

## 第五章 系統實驗與效能評估

### 5.1 實驗環境介紹

為了實驗家用網路開道服務架構，在家庭網路下需先啟動 Jini Lookup Service、SLP Daemon 做為基本的 Jini 網路和 SLP 網路目錄服務。另外，我們分別設計了 3 個不同服務搜尋機制下的資訊家電模擬程式，來模擬家庭網路下的資訊家電裝置。而家用開道服務器的輔助服務：Jini Service Agent、SLP Service Agent、Repository 也要分別啟動。最後啟動 UPnP 開道服務、SLP 開道服務。另外，本實驗主要在 2 台 700MHz CPU，384M RAM，OS 為 WinXP 的電腦上進行。如圖 5.1。

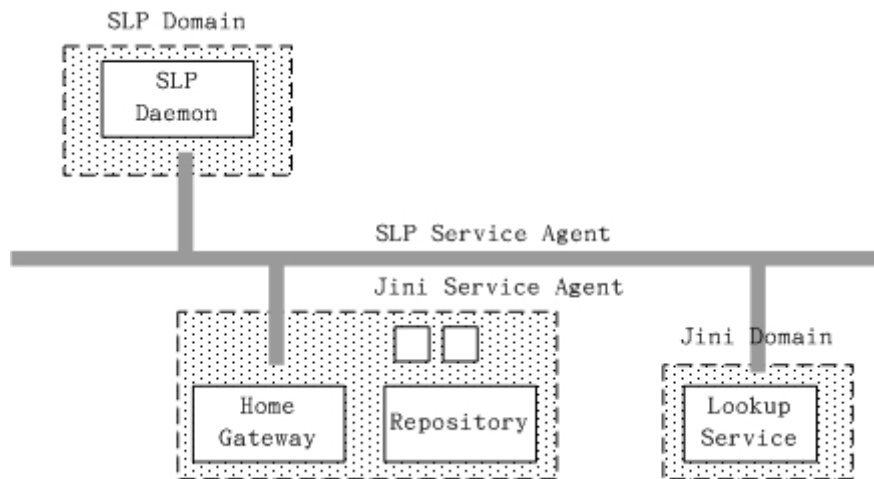


圖 5.1 家庭網路實驗環境

實驗環境成員包括：

1. Jini Lookup Service：使用 Jini Technology Starter Kit v.1.2.1 所提供的 Jini Lookup Service 啟動程式，分別行啟 RMID(Remote Method Invocation Daemon)、Web Server、Lookup Server(Reggie)，將 Jini 所需的基本環境啟始。如圖 5.2。

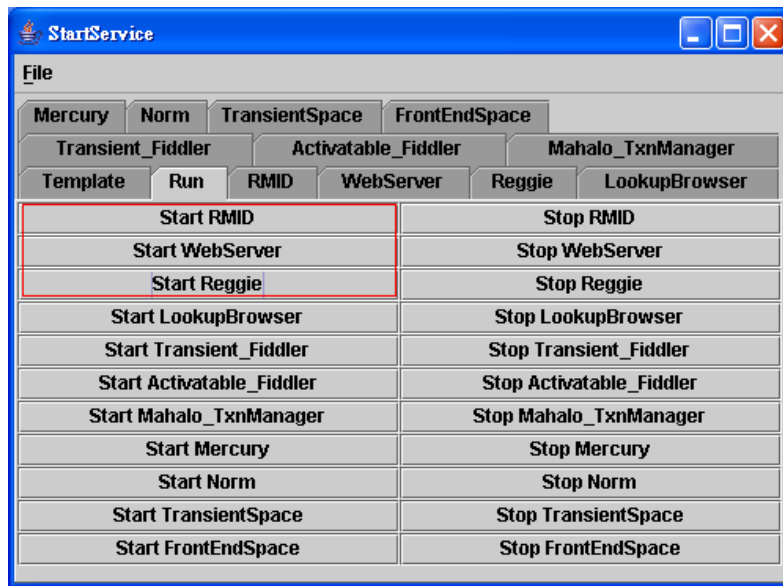


圖 5.2 Jini Lookup Service 啟動程式

2. SLP Daemon：使用 Solers OpenSLP 內附的 SLP Daemon，並啟動它，在背後提供 SLP SA 服務資訊的紀錄。如圖 5.3。

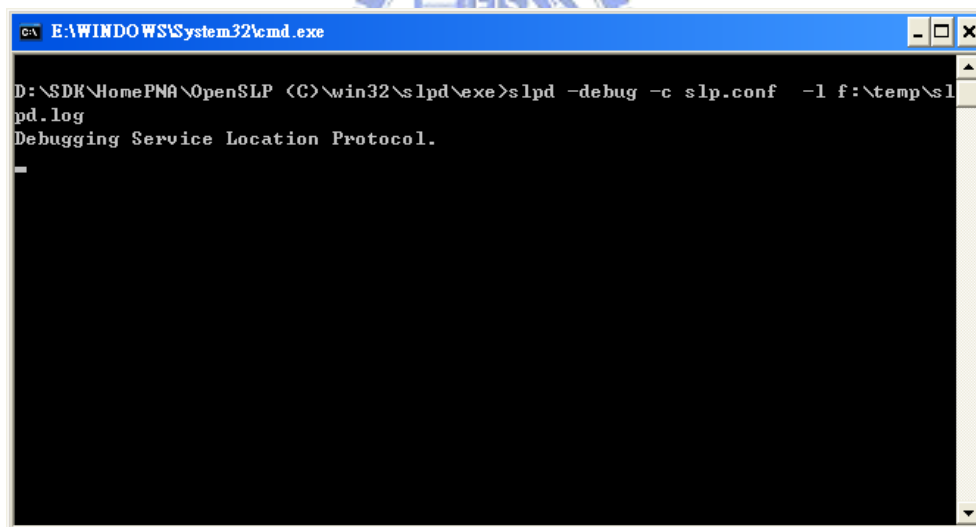


圖 5.3 啟動 SLP Daemon

3. Repository：本實驗主要是使用資料庫 MySQL 來扮演 Repository 的角色，用以儲存 Jini、SLP Service Agent 先行查詢的服務資訊。如圖 5.4。

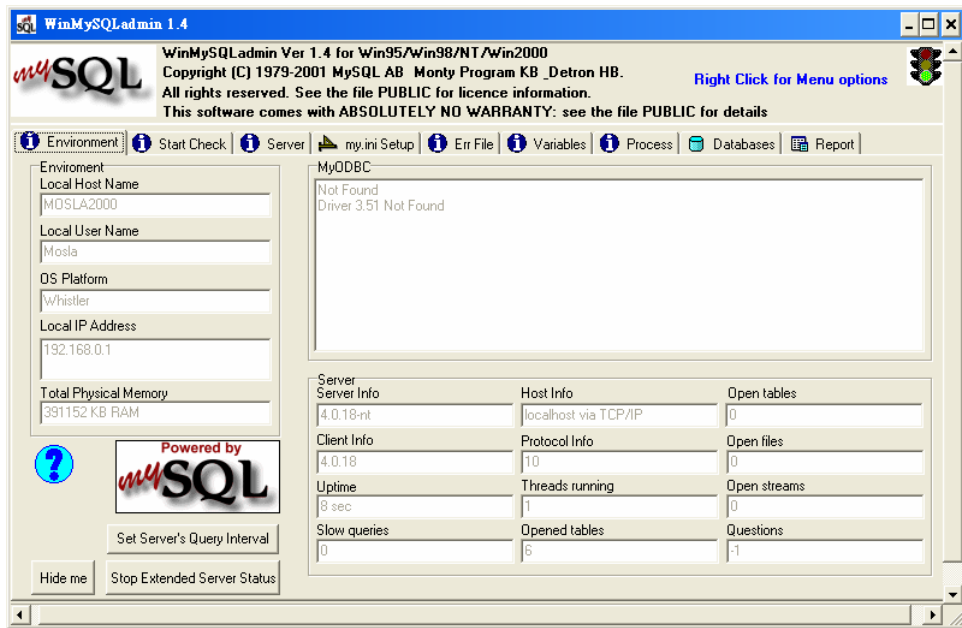


圖 5.4 啟動 MySQL

4. Jini Service Agent：使用家用閘道器提供的程式啟動程式，如圖 5.5，先行啟動各個 Service Agent，進行週期性的服務查詢，並將服務資訊紀錄於 Repository，供各閘道服務使用。在此啟動 Jini Service Agent，如圖 5.6。



圖 5.5 家用閘道器啟動程式

```

E:\windows\system32\cmd.exe - startjini searcher.cmd

I:\MOSLA\PROJECT\Java\HomePNA\Bridge\classes>startjini searcher.cmd
Using CLASSPATH: .;d:\jdk\mysql.jdbc-2.0.4\mm.mysql-2.0.4-bin.jar;D:\JDK\JINI1.2.1\lib\jini-core.jar;D:\JDK\JINI1.2.1\lib\jini-ext.jar
JiniSearcher init..
JDBC Connection
Connect Successfully
JINISERVICES: 0 records affected !
JINIMETHODS: 0 records affected !
! release.
Disconnect Successfully
New service found: 57a8704e-c4e3-4ad3-946e-e3490f0ca2f2
New service found: 1a4b9dcc-f6c6-4031-9369-89925a7cd4bb

Name = HelloWorldImpl
Manufacturer = NCTU IIM
Vendor = Mosla
Version = 0.1
Model = Jini Home
Serial Number = 1

Class Name = org.iin.jini.helloworld.HelloWorldImpl_Stub
String getMsg()
void setMsg(String arg1)
JDBC Connection

