Human-Centered Home Network Configuration

Design Specification – v2

Keith Edwards
Esther Yang

LISA, GATECH
August 2005
Contents

1. Introduction ................................................................. 1
   Complexities of Home Networking ...................................... 1
   Provisioning Box ........................................................... 2
   Environmental Assumptions ........................................... 2

2. PB Functionalities ......................................................... 4
   Functionalities .............................................................. 4
   Design Rationales in the Perspective of Functionality ........... 5
   Design Rationales in the Perspective of Human .................... 6

3. PB System Architecture ............................................... 8
   PB Components .......................................................... 8
   PB Operation Scenario .................................................. 13
   PB Client ................................................................. 15

4. Prototype & Evaluation Plan ........................................... 16
   Prototype Platform ....................................................... 16
   Implementation ........................................................... 17
   Evaluation ................................................................. 17
   Schedule ................................................................. 19

Appendix A. Implementation Details ................................. 20
1. Introduction

This document describes the high-level architecture of the Provisioning Box (PB) software system, defines the components that make up the system, discusses the interactions among components, and describes class-level implementation specification.

Complexities of Home Networking

Home broadband adoption is growing, and fueling an increase in home networking. However, despite obvious interest on the part of home users, concerns about complexity of home networking hinder its adoption. Studies reveal that people cite complexity of the technology as being the key impediment to adopting home networking technologies. Therefore, it is necessary to lower the human barrier to the adoption and use of home networking by reducing the complexity and increasing the usability of these technologies.

Earlier work [Beki: ECSCW2005] identified some of the sources of complexity, through ethnographic study of home network early adopters. One of the primary sources of complexity concerns correctly provisioning devices for the home network. This task involves configuring or adapting devices to the particular circumstances and context of a specific home network: configuring IP addresses, WEP keys, network SSIDs, and so forth. One particularly troubling aspect of provisioning is that it is inherently fragile – any change to the home network topology (installation of a second access point for example, or change of Internet Service Provider) has the potential to break the ability of existing devices on the network to communicate with one another, or with the broader Internet.

Currently, most network provisioning is done “by hand,” typically by a home user with above average networking knowledge. This use must understand the topology of the home network, and be able to translate his or her mental knowledge of this topology into settings appropriate for each individual device. Although a number of technologies (DHCP, ZeroConf discovery protocol) can help with certain aspects of provisioning, they clearly do not solve the problem.
**Provisioning Box**

Our solution to taming the complexity of home network provisioning is through a device called the Provisioning Box (PB). The PB is a special, privileged node on the home network, tasked with issuing network configurations to new devices, and to existing devices when the network topology changes. When a new device joins the home network, the PB authorizes the device, allocates an address to it, and introduces it to existing devices. Then, the PB propagates configuration changes onto the home network. The PB also detects networking problems and notifies users of those problems.

**Environmental Assumptions**

The PB relies on the existence of a networked home enabled by a home networking system that connects various information home appliances, mobile devices, sensors, and so forth. Some devices have their own processing power and memory, while others do not. Devices without processing power and memory may be connected to a home network, but those devices are controlled by another piece of hardware and/or software, called a proxy, that presents the device to the network and which itself contains both processing power and memory.

![Figure 1: A networked home environment](image-url)
These devices are inter-connected through a home networking technology. We adopt the wireless LAN, i.e., IEEE802.11, as a home networking technology because of its wide-use. The IEEE802.11 is operated in the infrastructure mode in which wireless devices associate with an access point and communicate with each other through that access point, instead of ad hoc mode.

Along with the IEEE 802.11, the home gateway enables data transfer between home appliances and access to the Internet, are requirements for end-users to use for various home network services. Home gateway is an interconnection device which plays the role of dividing and interfacing the subscriber network and the home network.

A subscriber network enables Internet access from home. It can be classified into wired network and wireless network according to the technology used and services offered. In a wired network, technologies using the telephone line such as xDSL or ISDN are widely utilized. Cable modem service using the cable TV network, optical cable, and power lines are also being utilized. In a wireless network, satellite mode and BWLL (Broadband Wireless Local Loop) mode are used.

The federation of these components enables users to use home network services that provide a means to improve the quality of life and resolve the information access disparity by offering enhanced information (home management, leisure, entertainment, education and etc.) regardless of time and place.

