In-Home Network Middleware Standards and interoperability

Peter van der stok

Thanks to Rob Udink, Andras Montvay and Michael van Hartskamp

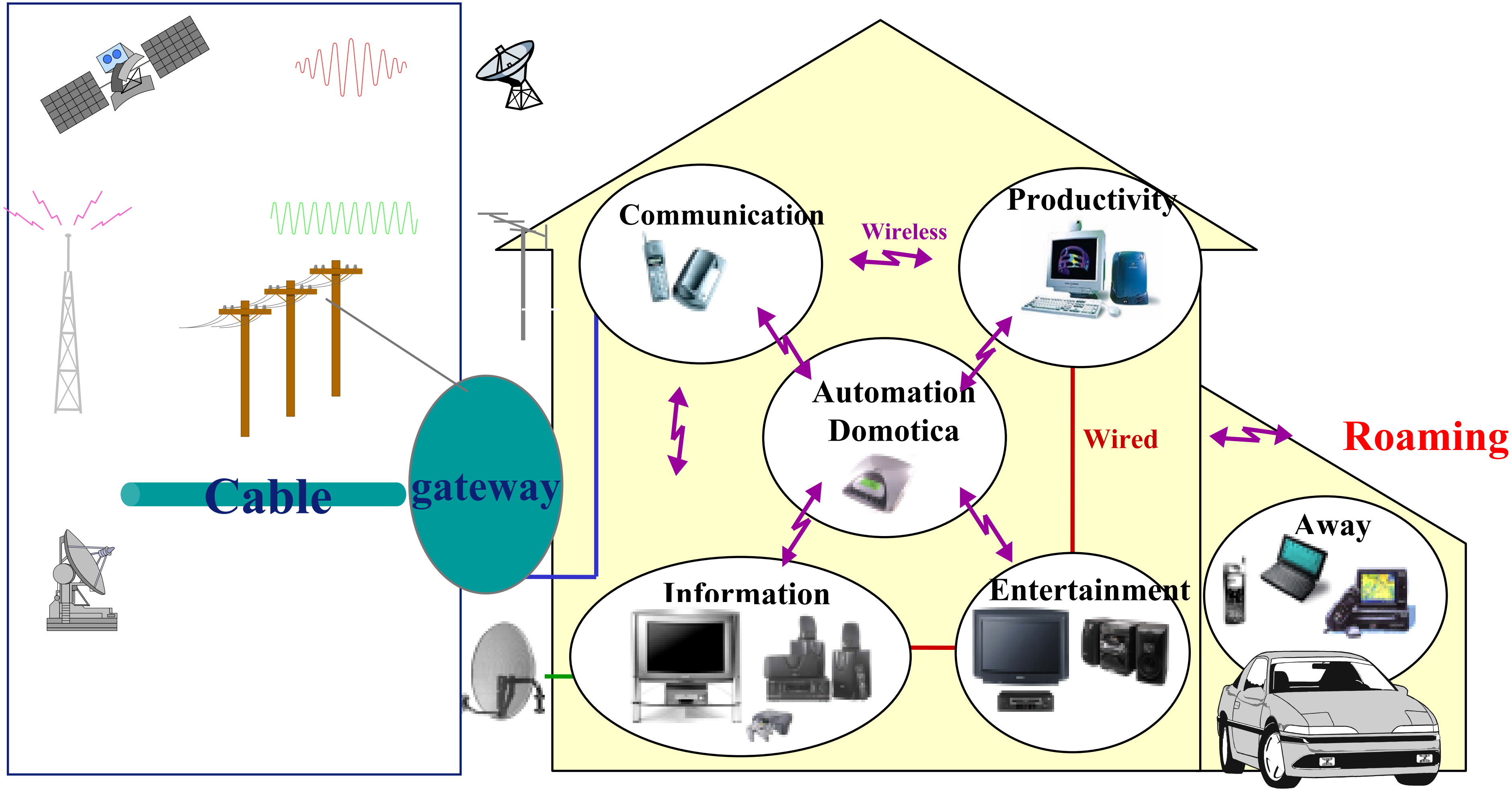
22 June 2005

Universitat Politècnica of Catalunya

PHILIPS

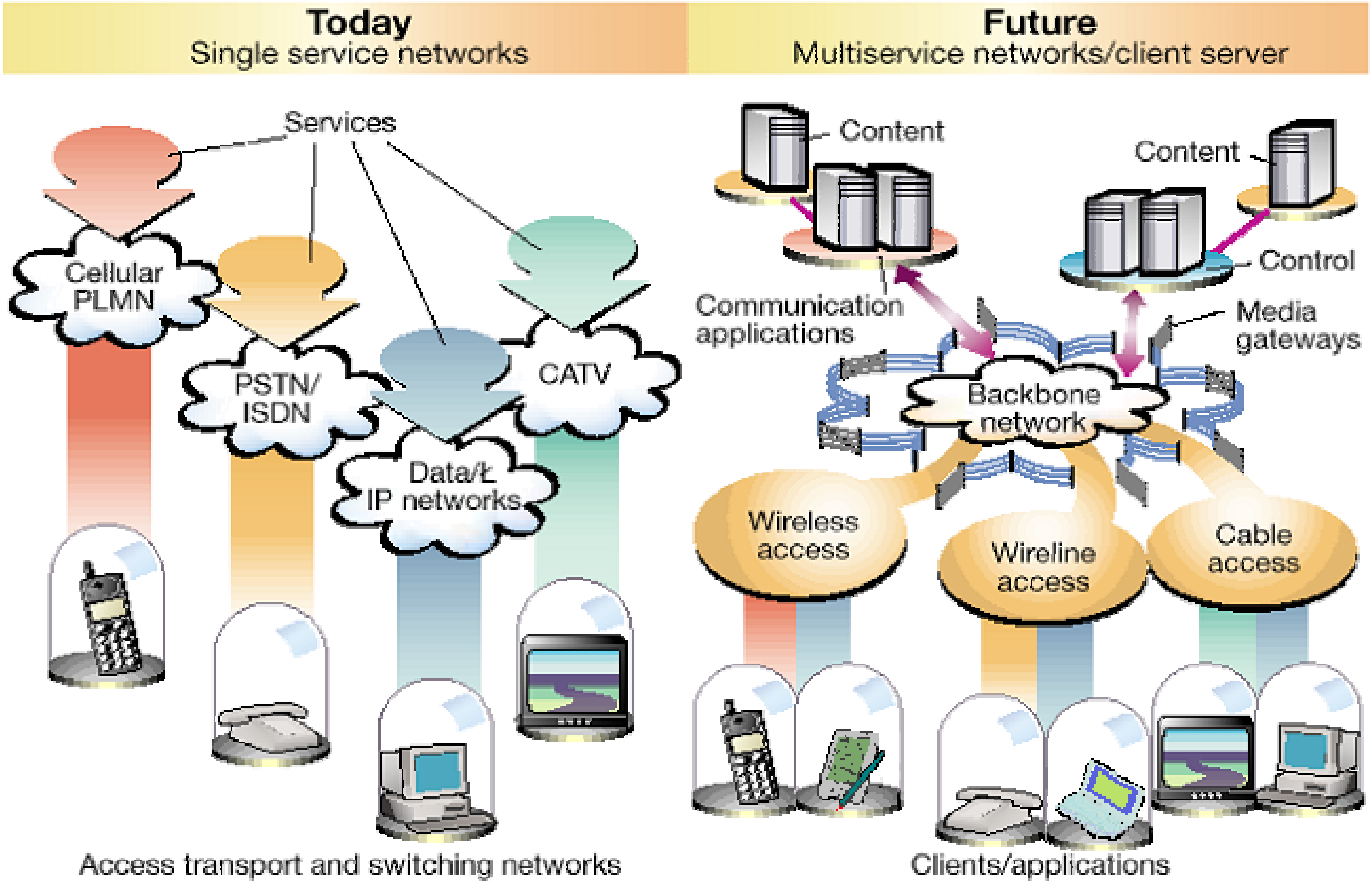
Contents

In-Home network connections



PHILIPS

Convergence of Networks



In-Home Digital Networks

Automation Networks

Security, Lighting, . . .

Low Bandwidth (<2 Mbps)

Powerline, CEBus, X-10, LonWorks

Communication Networks

Phone

Low/medium Bandwidth (1-10 Mbps)

Bluetooth, USB

Information Networks

PCs, Peripherals

Medium Bandwidth (10-100 Mbps)

TCP/IP, Ethernet, Home PNA, Home RF

Entertainment Networks

AV Devices

High Bandwidth (100-400 Mbps)

IEEE1394, switched Ethernet

Realities of the CE World

- Devices
  - heterogeneous: processing power, vendor, UI, ...
  - different interaction model and UI modalities
- Network
  - heterogeneous: bandwidth, wired/wireless, QoS, ...
  - unplanned topology, built incrementally
  - dynamic, esp. for mobile devices
- Users
  - diverse, broad group of users
    - Age, education, ….
  - untrained, unwilling to follow complicated procedures

Requirements for CE

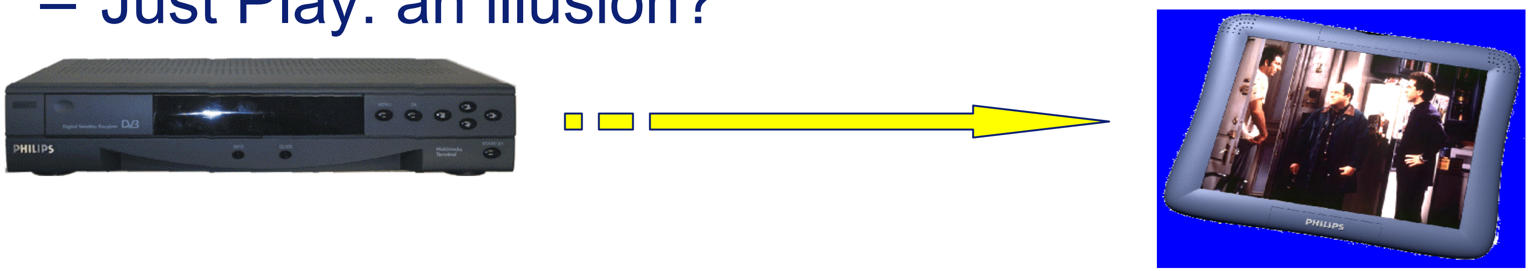
- User requirements
  - Quality
  - Just Play
  - User Interface
  - Dependability
- System requirements
  - e.g. bandwidth, protocols, codecs, architecture, ...

