PUCC Architecture

(Version 2.0 - 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, 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 standards. However, focusing on each technology, there are still some problems for users such as captive of venders, no interoperability and so on. In these surroundings we expect that the technologies which enable these equipments to connect to global networks, to be seamlessly utilized from various locations, are strongly desired with the widespread use of various digital equipments such as mobile phones, digital cameras, printers and digital TVs.

The goals of PUCC are to research, develop and deploy just such enabling technologies. For bring about breakthrough, PUCC aim to research and develop computing environment which provide interoperability of various networks and advanced services using the peer-to-peer communication technology without specific operations of users.

This PUCC architecture is intended to establish the system and protocol architecture for the peer-to-peer applications for the heterogeneous network environment which consists of various networks mentioned above. It should become the starting point for understanding the technologies of Peer-to-Peer applications for the heterogeneous network environment.

This specification provides an overview of the PUCC peer-to-peer architecture for the heterogeneous network environment which includes home appliances, printers and mobile phones. Details of related technologies are provided in the appropriate specification. It itself provides a framework for a variety of protocols, features, and services. It does not mandate any specific implementation and shall be considered informative.

2. Terminology

2.1. Definitions

– Peer-to-peer node – A peer-to-peer node is an independent communication entity on the Peer-to-Peer network. In the Peer-to-Peer architecture, it can be a mobile device, PDA, personal computer, server, sensor, home appliance, and so on;

– Control Node – A control node is an administration entity on the Peer-to-Peer network. Its functions include: peer node authentication, first peer discovery, peer-to-peer network topology optimization, route calculation and optimization, and peer community/group management, and network recovery;

– Gateway Node – A gateway node is a connection entity linking a pure peer-to-peer network to a hybrid peer-to-peer network;

– Hybrid peer-to-peer Network – A peer-to-peer network that consists of peer-to-peer nodes and control nodes;

– Pure peer-to-peer Network – A peer-to-peer network that consists of only peer-to-peer nodes;

– Network Community – A network community is a set of peer-to-peer nodes that have a common set of interests and that obey a common set of policies;

– Role – A role is a functional entity that is realized by node acting with regard to a specific Peer-to-Peer application. A role can be realized by using one or more node functions, and what role a node acts is dependent on the Peer-to-Peer application. A node can perform multiple roles on the network, and each role can be changed dynamically;

– Proxy – Theoretically, all capacity-aware devices can be independent nodes in the Peer-to-Peer architecture. Currently, some capacity-aware devices can't act as fully independent nodes due to their functional limitations such as limited CPU power and limited memory. A proxy is a function in a node that provides agent service for those devices with constrained capability and leads them to peer-to-peer network;

– Resource – In this architecture, a resource is defined as node, node attribute, node application function, node communication function and etc;

– Node ID – Node ID is a unique identifier for a node that distinguishes it from all other nodes on the network. For example, the Node ID is defined as FQDN (Fully Qualified Domain Name) or UUID (Universal Unique Identifier);

– Community – A community is a set of nodes that have a common set of interests or that obey a common set of policies. Each community is uniquely identified by a unique Community ID. Nodes belong to the same community can communicate one another;

– Community ID – Community ID is a unique identifier for a community that distinguishes it from all other communities. For example, the community id is defined as URN with specific namespace;

– Multicast Group – A multicast group is a collection of nodes under a defined multicast Group;

– Multicast Group ID – Multicast group ID is a unique identifier for a multicast group that distinguishes it from all other node groups on the network. For example, the Node Group ID is defined as URN with specific namespace;

– Message – The basic unit of P2P communication; consists of an XML-structured sequence of octets and transmitted via a connection;

– Message ID – Message ID is a unique identifier for a message that distinguishes it from all other messages. For example, the message ID can be defined as, node provided sequence number + current time + node ID;

2.2. Abbreviations

HTTP	Hypertext Transfer Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
IP	Internet Protocol
SOAP	Simple Object Access Protocol
XML	eXtensible Markup Language
CGI	Common Gateway Interface
iAppli	i-mode Java Application

3. References

- 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.

- P. Leach and R. Salz, “A Universally Unique Identifier (UUID) URN Namespace,” RFC 4122, September 2005

- Dijkstra, E. W. "A Note on Two Problems in Connection with Graphs." Numerische Math. 1, 269-271, 1959.

- Energy Conservation and Homecare NETwork,”ECHONET”, URL: http://www.echonet.gr.jp

4. Goals and Requirements

4.1. Goals

The goals of this specification are:

- ◆ Define a general, scalable, extensible Peer-to-Peer architecture for various devices such as mobile phone, home appliance and digital equipments. On the defined architecture, a wide range of heterogeneous network applications are expected to be implemented smoothly that can full utilize the power of peer-to-peer communication.
- ◆ Leverage existing standards where possible, especially existing and evolving Internet, home network, mobile network standards.

4.2. Requirements

4.2.1. Simplicity

Simplicity is a key element in making the Peer-to-Peer architecture easy to understand, implement, maintain, and evolve. The Peer-to-Peer architecture should be simple.

4.2.2. Scalability

The Peer-to-Peer architecture should be scalable so it can be applied to a large number of nodes, say a million or more devices.

4.2.3. Generality

The Peer-to-Peer architecture should be general so that it can be used to realize various Peer-to-Peer applications. Peer-to-Peer applications may include but are not limited to: Home appliance control, P2P streaming and direct printing.

4.2.4. Connectivity

The Peer-to-peer architecture should have the ability to keep the connectivity of peer-to-peer network.

4.2.5. Network Recovery

Because the peer-to-peer environment is a network community that peer-to-peer nodes can join or leave freely, it is possible that the entire peer-to-peer network could be split by network problems or machine troubles. To ensure every peer-to-peer node can be provided with access, the peer-to-peer architecture should have the ability to recovery the peer-to-peer network when it is split by unforeseen problems.

4.2.6. Mobility

Peer-to-peer nodes can join the peer-to-peer network independent of where they are and the physical network currently being used, such as general IP network or ad-hoc network.

4.2.7. Layered architecture

For providing a scalable and extensible application environment for each node on the architecture, layered architectures are essential in the nodes themselves and in the communication protocols between nodes. The architecture components should be defined independently.

4.2.8. Deployment

The Peer-to-Peer architecture should be designed to work in a wireless environment, and work regardless of device capability.

4.2.9. Small devices support

Because of the fundamental limitations of small devices, the proxy function for small devices is essential in the Peer-to-Peer architecture for overcoming the problems associated with device limitations such as limited CPU power, limited memory, limited battery life, restricted communication protocol support, and restricted MMI. The proxy function support may include but is not limited to: Content conversion, Protocol conversion, Information storage, Presentation management, User privacy management, and User preference management

4.2.10. Peer-to-Peer node autonomy

In the Peer-to-Peer architecture, each Peer-to-Peer node should behave as an autonomous system. To this end, the following functions should be considered when designing the Peer-to-Peer architecture: First peer-to-peer node discovery, Local topology optimization, Network Recovery, Node resource discovery, Node resource exchange, Connection creation between node and node, Connection release between node and node, Application information exchange between node and node,

4.2.11. Efficiency

The performance of Peer-to-Peer applications should be optimized, for example, response time for resource discovery should be minimized. To this end, various technologies such as network topology optimization, route optimization, resource sharing, and resource caching should be considered when designing the Peer-to-Peer architecture.

4.2.12. Extensibility

The Peer-to-Peer architecture should be extensible to satisfy future Peer-to-Peer applications requirements.

4.2.13. Co-existence with various networks

The Peer-to-Peer architecture should be realized over the existing Internet. To this end, existing components of the Internet such as i-mode, firewalls, and NAT should be considered and utilized, when designing the Peer-to-Peer architecture. In addition, it should be realized on not only IP networks, but also on Non-IP networks (Bluetooth etc).

4.2.14. Privacy and Preference

User privacy and user preference should be safeguarded to the same extent as in the fixed Internet.

4.2.15. Security

Security service is required by the Peer-to-Peer architecture. It may include but is not be limited to:

Peer-to-peer Node Authentication: Peer-to-peer node authentication will be done when a peer-to-peer node joins the network.

Authentication between peer-to-peer nodes: A peer-to-peer node identify will be checked when it wants to establish a connection to another node.

4.2.16. Independence of specific hardware and software

In order to realize the Peer-to-Peer architecture on different types of hardware (e.g. mobile device) and software (e.g. OS), the Peer-to-Peer architecture should be independent of device, OS and networks.

4.2.17. Convergence with existing standards

When designing the Peer-to-Peer architecture, the use of existing standards should be considered where applicable. Such standards may include but are not be limited to: W3C standards such as RDF, XML Protocol, SOAP, and Web services, and IETF standards such as HTTP, TCP.

4.2.18. Co-existence with billing, digital rights management (DRM) systems

The design of the Peer-to-Peer architecture should not impede the implementation of the functions needed to realize billing systems and digital rights management systems.

4.2.19. Node Naming

Node naming should be considered when designing the peer-to-peer architecture. The requirements of node naming should include but are not limited to:

Uniqueness: Node name should be unique.

Manageability: Node name should be easy to manage.

Accessibility: Node name should be easy to realize with current existing technology

Simplicity: Node name should be easy to understand

Scalability: Node name should be scalable to millions of peer nodes

Anonymity and Privacy: Node naming should be independent of well-known Internet information (IP address, domain name, etc) to protect user's privacy.

Security: Node authentication should be considered

Independence: Node naming should be independent of physical network environment and protocols

5. Architecture

This section provides an overview of the peer-to-peer architecture for the heterogeneous networks.

5.1. Overview

Our Peer-to-Peer network is a virtual network realized on the top of the current physical network infrastructure. It is designed simplicity and light, but it retains the interesting and powerful primitives needed by the services and applications. Furthermore, leading small devices into peer-to-peer networks and seamlessly combining the pure peer-to-peer network with hybrid peer-to-peer networks are also considered. (Figure 1) This architecture allows not only wired devices but also mobile phones, PDA, home appliance and sensor to share the advantages of the peer-to-peer network, regardless of their physical network. This architecture tries to hide all of the complexity of the underlying physical network topologies and provide a uniformly addressable network to all peer-to-peer nodes

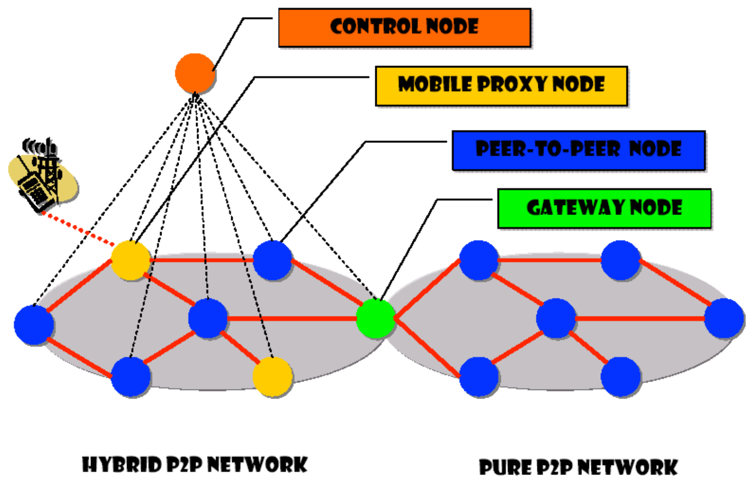


Figure 1. Architecture of the Peer-to-Peer network for the mobile Internet

Two kinds of network environment are defined in our architecture. One is the pure peer-to-peer network and another is the hybrid peer-to-peer network.

