Steuerung eines Roboters über unzuverlässige WLAN Verbindungen

Andreas Jabs
Lehrstuhl für Betriebssysteme der RWTH Aachen

PEARL 2005
Echtzeitaspekte bei der Koordinierung autonomer Systeme

LEHRSTUHL FÜR BETRIEBSSYSTEME
Univ.-Prof. Dr. habil. Thomas Bemmerl

RWTH AACHEN UNIVERSITY

Überblick

- Motivation
- Java Micro Edition
- Unzuverlässige Netze und Verbindungssicherheit
- Die Steuerungsplattform
- Die Robotersteuerung
- Zusammenfassung

Motivation

Ziele:

- Datenauslesen und Steuerung von eingebetteten und mobilen Systemen
- Definition von weichen Echtzeitbedingungen für die Steuerung
- Zugriff auf komplexe Steuerungen von einem PDA oder ähnlichem Gerät aus

Bedingungen:

- Möglichkeit der Autorisierung von Nutzern
- Kein Kontrollverlust über gesteuertes System
- Kontrolle der Verbindungsqualität
- Lauffähig auf Systemen mit geringen Ressourcen
- Hohe Portabilität

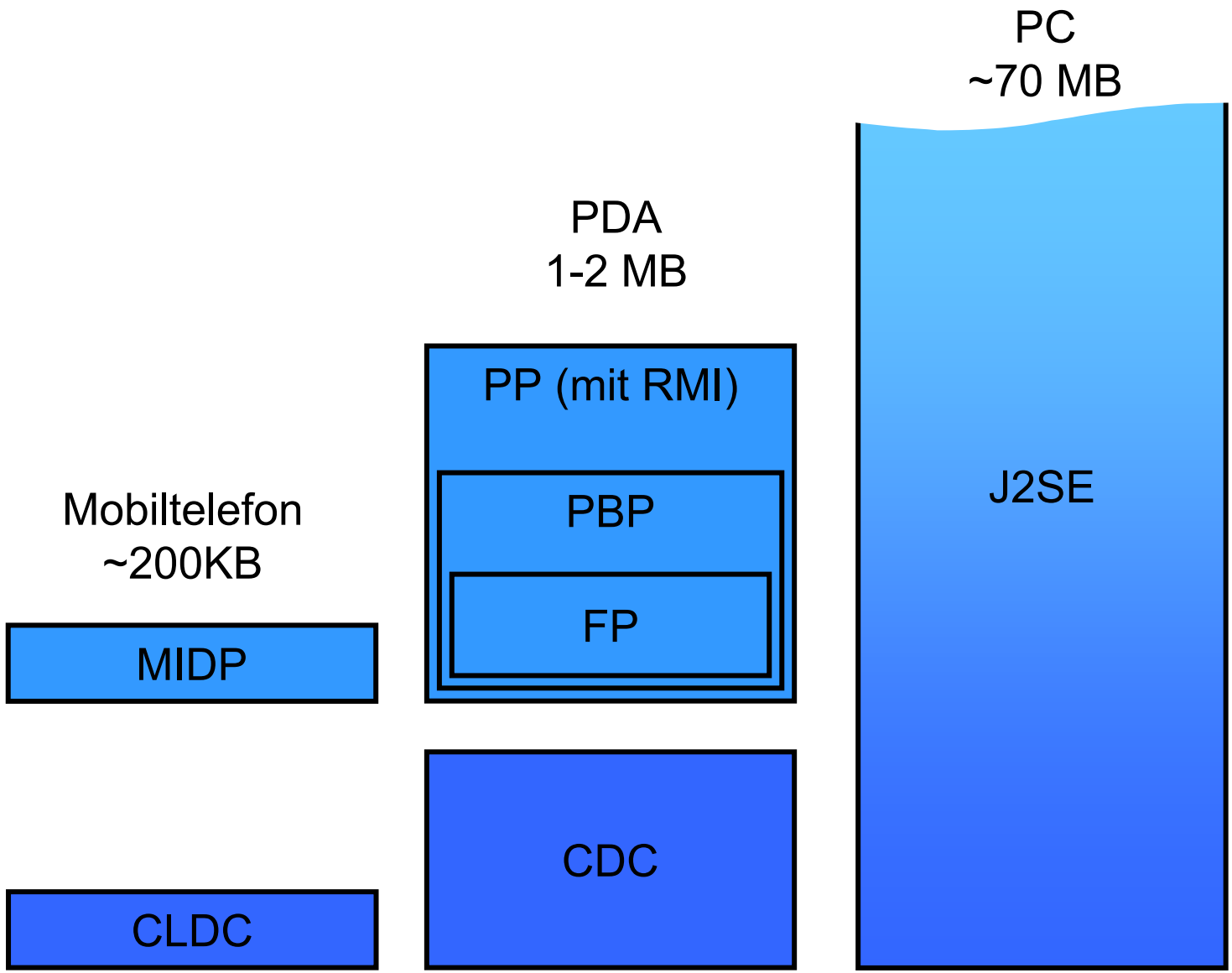
Java 2 Micro Edition

Java 2 Micro Edition

- Neuimplementierung der Java Virtual Machine für ressourcenbeschränkte Systeme
