Mobile Systems M

Alma Mater Studiorum – University of Bologna
CdS Laurea Magistrale (MSc) in
Computer Science Engineering

Mobile Systems M course (8 ECTS)
II Term – Academic Year 2019/2020

07 – Service Discovery

Paolo Bellavista
paolo.bellavista@unibo.it

Luca Foschini
luca.foschini@unibo.it

http://lia.disi.unibo.it/Courses/sm1920-info/

Resource/Service Discovery

Sometimes the term is used in a broad sense to indicate both the "real" discovery and also the configuration operations needed to access resources/services, as well as the resource/service requests themselves

Key support features for any open, dynamic, loosely coupled, and peer-to-peer system:

- ***Automated configuration***
- ***Discovery of resources and services***
- ***Resource/service delivery***

Main discovery standards and solutions:

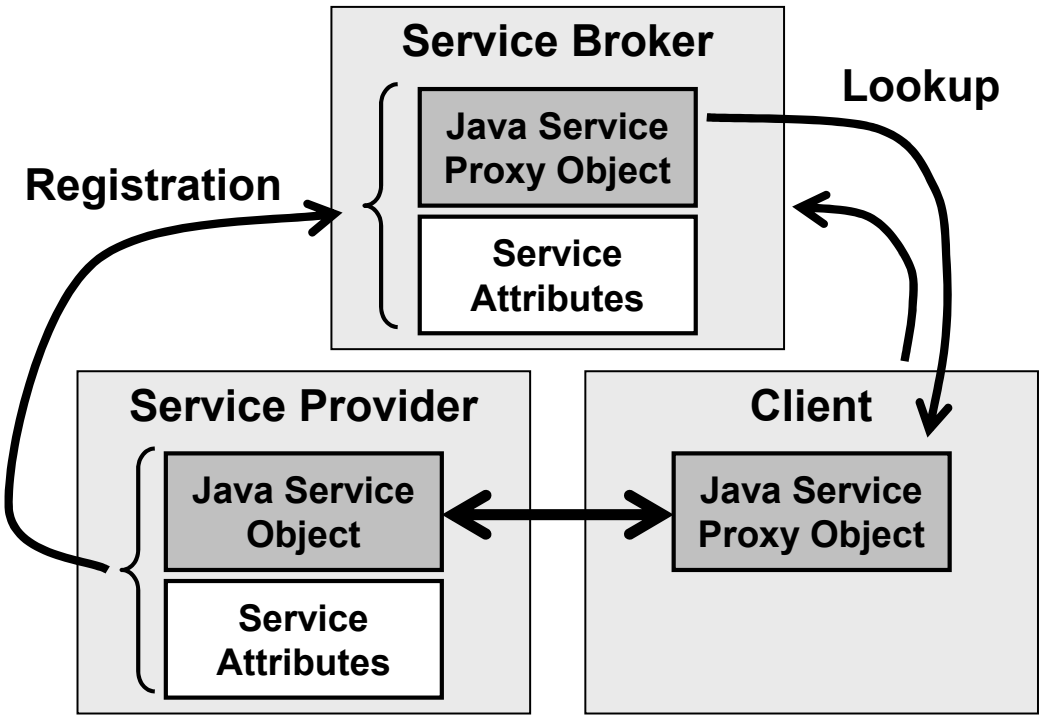
- Jini, Service Location Protocol (SLP), Universal Plug and Play (UPnP), …

Key Features

- ***Auto-configuration***
  - Device have to configure themselves to participate to offering/requesting resources and services
  - For instance, but not only, configuration of a temporary IP address in the current locality
- ***Discovery*** of available resources and services
  - Who offers resources and services (*service provider*) has to be able to make ***advertising of them***
    - Of which resources and services
    - Of how to make invocations, via which interfaces
  - Clients have to be able to ***find (local)*** resources and services
- ***Access*** to resources and services
  - Clients have to be able to communicate with service provider, invoke services, transfer input parameters, and possibly receive return results
  - Also support to authentication and authorization