5.2. Pure Peer-to-Peer Network

There are only peer-to-peer nodes in the pure peer-to-peer network. The connections between peer-to-peer nodes are based on mutual trust. (How to support this trust lies outside the scope of this specification.) Each peer-to-peer node is an independent entity and can join and leave the network at its convenience. Messages are sent from one peer-to-peer node to another one directly or via some intermediary peer-to-peer nodes. Route information is retrieved by broadcasting an inquire message to the network. In the pure peer-to-peer network, the peer-to-peer is a completely independent entity without any constraints.

5.3. Hybrid Peer-to-Peer Network

In the pure peer-to-peer network, each node is completely independent; this freedom is offset by several distinct drawbacks. For an instance, due to node independence, which means no system management, the node must carry the full cost of broadcasting many messages over the network for retrieving route information and exchanging messages. In order to reduce network traffic, hop counts can be used to restrict the range of the broadcast messages but this causes another problem: the messages may not reach the end point of the pure peer-to-peer network if there are numerous peer-to-peer nodes.

These disadvantages are eliminated by the hybrid peer-to-peer network since it provides the control node. The control node provides network services to the peer-to-peer nodes such as optimized route information and group node management. When a peer-to-peer node sends a message to another node, it can acquire the route information from the control node. In this way, network traffic can be reduces by a large margin and peer-to-peer nodes can focus on effective content searching or other functions. Other functions (first peer discovery, node authentication, network recovery and network topology optimization) are also provided by the control node, and so make the hybrid peer-to-peer network robust, extensible, and secure.

5.4. Inter-working of Hybrid Peer-to-Peer Network and Pure Peer-to-Peer Network

In our architecture, the hybrid peer-to-peer network and the pure peer-to-peer network are connected seamlessly by the gateway node, which acts as a connect entity and has the relay functions needed for forwarding the messages between the two networks. From the viewpoint of the hybrid peer-to-peer network, the gateway node acts as a hybrid peer-to-peer node. It acts as a pure peer-to-peer node for the pure peer-to-peer network.

5.5. Proxy node in the Peer-to-Peer Network

Another distinct feature of this architecture is that it provides the possibility of leading small devices to the P2P network if a proxy node is added. Though small devices, such as mobile phones, home appliance, and sensor, will have enhanced performance in the future, they will still suffer the following problems:

-Limited storage capacity

-Short runtime

-Very limited network bandwidth and high latency

-Modest processor performance

-Constrained electrical power budgets

Thus small devices can’t assume the role of more sophisticated peer nodes that offer services to other peer nodes in the P2P network. Some peer-to-peer nodes will have to provide support functions to the small devices. These nodes are called proxy nodes in our architecture.

In cooperation with the proxy nodes, a small device can act as a general peer-to-peer node and provide the general node functions of the peer-to-peer architecture. From the viewpoint of peer-to-peer architecture, proxy nodes can be divided into three types.

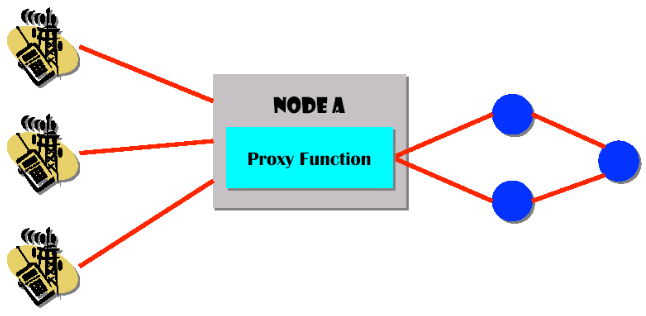


Figure 2. Type 1 proxy node in peer-to-peer architecture

In Figure 2, mobile devices (Mobile A, B, C) connect to a certain peer-to-peer node that offers the proxy function. In this case, all three mobile devices and the proxy node form a peer-to-peer node and share one Node ID (Node A) in the peer-to-peer network. To other peer-to-peer nodes, messages to these mobile devices will always have the same destination node ID and the messages to each mobile device have to be processed at the application layer by the proxy node.

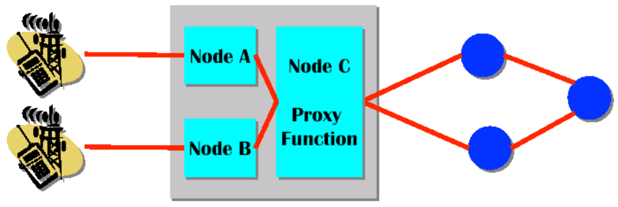


Figure 3. Type 2 proxy node in P2P architecture

In Figure 3, each mobile device has its own Node ID and acts as a separate peer-to-peer node in the peer-to-peer architecture. In this case, virtual peer-to-peer nodes are built by the peer-to-peer node and equipped with proxy functions. Each virtual node corresponds to a mobile device and is assigned a separate node ID. Messages to these mobile devices will pass the intermediary nodes and reach the mobile devices directly without any intermediate processing, like general communication between peer-to-peer nodes. Notice that, in this type, every message to the mobile device has to pass through a certain proxy node (node C).

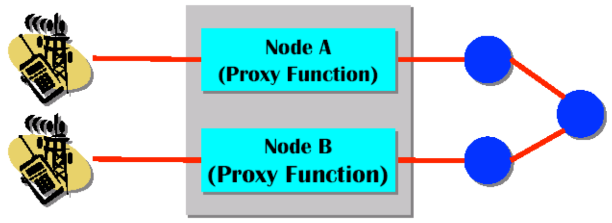


Figure 4. Type 3 proxy node in peer-to-peer architecture

In Figure 4, a mobile device pairs with a virtual proxy function model to construct a peer-to-peer node in the peer-to-peer architecture. In this case, both the proxy model and mobile device share the same node ID and achieve the same actions together.

We don't mandate which type should be used in the peer-to-peer architecture; it depends on the applications.

5.6. Naming

5.6.1. Node ID

A peer node ID is an endpoint for communication, which can be a computer, a user, a mobile device, or anything else that you want to resolve. Peer node IDs can be registered as unsecured or secured. Unsecured names are just text strings that are subject to spoofing, as anyone can register a duplicate unsecured name. Unsecured names are best used in private or otherwise secure networks. Secured names can only be registered by the owner and they are protected with a certificate and a digital signature. This architecture does not dictate when, where, or who assigns this unique Node ID and how it is created. This lies outside the scope of this specification and depends on the applications.

5.6.2. Multicast Group ID

Multicast Group ID is used for sending a message to a group of peer-to-peer nodes. Multicast Group ID is a unique ID to distinguish its messages from others. Peer-to-peer nodes can join and leave their favorite groups dynamically. A peer-to-peer node may join several groups as needed. This architecture does not dictate when, where, or who assigns this unique multicast Group ID and how it is created. This lies outside the scope of this specification and depends on the applications.

5.6.3. Community ID

A community is a set of peer-to-peer nodes that have a common set of interests and that obey a common set of policies. Community ID is a unique identifier to distinguish it from other communities. This specification also doesn't dictate when, where, or who creates the group name; it depends on the applications.

5.6.4. User ID

User ID is a unique identifier to distinguish a user on a peer-to-peer node. User ID is defined as a user friendly name written in text strings. User ID is used only on the nodes that belong to the leaf of the peer-to-peer network. This specification also doesn't dictate when, where, or who creates the user ID. This lies outside the scope of this specification and depends on the applications.

.

5.6.5. Session ID

A session is a virtual connection between two nodes on a peer-to-peer network that allows several users to exchange data between two nodes. Session ID is a unique identifier to distinguish to the session on each node. Session ID is set, when several users establish communication simultaneously on a node. This specification doesn't dictate the format of session ID.

5.7. Network Community

The hybrid peer-to-peer network has the ability to contain multiple communities. Each community has its own peer-to-peer nodes and functional nodes (control nodes, gateway nodes and proxy nodes). The nodes in different communities can't communicate with each other. It is possible to manage several network communities using one control or several control nodes.

In the hybrid peer-to-peer network, if a peer-to-peer node joins without the approval of the community being joined, it is recognized in the default community that managed by the control node the peer node connects to. If the community approval is given, the node is authenticated by the control node.

In the pure peer-to-peer network, the actions of peer-to-peer nodes are the same as those of nodes in the hybrid peer-to-peer network. When it is connected with a hybrid peer-to-peer network, over the gateway nodes, the network community it joins is the community which the gateway node participates in a hybrid peer-to-peer network. If the gateway node does not have the authority to join the hybrid peer-to-peer network, the entire pure peer-to-peer network will be refused connection with the hybrid peer-to-peer network.

5.8. Roles

A "role" is a functional entity on the Peer-to-Peer architecture consisting of one or multiple implemented node functions. Depending on the Peer-to-Peer applications, nodes can proactively or reactively play different roles at different times.

This specification defines three roles: provider, relay, and consumer. Provider role is to provide meaningful application data, relay role is to relay the data, and consumer role is to receive the data.

5.8.1. Role determination

Role determination is based on the Peer-to-Peer application and its mechanism could include, but is not limited to:

- Determining the role of a node based on user settings. This means that a node could be assigned a fixed role. For instance, a node on the Peer-to-Peer network could be assigned a provider role for providing node management service or route information.
- Determining the role of a node dynamically based on other nodes' needs. This means a node could have multiple roles, and at any time, the role is determined based on application requests, or a series of roles could be enacted at specific times. For instance, a node could first take a provider role for a contents providing service; if it does not have the content desired, it enters the consumer role to acquire the content and then enter the relay

role to transfer the content. At a specific time, the node may not only provide content by itself, but also transmit a content query message to other nodes. Roles may also change depending on network conditions.

This specification does not mandate how role determination is done, and all of the mechanisms described above can be used.

5.9. Communication mode

Two communication modes are defined for Peer-to-Peer communications: proactive communication mode and reactive communication mode. The two modes are used according to the communication need.

5.9.1. Proactive communication mode (advertise)

In the proactive communication mode, a node can initiate communication with another node without receiving any request. This mode is generally used to declare information of existence on the network when a node enters or leaves the network. It is also used to advertise the node's functional capabilities, network context information, and other information.

For example, once a node enters a network, it could advertise it's own node functional capabilities or network context information to other nodes to inform then of it's existence; these nodes could then access the node by using the advertised information.

5.9.2. Reactive communication mode (request and respond)

In the reactive communication mode, communication is established in response to a request by another node. This communication mode is generally used in transactions such as requests for the services of an application that resides on another node, and to establish a trusted connection between two nodes, for example, request the functional capabilities of a node and get the response.

5.10. Communication type

The Peer-to-Peer transport protocols could use unicast, multicast or broadcast type communication. Each communication type may be used at specific time, it depends on the Peer-to-Peer applications.

6. Components of node architecture

The Peer-to-Peer architecture aims to provide a scalable and extensible application development environment for each node. This is achieved through the layered design of the node architecture. Each layer provides a set of services and/or functions to the above services and applications through a set of well-defined interfaces. Each layer of the architecture is accessible by the layers above, as well as by other services and applications.

The node architecture is broken into three layers. At the bottom is the communication layer that deals with peer-to-peer node establishment, communication management such as routing, multicast communication for group peer-to-peer nodes, and authentication management, etc. In the middle is a service layer that deals with higher-level concepts, such as resource identification and first peer discovery. At the top is the layer of applications, such as instant messaging applications and metadata search applications. Security is featured in all three layers and throughout the entire system.