The PB is built on the home gateway although it can be built as a stand alone system.
2. **PB Functionalities**

The ultimate goal of the PB is to help home owners to build, maintain, and debug the home network in an easy and natural, often automatic way. To achieve this goal, the PB provides the simplicity of access, ease of administration, and support for sharing devices that are provided by a large monolithic home networking system while retaining the flexibility, uniform response, and control provided by various devices.

**Functionalities**

1. **Configuration** - The PB allows automatic client configuration made over location-limited channels such as infrared for a new device joining the home network. It assigns an address to the newly joined device in a DHCP manner after authentication process. It also sends the device the 802.11 configuration information (for example, the WEP key and the SSID), network configuration information (for example, network mask, default gateway, DNS server, and so on), and service configuration information (for example, default printer, net port for game box, firework, and so on) that are set for the home network. On the other hand, it should allow the device to leave the home network smoothly and automatically without leaving any unwanted state behind. Once the PB recognizes the device, it can save the device information in its repository for a while for next time like Internet Cookie. The PB should notify the topology changes to the home network to devices if necessary. Also, it propagates configuration changes onto the home network to existing devices if necessary. For example, introducing a new printer onto the network might cause the PB to tell existing nodes to change their default printer. More fundamental network changes can also be facilitated by the box. Changes of network keys, gateway IP addresses, and so forth can be propagated down to network nodes as alternate configurations before the changes take effect, allowing seamless continuity of connectivity after the change is made. Doing so again requires UI work that can allow users to indicate the desired state of the network in a way that makes sense to them.

2. **Authentication** – The PB prevents unauthorized devices or users from accessing the
home network. Before configuration process, it checks if a device is allowed to join the home network.

3. **Central repository** – The PB stores the information about the devices for lookup service and maintenance. The information varies ranging from device specific information to user-specific requirements.

4. **Lookup service** – The PB allows a device to find out about the presence and capabilities of other devices within the scope of the home network and to control those devices.

5. **Management** – The PB monitors the home network. If an error is detected, it recovers or troubleshoots to the failed node or a user.

6. **Recomposition** – The PB supports interconnections among arbitrary device types in the home network. It allows users to compose different services provided by multiple devices into one complex, seamless service on demand. To make this possible, the PB should play a role as a mediator between two different devices. This functionality will be added in the second step of the project.

7. **Proxy** – The PB performs functionalities such as filtering, transformation, etc, lying between non-IP devices and IP devices or between the home network and the Internet. This functionality will be added in the second step of the project.

**Design Rationales in the Perspective of Functionality**

**Simplicity:**
Jini and UPnP are distributed environments for devices to communicate with each other. They allow devices to automatically access the home network and use lookup services by collaboration between devices itself. While this distributed manner has advantages in the perspective of scalability or robustness, it makes the system complex. Therefore, we adopt the centralized scheme in which designates a single-point-contact for network installation, maintenance, administration, and user interface activities. And the single-point-contact is the PB.

**Manageability:**
Basically, our goal of the PB is to require no system administrator. Given the lack of system administrators in the home, the network must operate with little or no user intervention. How, then, will we design technologies for the smart home that require no on-site expert? In the telephone and cable TV networks, most of the “intelligence” in the system resides in the network itself. Home contains only the most simple and minimal “front end” functionality needed to access the network. This expanding functionality is available because the sophistication of the back-end network is increasing.

**Interoperability:**
The home network allows all appliances to communicate with one another. Thus, it needs a technology that can seamlessly integrate assorted devices into a monolithic communication network. Although no prior agreement on standard protocol and interfaces are desirable, it is very hard to integrate various devices at run time without a priori agreement. The PB should minimize the requirement of priori agreement and standard, for example, by sending service objects and interfaces at run time.

**Extensibility:**
The PB should recognize new types of devices without replacing the existing PB with the new one for upgrade.

**Thin-client:**
Difference in technological approaches: in current connected domestic technologies such as telephone and digital TV, the bulk of functionalities is placed in the network, not in the device itself. In this “utility” model, the client technologies are shielded from upgrades and enhancements in the network, and yet can take advantage of new functionality when available. We should balance of intelligence between the edges and the center of the network. The PB tries to require less intelligence from the edges while pushing more intelligence to the center of the network.