Requirements - Quality

- New technologies start in the high-end sector
- Expectations: comparison to current high-end equipment
  - Exception: completely new types of apps (GSM: low speech quality acceptable at first) - BUT: improvements slowly become necessities
- Quality can be: image / sound quality, reaction time, ...

Requirements - Just Play

- Compatibility
  - backward / forward
  - even to old devices & content
- Interoperability
  - cross-vendor, ...
- Unplug and still play
- Expectations of users, how to raise the right expectations
  - Just Play: an illusion?



Need for standards in CE Home Networks

- for dealing with devices from different vendors
  - interconnection between these devices
  - exchanging (streaming) data
  - remote (device) control

- for access to outside services from different suppliers
  - e.g. telephony, radio, television, DVB-MHP, (broadband) internet, ..

- for access from outside services to your house
  - remote monitoring, OSGi

CE Peculiarities

- Manufacturer of devices want to decide own look&feel
- Technology development goes (too) fast
  - standards are state of the art only shortly after launch
  - however, when adopted they are not easy to replace
  - slow and clear evolution path necessary
- Several middleware standards exist for partly the same services
  - e.g. HAVi, UPnP, JINI, DLNA

Conclusions

- Heterogeneous IHDN & Devices
- User requirements for CE-IDMS
  - Quality
  - Just Play
  - User Interface
  - Dependability
- Special needs for middleware standardization

Contents

Three Standards - One Goal?

HAVi “... provides a set of services which **facilitate** interoperability and the development of distributed applications on home **networks**.”

–8 CE companies (Philips, Sony,….), …

Jini “[‘s]... overall goal is to turn the **network** into a flexible, **easily**-administered tool with which resources can be found by human and computational clients.”

–Sun, …

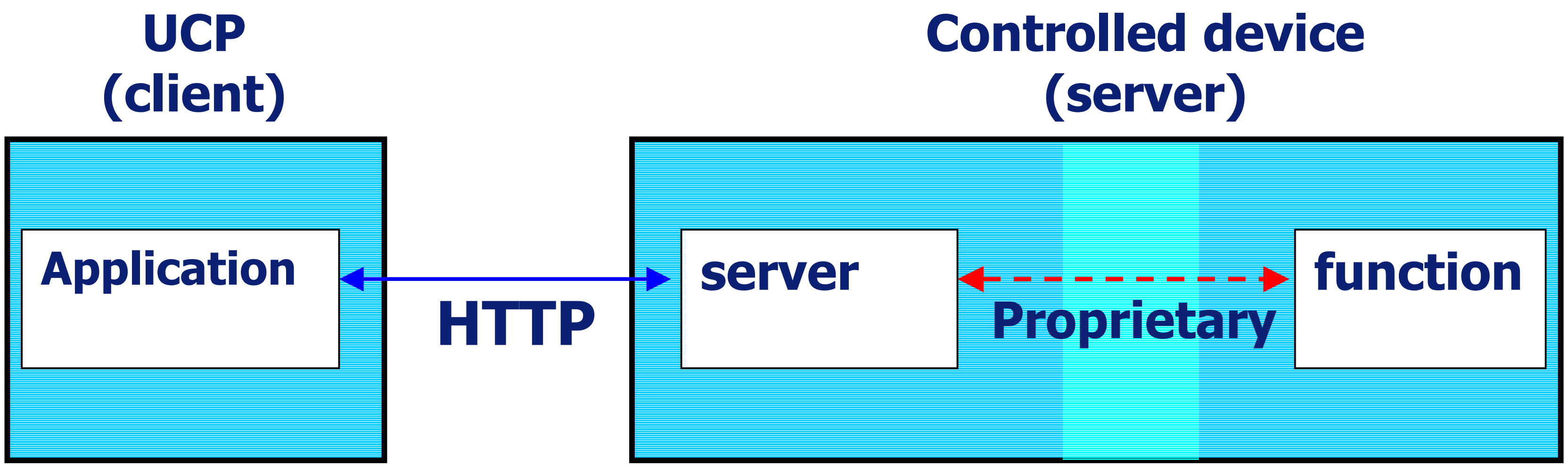
UPnP “... is designed to bring **easy**-to-use, flexible, standards-based connectivity to ad-hoc or unmanaged **networks** whether in the home, in a small business, or attached to the Internet.”

–Microsoft, …

Universal Plug and Play (UPnP)

UPnP network consists of the following logical nodes:

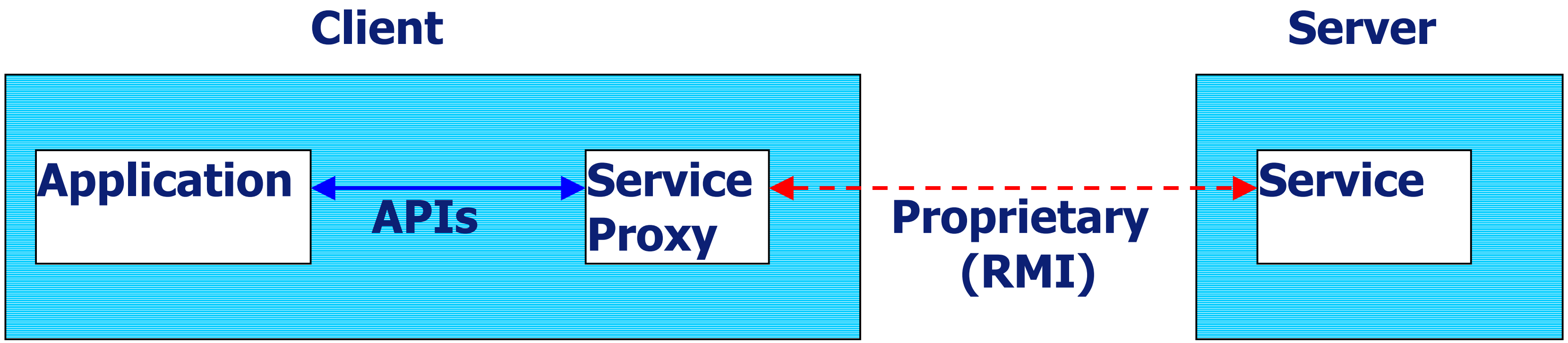
- client - User Control Point (UCP), e.g. a PC, a digital Settop-box
- server - Controlled device, a VCR, a DVD player, a TV set.



Jini - Services and Proxies

Service Proxy is downloaded on demand to the client side

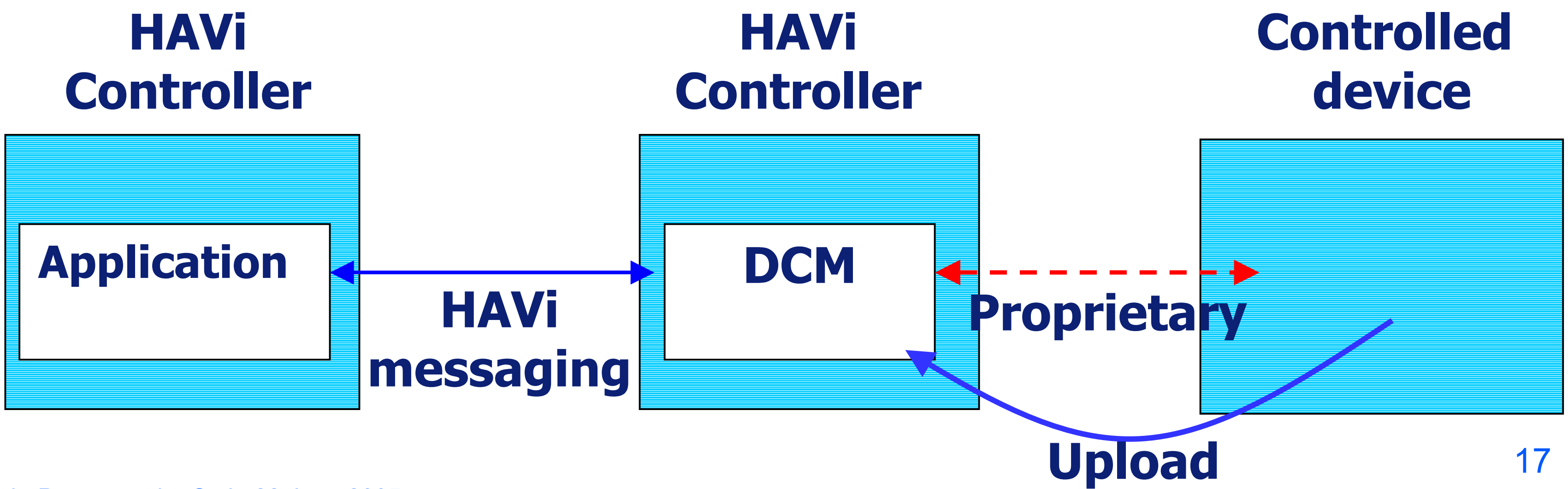
- it can perform the service itself
- it can (but does not need to) be an RMI stub
- it can have a private communication protocol



Home Audio Video interoperability (HAVi)

