UbiCom Book Slides

Chapter 1

Ubiquitous Computing: Basics and Vision

Name:

Email/Web:

Overview

- **Living in a Digital World √**
- Modelling the Key Ubiquitous Computing Properties
- Ubiquitous System Environment Interaction
- Architectural Design for UbiCom Systems: Smart DEI Model
- Course Outline

Ubiquitous Computing (UbiCom)

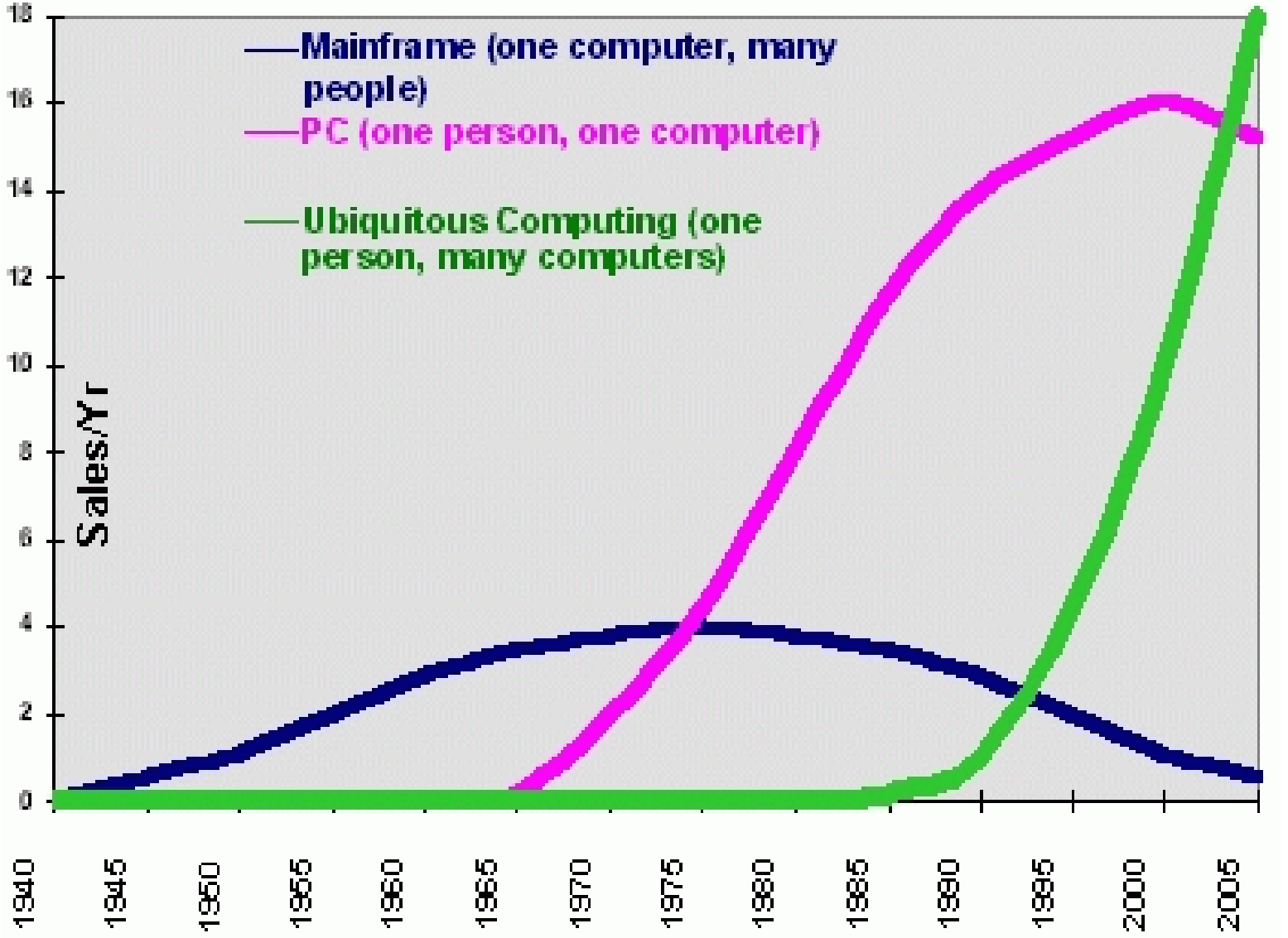
- A vision for computing to:
  - Enable computer-based services to be made available everywhere (Ubiquitous)
  - Support intuitive human usage
  - But yet, appear to be invisible to the user.
  - Also referred to as pervasive computing etc

Living in an Increasingly Digital, Interconnected World

- *What are the current technology trends in UbiCom?*

Trend: smaller, higher resource devices

Trend: Weiser's 3 waves of computing



Living in an Increasingly Digital, Interconnected World

- *What will the future be like?*

Scenarios

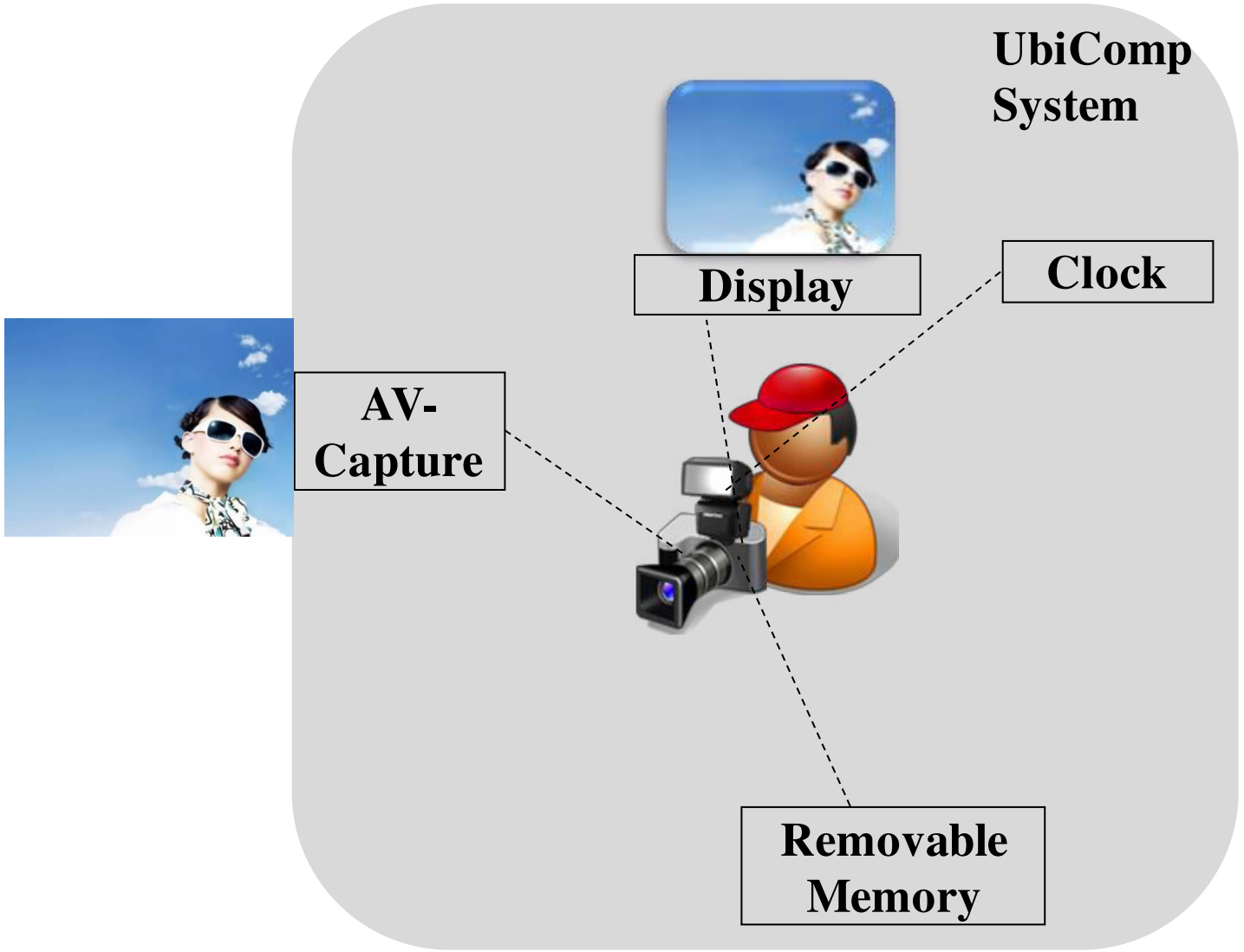
4 scenarios illustrate a range of benefits and challenges of ubiquitous computing:

- Personal memories
- 21st Century Scheduled Transport Service
- Foodstuff management
- Utility regulation

N.B. many other scenarios & applications given later

- e.g., Chapter 2 describes some key projects and gives an overview of applications.

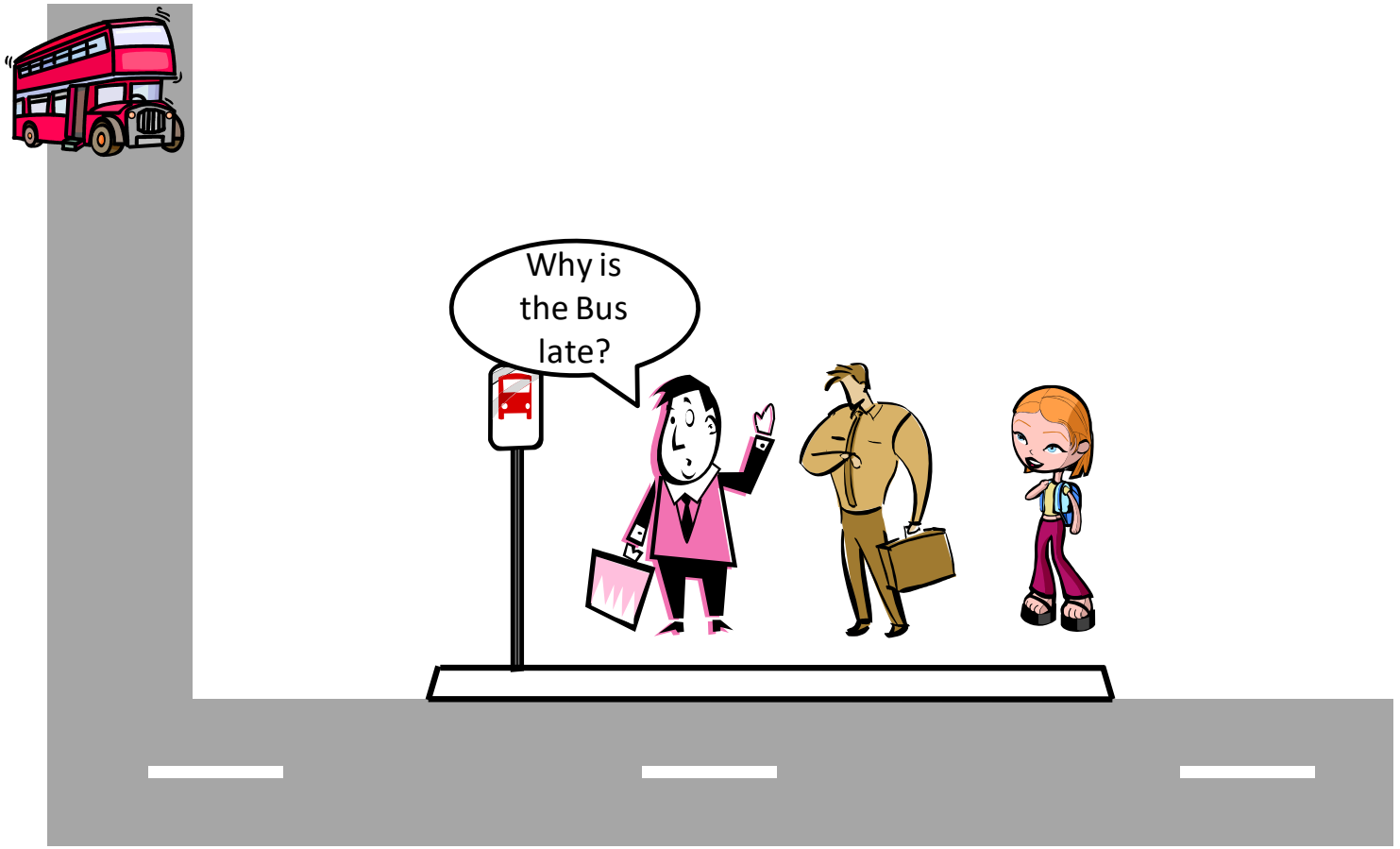
Personal Memories Scenario



Personal Memories Scenario

- How can we enhance the personal memories service using UbiCom?

21st Century Scheduled Transport Service Scenario



21st Century Scheduled Transport Service

How can we enhance the transport service using UbiCom?

Foodstuff Management Scenario

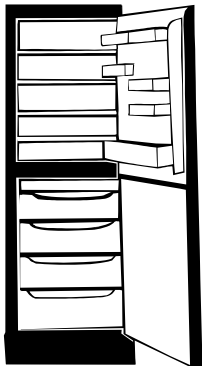
Select & Buy food at physical or virtual market

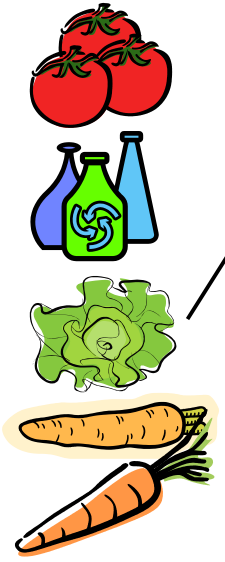
Transport food to home store

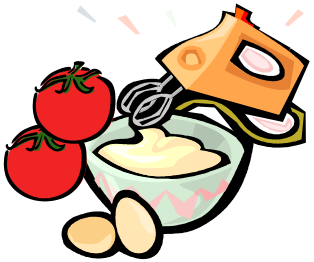
Put in home store

Select food from store, get & transform food into a meal

Consume food



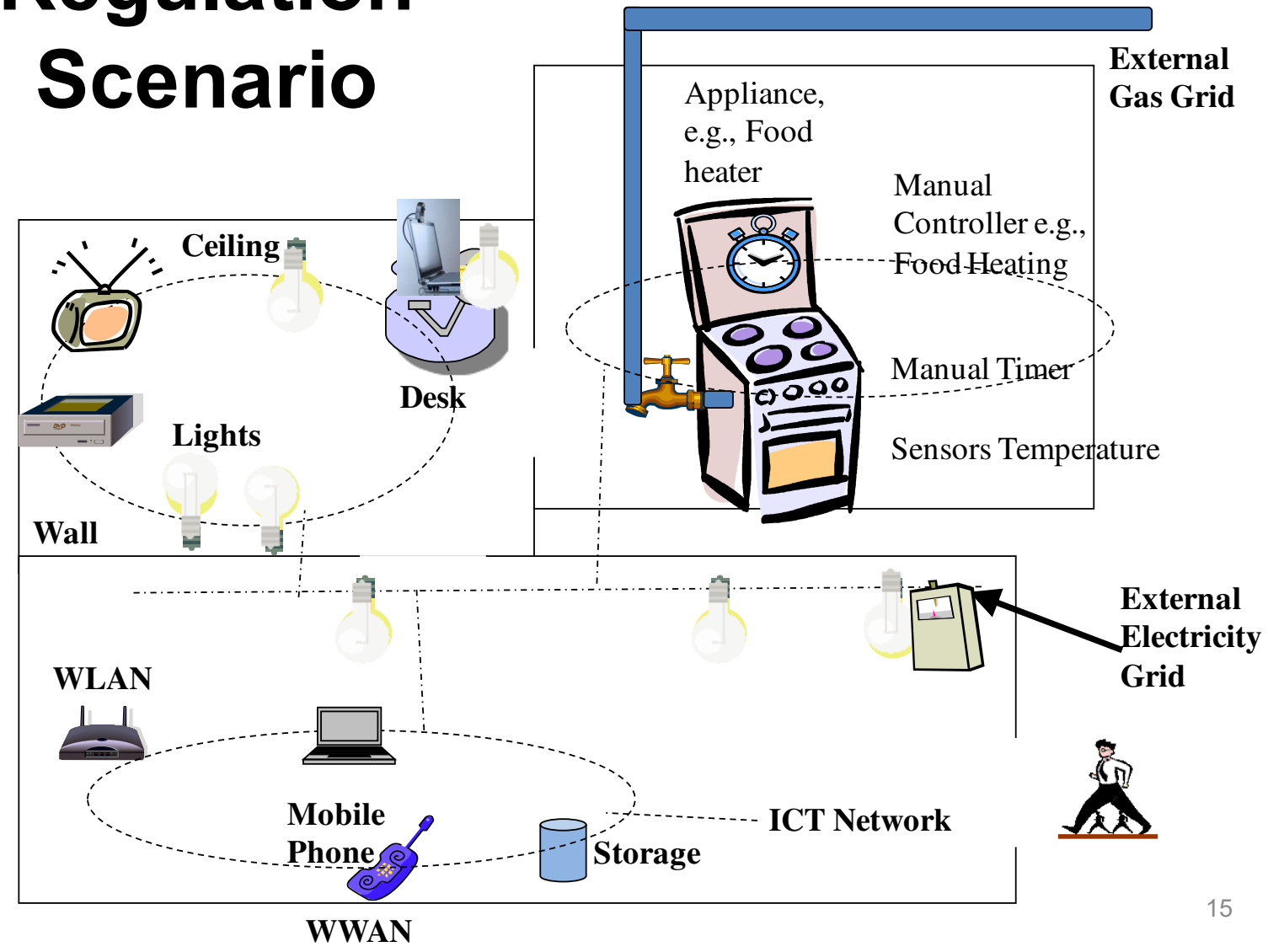




Foodstuff Management Scenario

How can UbiCom enhance the foodstuff management scenario?

Utility Regulation Scenario



Utility Regulation Scenario

- Utility regulation concerns energy, water, waste regulation by end-users.

How can UbiCom enhance the Utility Regulation scenario?

UbiCom System Design

For these scenarios

- Which system designs should be used for:
  - comms., data storage, processing, sensing, control etc
- How to model system - physical world interaction?
- How to model human computer system interaction?
- These are covered later in this course.

Overview

- Living in a Digital World
- **Modelling the Key Ubiquitous Computing Properties √**
- Ubiquitous System Environment Interaction
- Architectural Design for UbiCom Systems: Smart DEI Model
- Course Outline

UbiCom: Different Combinations of Core Properties versus a Single Definition

- No single, absolute definition for ubiquitous computing.
- Instead propose many different kinds of UbiCom based upon combining different sets of core properties
- What core system properties would you propose to define ubiquitous computing?

.

UbiCom: Weiser's 3 Internal System Properties

3 main properties for UbiCom Systems were proposed by Weiser (1991)

1. Computers need to be networked, *distributed* and transparently accessible
   - In1991, little wireless computing, Internet far less pervasive
2. Computer *Interaction* with Humans needs to be more *hidden*
   - Because much HCI is overly intrusive
3. Computers need to be *aware* of *environment context*
   - In order to optimise their operation in their physical & human environment.

Devices: Extended set of Internal System Properties

To which two additional properties are added:

4. Computers can operate *autonomously*, without human intervention, be self-governed
5. Computers can handle a multiplicity of dynamic actions and interactions, governed by intelligent decision-making and intelligent organisational interaction. This entails some form of *artificial intelligence*.

UbiCom: Different Combinations of Core Properties versus a Single Definition

- No single, absolute definition for ubiquitous computing.
- Instead propose many different kinds of UbiCom based upon combining different sets of core properties

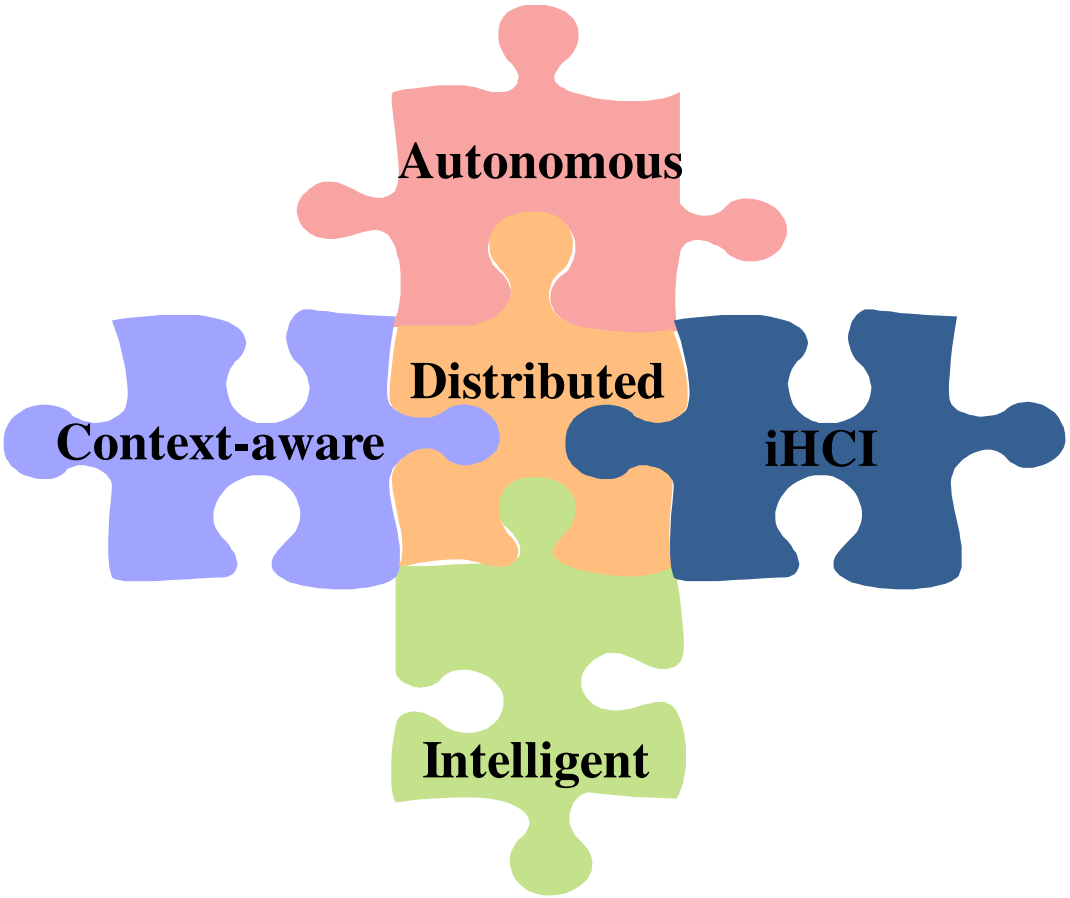
Here are some examples proposed by others

- Weiser (1991): distributed, iHCI, physical environment context aware
- Ambient Intelligence (AmI), similar to UbiCom - intelligence everywhere?
  - Arts and Marzano (2003) define 5 key features for Aml to be embedded, context-aware, personalised, adaptive and anticipatory.
- Buxton (1995): ubiquity and transparency
- Endres et al. (2005): distributed mobile, intelligence, augmented reality
- Millner (2006): autonomy, IHCI
- Etc.

Exercise: Do your own survey of UbiCom definitions and highlight the properties they define.

.