**Design Rationales in the Perspective of Human**

**UI:**
Although the underlying technologies is advanced and hidden from users, mainstream adoption of the home network will only happen when the network user interface masks the complexity of the underlying technology, offering ease-of-use comparable to traditional domestic appliances like telephones and VCRs as opposed to the PC. Uses should feel the
PB the stable and compelling routines of the home, rather than external factors.

Balance of invisibility and visibility:
We argue that our goal is to hide the underlying networking technology from users to get rid of the complexity of technologies. However, we believe hiding is not an all mighty solution for all cases. Sometimes, users need to know about how technologies work. Therefore, our other side goal is to help users understand the home network. Regardless of the overall model chosen, occupant users will still have some administration that they will have to do, simply because not all of the dynamics of the home can be known by the developer of the appliance, or the owner of the utility. The particular ways in which individual devices are used by members of the home, may need to be reflected in configurations, security parameters, and device interactions that can be only be implemented by the owners of those devices – not some external third party. Without the ability to understand the whole network, installing, adding, reconfiguration, and debugging are virtually impossible. For example, the intangible models of connectivity that wireless technologies bring must be learned. How will occupant users built up a model of how to control, use, and debug technologies that will interact with one another in the environment? A complex design challenge is to provide affordances to help users understand the technology. While UPnP targets “invisible” networking in which the underlying networking technologies are totally hidden to users, our other side goal is to help users understand the technologies that they need to know to control, user, and debug technologies. Another problem is one of intelligibility in the fact of radical- and perhaps unexpected-connectivity. The challenge for ubiquitous computing is to help home owners understand their accidentally smart homes by providing insights into what these devices can do, what they have done, and how we control it.
3. **PB System Architecture**

**PB Components**

The PB composes of seven modules and one repository.

![The PB internal architecture](image)

**Figure 2: The PB internal architecture**

**Authentication:**
The Authentication module performs authentication process before a new device’s joining the home network, according to authentication policies. Authentication policy is set by home owners. It generates a root key pair and root certificate. When a new device joins the network, it determines whether this device is allowed to access the home network or not according to
authentication policies. If the device is allowed, it sends the root key and certificate to the device. Only the devices with the root key and certificate can communicate within the home network.

**Configuration:**
The Configuration module performs configuration process when a device joins or leaves the home network. To a newly joined device, which passed the authentication process, this module sends a set of configurations. These may include simple networking parameters such as addresses, SSIDs, default routers, security parameters such as WEP keys, and service configuration information. In complex networks, it may issue multiple possible configurations along with an indication of when each is to be used. So, for example, the box may issue different configurations appropriate for wired and wireless networks; depending on which network the device ultimately finds itself, it may select one configuration over the other.

To assign an address to a new device, the Configuration module maintains an address pool from which it obtains an available address for a newly joined device. When the device leaves the home network, the module reclaims the assigned address from the device. An address format may be an IP address format, which is interoperable with the Internet address, or a locally defined address, which is non-routable. In the former case, the module need to interact with the DHCP server, which is a designated one for the network to which the home network subscribes, to request an IP address on behalf of the new device. In the latter case, it should perform address translation process. As a first step, we consider only IP address format.

<table>
<thead>
<tr>
<th>Key: Address</th>
<th>Status: Busy, Available</th>
<th>Used by: Device_Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>X.X.X.1</td>
<td>Busy</td>
<td>Device_1</td>
</tr>
<tr>
<td>X.X.X.2</td>
<td>Available</td>
<td></td>
</tr>
<tr>
<td>X.X.X.N</td>
<td>Busy</td>
<td>Device_k</td>
</tr>
</tbody>
</table>

The configuration information is set automatically if possible, for example, the default router, NETMASK, broadcast address, and so on. Otherwise, it is set by home owners if manual configuration, for example, the WEP key and the SSID, is required.

The service configuration information includes application-level configuration such as the
default printer, net port for gaming, and so on.

Once enrolled, the device will be able to be automatically recognized by the PB from the next time if the device or the PB wants it.

The Configuration module propagates configuration changes onto the home network to existing devices if necessary. For example, introducing a new printer onto the network might cause the PB to tell existing nodes to change their default printer. More fundamental network changes can also be facilitated by the box. Changes of network keys, gateway IP addresses, and so forth can be propagated down to network nodes as alternate configurations before the changes take effect, allowing seamless continuity of connectivity after the change is made. Doing so again requires UI work that can allow users to indicate the desired state of the network in a way that makes sense to them.