HAVi network consists of the following logical nodes:

- HAVi Controllers
  - host services and applications
  - must have HAVi stack embedded
- HAVi Controlled Devices
  - host the real functionality, e.g. VCR, DVD, TV-set



HAVi - Device Classification

Controller devices

Full AV device (FAV)
- Must provide Java run-time environment
- Set of Java packages as defined by HAVi
- Download and execute HAVi applications
- Download (host) and execute DCM for BAV / LAV

Intermediate AV device (IAV)
- No Java specific requirements
- Can have display

Controlled devices

Base AV device (BAV)
- May provide uploadable Java byte-code control-info in ROM

Legacy AV device (LAV)
- Conventional device with NO knowledge of HAVi

Comparison of HAVi, Jini and UPnP

HAVi

- complete set of functionalities
- resource and stream management
- support of legacy devices and P&P
- strong UI support
- IEEE 1394 “tied”

UPnP:

- use of well-established web techniques
- independent of the underlying physical media
- simple, but limited UI
- no resource management
- limited support of A/V streams
- infrastructure required on controlled device

Jini

- “lightweight” but extensible
- allows for smart proxies
- support for transactions
- independent of the underlying physical media
- no support of legacy devices and A/V streams
- no specific service definitions

Contents

1. Environment, network CE requirements
2. Home standards
3. Technologies
4. Internet Protocol
5. UPnP

“Just Play” Problem Statement

- User requirement: “Just Play”
- Plug and play is the “easy” part!
- Unplug and still play
  - Directly involved resources, switch to alternative ones
  - Resources not directly involved, e.g. a stream manager or a device host
- Roaming
  - Another form of unplug (and plug back in elsewhere)

- No complete solution to-day

Functionality Discovery v.s. Plug-in Discovery

- Functionality Discovery
  - High level functionalities, e.g. printing, storage, ...
  - Jini Lookup of Services
  - HAVi Registry for DCMs and FCMs
  - UPnP Discovery for Devices and Services
- Plug-in Discovery
  - Low level, establishing basic communication
  - Physical plugging in of a device
  - 1394 bus-reset, HAVi DCM Management, Auto-IP / ARP
- User thinks of boxes - developers think of functionality
  - In “classical” CE, there is often a 1-1 relation

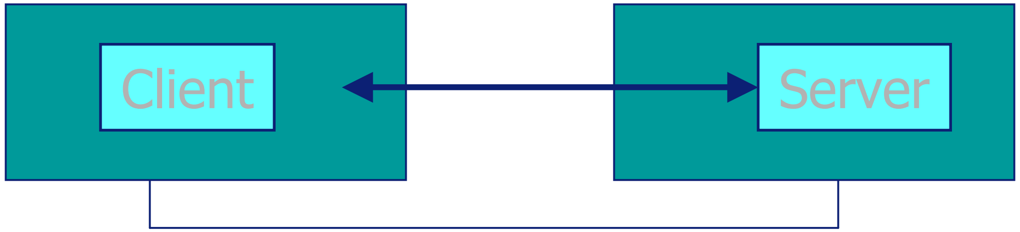
Plug-in Discovery Mechanisms

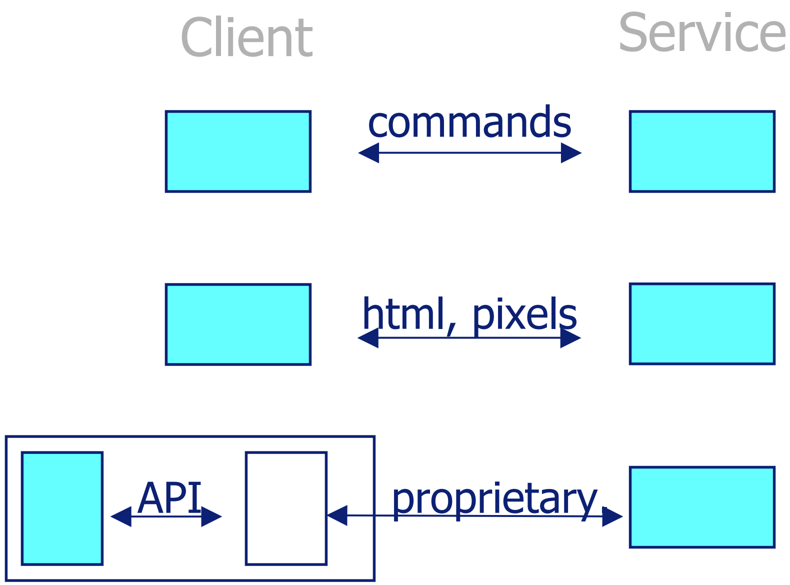
- IEEE 1394 (“FireWire”)
  - Built-in plug and play support on physical level
  - Address re-assignment after “bus reset”
  - Streams can be continued after bus-reset
- Ethernet (IEEE802.3)
  - None, but can be built on top
- Auto IP, Zeroconf
  - Link local IP addresses are probed, then taken
- Wireless networks
  - Beacons sent by base station
  - Heartbeat sent by mobile terminal

Service Discovery Mechanisms

- Registry based
  – Jini: central (lookup service)
  – HAVi: distributed (registries on controllers)
- Peer to peer
  – UPnP: announcements, queries (SSDP)
- Combined
  – Internet Protocol: Service Location Protocol (SLP)
- Dealing with unreliable networks
  – UPnP: broadcasts sent 3 times, TTL=1/255, advised to be 4/252
  – Jini: leasing mechanism
  – HAVi: not needed, as IEEE1394 is “reliable”

Client-Server Models





Different types of communication:

- Command based
- UI based
- Up/Down-loading
  - Possibly followed by proprietary communication

Motivation

- CE Devices can only support few protocols
  - So choose for best possible model
- Where are the “special features”
  - What can manufacturer decide, what is decided by others
- Consequence for who decides look&feel
  - CE manufacturers want to have control
- Different need for standardization

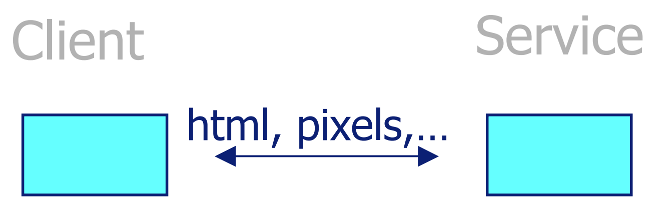
Command Based Communication

- Application and service exchange "commands"
  - e.g. play, stop,…
- Application fully determines functionality
  - ⇒ full control in application
- Difficult to standardize completely

Supported by:

- HAVi: "FCM" command sets (tuner, display, …)
- UPnP: service descriptions (AV Transport, …)
- Jini: in concept, but no specific command sets

UI Based Communication



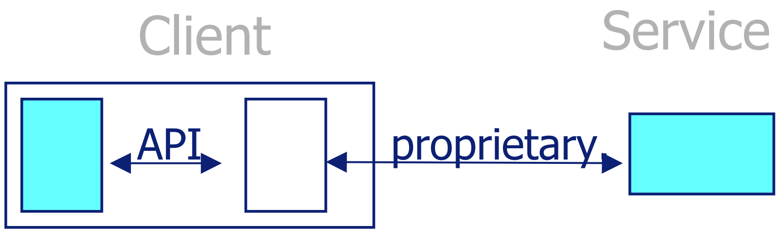
- Application and server exchange UI information
  - E.g., show button, button pushed
- Client presents UI to the user
  - Is a kind of browser

- Application has no control over functionality,
  - ⇒ all functionality is in the service

Supported by:

- HAVi: DDI level 1 user interface,
- UPnP: HTML based web pages (+scripting)

