUCC message. In place of the MsgBody element, the Encrypted Data element specified by the W3C XML Encryption scheme is inserted into the PUCC message. The session key ciphered with a password-based encryption shall be also set in the EncryptedData element.

(2) Public key generation by PBE

① The hash value shall be generated from the password used in the User Authentication Mechanism and the salt generated by the pseudo-random number generator. The algorithm used for hash value generation shall be stronger than SHA-256.

② Another hash value shall be calculated from the hash value generated from the password in ①. The algorithm used for hash generation shall be stronger than SHA-256. This hash generation process shall be repeated by the number of times specified.

The hash value obtained in ② shall be used as the public key (PBE key).)

(3) Encryption algorithm

① A block encryption algorithm shall be used. The algorithm to be used shall be chosen from the 6 types of algorithms shown below.
- AES-128
- AES-192
- AES-256
- Camellia-128
- Camellia-192
- Camellia-256

② The CBC mode shall be used as a block repetition method. The IV (Initialization Vector) in the CBC mode shall set a different value each time encryption is conducted.)

③)The method used for padding the last encryption block shall be the one specified in PKCS#7(RFC2315).

(4) How to store encrypted data

① As the data to be encrypted is described in XML, the encrypted data shall be stored in accordance with W3C XML Encryption scheme.

② Out of the elements specified in the W3C XML Encryption scheme, the following elements shall be used.

Element Value	Description
EncryptedData	Sets encrypted data. Includes Encryption Method element, ds: KeyInfo element, and CipherData element.
EncryptionMethod	Sets encryption algorithm
ds:KeyInfo	Sets encrypted session key. Includes Encryptedkey element.
EncryptedKey	Sets encrypted session key. Includes CipherData element.
CipherData	Sets encrypted session key and encrypted actual data. Includes CipherValue element.
CipherValue	Sets values of encrypted session key and encrypted actual data which are encoded in the BASE64 format.

③ Encryption shall be applied to the entire MsgBody element in the PUCC message. After the MsgBody is encrypted, it shall be deleted from the PUCC message. In place of the MsgBody element, the Encrypted Data element specified by the W3C XML Encryption scheme is inserted into the PUCC message. Samples of the PUCC message before and after encryption are shown below.

Sample of the PUCC message before encryption

```
<Core xmlns=" Namespace of PUCC 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>
  <SessionID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</SessionID>
  <MsgBody protocol=" Namespace of PUCC Control Message Protocol" >
    <Diagnose xmlns=" Namespace of PUCC Control Mesage Protocol" >
      <DiagnoseData>104353920000</DiagnoseData>
      <DiagnoseDestination              type=              "              NodeID              "
>968985ab-e6aa-5842-1234-f98e56f15687</DiagnoseDestination>
    </Diagnose>
  </MsgBody>
</Core>
```

Sample of the PUCC message after encryption

```
<Core xmlns=" Namespace of PUCC 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>
  <SessionID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</SessionID>
  <EncryptedData Type=  "http://www.w3.org/2001/04/xmlenc#Element"
    xmlns=  "http://www.w3.org/2001/04/xmlenc#" >
    <EncryptedMethod Algorithm= "AES-256" />
    <ds:KeyInfo xmlns:ds= "http://www.w3.org/2000/09/xmldsig#" >
      <EncryptedKey>
        <CipherData>
          <CipherValue>r7us902Ws</CipherValue>
        </CipherData>
```

```
      </EncryptedKey>
    </ds:KeyInfo>
    <CipherData>
      <CipherValue>A23B456C56XuysjfhShskeSplaedSarkfbseSkrgksANje49Ud9Js2</CipherValue>
    </CipherData>
  </EncryptedData>
</Core>
```

(5) Session key update

① The session key is updated after the session key update count message negotiated at the session establishment is completely encrypted.

E) Message Authentication Mechanism

(1) Principles

① A message authentication code shall be given to the PUCC message in order to guarantee the integrity of the message and the legitimacy of the message sender.

②)The message authentication code shall be the hash value calculated based on the password used in the User Authentication Mechanism and the transmitted (received) message. The algorithm used for hash value generation shall be stronger than SHA-256.

③ The message authentication code shall be applied to the all elements except those variable between the sending and receiving nodes.