- Relevance of achieving good ***reliability and scalability***

Jini
(Sun Microsystem, now Apache River)



Apache River

Most relevant discovery solution for the Java world

- Service provider dynamically discovers ***one or more lookup services (broker)***
- Service provider ***registers a resource/service object*** (modeled as Java object) and its attributes to the broker
- Client ***requests a service***, typically by specifying attributes of the looked-for service; ***one instance of object to simplify resource/service access*** moves to client at runtime
- Lookup service can ***notify registered clients*** when there is state change for their resources and services
- Client ***interacts with discovered resource/service via obtained Java object***

Reliability Management in River

- Possible failures (and not only, see versioning) are managed through ***lease mechanism***
  - River assigns resources to clients with ***lease of given time duration*** (less or equal to what requested by client)
  - Once terminated the lease interval, client has to ***re-fresh lease*** in order to continue accessing the resource/service
  - Also lookup registration is made with lease: therefore, all leases have an expiration deadline, for any user, in the case of “long” service provider fault
- River supports ***redundancy at the infrastructure level and resiliency against faults***
  - Possible to ***deploy several lookup services*** in the same network
  - Service providers can ***register their proxy objects in multiple lookup services***
  - Also usage within transactions and ***automated rollback when lease expires***

Scalability in Apache River

Scalability realized via ***dynamic organization in "communities" or "federations"***

- Groups of River resources/services can aggregate into a community, typically the one of ***local services*** registered at least in a local lookup service
- Different communities can be ***linked together in larger groups through lookup service***
  - One community registers itself at other communities by registering its own lookup service
  - How to manage the nesting of lookup services?

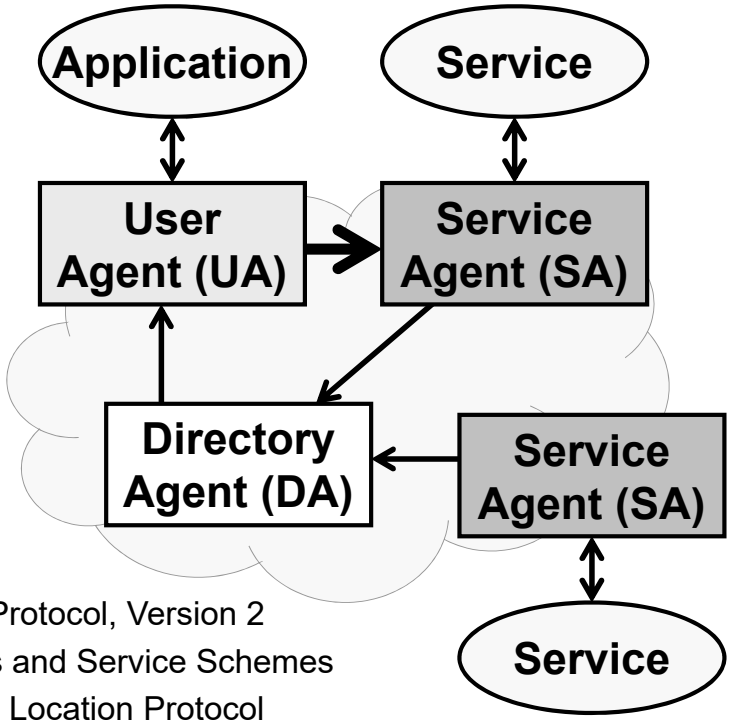
Service Proxy Object in River is Dynamic

If compared with more traditional solutions for resource/service access, anyway retrieved dynamically via intermediate stubs and skeletons:

- River overcomes ***limitations of stub/skeleton creation at compile-time***, which is for example typical of RPC
- River allow to clients ***to obtain service provider stubs at provisioning time***
- RPS stubs are similar to ***service proxy object that the River server dynamically loads in the lookup service***
- Service proxy object allows clients to use the discovered service with NO a priori knowledge of its implementation details