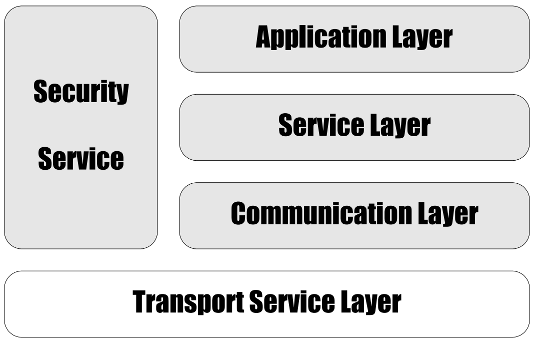


Figure 5. peer-to-peer node architecture

6.1. Peer-to-Peer Node Architecture

The peer-to-peer architecture assumes four kinds of peer-to-peer nodes.

6.1.1. General Peer-to-Peer Node

A general peer-to-peer node is a basic communication entity in the peer-to-peer architecture. The functions it needs to hold depend on where the node is and what actions it should achieve. For example, a peer-to-peer node in the pure peer-to-peer network doesn't need to install all of the peer-to-peer communication service protocols although it is necessary for peer-to-peer nodes in the hybrid peer-to-peer network, because in a pure peer-to-peer network, there is no control node to connect with, as Figure 1 in Chapter 5 shows.

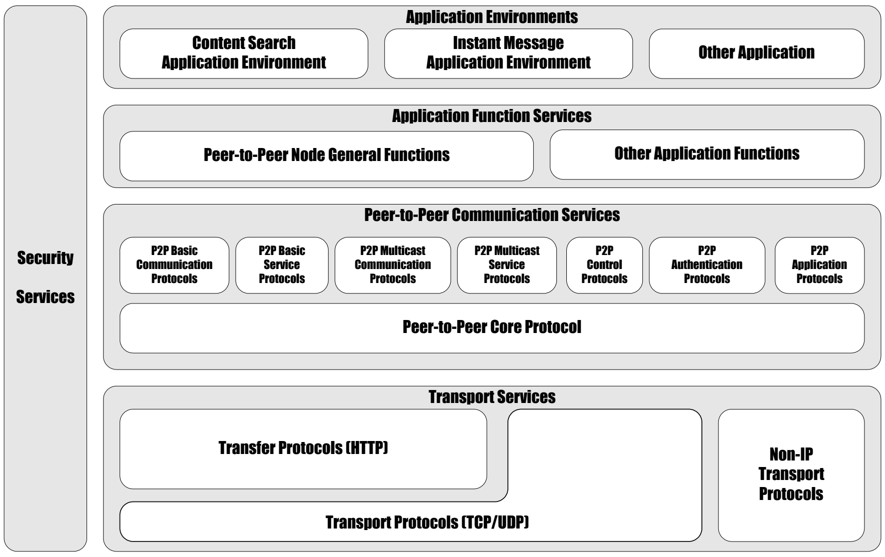

Security Services
Application Environments
Content Search Application Environment
Instant Message Application Environment
Other Application
Application Function Services
Peer-to-Peer Node General Functions
Other Application Functions
Peer-to-Peer Communication Services
P2P Basic Communication Protocols
P2P Basic Service Protocols
P2P Multicast Communication Protocols
P2P Multicast Service Protocols
P2P Control Protocols
P2P Authentication Protocols
P2P Application Protocols
Peer-to-Peer Core Protocol
Transport Services
Transfer Protocols (HTTP)
Transport Protocols (TCP/UDP)
Non-IP Transport Protocols

Figure 6. The peer-to-peer node architecture

6.1.2. Control Node

The control node plays an important role in the hybrid peer-to-peer network. For network management and to enhance the efficiency, scalability, and reliability of the peer-to-peer network, six services are defined in the service layer of the control node. These services are:

- First node discovery service
- Network recovery
- Route information optimization
- Node authentication
- Multicast group management
- Topology optimization

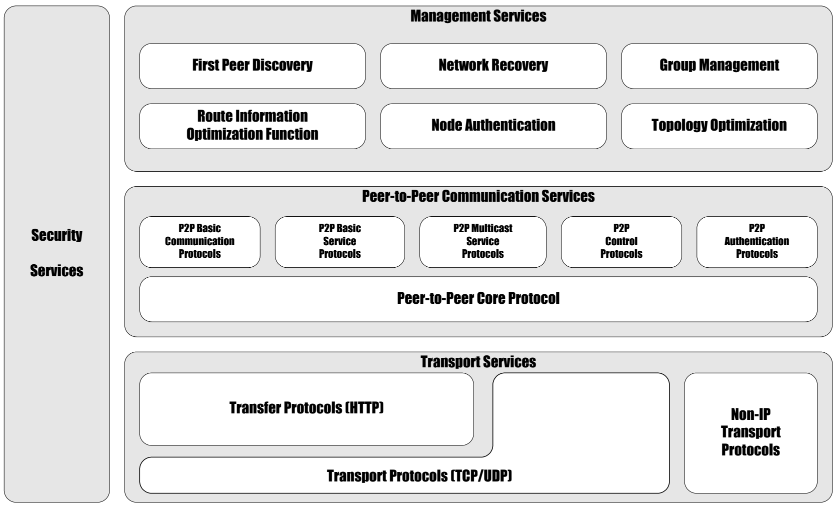

Management Services
First Peer Discovery
Network Recovery
Group Management
Route Information Optimization Function
Node Authentication
Topology Optimization
Security Services
Peer-to-Peer Communication Services
P2P Basic Communication Protocols
P2P Basic Service Protocols
P2P Multicast Service Protocols
P2P Control Protocols
P2P Authentication Protocols
Peer-to-Peer Core Protocol
Transport Services
Transfer Protocols (HTTP)
Transport Protocols (TCP/UDP)
Non-IP Transport Protocols

Figure 7. Control node architecture

6.1.3. Gateway Node

A gateway node is the connection entity that links the pure peer-to-peer network to the hybrid peer-to-peer network. A transformation function service is defined in the gateway node to relay the information between these two physically different networks. In addition to the basic functions needed to implement general activities as a hybrid peer-to-peer node, it also hosts the functions needed for working as a peer-to-peer node in the pure peer-to-peer network.

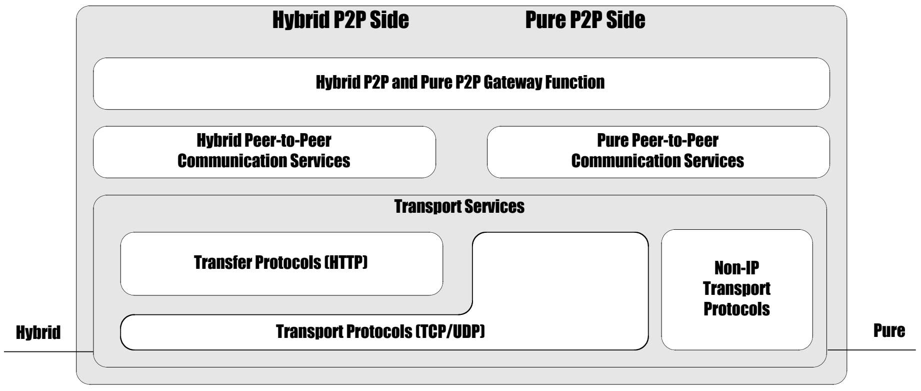

Hybrid P2P Side
Pure P2P Side
Hybrid P2P and Pure P2P Gateway Function
Hybrid Peer-to-Peer Communication Services
Pure Peer-to-Peer Communication Services
Transport Services
Transfer Protocols (HTTP)
Transport Protocols (TCP/UDP)
Non-IP Transport Protocols
Hybrid
Pure

Figure 8. Gateway node architecture

6.1.4. Peer-to-Peer Node with proxy function

A proxy node is a general peer-to-peer node with added proxy functions for associating its dependent devices with the peer-to-peer network. As described in Chapter 5, depending on how the proxy node is defined, the devices are presented in different ways to the other peer-to-peer nodes.

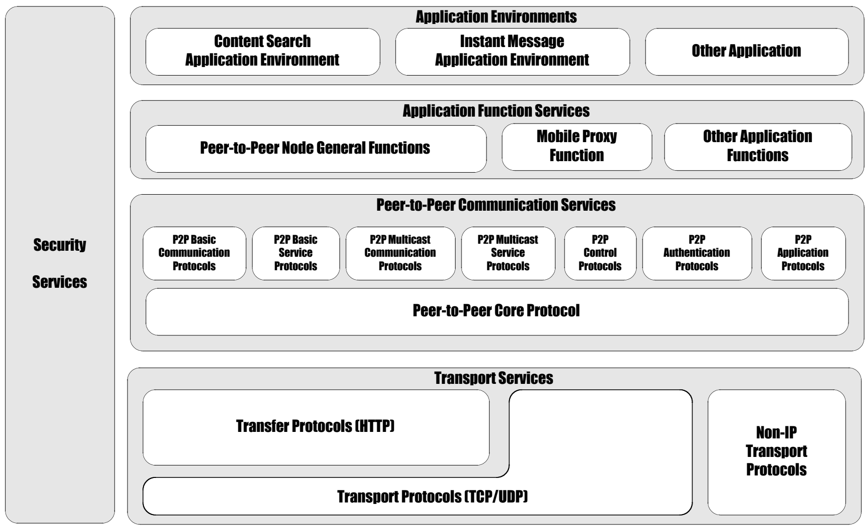

Security Services
Application Environments
Content Search Application Environment
Instant Message Application Environment
Other Application
Application Function Services
Peer-to-Peer Node General Functions
Mobile Proxy Function
Other Application Functions
Peer-to-Peer Communication Services
P2P Basic Communication Protocols
P2P Basic Service Protocols
P2P Multicast Communication Protocols
P2P Multicast Service Protocols
P2P Control Protocols
P2P Authentication Protocols
P2P Application Protocols
Peer-to-Peer Core Protocol
Transport Services
Transfer Protocols (HTTP)
Transport Protocols (TCP/UDP)
Non-IP Transport Protocols

Figure 9. Proxy node architecture

6.2. Peer-to-Peer Communication Services

The Peer-to-Peer communication services are achieved through the layered design of the protocol stack, independent of the transport layers. These protocols could be bound to HTTP, TCP, UDP or other Non-IP transport protocols directly. These protocols include peer-to-peer basic communication protocols, peer-to-peer basic service protocols, peer-to-peer multicast communication protocols, peer-to-peer multicast service protocols, peer-to-peer control protocols, peer-to-peer authentication protocols and peer-to-peer application protocols. It is not necessary to implement all of protocols on every node. A node can implement just the protocols needed.

6.2.1. Peer-to-Peer Core Protocol

Peer-to-Peer Core Protocols encapsulate the messages from the upper layer with the appropriate Peer-to-Peer communication mode based on the different communication needs and pass the encapsulated messages to the low layer protocol.

Peer-to-Peer Transport Protocols could include:

- Proactive type Peer-to-Peer Transport Protocol

  This protocol allows a node to communicate with other nodes without any request by using the proactive Peer-to-Peer communication mode.
- Reactive type Peer-to-Peer Transport Protocol

  This protocol allows a node to communicate with other nodes in response to the other nodes' requests by using the reactive Peer-to-Peer communication mode.

6.2.2. Peer-to-Peer Basic Communication Protocols

The Peer-to-Peer Basic Communication Protocols provide the basic communication services to the nodes in the

Peer-to-Peer architecture for communicating to each other,

The Peer-to-Peer Basic Communication Protocols include, but are not limited to:

- Hello Method

The Hello method is used to pass to other nodes information of own (node) existence on the network.

