

Ubiquitous Computing

(Ubiquitäre Informationstechnologien)

Vorlesung im WS 01/02



Prof. Lars Wolf

Michael Beigl

Universität Karlsruhe

Institut für Telematik

Telecooperation Office

www.teco.uni-karlsruhe.de

Aufbau der Vorlesung

1 Grundlagen

2 Geräte

3 **Vernetzung**

Netzwerktechnologien

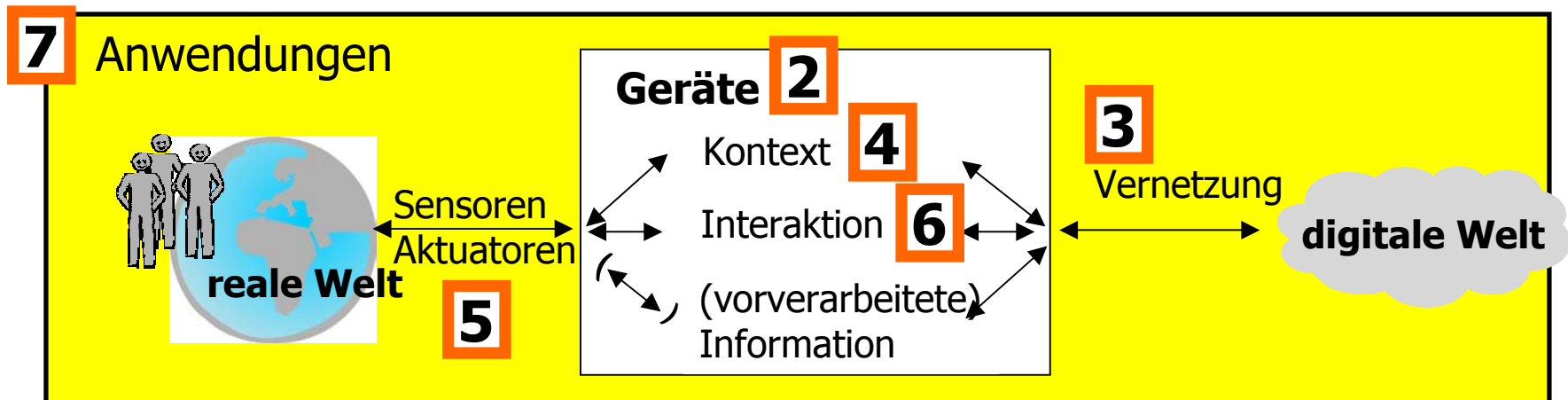
höhere Schichten

4 Kontext

5 Sensoren/Aktuatoren

6 Interaktion

7 Anwendungen



Vernetzung: höhere Schichten

- **Einführung**
- **Kommunikationsparadigmen**
- **Jini Service Infrastruktur**
- **JetSend Ad hoc Kommunikation**
- **OBEX Object Exchange Protokoll**
- **HAVi AV Kontrolle**
- **UPnP Plug&Play**

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, Phonenumber, Powerline, ...
- Extrem heterogene Umgebungen: Geräte und Netze
- hohes Maß an Dynamik: Hot&Plug 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 unterhält man sich 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