Downloading Based Communication

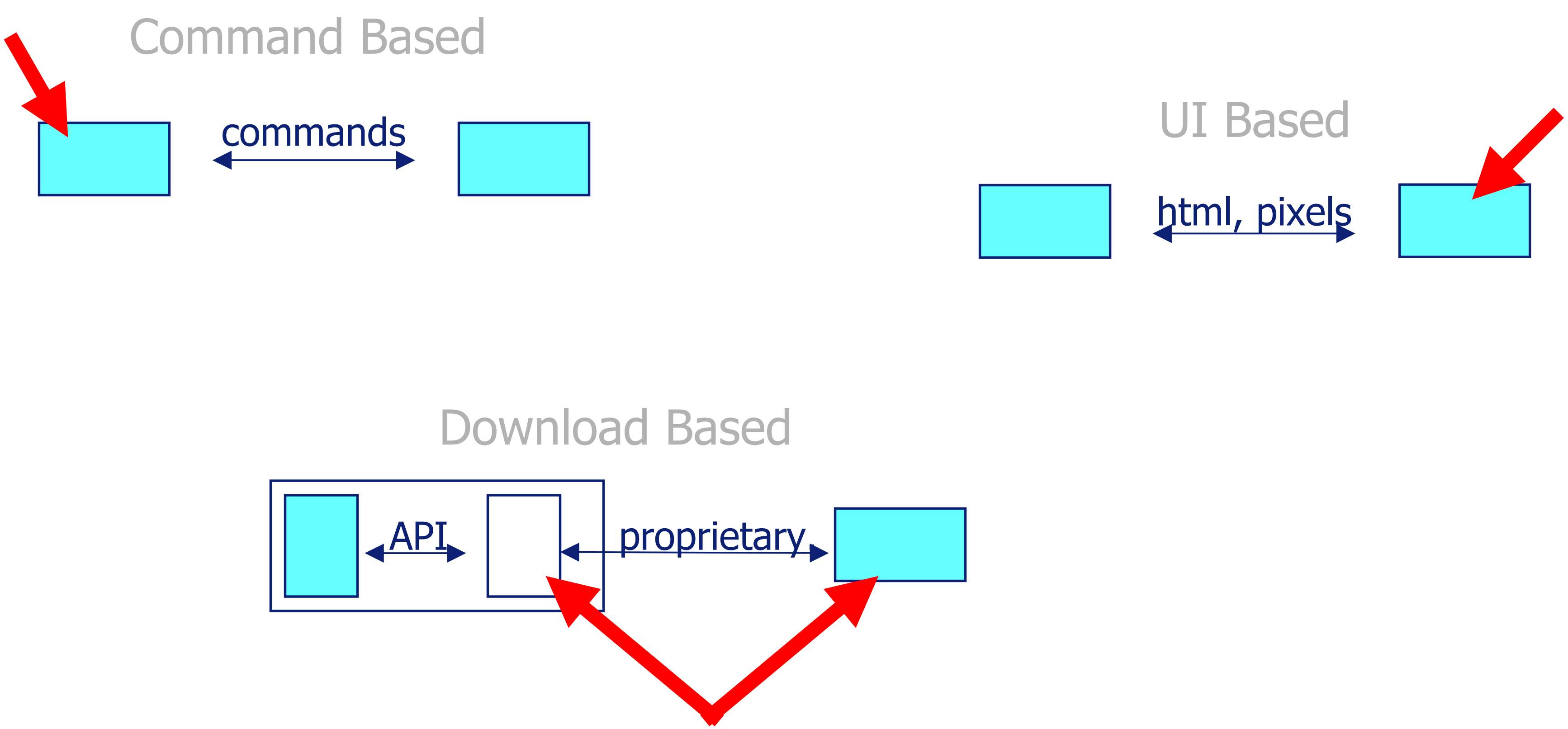


- Service provides downloadable application
  - E.g. applets
- Application has no control over functionality
  - ⇒ all control provided by service or uploaded code

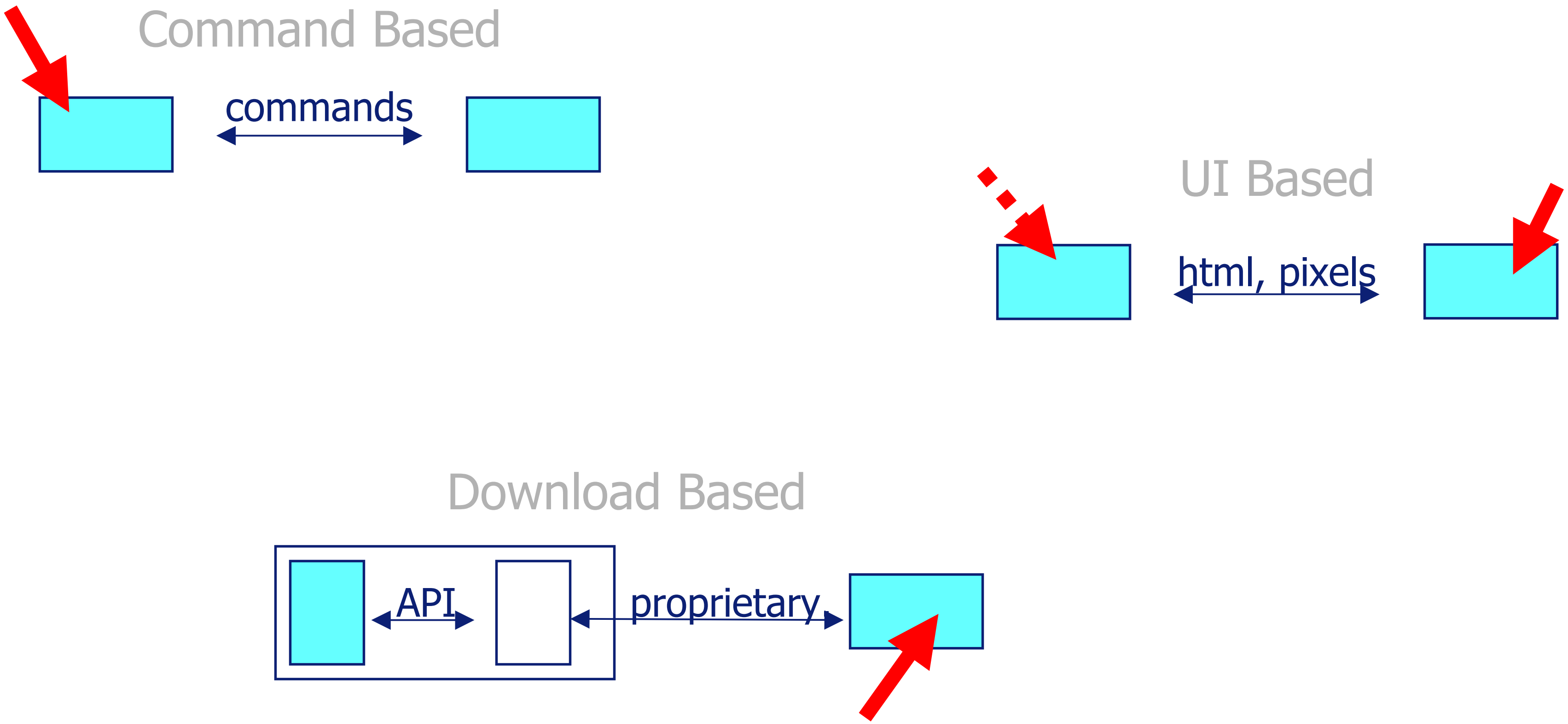
Supported by

- HAVi: Havlets
- Jini: applets

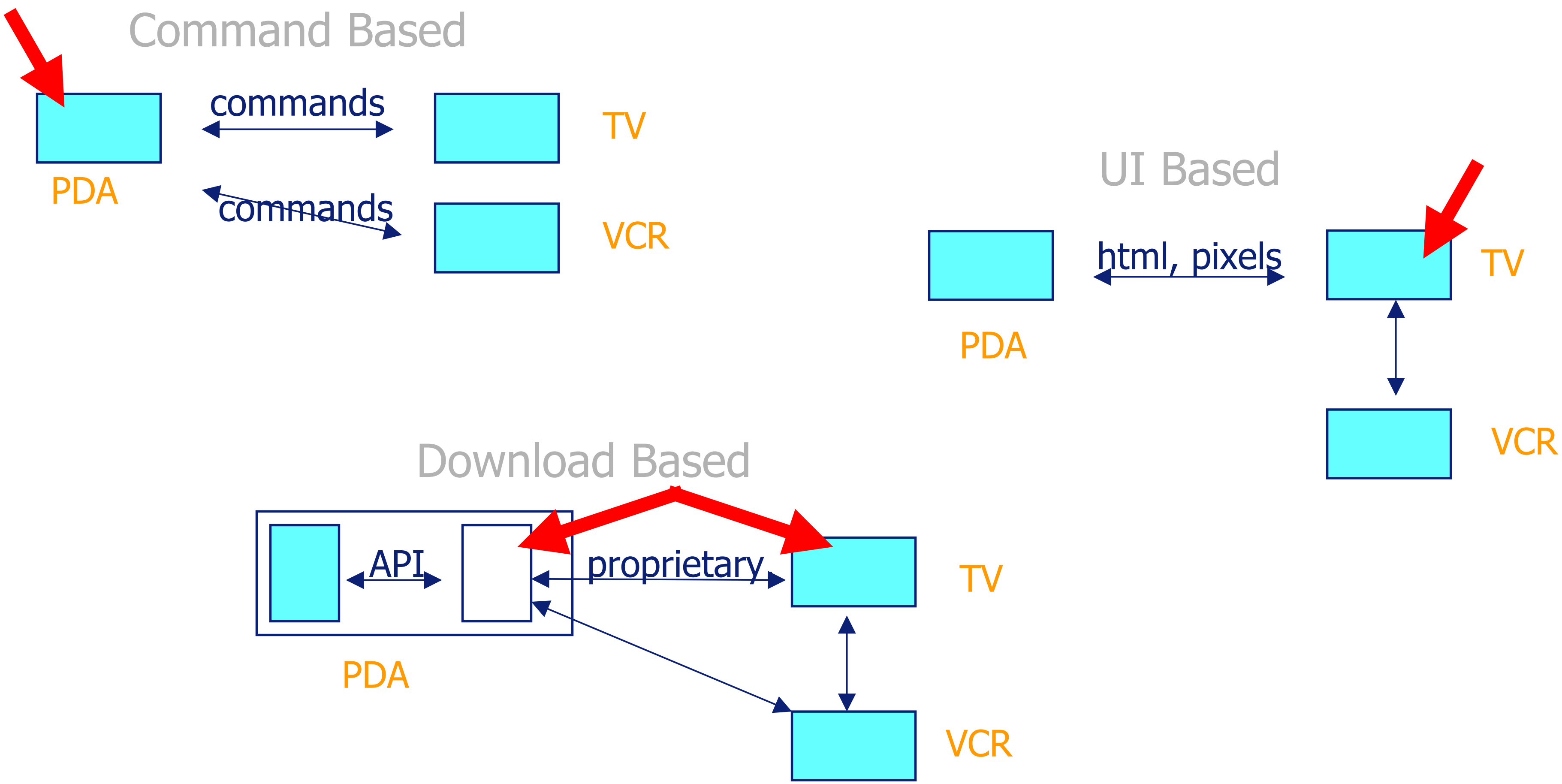
Where is the Real Application / Functionality



Who decides the Look and Feel?



Client - *multi* Server



Conclusion

- Different communication models have different characteristics
  - Where is the application
  - Who decides the look&feel
  - Multi server handling

- Different models are not alternatives, but complement each other
  - Several standards support multiple models

PHILIPS

Contents

1. Environment, network CE requirements
2. Home standards
3. Technologies
4. Internet Protocol
5. UPnP

Link-local Addressing

Use 169.254/16 addresses

Hosts can choose an address in this range, except for the first 256 and last 256 addresses. So actually 169.254.1.0-169.254.254.255 may be used.

