Echtzeitbetriebssysteme

RTLinux/RTAI

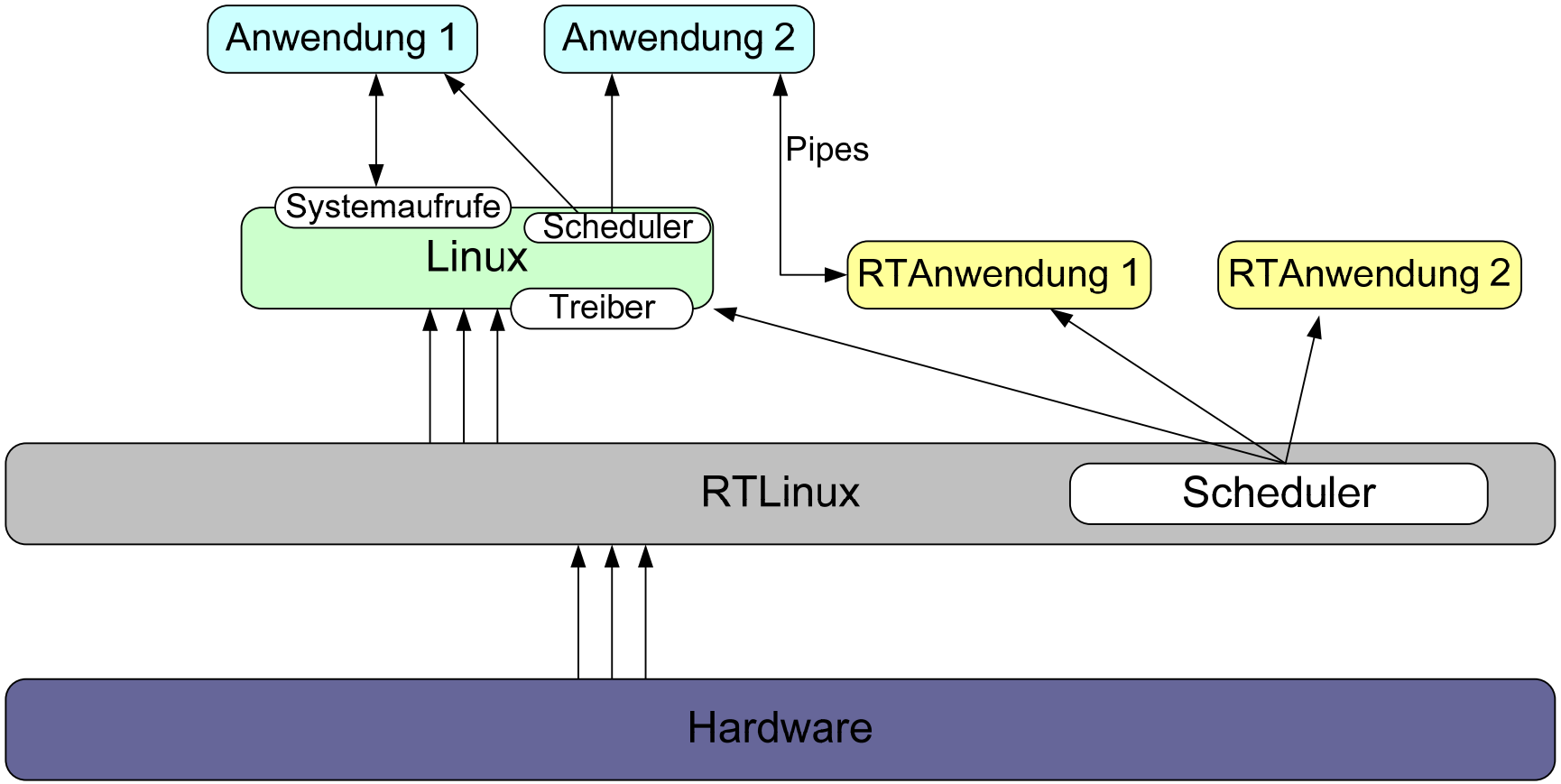
Motivation

- Aus diversen Gründen ist die Verwendung von Linux in Echtzeitsystemen erstrebenswert:
  - Linux ist weitverbreitet
  - Treiber sind sehr schnell verfügbar
  - Es existieren viele Entwicklungswerkzeuge $\Rightarrow$ die Entwickler müssen nicht für ein neues System geschult werden.
  - Häufig müssen nur geringe Teile des Codes echtzeitfähig ausgeführt werden.
- **Probleme**:
  - grobgranulare Synchronisation
  - trotz Patches oft zu lange Latenzzeiten
  - Hochpriorisierte Prozesse können durch andere Prozesse mit niedrigerer Priorität blockiert werden, Grund: Hardwareoptimierungsstrategien (z.B. Speichermanagement)
- **Ansatz**: Modifikation von Linux, so dass auch harte Echtzeitanforderungen erfüllt werden.

Ansatz

- Anstelle von Patches wird eine neue Schicht zwischen Hardware und Linux-Kernel eingefügt:
  - Volle Kontrolle der Schicht über Unterbrechungen