Five main properties for UbiCom



UbiCom System Properties: Distributed

- Networked ICT Devices
- Transparency
- Openness

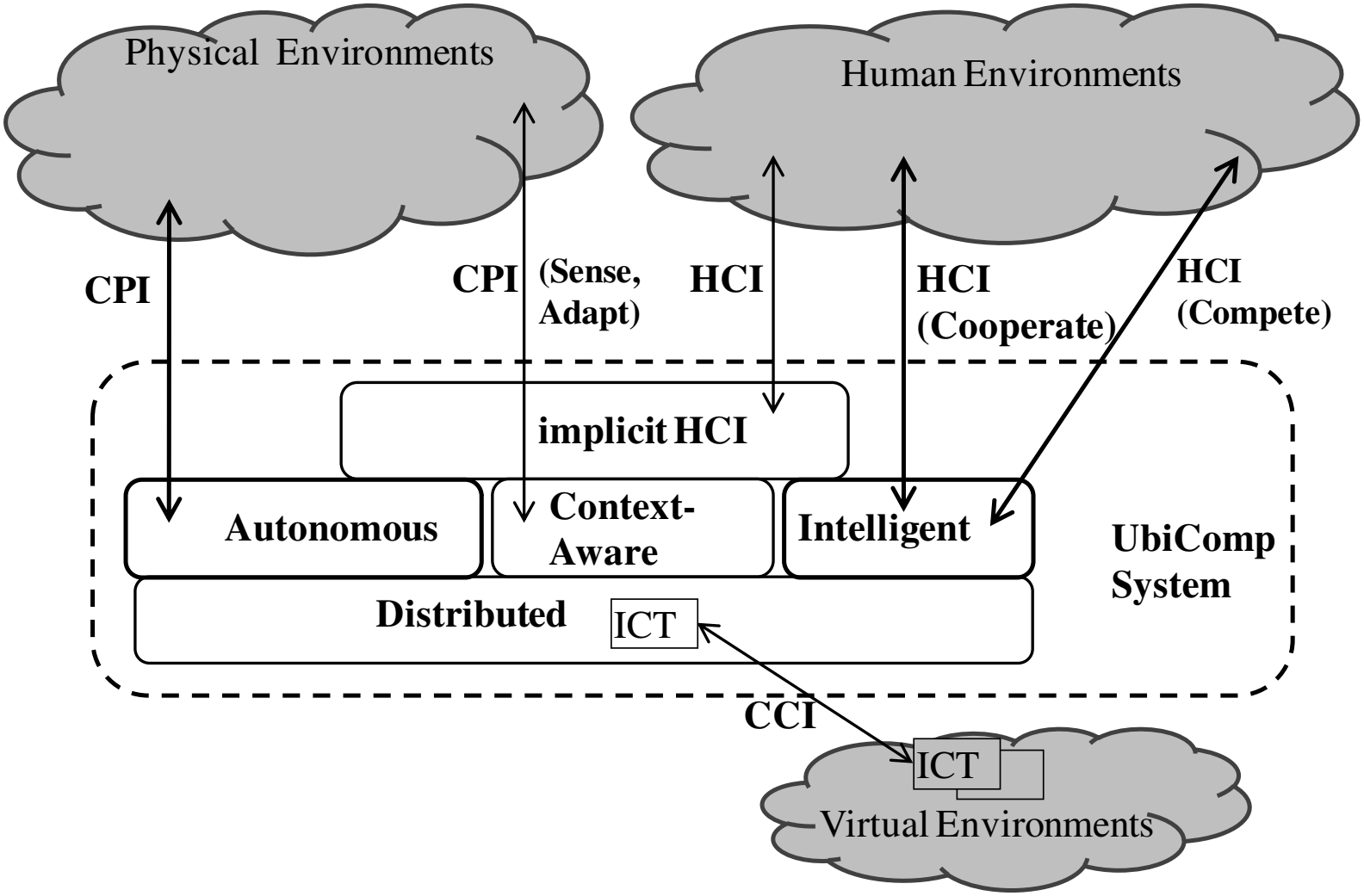
Distributed System: sub-properties

- Often designed middleware, set of generic services
- Universal, Seamless, Heterogeneous
- Networked
- Synchronised, Coordinated
- Open
- Transparent, Virtual
- Mobile, Nomadic
- (See Chapters 3 and 4)

Internal System Properties: iHCI

- Concept of *calm* / *disappearing computer* has several dimensions
- Implicit (iHCI) versus Explicit HCI
- Embodied Virtuality as opposite of VR (people in virtual world)

Devices: Extended set of Internal System Properties



iHCI: Sub-Properties

Implicit Human Device Interaction (iHCI)

- Non-intrusive, Hidden, Invisible, Calm, computing
- Tangible, Natural
- Anticipatory, Speculative, Proactive
- Affective, Emotive
- User-aware
- PostHuman
- Sense of Presence
- Immersed, Virtual , Mediated reality
- (See chapter 5)

Internal System Properties: context-aware

- Often *Context-based ubiquity* rather than global ubiquity
  - Why?

3 Main Types of Context

- *Physical Environment Context*
- *Human Context* (or *User context* or person context)
- *ICT Context* or *Virtual Environment Context*:

Context-aware: sub-properties

- Also referred to as Sentient, Unique, Localised, Situated
- Presupposes sensing of environment to be aware of its context
- Adaptive: active versus passive context-aware

Types of environment aware

- Person-aware, User-aware, Personalised, Tailored,
- Environment-aware, Context-aware, Physical context-aware
- ICT infrastructure aware
- (See Chapter 7)

Internal System Properties: Autonomy

- Challenge 1: increasing computer systems can overload humans – humans become a bottleneck
- What can be done?

Internal System Properties: Autonomy

- Challenge 2: automated system can be become to complex to maintain
  - must reduce maintenance of ↑ complex systems
- What can be done?

Autonomy: Sub-Properties

- Automatic
- Embedded, Encapsulated
- Embodied
- Resource-constrained
- Untethered, Amorphous
- Autonomic, Self-managing, self-star properties
- Emergent, self-organising
- (See Chapter 10)

Internal System Properties: Intelligence

Intelligent UbiCom systems (IS) can:

- Act more proactively, dynamically & humanely through:
- Model how their environment changes when deciding how it acts.
- Goal-based / planning
- Reasoning for re-planning
- Handle uncertainty.
- semantic based interaction
- etc

Individual Intelligence: Sub-Properties

- Referred to as Intelligent Systems, AI, agent-based system etc.

Sub-Properties (sub-types of individual intelligence)

- Reactive, Reflex
- Model-based,
- Rule/Policy-based
- Logic/Reasoning
- Goal-oriented, Planned, Proactive
- Utility-based, Game theoretic
- Learning, Adaptive

(See Chapter 8)

Multiple Intelligence: Sub-Properties

- Referred to as Distributed AI, Multi-Agent Systems, Collective or Social Intelligence

Sub-Properties

- Cooperative , Collaborative, Benevolent
- Competitive, self-interested, antagonistic, adversarial
- Orchestrated, Choreographed, Mediated
- Task-sharing
- Communal, shared meaning
- Shared knowledge
- Speech-act based , Intentional, Mentalistic
- (See Chapter 9)

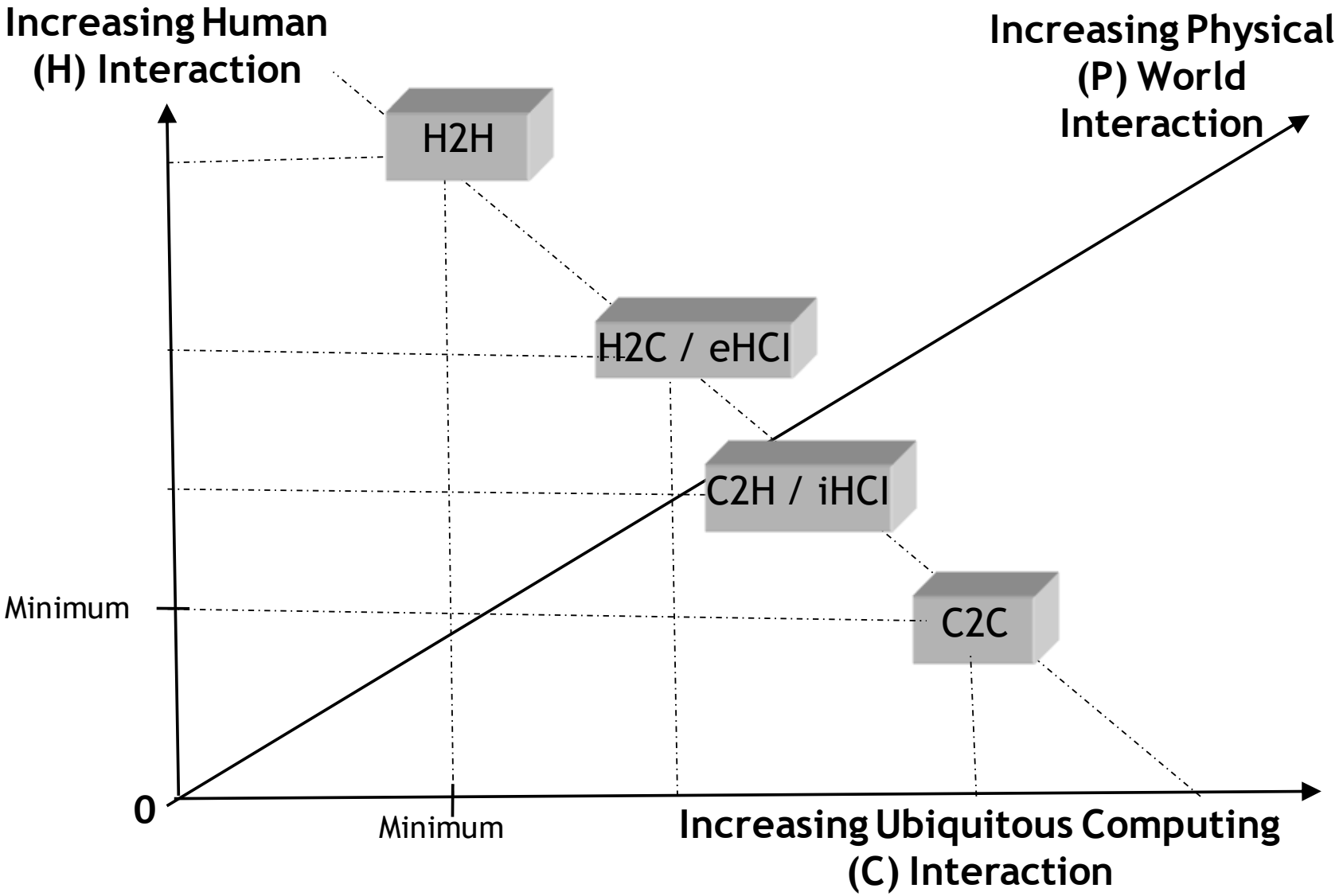
Overview

- Living in a Digital World
- Modelling the Key Ubiquitous Computing Properties
- **Ubiquitous System Environment Interaction √**
- Architectural Design for UbiCom Systems: Smart DEI Model
- Course Outline

UbiCom System versus ICT System focus

- Conventional focus on ICT systems
- Systems situated in a virtual world, in an environment of other ICT systems -> a system of ICT systems
- Conventional use a restrict view of physical environment interaction:
- Conventional ICT systems often require humans in the loop
- UbiCom represents a powerful shift in computation:

Different Degrees of HCI



Different Degrees of HCI

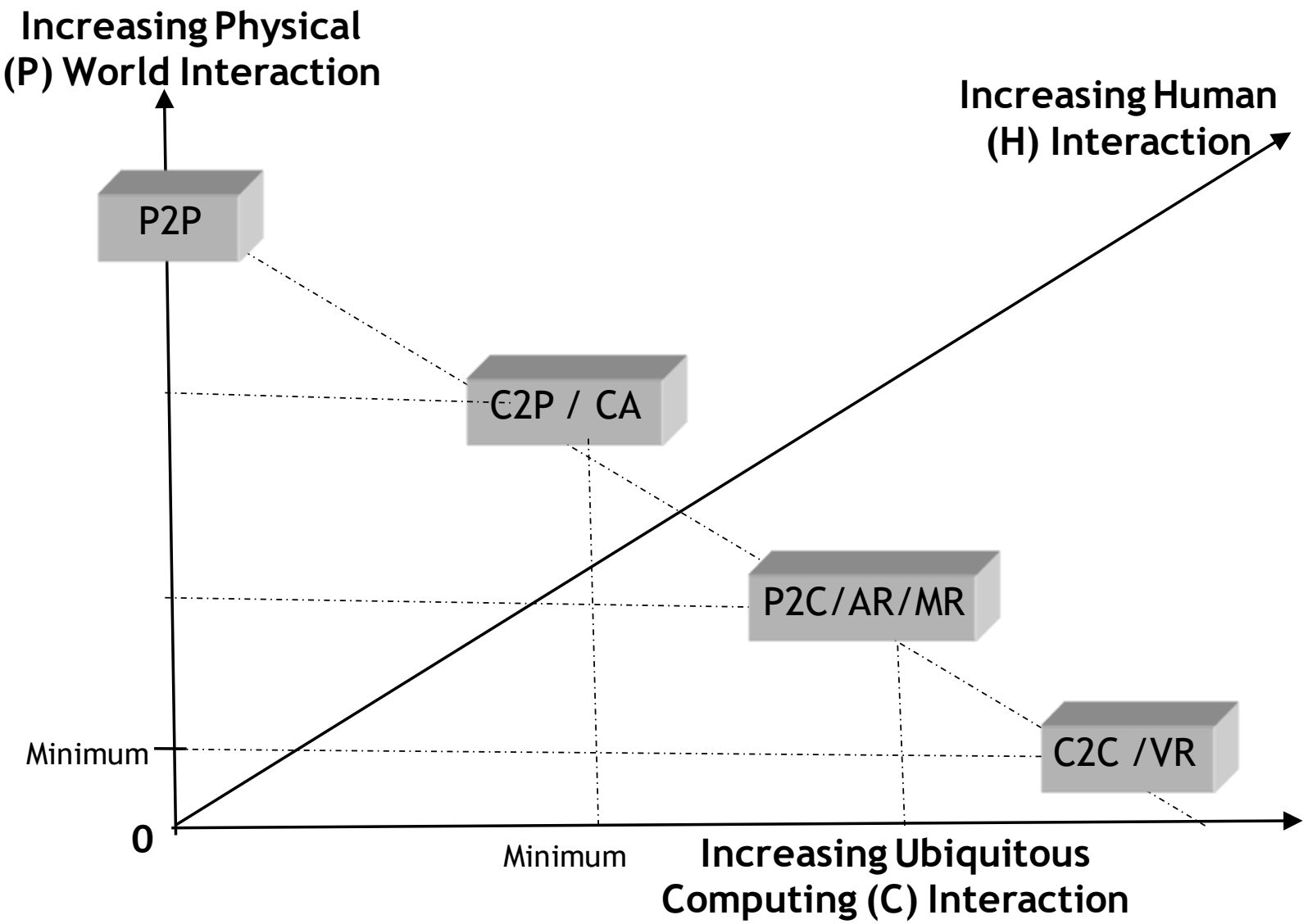
- From less to more C Interaction with H
- H2H: human interaction

- H2C or explicit (e)HCI:

- C2H or implicit (i)HCI:

- C2C:

Different Degrees of CPI



Different Degrees of CPI

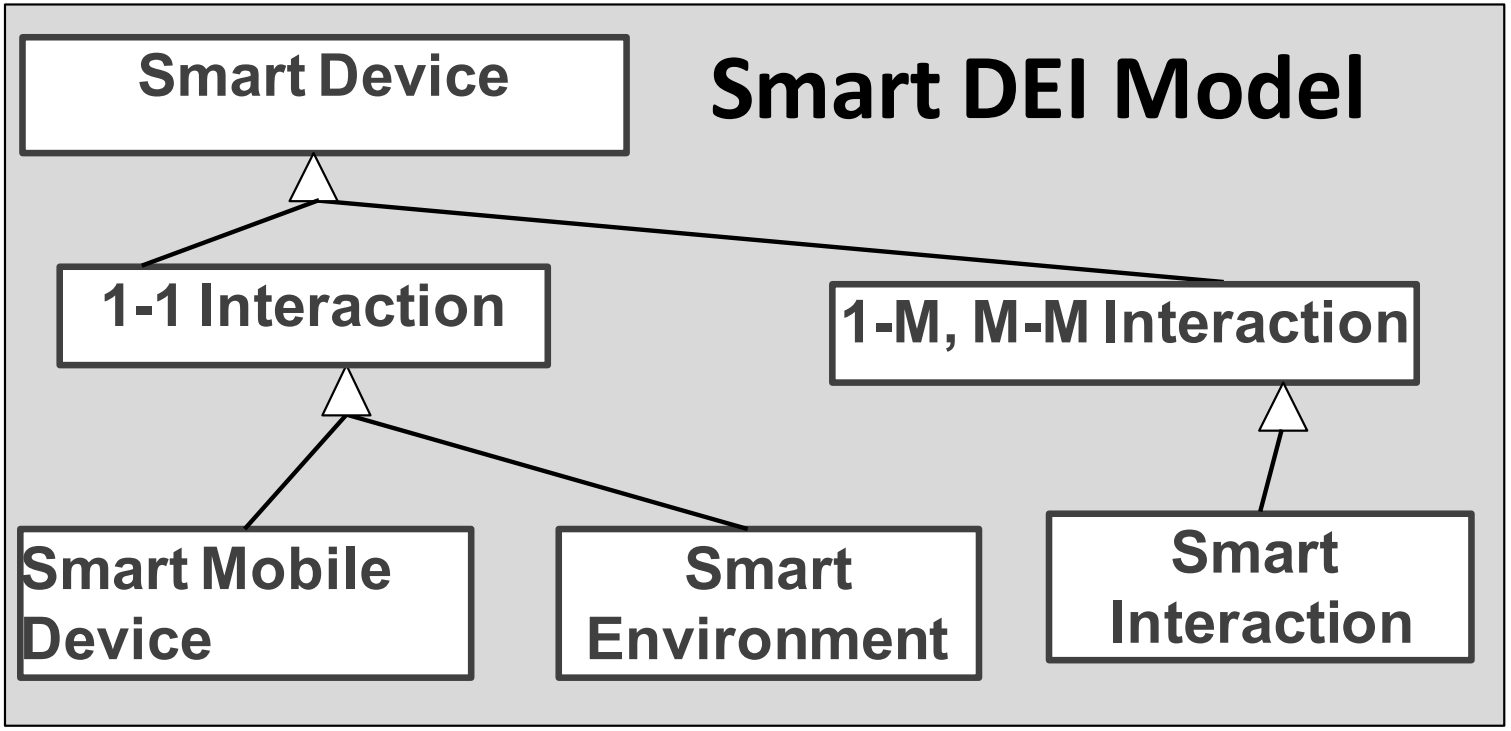
- From less to more C Interaction with P
- P2P
  - Physical interaction (No ICT mediation)
- C2P / CA (Physical Env. Context-aware)
  - C Senses P. C Aware of P's Context
- P2C/AR/MR
  - C augments or mediates P's reality.
  - C actively adapts to P's context
- C2C /VR
  - Virtualisation of reality facilitated by C

Overview

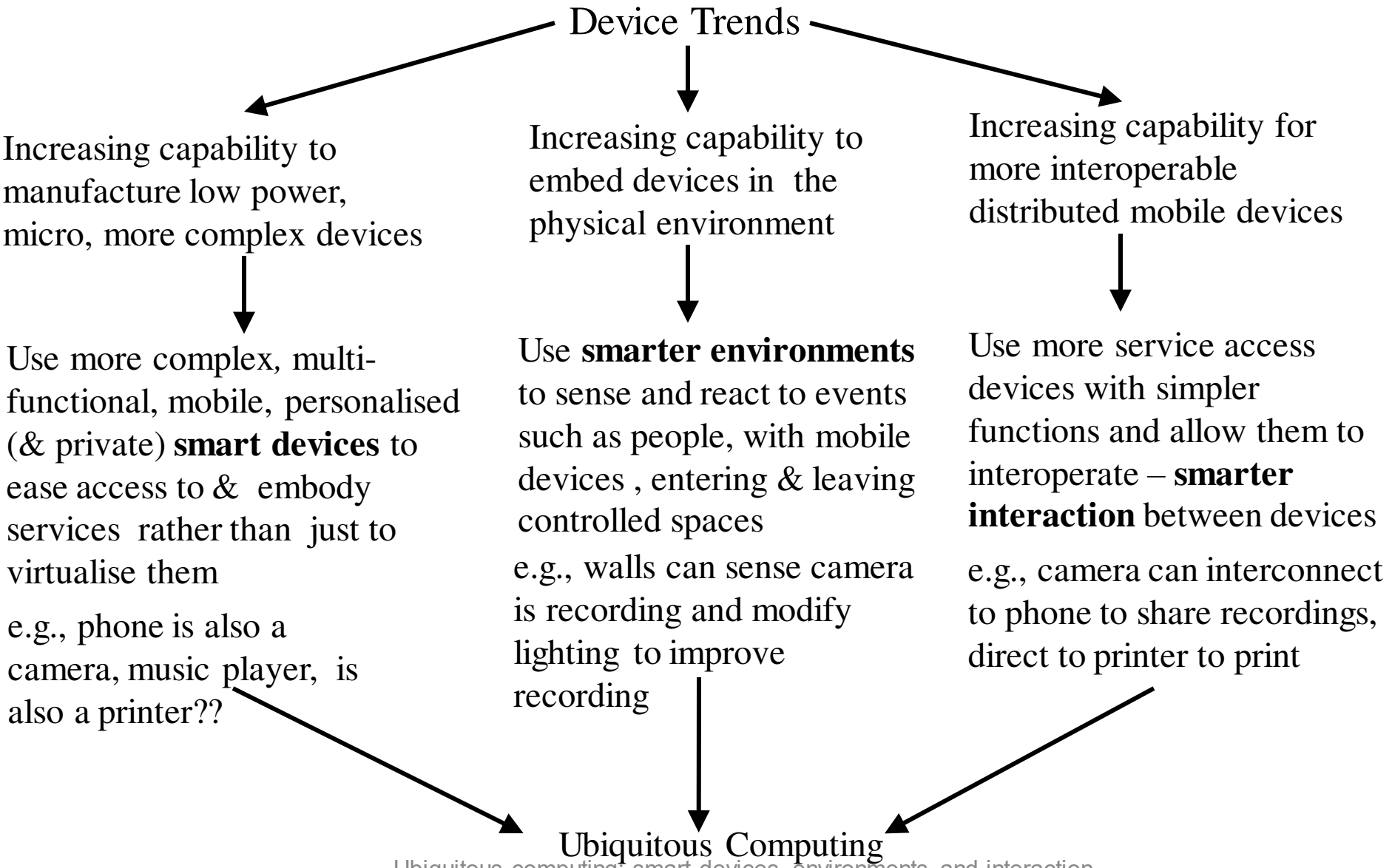
- Living in a Digital World
- Modelling the Key Ubiquitous Computing Properties
- Ubiquitous System Environment Interaction
- **Architectural Design for UbiCom Systems: √**
- Course Outline

UbiCom System Model: Smart DEI

- No single type of UbiCom system
- Different UbiCom systems support:
- 3 basic architectural design patterns for UbiCom:
  - smart **D**evices, smart **E**nvironments, smart **I**nteraction
- 'Smart' means systems are:
  - active, digital, networked, autonomous, reconfigurable, local control of its own resources, e.g., energy, data storage etc.


Smart Device
Smart DEI Model
1-1 Interaction
1-M, M-M Interaction
Smart Mobile Device
Smart Environment
Smart Interaction

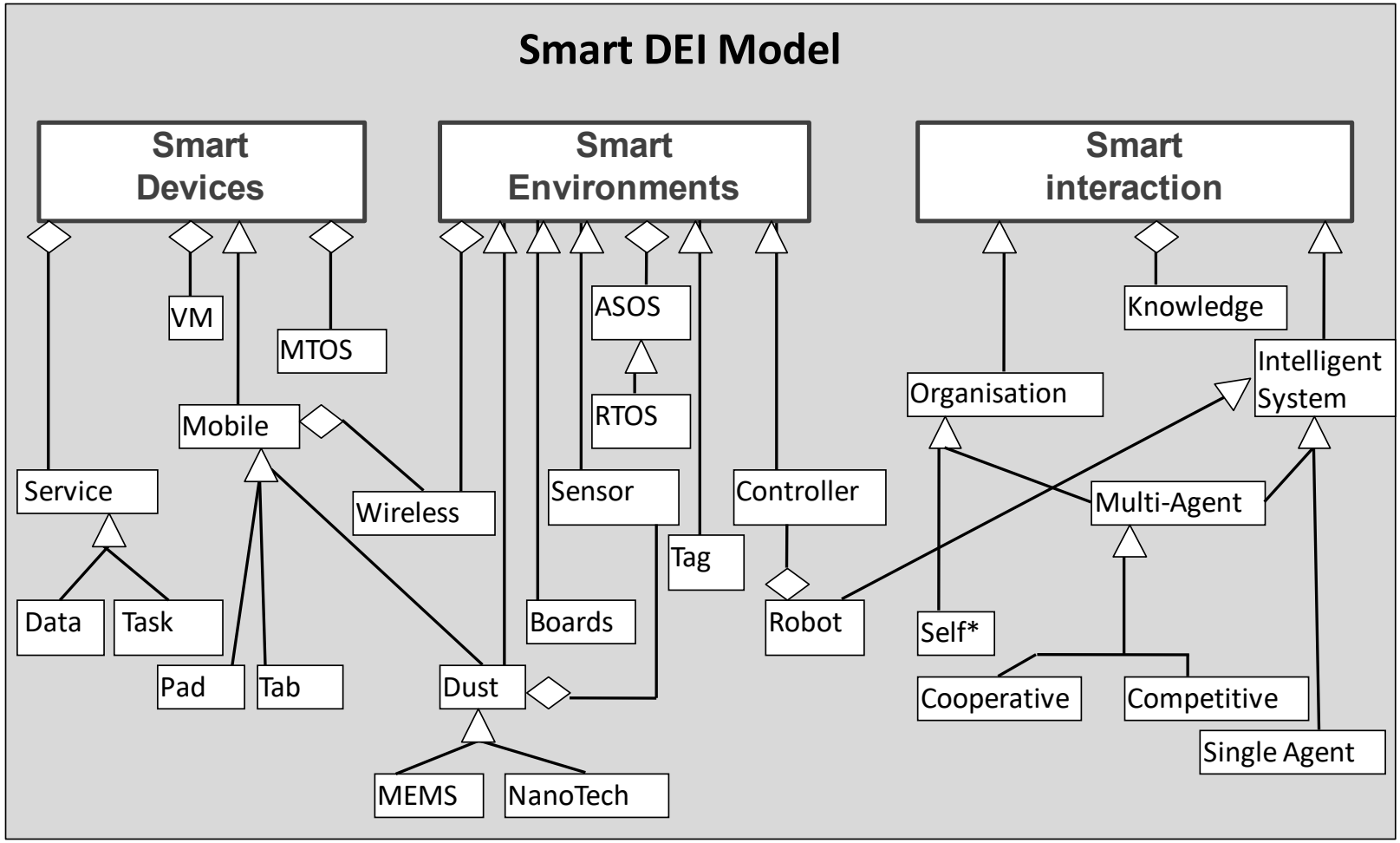
UbiCom System Model: Smart DEI



Smart DEI Model: types of smart device

- 3 main types of UbiCom system may themselves contain smart sub-systems at a lower level of granularity
  - e.g., a smart environment device may consist of smart sensors and smart controllers

UbiCom System: Smart Sub-Systems or Components



Smart Device Form Factors

- Devices tend to become smaller and lighter in weight, cheaper to produce.
- Devices can become prevalent, made more portable and can appear less obtrusive.