```

圖 5.6 啟動 Jini Service Agent

5. SLP Service Agent：啟動 SLP Service Agent，如圖 5.7。

```

E:\windows\system32\cmd.exe - startslp searcher

I:\MOSLA\PROJECT\Java\HomePNA\Bridge\classes>java -classpath "I:\MOSLA\PROJECT\Java\HomePNA\Bridge\classes;D:\JDK\JINI1.2.1\lib\jini-core.jar;D:\JDK\JINI1.2.1\lib\jini-ext.jar;D:\JDK\JINI1.2.1\lib\jini-examples.jar;D:\SDK\HomePNA\Siemens uPNP\upnp100.jar;H:\JBUILDER9\lib\jdom.jar;H:\JBUILDER9\lib\xercesImpl.jar;H:\JBUILDER9\lib\antlrParserAPIs.jar;D:\JDK\Log4j\1.2.8\dist\lib\log4j-1.2.8.jar;D:\SDK\HomePNA\OpenSLP\OpenSLP.jar;D:\SDK\HomePNA\JavaBluetooth\JavaBluetooth.jar;D:\JDK\JavaConn2.0\conn.jar;D:\JDK\MySQL.JDBC-2.0.4\mm.mysql-2.0.4-bin.jar;H:\JBUILDER9\thirdparty\junit3.8\junit.jar;H:\JBUILDER9\lib\unittest.jar;H:\JBUILDER9\lib\jbc1.jar;H:\JBUILDER9\lib\dx.jar;H:\JBUILDER9\lib\beandt.jar;D:\JDK\JDK1.4.1\jre\lib\charsets.jar;D:\JDK\JDK1.4.1\jre\lib\jce.jar;D:\JDK\JDK1.4.1\jre\lib\jsse.jar;D:\JDK\JDK1.4.1\jre\lib\rt.jar;D:\JDK\JDK1.4.1\lib\dt.jar;D:\JDK\JDK1.4.1\lib\tools.jar;D:\JDK\JDK1.4.1\jre\lib\ext\comm.jar" org.iin.slp.searcher.SLPSearcher

SLPSearcher init..
JDBC Connection
Connect Successfully
SLPSERVICES: 2 records affected !
SLPMETHODS: 10 records affected !
! release.
Disconnect Successfully
service:slpSA1
  service:slpSA1://192.168.0.1:501
  Name=(java.lang.String)SA1

```

圖 5.7 啟動 SLP Service Agent

6. UPnP 開道服務：啟動 UPnP 開道服務，如圖 5.8。

```

E:\windows\system32\cmd.exe - startupnmpgrserver

I:\MOSLA\PROJECT\Java\HomePNA\Bridge\classes>startupnmpgrserver
Using CLASSPATH: "D:\SDK\HomePNA\Siemens uPNP\upnp100.jar";d:\jdk\mysql.jdbc-2.0.4\mm.mysql-2.0.4-bin.jar;;D:\JDK\JINI1.2.1\lib\jini-core.jar;D:\JDK\JINI1.2.1\lib\jini-ext.jar
Siemens UPnP Stack (ControlPoint) U1.0.008 initiating...
Siemens UPnP Stack (DeviceHost) U1.0.008 initiating...
WebServer listening on port 80
Using Webdirectory I:\MOSLA\PROJECT\Java\HomePNA\Bridge\classes\www\
Received service id
Java2D Direct3D usage forced on by J2D_D3D env
JDBC Connection
Connect Successfully
JINISERVICES: get 1 record!
1 release.
Disconnect Successfully
JDBC Connection
1 use.
Connect Successfully
JINIMETHODS: get 2 record!
1 release.
Disconnect Successfully
Wait..
Discovered LUS:
URL: jini://mosla2000/

```

圖 5.8 啟動 UPnP 閘道服務

7. SLP 閘道服務：啟動 SLP 閘道服務，如圖 5.9。

```

E:\windows\system32\cmd.exe - startslpgrserver

I:\MOSLA\PROJECT\Java\HomePNA\Bridge\classes>startslpgrserver
Using CLASSPATH: "D:\SDK\HomePNA\OpenSLP\OpenSLP.jar";d:\jdk\mysql.jdbc-2.0.4\mm.mysql-2.0.4-bin.jar;;D:\JDK\JINI1.2.1\lib\jini-core.jar;D:\JDK\JINI1.2.1\lib\jini-ext.jar
0 [main] DEBUG net.slp.traceMsg - Sending: ServiceRequest: Type: service:directory-agent Filter:
0 [main] DEBUG net.slp.traceMsg - Sending: ServiceRequest: Type: service:directory-agent Filter:
Received service id
3015 [main] DEBUG net.slp.traceMsg - Sending: ServiceRequest: Type: service:directory-agent Filter:
3015 [main] DEBUG net.slp.traceMsg - Sending: ServiceRequest: Type: service:directory-agent Filter:
6009 [main] DEBUG net.slp.traceMsg - Sending: ServiceRequest: Type: service:directory-agent Filter:
6009 [main] DEBUG net.slp.traceMsg - Sending: ServiceRequest: Type: service:directory-agent Filter:
JDBC Connection
Connect Successfully
JINISERVICES: get 1 record!
1 release.
Disconnect Successfully
6580 [main] DEBUG net.slp.traceMsg - Sending: ServiceRegistration: URL: service>HelloWorldImpl://192.168.0.1:500 Attributes: ClassName=(java.lang.String)org.ii

```

圖 5.9 啟動 SLP 閘道服務

因此，分別啟動 Jini Look Service、SLP Daemon、家用閘道服務的 Service Agent 和各閘道服務、Repository 後，家用閘道器便已開始提供整合服務。

## 5.2 實驗方法

這裡模擬 3 個資訊家電設備，分別支援 Jini、UPnP、SLP 網路，它們分別提供基本的服務，我們再使用不同服務搜尋網路下的客戶端進行搜尋、呼叫，測試家用閘道器中各網路閘道服務是否可以正確的轉換和處理服務。

1. 資訊家電模擬程式初始狀態：首先介紹 3 個資訊家電模擬程式的初始狀態和它們所提供的服務。

### ● Jini 果汁機資訊家電模擬程式

Jini 果汁機提供 3 個主要功能的控制，包括電源、振動強度、搖動速度設定。可供呼叫的 method 共有下列 7 個：設定電源、取得電源狀態、設定電源和振動強度、設定振動速度、取得振動速度狀態、設定搖動速度、取得搖動速度狀態。

```
public interface BlenderDevice
    extends Remote {
    public void setPower(int power) throws RemoteException; // 設定電源
    public int getPower() throws RemoteException; // 取得電源狀態

    public void setPowerPulse(int power, int pulse) throws RemoteException; // 設定電源和振
                                                                    動強度

    public void setPulse(int pulse) throws RemoteException; // 設定振動速度
    public int getPulse() throws RemoteException; // 取得振動速度狀態

    public void setSpeed(int speed) throws RemoteException; // 設定搖動速度
    public int getSpeed() throws RemoteException; // 取得搖動速度狀態
    }
```

啟動 Jini 果汁機模擬程式後，便會自動跟 Jini Lookup Server 註冊，初始的狀態為電源 Off、振動強度 1、搖動速度 1。模擬程式的資訊會隨著被呼叫更新畫面。Jini 果汁機模擬程式提供 Pulse、Speed、On 三個按鈕可供使用者更改設備資訊。如圖 5.10。



圖 5.10 支援 Jini 網路的果汁機資訊家電模擬程式

- UPnP 電燈泡資訊家電模擬程式

UPnP 電燈泡僅提供 1 個功能控制，電源。可供呼叫的 method 共有下列 2 個：設定電源狀態、取得電源狀態。

```
public interface SwitchPower {
    public void setTarget(boolean newStatus); // 設定電源狀態，0：關閉，1：打開
    public boolean getStatus(); // 取得電源狀態
}
```

啟動 UPnP 電燈泡模擬程式後，再按下 Announce 按鈕才會進行 Advertise 於 UPnP 網路上，初始的狀態為電源 Off，畫面上 On 按鈕即為電源開關。如圖 5.11。



圖 5.11 支援 UPnP 網路的電燈泡資訊家電模擬程式

- SLP 印表機資訊家電模擬程式

SLP 印表機提供 3 個主要功能的控制，包括電源、列印品質、列印頁數設定。可供呼叫的 method 共有下列 6 個：設定電源狀態、取得電源狀態、設定列印品質、取得列印品質狀態、設定列印張數、取得列印張數狀態。

```
public interface PrinterDevice {  
    public void setPower(boolean power);           // 設定電源狀態，0：關閉，1：打開  
    public boolean getPower();                     // 取得電源狀態  
  
    public void setQuality(int quality);           // 設定列印品質  
    public int getQuality();                       // 取得列印品質狀態  
  
    public void setPage(int page);                 // 設定列印張數  
    public int getPage();                         // 取得列印張數狀態  
}
```

啟動 SLP 印表機模擬程式後，便會自動跟 SLP Daemon 註冊，初始的狀態為電源 Off、列印品質 1、列印張數 1。模擬程式的資訊會隨著被呼叫更新畫面。SLP 印表機模擬程式提供 Quality、Page、On 三個按鈕可供使用者更改設備資訊。如圖 5.12。



