Telecooperation Lab
Prof. Dr. Max Mühlhäuser

TK3: Ubiquitous Computing

Chapter 2: Infrastructure

Part 3: Service Discovery

Lecturer: Dr. Immanuel Schweizer

Motivation

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Find service by function & properties**
  - Idea: in UbiComp, „Smart Devices“ should easily „cooperate“ … and therefore,
  - Find/understand each other first!
  - Should not require administrator for configuration
  - Simple example: Mobile phone -> „unknown office“ -> use local printer
- Comparison with classical phonebook
  - White Pages (Naming)
    - Suppose your BMW bought at „BMW Darmstadt“ needs service, so you look on the **white pages** to obtain the phone number
  - Yellow Pages (Service Lookup / Trading)
    - Suppose a pipe were to burst in the evening in your home and you don‘t know anybody who could fix that
    - You‘d look on the **yellow pages** under „plumbers“
      - You‘d narrow your choice to plumbers in the same city
      - You‘d look at who offers a 24-hour emergency service
      - You‘d then choose the first one from the remaining list

Motivation

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Find service by function & properties**
  - Idea: in UbiComp, „Smart Devices“ should easily „cooperate“ … and therefore,
  - Find/understand each other first!
  - Should not require administrator for configuration
  - Simple example: PDA -> „unknown office“ -> use local printer
- Comparison with classical phonebook
  - White Pages (Naming)
    - Suppose your BMW bought at „BMW Darmstadt“ needs service, so you look on the **white pages** to obtain the **phone number**
  - Yellow Pages (Service Lookup / Trading)
    - Suppose a pipe were to burst in the evening in your home and you don‘t know anybody who could fix that
    - You‘d look on the **yellow pages** under „**plumbers**“
      - You‘d narrow your choice to plumbers in the same city
      - You‘d look at who offers a 24-hour emergency service
      - You‘d then choose the first one from the remaining list

How does this map to technical services?

→ Network address or URL

→ Service function

} Service properties

Architecture

TECHNISCHE UNIVERSITÄT DARMSTADT

- General architecture of the systems discussed in this chapter
  - Trading/Service Discovery/Lookup
    - Allows to **find service by function and properties**
    - Service provider registers service (identified by URL or similar) using **service description**
    - Service client finds service (URL) using **service query**
    - In many cases, builds on Naming
  - Naming
    - Provides address (URL or IP address) of resource (service or peer)
  - Addressing
    - We consider software services in a distributed system
    - Services run on different devices
    - How do the devices get network addresses in the first place?
      - -> DHCP, AutoIP

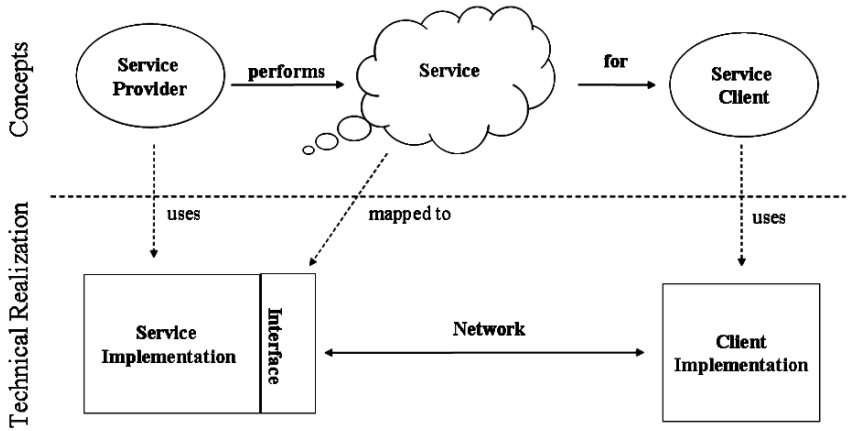
Trading

Naming

Addressing

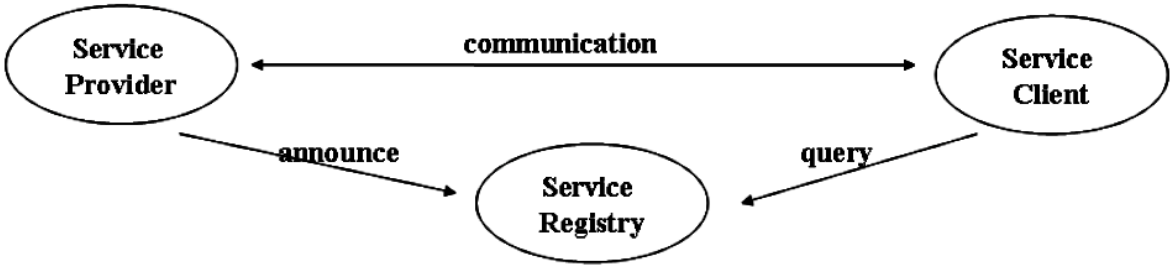
Service Oriented Architecture (SOA): Terminology

- **Service**: Collection of abilities (conceptual level)
- **Service Provider**: Provides one or more services
- **Service Client**: Uses services
- **Service Implementation**: Technical realization of the service
- **Client Implementation**: Technical realization of service user
- **Interface:** Mapping of service abilities to concrete technology or prog. language
- Network is used for comm. btw. provider and client



Finding Services

- Roles in SOA
  - Service Client and Provider may not know each other at runtime
  - Provider announces its services at a **Service Registry**
    - Sometimes also called: service repository, service broker, service mediator
  - Client queries the registry to find required services
- Finding Services
  - **Service Lookup**
    - (Central) service registry is used as lookup server
  - **Service Discovery**
    - Completely spontaneous method without (central) servers
  - Sometimes used in combination
    1. Discovery used to find registry (service)
    2. Lookup used to find target service



Service Registry: Information Storage

TECHNISCHE UNIVERSITÄT DARMSTADT

- Central Registry
  - A central repository stores all information
    - Either „single server“ or federated (only „logically centralized“)
  - Single Server
    - Central = for an organizational or technical unit
  - Federated Registry
    - All servers provide the same, global view
    - Uses caching and replication for robustness and scalability
    - UDDI: globally central -> failed!
- Distributed Registry
  - Extreme case: registry runs on each node
  - Registry only holds local information
    - Information about local services
    - May cache information about frequently used remote services
- No Registry
  - Solely relies on specific network protocols
  - Services announce their presence periodically (broadcast)

Service Registry: Interaction

TECHNISCHE UNIVERSITÄT DARMSTADT

- Service Provider
  - Announces its services providing **service descriptions**
  - Registration contains an „expiration date“ - called **lease**
    - Allows the registry to remove stale entries
  - Registration is renewed periodically
    - e.g., renewal interval = expiration date / 2.5
    - Depends on expected packet loss in network