Weiser proposed a range of device sizes

1. *Smart Tabs*
2. *Smart Pads*:
3. *Smart Boards*:

Smart Device Form Factors (2)

Form Factors can be extended to support

4. *Smart Dust*
5. *Smart Skins*
6. *Smart Clay*

(See Chapters 5,6)

Design challenges in Using Varying Form Factors

- What are the design Challenges?

Smart Device: Variations

- Many variations or sub-types of smart devices,
  - e.g. *smart mobile device*, smart environment device
  - Use different combinations of sub-components
  - Support different combinations of UbiCom properties
  - Interact in different types of environment
  - Take different form factors

Smart Device Variations: Smart Mobile Device

- Many flavours or variants of these basic designs
- One of the most important variants of the smart device is the *smart mobile device*
- Combines several main types of UbiCom property:
- Note mobile devices are sometimes taken to be synonymous with wireless devices but these are different (see Chapter 11

Smart Mobile Devices

- Multi-purpose ICT devices, operating as a single portal to multiple services
- eases access & interoperability versus decreased openness
- Usually personalised devices, specified owner.
- Locus of control and user interface resides in the smart device.
- Main characteristics: mobility, open service discovery, intermittent resource access.

Smart Devices: Mobility

- Many dimensions for mobility
- *Static*:
- *Accompanied*:
- *Wearable*:
- *Embedded* (into objects):
- *Implanted* (into humans):
- *Untethered* or unanchored:
- Division between statics and mobiles can be more finely grained and multi-dimensional (see Chapter 4)

Volatile Service Access

- Devices access software services and hardware intermittently . Why?
- Devices can dynamically discover services available.
- Context-aware discovery can improve basic discovery.
- Asymmetric remote service access, more downloads than uploads, tends to occur.

Smart Mobile Devices: Context-Aware

- Can use contexts to filter information & service access.
- Often designed to work with a reference location in physical environment called a *home location*,
- Mobile devices are ICT *resource constrained*.
- Mobile devices tend to use a finite internal energy cache.
- Devices' configuration, operation tends to be personalised

Smart Environment (Devices)

- Definitions
- Consists of a set of smart devices specialised to interact with their (virtual, phys., human) environments.
- Typically, embedded single task devices; not MTOS devices
- Can automatically respond to or anticipate users, using iHCI
- Smart environments support bounded, local user context
- Smart environment devices may also be:
  - fixed versus anchored mobile versus unanchored mobile devices
  - macro to micro to nano
- Smart environment device design issues what?
  - See Chapter 6, 7
- Smart environment device management issues what?
  - See Chapter 12

Types of Smart Environment Device Interaction

- Tagging and Annotating
- Sensing & monitoring
- Filtering
- Adapting
- Controlling
  - Assembling
  - Regulating

Smart Interaction

- Additional type of design is needed to knit together many individual system components and interactions.
- Smart interaction promotes unified & continuous interaction model between UbiCom applications & their UbiCom infrastructure, physical world & human environments.
- Internal self-organising system vs. externally driven system
- Components can interact cooperatively versus competitively
- Several benefits to designs based upon set of interacting components:
- A range of levels of interaction between UbiCom System components
- Distinguish between (basic) interaction and (smart) interaction

Basic Interaction

- Typically involves two interlinked parties, a sender and a receiver.
- Sender knows things in advance:
- Two main types of basic interaction synchronous versus asynchronous
- (Chapter 11)

Smart Interaction

Smart Interaction extends basic interactions as follows.

- Coordinated interactions
- Cooperative versus competitive interaction
- Policy and Convention based Interaction
- Dynamic Organisational Interaction
- Semantic and Linguistic Interaction
- (See Chapter 9)

Smart DEI Model Summary

- Basic Smart Device has many variations
  - 6 different physical form factors
  - 5 different groups of internal properties & over 70 sub-properties
- Multiple flavours of smart device,
  - e.g., Smart Mobile type of Smart device
  - e.g., Smart Environment type of Smart Device
- UbiCom System interact across 3 main types of environment: physical, virtual & human
- System of systems models in terms of multiple device combinations and interactions

Common Myths of Ubiquitous Computing

1. There is a single definition which will accurately characterises Ubiquitous Computing
2. The ideal type of Ubiquitous computing is where all the properties of ubiquitous must be fully supported
3. Ubiquitous computing means making computing services accessible everywhere.
4. Ubiquitous computing is boundless computing
5. Ubiquitous computing is just about HCI
6. Calm Computing should be used as a model for all HCI.
7. Ubiquitous computing is just about augmenting reality

Common Myths of Ubiquitous Computing 2

8. Ubiquitous computing is just distributed or virtual computing
9. Ubiquitous computing is just mobile wireless computing
10. Ubiquitous computing is just about smart environments
11. Ubiquitous computing need to be highly autonomous systems
12. Ubiquitous computing is just about physical world context-awareness
13. Ubiquitous computing is just distributed intelligence
14. Ubiquitous computing systems can operate effectively in all kinds of environments:

Overview

- Living in a Digital World
- Modelling the Key Ubiquitous Computing Properties
- Ubiquitous System Environment Interaction
- Architectural Design for UbiCom Systems: Smart DEI Model
- **Course Outline √**

Organisation of UbiCom Course Book: Chapter Comparison

No	Section Title	DEI	UbiCom Property	Environment Interactions
1	Vision and Basics Concepts	DEI	All	All
2	Applications and Requirements	DEI	Distributed, iHCI, Context-aware	All
3	Smart Devices and Services	Devices	Distributed	C2C
4	Smart Mobile Devices, Device Networks and Smart Cards	Devices	Distributed	C2C
5	Human Computer Interaction	Devices	iHCI	HCI
6	Tagging, Sensing and Controlling	Environment	Context-aware	CPI
7	Context-Awareness	Environment	Context-aware	CPI
8	Intelligent Systems	Interaction	Intelligent	C2C, HCI
9	Intelligent Interaction	Interaction	Intelligent, iHCI	H2H, C2C
10	Autonomous Systems and Artificial Life	Interaction	Autonomy, Intelligence,	C2C
11	Communication Networks	Devices	Distributed	C2C
12	Smart Device Management	Devices	Distributed, iHCI, Context-aware	C2C, HCI, CPI,
13	Ubiquitous System Challenges and Outlook	DEI	All	All

Organisation of UbiCom in this Course (Main Sections)

- *Basics*: Vision and Basics Concepts (Chapter 1); Applications and Requirements (Chapter 2);
- *Smart Devices*: Smart Devices and Services (Chapter 3); Smart Mobile Devices, Networks and Cards (Chapter 4); Human Computer Interaction (Chapter 5);
- *Smart Environments*: Tagging, Sensing, Control of the Physical World (Chapter 6); Context-aware Systems (Chapter 7);
- *Smart Interaction*: Intelligent Systems (Chapter 8); Intelligent Interaction (Chapter 9); Autonomous Systems and Artificial Life (Chapter 10).
- *Middleware* and Outlook*:* Ubiquitous Communication (Chapter 11); Smart Device Management (Chapter 12); Ubiquitous System Challenges and Outlook (Chapter 13).

Overview

- **Living in a Digital World √**
- **Modelling the Key Ubiquitous Computing Properties √**
- **Ubiquitous System Environment Interaction √**
- **Architectural Design for UbiCom Systems: Smart DEI Model √**
- **Course Outline √**

Summary & Revision

For each chapter

- See book web-site for chapter summaries, references, resources etc.
- Identify new terms & concepts
- Apply new terms and concepts: define, use in old and new situations & problems
- Debate problems, challenges and solutions
- See Chapter exercises on web-site

Exercises: Define New Concepts

- Ubiquitous Computing, etc

Exercise: Applying New Concepts

- What are the main properties of UbiCom?
- etc

UbiCom Book Slides

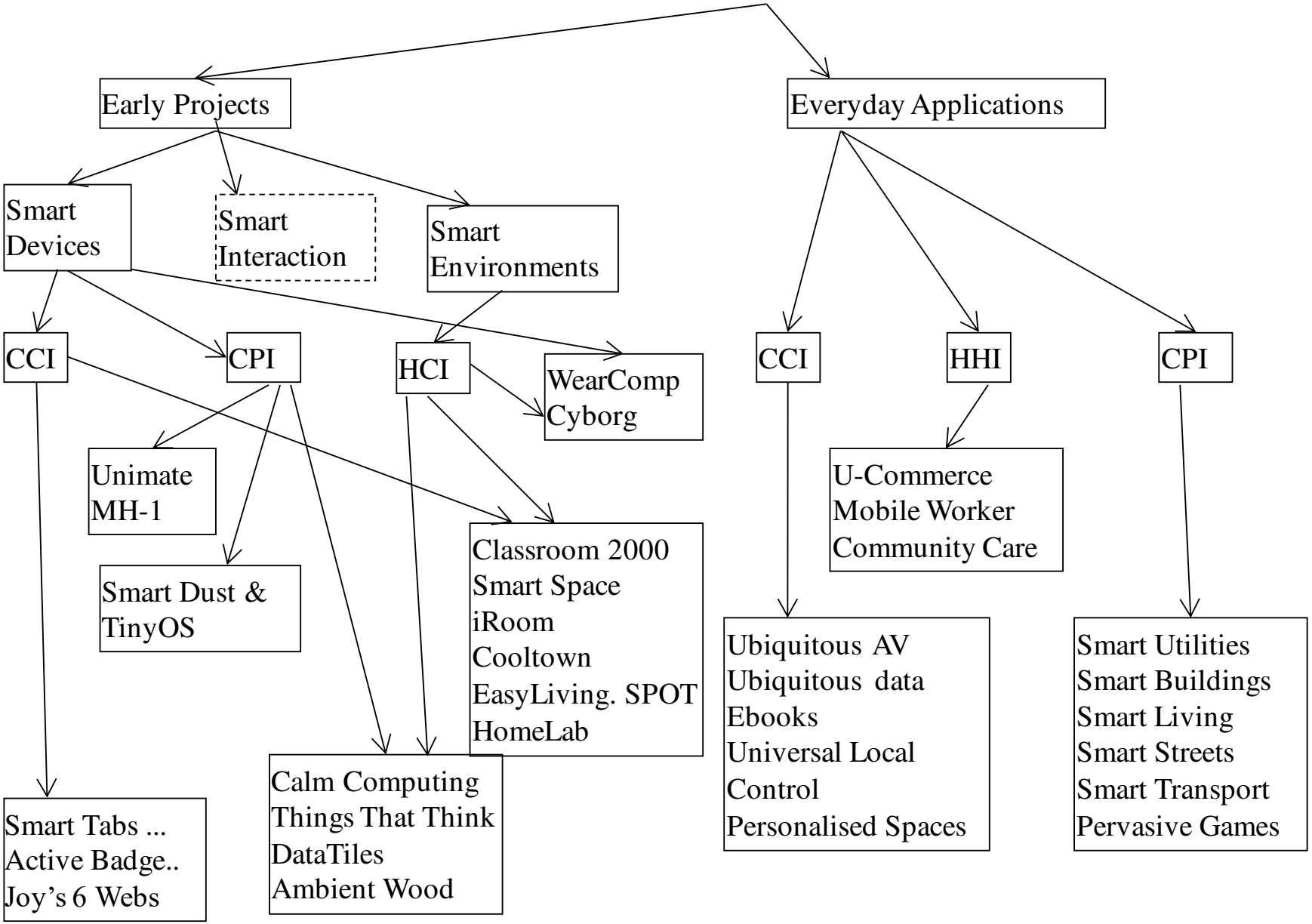
Chapter 2
Applications

Name:
Email: Web

Overview

- **Early UbiCom Research Projects √**
- Everyday Applications in the Virtual, Human and Physical World
- Some example projects in more detail

Ubiquitous Computing Applications and Projects



Early UbiCom Research Projects

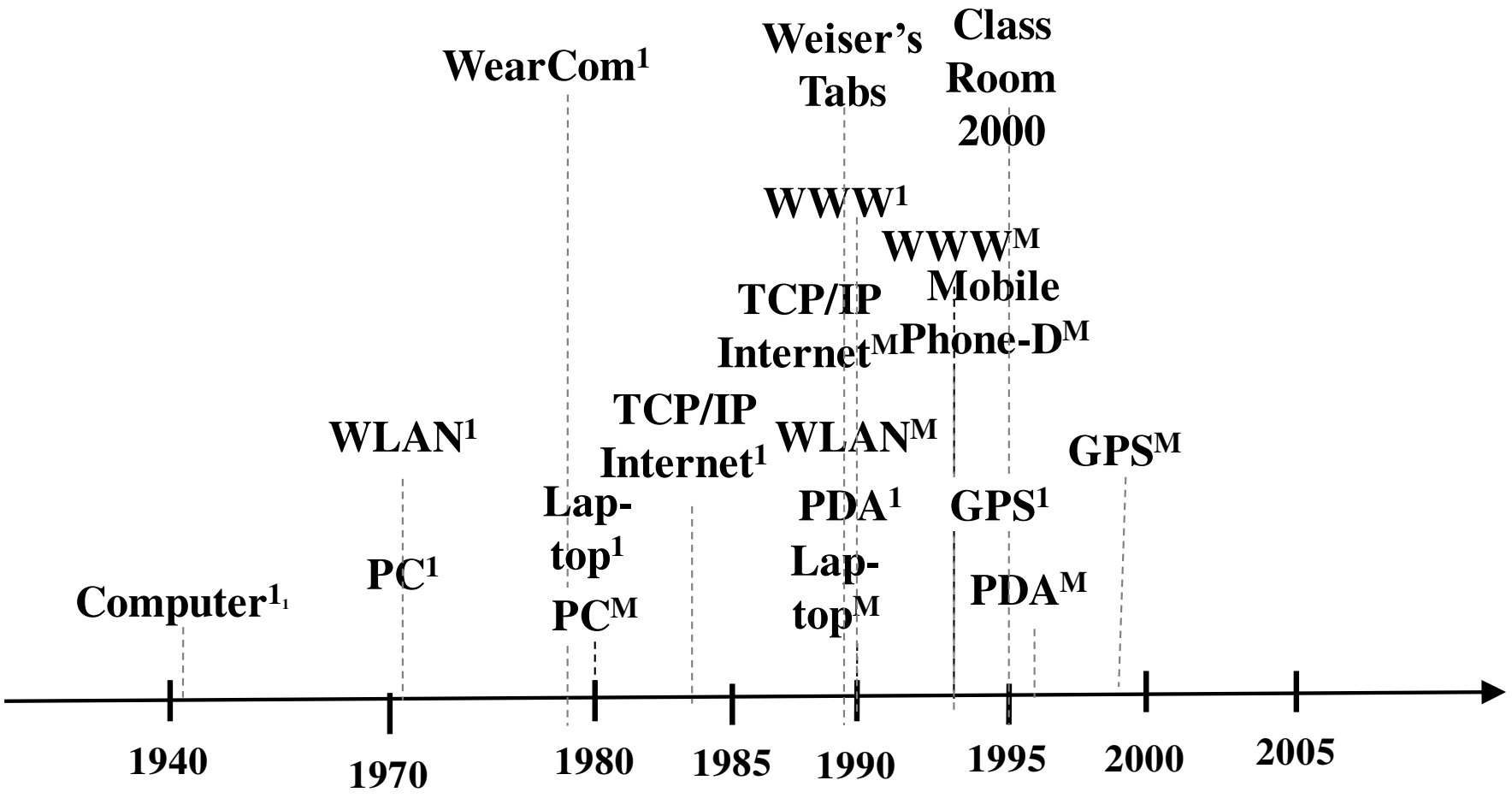
- **Smart Devices: CCI √**
  - **PARC Tab, MPad & LiveBoard; Active Badge, Bat and Floor**
- Smart Environments: CPI and CCI
  - Classroom 2000, Smart Space and Meeting Room, Interactive Workspaces and iRoom, Cooltown, EasyLiving and SPOT, HomeLab and Ambient Intelligence
- Smart Devices: CPI
  - Unimate and MH-1 Robots, Smart Dust and TinyOS
- Smart Devices: iHCI
  - Calm Computing, Things That Think and Tangible Bits, DataTiles, WearComp and WearCam, Cyborg
- Other UbiCom Projects

What ICT Environments Were Like

- Late 1980s, when much of the early work on UbiComp started
  - ???

- A distinction has been made in the history between the availability of the first prototypes or ICT product (1) and the first widespread commercial uptake of an ICT product (M).
  - Difference between the (1) and (M) phases seems to be averaging about 10 years give or take a few years.
- Today, it is hard to imagine such a world, where people were often unreachable if away from a fixed phone point and computing was only available at a desk computer, attached to the wired Internet.

Short History of ICT Technology



Could also note when specific PC technologies arose, e.g., hard-disk, mouse, removal memory cards, etc

Active Badge, Bat and Floor

Active Badge (forerunner of ParcTab) at Cambridge University / Olivetti

- 1st context-aware computing application
- Designed an aid for a telephone receptionist
- Active Badge periodically sends infrared signals to sensors embedded in rooms throughout the building.
- Limited Location determination accuracy
- See http://www.cl.cam.ac.uk/research/dtg/attarchive/bat/

Active Badge, Bat and Floor

Active Bat

- Uses ultrasound, greater accuracy ~ 3 cm.
- Base station used for position determined

Active Floor

- Identification by person's gait,
- Pros and Cons
- Special Floor design.

PARC Tab, MPad, LiveBoard

- 3 main intertwined devices and applications known as known as *Boards, Pads and Tabs developed at PARC*, Large wall-display program called LiveBoard

- Smaller computers Book-sized MPad

- Palm-sized ParcTab computer

- See http://www.parc.com

ClassRoom 2000

- To capture the live experiences of the occupants and to provide that record to users for later access and review.
- 1995, *Classroom 2000* (led by Abowd, Georgia Institute of Technology)
- Most research focussed on development of multimedia-enhanced materials
- Classroom 2000 researched content generation by instrumenting a room with the capabilities to automatically record a lecture

- See http://www.cc.gatech.edu/fce/eclass/pubs/

Smart Space and Meeting Room Projects

- NIST (1998-2003): use of pervasive devices, sensors & networks for context-aware smart meeting rooms that sense ongoing human activities and respond to them
- Meeting Room design.
- 2 sets of tools were used to manage sensor data.
- When people talk, system takes dictation, records a transcript of the meeting, tracks individual speakers, follow what the conversation is about and triggers associated services from the Internet.
- Design supports an iHCI model for taking notes and for assisting speakers by intuitively providing further information.
- See http://www.nist.gov/smartspace/talksAndPubs.html

Interactive Workspaces Project

- Interactive Workspaces project (Stanford University, 1999) investigated design and of rooms (iRooms) to create applications integrating the use of multiple types of devices of varying form factors
- Also developed several interaction patterns for interacting with large high resolution displays
- FlowMenu
- ZoomScape
- Typed drag and drop support:
- See http://graphics.stanford.edu/papers/iwork-overview/

Cooltown

- HP Project , 2000-2003, to develop a vision of UbiCom to support:
  - Key feature of Cooltown is that each physical and virtual world resource has a Web presence (URL)
- 3 important benefits of using the Web for Mobile users situated in the physical world:
  - Ubiquitous access
  - Just Enough Middleware
  - Local means local:

EasyLiving & SPOT

- EasyLiving project (Microsoft, 1997-2003) developed intelligent environments to support dynamic aggregation of diverse I/O devices into a single coherent user experience.
- SPOT devices (Microsoft, 2003) designed to listen for digitally encoded data such as news stories, weather forecasts, personal messages, traffic updates, and retail directories transmitted on frequency sidebands leased from commercial FM radio stations

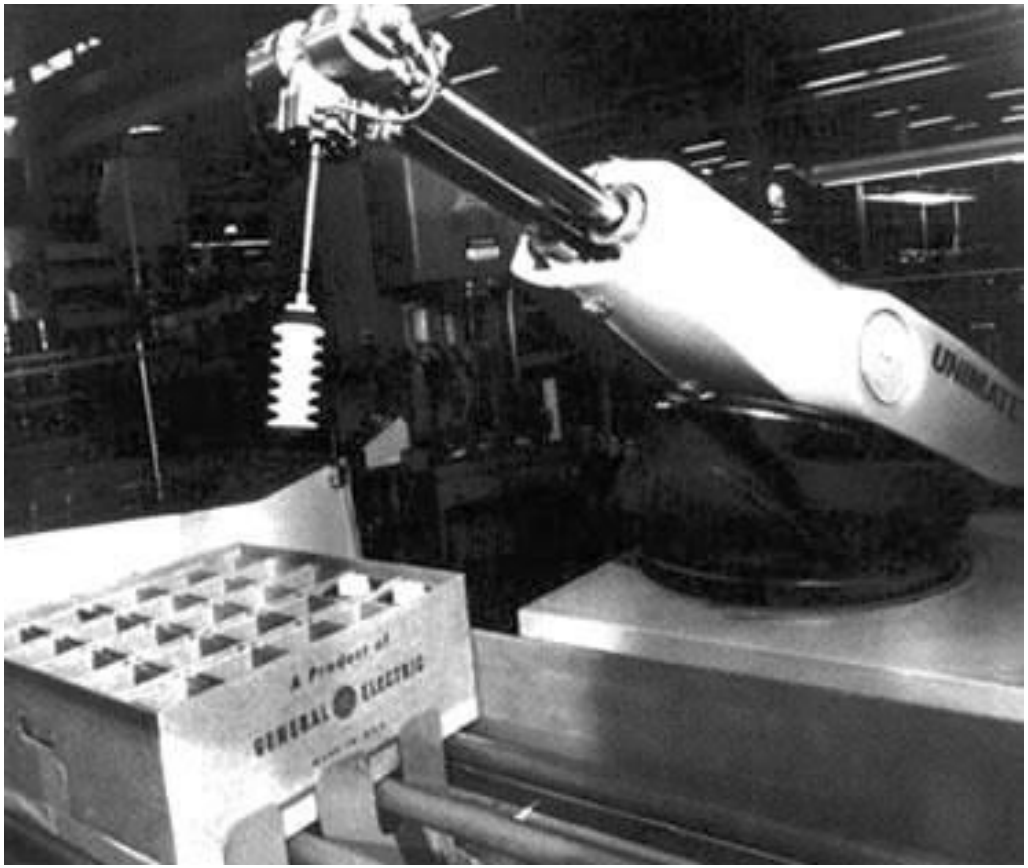
Ambient Intelligence (AmI)

- Proposed by Philips in late 1990s as a novel paradigm for consumer electronics that is sensitive responsive to the presence of people
- & became part of a EU Research Framework (FP6 IST)
- Key properties of AmI systems are:
  - User-aware / iHCI:
  - Intelligence?:
  - Embedded:

Unimate and MH-1 Robots

- Machines are used to perform physical tasks that are very labour intensive and repetitive or are too dangerous or difficult for humans to implement directly.
- Automated machines that just do one thing are not robots.
- Robots have the capability of handling a range of programmable jobs.
- 1961, Ernst developed the MH-1
- 1st first industrial computer controlled robot, the Unimate designed by Engelberger

Unimate Robot



See http://www.thocp.net/reference/robotics/robotics2.htm

Smart Dust

- Micro fabrication and integration of low-cost sensors, actuators and computer controllers, MEMS (Micro Electro-Mechanical Systems)
- Can be sprayed & embedded throughout the digital environment
- Creating a digital skin that senses physical & chemical phenomena
- See Smart Dust project (Pister, UCB)

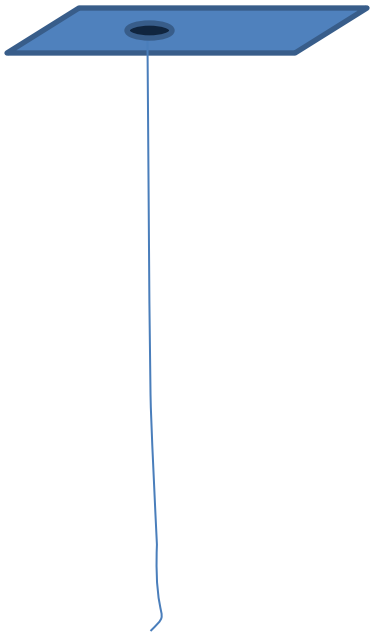
Smart Dust

Photo: courtesy of Brett Warneke

iHCI: Calm Computing

- Weiser noted whereas computers and games for personal use have focused on the excitement of interaction, when computers are all around we will interact with them differently. We often want to compute while doing something else.
- Calm technologies are said to encalm us as they can empower our periphery in three ways:

iHCI: Calm Computing



- Example of calm technology was the “Dangling String” created by artist Natalie Jeremijenko, situated at PARC
- String jiggled in proportion to the degree of subnet activity

iHCI: Tangible Bits & Things That Think (TTT)

- In 1997, and still to a large extent10 years later, GUI-based HCI displayed its information as "painted bits" on rectangular screens in the foreground
- In contrast, Tangible Bits project (led by Ishii, MIT, 1997) aimed to change "painted bits", into "tangible bits" by leveraging multiple senses & multimodal human interactions within the physical world
- "Tangible User Interfaces" emphasize both visually intensive, hands-on foreground interactions, and background perception of ambient light, sound, airflow, and water flow at the periphery of our senses.
- See http://ttt.media.mit.edu/, http://tangible.media.mit.edu/,

DataTiles

- DataTiles project (Sony, 2001, led by Rekimoto) focussed on interactive user interfaces that use task specific physical objects as alternatives to conventional HCI.
- System consists of acrylic transparent tiles with embedded RFID tags
- Advantages ?

- Three key types of interaction were embodied by the system.