TTL of IP packets must be 255

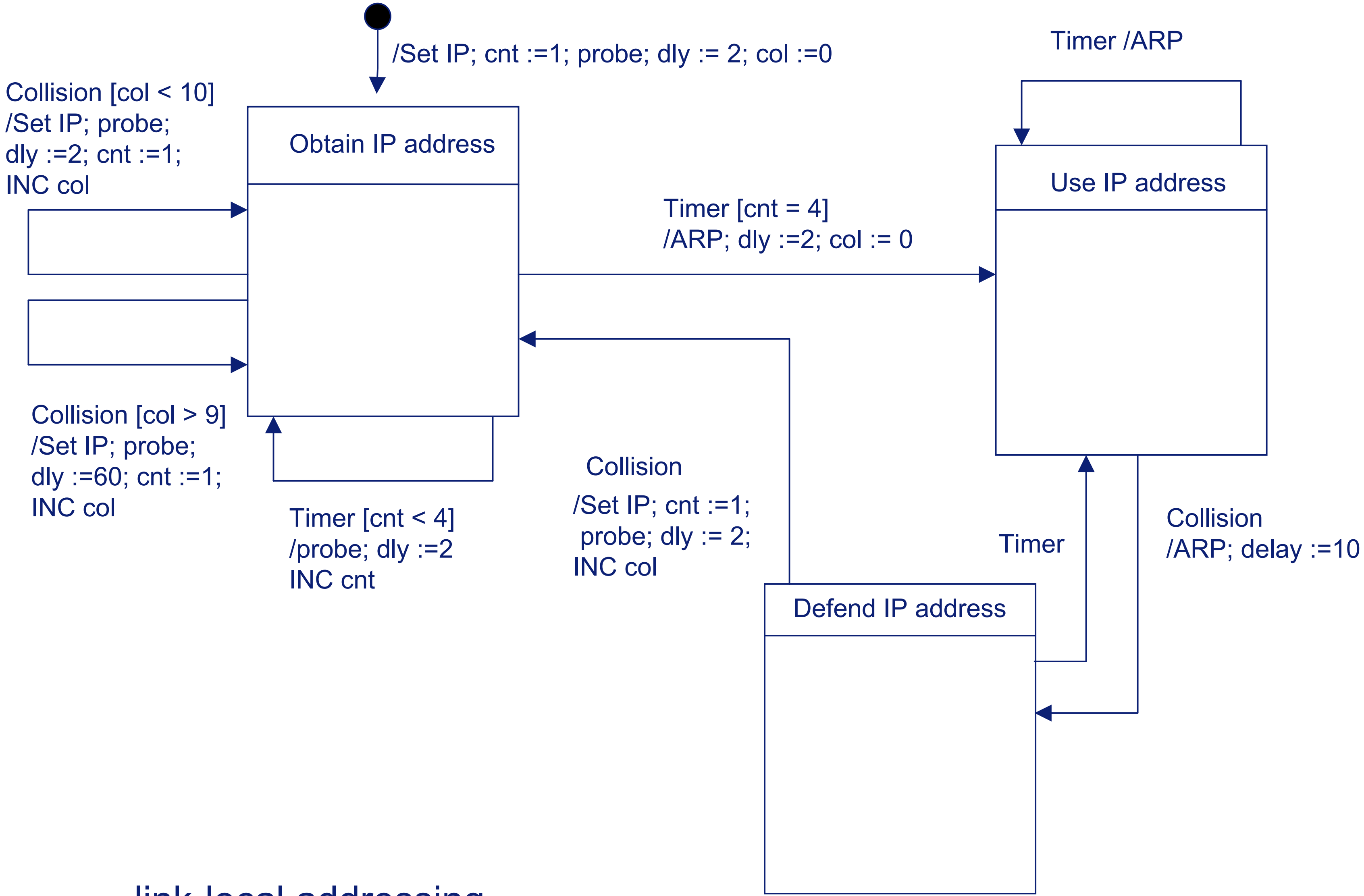
IP packets sent from a link-local address must have the TLL set to 255. A host receiving an IP packet on a link-local address, must check if the TLL is 255. This guarantees that link-local IP traffic can't pass a router, and the scope of the link-local network is limited to "the link".

How it works

Basic steps
1. Choose a random IP address in the link-local range
2. Check the availability of the address (takes 8 secs)
3. Use the address (configure the network device)
4. Announce that you're using it (send ARPs)
5. Wait for a collision to occur

Handling a collision can be done in 2 ways
- Drop the address immediately (go to step 1 again)
- Defend the address

Defending the address means
- Announce that you're using the address (send ARP)
- Wait 10 seconds for another collision:
   - *collision -> drop IP immediately (go back to step 1)*
   - *time passed -> continue using the address*



link-local addressing
state diagram

A few issues

- It takes at least 8 seconds to get online
- Handling a conflict is non-deterministic in the protocol
- Waiting 10s defending an address might be too short

It takes at least 8 seconds to get online

Why?

To check the availability of an address, four probes are sent. Between each probe there is a 2 seconds delay, so the complete check takes at least 8 seconds. If at any time a probe is replied by a host, this means that the address is not available, and a different address must be chosen.

Solutions?

- Compress the check to a shorter amount of time
- Reduce the number of probes
- A combination of the above

Handling a conflict is non-deterministic

Why?

The protocol document gives two options for handling a conflict. Either one is valid, and nothing is told about the decision.

Solutions?

- Always defend (easiest option)
- Make an API construction to let applications decide what to do
- Make the decision locally (random or deterministic)
- Let the colliding hosts negotiate

Dynamic Name Service (DNS)

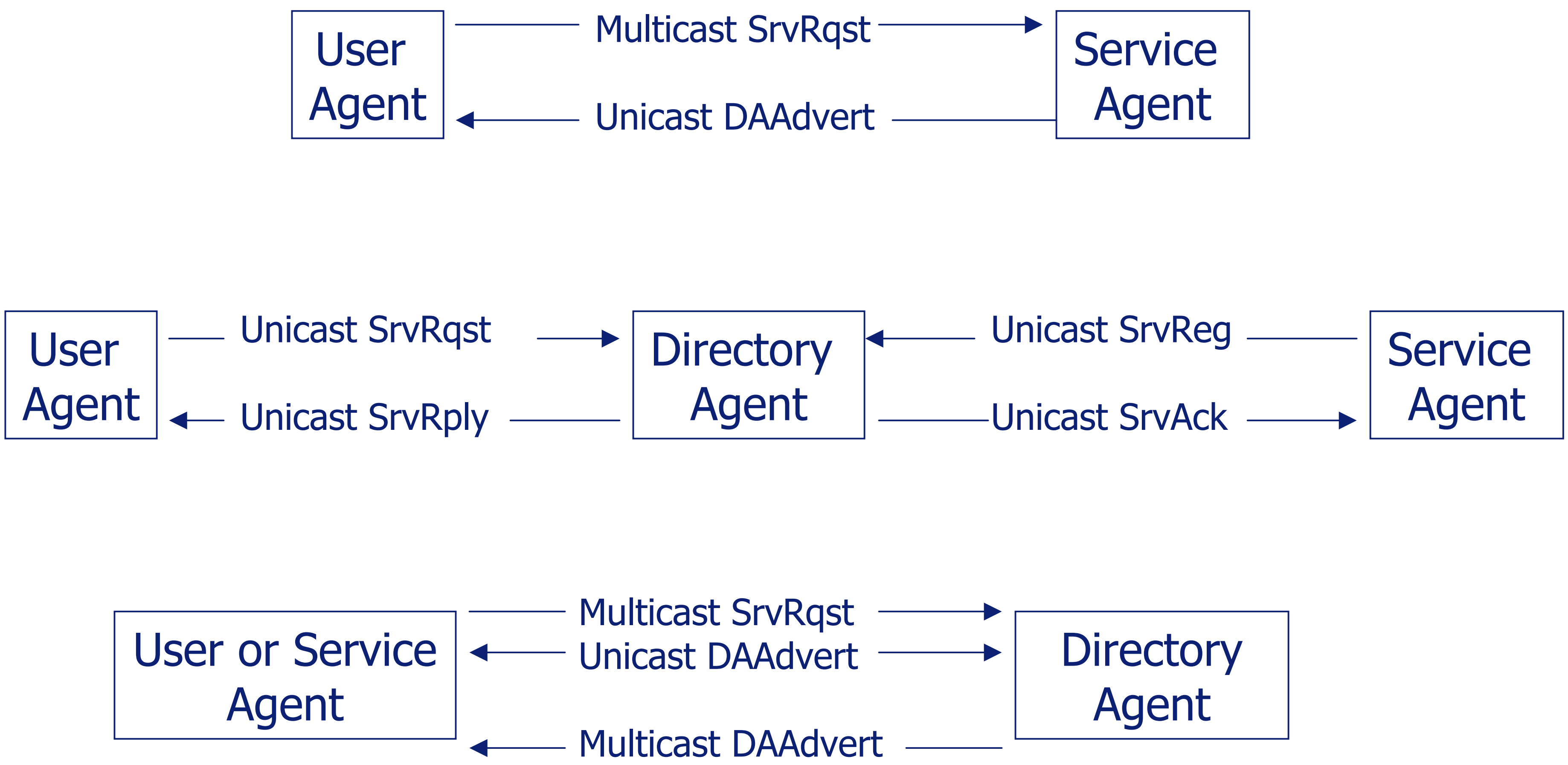
- Configured by hand
- Dynamic update
- Authoritative server required
- Relates IP address to name (name relates to device)
- IP address change involves server contents change

multicast DNS (mDNS)

- Each node is authoritative for itself
- Node multicasts name + IP address
- Request for IP address answered by involved node

Advantage:	name to device
A-D?	caching with TTL
Disadvantage:	no unique names

Service Location Protocol (SLP)



Issues

MC Request sent till no answer is returned or all destinations answered

Location is given as host name or IP address

Entries in Directory have life time

Replaces DNS server? Symbiosis with mDNS?

