

Ubiquitous Computing

(Ubiquitäre Informationstechnologien)

Vorlesung im WS 08/09

Christian Decker

Universität Karlsruhe
Institut für Telematik
Telecooperation Office
www.teco.uni-karlsruhe.de



Übersicht

Vorlesung Ubicomp

□ Geräte und Umgebungen

□ Kommunikation

- Grundlagen
- Kabelgebundene Kommunikation
- Kabellose Kommunikation
- Kommunikation Sensorknoten

□ Middleware

□ Kontext

□ HCI

Einführung: Ubicomp-Netze

Netze für Ubiquitous Computing

- Diversifikation von Endgeräten: mobil, eingebettet, spezialisiert
- Mobilität: mobile Nutzer, mobile Geräte
- Allgegenwart: überall, insbesondere auch im Heimbereich
- Spontaneität: ad hoc Vernetzung von Geräten
- Konvergenz: Daten, Audio/Video, Steuerung

Entwicklungstrends

- Nutzung „alter Infrastrukturen“ und Schaffung neuer
 - trad. LANs, Funk-LANs, Plug&Play-Busse, Bluetooth, IrDA, Phonline, Powerline, ...
- Extrem heterogene Umgebungen: Geräte und Netze
- hohes Maß an Dynamik: Hot&Plug'n'Play, mobile Netze, kurzlebige Verbindungen

Einführung: Ubicomp-Netze

Typische Szenarien und Herausforderungen

- über Mobiltelefone im Haus bedienen
 - heterogene Geräte und Netze (mobil/Heim)
 - kohärente Sicht auf Dienste im Heim
- Kamera sucht Drucker in fremder Umgebung
 - wie kann ein „geeigneter“ Drucker gefunden werden ?
 - wie kommuniziert man mit einem fremden Gerät ?
- Hausregelung
 - Heizung sucht Thermostat und Bewegungsmelder

Wichtigste Herausforderung: Komplexität verbergen

- vor allem vor dem Anwender
- keine manuelle Installation / Konfiguration (ad-hoc)
- Abstraktionen für die Anwendungsentwicklung → Middleware

Middleware

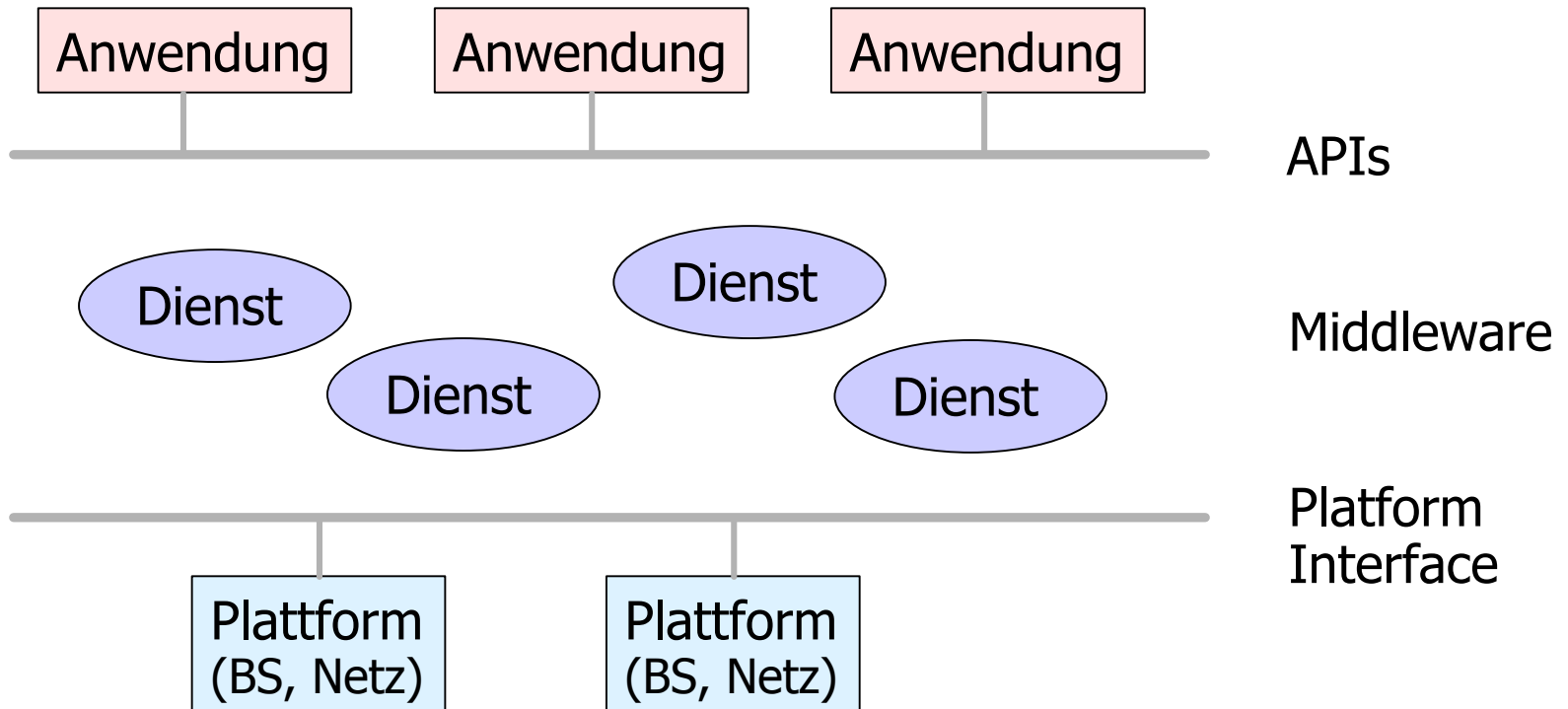
Was ist Middleware ?

- „der Slash in Client/Server“
- Komponenten für Entwicklung und Einsatz verteilter Systeme
- angesiedelt zwischen Netzwerktechnologie(n) und Anwendung

Wofür ?

- Abstraktion von Netzwerkprogrammierung
- Interoperabilität: Zwischenschicht über unterschiedlichen Geräte- und Netzwerkplattformen
- Komponenten für allgemeine Aufgaben in verteilten Systemen: z.B. Name Service, Security Service,...