DataTiles

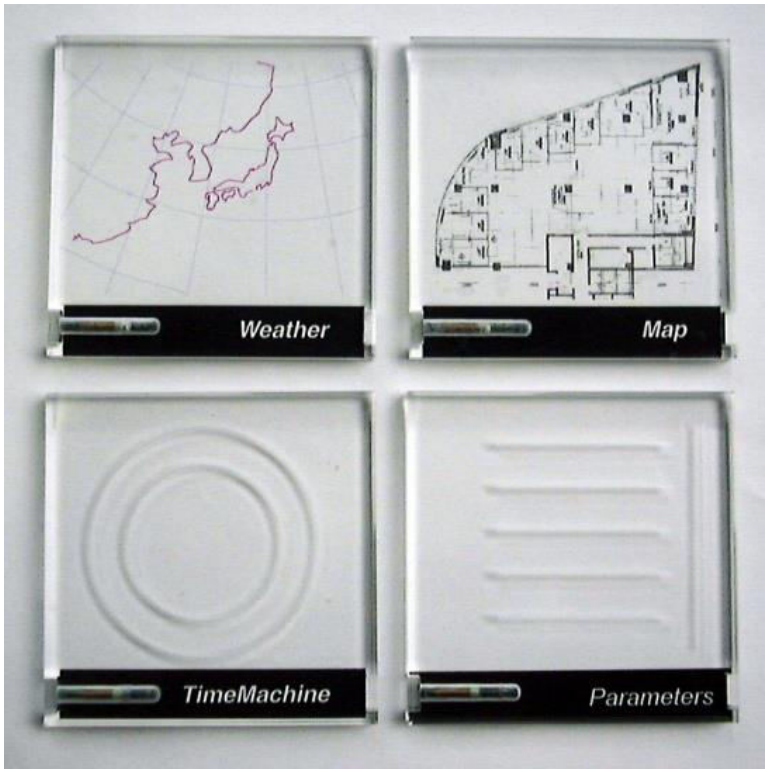


Photo courtesy of Sony Computer Science Laboratories, Inc.

- Allows users to manipulate data in form of tangible “tiles”
- Combinations of data streams and functions make it possible to create new applications

WearComp and WearCam

- Mann's experiments with wearable computers started in late 1970s.
- Main application was recording personal visual memories that could be shared with other via the Internet.

Photo courtesy of http://en.wikipedia.org/wiki/Wearable_computing

WearComp and WearCam

Later generations of WearComp supported three key features.

1. Wearable computer was hidden. Customised glasses used as HUD, conductive fabric used as BAN.
2. Mediated reality was supported, reality may be diminished or otherwise altered as desired, not just augmentered.
3. Thirdly, Homographic Modelling was supported in the display.

- See http://www.eecg.toronto.edu/~mann/

Cyborg 1.0 and 2.0

- Implanted into human mobile hosts are a form of embedded device.
- Cyborg 1.0, a silicon chip transponder implanted in a forearm which remained in place for 9 days (1998, Warwick)
- Cyborg 2.0 (2002, Warwick) new implant in his lower arm could send signals back and forth between the nervous system and a computer
- See http://www.kevinwarwick.com/

Cyborg 2.0

Electrode array surgically implanted into Warwick's left arm and interlinked into median nerve fibres is being monitored.

Photo Courtesy of University of Reading

R&D UbiCom Applications

- There are many other innovative UbiCom projects, only a selection of these is given here.
- Two of the main conferences that cover a greater range of UbiCom projects are:
  - IEEE http://www.UbiCom.org
  - ACM http:// www.percom.org
- Research here (add-link) for what your institute is doing in this area

Analysis of Early Projects Achievements and Legacy

- Focus on 3 basic UbiComp properties: iHCI, context awareness and distributed access
- Many innovative iHCI projects

.

Analysis of Early Projects: Distributed Access Support

- Early work at PARC and by Olivetti, late 1980s was focussed on basic smart mobile device model design for Tabs and Pads.
- Proprietary communication & location-awareness for mobile users : no commercial mobile ICT devices, widely available wireless networks.
- Late 2000s, mobile devices and wireless networks are widely available
- Service discovery of local network resources was weak and the discovery of other local environment resources is still virtually non-existent
- -> Much of the vision of Cooltown is not routinely available.
  - Reasons for this?

Analysis of Early Projects: context-awareness

- Context-awareness: mainly location awareness
- Early achievements based upon (local not global) location awareness indoors with heavily instrumented environment.
- Location-determinism today tends to be supported mainly as stand-alone devices and services that are not readily interoperable.
- GPS for outdoor use.
- Systems for indoor use are available today based, e.g., based upon trilateration using WLAN but not ubiquitous (See Chapter 8)

Analysis of Early Projects: iHCI

- Electronic boards
  - Allow users to collaboratively edit text and graphics were prototyped at PARC in the early 1990s -> later became commercial products.
  - Used in Classroom 2000 in 1995-1998 by Abowd et al. -> now routinely used in many educational establishments that support distance learning.
- Wearable smart devices
  - still in infancy, several products are available but they are not yet in pervasive use.
- iHCI
  - is a continuing research initiative.
  - Very many variations – not clear which will catch on, if there is a mass market for each of these.

Student Project Ideas

- ???

Overview

- Example Early UbiCom Research Projects
- **Everyday Applications in the Virtual, Human and Physical World √**
- Some Example Projects in More Detail

Everyware UbiCom Applications

- Vision: ubiquitous computer systems to support people in their daily activities in the physical world tasks to simplify these and to make these less obtrusive.
- People will live, work, and play in a seamless computer enabled environment that is interleaved into the world.
- Bushnell (1996) coined variations of term *ware* such as deskware, couchware, kitchenware, autoware, bedroomware and bathware to reflect the use of ubiquitous computing for routine tasks.
- Greenfeld (2006) used the term *everyware* to encompass the many different types of ware

Everyware UbiCom Applications

- Many ways to categorise UbiCom applications from an end-user perspective:

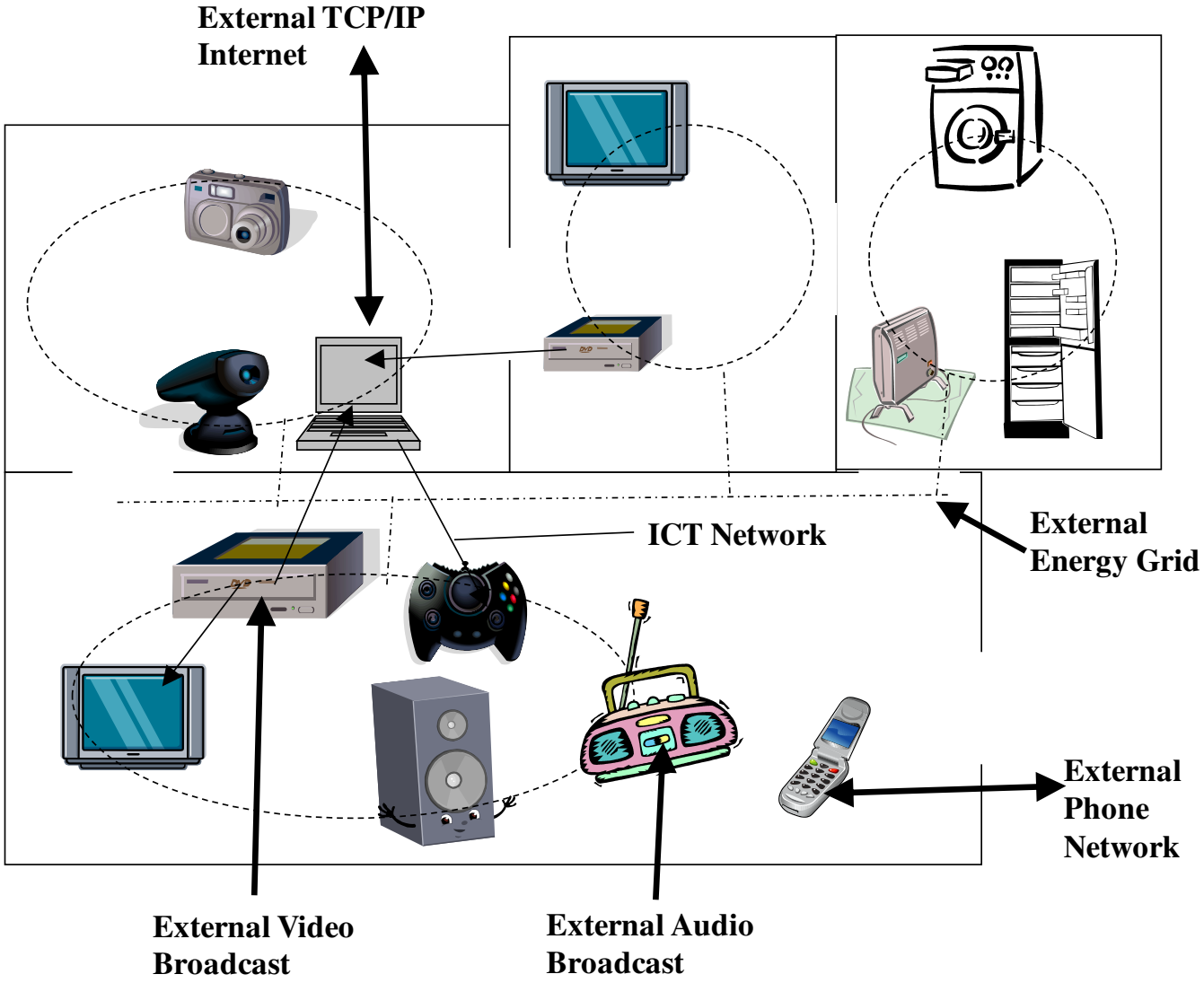
- Here we categorise applications with respect to:
  - smart mobile device versus smart environment
  - by type of environment interaction such as CCI, HCI and CPI.

Everyware UbiCom Applications: CCI

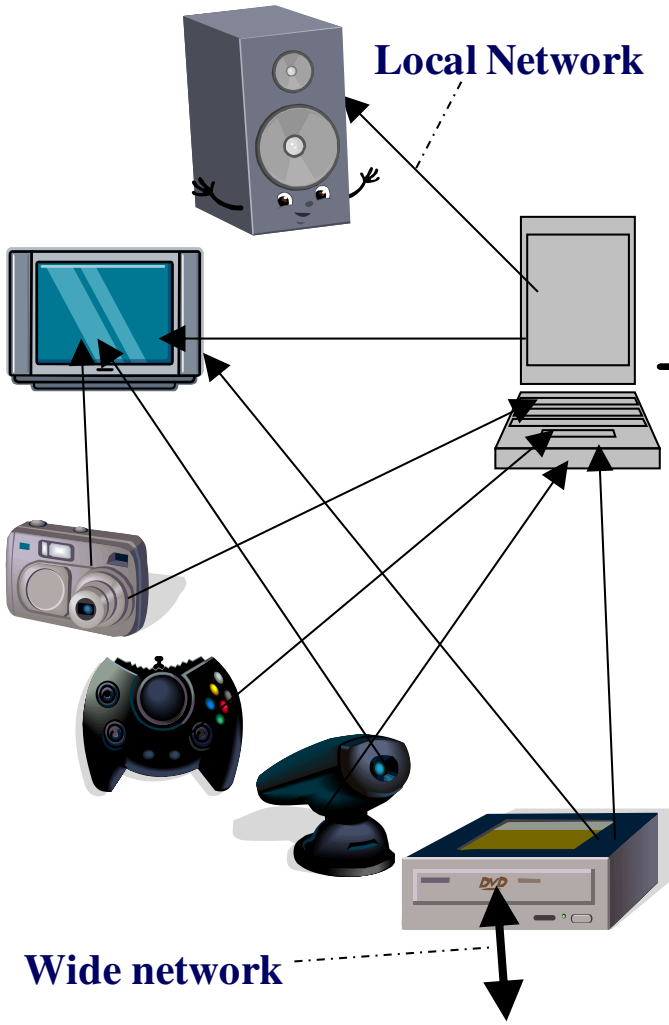
- Ubiquitous Networks of Devices: CCI
  - Human Computer Interaction
  - Ubiquitous Audio-Video Content Access
  - Ubiquitous Information Access and Ebooks
  - Universal Local Control of ICT Systems
  - User-awareness and Personal Spaces

Ubiquitous MM Content Access

- Multi-media (MM) content via any network & access device
- Broadcast MM content, professionally, created by third-parties, copyright, non-interactive, downloaded, read-only content, stored & manipulated in access device.
- User generated, locally created content that is modifiable
- From 1 to many content services per network


External TCP/IP
Internet
ICT Network
External
Energy Grid
External
Phone
Network
External Video
Broadcast
External Audio
Broadcast

Audio-Video Cluster



Computer as Hub of cluster

User Interaction

Processing:
edit, annotate, compose

Management:
Resources & Content

Internal Comms.

Storage

Email
Web
Chat
VoD
VoIP

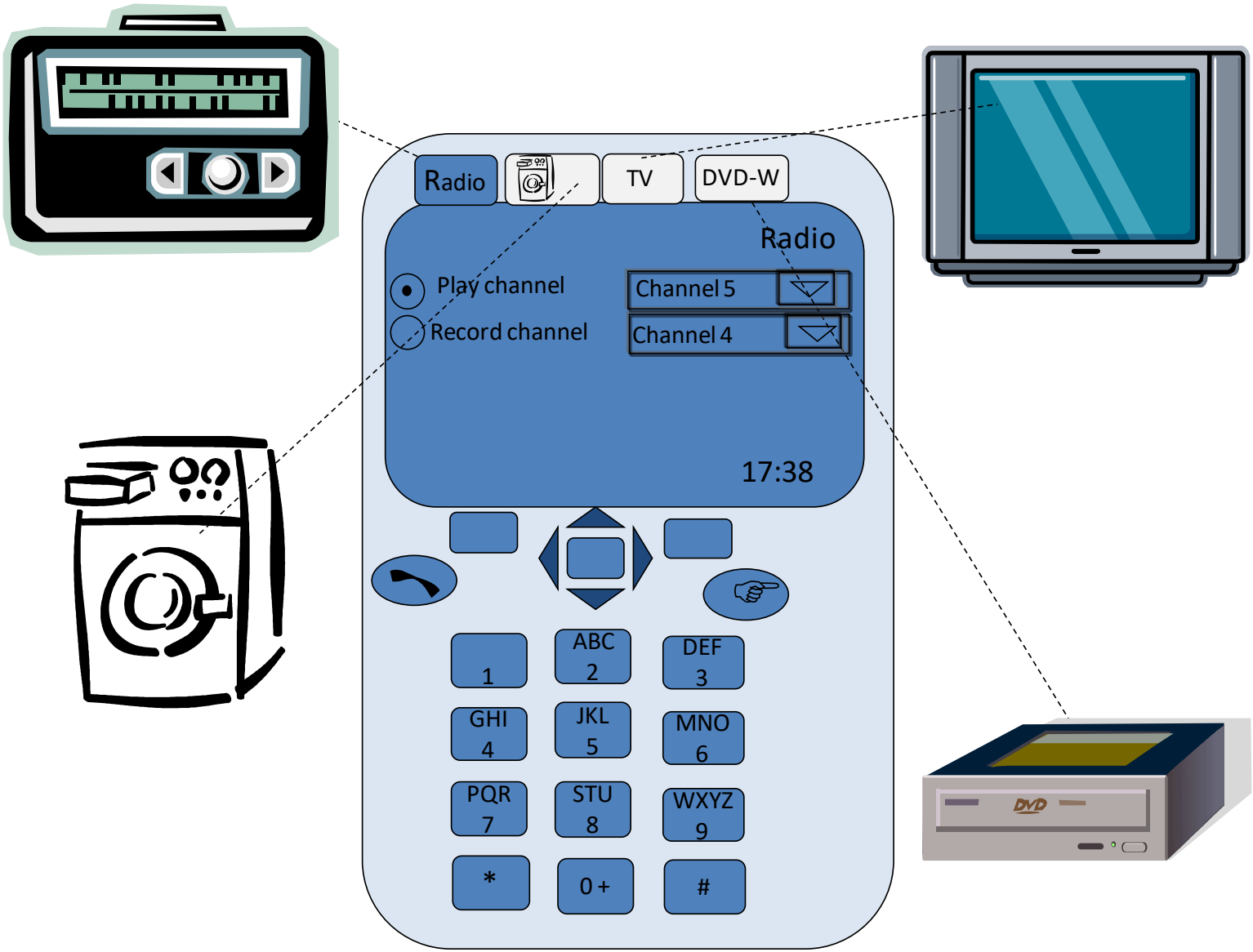
Universal Information Access

- PC still the dominant information access device
- PC suffers from a number of limitations compared to its paper counterpart. What?
- Specialised reading devices: ebooks, epaper
  - E.g. www.sonystyle.com, www.amazon.com (kindle, USA only when 1st released)
- Good Content adaptation & layout critical
  - Microsoft Word, Adobe Acrobat, Web browsers have many limitations

Universal Control of Local Devices

- Appliances controlled using Infrared, short-range controller
- Some well known conventions are used to label common function buttons
- No convention for less common functions –> read the manual.
- 1 controller per appliance -> many controllers, discarded each time, appliance is upgraded
- Two types of hand-held universal local control device have been proposed that can be configured for multiple local devices:

- Use of *mobile phones* and PDAs as universal local controllers?
- Focus on control of virtual rather than physical services


Radio
TV
DVD-W
Radio
Play channel
Record channel
Channel 5
Channel 4
17:38
1
ABC 2
DEF 3
GHI 4
JKL 5
MNO 6
PQR 7
STU 8
WXYZ 9
*
0 +
#
DVD

User Awareness & Personalisation

- Personalisation: content and services tailored to individuals based upon on knowledge about their preferences and behaviour. Benefits
- Greater convenience, more relevant filtered information
- but trade off against a loss in privacy.
- Users can personalise configuration of services, annotation of content
- Personal preferences could follow a user around
- Mobile devices provide an obvious means for users to personalise their environment

Student Project Ideas

- ???

Everyware UbiCom Applications: HHI

- Human to Human Interaction (HHI)
  - Transaction-based M-Commerce & U-Commerce services
  - Enhancing the Productivity of Mobile Humans
  - Care in the Community

M-commerce and U-Commerce

- M-Commerce: variant of E-Commerce with services over mobile-wireless infrastructure
- U-Commerce Sub-type of eCommerce / mCommerce
  - (Watson et al. 2002)
- Characterised by:
  - Universality:
  - Uniqueness:
  - Unison:
  - Ubiquity:

Enhancing Productivity of Mobile Humans

- Productivity can suffer from a bottleneck when people don't have the right information where & when they need it
  - E.g.,
- Mobile users can access calls, email, diary, calendar and notepads
  - Does greater mobile service access empower or enslave us?
- Two-way interaction versus unilateral workflow across time & space
  - e.g.,,.
- Communities of practice
- Challenges?

Care in the Community

- ‘Vulnerable’ individuals at home monitored by friends, family and health professionals situated elsewhere.

- There are two basic kinds of approaches in terms of:
  - whether the subject explicitly asks for help from others or
  - Whether others can anticipate when the subject requires help.

Project Ideas

- Mobile services: various
- Combined Indoor and outdoor spatial information system
- Care in the Community: sensing activity
- Etc.

Everyware UbiCom Applications: (HPI, CPI)

- Physical Environment Awareness
- (Physical) Environment Control
- Smart Utilities (See Chapter 1)
- Smart Buildings and Home Automation
- Smart Living Environments and Smart Furniture
- Smart Street furniture
- Smart Vehicles, Transport and Travel
- Pervasive Games and Social Physical Spaces

Physical Environment Awareness

- Services slanted towards specific physical environment contexts, e.g.,
- Short-range (point-based, static) context determination
- Longer range(regional, roaming) context access
- Sensors for specific physical world phenomena are statically embedded into specific devices and services, e.g.,

(Physical) Environment Control

- Mobile phone or other hand held device can use a wireless link to issue simple control instructions
  - E.g.,
- Resources may be public, private, commercial
- privately owned, e.g., garage door or car door
  - .
- provided as pay per use services, e.g. drinks dispenser

- N.B. control and reconfiguration of many devices is manual

Smart Buildings and Home Automation

- Sensors & automation is increasing used in buildings to automate control of light, climate, doors, windows, security, ↑ energy efficiency.
- Sensors & control devices can be put in physical environment in a variety of ways
- Home automation, e.g., X10, seems more common in U.S. vs Europe
- Building today not well suited to keep pace with rapid technological changes and with recent sustainability concerns.

Smart Living Environments and Smart Furniture

- Several smart environment devices can adapt to human activities.
- Doors, lighting, taps and air ventilation can be designed to detect the presence of humans, to be activated by them and to adapt to them.
- *Smart fridge* behaves as a stock-control system
- MediaCup (Beigl et al., 2001)
- *Smart chairs* such as SenseChair (Forlizzzi et al., 2005)
- *Smart clocks* e.g., Microsoft wherabouts clock.
- *Smart mirrors: ca*n move to adapt view, e.g., as car moves, can overlay other information, can link to cameras.
- *Smart wardrobe* , s*mart bed*, *smart pillow Smart Mat*, *smart sofa* (*Park et al. 2003).*

Smart Vehicles, Transport and Travel

- Embedded computer systems increasingly being used within vehicles.
- Improves operation such as automatically controlling or providing assisted control.
- Automatically guided vehicles along track
- Inform waiting passengers of the status of arriving & departing vehicles.
- Location determination for remote tracking of vehicles
- Access travel info. much more conveniently
- Travel Tickets are also smarter – see smart cards (Chapter 4)
- Access to Internet in moving smart vehicles

Social Physical Spaces & Pervasive Games

- On detecting friends within a local vicinity, suggest meeting point, e.g., ImaHima
- Local traders electronic offers.
- Many social and economic issues ,

- Games: a core type of entertainment, social, interactive, application.
- Traditional or pre-electronic games: 2 types of interaction, HPI and HHI, uses game control interface (d-pad interface)
- In pervasive gaming, social activities and games exploit the potential of combining the physical space
- Electronic game types: mobile games, location-based games, *Augmented reality games, Adaptronic games,* Pervasive games..

Project Ideas

- ???