Service Location Protocol (SLP)

- SLP is the ***standard approach of Internet Engineering Task Force (IETF)*** for resource/service discovery
- Basic idea: SLP makes resources/services visible through URL registration at intermediate agents



- RFC 2608 - Service Location Protocol, Version 2
- RFC 2609 - Service Templates and Service Schemes
- RFC 2614 - An API for Service Location Protocol

SLP is based on 3 Types of Agents

Agents as entities capable of SLP message processing

- ***Service agent***
  - Performs broadcast (usually periodic) of advertisement messages about its resources/services (associations with URLs)
- ***Directory agent*** (optional)
  - Performs caching of advertisement messages (from service agents) as a centralized repository
  - Processes discovery queries received from user agents by returning back the URLs that match
- ***User agent***
  - To discover resources/services at the client side
  - Also efficient usage of multicast to service agent groups

Standard specification is relatively rich and flexible, but industry-mature implementations not so widespread (OpenSLP, Sun SLP, Xerox printers, …)

Universal Plug-and-Play (UPnP)

Standard specification started by Microsoft (at the beginning, an internal proprietary solution)

- Primary goal: to enable advertisement, discovery, and control of networked devices, services, ***consumer electronics in typically domestic ad hoc envs*** (see the current exploitation in media centers, tvs, hi-fi devices, …)
- UPnP uses, as underlying technologies:
  - UDP or TCP/IP
  - HTTP
  - XML/HTML and SOAP

Universal Plug-and-Play (UPnP)

Via UPnP a device can:

- Dynamically ***join a network***, by obtaining an ***IP address that is locally valid***
- Making visible ***its capabilities***, by need and on-demand
- ***Discover the presence and capabilities*** of other devices
- Dynamically leave a network

UPnP supports:

- Automated IP configuration
- Discovery of resources and services
- ***Description of resources/services based on XML***
- ***Service control based on SOAP***
- ***Event management*** (via Generic Eventing and Notification Architecture - GENA)
- Presentation in HTML/XML

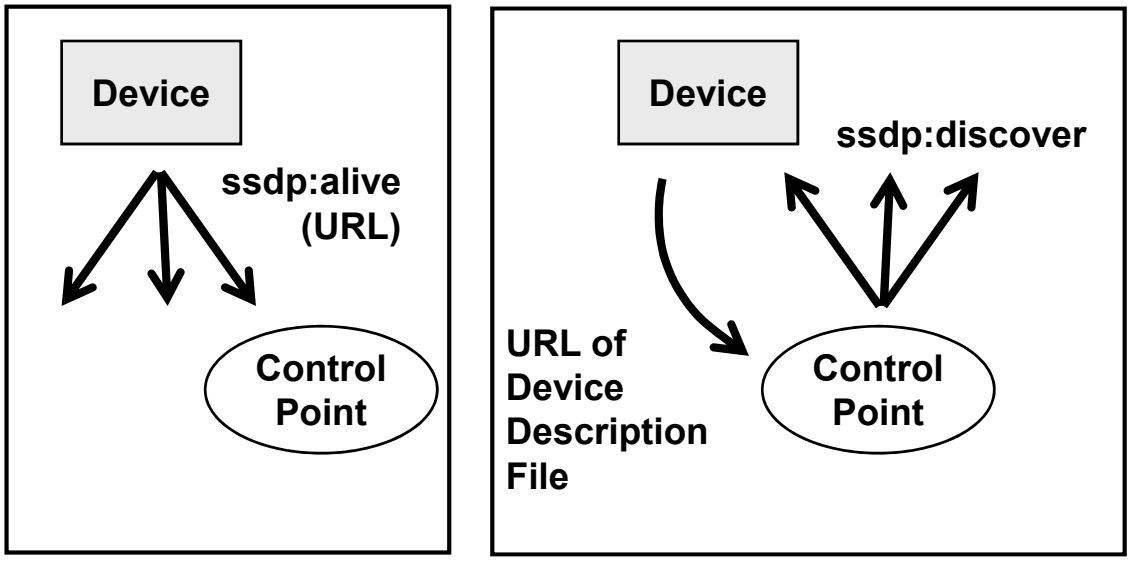
IP Addressing in UPnP

By delving into finer technical details:

- ***UPnP uses Auto IP*** (the real protocol part for auto-configuration) to enable devices to connect to the network with NO need of explicit administration
- When a device connects to a network, it tries to ***obtain an IP address*** from a DHCP server, if available
- If no DHCP server is available, an IP address is ***automatically claimed from a fixed reserved range for usage ONLY at local network***
  - An IP address from the link-local address range is ***selected randomly*** (169.254.0.0/16 for IPv4)
  - Request sent via Address Resolution Protocol (ARP) to check whether other devices have already picked up that address

Service Discovery in UPnP

UPnP exploits the Simple Service Discovery Protocol (SSDP) as discovery protocol based on ***usage of dedicated multicast address*** (239.255.255.250 on port 1900 via UDP)

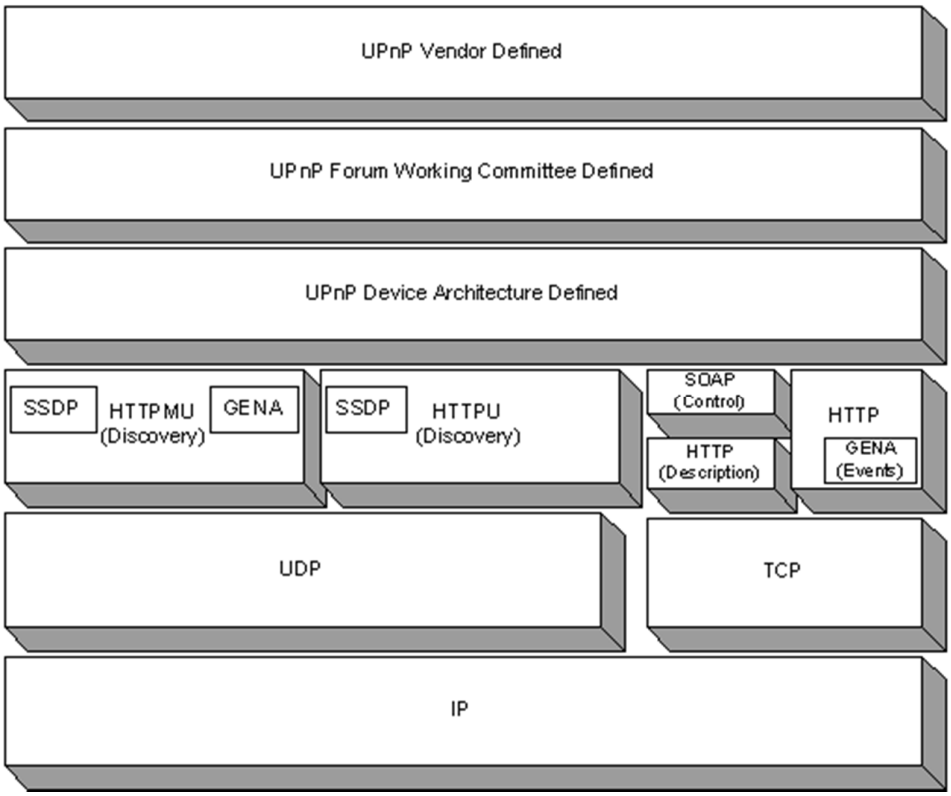


Service Discovery in UPnP

- Device (e.g., UPnP-compliant projector) ***multicasts*** an advertisement message ***(ssdp:alive)*** to ***exhibit its services to active control points*** (e.g., tablets or home gateways)
- Control point can perform ***multicast*** of ***search messages (ssdp:discover)***
  - Any device that receives a multicast message may reply with a ***unicast response message***
- The URL of XML Device Description File is returned back to the control point