- Bye Method

The Bye method is used to inform other nodes before the node releases its connection to the network.

- Resource exchange method

The Resource exchange method is used to exchange resource information with other nodes on the network.

6.2.3. Peer-to-Peer Basic Service Protocol

The Peer-to-Peer Basic Service Protocol is a protocol used by the control node to provide basic services to the peer-to-peer nodes. It is applied to the communications between the control node and the peer-to-peer nodes. The protocol includes but is not limited to:

- Service provide method

This method provides the following functions:

- First peer discovery

  This function retrieves a set of peer-to-peer node addresses that the new peer-to-peer node should connect to.

- Message Routing information

  This function provides message routing information to the requesting peer-to-peer node when it wants to send messages to another node. The message routing information includes the peer-to-peer nodes that the messages will pass through.

- Network optimization

  -Detects network splitting and provides network recovery information.

  -Optimizes the network topology by advertising the information leading to the establishment of new connections and the termination of existing connections between the peer-to-peer nodes with the goal of adjusting the topology.

- Name resolution

  The control node translates destination node ID into transport layer address and return this information to the requesting peer-to-peer nodes to allow them to send messages.

- Network cost information

  It provides the hop count from source peer-to-peer node to destination peer-to-peer node at the request of a peer-to-peer node.

6.2.4. Peer-to-Peer Multicast Communication Protocols

The Peer-to-Peer Multicast Communication Protocols provide services for multicast communication among group peer-to-peer nodes. The protocols include but are not limited to:

- Join protocol

This protocol allows a peer-to-peer node to express its intent to join a multicast communication group.

- Leave method

This method allows a node to express its intent to leave a multicast communication group.

6.2.5. Peer-to-Peer Multicast Service Protocols

The Peer-to-Peer Multicast Service Protocols provide multicast services for peer-to-peer nodes. The protocols include but are not limited to:

- Multicast Service provide method

This method includes following functions:

  - ➢ First multicast peer discovery:

    Like the first peer discovery function for general peer-to-peer nodes, this function provides a new multicast peer-to-peer node with the address of the first peer-to-peer node it should connect to.

  - ➢ Multicast Nodes List:

    It provides a list of peer-to-peer nodes in a designated group.

6.2.6. Peer-to-Peer Control Message Protocols

The Peer-to-Peer Control Message Protocols provide error report and node diagnosis report service to the nodes in the peer-to-peer architecture. The protocols include but are not limited to:

- Error report method: used to report an error.
- Lookfor method:
- Diagnose method
  - ➢ used to diagnose and solve network problems.
  - ➢ used to search node on the network.

6.2.7. Peer-to-Peer Security Management Protocols

The peer-to-peer authentication protocols provide node authentication services for nodes in the hybrid architecture. The protocols include but are not limited to:

- Authenticate Method

The Authentication method is used when a peer-to-peer node joins the peer-to-peer network or joins a multicast group. This method consists of two messages, *authenticatechallenge* and *authenticateresponse*. Authentication is performed after the request is sent to the control node.

- Inquiry Method

Inquire method is used to confirm that the node issuing a request for communication services is a legal node of the peer-to-peer network. This identification is performed by the control node.

6.2.8. Peer-to-Peer Application Protocols

The Peer-to-Peer Application Protocols depend on Peer-to-Peer application needs, and can be extended by Peer-to-Peer application developers.

For instance, with regard to content search Peer-to-Peer applications, the Peer-to-Peer application protocols could include, but are not limited to:

- Meta information advertising Protocol
- Meta information searching Protocol
- Meta information responding Protocol
- Meta information updating Protocol
- Meta information deleting Protocol

6.3. Application Function services

Application function services include functions for achieving a variety of applications. They collect the context of the application environment, arrange application data, and manage Peer-to-Peer communications.

Application function services include, but are not limited to:

- General functions
- Proxy function
- Other application functions

6.3.1. General functions

General functions provide basic, common functions for achieving Peer-to-Peer applications. These functions must be implemented on each node within the Peer-to-Peer architecture.

The general functions include, but are not limited to:

- Manage user's preference and node's capability.
- Arrange the application data.
- Protect user privacy.
- Manage grouping services.
- Topology optimization

  Each node monitors its local topology and periodically autonomously optimizes the topology according to a certain criteria and algorithm (e.g. hopcount, bandwidth, etc). This heuristic method needs the peer-to-peer nodes information exchange in a constrained range (e.g. 2-hops) and the weighted value (e.g. hopcount) between each pair peer-to-peer nodes within in the designated range should be taken into account.

The requirements of the topology optimization mechanism include, but are not limited to:

- Autonomous optimization based on local information
  - Peer-to-peer node should be able to optimize the local topology without any administrative interference
- Scalability
  - Incremental node quantity should be considered for deploying this function to the nodes in a large-scale network (e.g. 10000 nodes).
- Applicability to heterogeneous networks
  - A general mechanism that applicable to heterogeneous networks is expected.
- Adaptation to dynamic change of network
  - To adapt the dynamic changes of peer-to-peer network topology, such as a random departure or an unpredictable failure of its adjacent peer-to-peer nodes, peer-to-peer node should have the ability to detect these failures and survive from them.

6.3.2. Proxy functions

Proxy function offsets the limitations of small devices' capabilities so that small devices can join the Peer-to-Peer architecture.

The proxy functions include, but are not limited to:

- Provide protocol convergence (wireless to wired, et. al) services.
- Provide data storage service.
- Provide data format conversion services.

6.3.3. Other Application functions

Compared to general functions, these functions are more dependent on specific applications. They are assumed to provide value-added functions.

Other applications functions include, but are not limited to:

- Semantic route analysis
- Date base management
- Content search
- Index

6.4. Management Services

Management services are only defined in the control node for providing network services to the peer-to-peer nodes. These services include but are not limited to: first peer discovery, node authentication, network recovery, multicast ID management, route information optimization, network topology optimization, etc.

6.4.1. First Peer Discovery

When a peer-to-peer node joins a hybrid network, it has to connect to an existing peer-to-peer node. The first discovery function provides the addresses of existing peer-to-peer node to the new peer-to-peer node. The mechanism of neighbor node selection depends on the purpose of application. The neighbor peer-to-peer nodes may be those that have a close physical address (e.g. a similar IP address) or a similar interest (music amateurs).

6.4.2. Node Authentication

- The node authentication function provides an authenticating service for the peer-to-peer nodes. Its functions include but are not limited to: Node authentication function

  When a peer-to-peer node joins the hybrid peer-to-peer network, the control node will check its node name and password to confirm whether it is allowed to join this network.
- Authentication between peer-to-peer nodes function

  When a peer-to-peer node wants to build connection to another node, this function will be used by the intended peer-to-peer node to ask the control node to confirm the identity of the requesting party, so as to ensure the security of the connection between them.

6.4.3. Network Recovery

Network recovery functions ensure the connectivity of the hybrid peer-to-peer network and its recovery if the hybrid peer-to-peer network is split due to some unpredictable device or network problems. Its functions include, but are not limited to:

- Network connectivity function
- Network recovery function

6.4.4. Route Information Optimization

Route information optimization functions provide optimized route information for those peer-to-peer nodes that want to send messages to other peer-to-peer nodes.

6.4.5. Multicast Group Management

Multicast group management functions provide multicast ID management service and multicast routing optimization service for the peer-to-peer nodes.

6.4.6. Network Topology Optimization

Topology optimization functions provide network optimization management service by patrolling and periodic optimization of the dynamically changing network topology.

6.4.7. Node Name Resolution

Node name resolution retrieves peer-to-peer node transport layer address using its node ID or user ID or both.

6.5. Security Services

Security forms a fundamental part of the Peer-to-Peer architecture, and its services can be found in all layers of the Peer-to-Peer architecture including the network transport protocol layer.

The security services include, but are not limited to:

- Security for pair and group communication
- Authentication
- Peer access control (authorization)
  - Management of affiliation information of peers
- Digital Right Management

6.6. Application Environment

The application Environment supports the realization of a variety of Peer-to-Peer applications by applying the various Peer-to-Peer application functions.

The Peer-to-Peer applications include, but are not limited to:

- Content search application
- Instant messaging application

Each application environment depends on which Peer-to-Peer application is provided.

6.7. Gateway Communication Service

Gateway communication service provides route information for peer-to-peer nodes in the pure peer-to-peer network by acquiring the optimized route information from the control node in the hybrid network. Its functions include, but are not limited to:

- Information relay function

When the gateway node receives a route request message from the pure node-to-node network, it forwards it to the control node automatically. Whether the response to this request is sent directly to the requesting peer-to-peer node depends on the route information optimization function of the control node. Because the control node manages all information of the network's topology, it should know if the destination peer-to-peer node is in the pure network. If the destination peer-to-peer node is, the control node would not response this request. If the destination peer-to-peer node is not in the pure node-to-node network, the control node will find a route and send a response to the gateway node, which will then forward it to the requesting peer-to-peer node in the pure peer-to-peer network.

7. Resource discovery

In the Peer-to-Peer architecture, nodes, node attributes, node application functions, node communication functions etc. can be called resources. These resources are not persistent and may change dynamically, especially in wireless environments. To establish Peer-to-Peer applications, resource discovery is needed.

Resource discovery can be separated into two levels:

- Level 1: Find the first node for entering the Peer-to-Peer network.
- Level 2: Find the necessary resource information on the Peer-to-Peer network.

Since the defined Peer-to-Peer architecture can be mapped to variety of Peer-to-Peer network configurations, e.g. completely decentralized Peer-to-Peer network, completely centralized Peer-to-Peer network, or a hybrid of the two, on different Peer-to-Peer networks, a variety of resource discovery mechanisms are required.

The resource discovery mechanisms could include, but are not limited to:

- Send broadcast discovery message to all nodes in the Peer-to-Peer network
- Send multicast discovery message to all nodes in a specific group.
- Send unicast discovery message to a specific node pre-known by the node itself.
- Send unicast discovery message to a centralized node, which holds the information of other nodes.
- Advertise own (node) resources to other nodes.

This Specification does not mandate how node discovery is done, and all the mechanisms described above can be used.

8. Communications between Mobile device and proxy

Mobile devices that are not able to function as independent nodes need to use a proxy to enter the Peer-to-Peer network. The communications between mobile device and proxy could be realized by using current existing communication technologies on mobile Internet, e.g. HTTP on TCP/IP, and the data format could be i-html, CGI, iAppli and so on.

9. Network performance enhancement (cost-based routing)

For improving the performance of Peer-to-Peer applications, some nodes on the Peer-to-Peer architecture could provide optimum routing (cost-based routing) functions. This performance enhancement could be achieved by extending and implementing application functions (e.g. semantic routing mechanism) on some peer-to-peer nodes.