Overview

- Example Early UbiCom Research Projects
- Everyday Applications in the Virtual, Human and Physical World
- **Some Example Projects in More Detail √**

Example 1:

- Instructors to add examples here

Summary & Revision

For each chapter

- See book web-site for chapter summaries, references, resources etc.
- Identify new terms & concepts
- Apply new terms and concepts to old and new situations & problems
- Debate problems, challenges and solutions
- See Chapter exercises on web-site

Exercises: Define New Concepts

- Cooltown etc

Exercise: Applying New Concepts

- (See Web-site: http://www.elec.qmul.ac.uk/people/stefan/ubicom)

UbiCom Book Slides

Chapter 3
Smart Devices and Services

Stefan Poslad
http://www.eecs.qmul.ac.uk/people/stefan/ubicom

Related Links

- Basic Distributed Computer Interaction Models in this chapter are the basis for more advanced systems in later chapters, e.g., EDA Architecture can be used for:
  - Sense & Control systems (Chapter 6)
  - Context-based Systems (Chapter 7)
  - Reflexive Intelligent Systems (Chapter 8)
- Mobile Distributed Systems (Chapter 4)
- Management of Distributed Systems (Chapter 12)
- Advances in Distributed Systems (Chapter 13)

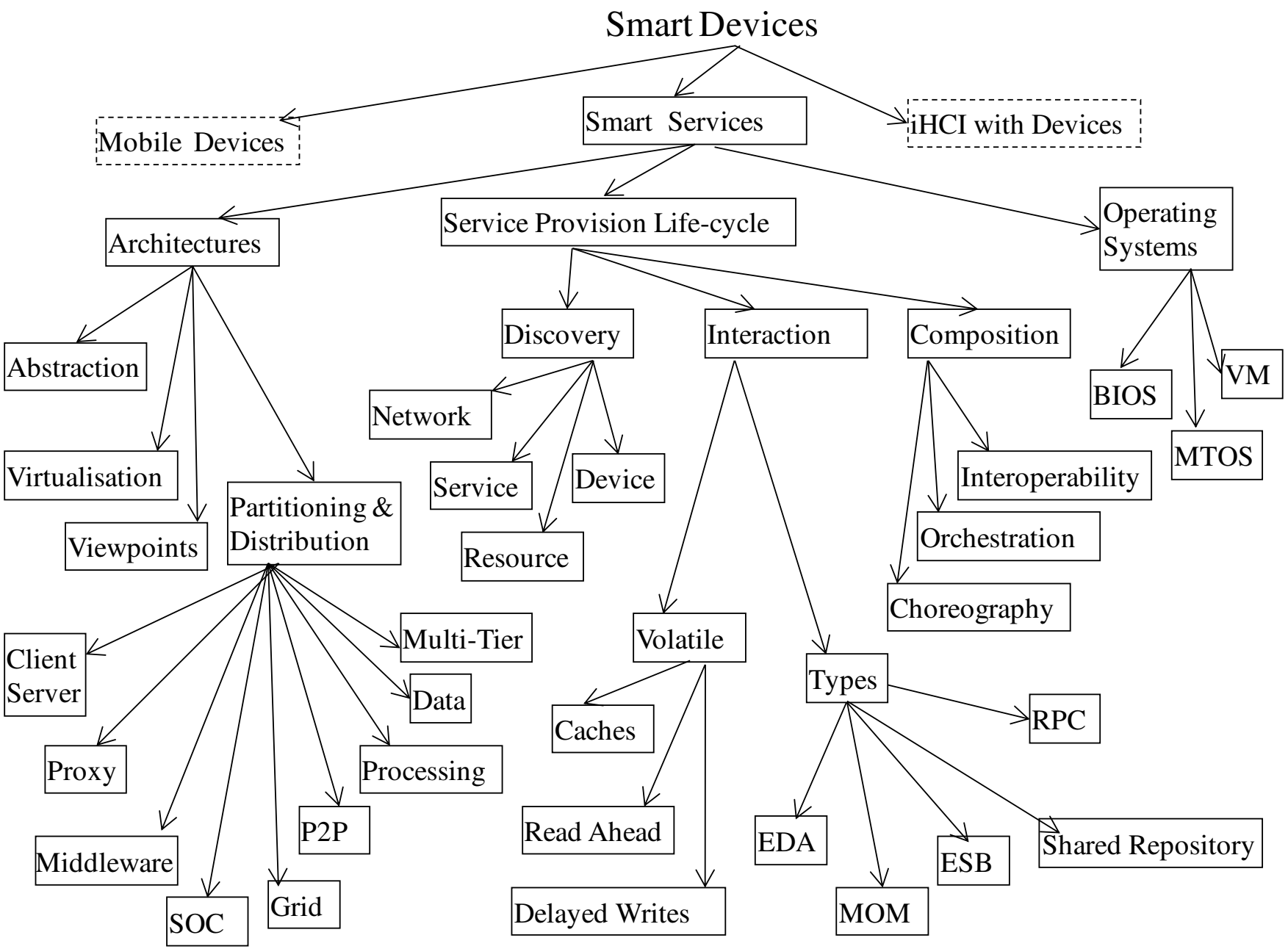
Chapter 3 Slides

The slides for this chapter are also expanded and split into several parts in the full pack

- **Part A: System Architectures**
- **Part B: Middleware, SOC & P2P**
- **Part C: Service Provision Life-cycle & Service Discovery**
- **Part D: Service Invocation**
- **Part E: Volatile Service Invocation & Service Composition**
- **Part F: MTOS, BIOS & VM**

Overview

- **Smart Device and Service Characteristics** √
- Distributed System Viewpoints
- System Abstraction
- Partitioning and Distribution of System Components
- Proxies and Middleware
- Service Oriented Computing (SOC) & Grid Computing
- Peer-to-Peer Systems (P2P)
- Service Provision Lifecycle □
- Service Discovery
- Service Invocation
- Service Composition
- MTOS, BIOS & VM


Smart Devices
Mobile Devices
Smart Services
iHCI with Devices
Architectures
Service Provision Life-cycle
Operating Systems
Abstraction
Virtualisation
Viewpoints
Partitioning & Distribution
Discovery
Interaction
Composition
Network
Service
Device
Resource
BIOS
VM
MTOS
Interoperability
Orchestration
Choreography
Client Server
Proxy
Middleware
SOC
Grid
P2P
Processing
Data
Multi-Tier
Volatile
Caches
Read Ahead
Delayed Writes
Types
RPC
EDA
MOM
ESB
Shared Repository

Smart Device Characteristics

- Multi-purpose ICT devices, operating as a single portal to multiple remote vs. local application services
- Usually personalised devices, specified owner.
- Locus of control and user interface resides in the smart device.
- Main characteristics of smart devices: mobility, open service discovery, intermittent resource access.
- Important type of smart device is smart mobile device
- Here, we focus on design issues for the service model used by UbiCom Applications

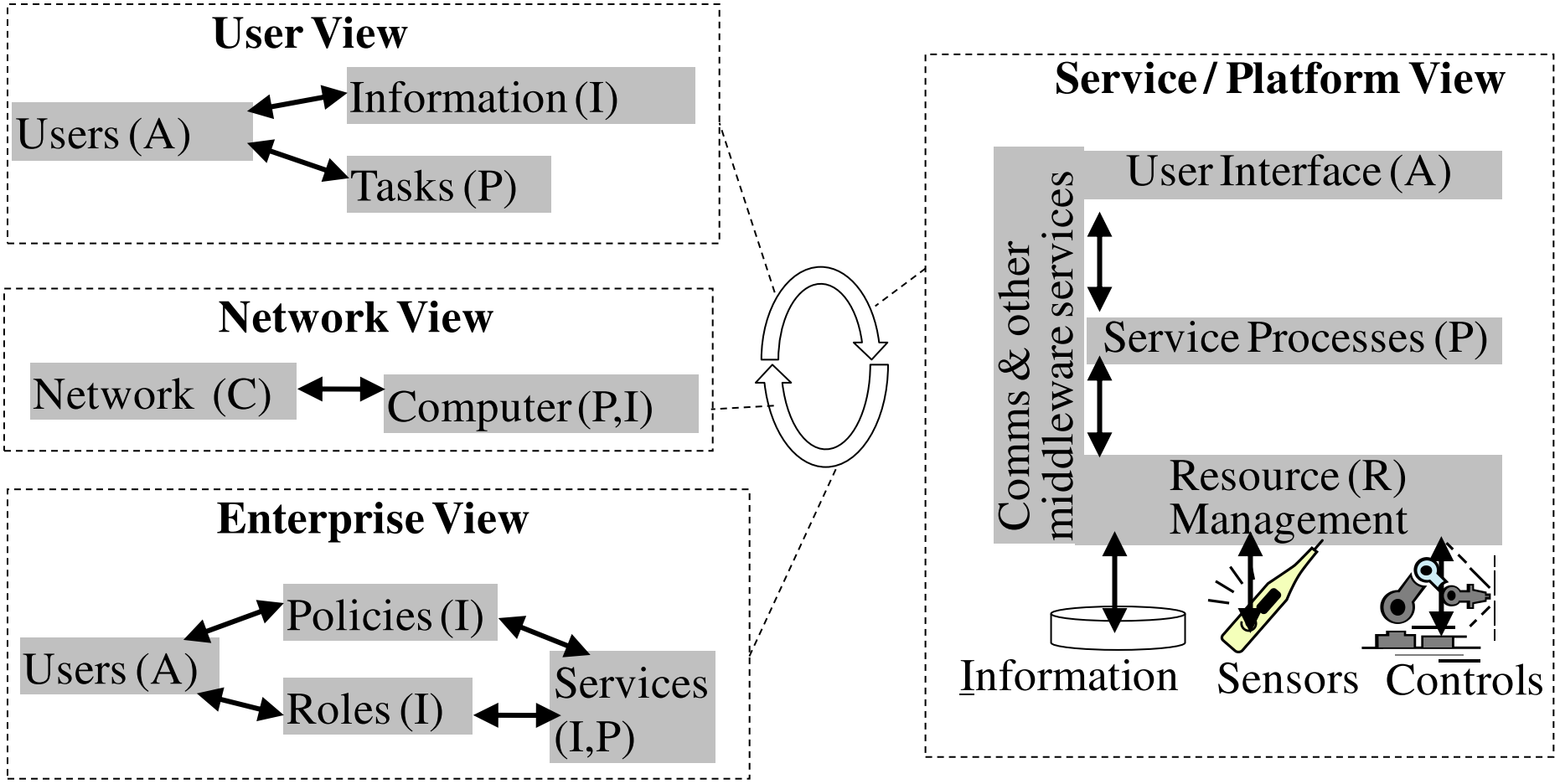
Overview

- Smart Device and Service Characteristics
- **Distributed System Viewpoints √**
- System Abstraction
- Partitioning and Distribution of System Components
- Proxies and Middleware
- Service Oriented Computing (SOC) & Grid Computing
- Peer-to-Peer Systems (P2P)
- Service Provision Lifecycle □
- Service Discovery
- Service Invocation
- Service Composition
- MTOS, BIOS & VM

Distributed System Viewpoints

- Distributed ICT Systems can be modelled from multiple complementary viewpoints with respect to:
- Viewpoints can be regarded as architectural patterns, conceptual models that capture the essential elements of an ICT system architecture and its interrelationships. Multiple viewpoints:
  - Individual user view
  - Enterprise user view:
  - Information system, service or computation platform view:
  - Network view: network elements and computer nodes
- Viewpoint model standards: RM-ODP (ISO), IEEE 1471 model

Distributed System Viewpoints



A = Access/presentation, I = Info./data, P = Processing/computation, C=Comms/networking

Overview

- Smart Device and Service Characteristics
- Distributed System Viewpoints
- **System Abstraction √**
- Partitioning and Distribution of System Components
- Proxies & Middleware
- Service Oriented Computing (SOC) & Grid Computing
- Peer-to-Peer Systems
- Service Provision Lifecycle □
- Service Discovery
- Service Invocation
- Service Composition
- MTOS, BIOS & VM

Reducing System Complexity using Abstraction (Modularisation)

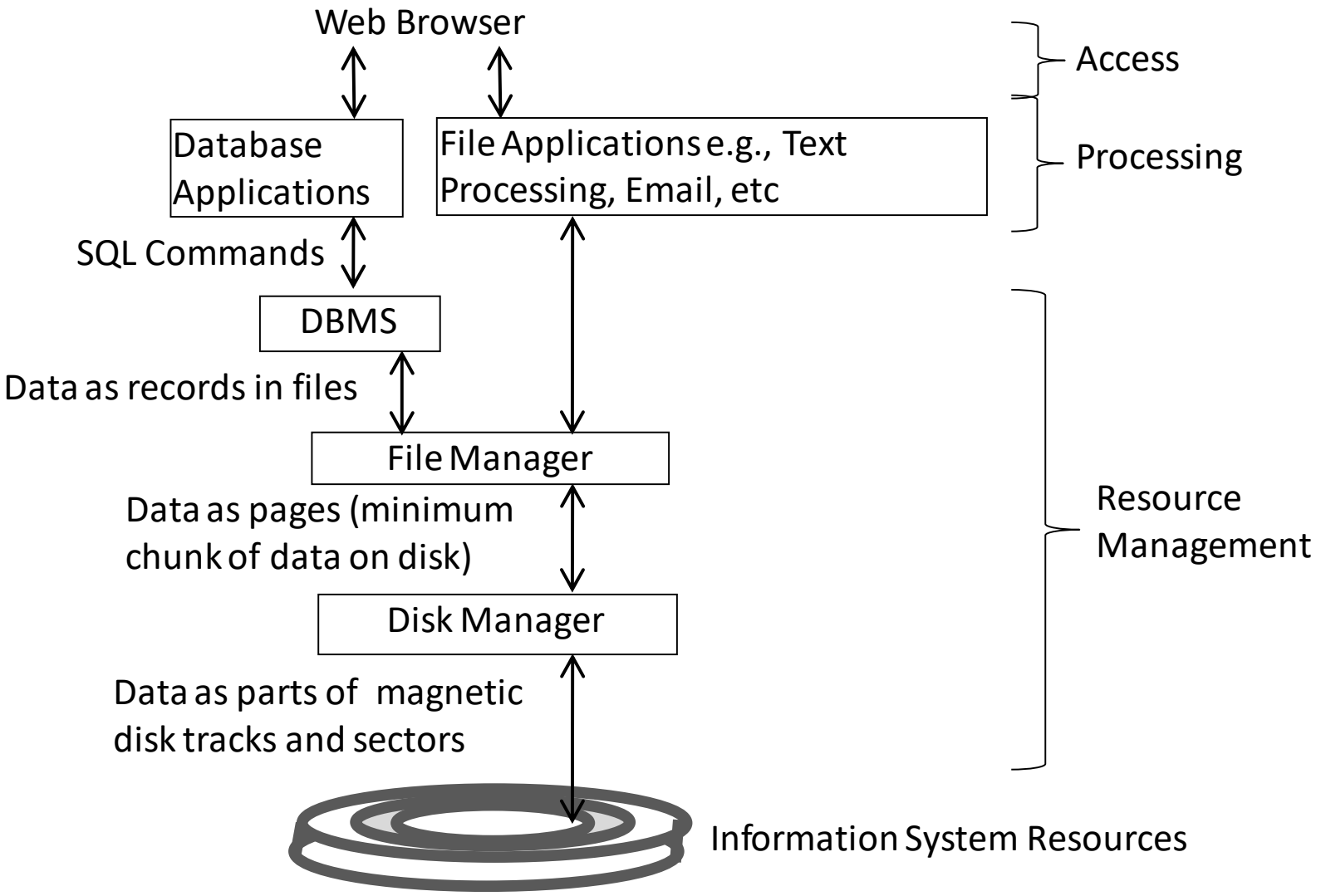
- System architectures focus on the idea of reducing complexity through both a separation of concerns using *modularisation* & *transparency*
- Two common criteria for modules:
  - *high cohesion*
  - *loose-coupling*
- Meyer (1998) uses five criteria for modularisation:
  - *Decomposability*:
  - *Composability*:
  - *Understandability:*
  - *Continuity*:
  - *Protection*:.

Reducing System Complexity using Abstraction (interoperability)

- Abstractions define those things that are important in a system

- Abstraction that simplifies the view or access to internal functionality to the outside, is also called an *interface*.

System View: Example of Abstraction



Reducing System Complexity using Abstraction (Transparency)

- Abstractions make transparent properties not needed by interactions
- Important types of transparency for distributed services include:
  - Access transparency:
  - Concurrency transparency
  - Failure transparency (Fault Tolerance)
  - Migration transparency
  - Scaling transparency
- In practice, ideal transparency of a single image for all resources, all the time, under all conditions is hard to achieve
  - Usually only when the distributed system is operating normally.

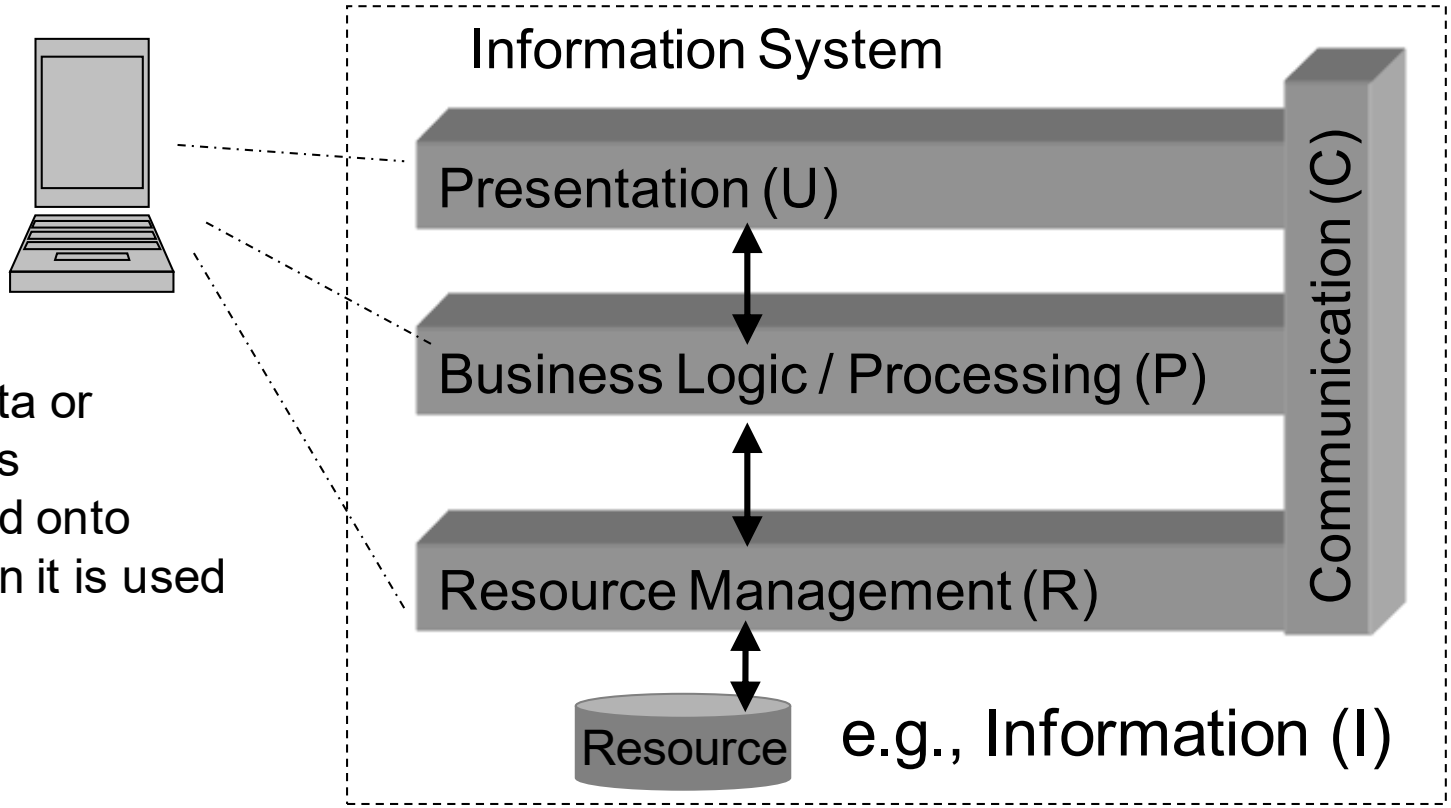
Reducing System Complexity using Abstraction (virtualisation)

- Abstractions alone do not necessarily support interoperability
  - Explain here
- *Virtualisation* provides a way to solve this limitation of abstraction:
  -
  - See later

Overview

- Smart Device and Service Characteristics
- Distributed System Viewpoints
- System Abstraction
- **Partitioning and Distribution of System Components √**
- Proxies and Middleware
- Service Oriented Computing (SOC) & Grid Computing
- Peer-to-Peer Systems (P2P)
- Service Provision Lifecycle □
- Service Discovery
- Service Invocation
- Service Composition
- MTOS, BIOS & VM

Partitioning & Distribution of System Components: None



Copy of data or applications downloaded onto device, then it is used off-line

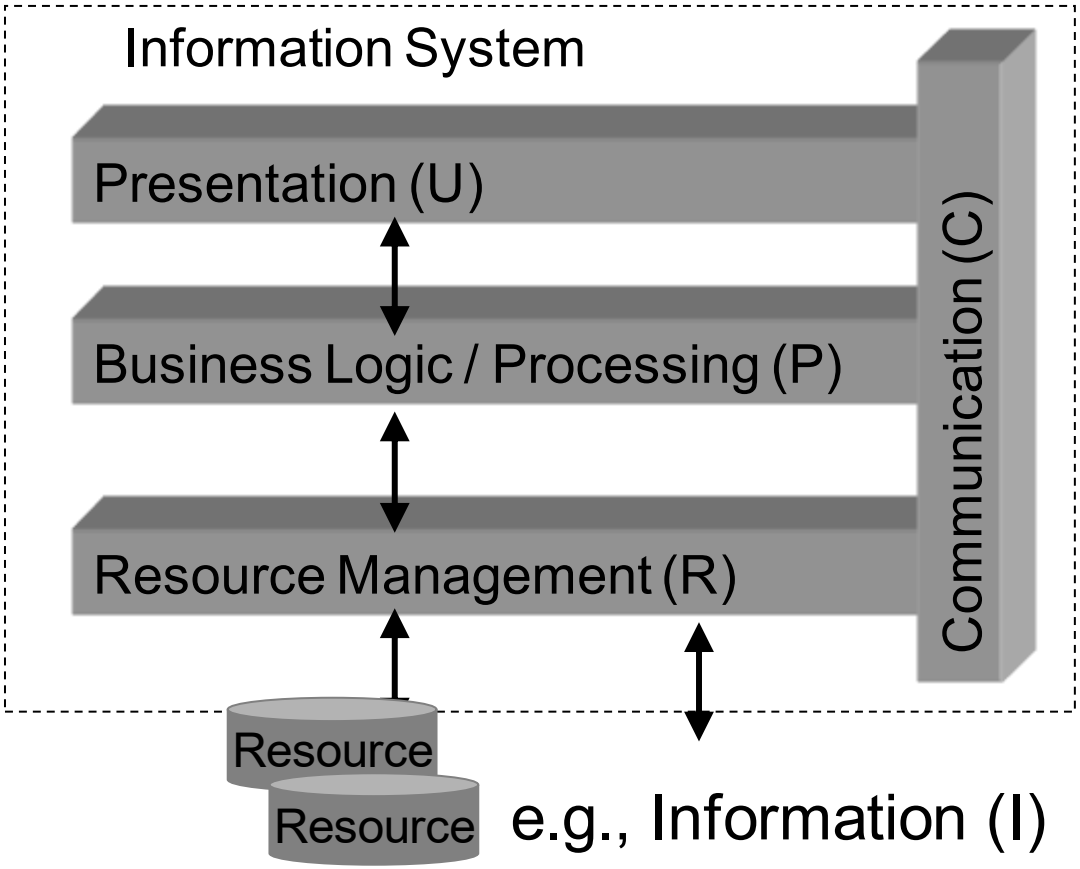
Partitioning & Distribution of System Components: None

- Advantages?

- Disadvantages?

Partitioning & Distribution of System Components

- Ex: how can we distribute these components?



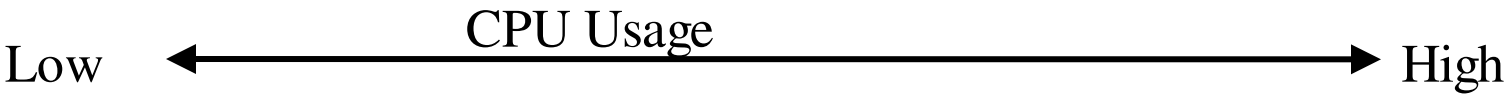
Partitioning & Distributing System Components

- Range of designs for partitioning and distributing services:

- Consider type of access device, resources, communication: several ways to distribute these, e.g.,
- *High resource access devices* can act self-sufficiently,
- *Low / poor resource access devices*

System Architectures: Partitioning Example

Discuss How to partition a 2 player Person versus Machine Chess Application in terms of a client-server design / for use on a mobile device



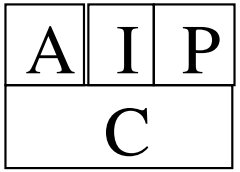
Architectures: Client Server model

- Asymmetric distributed computing model with respect to where resources reside and the direction of the interaction.

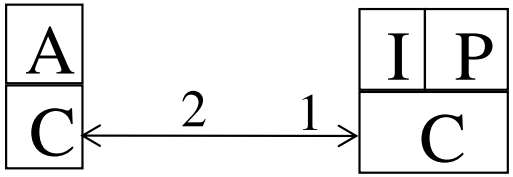
Client-server interaction is also asymmetric:

- Asymmetry benefits?

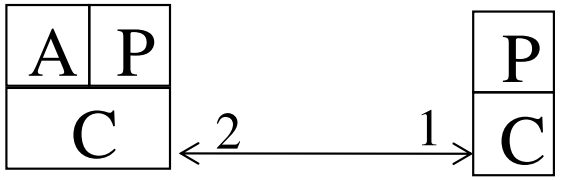
Partitioning and Distribution: Client-Server Model



Monolithic



Thin Client Servers



Fat Client Servers

Client Server Model

- System configuration (partitioning and distribution) depends upon:
  - network links;
  - local resources,
  - remote service availability;
  - type of application,
  - service maintenance model.
- Different degrees of resources on access devices (clients)
- Resource poor (thin-client server model):
  - reliance on external servers, network supports remote service access on demand

Client Server Model

- Processing needed to adapt content to different types of terminals
  -
- Thin-client server model is often considered to be easier to maintain
  -
- Thin-clients offer very limited application platform

Client Server Model

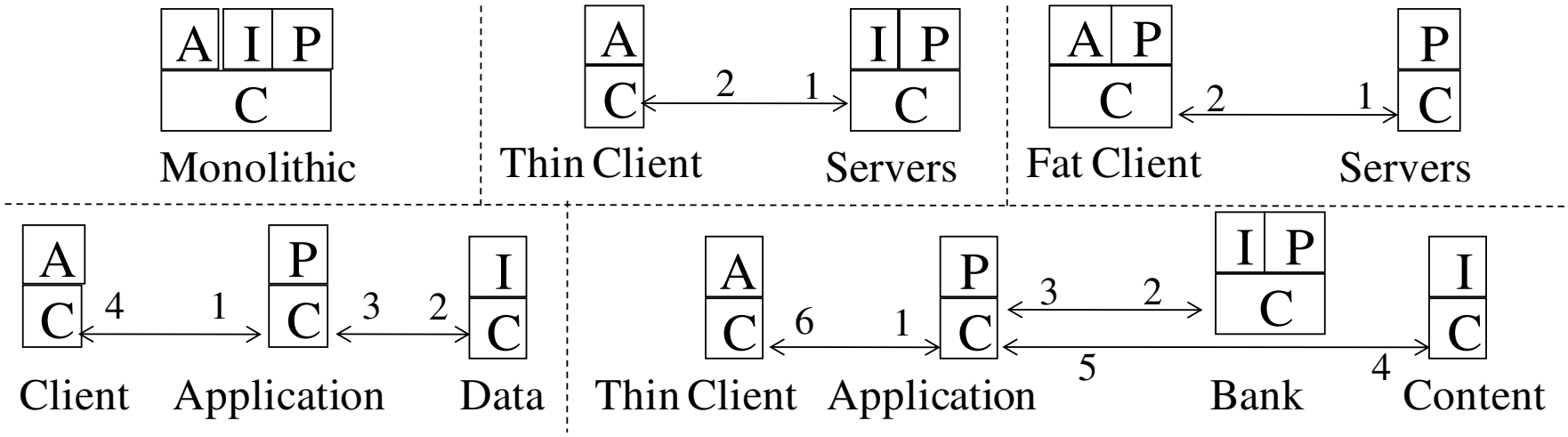
- How to cope with unreliable and low-performance networks using client-server model?
- Argues for a degree of self-reliance & use of local processing and data resources

- Fat client model is suitable when?