**Description:**
The Description module obtains the descriptions of a device. After the PB has discovered a device, it knows very little about the device. For the PB to learn more about device and its capabilities, or to interact with the device, the PB must retrieve the device descriptions. The descriptions include the vendor-specific, manufacturer information, and services provided by the device, and whatever like icon. The vendor-specific, manufacturer description includes the model name and number, serial number, manufacturer name, etc. The service description contains the service type, the service id, and the service description. Device and service descriptions are in XML syntax and based on a standard UPnP Device and Service Template, respectively.

These descriptions, along with the address assigned to the device, is saved in the Device repository, which is a central repository storing all device information within the home network.

**Lookup service:**
This module provides two commands, *search* and *advertise*. Search allows a device to discover services in the home network. Advertise allows a new device to advertise its services to other nodes on the network.

If a device wants to use a certain service, the device sends a LOOKUP message to the PB to
get references of the devices that provide that service. The Lookup service module processes this LOOKUP message and returns the result including appropriate device descriptions if any or NONE if nothing.

The device connects directly to one or more devices to use the service and get the services objects. A service object contains the definition of actions, arguments, and state variables and their data type. The device displays the supported functions on menus, usually on the screen. Through the menus, users send commands to the device to use the services.

If a new device wants to advertise itself to the nodes on the network, it sends the ADVERTISE message to the PB. Then, the PB rebroadcasts the ADVERTISEMENT message to other nodes on the network.

**Management:**
The Management module monitors and controls the home network. This module periodically broadcast an STATUS message to nodes. If a node receives an STATUS message from the PB, it replies. If the PB does not hear from a device within a predefined timeout several times, it assumes that the device is not available any more and reclaims all configuration information assigned to the device and deletes it from its active device list. This does not mean that the PB deletes the device from its repository.

Also, this module monitors the status of all devices and informs users of the status if necessary. For example, as a signal problem, such as loss of communication with a particular access point, because the PB knows the “downstream” devices that depend on this access point, it can reveal to the home user a more holistic picture of why certain devices may not be functioning properly. This will require UI design and evaluation work, to present a picture of network problems in a way understandable and actionable by end users, as well as techniques for generating appropriate troubleshooting guidelines based on discrepancies between the ideal network model and the actual detected network state.

**Device repository:**
The repository stores all information about devices in the home network.
**Communication:**
The Communication module, which lies at the most bottom of the PB, sends and receives the PB messages to and from the other devices in the network over infrared and IEEE 802.11. The PB is built on top of the standard TCP/IP stack in case of IEEE802.11.

**UI:**
The UI module is simple command and display. This module would be replaced with the module developed in the human-centered home network visualization project.
PB Operation Scenario

Figure 3: The PB operation scenario
**Step 0 - Initialization:**
The PB initializes all modules. It gets its own IP address, generates a root key pair and root certificate, and so on.

**Step 1 - Bootstrapping:**
Step 1 is bootstrapping. Obviously, the PB cannot communicate with nodes using IP until those nodes have been properly provisioned already. Thus, we rely on an out-of-band communication mechanism for the initial provisioning step: when a new device is brought into the home, it engages in a short-range exchange with the PB, using a mechanism such as IrDA. Such a channel provides a number of inherent benefits. First, because it is range-limited, it naturally supports intuitive gestural interactions (touch the PDA to the PB to indicate that it is the PDA that is to be configured); second, such channels can operate in the absence of the explicit configuration required by IP. The result is a channel that can be made intuitive and robust for the initial provisioning step.

During this step, the new device provides descriptive information about itself – manufacturer, icon, functionality, whatever UPnP provides it implements to the PB. This allows the PB to build up a model of the devices that are actually on the home network, as opposed to having to rely on some fragile discovery protocol to try to intuit this information. At the same time, the PB issues a set of configurations to the new device.

After Step 2, the user deploys the new device onto the home network. At this point, the PB box is tasked with ensuring that it can communicate with the device, and records its position on the home network, once again building up its model of the home network topology.

**Step 3 – Lookup:**
A new device asks the PB about a certain service. The PB returns the reference of the device that provides the service.

**Step 3 – Management:**
The PB periodically broadcast an STATUS message. The device replies to the PB.
PB Client

The PB client composes of three modules.