  - Virtualisierung von Unterbrechungen (Barabanov, Yodaiken, 1996): Unterbrechungen werden in Nachrichten umgewandelt, die zielgerichtet zugestellt werden.
  - Virtualisierung der Uhr
  - Anbieten von Funktionen zum virtuellen Einschalten und Ausschalten von Unterbrechungen
  - Das Linux-System wird als Prozess mit niedrigster Priorität ausgeführt.

RTLinux Architektur



Unterschiede RTAI/RTLinux

- RTLinux verändert Linux-Kernel-Methoden für den Echtzeiteingriff ⇒ Kernel-Versions-Änderungen haben große Auswirkungen.
- RTAI fügt Hardware Abstraction Layer (HAL) zwischen Hardware und Kernel ein. Hierzu sind nur ca. 20 Zeilen Code am Originalkern zu ändern. HAL selbst umfasst kaum mehr als 50 Zeilen ⇒ Transparenz.
- RTAI ist frei, RTLinux in freier (Privat, Ausbildung) und kommerzieller Version.
- Beide Ansätze verwenden ladbare Kernel Module für Echtzeitprozesse.
- RTAI (mit Variante LXRT) erlaubt auch die Ausführung von echtzeitkritischen Prozessen im User-Space, Vorteil ist beispielsweise der Speicherschutz