UPnP Architecture and Protocol Stack

Usage of HTTP over UDP, either multicast (HTTPMU) or unicast (HTTPU)



Description of Device and its Services in UPnP

- UPnP uses ***XML to describe resources and services*** (standardization effort for interoperable representation)
- ***Advertisement message includes a URL*** related to the XML device description file
- Device description file describes the ***capabilities of the device*** for which advertisement is done
- Control point can dynamically obtain the device description file via HTTP and process it
- Any device can offer one or more resources/services
- `<service>` element includes
  - Service type and service ID
  - Service URL for ***invocation via SOAP***
  - URL for event subscription to enable ***notifications***
  - An additional file (Service Description File) for any offered service, with more specific and detailed descriptions

Device Description File for a Projector

projector-desc.xml

```
<?xml version="1.0" ?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
<device>
  <deviceType>urn:schemas-upnp-
   org:device:projector:1</deviceType>
  <UDN>uuid:UPnP-Projector</UDN>
  <serviceList>
  <service>
  <serviceType>urn:schemas-upnp-
   org:service:control:1</serviceType>
  <serviceId>urn:upnp-org:serviceId:control</serviceId>
  <controlURL>isapictl.dll?control</controlURL>
  <eventSubURL>isapictl.dll?control</eventSubURL>
  <SCPDURL>projector-scpd.xml</SCPDURL>
  </service>
  </serviceList>
</device>
</root>
```

Service Description File for a possible "projector control" service

Service Control in UPnP

- The XML-based service description file (e.g., “projector control”) contains:
  - ***Action list with the operations*** that may be invoked on the service
  - ***Service state table*** including all exposed state variables (and their data type)
- To invoke a given service control for which a device has previously performed advertisement, ***control point sends a SOAP message to the URL specified*** for that service
  - ***Control point can access and update state variables in the table***
- Service performs the requested control action and ***returns the result via SOAP message***

Service Description File for a Service of Projector Control

projector-scpd.xml

```
<?xml version="1.0" ?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
<actionList>
<action>
  <name>SetPower</name>
   <argumentList>
        <argument>
        <name>Power</name>
         <relatedStateVariable>Power</relatedStateVariable>
               <direction>in</direction>
        </argument>
      </argumentList>
</action>
 … Other actions …
</actionList>
```

Service Description File for a Service of Projector Control

```
…
<serviceStateTable>
    <stateVariable sendEvents="yes">
        <name>Power</name>
        <dataType>Boolean</dataType>
        <defaultValue>0</defaultValue>
     </stateVariable>
    <stateVariable sendEvents="yes">
        <name>File</name>
        <dataType>string</dataType>
        <defaultValue>default.ppt</defaultValue>
    </stateVariable>
… Other state variables …
</serviceStateTable>
</scpd>
```


Event Subscription in UPnP

- ***Control point may register itself for receiving notification events*** generated by advertised services when ***state variables are modified***
  - ***Subscription URL*** included in device description file
- Messages that implement the notified event are ***expressed in XML and formatted according to the General Event Notification Architecture (GENA) standard***; they include the modified state variables that caused event generation

For example, projector service can notify events towards control point when one of the following situations occur:

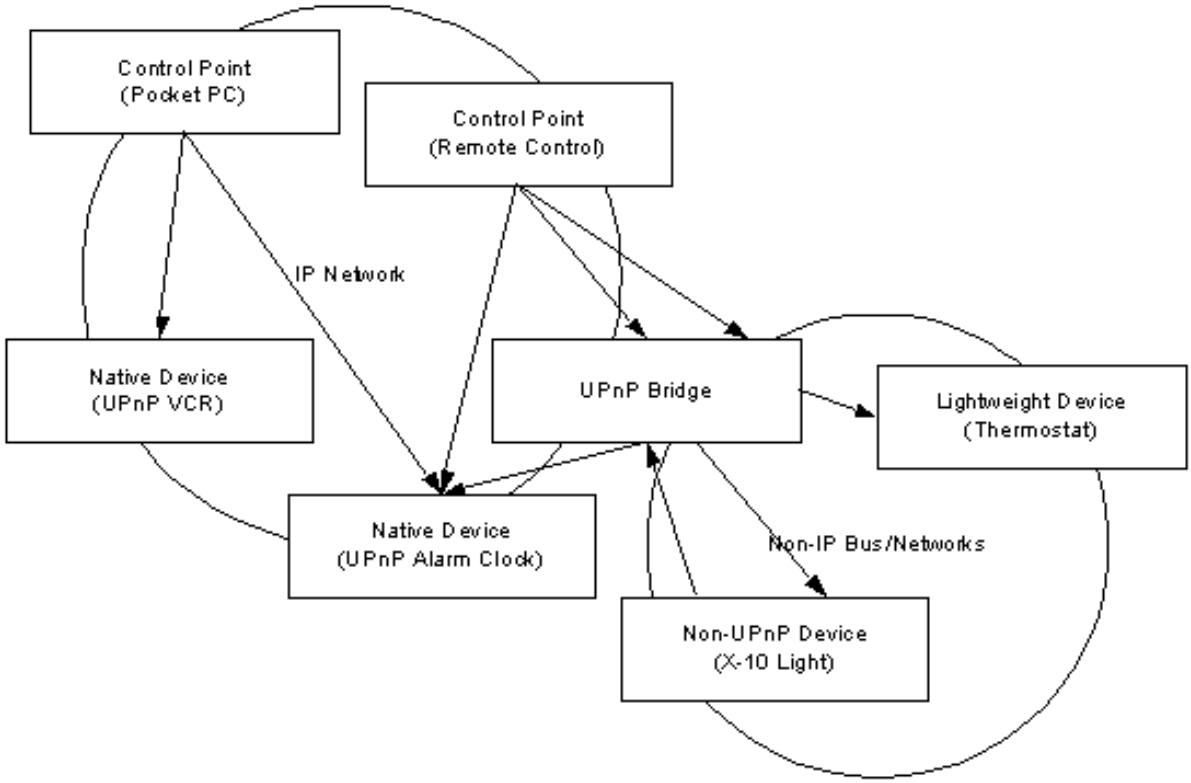
- Page up/down (change of pageNumber variable)
- Power on/off
- Files – change in ppt file list
- File – change in ppt file currently with ongoing presentation

UPnP ***does NOT allow subscription of control points to single state variables***; control point has to dynamically determine which state variable has been changed and has generated notification event

UPnP Presentation is based on HTTP

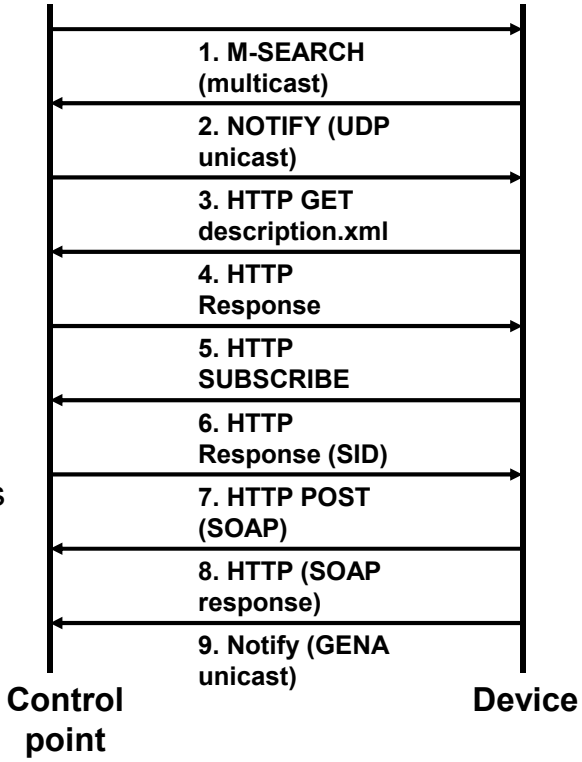
- Device may ***advertise a "presentation" URL for user interface describing service access via Web***:
  - Download Web page from URL and visualization at browser
  - Possibility for users to control the device
  - Possibility to visualize device state
- ***Given the usage of XML*** for data definition and exchange, ***UPnP potentially enables*** employment by a ***large set of resource-limited devices***: also ***automated XML transformations (reductions) based on XSLT***
  - Most UPnP implementations support only presentation based on HTML, slow transition towards XML