- Auslassung von Funktionen zu Gunsten einer straffen Strukturierung und effizienter Implementierung
- Modularität
- Implementierung der Virtual Machine in Soft- und Hardware erhältlich

- Leistungsfähigkeit gering, dafür hohe Portierbarkeit und geringere Entwicklungskosten, Applets in Browser einbettbar
- Hardwarezugriff über native Routinen bei reduzierter Portierbarkeit
- Echtzeitfähigkeit nicht gegeben, speziell nicht für verteilte Systeme

J2ME Überblick



Unzuverlässige Netze

- Zuverlässigkeit impliziert meist hohen Preis
- Viele Systeme besitzen günstige, aber unzuverlässige Netzanbindungen. (Ethernet, WLAN, Bluetooth)
- Betriebssysteme oft nicht echtzeitfähig

Unzuverlässigkeit hinsichtlich:

- Übertragungszeit (ping)
- Paketverlust
- Reihenfolge der Nachrichten
- Bandbreite

Mobile Geräte verfügen nur über unzuverlässige Netzanbindungen

Die Steuerungsplattform

Funktionalität:

- Datenübermittlung auf Push- und Pull-Basis
- Befehlsübermittlung
- Verbindungskontrolle

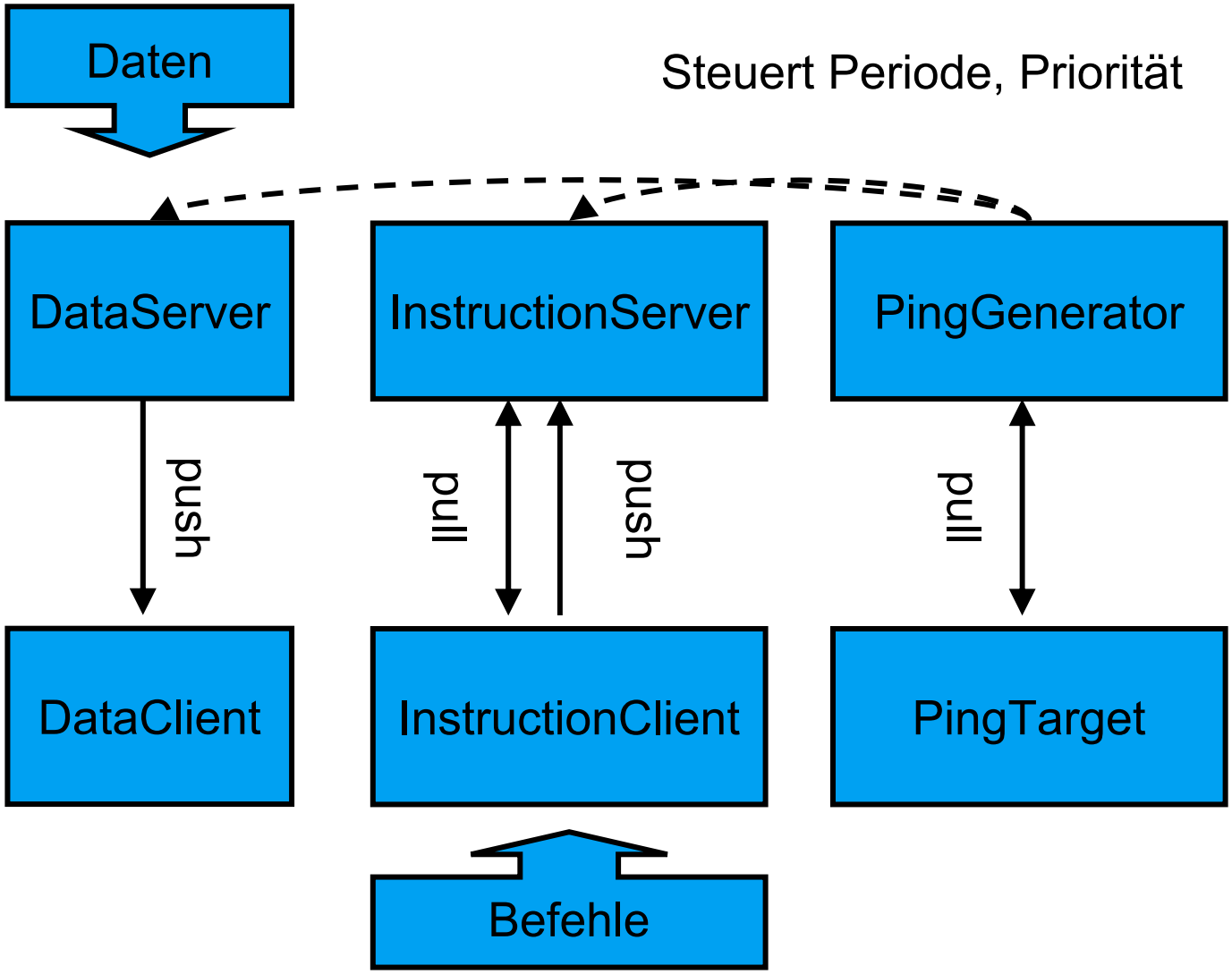
Pull-Modell (Anfrage)