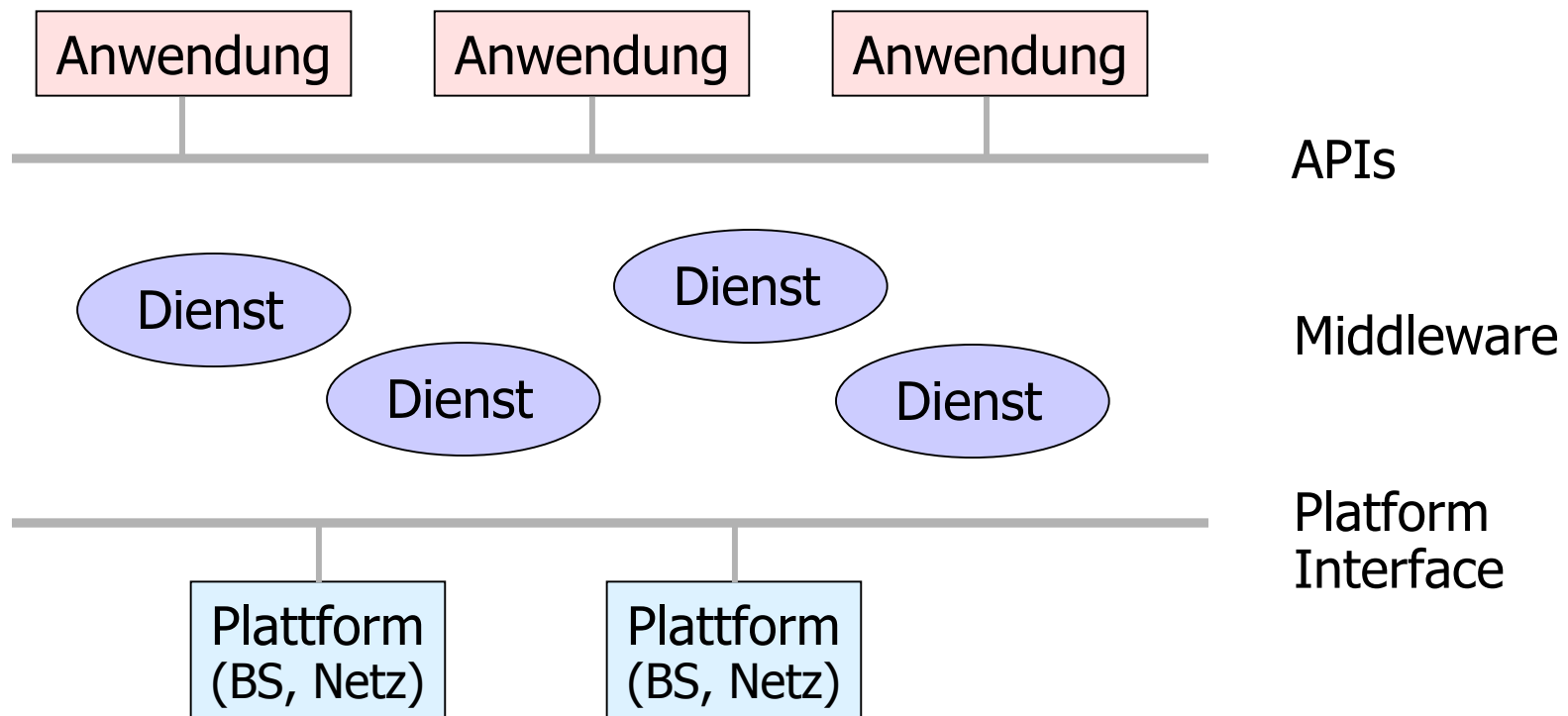
Was ist Middleware ?

- „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: dynamischer Verbund von Geräten ohne zentrale Komponente; 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 Diensteanbietern kohärenten Zugang zu Geräten/Infrastruktur im Haus (z.B. für Fernwartung, Sicherheitsüberwachung,...)
 - Administration durch Gateway Operator
 - Standardisierung: Open Service Gateway Initiative (OSGi)

Vernetzung: höhere Schichten

- Einführung
- **Kommunikationsparadigmen**
- Jini Service Infrastruktur
- JetSend Ad hoc Kommunikation
- OBEX Object Exchange Protokoll
- HAVi AV Kontrolle
- UPnP Plug&Play

Kommunikationsparadigmen

Ablauf einer Kommunikation

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

Auswahl

	ID	Dienst	Kontext
Info	HTTP		IrOBEX
Dienst	Jetsend	Jini, UPnP, HAVi	
Kontext			RAUM

Kommunikationsparadigmen

Ablauf einer Kommunikation

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

Auswahl

	ID	Dienst	Kontext
Info	HTTP		IrOBEX
Dienst	Jetsend	Jini, UPnP, HAVi	
Kontext			RAUM