NO XML descriptions

All services returned after request, UPnP (two invocations)

Easy transition from directory to directory-less

Contents

Device Control via UPnP™

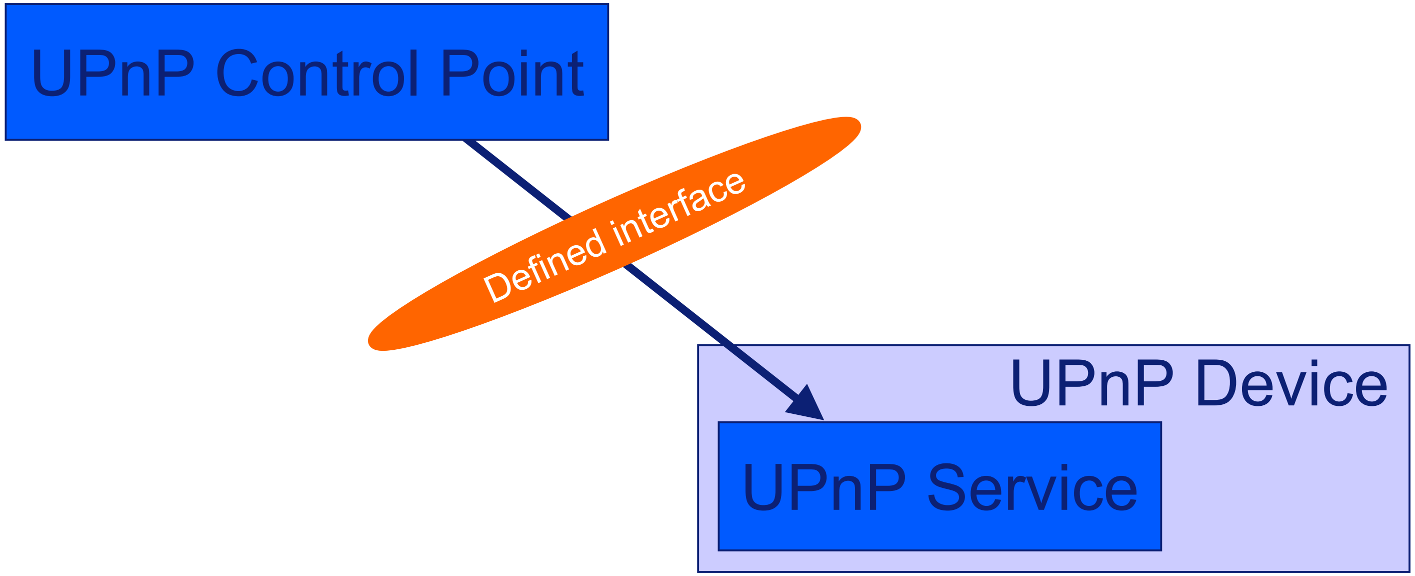
The UPnP™ Forum is an industry initiative designed to enable simple and robust connectivity among stand-alone devices and PCs from many different vendors. As a group, we are leading the way to an interconnected lifestyle.

UPnP is a middleware for distributed applications; it contains parts which are common for many distributed applications

UPnP is seen as the dominant home networking standard for the coming years

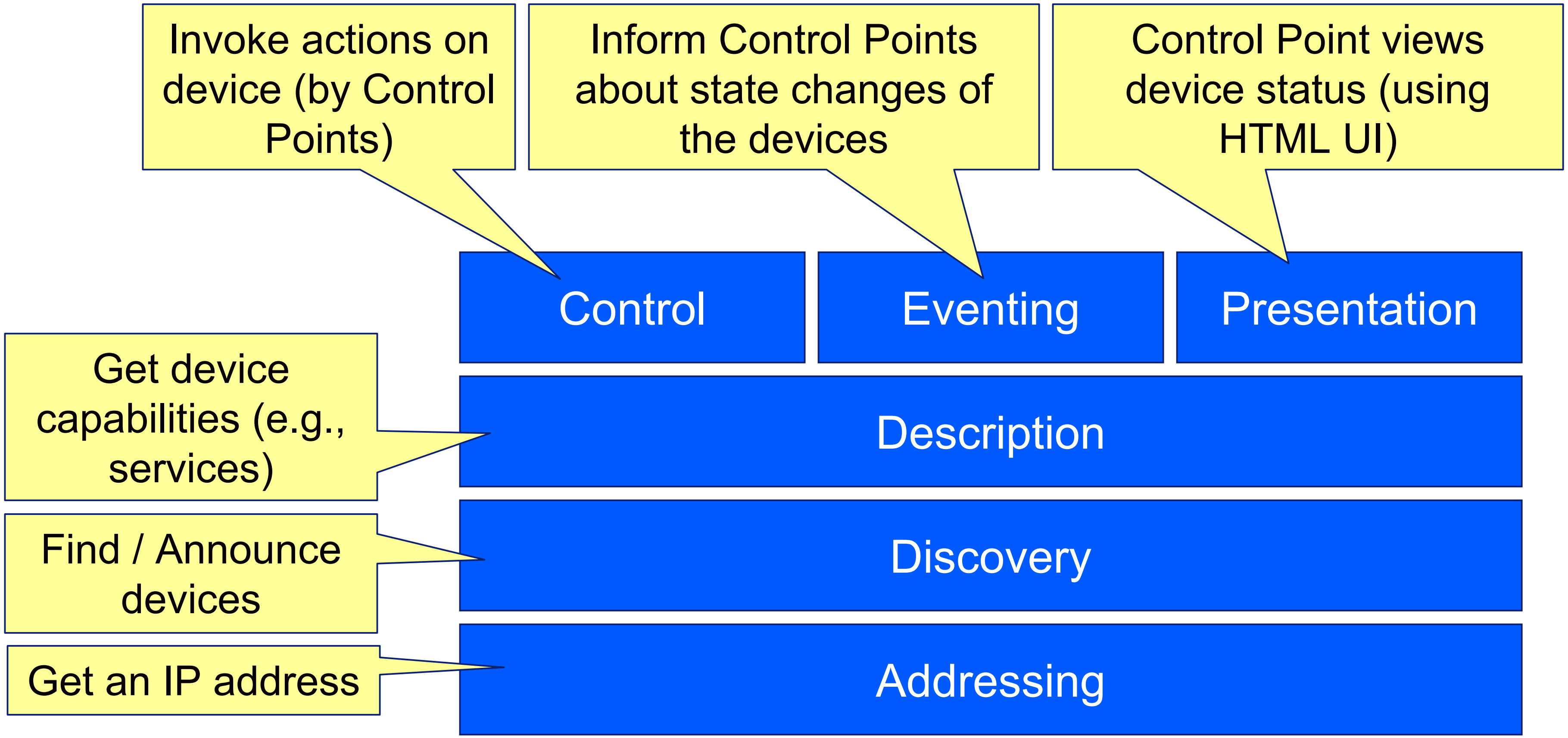
- e.g. 80% of all internet gateways (ADSL/cable) modems sold today have UPnP