- Service Consumer
  - **Query/Pull**
    - Clients query for services
  - **Notification/Push**
    - Clients can register and are notified when an appropriate service appears

Service Description

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Key/value pairs**
  - Service properties described as set of key/value pairs
  - Strict value comparison
    - Set of key/value pairs also specified with query
    - Pairs present in query set must exactly match
    - Wildcard allows to test presence of key; value does not matter
  - Query-Language
    - Allows specification of more operators (>, <, ...)
    - Allows more complex boolean expressions (OR, AND, NOT)
- **Markup-based description** (template-based description)
  - Not unlike key/value pairs, but allows nesting (and duplicate keys)
  - Schema strictly defines the structure of description documents
  - e.g., XML
- **Semantic description**
  - Ontologies are used for service description
  - Query described in (e.g.) SPARQL
  - Inference engine determines matching services

System Examples

TECHNISCHE UNIVERSITÄT DARMSTADT

- LAN/PAN scope
  - IP-based protocols
    - **UPnP / SSDP**
    - Bonjour / mDNS
  - Low level
    - Bluetooth SDP
  - Middleware
    - Jini
- Global scope
  - UDDI

UPnP

TECHNISCHE UNIVERSITÄT DARMSTADT

- UPnP: **U**niversal **P**lug a**n**d **P**lay
  - Originates from Microsoft
  - Standards by UPnP forum (since 1999), >800 members

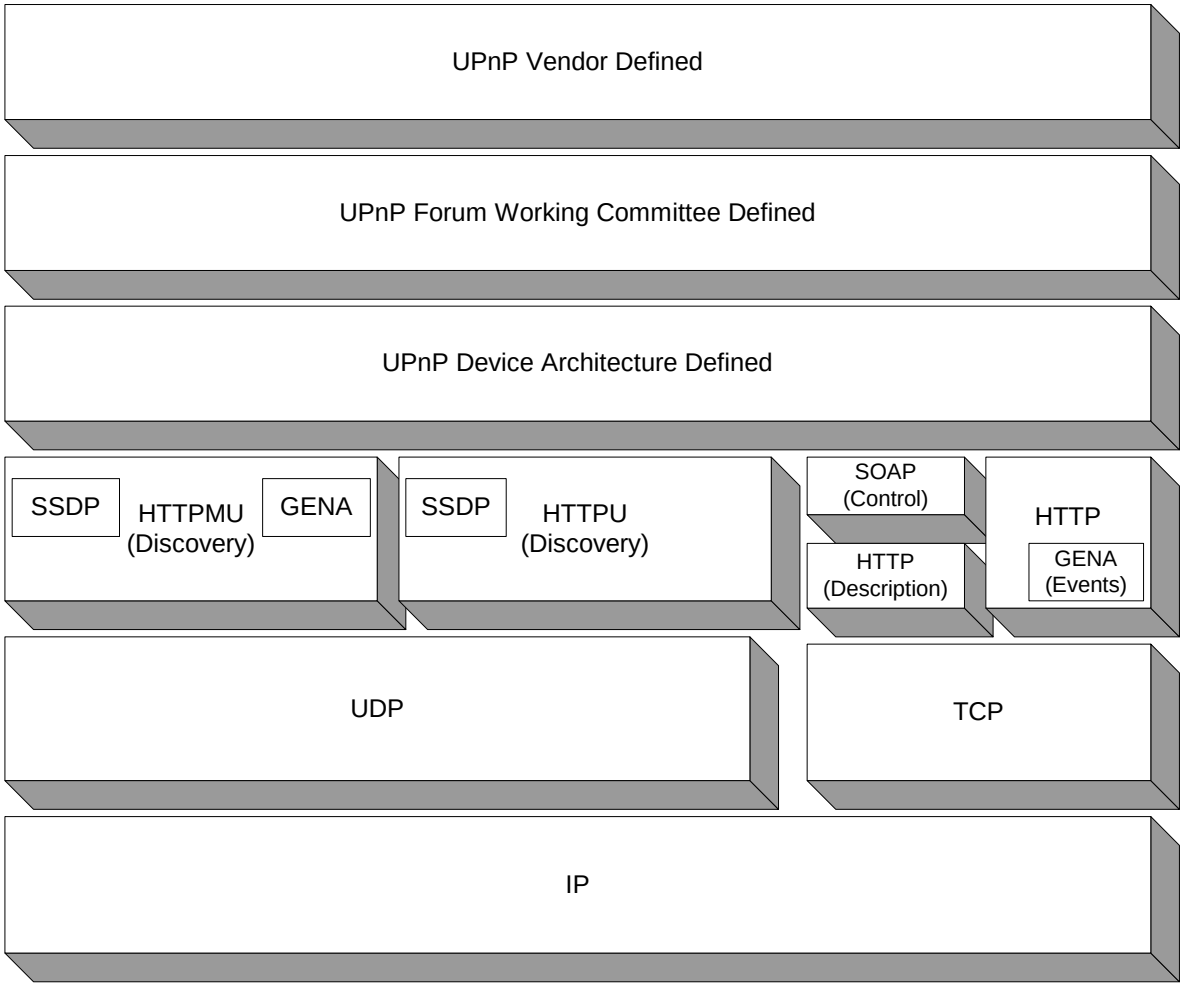
- Builds on Internet standards and defines:
  - **Addressing**: Assigning IP addresses to devices (AutoIP)
  - **Discovery**: Announcing the existence of devices and services and finding appropriate services (SSDP)
  - **Description**: Describing the capabilities of devices and services (HTTP)
  - **Controlling** (SOAP)
  - **Eventing**: Getting notifications about the state changes of services (GENA)
  - **Presentation**: Enabling human users to control the service through a web page (HTTP)

UPnP Protocols

TECHNISCHE UNIVERSITÄT DARMSTADT

- Protocols
  - UDP, TCP/IP, HTTP, XML
  - Simple Service Discovery Protocol (SSDP)
  - Generic Event Notification Architecture (GENA)
    - Send/receive event notifications using HTTP over TCP/IP and multicast UDP
  - Simple Object Access Protocol (SOAP)
    - XML and HTTP for remote procedure calls