- Type of processing in access device depends on type of application.
  - E.g.,

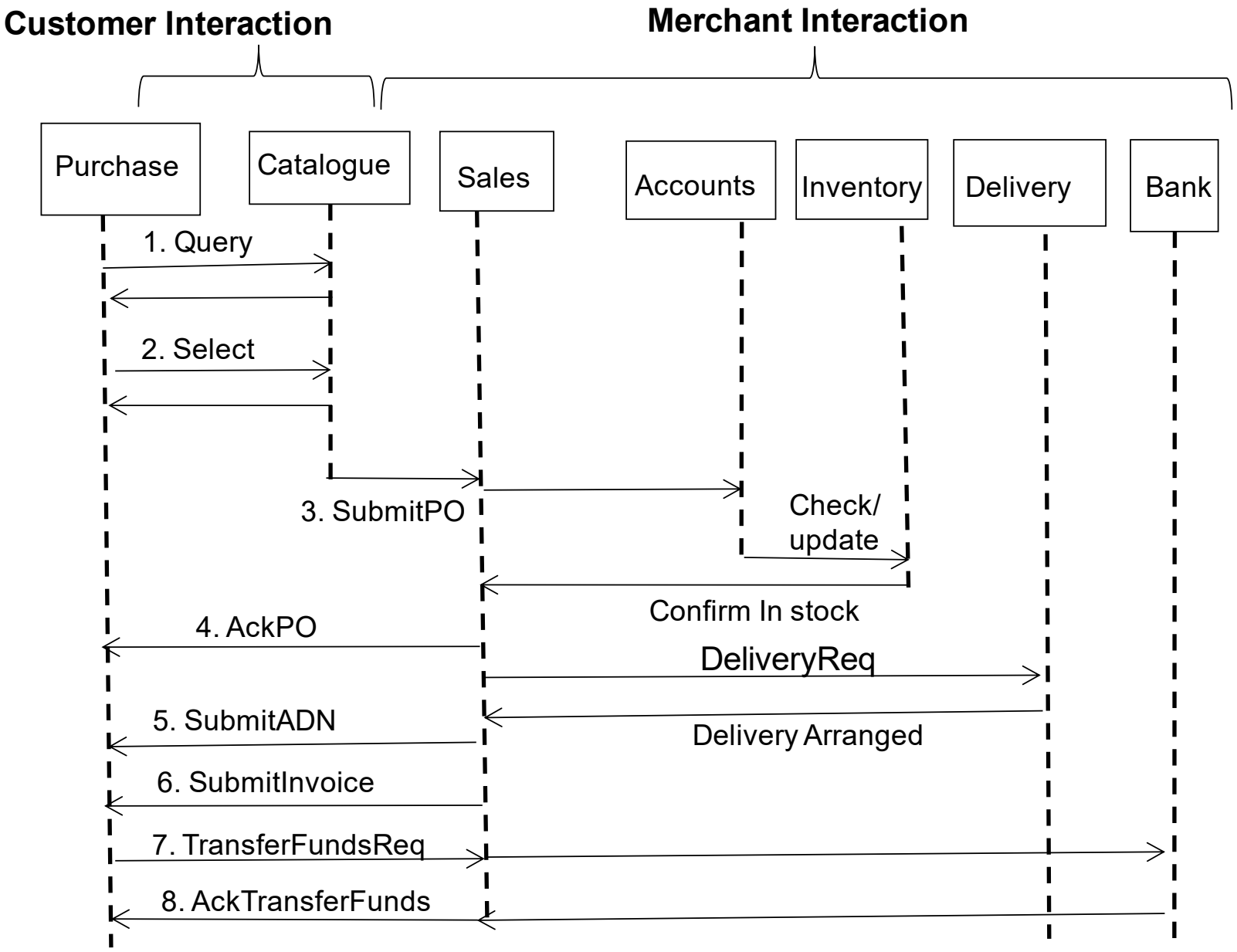
  - E.g.,

Partitioning & Distributing System Components: Summary of Models



Partitioning & Distributing System Components: Summary of Models

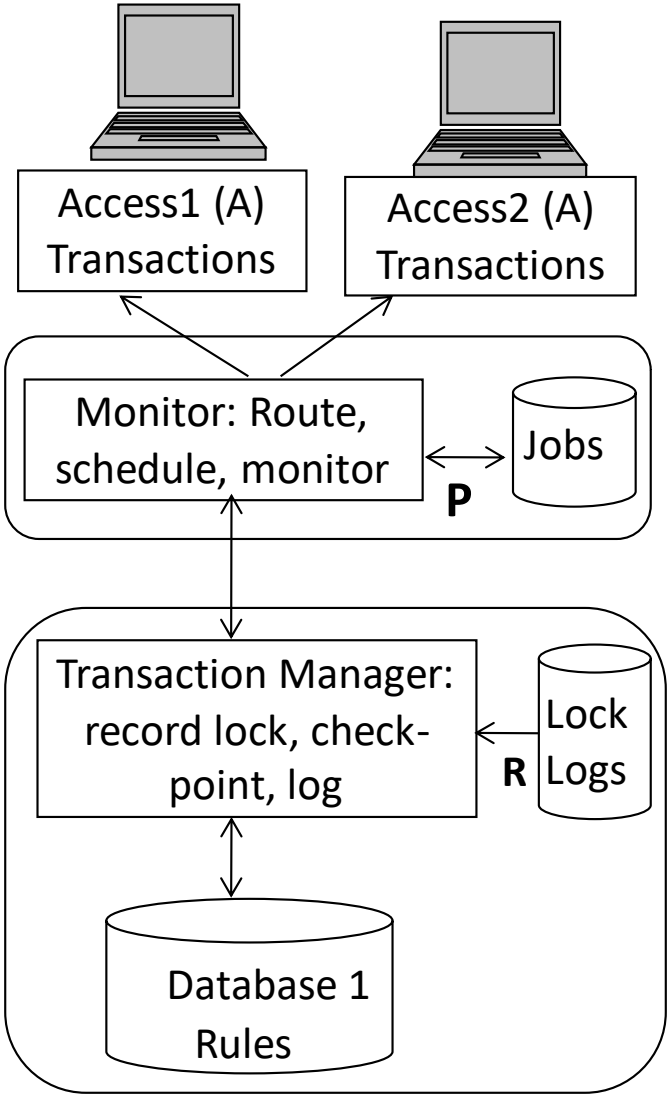
- Different designs for Information-based UbiCom systems:
  - based upon how their A, P and I components are distributed.
- Functions can be distributed over multiple different computer nodes or tiers:
  - 1-tier, monolithic system, appliance model:
  - 2-tier, thin-client server:
  - 2-tier, fat-client server model:
  - Multi-tier (3,4 … N-Tier) systems:


Customer Interaction
Merchant Interaction
Purchase
Catalogue
Sales
Accounts
Inventory
Delivery
Bank
1. Query
2. Select
3. SubmitPO
Check/
update
Confirm In stock
4. AckPO
DeliveryReq
5. SubmitADN
Delivery Arranged
6. SubmitInvoice
7. TransferFundsReq
8. AckTransferFunds

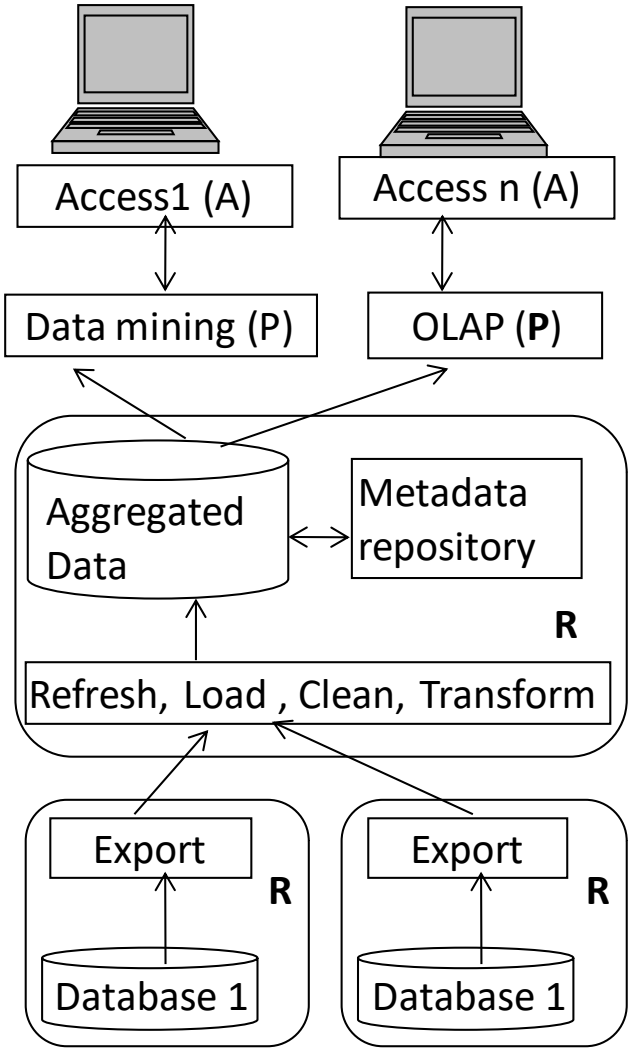
Partitioning and Distributed Data (D) Storage

- I, P, A and D can themselves be partitioned & distributed
- Examples of Partitioned & Distributed D
  - *Transaction Monitors* (TM): distributed data transactions;
  - *Data Warehouses*, centralised analysis of distributed data
  - *Distributed Databases*: distributed queries

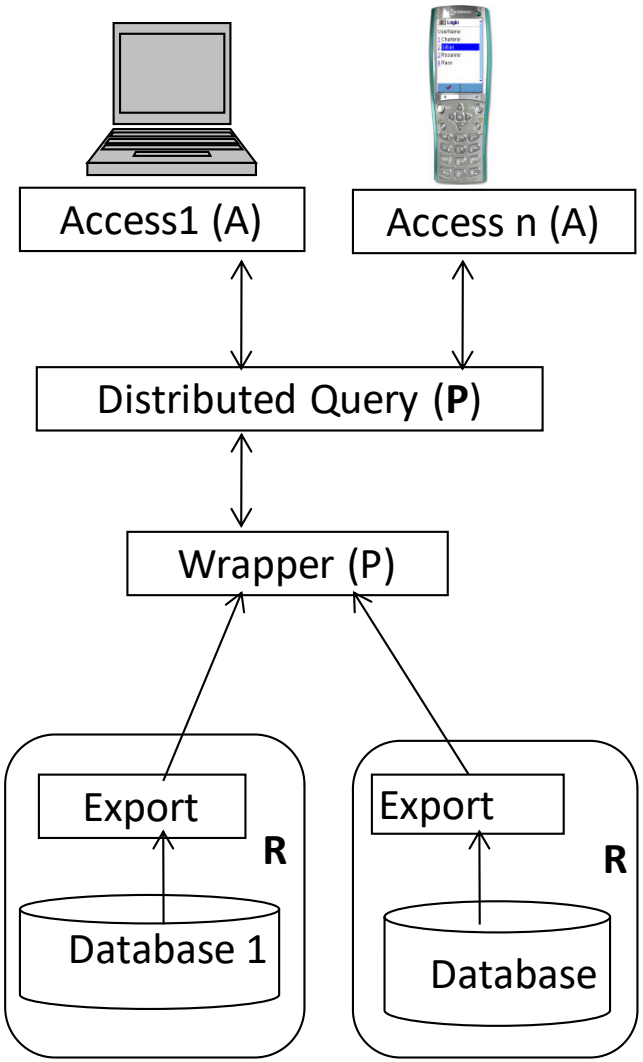
Distributed Data (D) Storage: Transaction Processing



Distributed Data (D) Storage: Data Warehouse



Distributed Data (D) Storage: Distributed Database



Distributed Processing

- Partitioning & distributing processing onto multiple CPUs
- Use for computation intensive tasks, e.g., ??
- Time gained in ↓ processing time must be > time to partition & distribute tasks, collect individual results & combine them.

Many different architectures

- Super-computers - specialised multiple CPU systems
- Clusters of networked MTOS computers, e.g., Grids.
- Multiple CPUs in MTOS computers. e.g., multi-core processor
- P2P computing
- Cellular computing

What about

- distributed UIs?
- Distributed communication?

Distributed Processing Architectures

- Examples can be added here

Overview

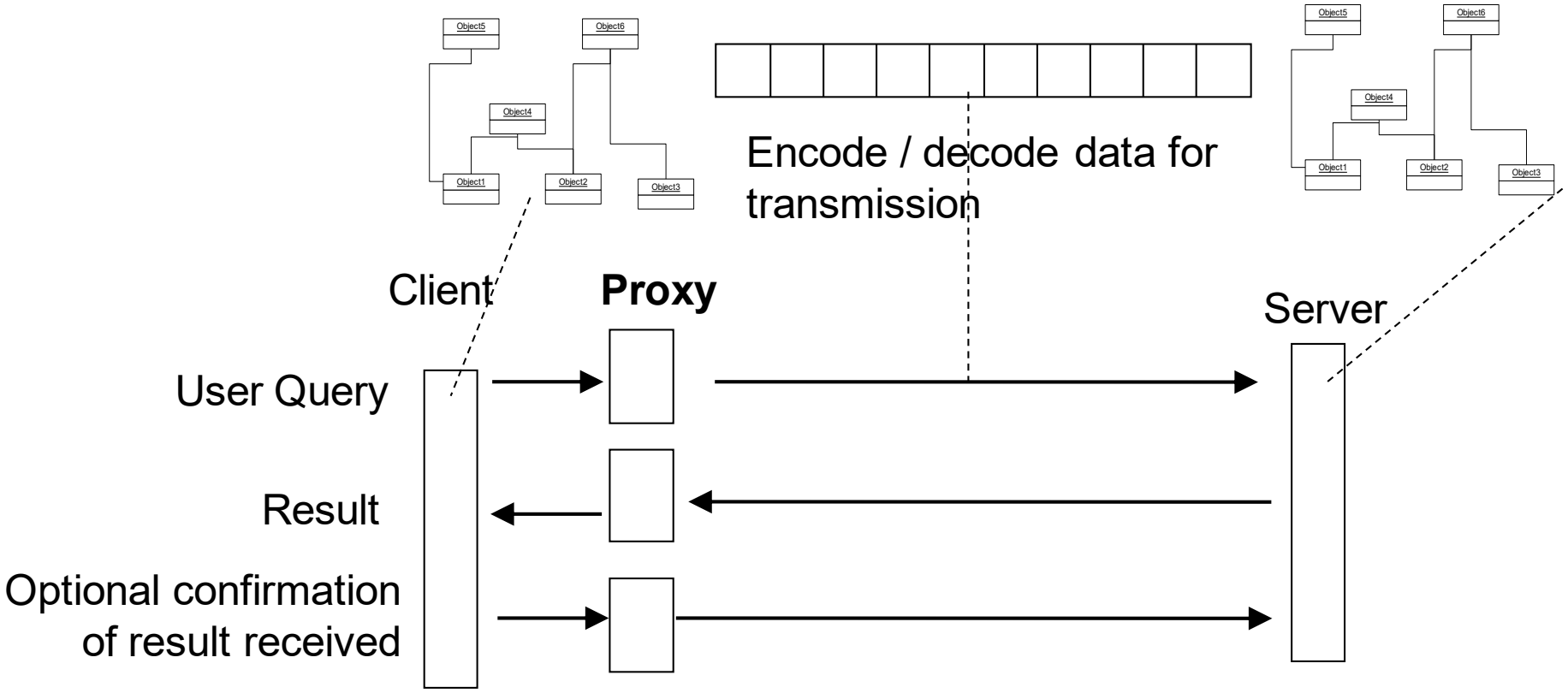
- Smart Device and Service Characteristics
- Distributed System Viewpoints
- System Abstraction
- Partitioning and Distribution of System Components
- **Proxies and Middleware √**
- Service Oriented Computing (SOC) & Grid Computing
- Peer-to-Peer Systems
- Service Provision Lifecycle □
- Service Discovery
- Service Invocation
- Service Composition
- MTOS, BIOS & VM

Proxy based Service Access

Advantages of using client proxies

- Some applications use a client proxy to simplify access processes in client, How?

Proxy based Service Access



Use of proxies to simplify network access by transparently encoding and decoding the transmitted data on behalf of clients and / or servers

Proxy based Service Access

What are the disadvantages of Proxy-based access?

Where does the proxy reside?

Middleware

- ↑ Variety & heterogeneity & complexity of services access
- Middleware introduced in between applications & OS to simplify access to services
- Middleware factors out set of generic services, e.g., database access, file system access etc. to make them:

- Advantages for Application?
  –
- Advantages for OS?

Middleware: Design Issues

- May be useful for applications to have an awareness of lower level interaction, for resource access not to be completely hidden by middleware.
- Why?

Middleware

Full

Application sees full ICT system interface, no Middleware used

Partial

Middleware handles some of the complexity in interfacing to ICT system

None

Middleware hides complexity of ICT system from application

Overview

- Smart Device and Service Characteristics
- Distributed System Viewpoints
- System Abstraction
- Partitioning and Distribution of System Components
- Proxies and Middleware
- **Service Oriented Computing (SOC) & Grid Computing √**
- Peer-to-Peer Systems
- Service Provision Lifecycle √
- Service Discovery
- Service Invocation
- Service Composition
- MTOS, BIOS & VM

Service Oriented Computing (SOC)

- SOA (Architectures) , also referred to as SOC (Computing)
- Services as computational or information processing components
  - That are autonomous and heterogeneous
  - Can run on different platforms
  - Are possibly owned by different organizations.

SOC Standards

Several different standards for SOC

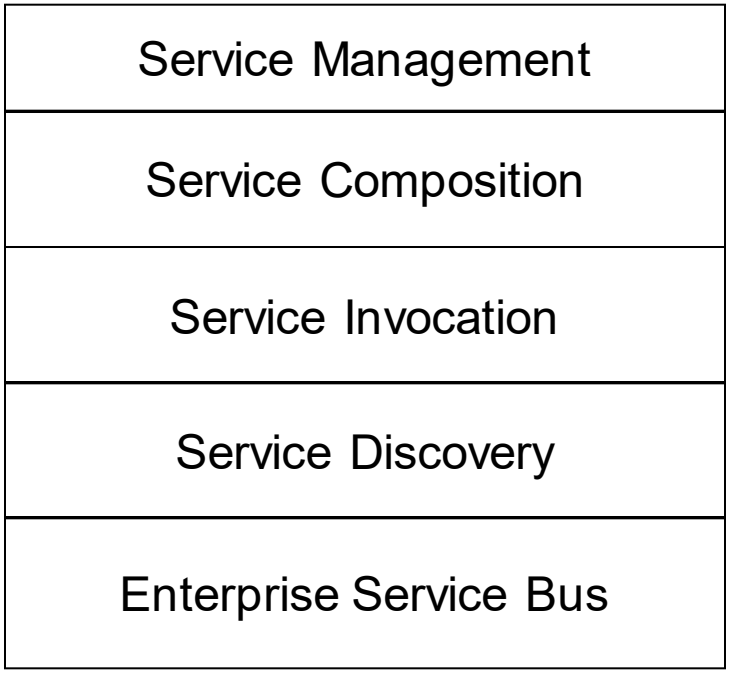
- (XML based) Web Services
- Computer Grids OGSI
- OASIS SOA RM
- Open Group SOA Working Group
- Semantic Web Services?

Service Oriented Computing (SOC)

Notion of service characterised by:

- *Descriptions*:
- *Outcomes*:
- *Offers:*
- *Competency*:
- *Execution*:
- *Composition*:
- *Constraints or policies*:

Service Oriented Computing (SOC)



SOC: Grids

- Grid computing: distributed systems that enable:
  –
- (Early) Grid computing system design tends to focus on high performance computing rather than fault-tolerance & dynamic ad hoc interaction,

SOC: Grids

Three main types of Grid system occur in practice:

- *Computational Grids*:
- *Data Grids*,
- *Service Grids:*

SOC Grid

- GRID System design can be discussed in more detail in a few slides

Overview

- Smart Device and Service Characteristics
- Distributed System Viewpoints
- System Abstraction
- Partitioning and Distribution of System Components
- Middleware
- Service Oriented Computing (SOC)
- **Peer-to-Peer Systems √**
- Service Provision Lifecycle √
- Service Discovery
- Service Invocation
- Service Composition
- MTOS, BIOS & VM

Peer-to-Peer Systems (P2P)

P2P can be defined as:

- distributed systems consisting of interconnected nodes
- able to self-organize into network topologies
- with the purpose of sharing resources such as content, CPU cycles, storage and bandwidth,
- capable of adapting to failures and accommodating transient populations of nodes
- while maintaining acceptable connectivity and performance
- without requiring the intermediation or support of a global centralized servers or authorities.

P2P Benefits

- What are the main benefits?

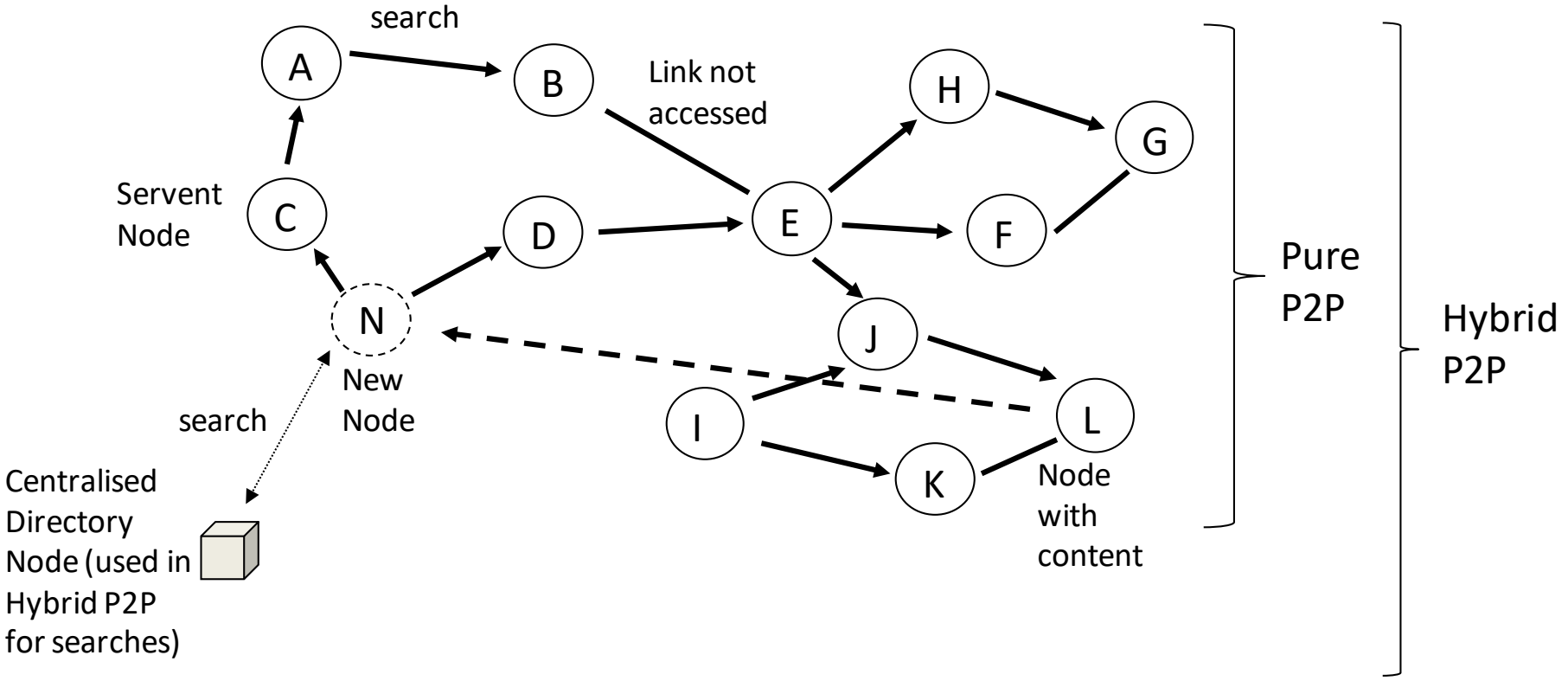
P2P System Design Challenges (Limitations)

- What are the main challenges or limitations?