圖 5.12 支援 SLP 網路的印表機資訊家電模擬程式

- UPnP 客戶端模擬程式

UPnP 客戶端模擬程式是一個 UPnP 服務的一般性控制程式。如圖 5.13。它提供所有設備查詢、依設備種類、服務種類、服務編號進行搜尋。如圖 5.14。



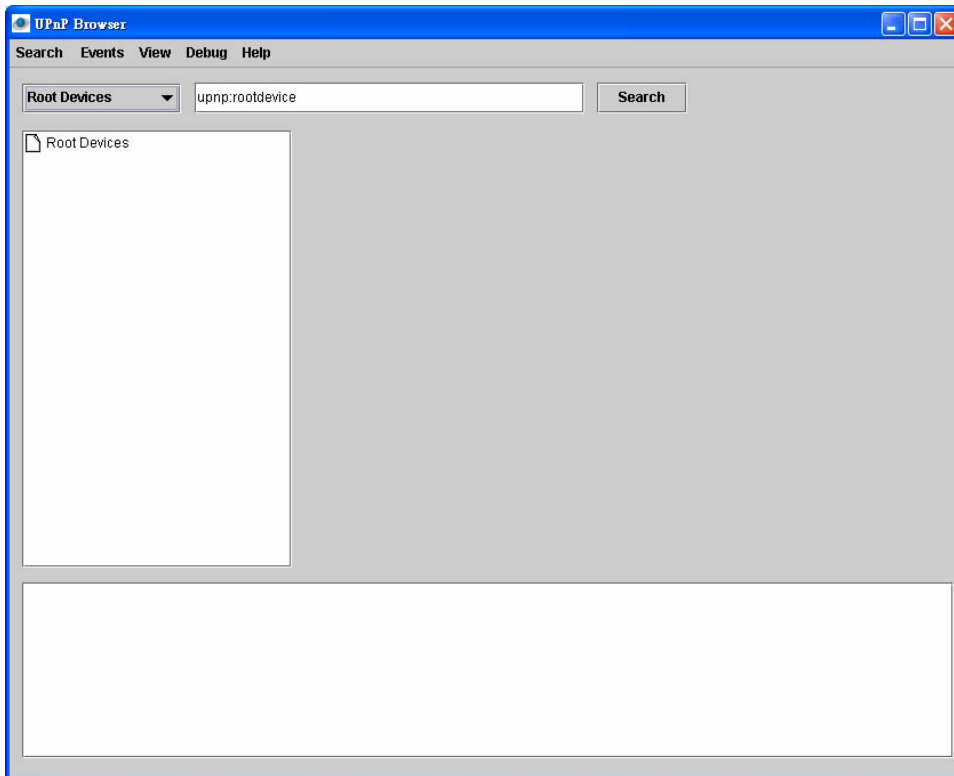


圖 5.13 UPnP 客戶端模擬程式

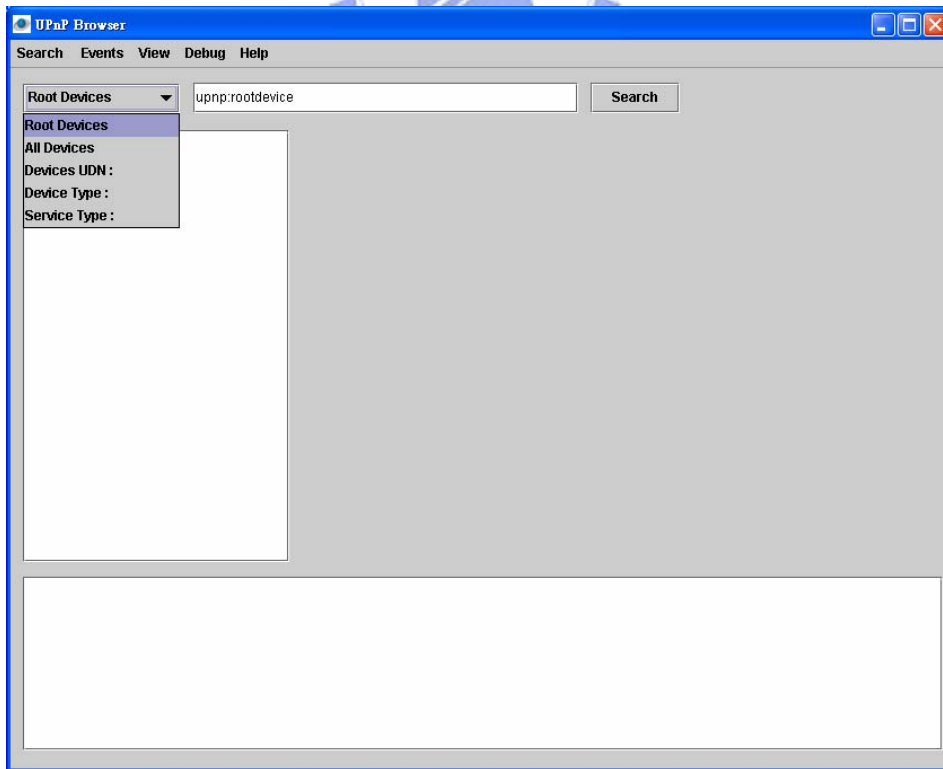


圖 5.14 UPnP 客戶端模擬程式搜尋方式

UPnP 客戶端模擬程式並提供聆聽服務功能，因此當新的設備 advertise 通知其它成員時，它便會自動新增至樹狀清單內。如圖 5.15。

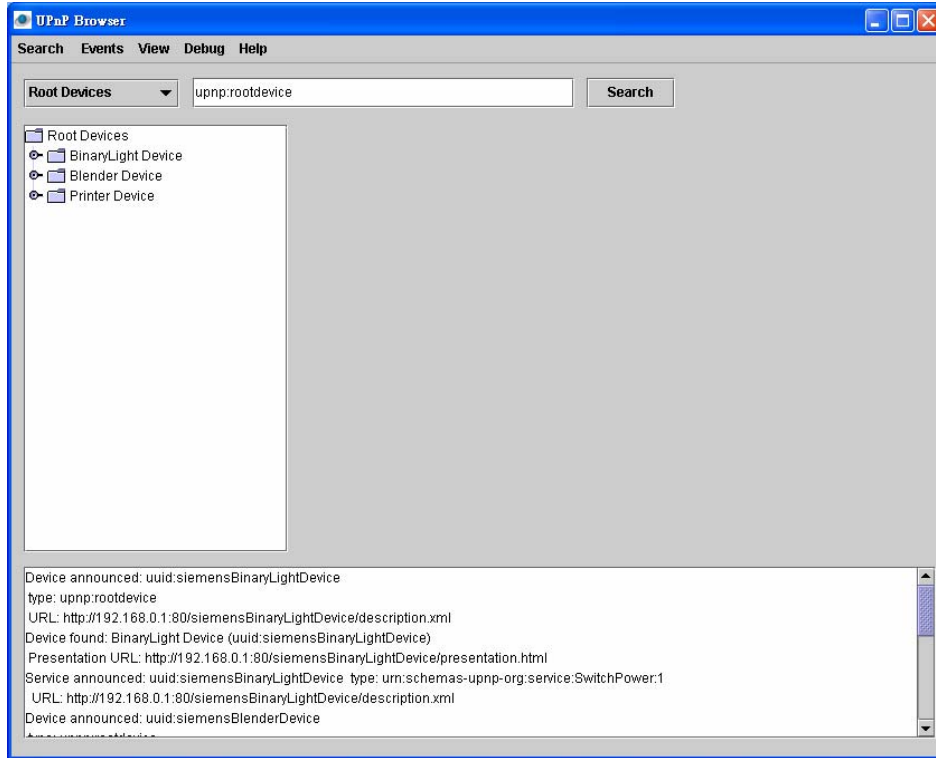


圖 5.15 UPnP 客戶端模擬程式顯示目前網路上提供的服務

點選任一設備後，會列出它所提供的服務。如圖 5.16。右上方是設備的相關資訊，包括服務種類、服務編號、服務資訊的描述和服務本身狀態的 presentation。右下包括服務所提供的 method 資訊和它的變數狀態描述。點選一下 Action 視窗內欲呼叫的 method，輸入設定，即可進行遠端呼叫該服務。

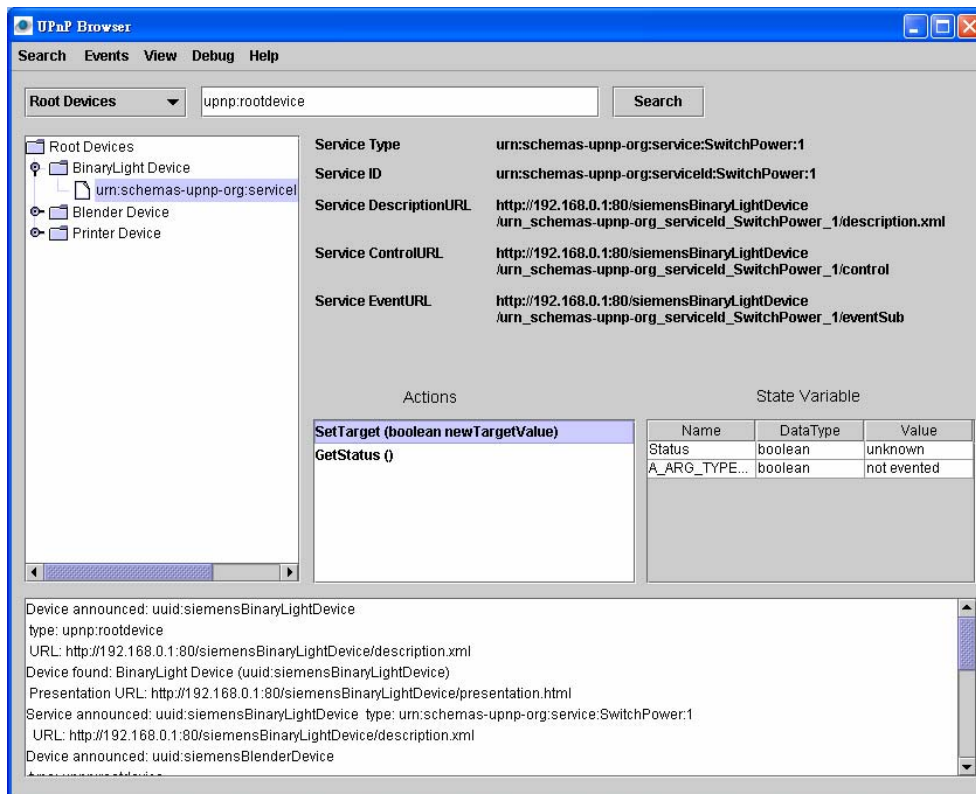


圖 5.16 UPnP 客戶端模擬程式的服務資訊

簡單介紹完實驗環境內需使用的資訊家電模擬程式後，再來是進行不同 Domain 的服務呼叫，來測試家用閘道器的整合功能。在這測試的項目包括：

- (1) 測試 UPnP 客戶端呼叫 UPnP 服務
- (2) 測試 UPnP 客戶端呼叫 Jini 服務
- (3) 測試 UPnP 客戶端呼叫 SLP 服務
- (4) 測試 Jini 客戶端呼叫 UPnP 服務
- (5) 測試 Jini 客戶端呼叫 SLP 服務