UPnP Protocol Stack



AutoIP

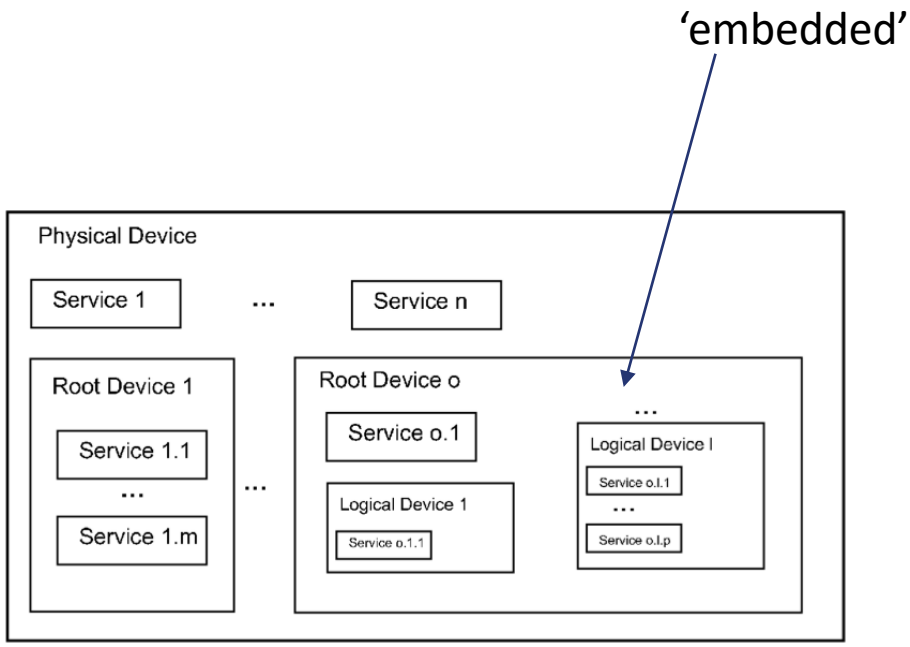
- IP address assignment
  - Try to find DHCP server; if no success...
  - Pick a random address from a range of local addresses (169.254.x.x)
  - Address Resolution Protocol (ARP): Verify address is not in use
    - If in use, retry until some max attempts exceeded
  - Configure and use address
  - Periodically try to find DHCP server by sending discover query
    - If DHCP offer is received, use it
- These addresses are not routable!!

"Nice to be a part of this network.
Hello? DHCP? Nope.
169.254.1.5 might work.
Anybody using 169.254.1.5?
Oops. Sorry!
Anybody using 169.254.1.37?
OK. 169.254.1.37 it is.

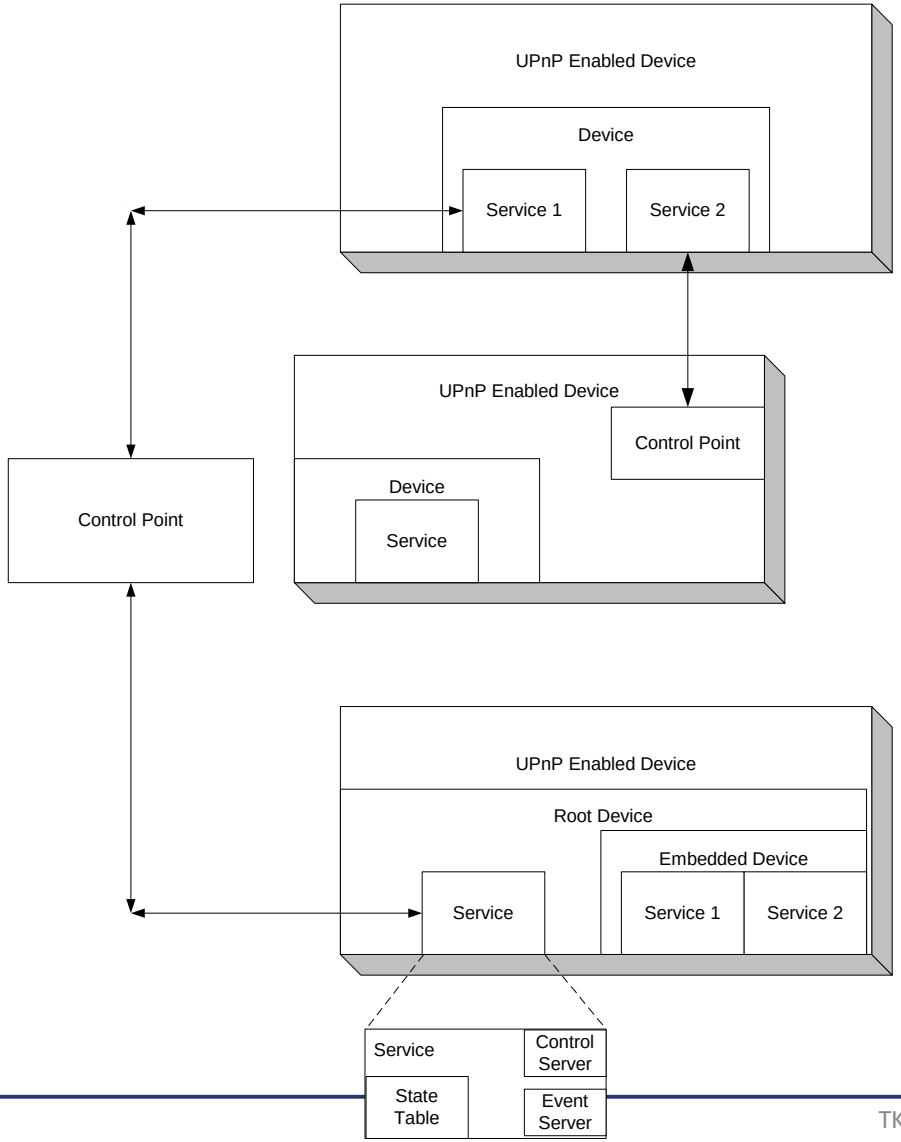
I'll retry a DHCP request in 5 minutes…"

UPnP: Architecture

- Controlled Device (= Service Provider)
  - Contains several services
  - May contain further logical devices (1-n ‘root devices’, with 0-m ‘embedded devices’ each)
- Control Point (= Service Client)
  - Retrieve the device description and get a list of associated services.
  - Retrieve service descriptions for interesting services.
  - Invoke actions to control the service.
  - Subscribe to the service’s event source. Anytime the state of the service changes, the event server will send an event to the control point.