Service Paradigma

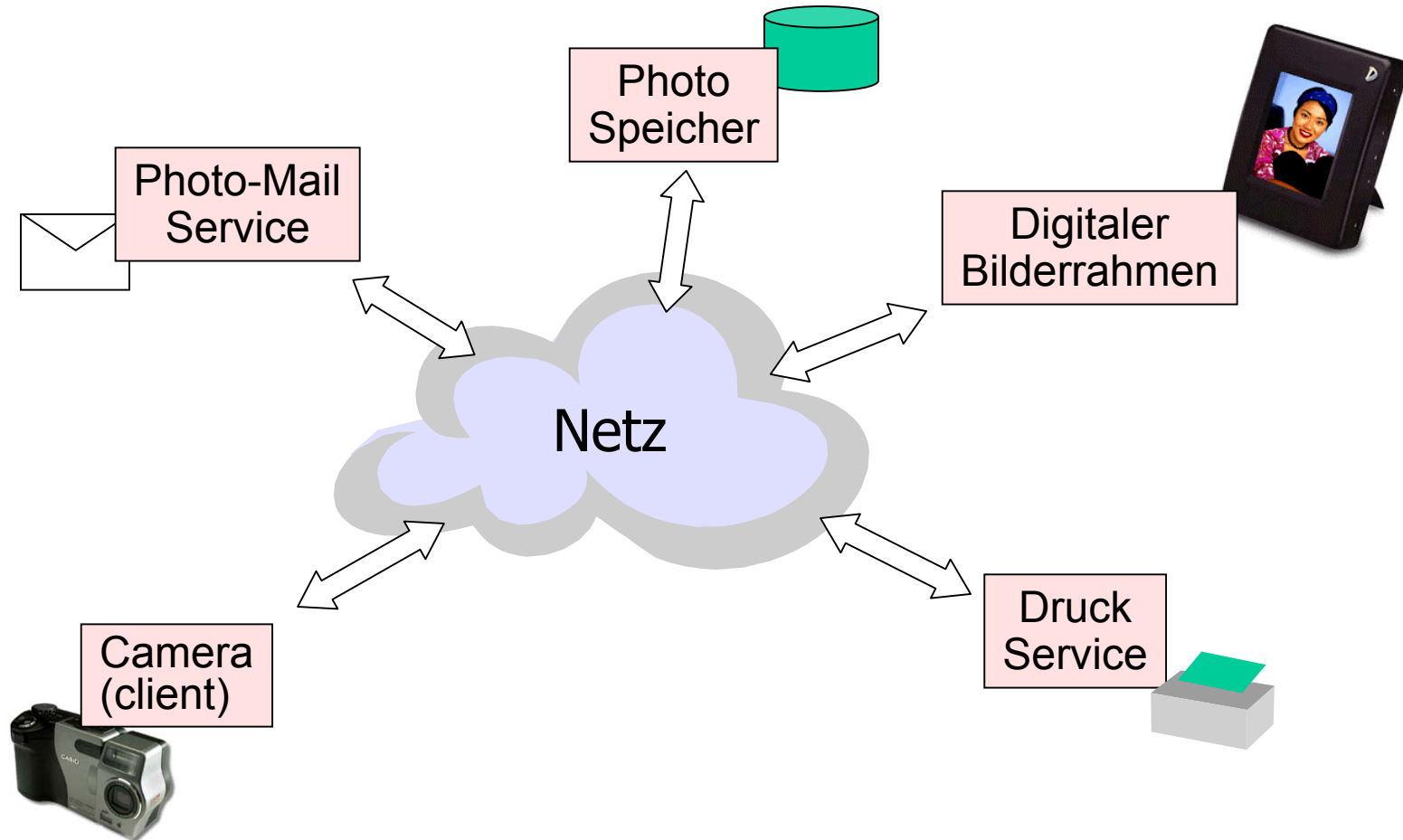
Everything is a Service

- Geräte ebenso wie Software
- vgl. Objekt-orientierung: „everything“ is an object
- Services werden durch Interfaces definiert, 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 Federation: Beispiel



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 ?
- wie verständigen sich Services, wenn sie sich gefunden haben ?

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

Vernetzung: höhere Schichten

- Einführung
- Kommunikationsparadigmen
- **Jini Service Infrastruktur**
- JetSend Ad hoc Kommunikation
- OBEX Object Exchange Protokoll
- HAVi AV Kontrolle
- UPnP Plug&Play