Echtzeitbetriebssysteme

Windows CE & Windows Embedded

Eigenschaften

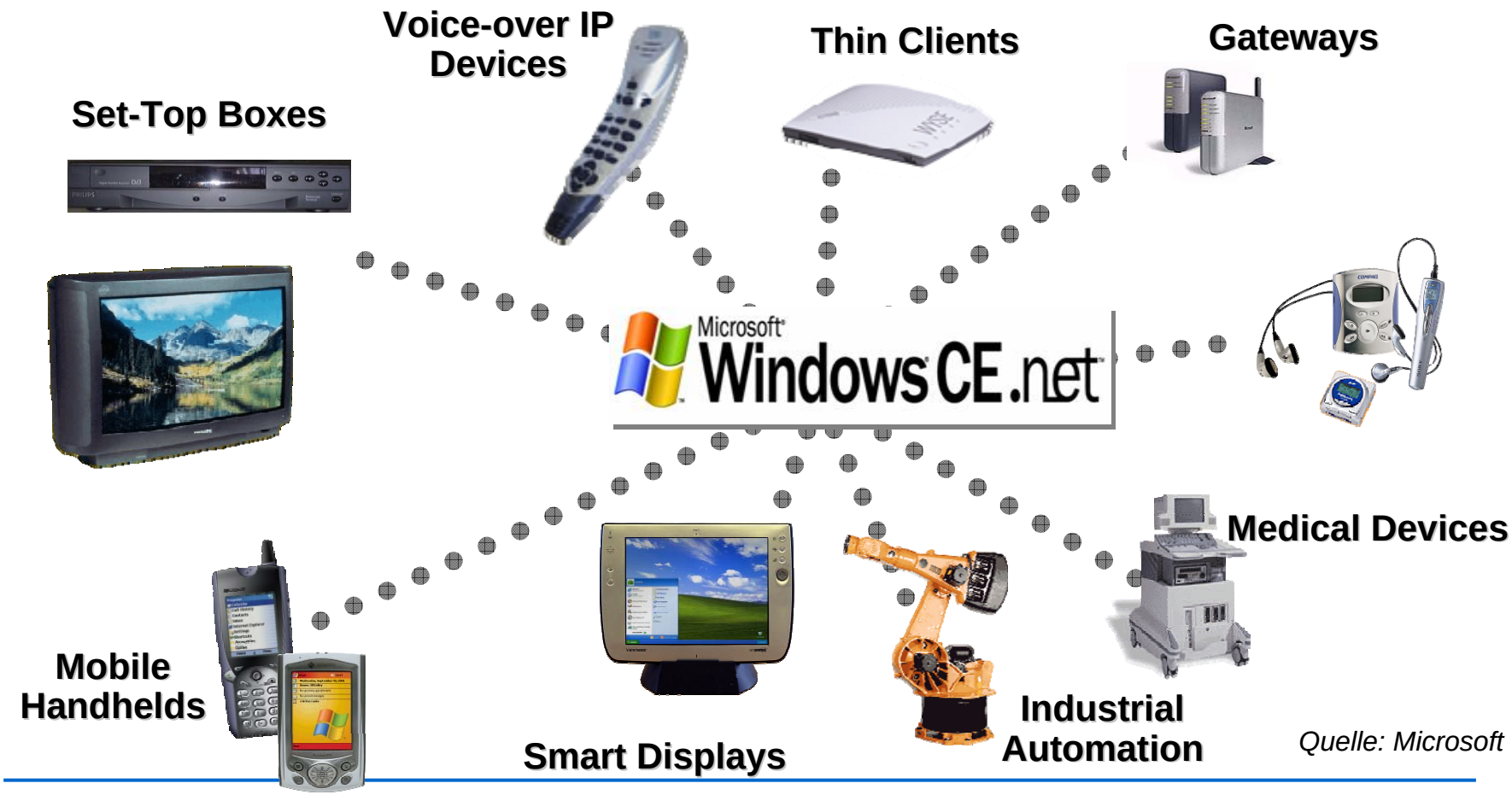
- Windows CE
  - 32-bit, Echtzeitbetriebssystem
  - Unterstützung von Multitasking
  - Stark modularer Aufbau
  - Skalierbar entsprechend der gewünschten Funktionalität
- Windows Embedded
  - „Skalierbares Windows XP“
  - Komponenten von XP können entfernt werden um den benötigten Speicherplatz zu minimieren

Windows CE und Embedded im Vergleich

x86 processors	**Processor Support**	Multiple processors / power management
Full Win32 API compatibility	**Win32 API Compatibility**	Requires additional effort
Basic images from 8MB (“Hello World”)	**Footprint**	Basic images from 350 KB
With 3rd party extensions	**Real-time**	Native