Middleware



Middleware

RPC: Remote Procedure Call (80er Jahre)

- Prozedurales Paradigma
- entfernter Prozeduraufruf analog zu lokalem Aufruf
- Code für die Kommunikation wird automatisch erzeugt

Objekt-orientierte Middleware (90er Jahre)

- Objekt-orientierte Programmierung
- Kommunikation mit entfernten Objekten über automatisch erzeugte lokale Proxies (z.B. „stubs“ in CORBA)
- Vermittlungsdienste (z.B. Object Broker)

Java Remote Method Invocation

- Java's Middleware für Methodenaufrufe von Objekten, die in verschiedenen VM ablaufen

Middleware für Ubicomp

Integration heterogener Geräte

- Gateways für kohärenten Zugang zu heterogenen Umgebungen
- Service Paradigma: abstrahierte, beschriebene Nachrichten; spontane Bildung verteilter Systeme

Service Gateways

- Bündelung von Diensten über Gateways
- Residential Gateways: Verbindung zwischen Heimnetz und Außenwelt (= Internet)
 - bietet Geräten im Haus Zugriff auf Internet-Dienste
 - bietet externen Dienst Anbietern kohärenten Zugang zu Geräten/Infrastruktur im Haus (z.B. für Fernwartung, Sicherheitsüberwachung,...)
 - Administration durch Gateway Operator
 - Standardisierung: Open Service Gateway Initiative (OSGi)

Kommunikationsparadigmen

Ablauf einer Kommunikation

- Initiierung: Auswahl der Kommunikationspartner
→ Wie **Auswahl**
- Durchführung: Austausch von Daten
→ **Grund der Kommunikation**
- Beendigung

Auswahl

	ID	Dienst	Kontext
Info	HTTP		IrOBEX
Dienst		Jini, UPnP, HAVi, Salutation	Forschung
Kontext			Forschung

Service Paradigma

Everything is a Service

- Geräte ebenso wie Software
- vgl. Objekt-orientierung: „everything“ is an object
- Services werden durch Interfaces deklariert, über die sie ihre Funktionalität zur Verfügung stellen
- Services werden beschrieben durch Typ und Attribute
- Services können sich zu Systemen verbünden („federation“)

Beispiele für Services

- Kamera, Drucker, Fax, Scanner, Speicher, Rechenleistung
- Türöffner, Beleuchtung, Alarmanlage, Stromzähler, ...
- Rechtschreibprüfung, Formatkonvertierung, ...
- Online Banking, Aktienhandel, ...
- Hotelführer, Stadtplan, ...

Service Paradigma

Netzwerk-zentrisch

- „the network is the computer“
- Netzwerk = Hardware und Softwareinfrastruktur für Dienste
- Sichtweise: „Netzwerk, an das Geräte angeschlossen sind“ (statt „Geräte, die vernetzt werden“)
 - Netzwerk existiert immer, Geräte/Dienste sind transient
 - Komponenten und Kommunikationsbeziehungen kommen und gehen

Spontane Vernetzung

- Services finden sich in der offenen Netzwerkkumgebung zu zeitweiligen Verbundsystemen zusammen
- müssen sich dazu nicht a priori kennen
- typisches Szenario: Client wacht auf und fragt nach Diensten in der lokalen Umgebung

Service Paradigma

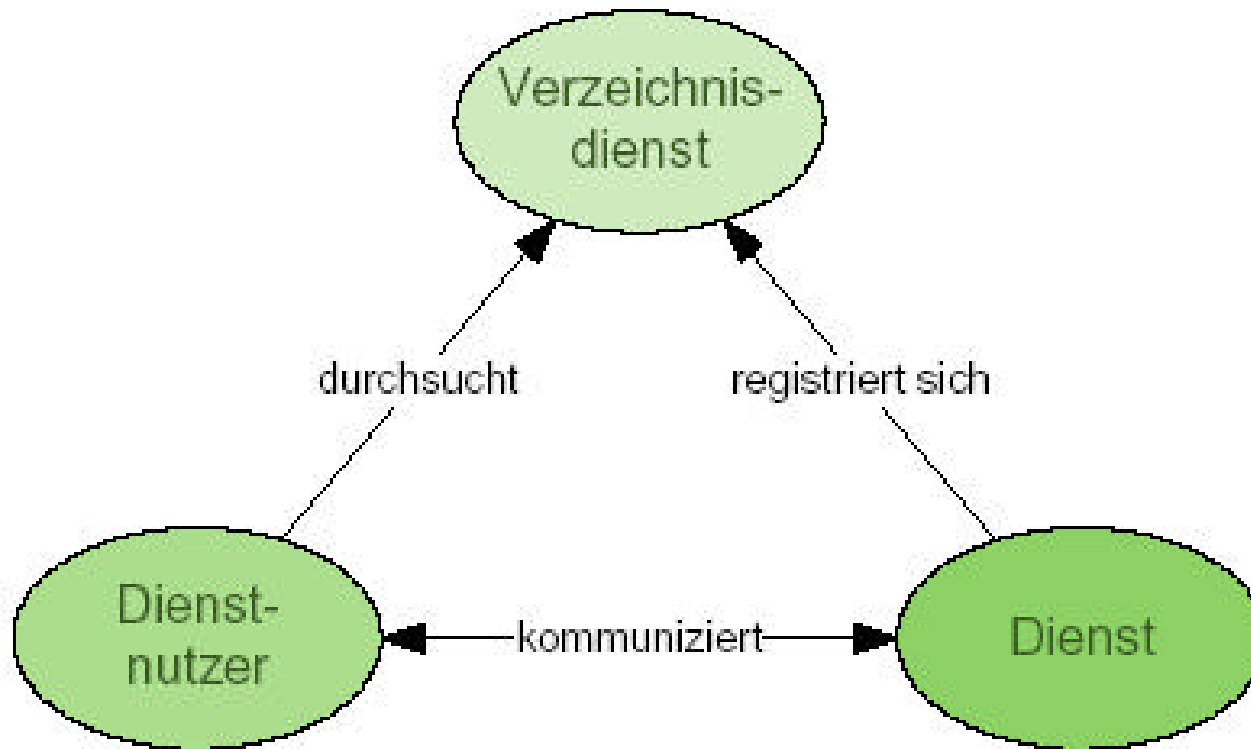
Spontane Vernetzung von Services

- wie werden Services aufeinander aufmerksam ?
- wie können bestimmte Services in einer fremden Umgebung gefunden werden (Flexibilität!)?
- wie verständigen sich Services, wenn sie sich gefunden haben ?
- Werden Dienstinfo. in Infrastruktur gehalten oder ad-hoc ermittelt?