UPnP Architectue and Bridging Possibilities



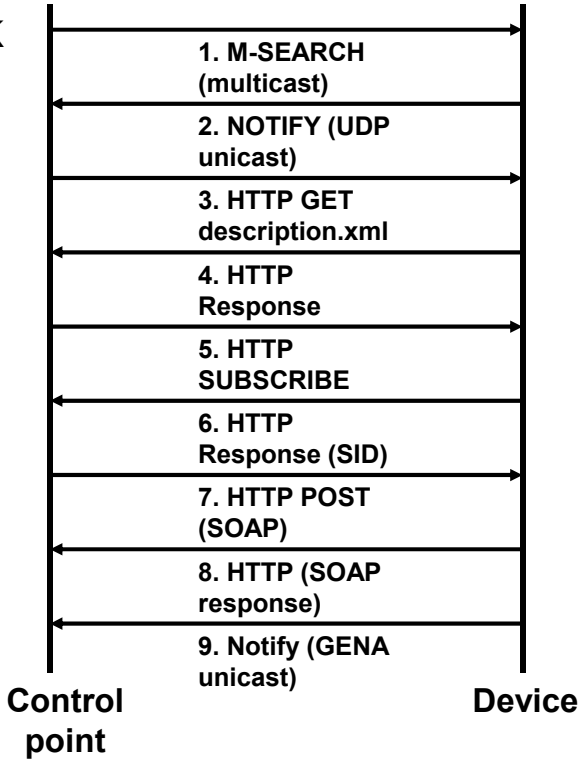
Details on Service-Control Point Interaction

1. Control point sends SSDP search request
2. Device replies with unicast UDP NOTIFY, which contains the URL of XML file with device description
3. Control point requests XML description document via HTTP
4. ***Web server included in the device*** replies to request and returns XML document
5. To automatically receive notifications of changes at device, control point can register itself to the services of interest via HTTP



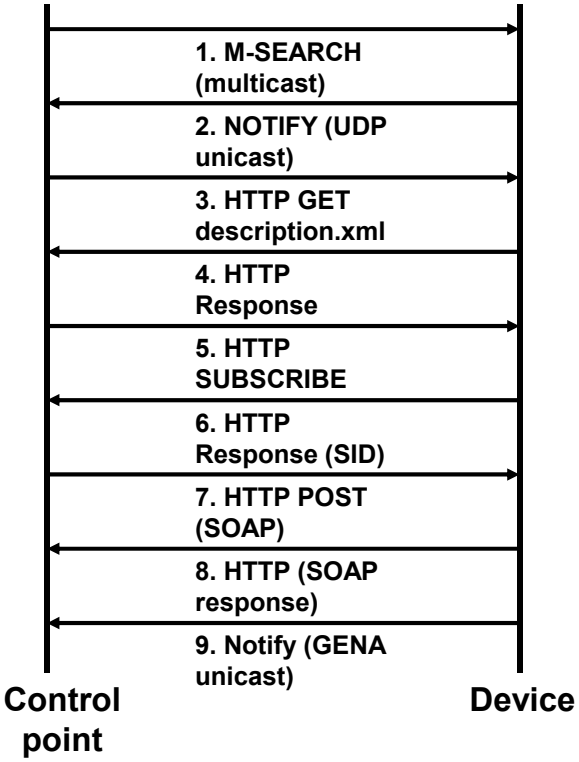
Details on Service-Control Point Interaction