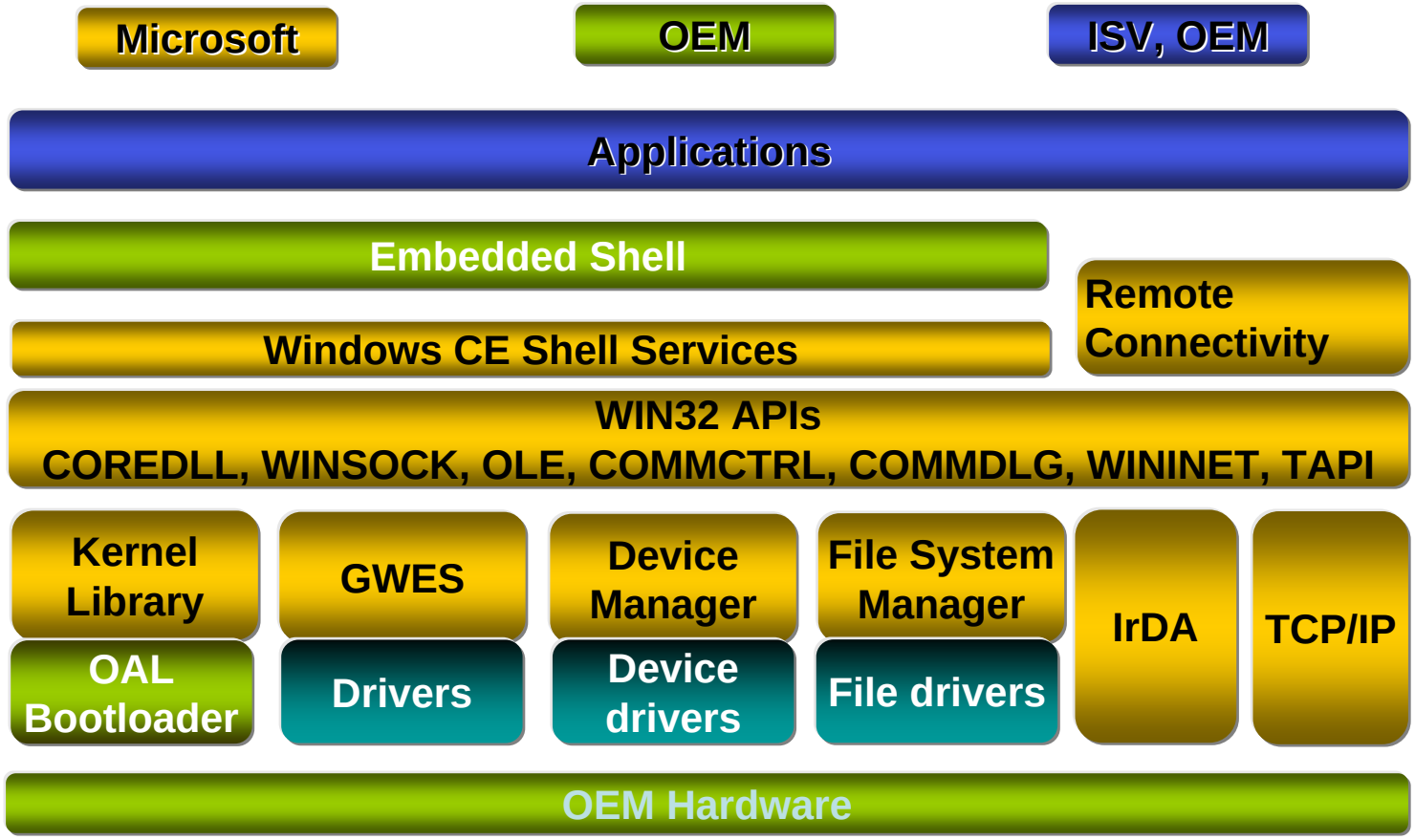
Quelle: Microsoft

Einsatzbereiche



Quelle: Microsoft

Windows CE Architektur



Quelle: Microsoft

Funktionen des Betriebssystemkerns

- Kernel, Speicherverwaltung
  - Shared heap
  - Unterstützung von Watchdogs
  - 64 Systeminterrupts
- Geräteunterstützung
  - Unterstützung diverser Massenspeicher, z.B. USB, Flash,..
- Browser
- Multimedia
  - Diverse Graphiktreiber
  - umfassende Codecunterstützung
- Kryptographie-Funktionen

Echtzeitunterstützung

- Unterstützung verschachtelter Interrupts
- 256 Prioritätslevel
- Thread quantum level control
- Speicherschutz (Pinning) zur Umgehung von Virtual Memory
- Eingebaute Leistungsüberwachungswerkzeuge
- Niedrige ISR/IST Latenz
  - ISR/IST Latenz von 2.8/26.4 Mikrosekunden auf Intel 100MHz Board

Echtzeitbetriebssysteme

Zusammenfassung

Zusammenfassung

- Es gibt kein typisches Echtzeitbetriebssystem da je nach Einsatzbereich die Anforderungen sehr unterschiedlich sind.
- Der minimale Speicherbedarf reicht von wenigen Kilobyte (TinyOS, QNX) bis hin zu mehreren Megabyte (Windows CE / XP Embedded).
- Die Betriebssysteme sind typischerweise skalierbar. Zur Änderung des Leistungsumfangs von Betriebssystemen muss das System entweder neu kompiliert werden (VxWorks) oder neue Prozesse müssen nachgeladen werden (QNX).
- Die Echtzeitfähigkeit von Standardbetriebssysteme kann durch Erweiterungen erreicht werden (RTLinux/RTAI).
- Die Schedulingverfahren und die IPC-Mechanismen orientieren sich stark an den in POSIX vorgeschlagenen Standards.
- Das Problem der Prioritätsinversion wird zumeist durch Prioritätsvererbung gelöst.