Infrastruktur für Service Discovery

- „Registry“: Verzeichnis/Vermittlung von Services
- Protokolle zum Registrieren und zum Anfragen von Services
- Protokolle für Client-Zugriff auf Service, und für die Nutzung von Services durch Clients
- z.B. Sun's Jini aufbauend auf Java/RMI, Microsoft's UPnP (Universal Plug & Play)
- z.B. HAVi (Home Audio/Video interoperability) aufbauend auf IEEE.1394 für Home Entertainment Dienste

Service Orientierte Architektur (SOA)



Middleware

Beispiel: Jini

Jini Service Infrastruktur

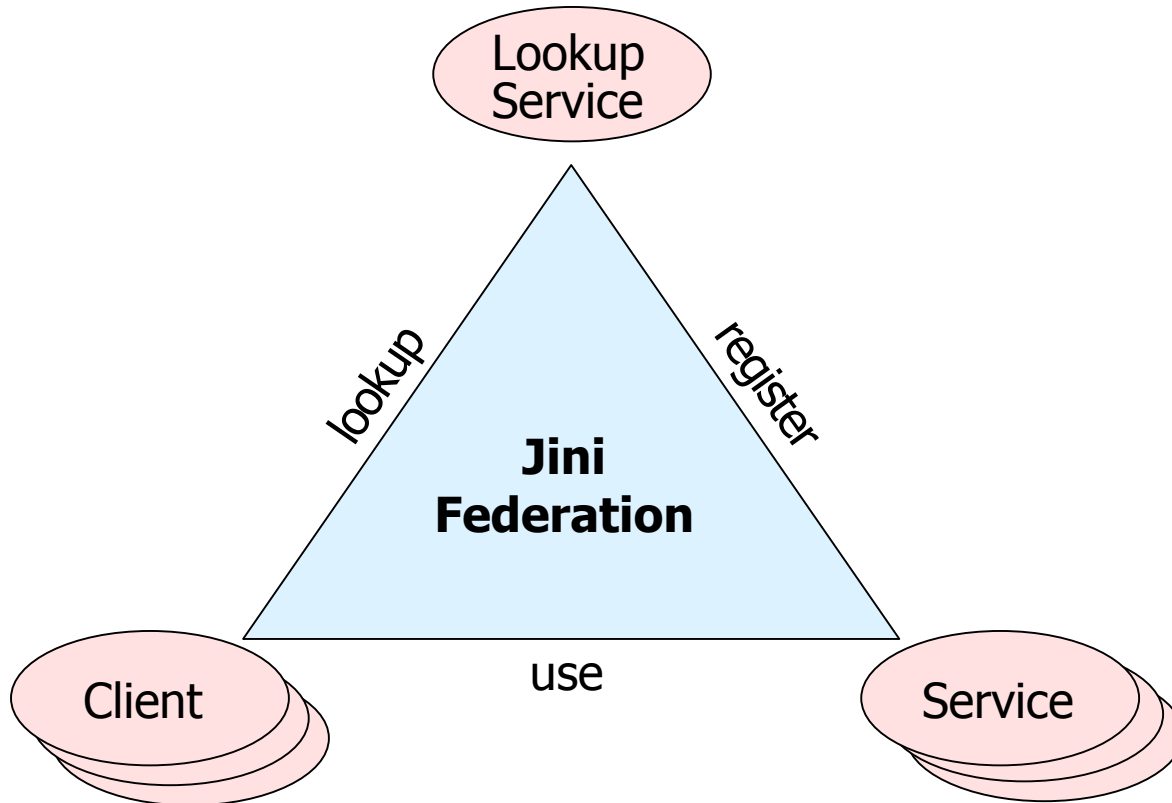
Hauptkomponenten

- Lookup Service (LUS): „Registry“ für Services
- Protokolle basierend auf TCP/UDP/IP
 - Discovery & Join, Lookup von Services
- Proxy Objekte
 - als lokale Vertreter für Services

Lookup Service

- Verzeichnis ähnlich RMI Registry
- Aufgabe: „Helpdesk“ für Services/Clients
 - Registrierung von Services, die angeboten werden
 - Verteilung von Diensten an anfragende Clients

Jini Service Infrastruktur



Discovery: Finden eines LUS

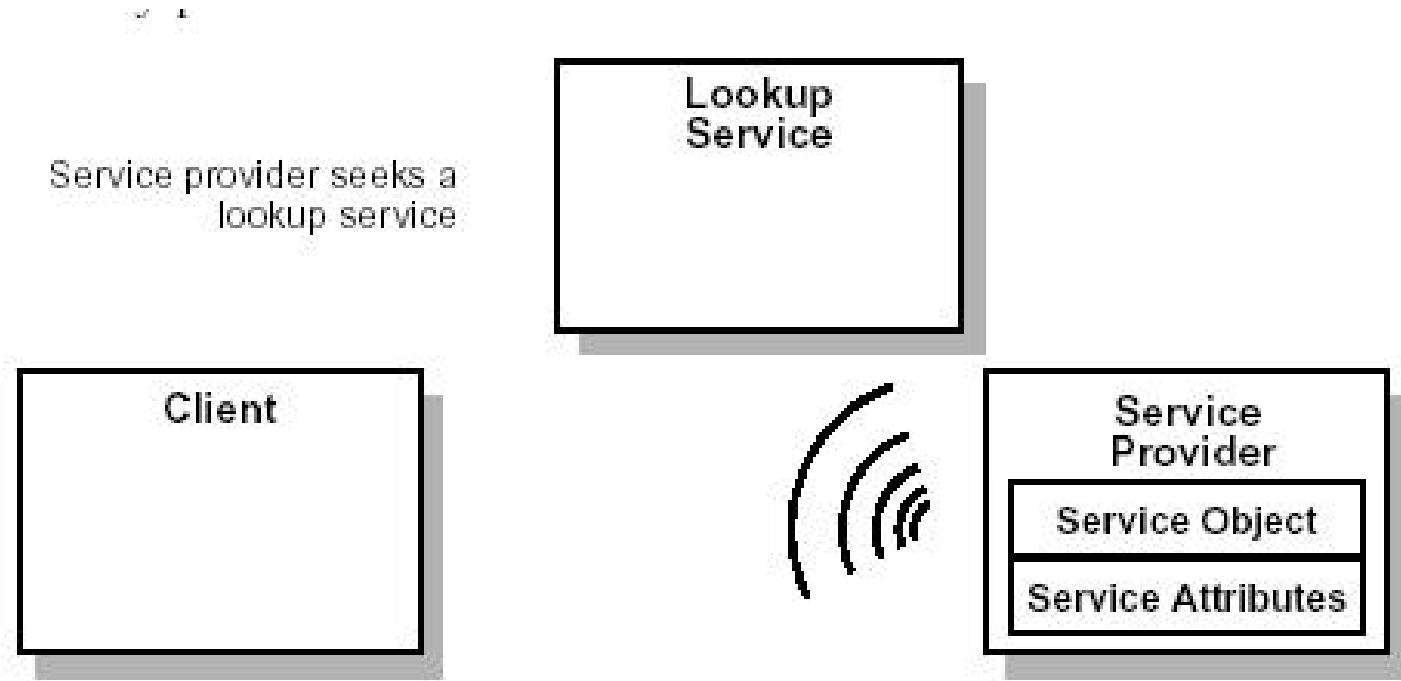
Finden eines Lookup Services

- ... ohne a priori Kenntnis des Netzwerks
- Service Provider sucht LUS um Service anzumelden (register)
- Client sucht LUS um einen Service anzufragen (look up)