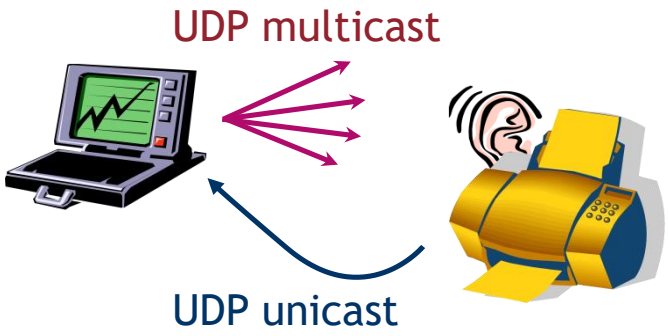


UPnP: Architecture

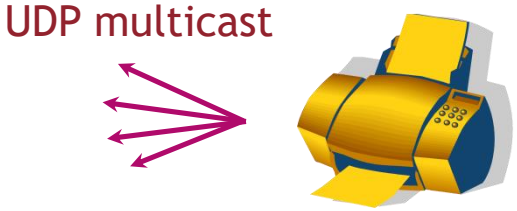


SSDP

- Simple Service Discovery Protocol
- Based on “HTTP over UDP-multicast”
  - Local admin. scope → multicast address 239.255.255.250 (see RFC 2365)
- Supports both discovery and advertisement

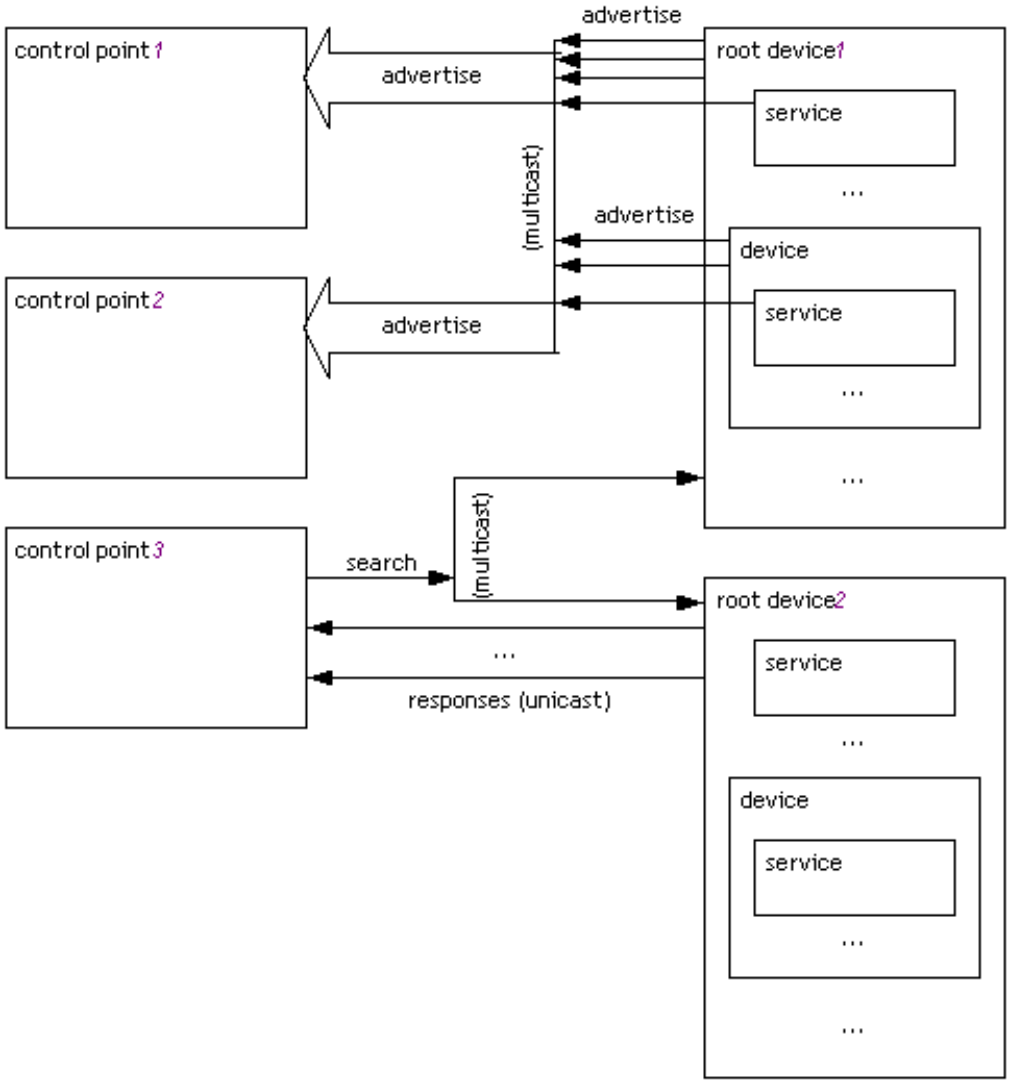


discovery: "I need a printer"
response: "I'm a printer.
see: http://171.3.7.5/description.xml



advertisement: „anyone need a printer?“
see: http://171.3.7.5/description.xml
plan to be around for at least 10 minutes…"

SSDP: Notify


control point 1
control point 2
control point 3
advertise
advertise
(multicast)
advertise
advertise
root device 1
service
...
device
service
...
...
search
(multicast)
root device 2
...
responses (unicast)
service
...
device
service
...
...

SSDP: NOTIFY

- Periodically sent by device as advertisement
- **NOTIFY** message
  - *LOCATION:* URL of root device description
  - *Notification Type (NT):* root device, ..., service
  - *Notification SubType (NTS):* ssdp:alive or ssdp:byebye
  - *Unique Service Name (USN)*
    - allows unique identification of device (e.g., after IP address change)
  - *CACHE-CONTROL: lease time*
- Advertisements do not contain detailed service descriptions

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
SERVER: Linux/2.6.15.2 UPnP/1.0 Mediaserver/1.0
CACHE-CONTROL: max-age=1800
LOCATION: http://192.168.0.10:8080/description.xml
NTS: ssdp:alive
NT: urn:schemas-upnp-org:service:ConnectionManager:1
USN: uuid:550e8400-e29b-11d4-a716-446655440000::urn:schemas-upnp-org:service:ConnectionManager:1
```

SSDP: M-SEARCH

TECHNISCHE UNIVERSITÄT DARMSTADT

- Sent from control point to device
- M-SEARCH message
  - MAN (message type): must be „ssdp:discover“
  - MX (maximum waiting time until reply): time in seconds
    - Device supposed to wait random time in [0 … MX] to avoid congestion
  - Search Target (ST): must exactly match Notification Type (NT)
- Device replies if ST==NT
  - Device sends NOTIFY using UDP unicast to host
  - Host address = source address of M-SEARCH
- Model is trivial
  - Host will have to download many descriptions and do local matching

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
ST: urn:schemas-upnp-org:service:ConnectionManager:1
MX: 3
```

UPnP: Device Description

TECHNISCHE UNIVERSITÄT DARMSTADT

```
<?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 URL for eventing</eventSubURL>
      </service>
    </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>
```