Jini Service Infrastruktur

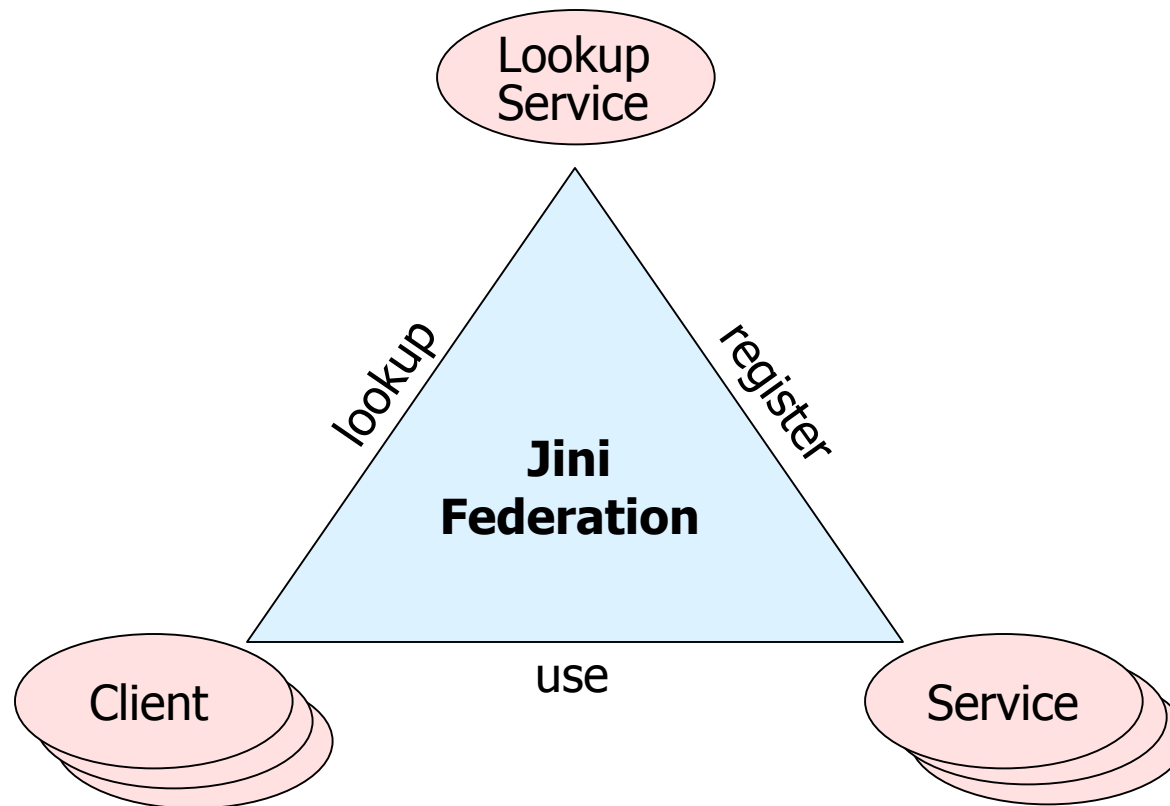
Hauptkomponenten

- 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 Lookup Service



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 (interface `ServiceRegistrar`)
 - Proxy Objekt wird in die anfragenden Service 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

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

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 exaktes Übereinstimmung, kein „größer als“, keine Query-Sprache
 - z.B. Anfrage an „600dpi“ Drucker stimmt nicht mit registriertem „1200dpi“ Drucker überein

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

Vernetzung: höhere Schichten

- Einführung
- Kommunikationsparadigmen
- Jini Service Infrastruktur
- **JetSend Ad hoc Kommunikation**
- OBEX Object Exchange Protokoll
- HAVi AV Kontrolle
- UPnP Plug&Play

HP Jetsend^{+sept01}

Hewlett-Packard Company will not be initiating any new projects or project extensions relating solely to HP Jetsend beyond those currently in development at HP. HP will continue to protect HP Jetsend intellectual property.

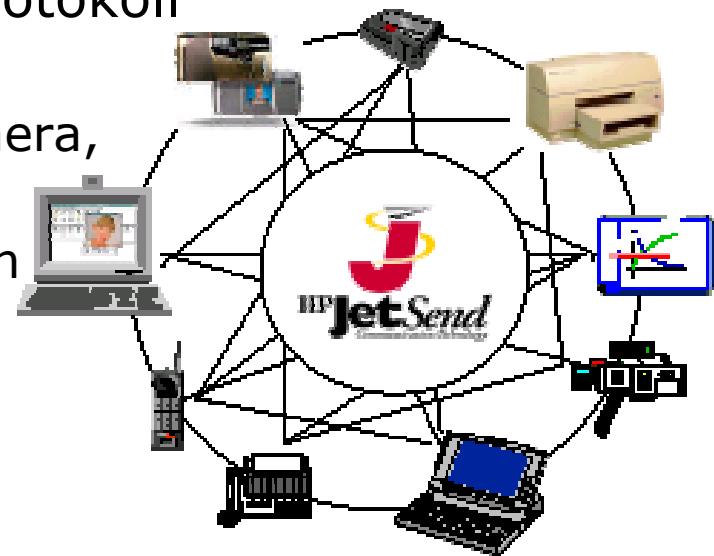
JetSend Ad hoc Kommunikation

Spontane Vernetzung von Services

- wie verständigen sich Services, wenn sie sich gefunden haben ?