Discovery Protokoll

- Multicast-Anfrage an bekannten Port
- Lookup Service lauscht auf entsprechendem Port und antwortet mit Proxy Objekt
 - Proxy Objekt wird in den anfragenden Client od. Provider geladen
 - Kommunikation mit LUS dann über den Proxy
- weitere Discovery-Protokolle
 - Unicast: Service kann LUS direkt ansprechen, wenn er die IP-Adresse schon kennt
 - Multicast Announcement: LUS meldet sich per Multicast, z.B. nach Ausfall

Discovery



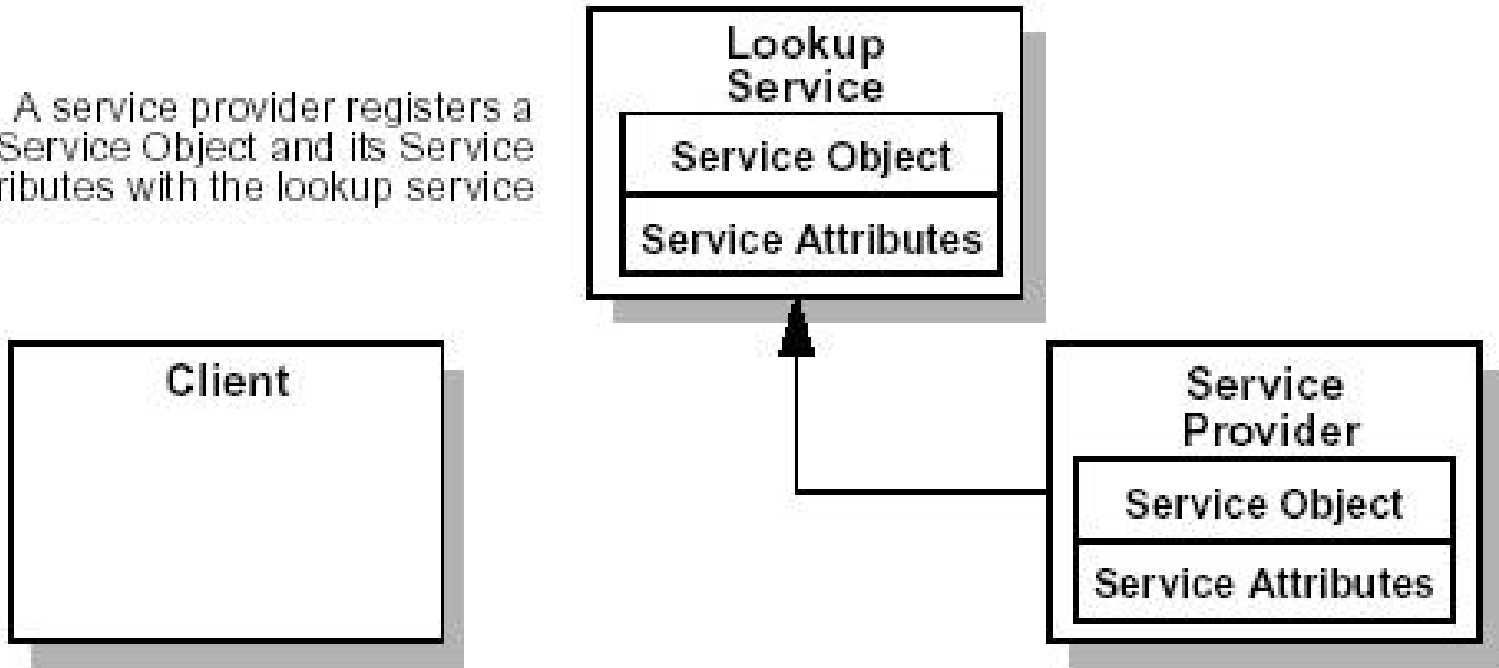
Join: Registrieren eines Services

Join Protokoll

- Service Provider hat einen Proxy des LUS für die Kommunikation empfangen
- Provider registriert über den Proxy seinen Service: register()
- Provider übergibt dabei dem LUS
 - den eigenen Service Proxy
 - Attribute, die den Dienst beschreiben (z.B. „600 dpi“, „version 21.1“, ...)
- Service tritt mit dem Join in den Jini-Verbund ein
 - Provider kann nun über den LUS gefunden und von anderen Services genutzt werden

Join

A service provider registers a Service Object and its Service Attributes with the lookup service



Lookup: Suchen von Services

Lookup Protokoll

- Client sucht bestimmten Service
- kennt LUS und verfügt über Proxy für die Kommunikation (via Discovery Protokoll)
- sendet Anfrage an LUS in Form eines „Service Template“
 - ID, Typ, Attribute
- LUS antwortet mit keinem/einem/mehreren passenden Services
 - ggf. Auswahl im Client
- Client erhält vom LUS Proxy des vermittelten Services
- Client nutzt Proxy für direkte Kommunikation mit dem Provider
 - beliebiges Protokoll
 - Proxy: Gateway zu Service-Funktionalität beim Provider
 - Proxy kann aber auch (Teil der) Service-Funktionalität implementieren, d.h. Ausführung beim Client