UPnP: Using a Service

TECHNISCHE UNIVERSITÄT DARMSTADT

- Controlling
  - Call functions using „Simple Object Access Protocol“ (SOAP)
- Eventing: General Event Notification Architecture (GENA)
  - Uses HTTP over TCP or multicast UDP
  - Requests: SUBSCRIBE, UNSUBSCRIBE, NOTIFY
  - Uses topic-based addressing

```
SUBSCRIBE publisher path HTTP/1.1
HOST: publisher host:publisher port
CALLBACK: <delivery URL>
NT: upnp:event
TIMEOUT: subscription duration
```

- Presentation
  - Device description points to URL of HTML UI for controlling the device

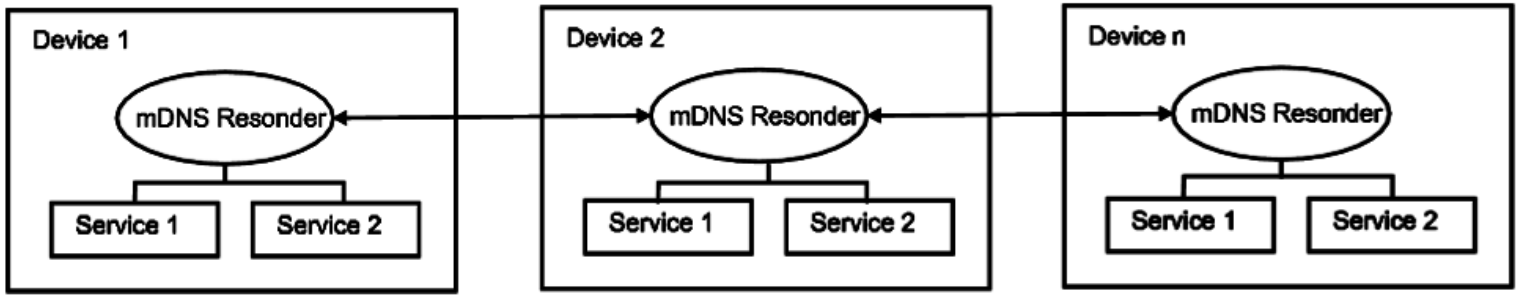
System Examples

TECHNISCHE UNIVERSITÄT DARMSTADT

- LAN/PAN scope
  - IP-based protocols
    - UPnP / SSDP
    - **Bonjour / mDNS**
  - Low level
    - Bluetooth SDP
  - Middleware
    - Jini
- Global scope
  - UDDI

Bonjour

- Builds on Multicast DNS (mDNS)
  - Service description packed into DNS records (Key/Value-Model)
  - Service discovery -> DNS query over IP Multicast
  - solely builds on DNS protocol - does not define own protocols
- Architecture
  - Addressing: AutoIP (Zeroconf)
  - Naming: mapping unique names to IP addresses
    - Bonjour requires devices to have unique names
    - Autoconfiguration of hostname using mDNS, similar to AutoIP
  - Service discovery: finding appropriate services …



Bonjour

TECHNISCHE UNIVERSITÄT DARMSTADT

- Service Records (SRV) describe service instances

```
PrintsALot._printer._tcp.local 120 IN SRV 0 0 515 blackhawk.local
```

Instance
Service
Domain
Time-to-Live (TTL)
Internet
Service Record
Priority
Weight
Port
Target

- Pointer Resource Records (PTR) map from service type to instance

```
_printer._tcp.local. 28800 IN PTR PrintsALot._printer._tcp
```

Service Type
Domain
TTL
Service Instance Name

- Text Records (TXT) store information about a service

```
PrintsALot._printer._tcp.local 120 IN TXT "Color=T" "Duplex=F" ...
```

Bonjour

TECHNISCHE UNIVERSITÄT DARMSTADT

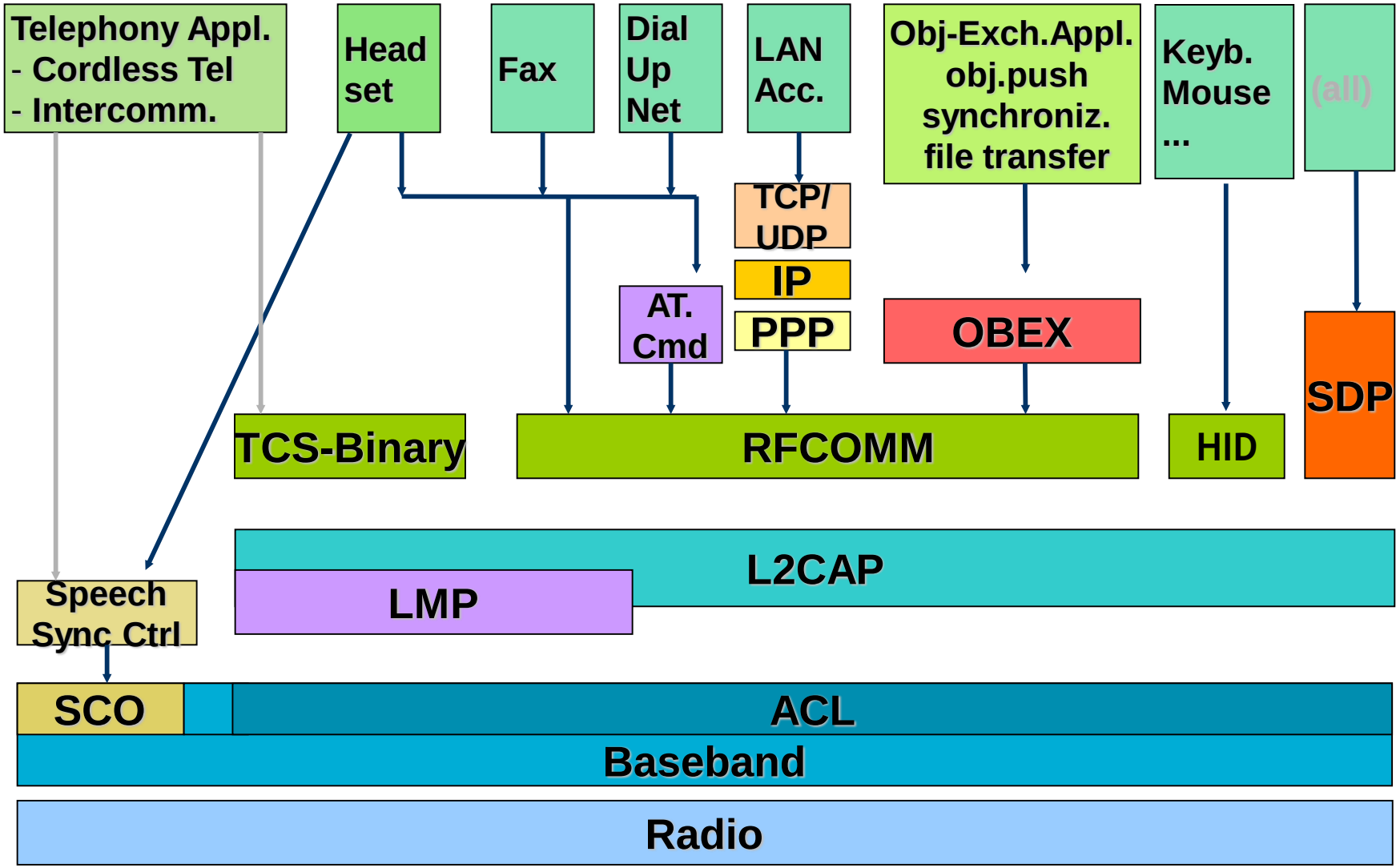
- Service Discovery
  - e.g., multicast query for „_printer._tcp“
  - Any mDNS Responder in the network will answer
  - Result is every printer service running on any device in the local network