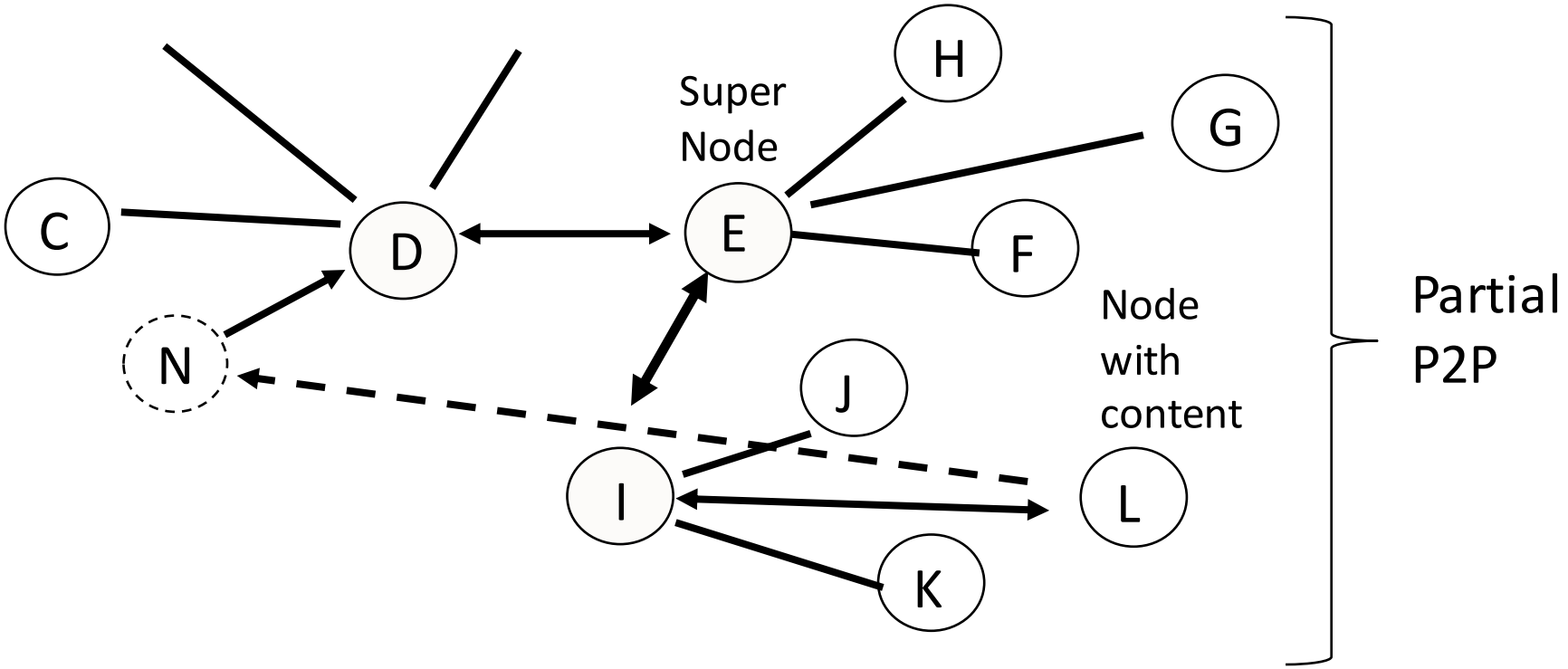
P2P System: Types

- 3 main types of P2P system depending on the types of computer nodes:
- *Pure P2P*
- *Partial P2P*:
- *Hybrid P2P* networks,
- 3 basic content access processes in distributed systems to:
  - identify nodes,
  - register content provision nodes,
  - search & retrieve content

P2P System Types: Pure, Hybrid



P2P System Types : Partial P2P



P2P System Types

- Can also be grouped into two main types of topologies for P2P systems that overlay the underlying physical network:
- *Unstructured overlay networks*

- *Structured overlay networks*

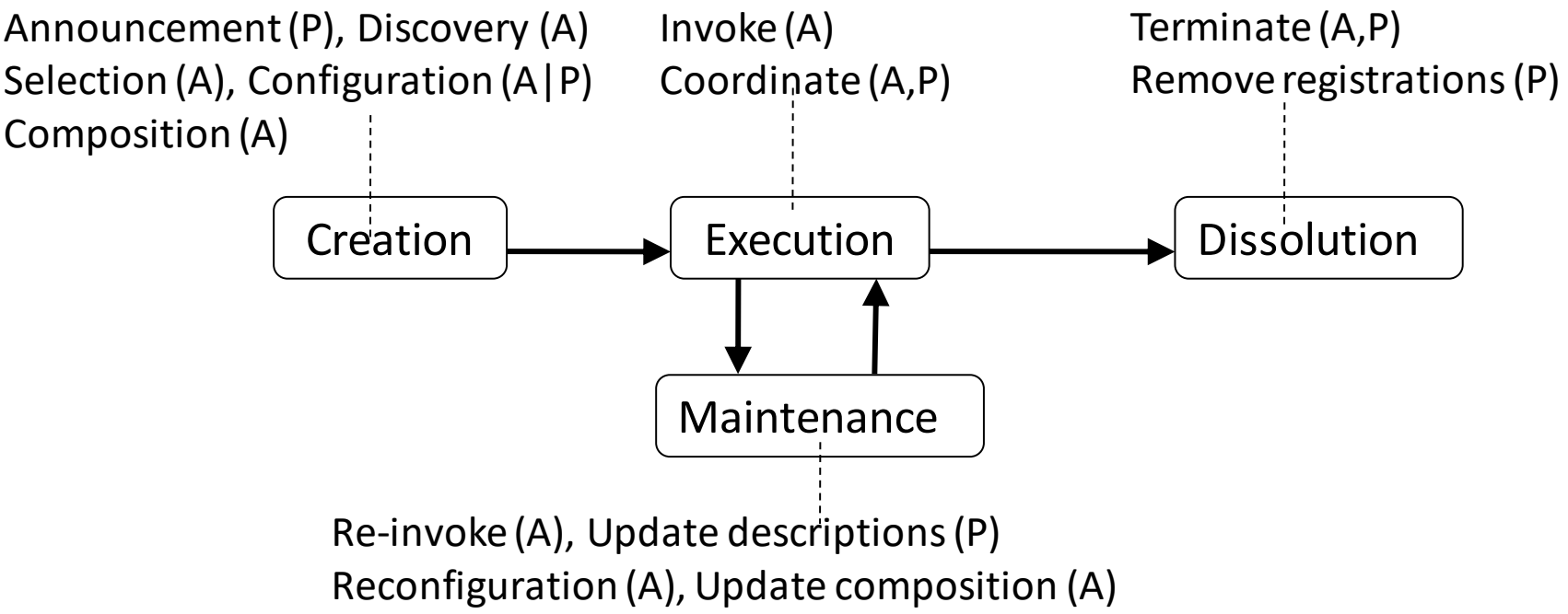
Overview

- Smart Device and Service Characteristics
- Distributed System Viewpoints
- System Abstraction
- Partitioning and Distribution of System Components
- Proxies & Middleware
- Service Oriented Computing (SOC)
- Peer-to-Peer Systems
- **Service Provision Lifecycle √**
- Service Discovery
- Service Invocation
- Service Composition
- MTOS, BIOS & VM

Service Provision Life-cycle

- Service creation
- Service operation
- Service maintenance phase
- Service dissolution

Service Provision Life-cycle



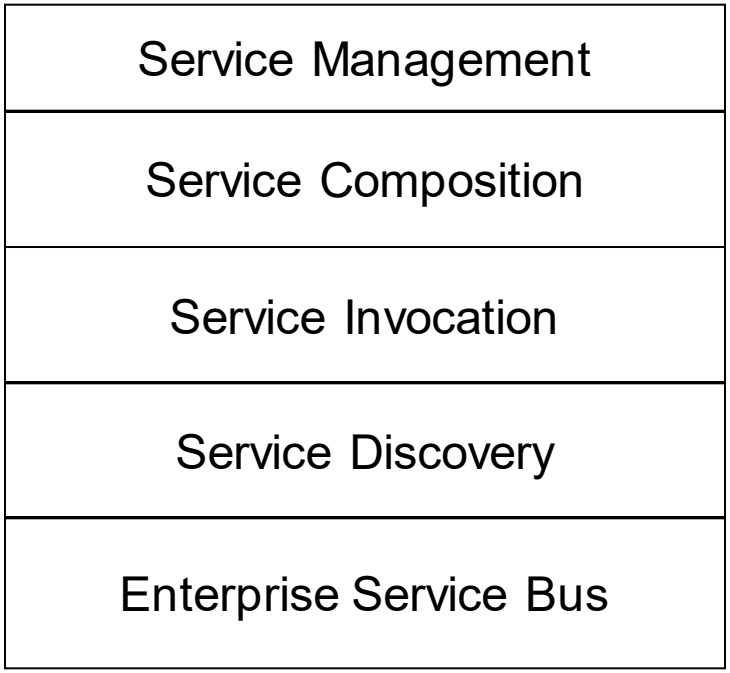
Service Provision Life-cycle

- Exercise: Consider creation, operation, maintenance, dissolution for the following types of devices & services:
- Laptop / Internet
- Set-top box audio-video receiver
- Mobile phone
- Email Service

WS SOA Support for Service Life-cycle

- Web Services (WS) support machine-to-machine interaction
- Service Interfaces are machine-processable, syntactical
- WS SOAs consist of many possible WS protocols depending on the application and service requirements.
- Core WS SOA protocols are:
  - SOAP:
  - WSDL:
  - UDDI :

Service Oriented Computing (SOC)



Service Life-Cycles / SOAs in More Details

- Instructors can add further slides here to explain SOAs in more detail or delete this slide.

Overview

- Smart Device and Service Characteristics
- Distributed System Viewpoints
- System Abstraction
- Partitioning and Distribution of System Components
- Proxies & Middleware
- Service Oriented Computing (SOC)
- Peer-to-Peer Systems
- Service Provision Lifecycle
- **Service Discovery √**
- Service Invocation
- Service Composition
- MTOS, BIOS & VM

Service Announcement, Discovery, Selection and Configuration

- Service discovery scope & functions depends on design.
- What's involved in service discovery?

- Which happens first in service discovery?
  - Network discovery

Network Discovery

What Is it? Why do we need it?

Which Network Protocols support Network Discovery?

Network Discovery: Zeroconf

- Zero Configuration Networking (Zeroconf)
  –
- Allows inexpert users to connect computers, networked printers etc together & expect them to work automatically.
- Without Zeroconf or something similar, need to?
  –
- Zeroconf currently solves automating three tasks

Network Discovery: dynamically assigning IP addresses

- Both IPv4 and IPv6 have standard ways of automatically choosing / assigning IP addresses.
- IPv4 uses the 169.254.any, link-local set of addresses, see RFC 3927.
- IPv6, zeroconf, see RFC 2462 can be used.
- 2 similar ways of figuring out which network node has a certain name.
  - Apple's *Multicast DNS (mDNS)*
  - Microsoft's *Link-local Multicast Name Resolution (LLMNR)*

Dynamic Service Discovery

- Dynamic versus Static service discovery
- Allow requesters to change providers , Why? Vice-versa?

- What is involved in Dynamic Service discovery ?

Dynamic Service Discovery: Pull versus Push

- 2 main approaches : push or pull.

Pull: How does it work?

Discovery Services: Push

Push: How does it work?

Discovery services Push: Design

Broadcast / Announcements can be designed to occur:

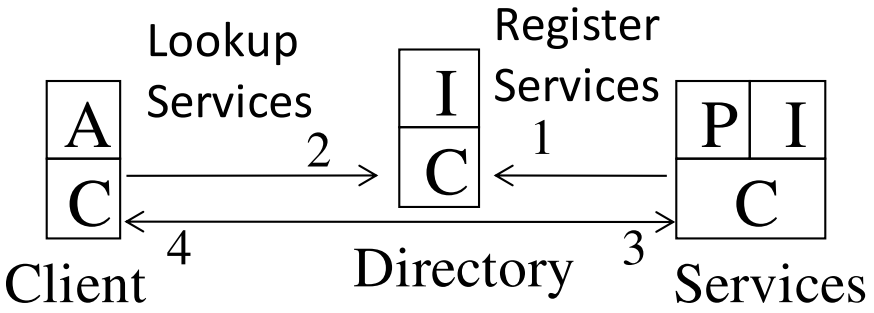
- Periodically irrespective of whether any audience exists or not;
- Only when any kind of audience is available;
- Only when a specific type of audience is detected
  - multicast versus broadcast.

Discovery Services: Pull vs. Push

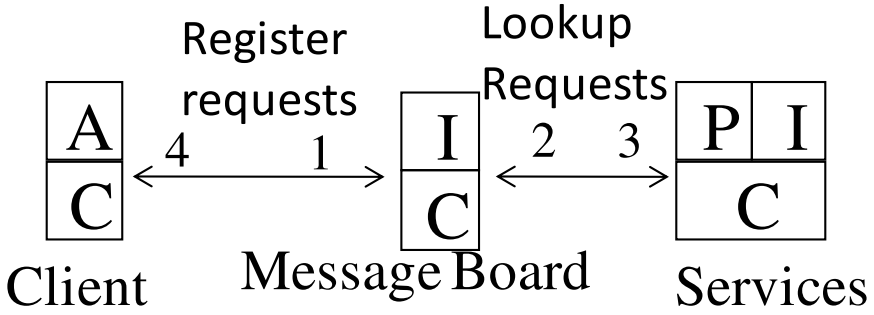
- Advantage of Pull?

- Disadvantage of Pull?

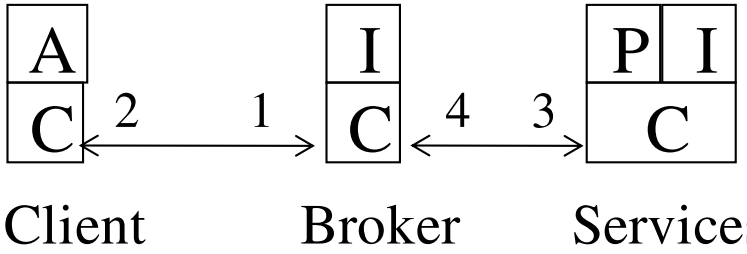
Service Discovery Interaction Patterns



Directory Service



Blackboard



Broker

External versus Internal service selection

- External $\rightarrow$ services to satisfy request exist in virtual environment to the ICT system, rather than internally within the system itself.
- How does a requester of a service know if it needs someone else to perform the service

- How does a service requester choose between external service versus internal service invocation when both are available?

External versus Internal service selection

- Process of establishing whether or not a service exists internally can involve
  - Self-descriptions
  - Self-awareness
  - Reflection
- See chapter 10

Semantic Web (SW) and Semantic Resource Discovery

- Why is Syntactic level matching and discovery challenging in pervasive environments?

- What are benefits of Semantic matching rather than syntactic service matching?

- Service Requests Benefits?
  –

Semantic Web (SW) and Semantic Resource Discovery

- SW represents resources using RDFS and OWL
  –
- SW defines much richer XML based data structures and relationships
  –
- Design choices:

Semantic Web Service Models

- Instructors can decide to explain this in more detail in the following slides or delete this slide.

Overview

- Smart Device and Service Characteristics
- Distributed System Viewpoints
- System Abstraction
- Partitioning and Distribution of System Components
- Proxies & Middleware
- Service Oriented Computing (SOC)
- Peer-to-Peer Systems
- Service Provision Lifecycle
- Service Discovery
- **Service Invocation √**
- Service Composition
- MTOS, BIOS & VM

Distributed Service Invocation

- Specifying an application protocol in terms of a set of service descriptions of service actions is often insufficient to invoke a service. Why?

Distributed Service Invocation

- Design of remote interaction across different computer nodes differs from design of local process interaction within the same computer node
- Why?

Distributed Service Invocation: Ordered Service Actions

- Service requesters may not know:
  - in what order to invoke service actions
  - how to handle out of order message sequences in a process without terminating service processes.
- Interaction in the process coordinated needs to be coordinated.
- Often coordination may be hard-coded into each service API and under the control of the provider.
  - Makes the coordination of multiple services inflexible.

Distributed Service Invocation: Ordered Service Actions

- Clients often need to invoke not just individual service actions in isolation but to invoke a whole series of service interactions as part of a business process
- Multiple heterogeneous processes often need to be interleaved:.
- Network Transmission may or may not maintain order of action messages when these are sent
- Complex to design in order to make remote communication (remote procedure calls (RPC) or remote method invocation (RMI) look like local calls , e.g., need parameter marshalling

Distributed Service Invocation: Fully Ordered Service Actions

- Two types of design based upon service action ordering
  - Full versus Partial

Fully ordered system processes of actions

- Specify actions executed as fixed sequences of actions.
- Earlier actions in sequence output data used by later ones
- Control of flow may contain some flexibility in terms of branches, conditions and loops.
- Example uses?
- Suitable for ??
- Less suitable for ??

Distributed Service Invocation: Partially Ordered Actions

- Two types of design based upon action ordering

2. Partially or non ordered system processes of actions Non-ordered System

- Specify action triggers (events) and action (responses, handling)
- Don't fully order these, may partially order these?
- Example Uses
  - ??
- Suitable for open dynamic environments

- See Chapters 8 and 9.

Distributed Service Invocation: fully versus partially ordering

- Advantages and disadvantages of full action ordering?

- Advantages and disadvantages of partial action ordering?

Service Invocation: Separating Coordination & Computation

- Should coordination mechanisms be separated from computation mechanisms.?

This supports several key benefits:

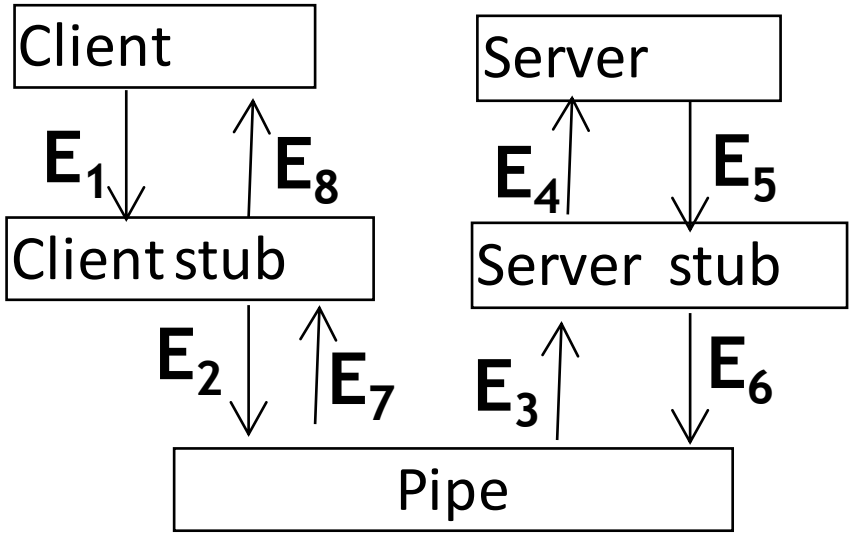
- Portability
- Heterogeneity
- Flexibility

Distributed Service Invocation Coordination Models

Designs for distributed interaction include:

- (*Remote) Procedure Calls / object-oriented Remote Method interaction*:
- *Layered model*:
- *Pipe-filter model*
- *Event-driven Action or EDA Model*:
- *Shared data repositories*:

Distributed Service Invocation Data Model: RPC model

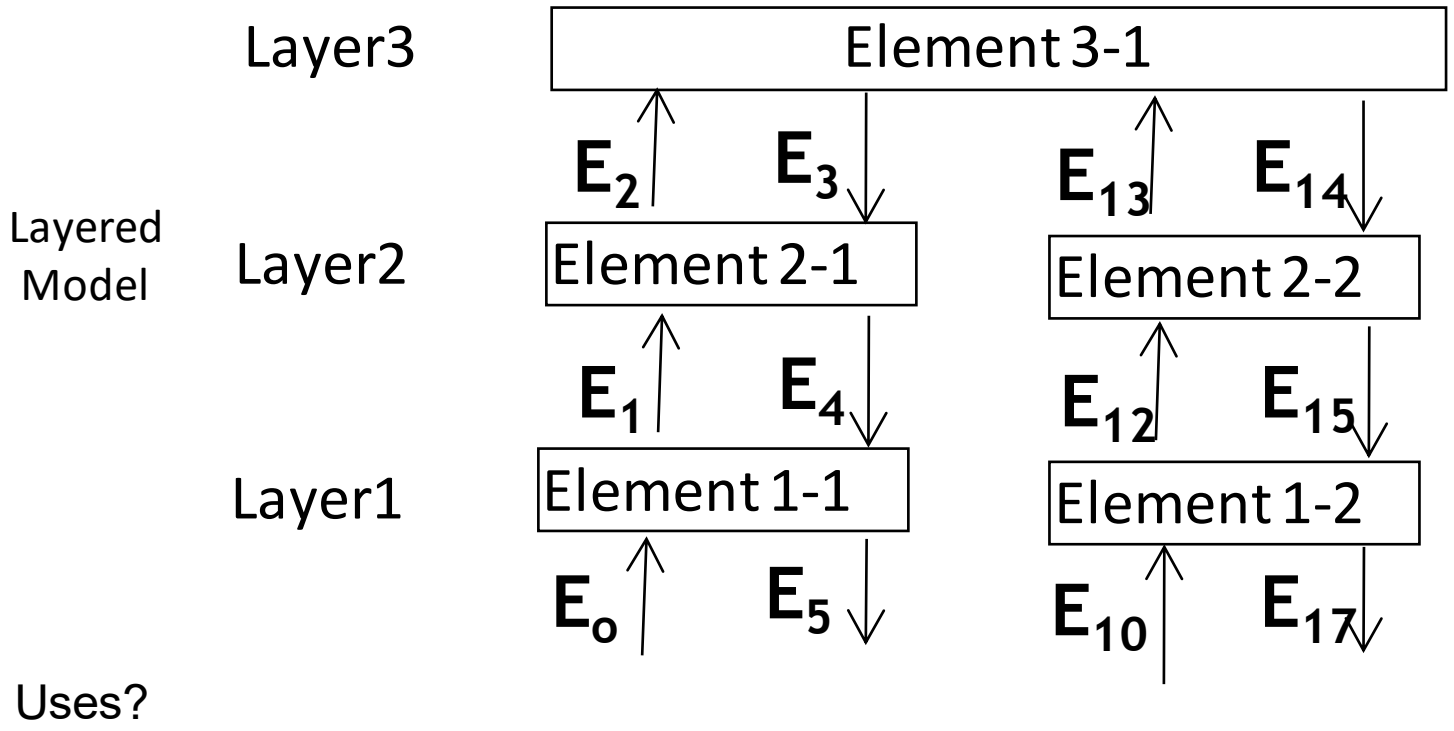


Uses?

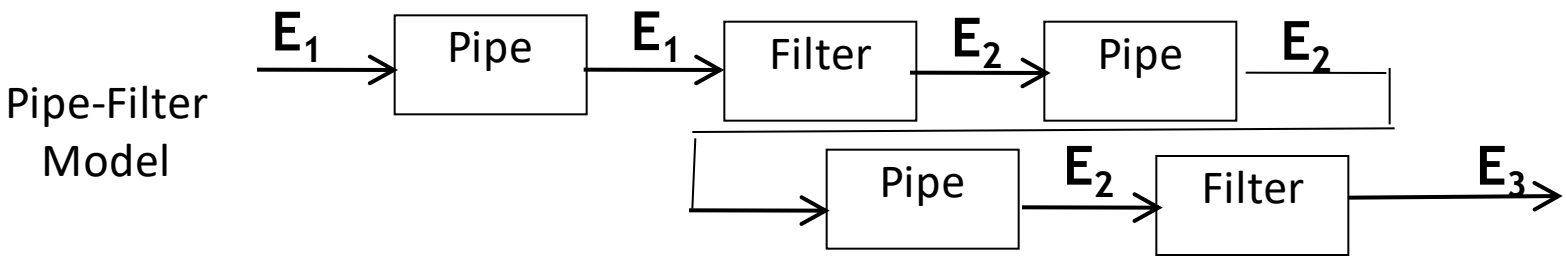
Distributed Service Invocation Data Model: RPC model

- For each type of service invocation data model we can give more detail about how the interaction model works
- (Remote) Procedure Call Model – makes remote calls look like local calls
- etc

Distributed Service Invocation Data Mode: Layered Model



Distributed Service Invocation Data Model: Pipe-Filter Model



Uses?

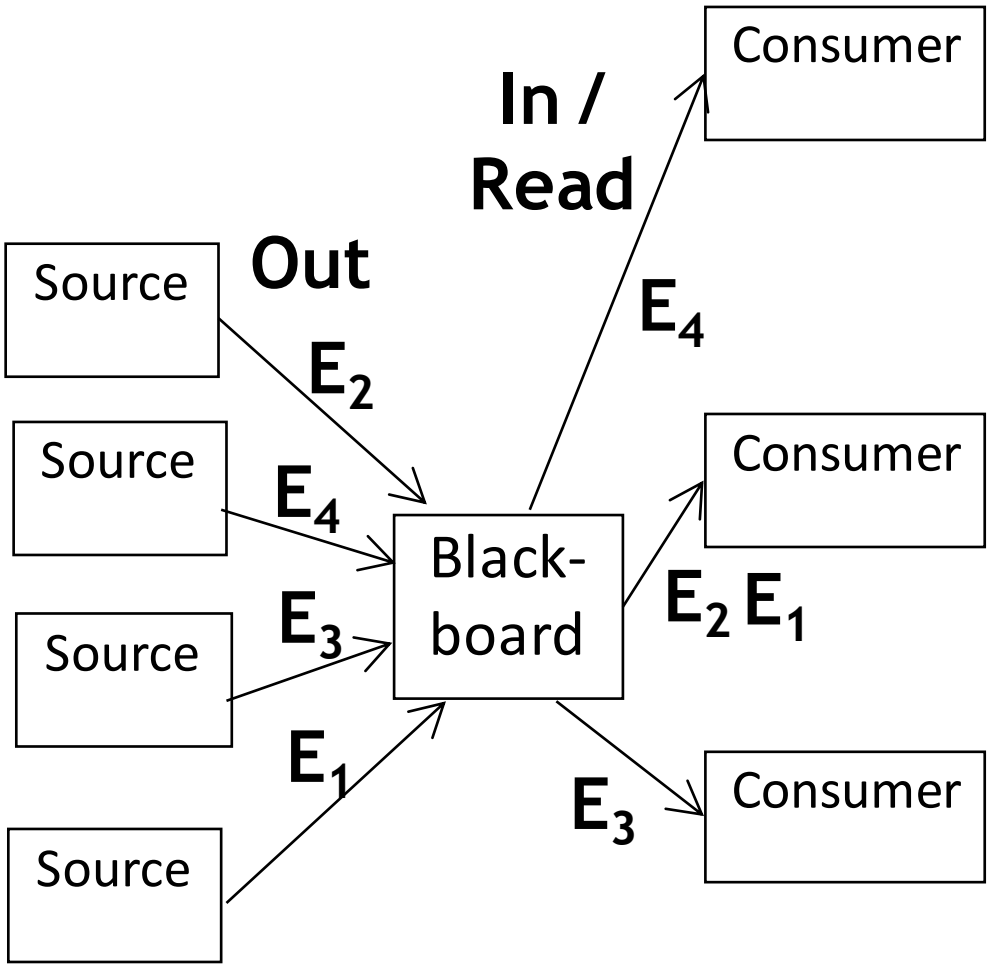
Distributed Service Invocation Data Model: Shared Data Repository

- 2 participants communicate by leaving messages for others via some shared intermediary
- Examples
  - ??
- Shared repository system consists of two types of components:
  - central data structure represents the current state
  - collection of independent components operate on central data store.
- 2 major sub-types of coordination depending on:
  - if transactions in an input stream trigger the selection of executing processes, e.g., a database repository
  - if the current state of the central data structure is the main trigger of selecting processes to execute, e.g., a blackboard repository.