Lookup: Service Matching

Service Template

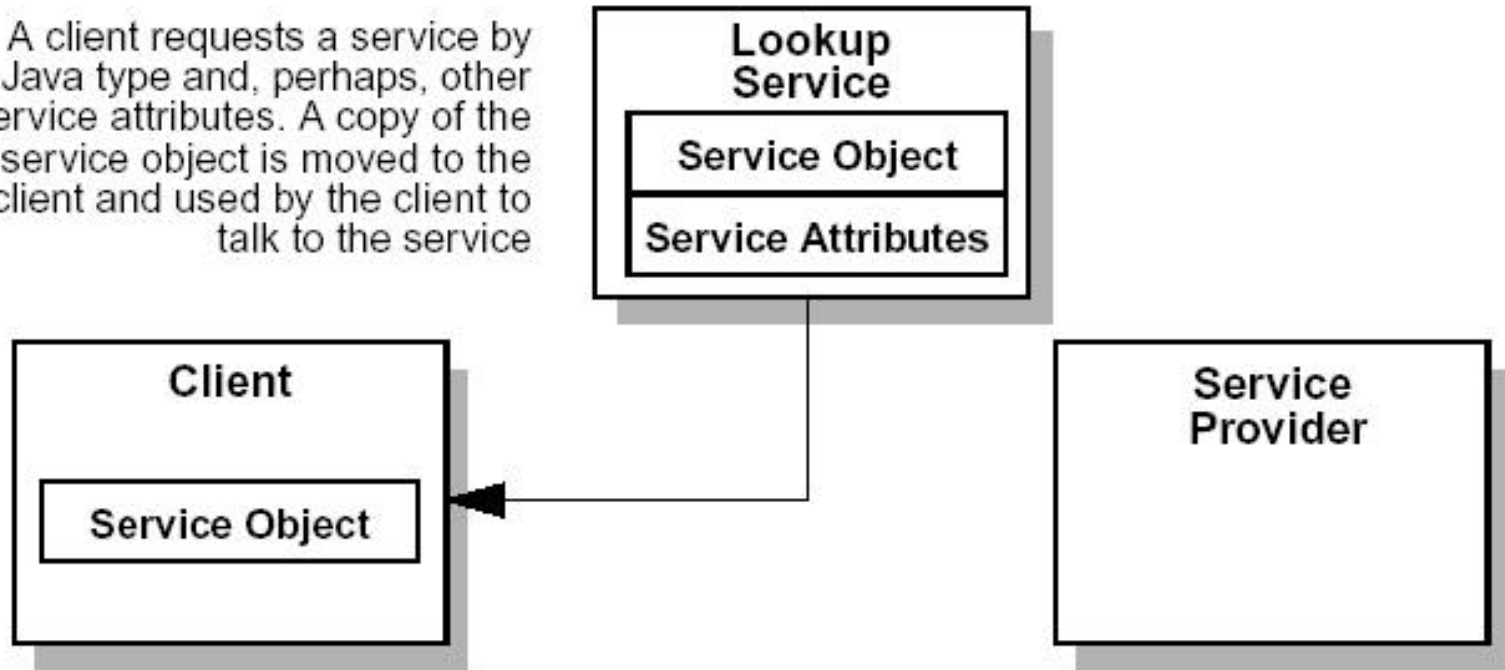
- Service ID (Registration Number): kann angegeben werden, falls Dienst bereits bekannt ist (durch frühere Nutzung)
- Service Typ: definiert durch die Schnittstelle
- Attribute (sog. Entries), die den Service beschreiben

Service Matching

- Übereinstimmung via Attribute: Mehrwert gegenüber traditionellem Naming Service: Service-Auswahl über beschreibende Merkmale
- aber nur exakte Übereinstimmung, kein „größer als“, keine Query-Sprache
 - z.B. Anfrage an „600dpi“ Drucker stimmt nicht mit registriertem „1200dpi“ Drucker überein

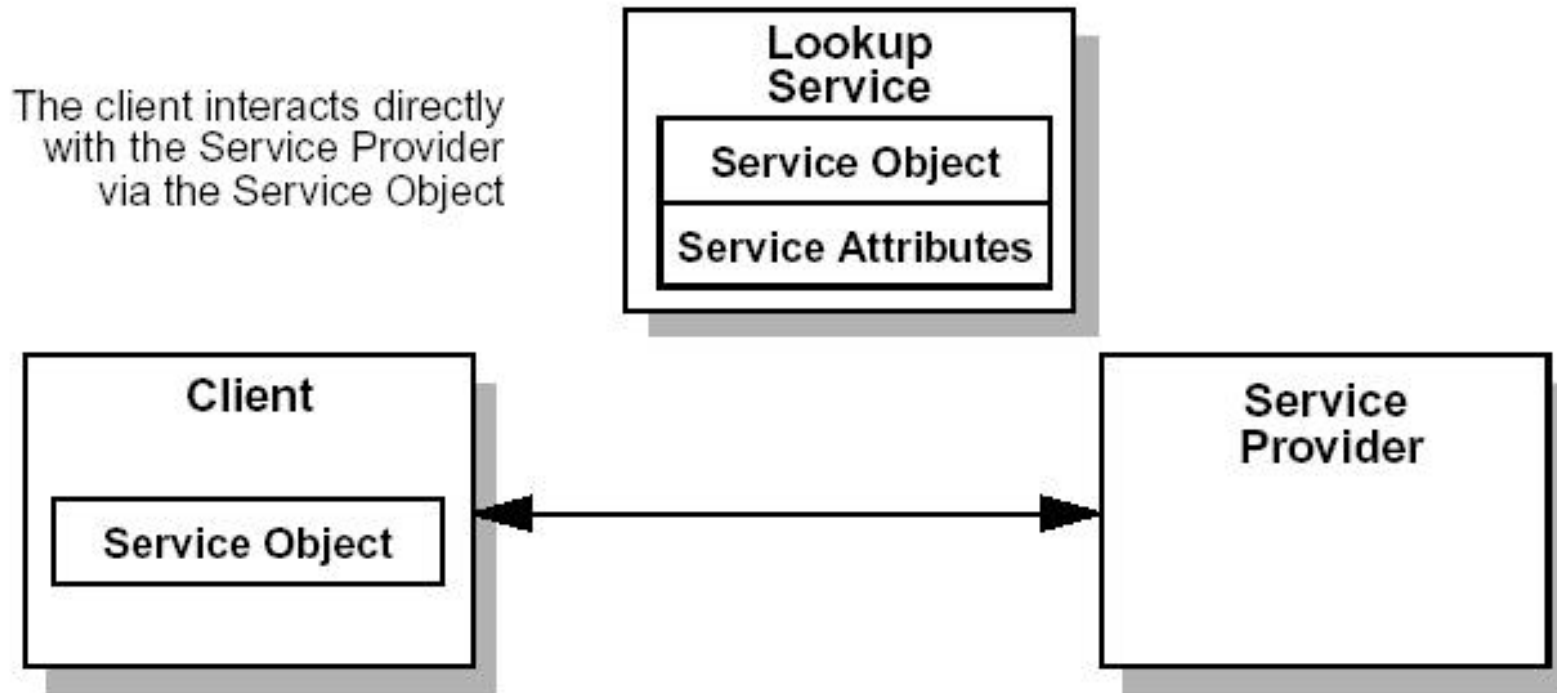
Lookup

A client requests a service by Java type and, perhaps, other service attributes. A copy of the service object is moved to the client and used by the client to talk to the service