- Server sendet Daten auf Anforderung von Clients, geeignet für seltene Aufgaben wie Datensammlungen

Push-Modell (Abonnement)

- Server sendet Daten unaufgefordert an abonnierte Clients, Antwort der Clients optional
- Push-Modell ermöglicht bessere Kontrolle des Servers über Netzwerkverkehr, besseres Zeitverhalten bei periodischen Aufgaben

Kommunikationsmodell



Datenübermittlung

- Abstraktionsschicht verbirgt das verwendete Protokoll und die Sockets
- TCP- und UDP-Sockets werden durch Java unterstützt, andere bedürfen nativer Bibliotheken
- Datenquellen implementieren nur eine Schnittstelle mit der Methode

```
Object getClientData();
```

- Verwendung einer oder mehrerer getrennter Verbindungen für einzelne Datendienste möglich
- Verbindungskontrolle nutzt getrennte Verbindung, um Prioritätsinversion zu vermeiden

Verbindungskontrolle

Prinzip:

Server fordert Antwort auf ein Signal innerhalb einer Frist

Antwortzeit bestimmt durch

- Netzverbindung
- Rechenlast auf Server
- Rechenlast auf Client

- Kategorisierung der Clients in Klassen hinsichtlich Antwortzeit, Abweisen von ungenügend angebundenen Clients
- Zur Steuerung autorisierter Client erhält höchste Priorität

Die Robotersteuerung

Roboterdaten

- Endogene Datenquellen (Lage, Geschwindigkeit, …)
- Exogene Datenquellen (Sonarwerte, Kamerabild)

Steuerung

- Verwaltungsebene (login)
- Bewegungssteuerungsebene

Verbindungskontrolle

- Messung der Verbindungsqualität
- Verwaltung der Daten- und Steuerverbindungen

Zusammenfassung

Die erstellte Plattform ermöglicht

- Auslesen und Steuern von eingebetteten und mobilen Systemen
- Kontrolle der Verbindungsqualität, adaptive Anpassung der Übertragungsrate
- Lauffähigkeit auf allen Java Plattformen ab CDC aufwärts
- Steuerung als Applet in Webseiten einbettbar

Ausblick

- Unterstützung von CLDC (Mobiltelefone etc.)
- Erzeugung und Verwaltung von Gruppen von Datenquellen