10. Sample configurations of Peer-to-Peer architecture for the mobile Internet

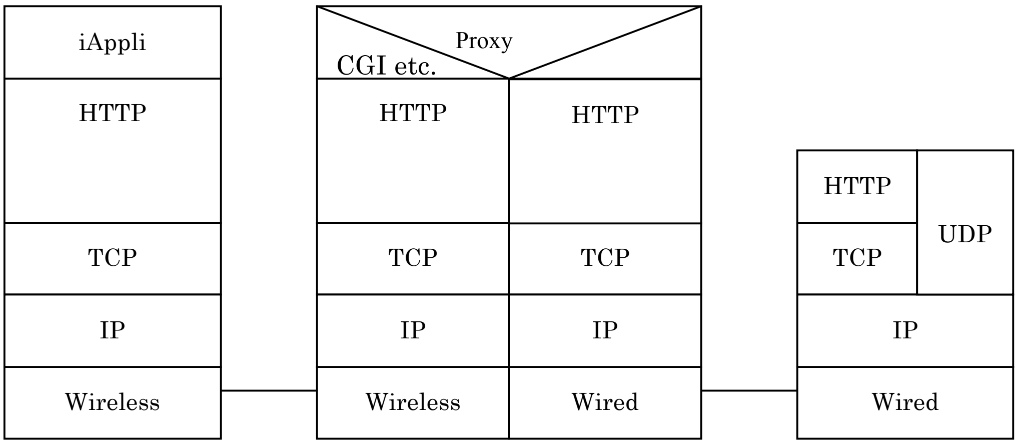


Figure 10. Access via a proxy

Figure 10. shows a node on a wireless network accessing a node on a wired network via a proxy node.

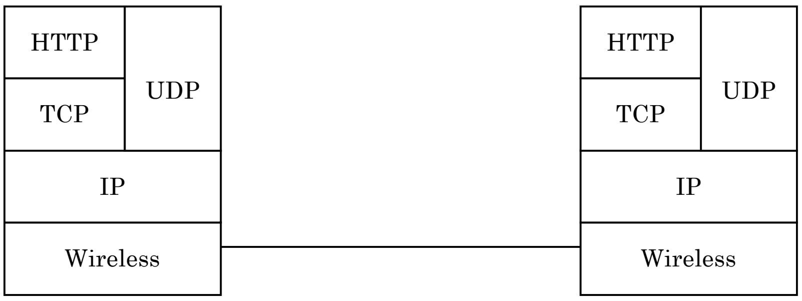


Figure 11. Direct Access

Figure 11. shows a node on a wireless network directly accessing a node on a wireless network.

Appendix A. Version History

Document number	Date	Note
PUCC P2P Architecture V0.1	30 Sep. 2007	Version 1.0
PUCC P2P Architecture V2.0	22 March. 2012	Version 2.0

Appendix B. Use Cases

A study of the literature reveals a number of possible use cases for the peer-to-peer architecture. They are briefly described below. The architecture for them will be developed as part of this project.

In principle, all peer-to-peer applications can be divided in two categories: Transactions (including messages); and multicasts (e.g. of video streams). The difference in principle is that the messages are discrete entities, whereas the streams do not have a termination.

It is possible to solve the communication problems with many different architectures, including client-server architectures (it is even possible to make the claim that there are no peer-to-peer architectures, only applications). This means that when designing an architecture for a peer-to-peer system, it is more a question of selecting a suitable architecture and expressing that in a communications system than defining a communications system from scratch.

While the current mobile environment presents a number of problems (e.g. low network capacity and communication failures), there are several advantages as well. The mobile environment can be used to provide position information, which in turn can be used to create positioned "buddylists" (such as in the now defunct iPulse Locator product from Ericsson). However, position information is in principle nothing but an additional variable describing the situation of the user, and so does not have to be taken in consideration, especially if the environment allows for personalization of the presentation (which is another discussion).

It is also notable that there is nothing that says anything about the nature of the nodes. A P2P system could as easily be adapted to a machine-to-machine (or M2M) environment as an environment where humans communicate. It all depends on the use case.

In the environment addressed by the current project, we are constrained further by the fact that our user devices are handheld devices with limited display capabilities (e.g. mobile phones). This makes certain user behaviors more likely than others (e.g. scrolling is almost totally uninteresting).

In this document, we briefly discuss the following use cases:

1. Semantic search engine and index server
2. Instant messaging and notification
3. Machine-to-machine communication

4: Transactions (e-commerce) against other peers

5: Distributed filtering and processing (e.g. semantic firewall, my friends toplist)

6: Annotations of (static) material (including "real-world" objects); e.g. distributed helpdesk

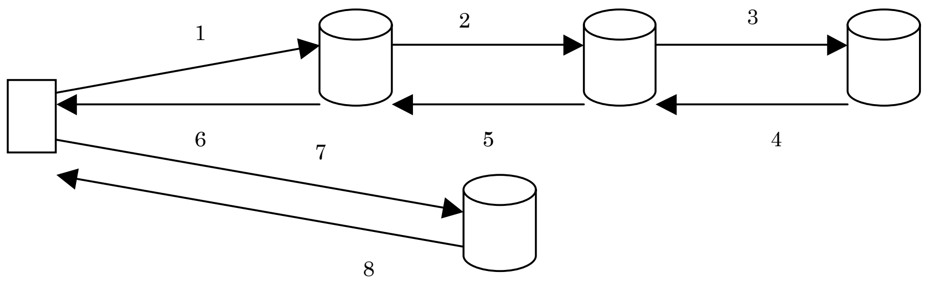
7: Load balancing (e.g. by requesting processes or functions to be remotely executed)

Actually, the remote monitoring of peers may be constructed as a use case as well (it is probably the case that some kind of keepalive signals must be used to check whether a network connection has been interrupted).

The following discusses each of these use cases.

1. Semantic search engine and index server

This use case is exemplified by Napster, possibly the most famous P2P application. A user sends a request to a server, which has an index of objects it knows about. If the object is not in its index, it forwards it to other servers it knows about, and they will do the same. When a server is found, a reply is returned, and the user can initiate the download (out-of-band from the request, e.g. using HTTP).



Here are the messages in pseudo-XML:

1: <request/>

2:<server1<request/>/>

3:<server2<server1<request/>/>/>

4:<server2<server1<response/>/>/>

5:<server1<response/>/>

6:<response/>

7:<get/>

8:<response/>

This enables us to use progressive envelopes as message route descriptors. The servers can add an encapsulation envelope to show that they have passed the message. This can also be used for message routing (e.g. adding topological distances or calculating cost-based routes).

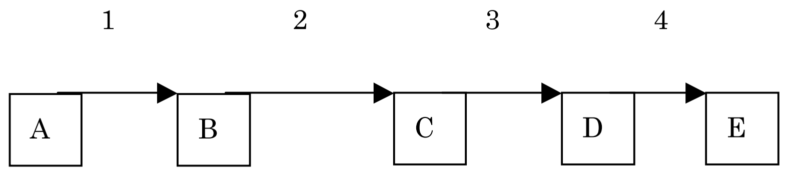
There must be a keepalive function in the messages, otherwise the request could travel an infinite distance (i.e. max 5 hops from the user, or something like that).

It can also be used inside a firewall.

2. Instant messaging and notification

In an instant messaging system, small messages are passed rapidly between the members of a group. One frequent function of this type of system is the "buddy list", which contain entries for users that the user wants to communicate with. A standardized version could be reused in the system.

SMS in GSM is often used in this way, but it permits only one-to-one communication. Using a P2P system instead, messages could be passed to several users at once. Note that such a system is inherently a message routing system, since peers who receive a message not intended for them must pass it on to the next peer.



If user A wants to pass a message to B, D, and E, it would look something like the following:

1: <message<recipient="B D E"/>/>

C would pass the message on without opening it. If this method is used for notification, it can also include encryption, signing, and transaction facilities (in an OLTP system, a transaction is not committed to the database before an acknowledgement from the user has been received). This method would be used in use case 4.

If the "buddy list" was distributed, the system could retrieve buddy lists from all users as the message passed through each one, append them, and correlate the buddy lists as necessary.

3. Machine-to-machine communication

Machine-to-machine communication is conceptually no different from human communication, except that computers are more direct and literate. Messages have to be highly structured and actions clearly defined.

4: Transactions (e-commerce) against other peers

This use case is very similar to use case 2, and the only difference is a higher degree of security; the receiver may also be a software program in a legacy server.

5: Distributed filtering and processing (e.g. semantic firewall, my friends toplist)

In use case 2, the intermediate users could correlate the buddy lists to arrive at a common buddy list. This mechanism can be expanded; the messages being passed could contain, for example, log files, lists of downloads, etc. Specialized servers could handle the correlation and distribution of material. Access could be made contingent on belonging to a list (e.g. "You can only download this song if you are one of Yuki's friends" – "Only members of the Yokosuka iArea Gold Club can download this document"). The lists could be generated on an ad-hoc or temporary basis.

6: Annotation of (static) material (including "real-world" objects); e.g. distributed helpdesk

One of the use cases brought forward for P2P is annotation, e.g. annotation of web sites. One of the more well-know examples of this is Third Voice, but their system is entirely contingent on using their database. The W3C Annotea system represents a client-server solution. Annotations, however, could be handled the same way as queries in use case 1. In this case, the user would be able to annotate electronic materials. But given the availability of accurate enough position data, the user could annotate any object in the physical space (e.g. "This restaurant rocks!"). Annotations could then be correlated with buddy lists, and made available only to a select group.

If there was a payment system involved, this could also be used to create services where users were rewarded for their work, e.g. if a user sent out a query asking "how do I change my email address" and other users answered, the first user could thank the other users by clicking the "I know now" button, which would credit the user who gave the answer with a small sum (e.g. 100 JPY).

7: Load balancing (e.g. by requesting processes or functions to be remotely executed)

Strictly speaking, most servers are peers, but they are not enabled to communicate with each other. Efforts such as IETF WEBI are intended to enable servers to communicate (like the servers do now in the Akamai network, for instance). However, it can be posited that in HTTP, the servers can offload functionality by calling a function on another server (e.g. a file). This is a very lightweight and high-level functional load balancing, which could easily be extended to the P2P environment.

Note also that this might mean that some resources, e.g. disk space, could be held by the operator, and provided as a service by him at the request of the user. Redirects of information would not have to go by the user at all.

This type of application is used in a class of peer-to-peer applications exemplified by SETI@home, where the processing is performed in a distributed manner by computers who donate their time, under the coordination of a central node.

Appendix C What is UUID?

A UUID is an identifier that is unique across both space and time, with respect to the space of all UUIDs. To be precise, the UUID consists of a finite bit space. Thus the time value used for constructing a UUID is limited and will roll over in the future (approximately at A.D. 3400, based on the current algorithm). A UUID can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects across a network.

The current generation of UUIDs does not require that a registration authority be contacted for each identifier. Instead, it sets a unique value over space for each UUID generator. This spatially unique value is specified as an IEEE 802 address, which is usually already available to network-connected systems. This 48-bit address can be assigned based on an address block obtained through the IEEE registration authority. This section of the UUID specification assumes the availability of an IEEE 802 address to a system desiring to generate a UUID, but if one is not available, Section 4 specifies a way to generate a probabilistically unique one that can't conflict with any properly assigned IEEE 802 address.

One of the main reasons for using UUIDs is that no centralized authority is required to administer them (beyond the one that allocates the IEEE 802.1 node identifiers). As a result, generation on demand can be completely automated, and they can be used for a wide variety of purposes. The UUID generation algorithm described here supports very high allocation rates: 10 million per second per machine if you need it, so that they could even be used as transaction IDs.

UUIDs are fixed-length (128-bits) and are small relative to the alternatives. This fixed, relatively small size lends itself well to sorting, ordering, and hashing operations of all sorts, storing in databases, simple allocation, and ease of programming in general.