Service Invocation

- Client kann nun auf Service zugreifen
- Protokoll zwischen Client und Service nicht festgelegt



Service Paradigma

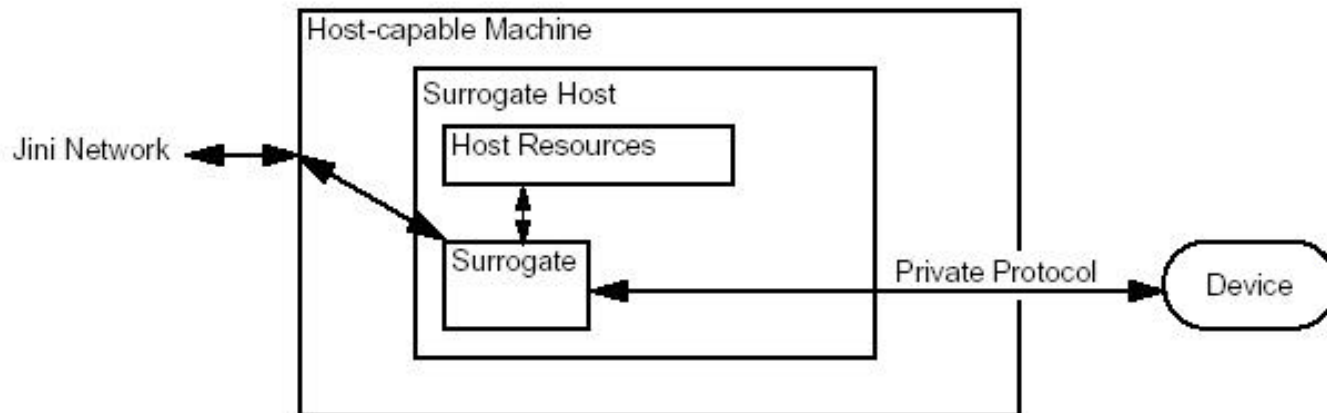
Spontane Vernetzung von Services: Jini Konzepte

- wie werden Services aufeinander aufmerksam ?
 - Jini: Lookup Services als Vermittlungsstelle, Registrierung von Services über Discovery & Join
- wie können bestimmte Services in einer fremden Umgebung gefunden werden ?
 - Discovery von Lookup-Services als Verteiler in fremder Umgebung
 - Lookup anhand von Service Templates, insbesondere auch anhand beschreibender Attribute
- wie verständigen sich Services, wenn sie sich gefunden haben ?
 - Service Proxy wird in den anfragenden Client geladen
 - beliebiges Protokoll, kein spezifisches Aushandlungsverfahren

Jini Surrogates

Motivation

- Anbindung von Geräten, die nicht direkt Jini unterstützen
- Für Geräte, die
 - Kein Code Download unterstützen
 - Limitierte Rechen- und Netzwerkressourcen besitzen



Eigenschaften der Surrogate Architektur

- Device type independence
- Network type independence
- Preserve plug-and-work

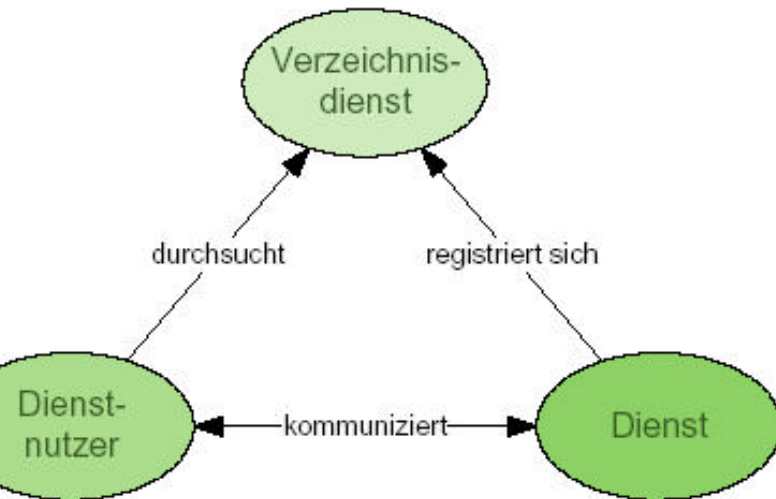
Quelle: <https://surrogate.dev.java.net>

Middleware

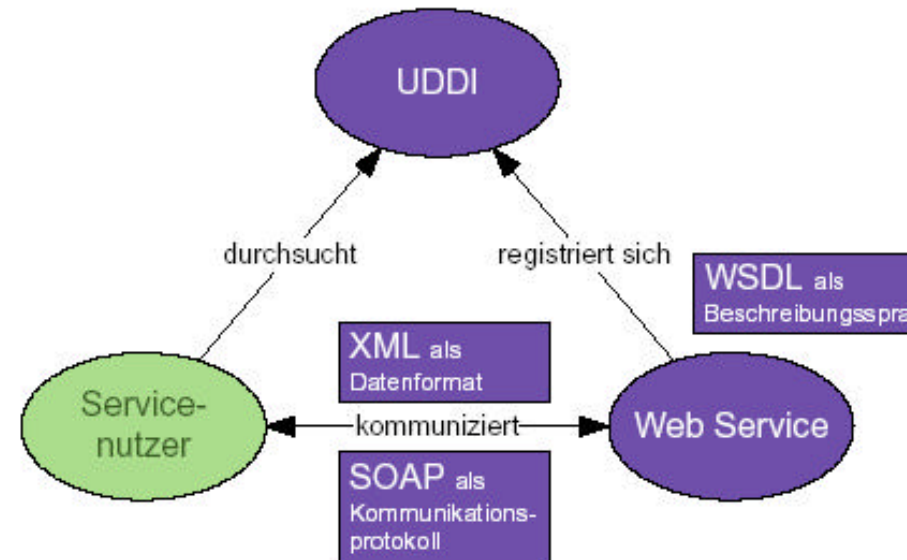
Webservices

Architektur

Allg. SOA



Webservices



- SOA über Standard-Webtechnologien, z.B. http, XML Beschreibung

Bestandteile

- **WebService (WS)**: Endpunkt, der Nachrichten sendet und empfängt
- **WSDL**: Beschreibungssprache des WS, Beschreibung der Schnittstelle = Funktionen, Daten, Datentypen und Austauschprotokolle
- **SOAP**: „Protokoll“ für den Austausch strukturierter Informationen
 - Keine Semantik, lediglich Rahmen zur Beschreibung beliebiger applikationsspezifischer Nachrichten und deren Austausch
 - transportneutral
 - Format: XML
 - Schicht: Anwendungsschicht (über http)