UPnP Basics



UPnP Control Point
Defined interface
UPnP Device
UPnP Service

UPnP Device Architecture



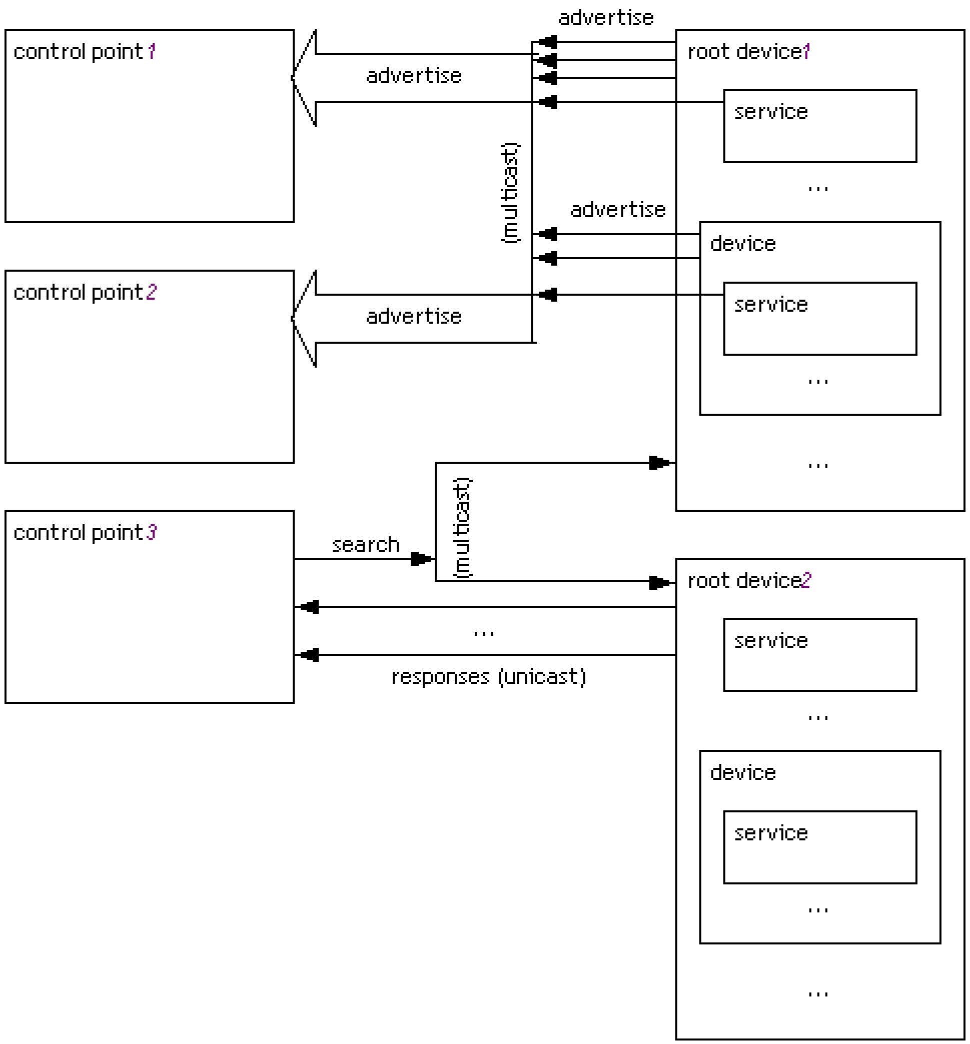
UPnP Protocol Layers

<table>
<tr><td colspan="3">UPnP vendor [purple-italic]</td></tr>
<tr><td colspan="3">UPnP Forum [red-italic]</td></tr>
<tr><td colspan="3">UPnP Device Architecture [green-bold]</td></tr>
<tr><td rowspan="2">SSDP [blue]</td><td>SOAP [blue]</td><td>GENA [navy-bold]</td></tr>
<tr><td>HTTP [black]</td><td>HTTP [black]</td></tr>
<tr><td>UDP [black]</td><td colspan="2">TCP [black]</td></tr>
<tr><td colspan="3">IP [black]</td></tr>
</table>

Discovery

Devices advertise themselves and their services such that control points know about their existence

Control Points can also search for specific devices or services



PHILIPS

Device Description

An XML document describes the device

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <URLBase>base URL for all relative URLs</URLBase>
  <device>
    <deviceType>urn:schemas-upnp-org:device:deviceType:v</deviceType>
    <friendlyName>short user-friendly title</friendlyName>
    <manufacturer>manufacturer name</manufacturer>
    <manufacturerURL>URL to manufacturer site</manufacturerURL>
    <modelDescription>long user-friendly title</modelDescription>
    <modelName>model name</modelName>
    <modelNumber>model number</modelNumber>
    <modelURL>URL to model site</modelURL>
    <serialNumber>manufacturer's serial number</serialNumber>
    <UDN>uuid:UUID</UDN>
    <UPC>Universal Product Code</UPC>
    <iconList>
      <icon>
        <mimetype>image/format</mimetype>
        <width>horizontal pixels</width>
        <height>vertical pixels</height>
        <depth>color depth</depth>
        <url>URL to icon</url>
      </icon>
      XML to declare other icons, if any, go here
    </iconList>
    <serviceList>
      <service>
        <serviceType>urn:schemas-upnp-org:service:serviceType:v</serviceType>
        <serviceId>urn:upnp-org:serviceId:serviceID</serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
      </service>
      Declarations for other services defined by a UPnP Forum working committee (if any)
        go here
      Declarations for other services added by UPnP vendor (if any) go here
    </serviceList>
    <deviceList>
      Description of embedded devices defined by a UPnP Forum working committee (if any)
        go here
      Description of embedded devices added by UPnP vendor (if any) go here
    </deviceList>
    <presentationURL>URL for presentation</presentationURL>
  </device>
</root>
```

PHILIPS

Service Description

An XML document describes the service

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <actionList>
    <action>
      <name>actionName</name>
      <argumentList>
        <argument>
          <name>formalParameterName</name>
          <direction>in xor out</direction>
          <retval />
          <relatedStateVariable>stateVariableName</relatedStateVariable>
        </argument>
      Declarations for other arguments defined by UPnP Forum working committee (if any)
         go here
      </argumentList>
    </action>
    Declarations for other actions defined by UPnP Forum working committee (if any)
      go here
    Declarations for other actions added by UPnP vendor (if any) go here
  </actionList>
  <serviceStateTable>
    <stateVariable sendEvents="yes">
      <name>variableName</name>
      <dataType>variable data type</dataType>
      <defaultValue>default value</defaultValue>
      <allowedValueList>
        <allowedValue>enumerated value</allowedValue>
        Other allowed values defined by UPnP Forum working committee (if any) go here
      </allowedValueList>
    </stateVariable>
    <stateVariable sendEvents="yes">
      <name>variableName</name>
      <dataType>variable data type</dataType>
      <defaultValue>default value</defaultValue>
      <allowedValueRange>
        <minimum>minimum value</minimum>
        <maximum>maximum value</maximum>
        <step>increment value</step>
      </allowedValueRange>
    </stateVariable>
    Declarations for other state variables defined by UPnP Forum working committee
      (if any) go here
    Declarations for other state variables added by UPnP vendor (if any) go here
  </serviceStateTable>
</scpd>
```

Control: invocation of an action on a service

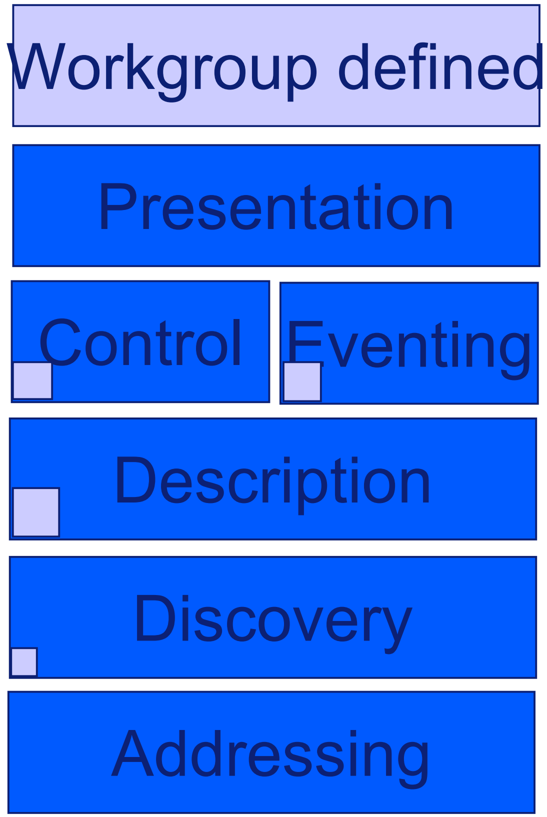
```
POST path control URL HTTP/1.0
HOST: hostname:portNumber
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"
USER-AGENT: OS/version UPnP/1.0 product/version
SOAPACTION: "urn:schemas-upnp-org:service:serviceType:v#actionName"

<?xml version="1.0"?>
<s:Envelope
    xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
    s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:actionName xmlns:u="urn:schemas-upnp-org:service:serviceType:v">
      <argumentName>in arg value</argumentName>
      other in args and their values go here, if any
    </u:actionName>
  </s:Body>
</s:Envelope>
```

Within UPnP, the behavior and signature of actions of services for devices are specified. Control point behavior is not specified

Device architecture vs. workgroups

UPnP is a multi-stage standard:

- Architecture is framework and template
- Specifics filled in by workgroups of the UPnP Forum
  - Internet Gateway,
  - Printing & Imaging,
  - Audio Video,
  - Security,
  - QoS,
  - Remote UI,
  - EEDS,
  - Low Power, Remote Access, …
  - …



UPnP-AV

UPnP-AV defines control of AV functionality. It defines two UPnP devices: A MediaServer (MS) and a MediaRenderer (MR) and an AV Control Point (AVCP). There are four important services that run on these devices

- Search and Browse content
- Play, Stop, … content
- Change volume, color, …

- ContentDirectoryService
- AVTransport
- RenderingControl
- ConnectionManager

PHILIPS

But UPnP-AV does not…

- Define transport protocols.
  - HTTP,
  - RTP-UDP-IP
  - IEC61883 (1394)
  - Your own protocol
- Select codecs
  - MPEG 2
  - MPEG 4
  - DivX
  - Etc.

So UPnP-AV restricts itself to the management of the streaming but the streaming as such is out of its control or as it is commonly called "out of band"