Appendix D Consideration of Control Node

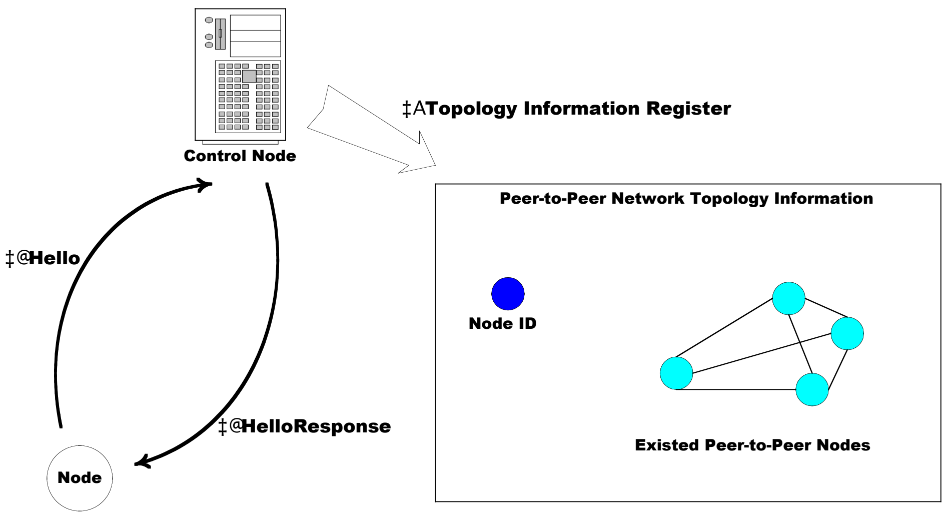
The Control Node is defined for providing network services to general peer-to-peer nodes. In the following sections, we will enumerate mechanisms on how to realize these services.

1. Network topology information management

Control node constructs peer-to-peer network topology information based on three messages defined in the peer-to-peer protocols. They are “Hello”, “ResourceInformationAdvertise” and “Bye” messages.

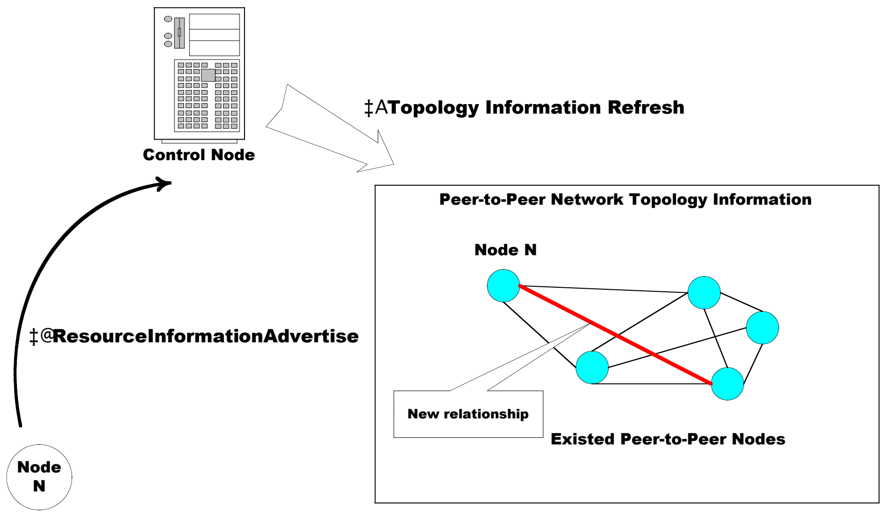
1) Hello Message

As the figure shows, when a peer-to-peer node enters the peer-to-peer network, it first sends a Hello message to the control node. The control node then returns a HelloResponse message to the new peer-to-peer node and reflects this new node in its topology information.


Control Node
‡ATopology Information Register
Peer-to-Peer Network Topology Information
‡@Hello
Node ID
‡@HelloResponse
Existed Peer-to-Peer Nodes
Node

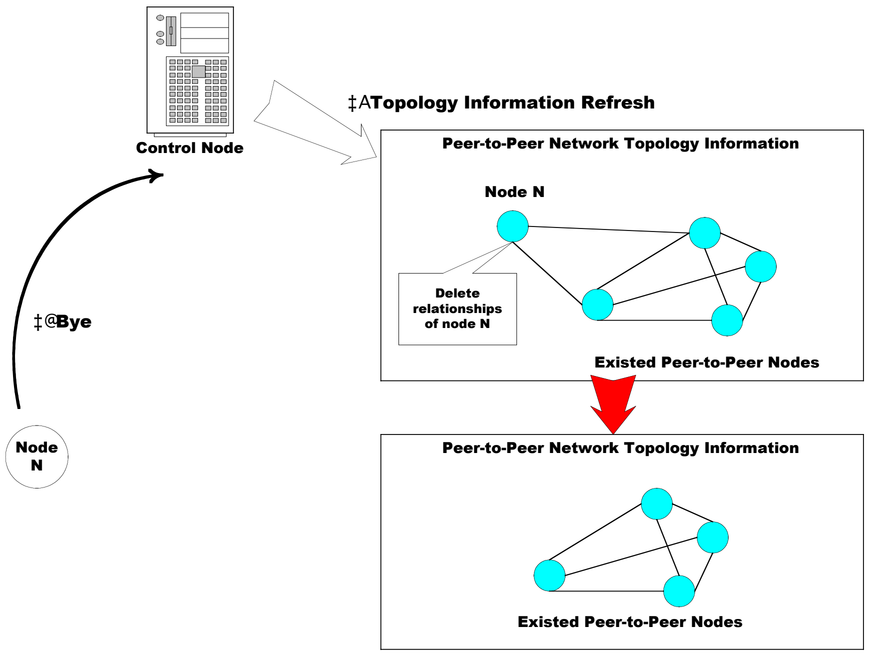
2) ResourceInformationAdvertise Message

ResourceInformationAdvertise message is sent by a peer-to-peer node when its resource information changes. The control node has to refresh its topology information upon receiving this advertising information.


‡ATopology Information Refresh
Control Node
Peer-to-Peer Network Topology Information
Node N
‡@ResourceInformationAdvertise
New relationship
Existed Peer-to-Peer Nodes
Node N

3) Bye Message

Control node delete node information from network topology information when it receives a bye message from the departing node.


‡ATopology Information Refresh
Control Node
Peer-to-Peer Network Topology Information
Node N
‡@Bye
Delete relationships of node N
Existed Peer-to-Peer Nodes
Node N
Peer-to-Peer Network Topology Information
Existed Peer-to-Peer Nodes

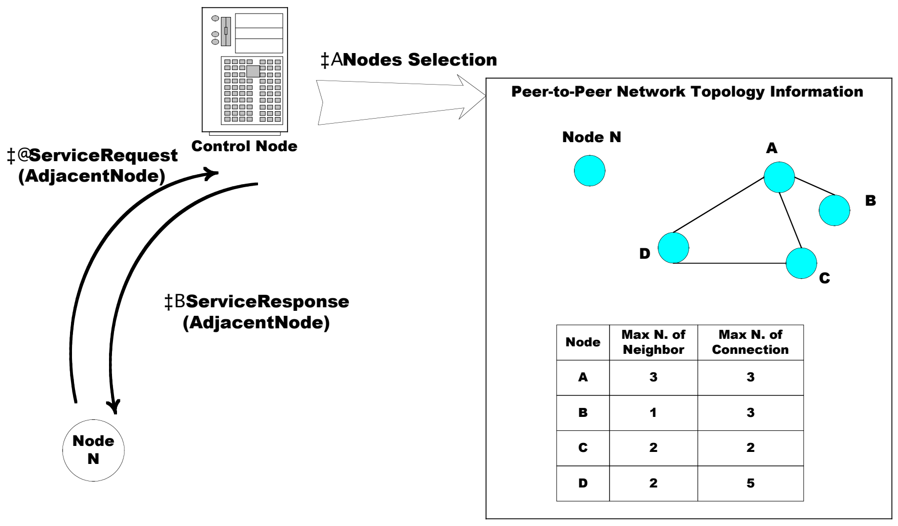
2. First peer discovery

First peer discovery service provides a new node with a list (including node ID and node transport layer addresses) of existed nodes for it to connect to when the new node joins the peer-to-peer network for the first time. It is not necessary to request this service every time when it enters. The new node also can build connections to its fellow using prior information. This service is just defined as a default status when a new node enters without any declaration.

There are four mechanisms being considered. No definitive selection is needed since usage depends on the application.

1) Random selection

This is the simplest method. The control node just randomly selects peer-to-peer nodes from the topology information it holds and returns the list to the requesting node.


‡ANodes Selection
Control Node
‡@ServiceRequest (AdjacentNode)
‡BServiceResponse (AdjacentNode)
Node N
Peer-to-Peer Network Topology Information
Node N
A
B
C
D

Node	Max N. of Neighbor	Max N. of Connection
A	3	3
B	1	3
C	2	2
D	2	5

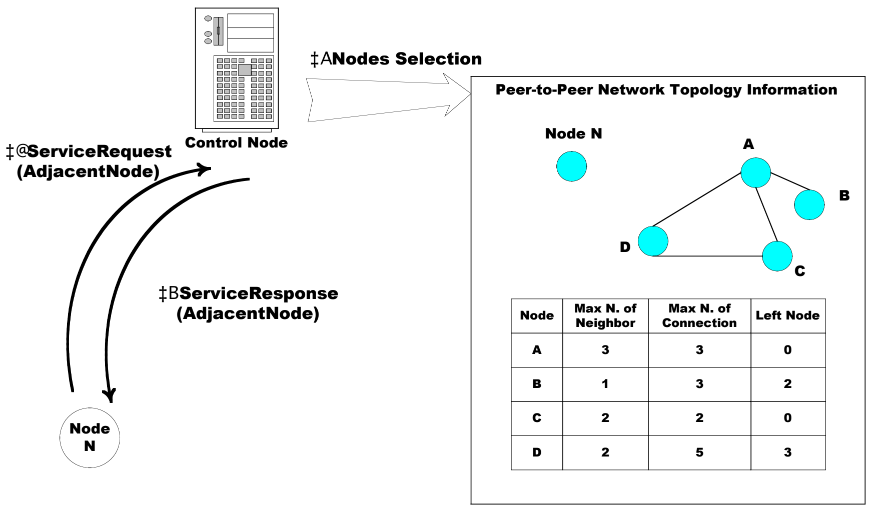
In the above figure, Node B and Node D are selected and returned to Node N to build a connection.

2) Minimum neighbor connection selection

This method selects the nodes by their number of neighboring nodes. In the previous figure, the node candidates are listed in the order Node B and Node D because B has fewer neighbor nodes than B.

3) Maximum possible connection selection

This method selects the nodes by their left node number. The following figure has an additional column at the end of the node list table. Node B and Node D are selected but the order is Node D and Node B.


Control Node
‡ANodes Selection
‡@ServiceRequest (AdjacentNode)
‡BServiceResponse (AdjacentNode)
Node N
Peer-to-Peer Network Topology Information
Node N
A
B
C
D

Node	Max N. of Neighbor	Max N. of Connection	Left Node
A	3	3	0
B	1	3	2
C	2	2	0
D	2	5	3

4) User ID selection

This method selects the node which accords the requested user ID.In the following figure, the "User ID" column is added to the node list table.