## 2. 測試 UPnP 客戶端呼叫 UPnP 服務

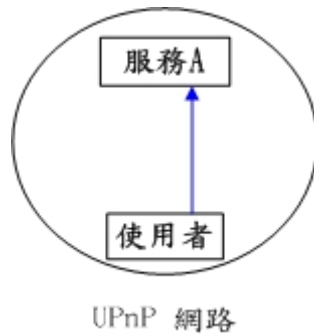


圖 5.17 UPnP 客戶端呼叫 UPnP 服務流程說明

這個模擬主要是由 UPnP 客戶端模擬程式呼叫 UPnP 電燈泡設備，設定 UPnP 電燈泡的電源狀態開啟；呼叫 UPnP 電燈泡設備所提供的 SetTarget method，並設定電源狀態為 1。最後，可看到 UPnP 電燈泡的狀態變為開啟，成功完成 UPnP 電燈泡服務的呼叫。如圖 5.18 至圖 5.24。



圖 5.18 UPnP 電燈泡模擬程式初始狀態

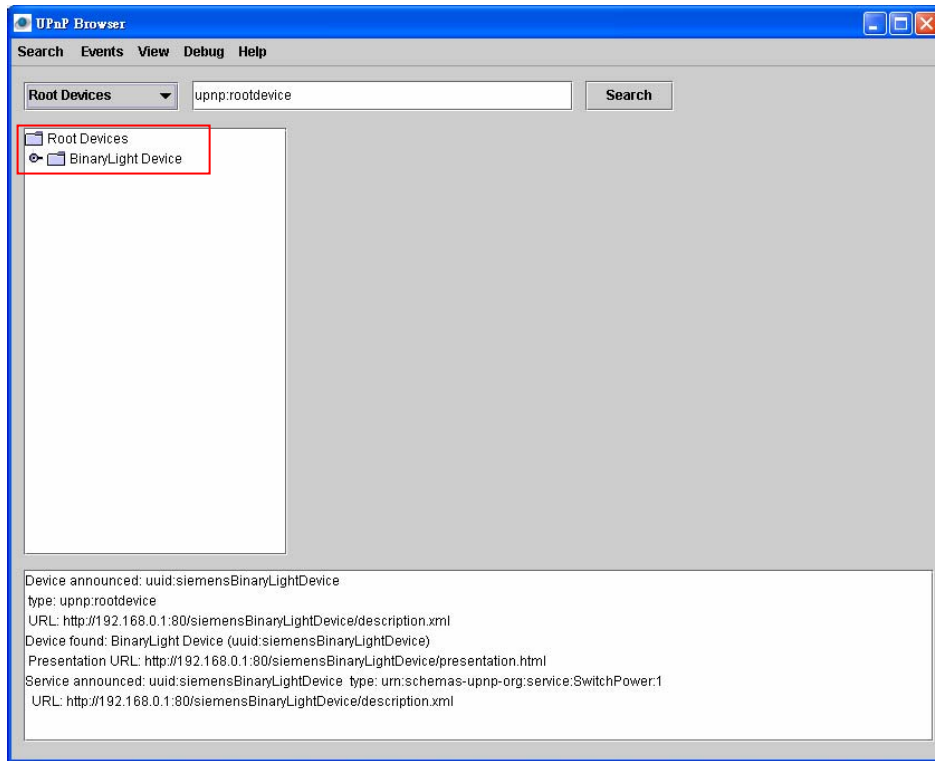


圖 5.19 UPnP 客戶端模擬程式查詢到電燈泡服務

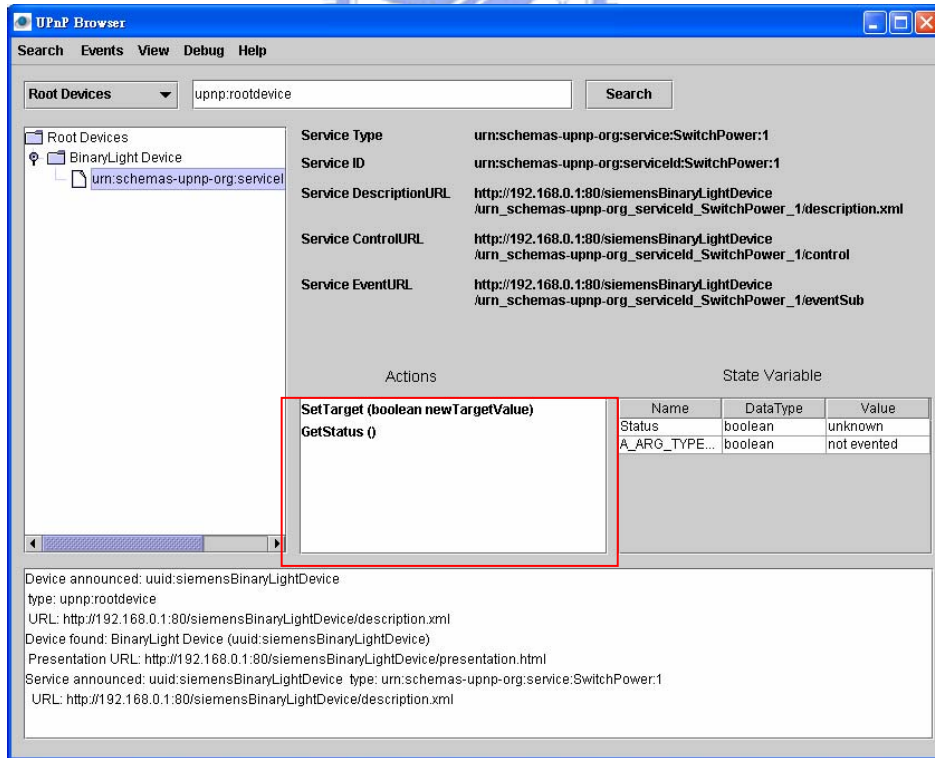


圖 5.20 UPnP 電燈泡所提供的 method 資訊

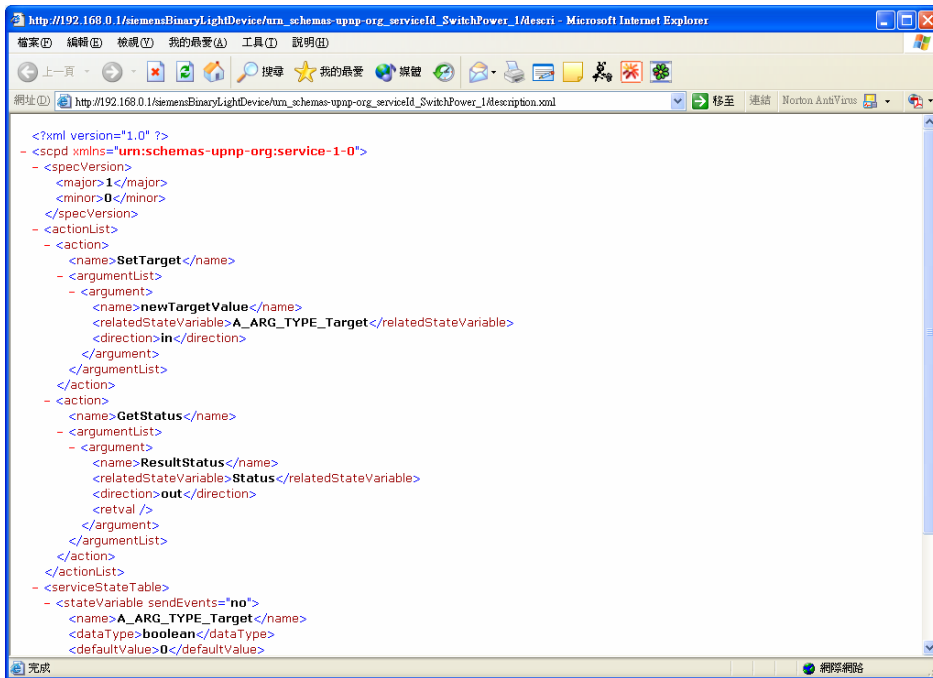


圖 5.21 UPnP 電燈泡 SSDP 提供的服務資訊

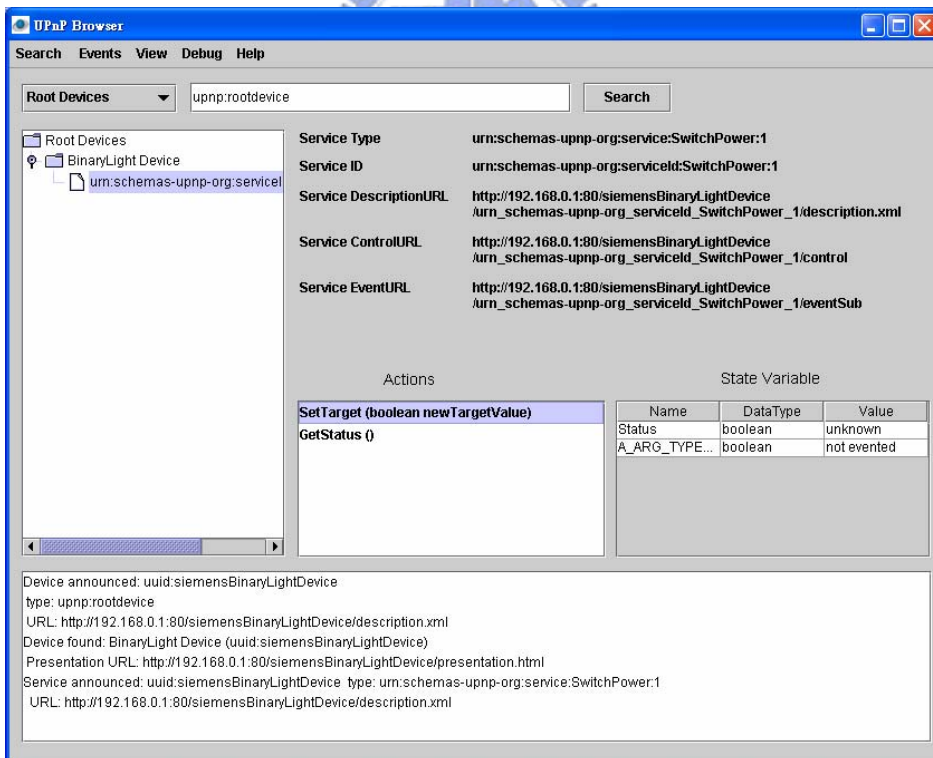


圖 5.22 呼叫 SetTarget method 設定電源狀態

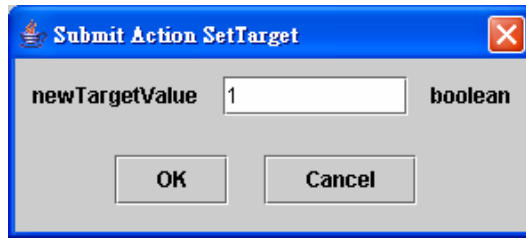


圖 5.23 將 UPnP 電燈泡電源開啟



圖 5.24 進行遠端呼叫後，UPnP 電燈泡設備會改變自己的狀態為開啟

### 3. 測試 UPnP 客戶端呼叫 Jini 服務

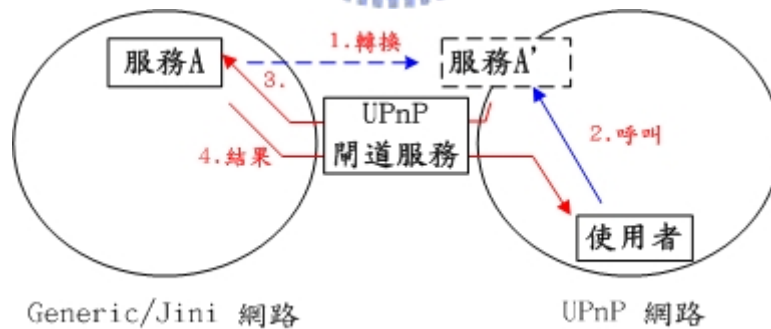


圖 5.25 UPnP 客戶端呼叫 Jini 服務流程說明

這個模擬主要是由 UPnP 客戶端模擬程式呼叫 Jini 果汁機設備，設定 Jini 果汁機的搖動速度為 2；呼叫 Jini 果汁機設備所提供的 setSpeed method，並設定搖動速度狀態為 2。最後，可看到 Jini 果汁機的搖動速度狀態變為 2，成功完成 Jini 果汁機服務的呼叫。如圖 5.26 至圖 5.30。在設定完成搖動速度後，再呼叫 Jini 果汁機設備所提供的 getSpeed method，可以看到目前搖動速度狀態的確為 2。如圖 5.31 至圖 5.32。

在圖 5.27，UPnP 客戶端模擬程式查詢到 UPnP 閘道服務下轉換的果汁機服務。由於 UPnP 閘道服務會處理由 Generic Domain 轉換過來的服務，並把該服務加在 UPnP 閘道服務下，如圖 5.26 中可看到 uPnPMgr 閘道服務下提供 urn:schemas-upnp-org:serviceId:BlenderDeviceImpl:1 這個果汁機服務。

在圖 5.30，UPnP 客戶端模擬程式設定 Jini 果汁機搖動速度為 2 時，會先呼叫 uPnPMgr 閘道服務，由閘道服務再行呼叫 Jini 果汁機。最後，閘道服務會將結果傳回給 UPnP 客戶端模擬程式。