Elements in Core Element	Message Authentication
ComType	Yes
MsgID	Yes
ReplyID	Yes
MsgType	Yes
CommunityID	Yes
Source	Yes
Destination	No
TraceRoute	No
HopCount	No
GatewayAction	No
SessionID	Yes
MsgBody	Yes
EncryptedData	Yes

)In the node to which the message authentication is applied, the whole part in the PUCC message except those specified otherwise as shown above shall undergo canonicalization and signature processing in accordance with the digital signature setting procedure of the W3C XML Signature scheme.
The receiving node which verifies the message authentication code and the message verifies the whole part excluding the elements shown above in accordance with the digital signature verification procedure of the W3C XML Signature scheme.

(2) How to store message authentication code

① As the message to be authenticated is described in XML, the message authentication code shall be stored in accordance with W3C XML Signature scheme.

② Out of the elements specified in the W3C XML Signature scheme, the following elements shall be used.

Element Value	Description
Signature	Sets message authentication code. Includes SignedInfo element and SignatureValue element.
SignedInfo	Sets signed information.
CanonicalizationMethod	Sets canonicalization method.。
SignatureMethod	Sets method to generate message authentication code.
SignatureValue	Sets value of message authentication code encoded in BASE64 format.

③ The Signature element specified in the W3C XML Signature scheme shall be included in the Core element of the PUCC message in order to embed the message authentication code into the PUCC message. Samples of the PUCC message before and after the message authentication code is shown below.

Sample of the PUCC message before the message authentication code is set

```
<Core xmlns=" Namespace of PUCC 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>
  <SessionID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</SessionID>
  <MsgBody protocol=" Namespace of PUCC Control Message Protocol" >
    <Diagnose xmlns=" Namespace of PUCC Control Mesage Protocol" >
      <DiagnoseData>104353920000</DiagnoseData>
      <DiagnoseDestination              type=              "              NodeID              "
>968985ab-e6aa-5842-1234-f98e56f15687</DiagnoseDestination>
    </Diagnose>
  </MsgBody>
</Core>
```

Sample of the PUCC message after the message authentication code is set

```
<Core xmlns=" Namespace of PUCC 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>
  <SessionID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</SessionID>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#" >
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
      <SignatureMethod  Algorithm="" />
      <Reference URI="xpointer(/Core)" ></Reference>
    </SignedInfo>
    <SignatureValue>NK8A3AS5USH874GH</SignatureValue>
  </Signature>
  <MsgBody protocol=" Namespace of PUCC Control Message Protocol" >
    <Diagnose xmlns=" Namespace of PUCC Control Mesage Protocol" >
      <DiagnoseData>104353920000</DiagnoseData>
      <DiagnoseDestination              type=              "              NodeID              "
>968985ab-e6aa-5842-1234-f98e56f15687</DiagnoseDestination>
    </Diagnose>
  </MsgBody>
</Core>
```

F) DTD for Encrypted PUCC message

```
<?xml version="1.0"?>
<!DOCTYPE Core [
<!ELEMENT Core (ComType, MsgID, ReplyID?, MsgType, CommunityID?, Source, Destination, TraceRoute?, HopCount?, GatewayAction?, SessionID, Signature?, EncriptedData)>
<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">
<!ELEMENT ComType (#PCDATA)>
<!ELEMENT MsgID (#PCDATA)>
<!ELEMENT ReplyID (#PCDATA)>
<!ELEMENT MsgType (#PCDATA)>
<!ELEMENT CommunityID (#PCDATA)>
<!ELEMENT Source (#PCDATA)>
<!ELEMENT Target (#PCDATA)>
<!ENTITY % Route.class "(Route? | Target)" >
<!ELEMENT Route %Route.class;>
<!ATTLIST Route Node CDATA #REQUIRED>
<!ELEMENT Destination %Route.class;>
<!ELEMENT TraceRoute %Route.class;>
<!ELEMENT HopCount (#PCDATA)>
<!ELEMENT GatewayAction (#PCDATA)>
<!ELEMENT SessionID (#PCDATA)>
<!ELEMENT EncriptedData (EncryptedMethod?, ds:KeyInfo?, CipherData?)>
<!ATTLIST EncriptedData Type CDATA #FIXED "http://www.w3.org/2001/04/xmlenc#Element">
<!ATTLIST EncriptedData xmlns CDATA #FIXED "http://www.w3.org/2001/04/xmlenc#">
<!ATTLIST EncryptedMethod Algorithm CDATA #REQUIRED>
<!ELEMENT ds:KeyInfo (EncryptedKey?)>
<!ATTLIST ds:KeyInfo xmlns:ds CDATA #FIXED "http://www.w3.org/2000/09/xmldsig#">
<!ELEMENT EncryptedKey (CipherData?)>
<!ELEMENT CipherData (CipherValue?)>
<!ELEMENT CipherValue (#PCDATA)>
]>
```