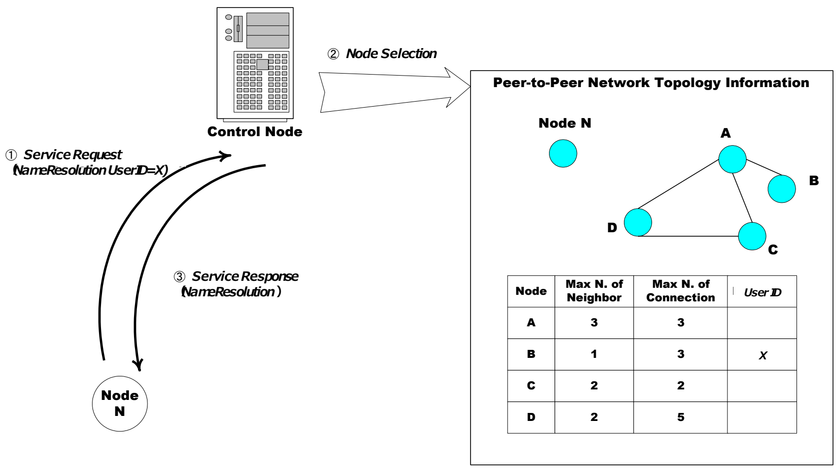
e.g.

Case : the node N resolves an node which has the user ID "X".

1) The node N requests the name resolution of the node which accords with the user ID "X" to the control node.

2) The control node selects the node B by referring the node list table.

3) The control node responds the transport layer address of the node B to the node N.

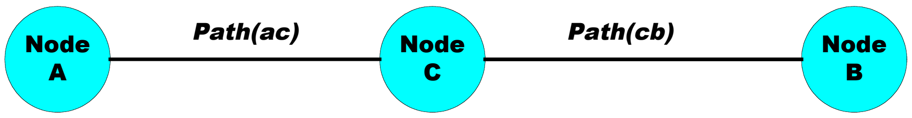

Control Node
② Node Selection
① Service Request (NameResolution UserID=X)
③ Service Response (NameResolution)
Node N
Peer-to-Peer Network Topology Information
Node N
A
B
C
D

Node	Max N. of Neighbor	Max N. of Connection	User ID
A	3	3	
B	1	3	x
C	2	2	
D	2	5	

3. Shortest Route Information Computation

Dijkstra's Algorithm can be used to supply a shortest route information computation service.

Dijkstra's algorithm is efficient at solving the single-source shortest path problem. The central concept of Dijkstra's algorithm is based on the fact that every subpath is also a shortest path. That means, if there is a shortest path ***Path(ab)*** between two nodes (Node A, Node B) and we break the path into two subpaths ***Path(ac***) and ***Path(cb)***, then ***Path(ac)*** should be the shortest path between Node A and Node C , ***Path(cb)*** should be the shortest path between Node C and Node B. Full details aren't given here. Please read the reference for more information about Dijkstra's algorithm. [Dijkstra's algorithm]



Path(ab) = Path(ac) + Path (cb)

When a general peer-to-peer node wants to send one message to another node, it sends a route information request message to the control node. The control node computes the shortest route information using Dijkstra's algorithm and returns the route information to the requesting node.

4. Network Split Diagnosis

Considering the unforeseen problems that can happen in the peer-to-peer network, such as unexpected power down or network access problems, the peer-to-peer network may be broken into several independent parts. The control node provides the network split diagnosis service for ensuring network integrity.

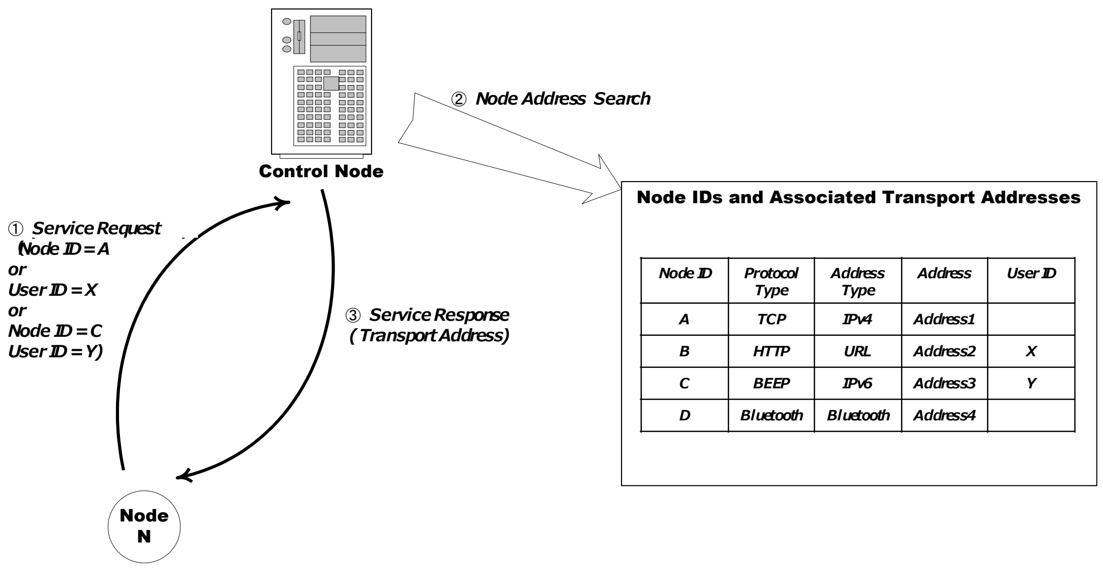
(Under construction)

5. Network Recovery

(Under construction)

6. Name Resolution

This service is used to handle name resolution requests from general peer-to-peer nodes that want to know the transport address of the destination peer-to-peer node from its Node ID or User ID or both. The control node holds a table listing all node IDs and their transport addresses and some user IDs. This table is built as the nodes join the peer-to-peer network and each entry is retained until the corresponding node leaves.


Control Node
② Node Address Search
① Service Request (Node ID=A or User ID=X or Node ID=C User ID=Y)
③ Service Response (Transport Address)
Node N
Node IDs and Associated Transport Addresses

Node ID	Protocol Type	Address Type	Address	User ID
A	TCP	IPv4	Address1	
B	HTTP	URL	Address2	X
C	BEEP	IPv6	Address3	Y
D	Bluetooth	Bluetooth	Address4	

Appendix E Autonomous Topology Optimization Mechanism and Recovery Mechanism

1. Introduction

In the previous section (Appendix C), the functionality of control node is discussed and all of functions are interpreted. In the solution of topology optimization, control node monitors the entire network topology and provides network optimization and recovery service for peer-to-peer nodes in the hybrid peer-to-peer network environment. Although this method can be introduced to the general peer-to-peer nodes, there are some obviously weaknesses. At first, the traffic load of collection of all of the peer-to-peer node information will be exponentially increased considering the rapidly incremental number of peer-to-peer nodes. Second, the calculation on the huge node information is an additional load for the general peer-to-peer nodes. In this way, an autonomous optimization without any administrative interference and huge computation capability is required. In this appendix, we introduce an autonomous topology optimization mechanism and a recovery mechanism. Both of them can be easily realized by exchange peer-to-peer node information in a limited range. The autonomous topology optimization mechanism is a mechanism which reconstructs its relationship with its adjacent peer-to-peer nodes in terms of weighted value between each pair peer-to-peer nodes within *n*-hops range around it. The recovery mechanism is a mechanism which recovery split network according to the sequence of peer-to-peer node participant into the peer-to-peer network. Requirements for topology optimization has been described in the Section 6.3.1, the general function of the peer-to-peer node.

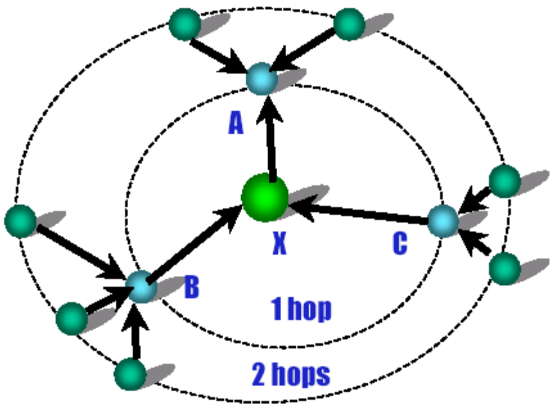
2. Design Concept

To realize the autonomous topology optimization, a limitation node information exchange and a mechanism which evaluates the current topology according to the metric information between nodes are proposed. At first, to avoid the heavy traffic, node information exchanging is constrained in a predefined range (e.g. 2-hops). Second, to optimize the topology in this range, a metric is defined and calculated. This metric is an "edge" value between each pair nodes that shows a distance between each pair nodes. Third, a node periodically find a connection around itself by which the sum of "distance" consumption to reach all nodes in the network is minimum. In the following section, details will be interpreted.

Following table is an example of node information. "Node ID" is the logical name of current peer-to-peer node. "IP" is the physical address for connection creation. It can be non-IP addresses in heterogeneous networks. "MaxConnection" is the maximum connection ability of the peer-to-peer node. "NeighborID" is the logical names of its adjacent nodes. "Relation" is defined to denote the relationships between current peer-to-peer node and its neighbors. Here, "Parent" means the neighbor node is connected by the current peer-to-peer node and "child" means the node is connected by the neighbor node. in this way, total network topology can be display as a directed graph while peer-to-peer nodes are connected by directed valued edges according to their participation order in the network. "Metric" provides the value between node and its neighbors based on certain criteria.

Node ID	IP	Max Connection	Neighbor	Relation	Metrics
X	xxxx	5	A	Parent	3
			B	Child	4

Then, this information is broadcasted to all nodes within n2-hops range. See the following figure as an example.

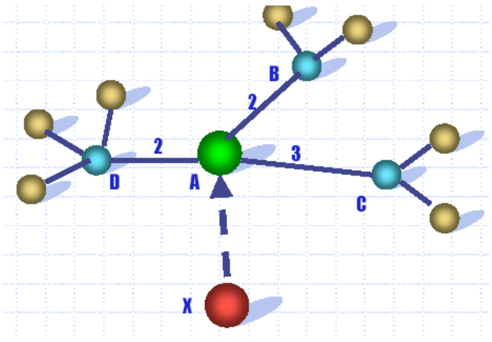


Node X in the same way, will collect all of its adjacent node information and a table as the following can be built.

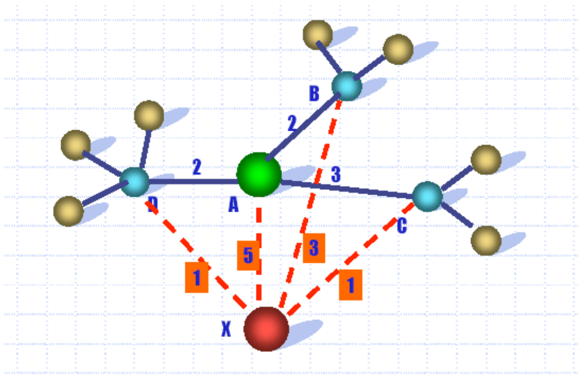
Node ID	IP address	Max Connection	Neighbor ID	Relationship	Metrics
X	**xxxx**	**5**	**A**	**Parent**	**3**
			B	**Child**	**4**
A	**xxxx**	**4**	**A1**	**Child**	**3**
			A2	**Child**	**4**
			X	**Child**	**3**
A1	**…**	**…**	**…**	**…**	**…**
A2	**…**	**…**	**…**	**…**	**…**
B	**xxxx**	**3**	**X**	**Parent**	**3**
			B1	**Child**	**4**
			B2	**Child**	**2**
			B3	**Child**	**1**

B1	…	…	…	…	…
B2	…	…	…	…	…
B3	…	…	…	…	…
C	xxxx	7	X	Parent	3
			C1	Child	4
			C2	Child	4
C1	…	…	…	…	…
C2	…	…	…	…	…

Assuming the Node A is node X's first peer-to-peer node. To optimize its local topology, Node X will invoke a process in 4 steps as the following along its parent node direction. This one-way processing will greatly reduce the traffic for topology optimization.