6. Device replies with registration ACK and returns unique Subscription Identifier (SID)
7. Control point can command the execution of operations at device, with possible modification of state variables
   - URL to send control requests included in the XML document with device description
   - Control point sends SOAP request over HTTP



Details on Service-Control Point Interaction

8. Device possibly changes state variables and returns response as a SOAP message
9. Device can notify clients of the occurred state change, either stemming from invoked actions (as in the case of 8) or generated by implicit modifications at device
   - Device performs notifications via unicast NOTIFY messages over HTTP



By Summarizing… UPnP Middleware Features

- Service Discovery
  - Support designed for peer-to-peer environments without hierarchical structuring
- Adaptability
  - IP addresses are dynamically allocated
  - State modifications are made available via event notification
  - No support (yet) for service routing/selection based on client location
- Transparent support to communication
  - Exploitation of Internet standards
  - No support to multi-hop ad-hoc communications
- Data Transformation
  - Possible data transformation from standard XML format to proprietary formats that may be control-point-specific (e.g., proprietary Microsoft ones)

Additional Useful Links about UPnP

- S. Helal, “Standards for Service Discovery and Delivery,” *IEEE Pervasive Computing*, Vol. 1, No. 3, pp. 95-100, July-Sept. 2002
- Jini Forum, at http://www.jini.org/
- Service Location Protocol Working Group (svrloc), at http://www.ietf.org/html.charters/svrloc-charter.html
- UPnP Forum, at http://www.upnp.org/
- UPnP Forum, “Universal Plug and Play Device Architecture,” at http://www.upnp.org/resources/documents.asp
- Intel, “UPnP Technology,” at http://www.intel.com/technology/UPnP/

Exercise on UPnP (1)

To design and implement a small application that uses ***UPnP to discover the availability of file multimedia files*** offered in a locality

To try to respect, as much as possible, the design architectural choice of ***out-of-band multimedia communication (direct between end points) wrt discovery***

Situation close to realistic scenario where UPnP is used as solution for configuration, discovery, and service access in home-oriented networks, ***in particular for media servers, rendering devices, data sources, control points, …*** (see the approach supported by Digital Living Network Alliance – DLNA - http://www.dlna.org/)

Please refer to docs and development tools, largely available in the community, such as:

- Microsoft, “Using the UPnP Control Point API”, http://msdn.microsoft.com/en-us/library/ms898948.aspx
- ***Cling*** - Java/Android UPnP library and tools (Java sw stack compliant with UPnP), http://teleal.org/projects/cling/
- ***CyberLink*** (Java/Android), http://www.cybergarage.org/twiki/bin/view/Main/CyberLinkForJava

Exercise on UPnP (2)

Other reference development tools and management instruments, widely adopted in the developers' community:

- Reference tool for testing; it offers media servers, media renderers, spies, controllers, ... useful to test and validate applications; it also offers a C# stack to create UPnP devices and services
  http://opentools.homeip.net/dev-tools-for-upnp
- ***Coherence*** (for development in Python)
  http://coherence.beebits.net/
- ***BRisa***, for both Python (UPnP 1.0) and qt (UPnP 1.1), specifically designed for the Maemo platform
  https://garage.maemo.org/projects/brisa

As usual, the exercise could be a starting seed for a possible further project activity…

Alternatively, Solution based on Apache River

To design and implement a small application that uses ***Apache River to discover the availability of multimedia files*** offered in a locality

To try to respect, as much as possible, the design architectural choice of **out-of-band multimedia communication** (direct between end points) wrt discovery

In this case, please refer to docs and development tools available at:

- River home page - http://river.apache.org/
- River Starter Kit - http://river.apache.org/user-guide-basic-river-services.html
- StartNow project - http://java.net/projects/startnow/