JetSend (HP, 1997* 2001+)

- Universelles Kommunikationsprotokoll für Informationsgeräte
 - beliebige Geräte: Drucker, Kamera, Scanner, Projektor, ...
 - Aushandlung von Formaten zum Austausch von Information
- Punkt-zu-Punkt
 - geht von direkter logischer Verbindung zw. Geräten aus
 - Abstraktion vom Netzwerk (Medium, Verbindung, Transport)



JetSend Ad hoc Kommunikation

JetSend Protokoll

- Geräte- und gerätetypunabhängig
 - Im Protokoll und in den Geräten befinden sich keine Informationen über andere Geräte/Typen
 - Beispiel: JetSend-Kamera kann Information mit beliebigem JetSend-Drucker austauschen ohne den Druckertyp zu kennen
- Aushandlung der Informationsbeschreibung
 - Geräte verhandeln über unterstützte Codierungen mit dem Ziel, den größten gemeinsamen Nenner zu finden
- Hauptkonzepte
 - Surface: Gegenstand des Informationsaustauschs
 - Surface Interaction: das Protokoll für Informationsaustausch
 - E-Material: Austauschformat für Surface Interaction

JetSend Ad hoc Kommunikation

Surface

- „Oberfläche“: darstellungsnaher Information, z.B. Bilder, Dokumente, Statusmeldungen
- bestehen aus Beschreibung und Inhalt

Surface Interaction

- Austausch von Surfaces zwischen Geräten
 - Original-Surface wird als *Expression* („Ausdruck“) bezeichnet
 - ein empfangendes Gerät erhält eine *Impression* („Eindruck“)
- Aushandlung von Inhalts-Codierungen
 - beim Erzeugen einer Impression wird nur die Beschreibung gesendet, die eine Liste alternativer Codierungen enthält
 - der Empfänger wählt anhand der Beschreibung eine geeignete Codierung, in der der Inhalt angefordert wird

JetSend Ad hoc Kommunikation

Surface Interaction Messages

SurfaceRequestMsg

- Anforderung einer Surface (identifiziert durch einen Namen)

SurfaceMsg

- Erzeugen einer Impression, Senden der Beschreibung

ContentRequestMsg / ContentReplyMsg

- Übertragung des Inhalts

DescriptionRequestMsg / DescriptionReplyMsg

- Übertragung weiterer Beschreibungen

SurfaceChangeMsg

- Benachrichtigung über Änderungen einer Surface

SurfaceReleaseMsg

- Verbindung zwischen Expression und Impression aufheben

JetSend Ad hoc Kommunikation

E-Material

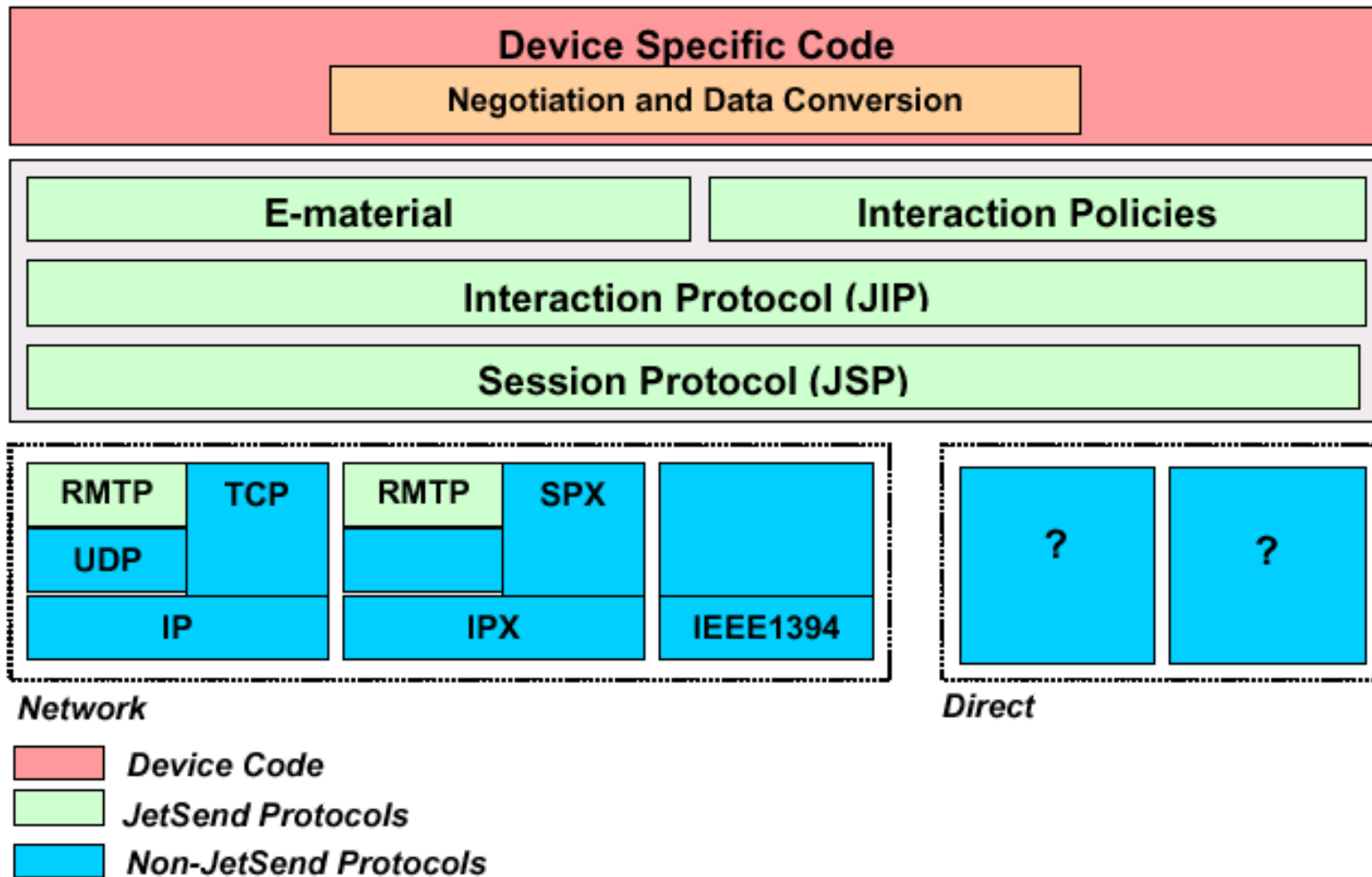
- Format für Surface Messages
 - definiert Format, in dem Information (Beschreibung und Inhalt) zwischen Geräten ausgetauscht wird
 - aber nicht wie Geräte diese Information verarb./speichern
- Beschreibung von Codierungsalternativen
 - Tabelle mit Attribut-Wert-Paaren
 - jede Codierung wird mit Standardattributen assoziiert
 - einige Attribute können Alternativen beinhalten
 - Hierarchische Beschreibung ermöglicht geschachtelte Auswahl aus Alternativen
 - Weltbild notwendig