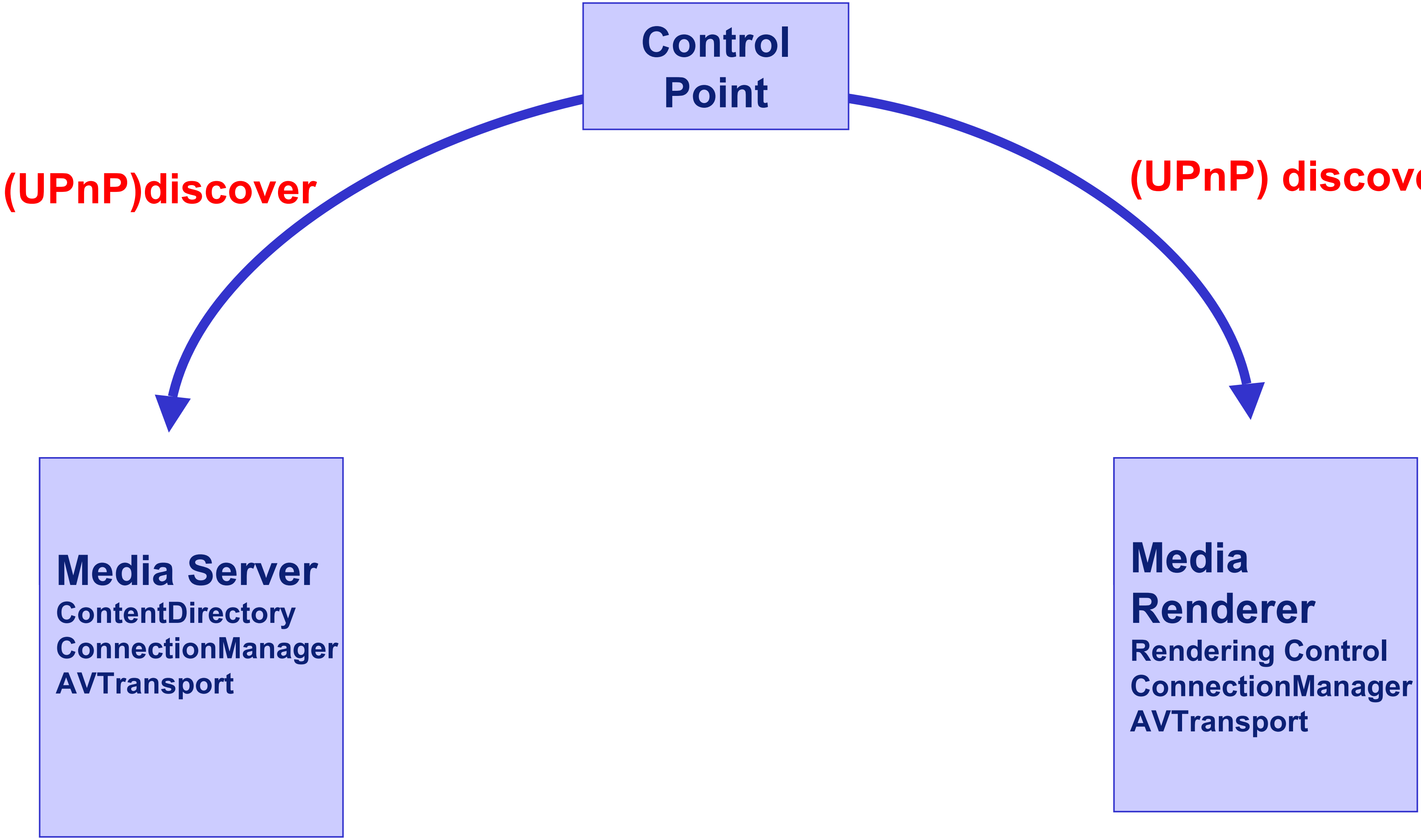
An interesting Use Case: Remote playback

Control Point

Media Server
ContentDirectory
ConnectionManager
AVTransport

Media Renderer
Rendering Control
ConnectionManager
AVTransport

UPnP Discovery



Browse/Search

select content

- **browse** or search
- **(get URL to content)**

- includes formats

Control Point

Media Server

ContentDirectory

ConnectionManager

AVTransport

```
response: Browse(
"<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
      xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/">
      <item id="6" parentID="3" restricted="false">
            <dc:title>Chloe Dancer</dc:title>
            <dc:creator>Mother Love Bone</dc:creator>
            <upnp:class>object.item.audioItem.musicTrack</upnp:class>
            <res protocolInfo="http-get:*:audio/x-ms-wma:*"
size="200000">
                  http://10.0.0.1/getcontent.asp?id=6
            </res>
      </item>
      <item id="8" parentID="3" restricted="false">
            <dc:title>Drown</dc:title>
            <dc:creator>Smashing Pumpkins</dc:creator>
            <upnp:class>object.item.audioItem.musicTrack</upnp:class>
            <res protocolInfo="http-get:*:audio/mpeg:*" size="140000">
                  http://10.0.0.1/getcontent.asp?id=8
            </res>
      </item>
      <item id="7" parentID="3" restricted="false">
            <dc:title>State Of Love And Trust</dc:title>
            <dc:creator>Pearl Jam</dc:creator>
            <upnp:class>object.item.audioItem.musicTrack</upnp:class>
            <res protocolInfo="http-get:*:audio/x-ms-wma:*"
size="70000">
                  http://10.0.0.1/getcontent.asp?id=7
```

(fragment)

Media Renderer

Rendering Control

ConnectionManager

AVTransport

PHILIPS

Get supported protocols/formats

getProtocolInfo()
- **Get transfer protocols**
- **Get data formats**

Control Point

getProtocolInfo()
- **Get transfer protocols**
- **Get data formats**

In the context of this document, the term "protocol info" is used to describe as a string formatted as:

<protocol>':' <network>':'<contentFormat>':'<additionalInfo>

where each of the 4 elements may be a '*'. Control points can match protocol info by (protocol independent) string comparison operations on the <protocol>, <network> and <contentFormat> elements, taking into account the "*" wildcard which 'matches' with anything. The <additionalInfo> part does not need to match between source and sink. Its purpose is to convey any additional information needed to set up the out of band stream (e.g., 1394 addresses). The table below summarizes how the protocol info strings are defined for the protocols currently standardized by the ConnectionManager service, as well as for vendor-defined protocols. Section 5 provides a more detailed explanation per protocol.

Protocol	Network	Content Format	Additional Info	Reference
`http-get`	Not needed (use '*'), since all devices supporting http are part of the same IP network.	MIME-type.	Not needed, use "*".	Section 5.1.1
`rtsp-rtp-udp`	Not needed (use '*'), since all devices supporting rtsp are part of the same IP network.	Name of RTP payload type.	Not needed, use "*".	Section 5.1.2
`internal`	IP address of the device hosting the ConnectionManager.	Vendor-defined, may be '*'.	Vendor-defined, may be '*'.	Section 5.1.3
`iec61883`	GUID of the 1394 bus' Isochronous Resource Manager.	Name standardized by IEC61883.	GUID and PCR index of the 1394 device.	Section 5.1.4
`«registered ICANN domain name of vendor»`	Vendor-defined, may be '*'.	Vendor-defined, may be '*'.	Vendor-defined, may be '*'.	Section 5.1.5

Media Server
ContentDirectory
ConnectionManager
AVTransport

Media Renderer
Rendering Control
ConnectionManager
AVTransport

Prepare for connection

prepareForConnection()
- **specify protocol+format**
- **returns ConnManagerID**
- **returns AVTransport Instance ID***

Control Point

prepareForConnection()
- **specify protocol+format**
- **returns ConnManagerID**
- **returns AVTransport Instance ID***
- **returns RenderControl ID**

2.4.2. PrepareForConnection

This action is used to allow the device to prepare itself to connect to the network for the purposes of sending or receiving media content (e.g. a video stream). The RemoteProtocolInfo parameter identifies the protocol, network, and format that should be used to transfer the content. Its value corresponds to one of the ProtocolInfo entries returned by the GetProtocolInfo() action from the remote device. If the remote device does not implement GetProtocolInfo(), then the RemoteProtocolInfo parameter should be set to one of the ProtocolInfo entries returned by the GetProtocolInfo() action on the local device.

2.4.2.1. Arguments

Table 5: Arguments for PrepareForConnection

Argument	Direction	relatedStateVariable
RemoteProtocolInfo	IN	A_ARG_TYPE_ProtocolInfo
PeerConnectionManager	IN	A_ARG_TYPE_ConnectionManager
PeerConnectionID	IN	A_ARG_TYPE_ConnectionID
Direction	IN	A_ARG_TYPE_Direction
ConnectionID	OUT	A_ARG_TYPE_ConnectionID
AVTransportID	OUT	A_ARG_TYPE_AVTransportID
RcsID	OUT	A_ARG_TYPE_RcsID

Media Server
ContentDirectory
ConnectionManager
AVTransport

Media Renderer
Rendering Control
ConnectionManager
AVTransport

* At least one of them returns the id

Set&start content with Transport ID

Control Point

SetTransPortURI()
Play()

Media Server
ContentDirectory
ConnectionManager
AVTransport

Table 4: Arguments for *SetAVTransportURI*

Argument	Direction	relatedStateVariable
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
CurrentURI	*IN*	*AVTransportURI*
CurrentURIMetaData	*IN*	*AVTransportURIMetaData*

Table 12: Arguments for *Play*

Argument	Direction	relatedStateVariable
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
Speed	*IN*	*TransportPlaySpeed*

Media Renderer
Rendering Control
ConnectionManager
AVTransport

Remote playback; rendering control

Control Point

(optional)
Rendering controls
(e.g. volume up/down)

2.4.29. GetVolume

This action retrieves the current value of the Volume state variable of the specified channel for the specified instance of this service. The CurrentVolume (OUT) parameter contains a value ranging from 0 to a device-specific maximum. See Section 2.2.16 (Volume) for more details,

2.4.29.1. Arguments

Argument(s)	Direction	relatedStateVariable
InstanceID	IN	A_ARG_TYPE_InstanceID
Channel	IN	A_ARG_TYPE_Channel
CurrentVolume	OUT	Volume

Media Server
ContentDirectory
ConnectionManager
AVTransport

(out-of-band) **streaming**

Media Renderer
Rendering Control
ConnectionManager
AVTransport

2.4.30. SetVolume

This action sets the Volume state variable of the specified Instance and Channel to the specified value. The DesiredVolume input parameter contains a value ranging from 0 to a device-specific maximum. See Section 2.2.16 (Volume) for more details,

2.4.30.1. Arguments

Argument(s)	Direction	relatedStateVariable
InstanceID	IN	A_ARG_TYPE_InstanceID
Channel	IN	A_ARG_TYPE_Channel
DesiredVolume	IN	Volume

UPnP-AV is flexible but this is also one of its limitations

One can use all kind of (vendor-specific) protocols and media formats but in that case interoperability is not guaranteed:

Vendor	X1	X2
Protocol	Y1	Y2
Format	Z1	Z2

Interoperable if and only if Y1=Y2 and Z1=Z2

Practice: protocol = HTTP-GET, format = MPEG

Recapitulation

Multiple network standards
CE-requirements - just play
3 standards: HAVi, UPnP, JINI
Function discovery – plug-in discovery
Client-server look and feel
Internet protocol, zeroconf, mDNS, SLP
UPnP basics, A/V connection, play-back