**PBCClient:**
The PBCClient is responsible for all functionalities of the PB client, counteracting to the PB job of configuration, description, lookup, and management. It get user input from the UI module, processes messages, and responds to those messages if necessary.

**UI and Communication:**
The UI and the Communication modules are same as those of the PB.

---

**Figure 4: The PB client internal architecture**
4. Prototype & Evaluation Plan

We build a functional prototype and perform evaluation exercises aimed to validate usability criteria.

Prototype Platform

Although the PB would ultimately find its way into the home router or set-top box, we can most easily prototype the system as a standalone computer, thus obviating the need for any special hardware. Client-side software would likewise be developed for the PC platform; we can thus simulate things like network connected appliances through a proxy approach, using a small PC or PDA connected to both the appliance and to the network.

Figure 5: The prototype platform
Implementation

- Platform: Windows XP/CE
- Language: PB – Java/C++ (infrared communication)
  PB client – Java/C++ (necessary)

Evaluation

We undertake a series of usability studies both to test if our system actually makes it easier to set up a home network, and to obtain feedback to iteratively improve our design.

Our usability tests had two objectives. First, we want to know whether our system allows users to easily connect to a home network. Our second objective is to compare our solution against a commercially available alternative. We will pick a commercially available access point based on several criteria: it should be designed for end users, not enterprises; it should be a market leader; and, enrollment should occur on the same operating system as our solution (to avoid confounding variables by switching platforms).

Usability tests assign tasks to users to see whether they can complete them successfully and to learn what errors they make. The task we chose is to ask a user to connect a PDA to a home network. Through practical deployments of our client software is done on a wide variety of PDA hardware and wireless network cards, we asked all participants to use the same PDA in order to limit setup time and variability in our quantitative studies. The PDA pre-loaded with the setup software for both PB and the commercial AP. In both cases, the setup software provided as “wizard”-style interface for the user.

Study Design:
The participants are family members, including men, women and children. We select ten families. To prevent experimental biases, each phase of the study was counterbalanced such that half the participants (three male and three female) joins the home network using the PB, then using the alternative. The other half of participants (three male and three female) joins the home network using the alternative, then the PB.

Qualitative Data Collection:
Qualitative data related to user interaction issues associated with the PB is gathered using:
- **Pre-trial questions** – The participants are asked how much they are familiar with the home network, and how comfortable they are with the home network devices.

- **Post-trial questions** – The participants are asked to make short answers regarding the advantages, disadvantages, ease of use, and preference for the method of the home network configuration work.

<table>
<thead>
<tr>
<th>Post-trial questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. I understand how to use the PB.</td>
</tr>
<tr>
<td>2. I understand how to use the alternative.</td>
</tr>
<tr>
<td>3. I can join the home network easily using the PB.</td>
</tr>
<tr>
<td>4. I can join the home network easily using the alternative.</td>
</tr>
<tr>
<td>5. I can understand the home network topology using the PB.</td>
</tr>
<tr>
<td>6. I can understand the home network topology using the alternative.</td>
</tr>
<tr>
<td>7. I like using the PB.</td>
</tr>
<tr>
<td>8. I like using the alternative.</td>
</tr>
<tr>
<td>9. I would feel comfortable using the PB at home.</td>
</tr>
<tr>
<td>10. I would feel comfortable using the alternative at home.</td>
</tr>
</tbody>
</table>

**Expected Results:**
The PB can improve the user experience in the home network by facilitating significantly easy and natural way of network configuration, administration, and repair.
**Schedule**  
(Year: 2005)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Design of PB &amp; client</td>
<td>Implementation</td>
<td>Evaluation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix A. Implementation Details

PB and PB Client Implementation Architecture

![Diagram of PB components and their interactions]

**Figure 6**: Interactions among the PB components
Figure 7: Interactions among the PB client components
PB Messages

Message types

ANNOUNCE
- Sender: PB
- Payload: <none>
- Function: The PB periodically broadcasts this to notify nodes within its infrared transmission range of its existence.

ATTACH:REQ
- Sender: PB client
- Payload: authentication permit such as password
- Function: The client requests attachment to the home network.

ATTACH:ACK
- Sender: PB
- Payload: configuration information in case of ACCEPT, deny reason in case of DENY
- Function: The PB sends configuration information or deny reason to the client.