Klausurfragen

- Wiederholungsklausur WS 2006/2007
  - Erläutern Sie die Unterschiede zwischen Betriebssystemen mit kooperativem Scheduling, mit präemptiven Scheduling und präemptiblen Betriebssystemen.
- Klausur WS 2007/2008
  - Erläutern Sie kurz (jeweils 1-2 Sätze) die Hauptkonzepte von TinyOS, QNX und PikeOS.

Klausurfragen - Klausur WS 2006/2007

- Gegeben seien folgende fünf Echtzeitbetriebssysteme:
  1. TinyOS
  2. OsekTime
  3. QNX
  4. VxWorks
  5. WindowsCE

  und folgende fünf Anwendungen:
  a. Sicherheitsüberwachung für Robotersteuerung: auf Basis eines Controllers mit geringen Rechen- und Speicherkapazitäten wird eine Anwendung zur Überwachung der Robotersteuerung implementiert. Dringt eine Person in den Arbeitsbereich des Roboters ein, so wird dieses Ereignis durch Lichtschranken detektiert und der Roboter unverzüglich gestoppt.
  b. ZebraNet: Zur Erforschung der Wanderwege von Zebras, werden kleine Funkmodule ausgestattet mit GPS-Sensoren zur Lokalisation und Solarzellen zur Stromversorgung an Zebras angebracht. Nähern sich zwei Zebras, so tauschen die Funkmodule die gesammelten Daten aus.
  c. Zugsteuerung: Zur Steuerung und Überwachung eines Zuges werden in einem verteilten System verschiedene periodische Regelungsfunktionen ausgeführt. Diese Funktionen werden dabei als Komponenten von unterschiedlichen Entwicklergruppen zur Verfügung gestellt. Insbesondere eine reibungslose Integration dieser Komponenten steht im Vordergrund.
  d. Fabrikanlagensteuerung: In einer Chemiefabrik wird die Produktion von leistungsfähigen Feldrechnern gesteuert. Diese Rechner sind mit einer Vielzahl von Sensorik verbunden. Auf kritische Ereignisse muss schnell reagiert werden.
  e. PDA: Auf einem Handheld werden unterschiedlichste Anwendungen ausgeführt: unter anderem Routenplaner, Browser, Musik-/Videospieler und Programme zur Terminverwaltung.

  Ordnen Sie jedem der fünf Anwendungsgebiete ein Echtzeitbetriebssystem zu und begründen Sie Ihre Zuordnung knapp mit einem Satz.

Kapitel 8 (vorgezogen)

Echtzeitfähige Kommunikation

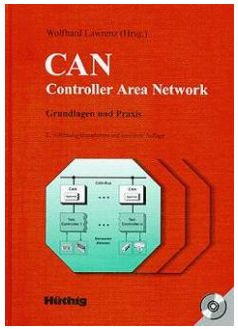
Inhalt

- Grundlagen
- Medienzugriffsverfahren und Vertreter
  - CSMA-CD: Ethernet
  - CSMA-CA: CAN-Bus
  - Tokenbasierte Protokolle: Token Ring, FDDI
  - Zeitgesteuerte Protokolle: TTP
  - Real-Time Ethernet

Literatur

Andrew S. Tanenbaum,
Computernetzwerke, 2005

Wolfhard Lawrenz: CAN Controller Area Network. Grundlagen und Praxis, 2000



- Spezifikationen:
  - TTTech Computertechnik AG, Time Triggered Protocol TTP/C High-Level Specification Document, 2003 (http://www.vmars.tuwien.ac.at/projects/ttp/)
  - http://www.can-cia.org/
  - http://standards.ieee.org/getieee802/portfolio.html