圖 5.26 Jini 果汁機模擬程式初始狀態



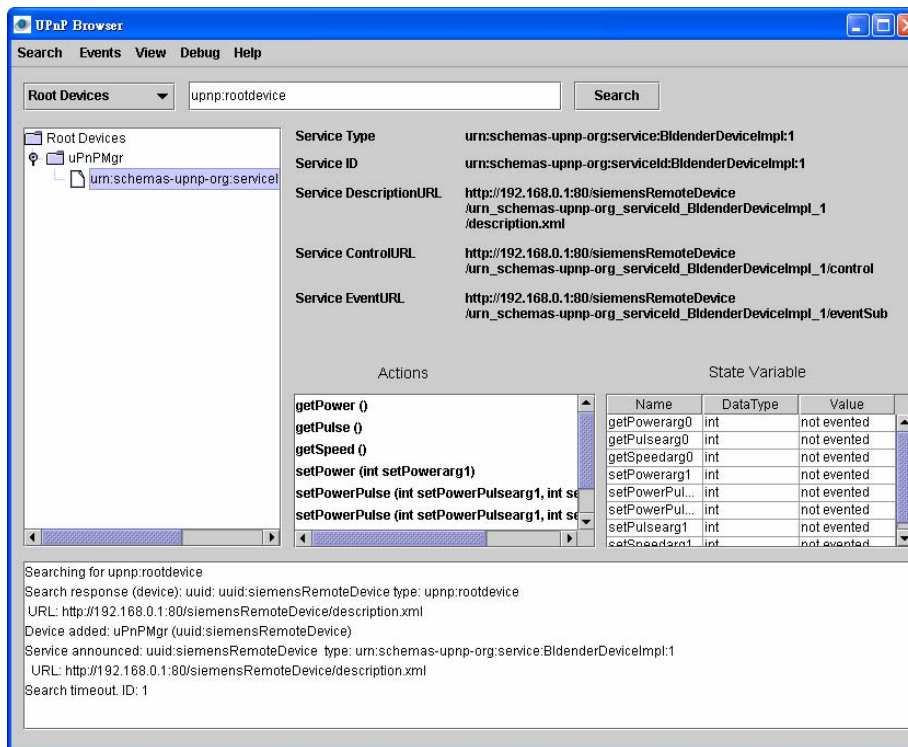


圖 5.27 UPnP 客戶端模擬程式查詢到 UPnP 鬧道服務下轉換的 Jini 果汁機服務

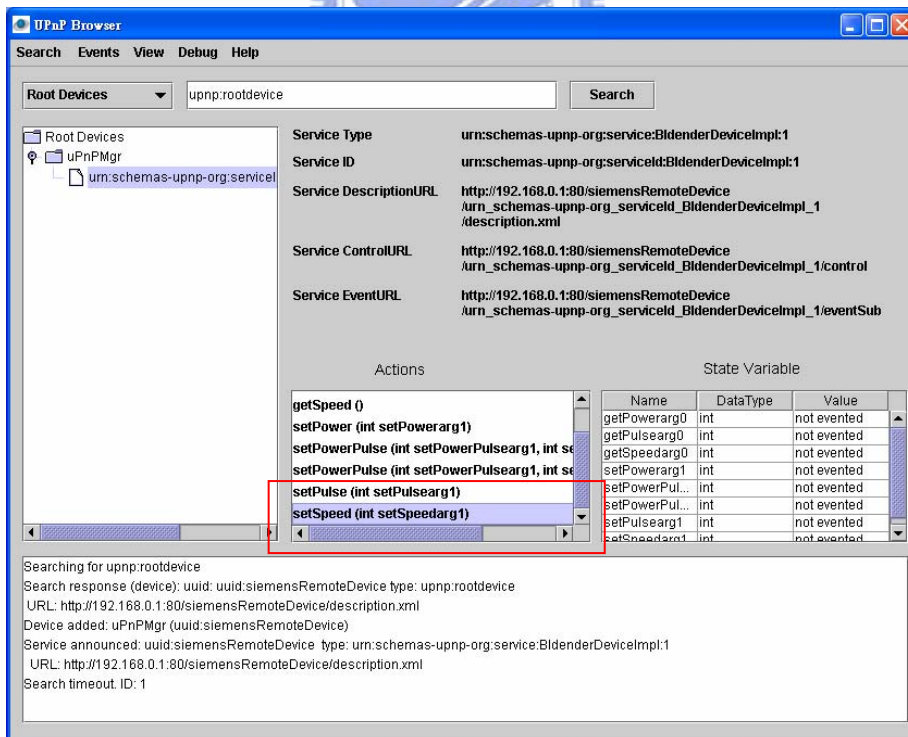


圖 5.28 呼叫 setSpeed method 設定果汁機搖動速度



圖 5.29 將 Jini 果汁機搖動速度設為 2



圖 5.30 進行遠端呼叫後，Jini 果汁機設備搖動速度會改變為 2

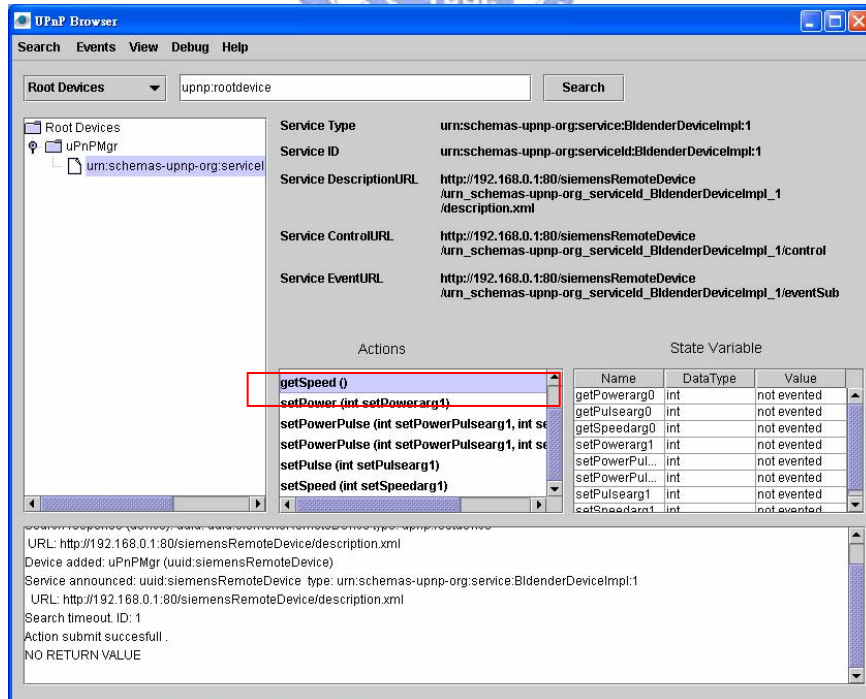


圖 5.31 呼叫 getSpeed method 取得果汁機目的搖動速度

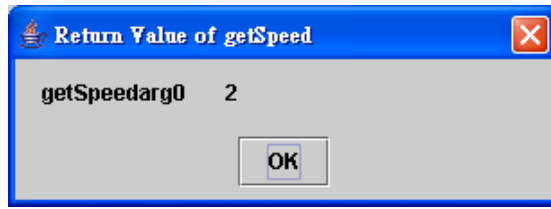


圖 5.32 進行遠端呼叫後，傳回目前 Jini 果汁機設備搖動速度為 2

#### 4. 測試 UPnP 客戶端呼叫 SLP 服務

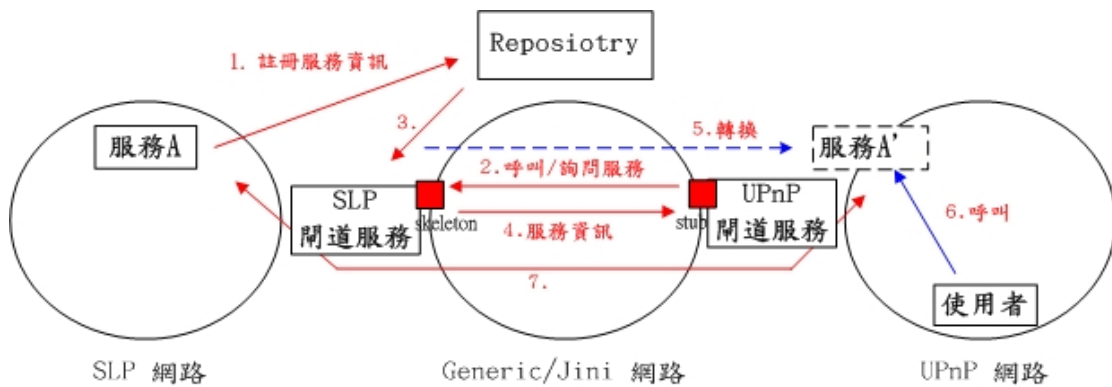


圖 5.33 UPnP 客戶端呼叫 SLP 服務流程說明

這個模擬主要是由 UPnP 客戶端模擬程式呼叫 SLP 印表機設備，設定 SLP 印表機的列印紙張為 2；呼叫 SLP 印表機設備所提供的 setPage method，並設定列印紙張狀態為 2。最後，可看到 SLP 印表機的列印紙張狀態變為 2，成功完成 SLP 印表機服務的呼叫。如圖 5.34 至圖 5.38。

在圖 5.35，UPnP 客戶端模擬程式查詢到 UPnP 閘道服務下轉換的印表機服務。由於 UPnP 閘道服務會處理由 Generic Domain 轉換過來的服務，並把該服務加在 UPnP 閘道服務下，如圖 5.28 中可看到 uPnPMgr 閘道服務下提供 urn:schemas-upnp-org:service:slpPrinter:1 這個印表機服務。

在圖 5.31，UPnP 客戶端模擬程式設定 SLP 印表機列印紙張為 2 時，會先呼叫 uPnPMgr 閘道服務，由閘道服務再行呼叫 SLP 印表機。最後，閘道服務會將結果傳回給 UPnP 客戶端模擬程式。