System Examples

TECHNISCHE UNIVERSITÄT DARMSTADT

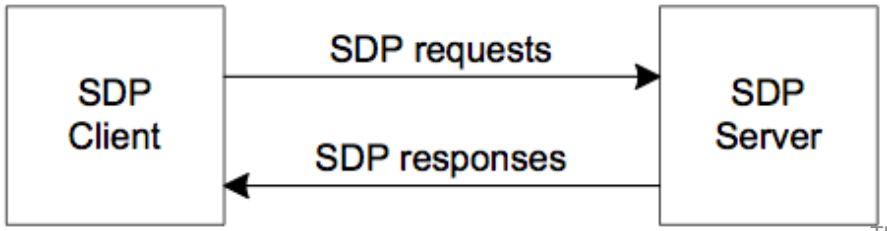
- LAN/PAN scope
  - IP-based protocols
    - UPnP / SSDP
    - Bonjour / mDNS
  - Low level
    - **Bluetooth SDP**
  - Middleware
    - Jini
- Global scope
  - UDDI

Bluetooth Protocol Stack



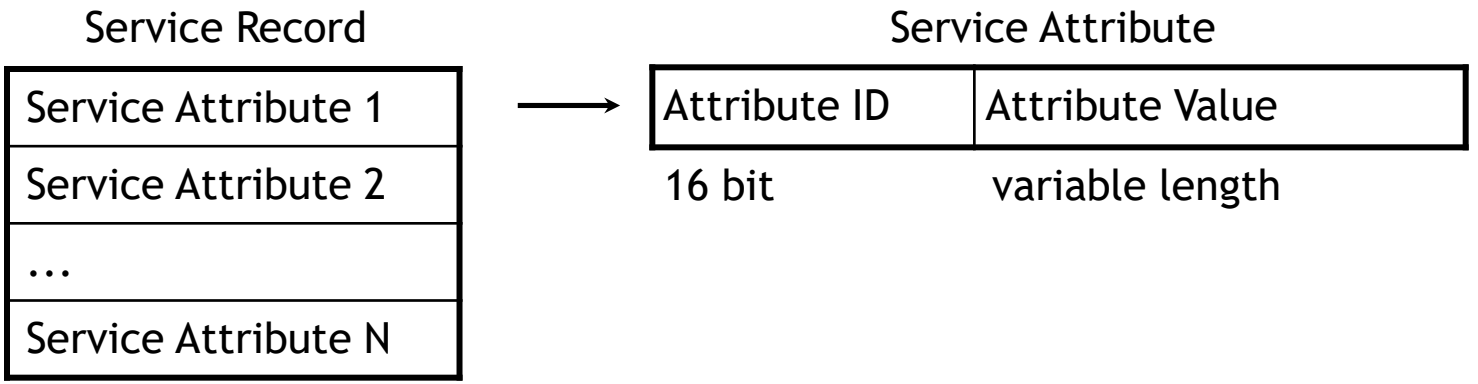
Bluetooth SDP

- Architecture
  - Works on a relatively low level (L2CAP)
  - Can determine that services are (un)available based on RF proximity
- Registry
  - Uses decentralized registry
  - Each Bluetooth device has an SDP server (for local services)
- Service Description
  - Based on Key/Value-Model
- Service Discovery (comes after *device* discovery in Piconet: 3bit adr. assigned)
  - Allows clients to search for services based on specific attributes
  - Allows clients to browse all available services
  - Only query, no notifications!



Bluetooth: Service Description

- Service Record
  - Stores information about a service (Key/Value-pairs)
  - Each record is identified by a 32-bit handle, which is SDP Server specific
- Service Attributes (selection):
  - *ServiceID*: uniquely identifies a specific instance of a service (UUID, 128 bit)
  - *ProviderName*: organization that provides service
  - *ServiceName*: human readable name
  - *ServiceDescription*: human readable description
  - *ProtocolDescriptorList*: protocols for accessing the service
  - *ServiceClassIDList*: see next slide



Bluetooth: Service Description

TECHNISCHE UNIVERSITÄT DARMSTADT

- ServiceClassIDList
  - List of classes of which the service is an instance
  - The meaning of other attributes depends on the service classes
  - Attribute IDs are only guaranteed to be unique within a service class

Service Attribute

ServiceClassIDList	DuplexColorPostscriptPrinterServiceClassID ColorPostscriptPrinterServiceClassID PostscriptPrinterServiceClassID PrinterServiceClassID
16 bit	UUIDs (128-bit each)

Bluetooth: Discovery

TECHNISCHE UNIVERSITÄT DARMSTADT

- **General process:** 3 phases prior to Service discovery (reality: complicated details!)
  1. Inquiry (device discovery in inquiry mode → page mode → …)
  2. Name discovery (user friendly device name)
  3. Pairing (establishment of physical connection)
  4. → SDP Queries
- **Service search** transaction
  - Service search pattern contains a list of UUIDs
    - Can search only for attributes whose values are UUIDs
      - Service ID, service class ID, transport protocol, …; wildcard: PUBLIC_BROWSE_GROUP
    - Service search pattern matches a service record if each and every UUID in the service search pattern is contained within any of the service record's attribute values
    - Search based on the values of arbitrary attributes is not provided
  - Result: list of record handles
    - Used to access further (e.g., non-UUID) attributes
- **Service browsing**
  - UUID of „root browse group" is specified in search transaction