SOAP

- SOAP: Modell zur Beschreibung der Anforderungen an die Nachrichtenverarbeitung
 - dokumenten-zentrierte Kommunikation
 - Vgl. Briefverkehr
- Beschreibung ermöglicht
 - Hohe Flexibilität, Erweiterbarkeit, Sprach- und Plattformunabhängigkeit
 - Lose Kopplung, d.h. keine Kopplung über Funktionssignatur → erlaubt komplexe Interaktionen zwischen Dienstanbieter und Nutzer
 - Komplexe Transaktionen in kompakten Nachrichten
- Aber:
 - Mitführen von Meta-Daten
 - XML overhead

Probleme für
Kleinstgeräte

Lesen Sie ...!!!

Ingo Rammer. SOAP is Not a Remote Procedure Call

- „Well, you now have to change the way you think about Web Services. Forever!“
- Quelle:
<http://static.thinktecture.com/Resources/ArchitectureBriefings/SoapIsNotARemoteProcedureCall.pdf>

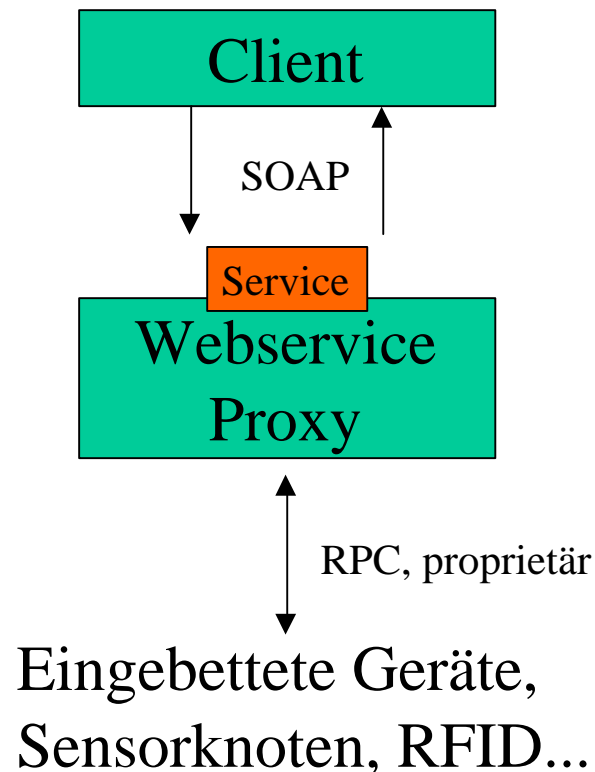
Charles Skinner. Soap Fight: RPC vs. Document

- Quelle: <http://www.eherenow.com/soapfight.htm>

Anbindung via Webservice

Stand heute:

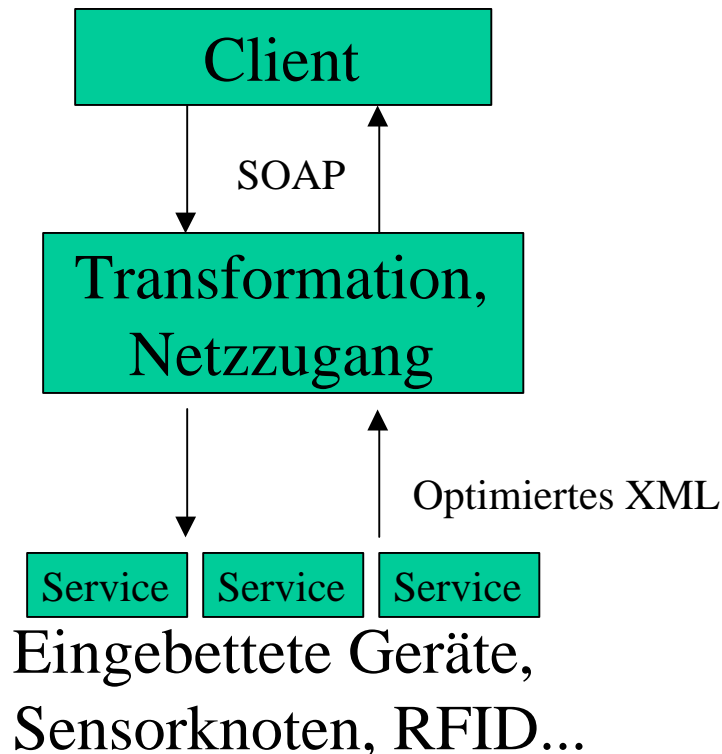
- Zwischensystem bietet Service an, RPC zu Geräten



Anbindung via Webservices

Ziel: Ende-zu-Ende (aktuelle Forschung)

- Dokumentenzentrierte Kommunikation erlaubt Transformation → Reduktion des Overhead von SOAP
- Service wird von Geräten angeboten