圖 5.34 SLP 印表機模擬程式初始狀態

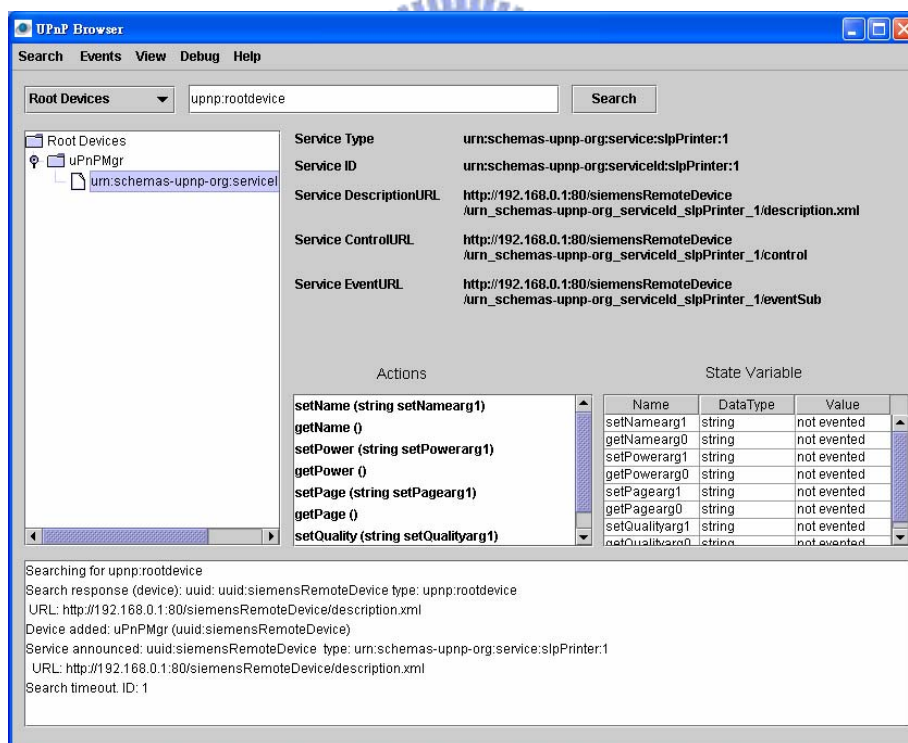


圖 5.35 UPnP 客戶端模擬程式查詢到 UPnP 閘道服務下轉換的 SLP 印表機服務

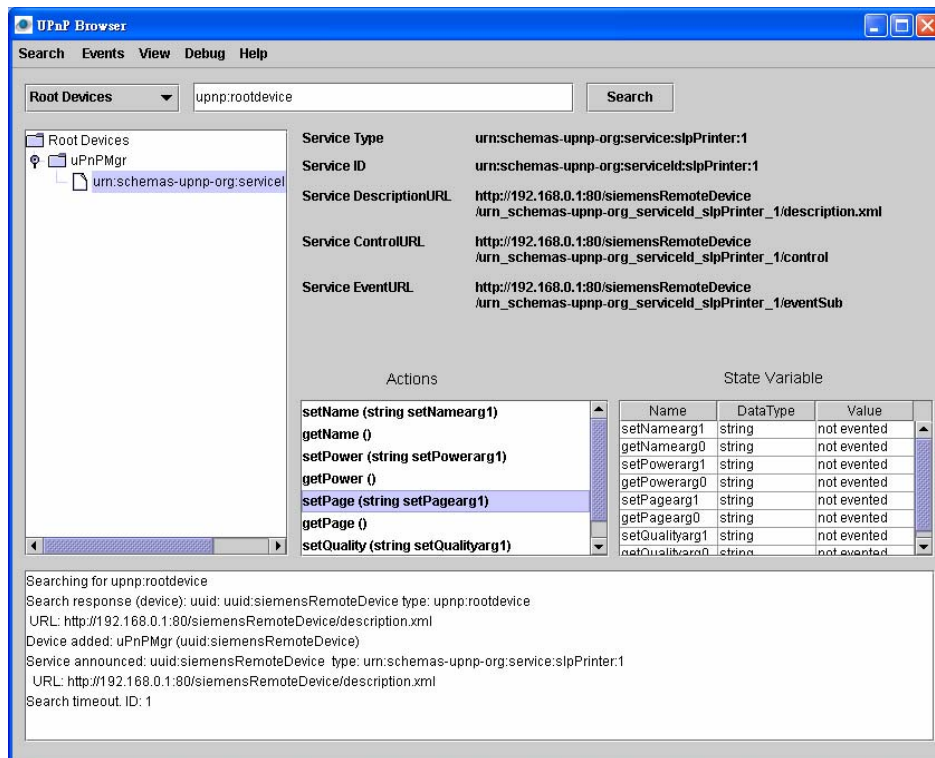


圖 5.36 呼叫 setPage method 設定印表機列印紙張數目

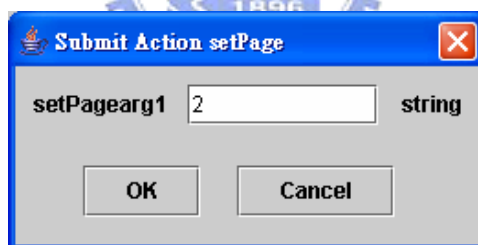


圖 5.37 將 SLP 印表機列印紙張設為 2



圖 5.38 進行遠端呼叫後，SLP 印表機設備列印紙張數目會改變為 2

#### 5. 測試 Jini 客戶端呼叫 UPnP 服務

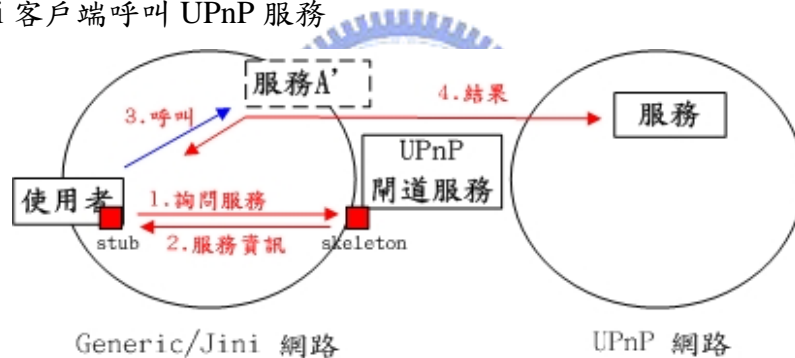


圖 5.39 Jini 客戶端呼叫 UPnP 服務流程說明

這個模擬主要是由 Jini 客戶端呼叫 UPnP 電燈泡設備，設定 UPnP 電燈泡的電源為開啟；呼叫 UPnP 電燈泡 印表機設備所提供的 setTarget method，並設定列電源狀態為 1。最後，可看到 UPnP 電燈包的電源狀態變為開啟，成功完成 UPnP 電燈泡服務的呼叫。如圖 5.40 至圖 5.52。

由於 UPnP 閘道服務會提供 Generic Domain 查詢 UPnP 網路有提供那些服務和代為呼叫，因此，Jini 客戶端設定 UPnP 電燈泡電源狀態為 1 時，會先呼叫 uPnPMgr 閘道服務，由閘道服務再行呼叫 UPnP 電燈泡。最後，閘道服務會將結果傳回給 Jini 客戶端。

[Jini 客戶端程式流程]

```
Class[] types = new Class[] {uPnPMgr.class};
ServiceTemplate template = new ServiceTemplate(null, types, null);
```

```

uPnPMgr upnPMgr = (uPnPMgr) lookup(template);    // 取得 UPnP 開道服務 Proxy Object
upnPMgr.setuPnPMethod("uuid:siemensBinaryLightDevice;urn:schemas-upnp-
org:serviceId:SwitchPower:1;void setTarget(1)"); // 藉由開道服務提供的委任呼叫功能，
                                                    // 呼叫 UPnP 電燈泡的 setTarget，