Beispiel: JetSend E-Material

<i>Level</i>	<i>Attribute</i>	<i>Value</i>	
	vEncoding	vImage	
vImage	vSize	(576000,756000)	
vImage	vOpacity	255	
vImage	vColorSpace	vSRGB vGray	Alternativen
vImage.vSRGB	vPixelDepth	24	
vImage.vSRGB.24	vResolution	(150,150)	
vImage.vSRGB.24.(150,150)	vPixels	(1200,1575)	
vImage.vSRGB.24.(150,150)	vCompression	vJPEG	
vImage.vGray	vPixelDepth	1	
vImage.vGray.1	vResolution	(300,300)	
vImage.vGray.1.(300,300)	vPixels	(2400,3150)	
vImage.vGray.1.(300,300)	vCompression	vRLE	abhängig von Auswahl

Level: Ebene, auf die sich Attribut-Wert-Paar beziehen

JetSend Architektur



JetSend Zusammenfassung

Informationsaustausch garantiert

- alle JetSend-Geräte unterstützen zumindest eine Default-Codierung ihrer Geräteklasse (Image, Video, Audio, ...)

Interoperabilität

- Peer-to-Peer Informationsaustausch zwischen Geräten ohne vorherige Kenntnis voneinander

Modularität

- Separate Entwicklung von Systemkomponenten (Hi-fi Modell)
- Protokoll und Gerätefunktionalität sind getrennt
 - Protokoll definiert „Rohr“ durch das Information fließt
 - Protokoll beeinflusst Funktionalität nicht

Vernetzung: höhere Schichten

- Einführung
- Kommunikationsparadigmen
- Jini Service Infrastruktur
- JetSend Ad hoc Kommunikation
- OBEX Object Exchange Protokoll
- **HAVi AV Kontrolle**
- UPnP Plug&Play

HAVi

Eigenschaften HAVi

- Ein Medium für Kontrolle und Daten
- Verbindungslose Kommunikation
- Geräte = Objekte, Methoden = Funktionen, „RFC“ über Pakete
- IAVs (Intermediate AV devices) (native implementierung)
- FAVs (Full AV devices): Java Runtime
- BAV (Base Audio/Video Devices): nur bytecode upload
- LAVs (Legacy AV devices): Aufrufe müssen von FAV umgesetzt werden
- Device Control Module (DCM) and Functional Component Module (FCM) austauschbar, da in Java
- Java AWT 1.1 und spezielle Klassen, UI Programmierung (Havlets)