System Examples

TECHNISCHE UNIVERSITÄT DARMSTADT

- LAN/PAN scope
  - IP-based protocols
    - UPnP / SSDP
    - Bonjour / mDNS
  - Low level
    - Bluetooth SDP
  - Middleware
    - **Jini**
- Global scope
  - UDDI

Jini: Introduction

TECHNISCHE UNIVERSITÄT DARMSTADT

- What‘s Jini?
  - Java-based middleware
  - Every device represented by proxy object implementing Java interfaces
- Goals:
  - Allow access to resources despite mobility
  - Simplify configuration and maintenance of large groups of users, devices, and software
- History:
  - Started at Sun (1998), then disappeared for years
  - Now continued in open source project **Apache River** (last news/release from 2013)
- Terms:
  - A distributed Jini system is called a **djinn**
  - Entities can provide or obtain services from a djinn
  - **Code in the djinn is portable and mobile**
- Underlying technology:
  - Object serialization: can marshal/unmarshal arbitrary objects
  - RMI: Remote Method Invocation (essentially, „RPC for Java objects“)
  - Mobile code (RMI code downloading)

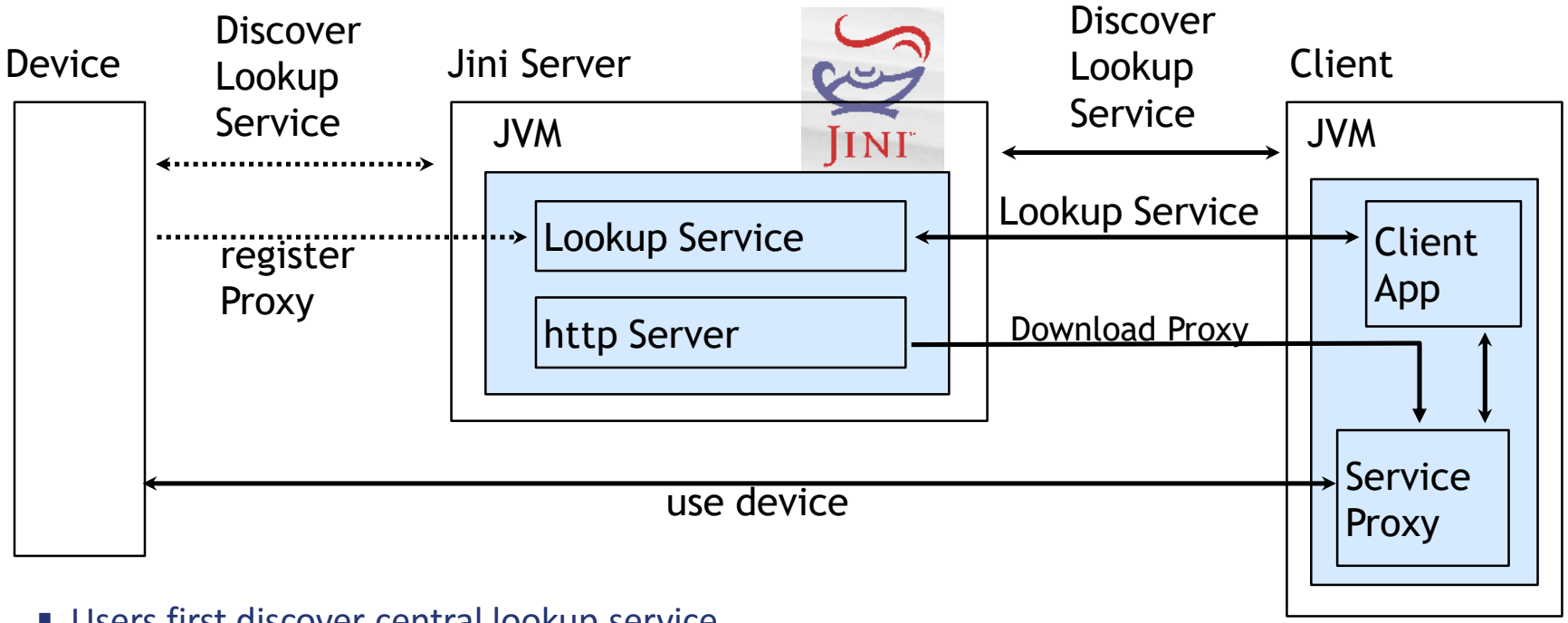
Jini: Requirements

TECHNISCHE UNIVERSITÄT DARMSTADT

Needed for participation in Jini:

- Properly functioning JVM, with all classes needed to run Jini
  - → Sufficient processing power and memory
  - → Java Virtual Machines (JVMs) everywhere?
    - If repository should be distributed -> yes
    - Proxy on Jini node can also represent external device
      - Repository then considered to be centralized
- Network protocol stack
  - Currently implementation uses IP
  - Other network protocols possible
  - Primarily, multicast facility and point-to-point communication
  - For IP, need TCP and multicast UDP
- Facility for determination of IP address
  - Not specified, i.e., may use DHCP, AutoIP or similar
- Facility for making code stubs available (e.g., a simple web server)

Jini: Architecture



- Users first discover central lookup service
- Device registers Java Proxy object with service description at registry (if it is capable of Java RMI)
- Client looks up service and downloads Proxy implementation
- Proxy then runs on client locally

Jini: Architecture

TECHNISCHE UNIVERSITÄT DARMSTADT

Services in Jini:

- A client in need of service ultimately downloads a Java object
- The object provides the service either:
  - Locally (e.g., algorithmically)
  - By invoking operations on remote server, possibly using private protocol
  - By interacting with a remote hardware device
  - By interacting with a remote human being
- Client view: no essential difference between these choices

Lookup Service:

- Centralized registry of services
- Repository of Java objects
- Objects are downloadable (mobile code)
- Object serves as client <-> service proxy
  - e.g., printer proxy: knows how to contact / talk to print service
  - e.g., equation solver: solves equation on remote server

Jini: Lookup Service

TECHNISCHE UNIVERSITÄT DARMSTADT

- Services and clients have to find the lookup service first
- Three protocols:
  - **Multicast request protocol** (UDP)
    - Used when service or client starts up
    - Request is sent to 224.0.1.85:4160 (7 times)
    - Lookup service sends unicast reply to address in request
  - **Multicast announcement protocol** (UDP)
    - Used when a lookup service starts up
    - Sent periodically
    - Packet contains hostname:port of lookup service
  - **Unicast discovery protocol** (TCP)
    - Used by service/client when address of lookup service is known
- Reply from lookup service contains proxy object of lookup service