Step 1: Based on information of neighbors, Node X retrieves the metrics from other by building virtual connections with its neighbors within 2-hops.



In this step, virtual connections are built by Node X to retrieve distance parameters along its parent node direction. "Tracecount" is a general technology of Internet to get this parameter as we mentioned previous. Then the process is moved to step 2 to build a set table to calculate based on the assumption that if Node X is connected to one of them, what is the mean distance to arrive all of them.

Step 2: Connection parameter table is built for further computation

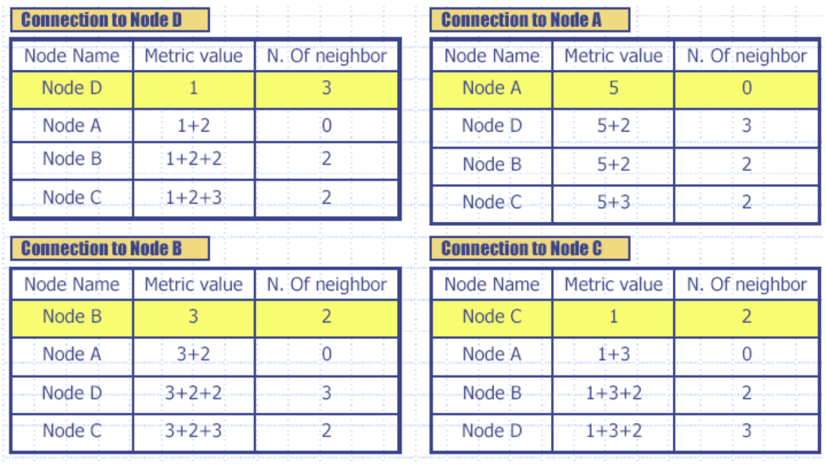
Points in step 2 is that when node number is calculated, the node not belong to the limited range (2-hops range) is also considered. This consideration means that the route we try to compute is not only the distance to each neighbors, but also associate with their further connections. This mechanism provides a method to know which route is the shortest to arrive most of the node in the peer-to-peer network.

The result of this example is:

f(X,A)= 78/11, f(X,B)=59/11,

f(X,C)=50/11, f(X,D)=40/11

Obviously, the connection with Node D will provide shortest route to Node X to arrive all of the nodes in its local area.



Connection to Node D

Node Name	Metric value	N. Of neighbor
Node D	1	3
Node A	1+2	0
Node B	1+2+2	2
Node C	1+2+3	2

Connection to Node A

Node Name	Metric value	N. Of neighbor
Node A	5	0
Node D	5+2	3
Node B	5+2	2
Node C	5+3	2

Connection to Node B

Node Name	Metric value	N. Of neighbor
Node B	3	2
Node A	3+2	0
Node D	3+2+2	3
Node C	3+2+3	2

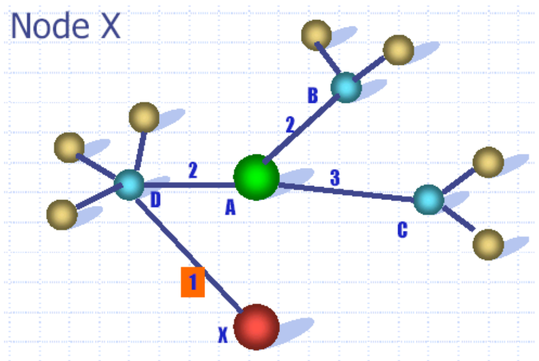
Connection to Node C

Node Name	Metric value	N. Of neighbor
Node C	1	2
Node A	1+3	0
Node B	1+3+2	2
Node D	1+3+2	3

Virtual connection table

Step 3: The Node with least value is selected and the connection is redirected to this node. Figure 11 shows the final result.

As the result, the Node X releases its connection with Node A and redirects its connection to Node D for better performance and less traffic consumption. Following figure shows the result.



3. Formula

The above computation can be expressed in the following formula

$$V_i = \frac{\sum_{i=0}^{n-1} (V_{Mi} \times N_i)}{\sum_{i=0}^{n-1} N_i}$$

Ni is the number of nodes it connected. *Vmi* means the distance from node to the destination set of nodes. *n* is the number of nodes in the local area. Vi is the mean value which represents the mean distance from current node (Node X) to one of other nodes in the peer-to-peer network.

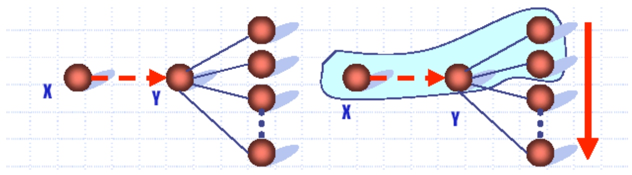
4. Considerations

4.1. Autonomous optimization based on local information

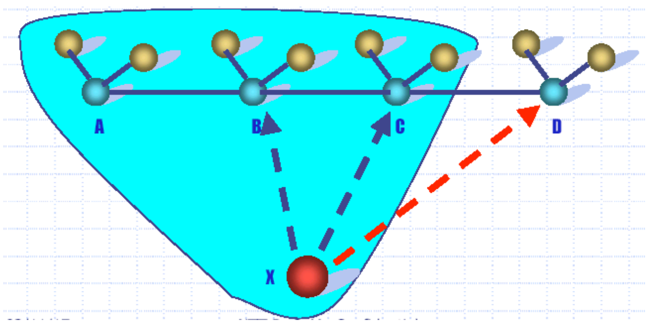
In our proposal, each node is an administrator and monitors its local topology. The process of topology optimization is invoked periodically and autonomously. Topology optimization behaviors of peer-to-peer nodes are completely independent as well as other functions. The optimization process of a peer-to-peer will not be interfered by other processes of topology optimization.

4.2. Scalability

Assuming the parent node has numerous neighbor nodes. To build virtual connection and retrieve all the metrics from those nodes is really time waste. For speedy topology optimization, we believe that building a list of neighbors based on their ability of connections is helpful. This process is shown in the following figure.



Another issue associated with scalability is also important. When a node is selected by the mean value of the distance metrics, this connection may not be the optimal route comparing to the connections with a node which is 3-hop far away the node. In this situation, an iteration processing is necessary to find the best one a node should redirect its connection to (see following figure while Node D should the best node for building connection).



This process can be described in the following steps.

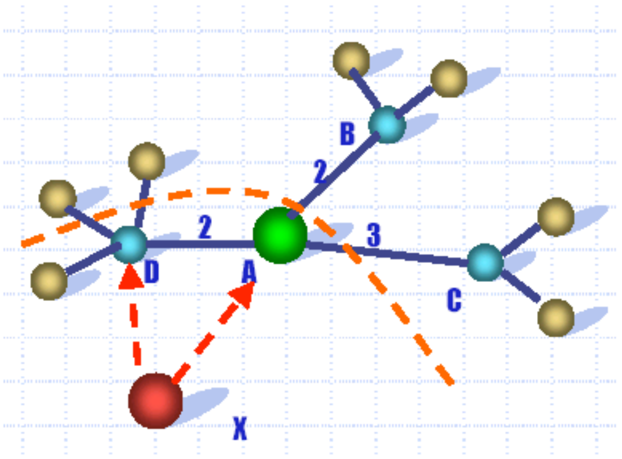
(1) Execute the topology optimization mechanism

(2) Get the result node. If it is the parent node, then finish

(3) Else set the result node as parent node, collect the neighbor information, go to (1)

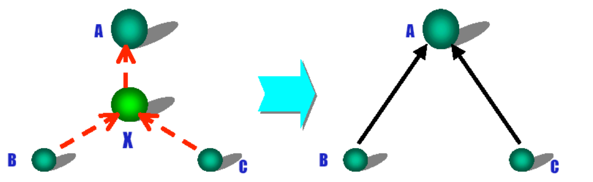
4.3. Applicability to heterogeneous networks

To be adaptive to the nodes in heterogeneous networks, where some nodes maybe unreachable and un-accessible, this mechanism can be simplified by ignoring those who can't answer its retrieving request on the Step 1 in a deterministic time interval. The imagination is showed in the following, where only Node D participates the computation of the optimization process.



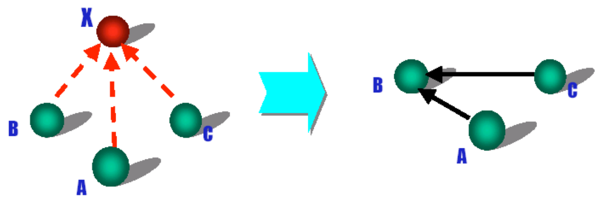
4.4. Adaptation to dynamic change of network

Recovery mechanism is independent of the network optimization mechanism. It is also proposed based on the same exchange node information that is broadcasted over 2-hops range. Current system provides each node the function to keep connection with its 1-hop neighbors. The "keepalive" message is sent periodically and respond is received. When respond of a neighbor node is not received during a designated time, we said this node is down or be attacked. And a mechanism for the node to rebuild connection with other nodes around it is invoked. The point of our proposal is a) the recovery mechanism is only executed along parent node direction, as well as network optimization mechanism. b) Connection request is always started from child nodes to parent node or weaker nodes to stronger node. Following figure is a simple example of this recovery process.

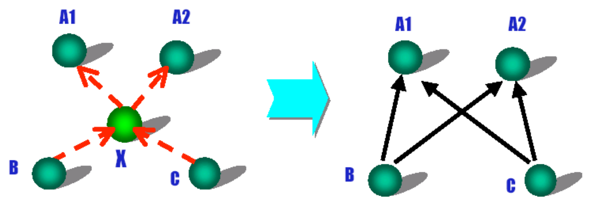


Node B and node C hold a list show that Node X has a parent node, Node A. During their existing, Node B and Node C monitor Node X by sending "keepalive" message to Node X periodically and get responds. When Node X fails to reply this message, Node B and Node C will start a recovery process using the table of node information to track the parent node of Node X. As a consequence, the Node C and Node B will rebuild their connections with Node A.

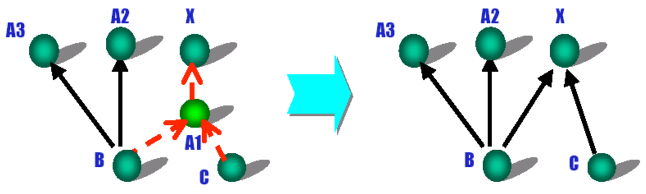
There are several special cases in this recovery mechanism. In case one, when the parent node is the root of the whole network, its child has to be connected to keep integrity of total network. The connections between its child peer-to-peer nodes can be rebuilt according to their maximum connection ability. Because each node holds a table including the neighbors within 2-hops, all of them understand the node which is strongest in this set of nodes. Consequently, the strongest node becomes the new parents of all other child nodes and waiting for their connections to recover the local network topology. See following figure for topology conversation:



In case 2, to support mesh topology while one node (Node X) has multiple parent nodes, the child nodes have to build same connection as their parent node had. Following figure shows the topology before the recovery process and after the process.



Case 3 is another case in the mesh topology. Node B has three parent nodes and one of them fails to answer. To protect the connectivity of local network, Node B has to recover its connection with A1 to Node X.


A3
A2
X
B
A1
C
A3
A2
X
B
C