HAVi FCM

Functional component Modules (FCM)

- Vordefinition von APIs zu FCM in HAVi Spezifikation
- FCMs setzen abstrakte Info in gerätespezifische Info/Kommando um
- Von dort wird das Kommando an Hardware weitergesendet.
- FCM Beispiele:
 - Tuner FCM: Setzen und Erhalten von Kanälen, Auswahl von Attributen (Musiksender...)
 - VCR FCM: PLAY, REC, REW..., Uhrzeit setzen

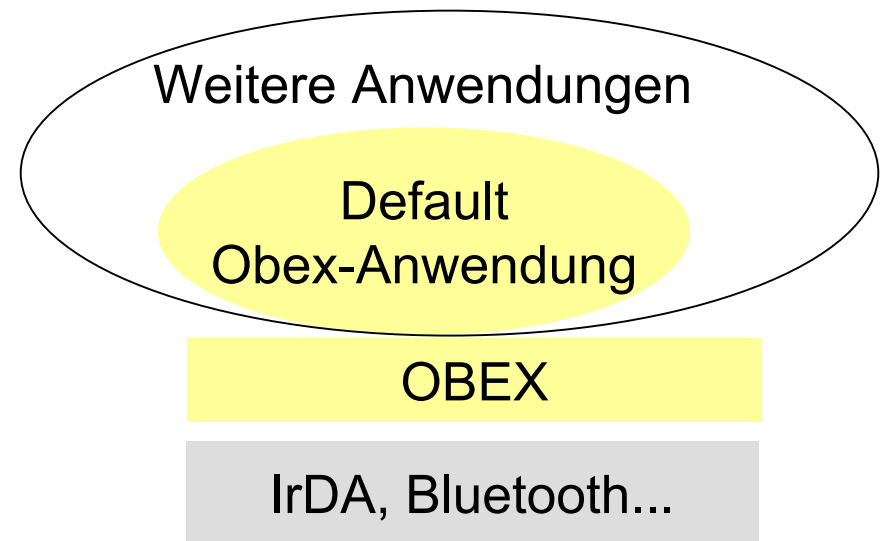
Vernetzung: höhere Schichten

- Einführung
- Kommunikationsparadigmen
- Jini Service Infrastruktur
- JetSend Ad hoc Kommunikation
- **OBEX Object Exchange Protokoll**
- HAVi AV Kontrolle
- UPnP Plug&Play

(Ir)OBEX: IrDA Object Exchange

Beliebige „Dinge“ austauschen

- Protokoll für den Austausch, das Abfragen, die Anforderung von Objekten und den damit verbundenen Diensten
- Dienste primär Datenübertragungsdienste
- Spezifikation besteht aus zwei Teilen:
 - Beschreibung der Objekte
 - Kommunikationsprotokoll
- Einfaches Protokoll



(Ir)OBEX

Ablauf Kommunikation

- Geräte bieten Dienste in Form von Objekten an
- Client erfragt einen Dienst (request) und erhält vom Server eine Antwort (response)
- Sitzungsorientiertes Protokoll
- Bsp: Client erfragt, ob er eine vCard ablegen darf
- Paket: Opcode len Objekte
- Objektmodell definiert Objektbezeichner und Objekte
- Modell besteht aus einer Liste Tupeln der Form <Bezeichner&Format> <Objekt>
- Tupel sind einfach zu parsen
- „Byte“-Kodierung statt Text-Kodierung orientiert sich an leistungsschwachen Geräten