                                                    // 並將電源狀態設為 1

```



圖 5.40 UPnP 電燈泡模擬程式初始狀態

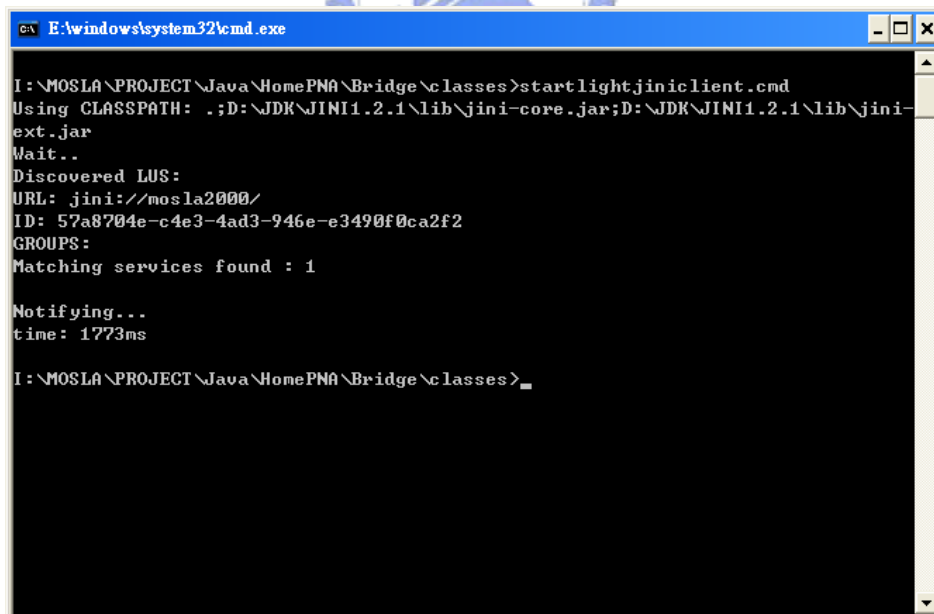


圖 5.41 遠端呼叫，設定電燈泡電源為開啟



圖 5.42 進行遠端呼叫後，SLP 電燈泡設備電源會改變開啟

## 6. 測試 Jini 客戶端呼叫 SLP 服務

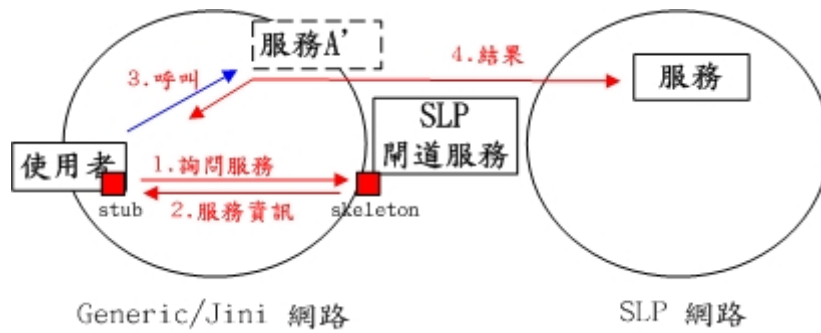


圖 5.43 Jini 客戶端呼叫 SLP 服務流程說明

這個模擬主要是由 Jini 客戶端呼叫 SLP SA1。SLP SA1 服務僅提供簡單的 2 個 method：setMsg、getMsg，分別為設定和取得 SA1 服務內的字串。首先由 Jini 客戶端呼叫 SLP SA1 所提供的 setMsg method，設定訊息為 John 後，再呼叫 getMsg method，取回剛設定的訊息，由 Jini 客戶端取得回傳值，可知遠端呼叫成功。如圖 5.44 至圖 5.45。

由於 SLP 閘道服務會提供 Generic Domain 查詢 SLP 網路有提供那些服務和代為呼叫，因此，Jini 客戶端設定 SLP SA1 服務訊息狀態為 John 時，會先呼叫 SLPMgr 閘道服務，由閘道服務再行呼叫 SLP SA1 服務。最後，閘道服務會將結果傳回給 Jini 客戶端。

[Jini 客戶端程式流程]

```
Class[] types = new Class[] {SLPMgr.class};
ServiceTemplate template = new ServiceTemplate(null, types, null);
```



```

SLPMgr slpMgr = (SLPMgr) lookup(template);           // 取得 SLP 閘道服務 Proxy Object
String sExec = "";
String sRet = "";

```

```

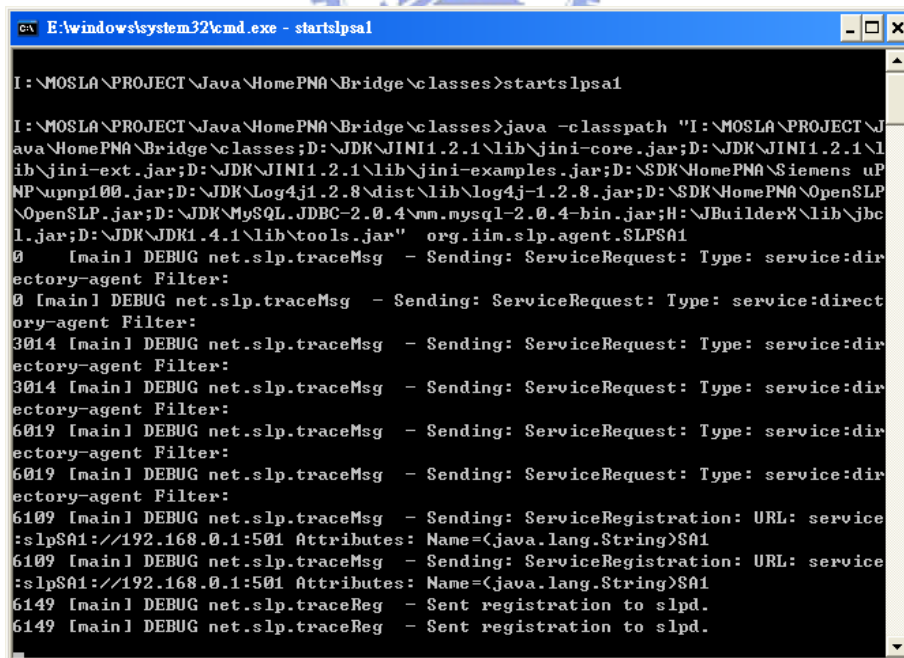
// 藉由閘道服務提供的委任呼叫功能，呼叫 SA1 印表機的 setMsg，並將訊息設為 John
sExec = "service:slpSA1://192.168.0.1:501;void setMsg(John)";
sRet = (String) slpMgr.setSLPMethod(sExec);
if (sRet != null) {
    System.out.println("Result: " + sRet);
}