Jini

TECHNISCHE UNIVERSITÄT DARMSTADT

- Service Description
  - Service „attributes“ described in serializable *Entry* object
  - Typed Key/Value-pairs
  - Allows custom types -> mobile code
- Service Lookup
  - Query specified in serializable *ServiceTemplate* object
  - **serviceID**: find specific service by ID
  - **serviceTypes**: list of interfaces which the service has to implement
  - **attrSetTemplates**: same as attributes
    - if present, must exactly match (equals)

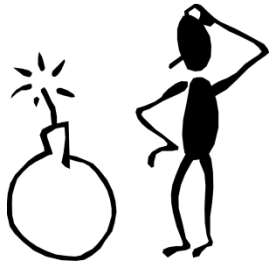
```
enum Technology {DOT_MATRIX, INK, LASER};

public class PrinterDescription
        implements Entry {    public Integer numberOfTrays;
   public Boolean isDuplex;
   public Boolean supportsColor;
   public Integer numberOfColors;
   public Technology printingTechnology;
   public PrinterDescription() {}
}
```

```
public class ServiceTemplate
      implements Serializable {
   public ServiceID serviceID;
   public Class[] serviceTypes;
   public Entry[] attrSetTemplates;}
```

Jini: Security

- Jini makes heavy use of mobile code
- A service could also provide this proxy object:

```
public class Printer implements Print {
 …
public void print(String text) {
   // death for Unix
   Runtime.getRuntime().exec("/bin/rm -rf /");
   // death for Windows
   Runtime.getRuntime().exec("format c: /u");
   Runtime.getRuntime().exec("format d: /u");
 }
```



OUCH!

➢ use digital signatures to verify authenticity of downloaded code

- but even with that...
  - DenialOfServiceAttack: memory usage, creation of large no. of threads
  - annoying sounds, „offensive" images, eMail to Microsoft, …
  - security *very critical* for mobile-object approaches!

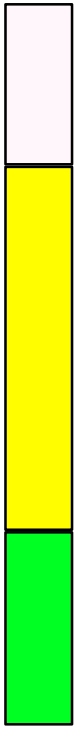
System Examples

TECHNISCHE UNIVERSITÄT DARMSTADT

- LAN/PAN scope
  - IP-based protocols
    - UPnP / SSDP
    - Bonjour / mDNS
    - SLP
  - Low level
    - Bluetooth SDP
  - Middleware
    - Jini
- Global scope
  - **UDDI**

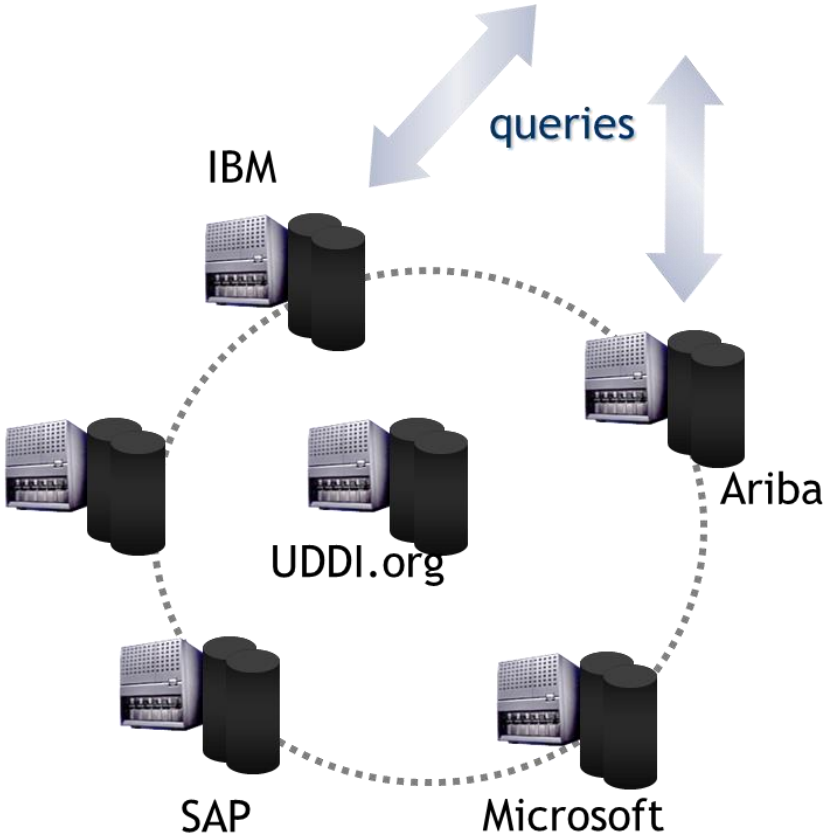
UDDI

- XML-based registry for businesses worldwide
  - **White pages**
    - Information about a business
    - Name, contact information, textual description
  - **Yellow pages**
    - Information about the business sector through standardized industry classification systems
      - North American Industrial Classification System (NAICS)
      - Universal Standard Products and Services Classification (UNSPSC)
      - Geographic Classification System (GCS)
  - **Green pages**
    - Technical information about services
    - Reference to WSDL description
- UDDI also defines a Web Service API for accessing it
  - Java API: JAXR (registry)



UDDI Vision

- Original plan was a global, replicated directory for all Web Services
  - Federated registry
  - Complete information at all nodes
  - Information registered with any node
  - Registrations replicated on a daily basis
- UDDI Business Registry (UBR) project discontinued by Microsoft, IBM & SAP since 2006



Summary: Service Discovery

TECHNISCHE UNIVERSITÄT DARMSTADT

- Idea: Smart Devices should easily „cooperate“
- General principles
  - Architecture layers: trading, naming, addressing
  - Information storage: central (federated), distributed, no registry
  - Service Description
- Systems
  - UPnP: Many protocols (SSDP, HTTP(M)U, SOAP, GENA) & XML-Templates
  - Bonjour: no new protocols, builds on DNS (mDNS)
  - Bluetooth SDP: low-level, non-IP
  - Jini: mobile code, Java everywhere
  - MundoCore
    - Distinction btw. node discovery and service discovery
    - Discussion of efficiency and reliability
  - UDDI: the standard for web services
- There is no magic!
  - Must agree on common protocol or interfaces (Jini)
  - Even worse: „service“ must have been „imagined“ before
    - e.g., connect PocketOffice to a Smart Smoke Detector: so what?

Summary: Infrastructure

TECHNISCHE UNIVERSITÄT DARMSTADT

- Basics of wireless technology
  - Understand the limitations
  - Understand the evolution going on
- Wireless technology
  - One size doesn’t fit all
- Distributed systems / Communication
  - TK1 et al.
- Service Discovery
  - Which service is available adhoc or pre-build?