DESCRIPTION:REQ
- Sender: PB
- Payload: UPnP Device/Service Template
- Function: The PB requests the device description to the client.

DESCRIPTION:ACK
- Sender: PB client
- Payload: Device/Service descriptions
- Function: The client sends its device description to the PB.

LOOKUP:REQ
LOOKUP:ACK
STATUS:REQ
STATUS:ACK
Class PBMessage
This class marshalls and unmarshalls the PB messages.

class PBMessage {
external:
    char* marshall (PBMessage);

    PBMessage unmarshall (char*);
}

< coming soon>
PB main thread

The main thread is the core of the PB system. It initializes the PB system and all modules, processes messages from users and home network devices, and responds to those messages.

init ()
This method initializes the PB system.

initModules ()
This method initializes CommThread and all the modules.

uiCallback (OP, user input)
This method is called by UI when the UI has inputs from a user.

commCallback (message)
This method is called by CommThread when CommThread receives a message. It parses the PB messages and sends those message to the appropriate module according to the message type.

switch (messageType)
   case ATTACH:REQ:
      call Authentication module;
      if (Authentication == ok)
         call Configuration module;
         send ATTACH:ACCEPT;
      else
         send ATTACH:DENY;
   case DETACH:REQ:
      call Configuration module;
   case DESCRIPTION:ACK:
      call Description module;
   case LOOKUP:REQ:
      call Lookup module;
      send LOOKUP:ACK;
   case STATUS:ACK:
      call Management module;
}
**CommThread**

This thread opens all IrDA and TCP stream sockets for sending and receiving messages over infrared and WLAN, and listens to those sockets. If a message arrives, it sends it up to the PB main thread. Also, it provides the PB main thread with the interface to send messages.

`init()`  
This method opens all sockets.

`run()`  
This method is listening to sockets. If it receives a message, it sends it up to the PB main thread.

**PB Client main thread**

The PB client main thread is similar to the PB main thread except that it does not have modules but process messages by itself.

`init()`  
This method initializes the PB client system.

`uiCallback (OP, user input)`  
This method is called by UI when the UI has inputs from a user.

`commCallback (message)`  
This method is called by CommThread when CommThread receives a message. It parses the PB messages, and processes and responds to those messages.
Authentication Module

Authentication policies are stored in an “authentication.txt” file. Each authentication policy constitutes of a pair of policy name and value. For example,

\[ \text{policy} = \{ \text{“password”, “myhome”} \} \]

indicates that only users who inputs the password, “myhome”, can access the home network.

Each policy can be edited from UI or by direct file access.

Class Authentication
This class defines the methods that are called for authentication process.

class Authentication {
    external:
        init (); // initialize authentication module; read policy file
        addAttr (attr_name); // add new attribute
        deleteAttr (attr_name); // delete attribute
        modifyAttr (OP, attr_name, value) // modify attribute according to OP; OP={ADD, DELETE}
        check (strr_name, value); // check authentication of user

    internal:
        readFile (); // read authentication policy file
        writeFile (); // write to authentication policy file
}

Configuration Module

This module obtains all configuration information for a newly joined device. Configuration information is stored in a “config.txt” file.

Class Configuration

external:

    init (); // initialize configuration module; read configuration file
    getIPaddr (); // get an IP address
    getNetConfig (); // get an network configuration
    getWLANConfig (); // get 802.11 configuration
    getServiceConfig (); // get service level configuration
    composeConfig (); // compose all configuration information

internal

}
Description Module

This module parses UPnP compliant device and service descriptions and save them to the device repository, a “repository.txt”.

Class Description

class Description {

external:
    init ();  // initialize configuration module; read configuration file
    parseXML ();  // parse XML
    save ();  // save description to repository
    find ();  // get 802.11 configuration

internal:
}

Lookup service module

This module provides lookup service.

Class Lookup

class lookup {

  external:

    init();

    find();

  internal

}
Management module

This module provides lookup service.

Class Management

class Management {
    external:
        init();
    internal
}
Communication module

This module provides communication service.

**Class Socket**
class Socket {
  external:
    open ();
    send ();
    receive ();
    close ();

  internal
}

**Class StreamSocket**
class IrStreamSocket {
  external:
    listen ();
    accept ();
    connect ();

  internal:
}

**Class IrStreamSocket**
class IrStreamSocket {
  external:
    listen ();
    accept ();
    connect ();

  internal:
}