Middleware

Beispiel: DPWS

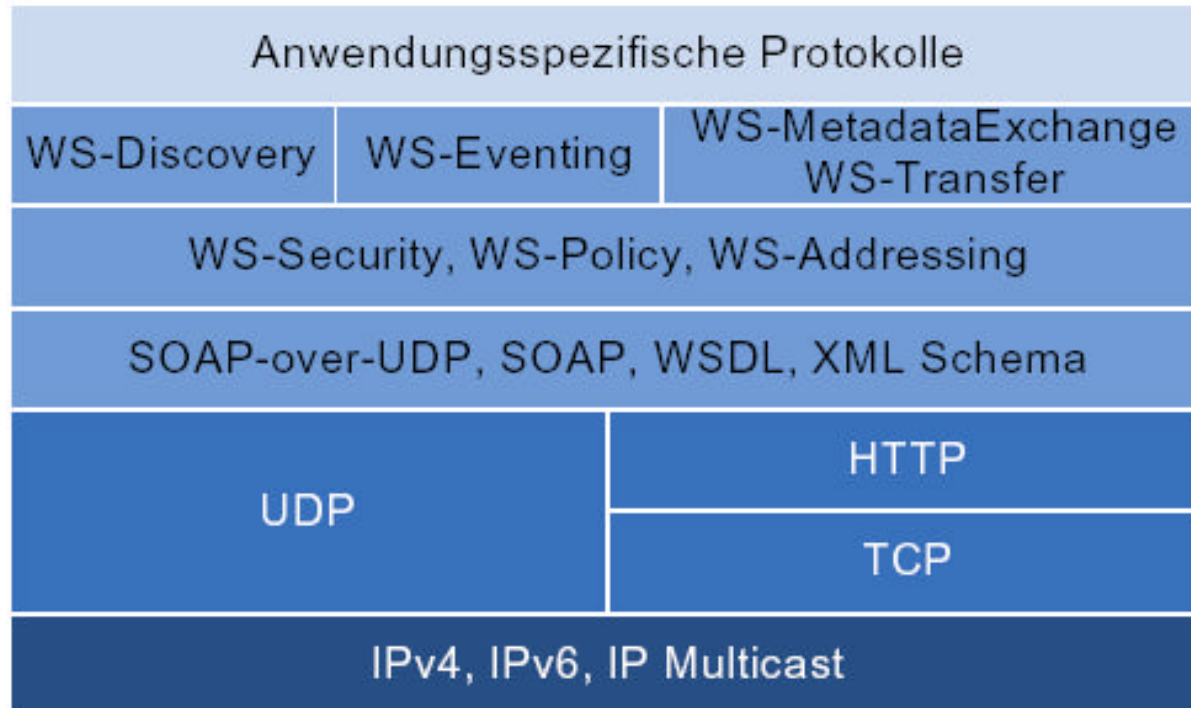
DPWS

= Device Profile for Webservices

- Standard für WebServices auf eingebetteten, ressourcenbeschränkten Geräten
- Hosting Services: Geräte sind auch Services
- Hosted Services: Services auf Geräten

- DPWS standardisiert Funktionen für die Anbindung von Geräten mit Hilfe von WS Technologien
- Discovery: Finden von Geräten ohne Verzeichnisinfrastruktur (adhoc)
- Metadatenaustausch: Beschreibung des Gerätes und der Services, z.B. Hersteller, Version, WSDL(!)
- Ereignisbehandlung (Eventing):
Publish/Subscribe/Notification

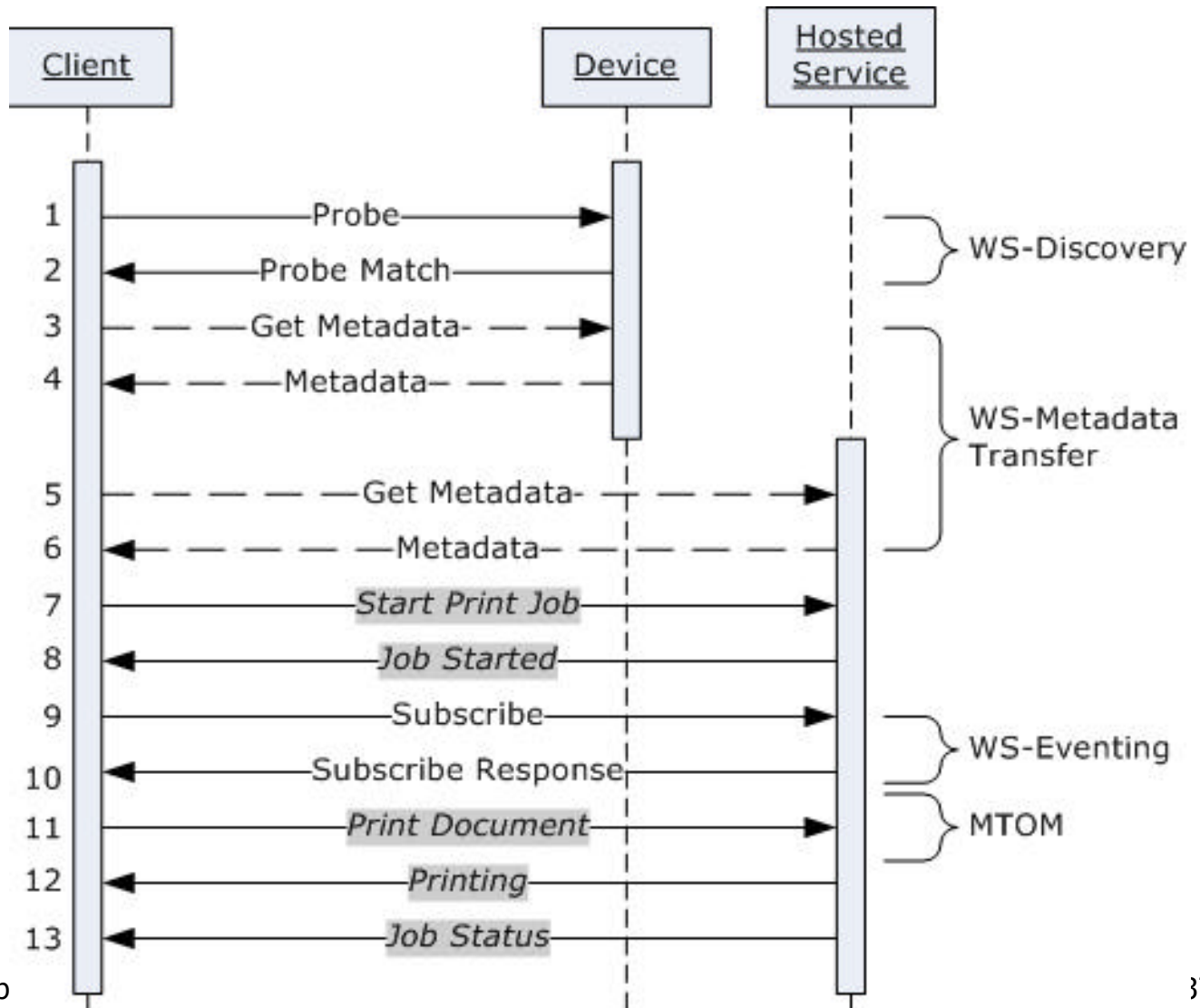
DPWS Stack



Middleware as a Service

- Ereignisbehandlung, Entdecken, ... sind als Services beschrieben

DPWS Beispiel: Drucker nutzen



Wertung DPWS

- Festgelegte Discovery: ProbeMatching over UDP
- SOAP erzeugt hohen Overhead
- Geräte müssen XML und SOAP unterstützen
 - Für Mobiltelefone möglich
 - Für Sensorknoten schwierig (BinaryXML)
- Verbreitung:
 - DPWS integriert in Windows Vista
 - Aber: UPnP war in Windows XP → UPnP hat sich nicht breit durchgesetzt
- Gesamtwertung: erster Schritt in die richtige Richtung