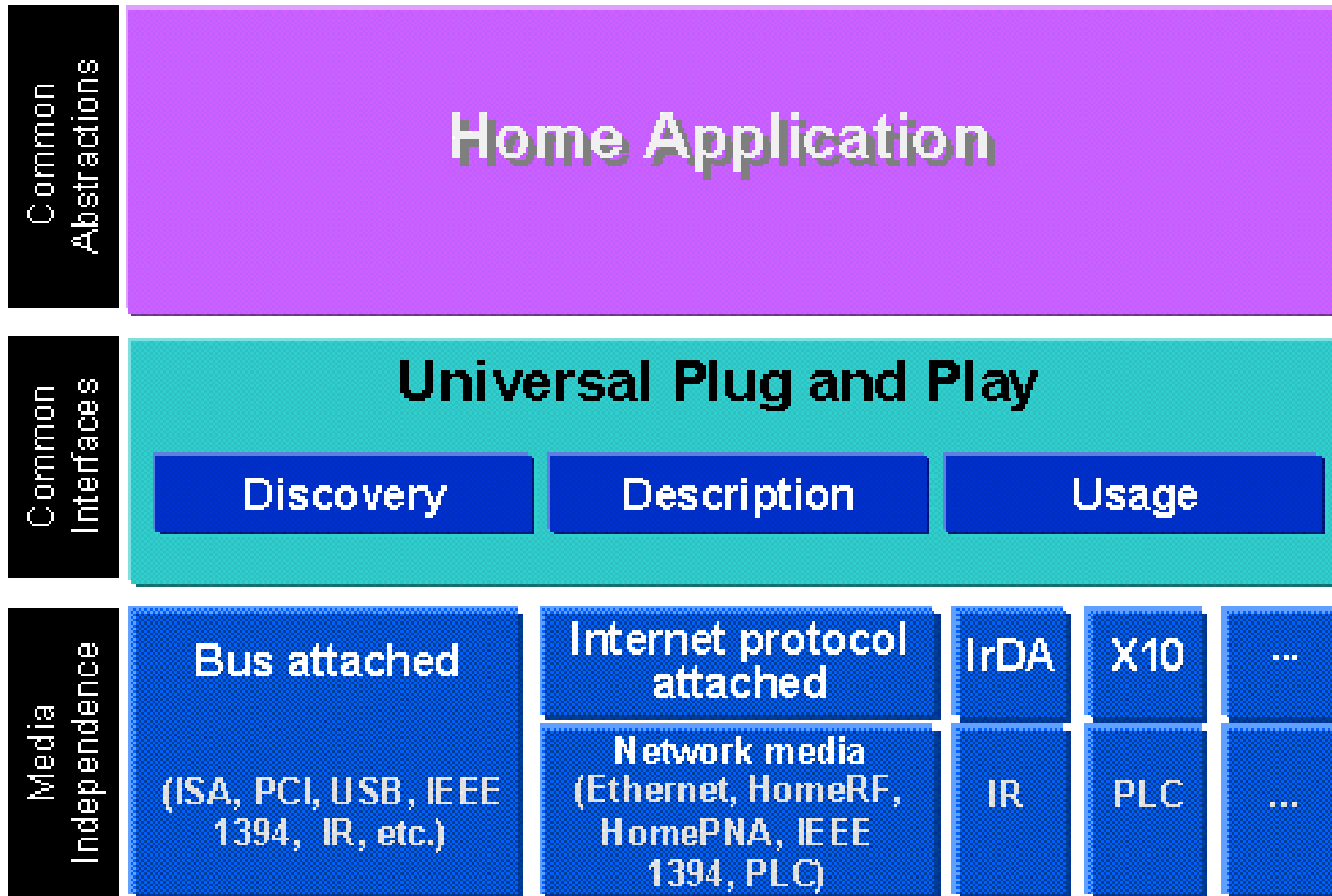
OBEX Kommunikationsbeispiel

Client Request:	bytes	Meaning
opcode	0x80	CONNECT, Final bit set
	0x0007	7 bytes is length of packet
	0x10	version 1.0 of OBEX
	0x00	no connect flags
	0x2000	8K max packet size
Server Response:		
response code	0xA0	SUCCESS, Final bit set
	0x0007	packet length of 7
	0x10	version 1.0 of OBEX
	0x00	no connect flags
	0x0800	2K max packet size
Client Request:		
opcode	0x02	PUT, Final bit not set
	0x0422	1058 bytes is length of packet
	0x01	HI for Name
	0x0017	Length of Name header
	THING.DOC	name of object, null terminated
	0xC3	HI for Length
	0x00006000	Length of object is 0x6000 bytes
	0x48	HI for Object Body chunk
	0x0403	Length of Body header (1K) plus HI and header length
	0x.....	1K bytes of body

Vernetzung: höhere Schichten

- Einführung
- Kommunikationsparadigmen
- Jini Service Infrastruktur
- JetSend Ad hoc Kommunikation
- OBEX Object Exchange Protokoll
- HAVi AV Kontrolle
- **UPnP Plug&Play**

Universal Plug and Play (UPnP)



UPnP

Discovery von Geräten

- SSDP (Simple Service Discovery Protocol): Meldung der Anwesenheit
- LAN Broadcasts oder direktes Ansprechen eines Verzeichnisdienstes (Proxy)
- HTTP/XML basierte verbindungslose Kommunikation z.B. als UDP
- Melden der eigenen Fähigkeiten durch ANNOUNCE-Kommando, enthält zudem URL zu XML Beschreibungsdatei
- Abfrage durch OPTIONS Kommando

Zudem: Broadcast-DNS Abfrage, DHCP Erweiterung