```

```

// 藉由閘道服務提供的委任呼叫功能，呼叫 SA1 印表機的 getMsg，取回目前 SA1 內的訊息
sExec = "service:slpSA1://192.168.0.1:501;String getMsg()";
sRet = (String) slpMgr.setSLPMethod(sExec);
if (sRet != null) {
    System.out.println("Result: " + sRet);
}

```



```

E:\windows\system32\cmd.exe - startslpsal

I:\MOSLA\PROJECT\Java\HomePNA\Bridge\classes>startslpsal

I:\MOSLA\PROJECT\Java\HomePNA\Bridge\classes>java -classpath "I:\MOSLA\PROJECT\Java\HomePNA\Bridge\classes;D:\JDK\JINI1.2.1\lib\jini-core.jar;D:\JDK\JINI1.2.1\lib\jini-ext.jar;D:\JDK\JINI1.2.1\lib\jini-examples.jar;D:\SDK\HomePNA\Siemens uPNP\upnp100.jar;D:\JDK\Log4j1.2.8\dist\lib\log4j-1.2.8.jar;D:\SDK\HomePNA\OpenSLP\OpenSLP.jar;D:\JDK\MySQL.JDBC-2.0.4\bin.mysql-2.0.4-bin.jar;H:\JBuilder8\lib\jdbc1.jar;D:\JDK\JDK1.4.1\lib\tools.jar" org.iim.slp.agent.SLPSA1
0 [main] DEBUG net.slp.traceMsg - Sending: ServiceRequest: Type: service:directory-agent Filter:
0 [main] DEBUG net.slp.traceMsg - Sending: ServiceRequest: Type: service:directory-agent Filter:
3014 [main] DEBUG net.slp.traceMsg - Sending: ServiceRequest: Type: service:directory-agent Filter:
3014 [main] DEBUG net.slp.traceMsg - Sending: ServiceRequest: Type: service:directory-agent Filter:
6019 [main] DEBUG net.slp.traceMsg - Sending: ServiceRequest: Type: service:directory-agent Filter:
6019 [main] DEBUG net.slp.traceMsg - Sending: ServiceRequest: Type: service:directory-agent Filter:
6109 [main] DEBUG net.slp.traceMsg - Sending: ServiceRegistration: URL: service:slpSA1://192.168.0.1:501 Attributes: Name=<java.lang.String>SA1
6109 [main] DEBUG net.slp.traceMsg - Sending: ServiceRegistration: URL: service:slpSA1://192.168.0.1:501 Attributes: Name=<java.lang.String>SA1
6149 [main] DEBUG net.slp.traceReg - Sent registration to slpd.
6149 [main] DEBUG net.slp.traceReg - Sent registration to slpd.

```

圖 5.44 SLP SA1 服務啟動

```
E:\windows\system32\cmd.exe

I:\MOSLA\PROJECT\Java\HomePNA\Bridge\Classes>startslpsaljiniclient
Using CLASSPATH: "D:\SDK\HomePNA\OpenSLP\OpenSLP.jar";d:\jdk\mysql.jdbc-2.0.4\mm
.mysql-2.0.4-bin.jar;. ;D:\JDK\JINI1.2.1\lib\jini-core.jar;D:\JDK\JINI1.2.1\lib\j
ini-ext.jar
Wait..
Matching services found : 1
Notifying...
Result: setMethodOk
Result: getMethodOk value=John
I:\MOSLA\PROJECT\Java\HomePNA\Bridge\Classes>
```

圖 5.45 Jini 客戶端經由 SLP 閘道服務呼叫 SLP SA1 回傳結果

經由家用閘道器下進行跨搜尋服務機制的遠端呼叫實驗，可以發現使用閘道服務，可以輕易的整合不同網路環境，使用者可以在不需考慮底層協定為何的情況下，來達成異質環境的協同工作目的。

### 5.3 效能評估

為了要評估使用經由家用閘道器，進行不同 Domain 的遠端呼叫後，效率如何的問題，因此，本實驗在 2 台 700MHz CPU，384M RAM，OS 為 WinXP 的電腦上進行測試。

效能評估的參數為客戶端呼叫服務後的回應時間(Response time)，回應時間主要包含：客戶端進行服務搜尋時間後，客戶端呼叫該服務，伺服器執行完該功能後回應結果。即

$$\text{回應時間} = \text{客戶端服務搜尋時間} + \text{客戶端遠端呼叫} + \text{伺服器執行後回應時間}$$

伺服器執行功能以單純的功能設定為主，例如 5.2 節描述的呼叫 UPnP 電燈泡開啟電源功能。這裡採行同 5.2 節使用的交互整合測試方式，以供測量回應時間。模擬情況（各重複執行 10 次取平均時間）：

- |                                      |             |
|--------------------------------------|-------------|
| (1) Jini 客戶端呼叫 Jini Hello setMsg 服務  | 平均時間 1282ms |
| (2) Jini 客戶端呼叫 UPnP 電燈泡 setTarget 服務 | 平均時間 1161ms |
| (3) Jini 客戶端呼叫 SLP 印表機 setPage 服務    | 平均時間 1031ms |

- |                                      |             |
|--------------------------------------|-------------|
| (4) UPnP 客戶端呼叫 Jini Hello setMsg 服務  | 平均時間 571ms  |
| (5) UPnP 客戶端呼叫 UPnP 電燈泡 setTarget 服務 | 平均時間 611ms  |
| (6) UPnP 客戶端呼叫 SLP 印表機 setPage 服務    | 平均時間 741ms  |
| (7) SLP 客戶端呼叫 Jini Hello setMsg 服務   | 平均時間 1051ms |
| (8) SLP 客戶端呼叫 UPnP 電燈泡 setTarget 服務  | 平均時間 1032ms |
| (9) SLP 客戶端呼叫 SLP 印表機 setPage 服務     | 平均時間 81ms   |

表 5.1 不同 Domain 下，客戶端進行遠端呼叫的平均回應時間(灰色為經過閘道服務)

客戶端 \ 伺服器端	Jini 服務	UPnP 服務	SLP 服務
Jini 客戶端	(1)1282ms	(2)1161ms	(3)1031ms
UPnP 客戶端	(4) 571ms	(5) 611ms	(6) 741ms
SLP 客戶端	(7)1051ms	(8)1032ms	(9) 81ms

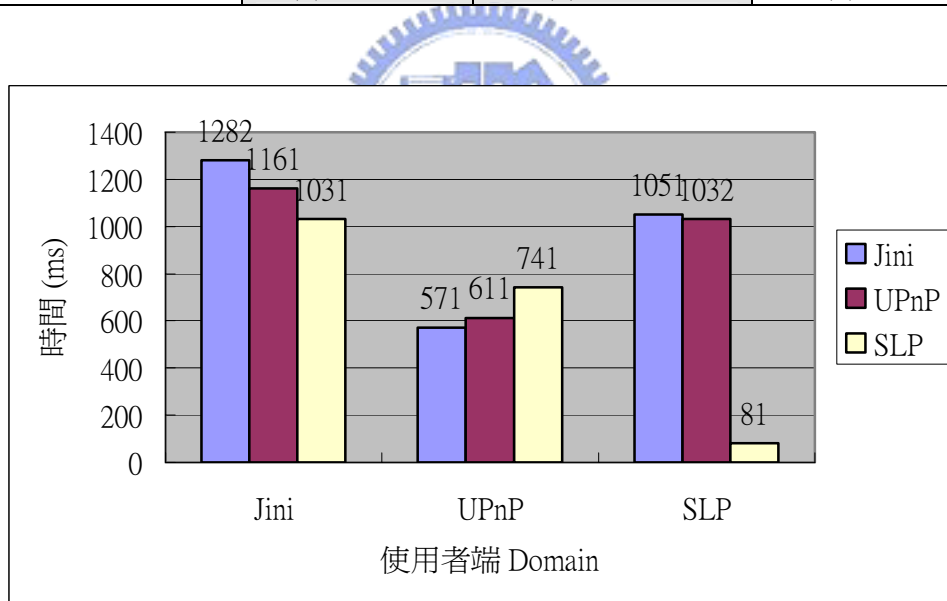


圖 5.46 不同 Domain 下，客戶端進行遠端呼叫的平均回應時間

如表 5.1 所示。模擬結果可以發現，使用 Service Agent 事先搜集好不同 Domain 的服務資訊，可避免需使用時再進行 Multicast 查詢的負擔，可加快閘道服務處理的時間。跨 Domain 存取服務時，經過閘道服務進行轉換或是代為呼叫，可以看到回應時間和不經閘道服務不會相差太多，這主要是因為閘道服務只單純處理訊息的傳送，沒有實作其它功能（例如 DA 或 Lookup Service 功能），只有在代為呼叫時會較原本直接呼叫的方式慢些。

另外，不同的服務搜尋機制間的效率為 SLP>UPnP>Jini，SLP 回應時間會較快主要是因為 SLP 並未定義遠端呼叫的機制，在實驗裡我們是採行自行定義的協定，使用 SOCKET 傳送資料，自然會較 Jini 和 UPnP 使用遠端呼叫、進行物件串流轉換(Marshal/Unmarshal)來的快。