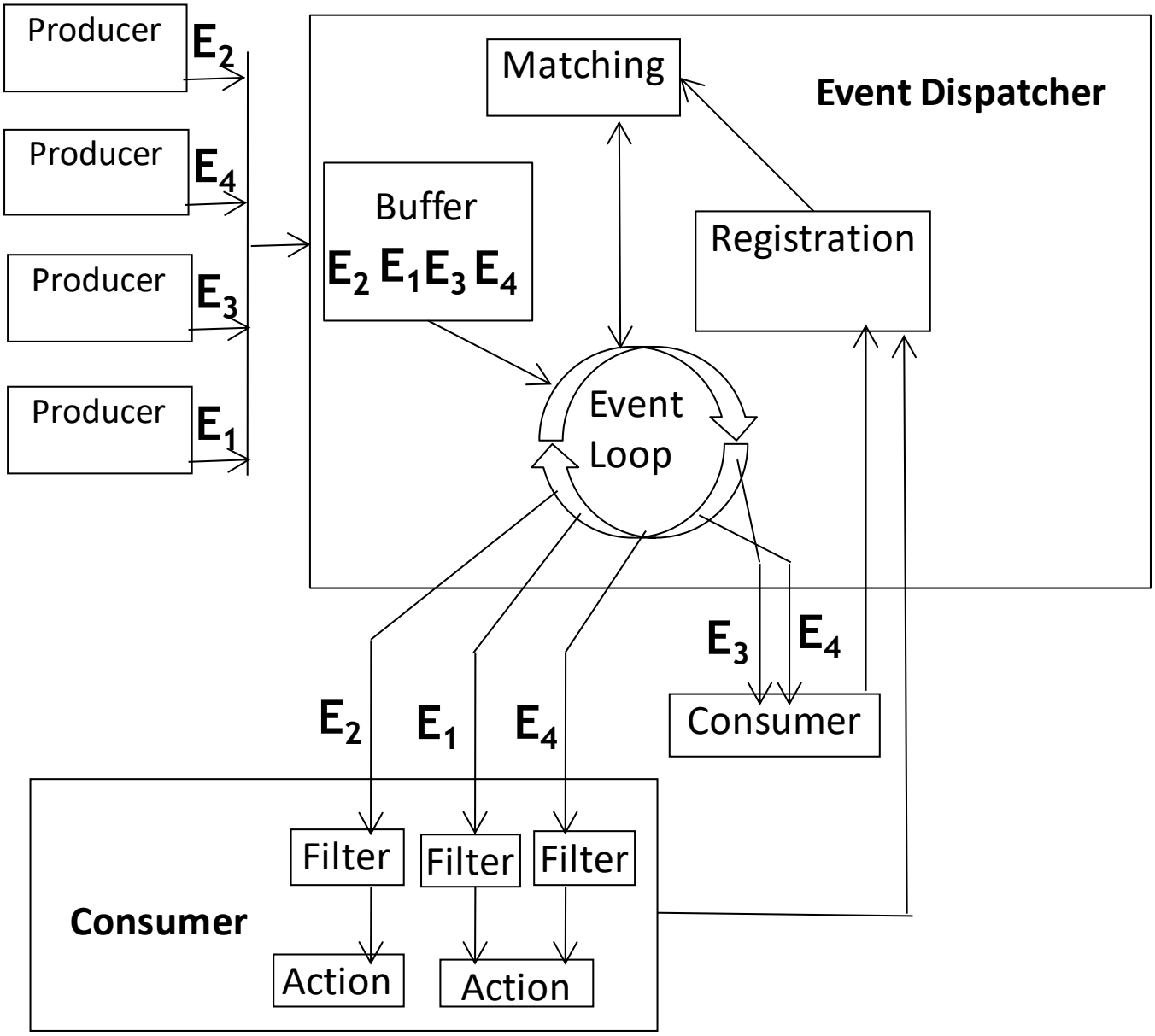
Shared Data Repository: Blackboard

- Represents & stores data created & used by other components.
- Data is input a repository from data producers.
- Data is output from a repository to data consumers.

Shared Data Repository: Blackboard



Service Invocation Data Model: EDA



Distributed Service Invocation Data Model: EDA

- Event-Driven Architectures or EDA
- EDA model important design for SOC and MOM Architectures
- Event is some input such as a message or procedure call of interest
- EDA is also known as *Publish-and-Subscribe interaction*.
- Some nodes publish events while others subscribe to being notified when specified events occur

Distributed Service Invocation Data Model: EDA

- A Few events may be significant
  - .
- Event may cause some predefined threshold to be crossed
  - .
- Event may be time-based,
  - .
- External events can trigger services. Services may in turn trigger additional internal events,
  - .
- Many events may not be significant

Distributed Service Invocation Data Model: EDA Challenges

- Design challenges complexity of EDA?
  - Event floods:
  - EDA generally have no persistence
  - Can be difficult to keep things running through a failure,
- Solutions
  - Prioritising events
  - Event persistence
  - Event coordination.
  - Highly selective event generation and transmission

Overview

- Service Provision Lifecycle
- Service Discovery
- **Service Invocation √**
  - Coordination Models
  - **On-demand Service Invocation √**
  - Volatile Service Invocation
  - ESB versus MOM
- Service Composition

Blackboard versus Event Driven

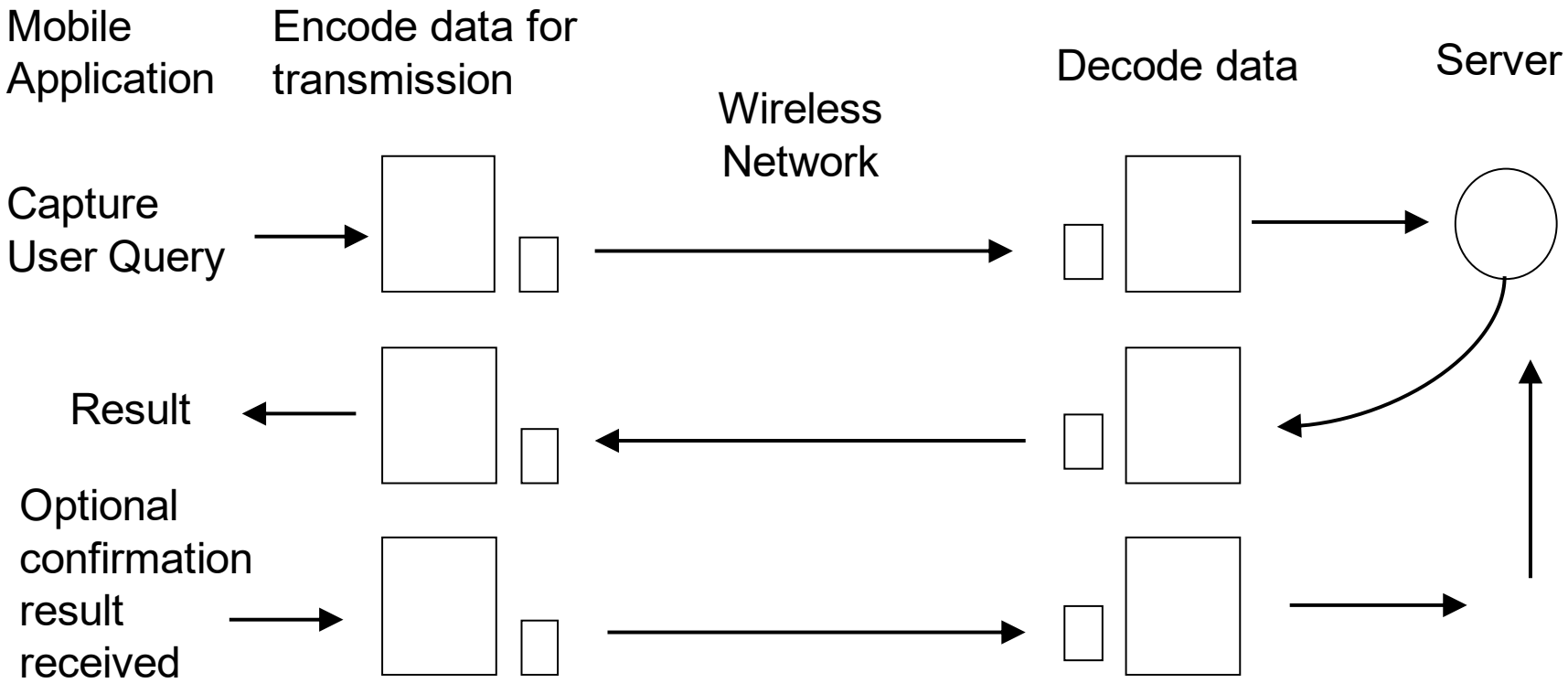
Compare and contrast?

On-Demand Distributed Service Invocation

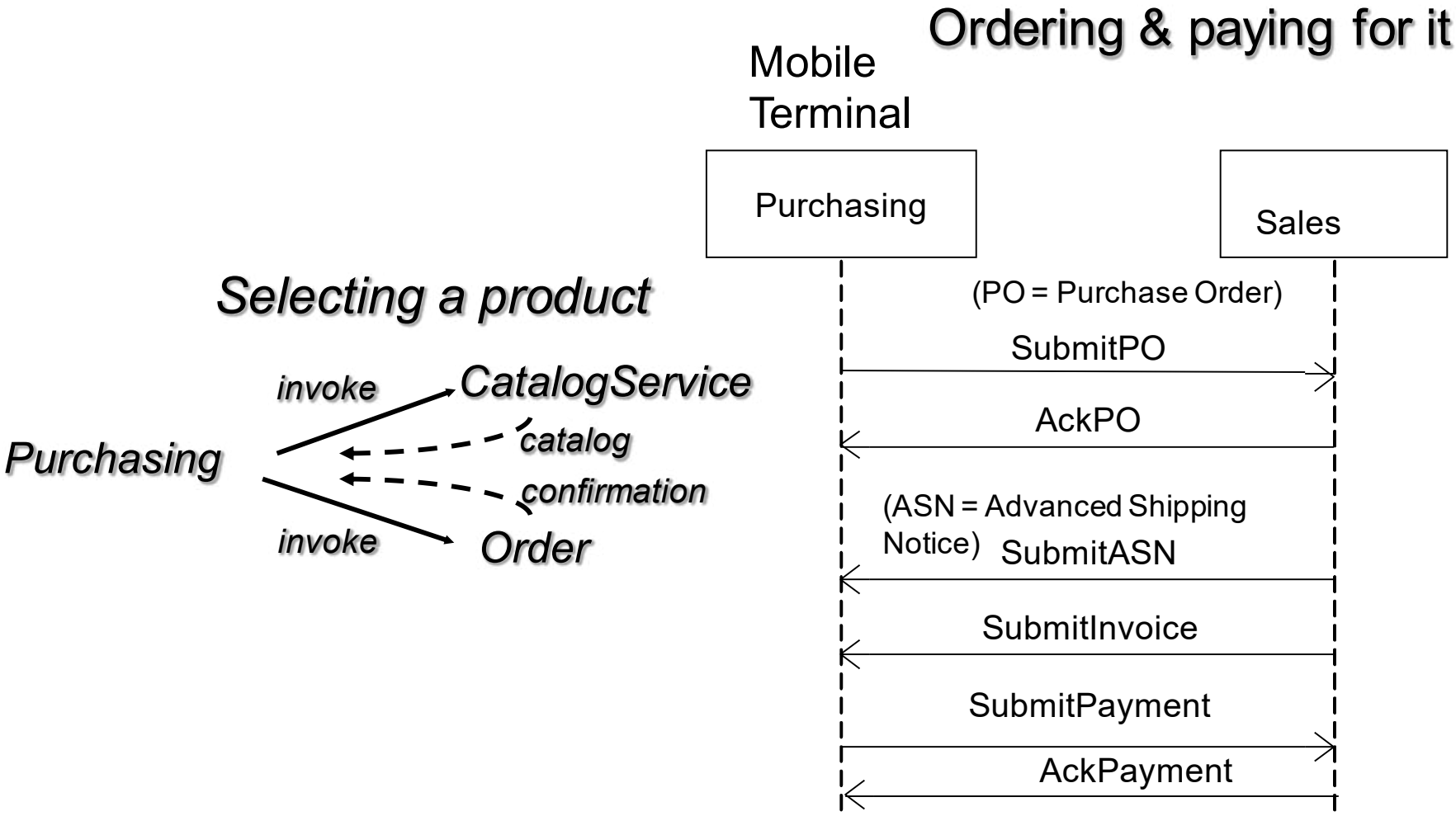
- On-demand: remote service access whenever needed
- Some data created locally & stored or processed remotely
  - E.g., ??
- Some data is stored remotely & accessed locally
  - E.g., ??
- Remote service invocation may involve single read or write data operations
- Remote service invocation may involve multiple read or write data operations,
  - E.g., ??
- Multiple service actions may be integrated into a whole
  - E.g., ??

On-Demand Distributed Service Invocation

Service invocation e.g., what is the best flight given these constraints?

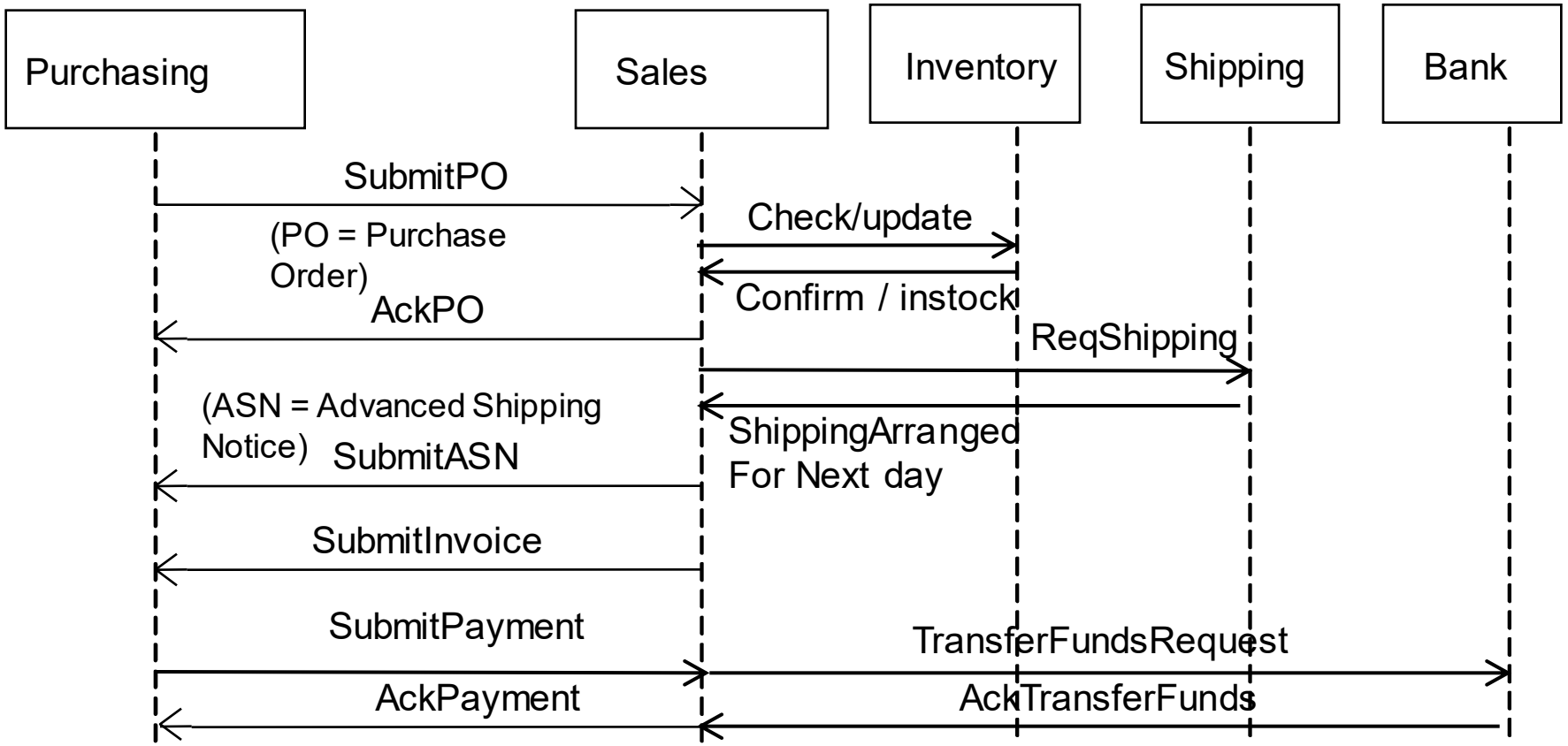


On-Demand Distributed Service Invocation can be Complex



On-Demand Distributed Service Invocation can be Complex

Ordering & paying for it



On-demand Service access

- On-demand service interaction suited to thin client interaction. Why?
- Suitable for small foot-print / low resource / devices or terminals
- Back-end server required continually throughout the transaction
- Server-side processing used to
  - Process transaction
  - Dynamic Server-side device profiling of clients

On-Demand Services: Request-reply, Pull Design

- Request/response is “pull-based” approach
- Client, requires instantaneous updates of information
- Need highly available network connection
- Clients continuously poll service providers
- In many mobile applications energy is a scarce resource
- Pure pull-based solution may not support the high dynamicity of information resources, that change and move
- Pull works if directory service is up to date
- Publish/subscribe or EDA can be used so that timely notification of events are sent to interested subscribers.

Overview

- Service Provision Lifecycle
- Service Discovery
- **Service Invocation √**
  - Coordination Models
  - On demand Service invocation
  - **Volatile Service Invocation √**
  - ESB versus MOM
- Service Composition

Volatile Service Invocation: Networks

Sometimes service access may be quite intermittent because of intermitted network, Causes?

Volatile Service Invocation: services

- Intermittent service access causes?

  .

- Designs of the application and middleware must take volatile into account Why?
  - .
- Basic designs to handle volatile service access?

Volatile Service Invocation: Designs

Designs to support volatile service invocation?

Volatile Service invocation: Over Unreliable Networks

- Networks may offer a QoS to deliver messages without loss or delay and in order
- Service access over wireless networks often more unreliable than wired networks.
- Applications can assume no network guarantee about delivery - need to detect & handle message corruption & message loss. How?
  - .

Volatile Service invocation Design: Stateful Senders & Receivers

- Senders and receivers can be stateful
- Stateful senders don't need to create message replacements from scratch
- Stateful communication may be more complex to synchronise than stateless communication because the equivalence of intermediate states may need to be compared.

Volatile Service Invocation: Repeating Service Requests

- Before repeating message transmission is to consider the consequences of doing this.
- Messages that can be repeated, at least once, without side-effects are called *idempotent messages*,
  - e.g., ??.
- Other messages may be non-idempotent,
  - e.g., ??.

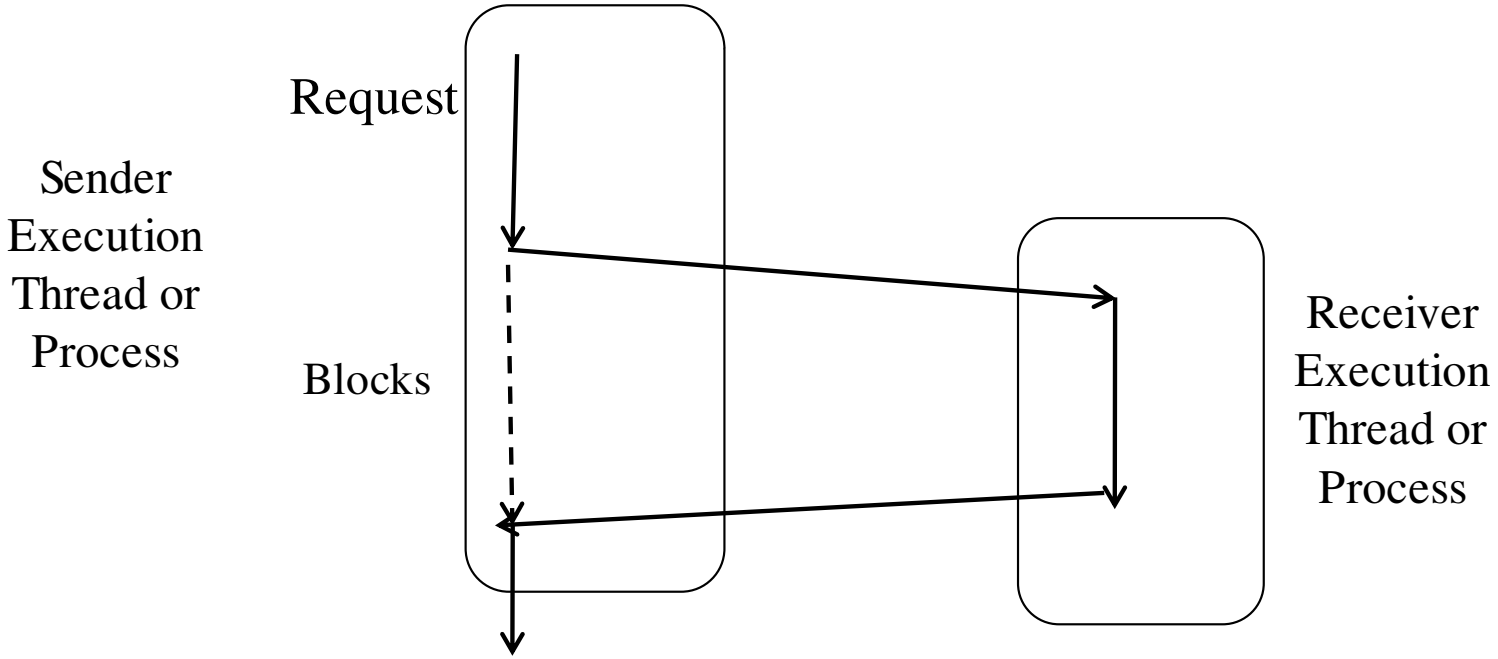
Volatile Service Invocation: Repeating Service Requests

- Partial observability at sender / requester ↑ complexity. Why?
  - –

Volatile Service Invocation: Asynchronous (MOM) I/O

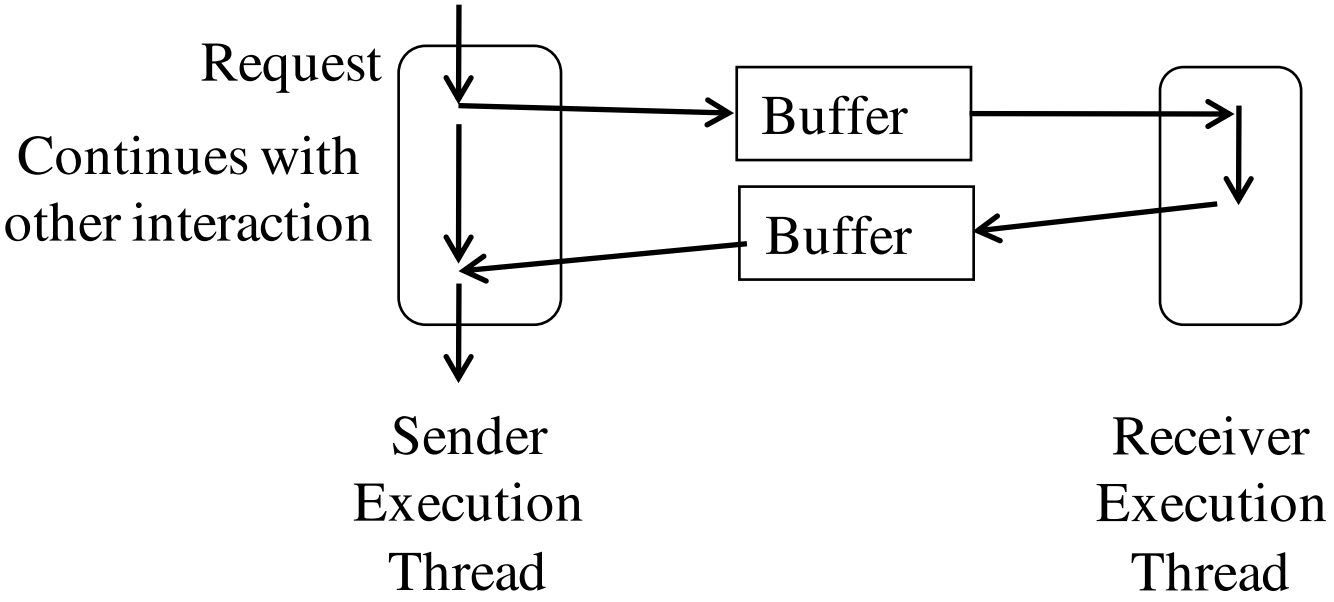
- Problems when senders issues requests to receivers
  - ??
- Asynchronous messaging can solve this issue. Asynchronous messaging applications such as email over the Internet, SMS over mobile voice networks are often regarded as the first important data applications over these networks respectively.
- Two basic variants of asynchronous messaging exist:
  - Sender side asynchronous requests
  - Receiver side asynchronous requests

Volatile Service Invocation: Synchronous I/O



- Advantages?
- Disadvantages?

Asynchronous I/O: design based upon buffering

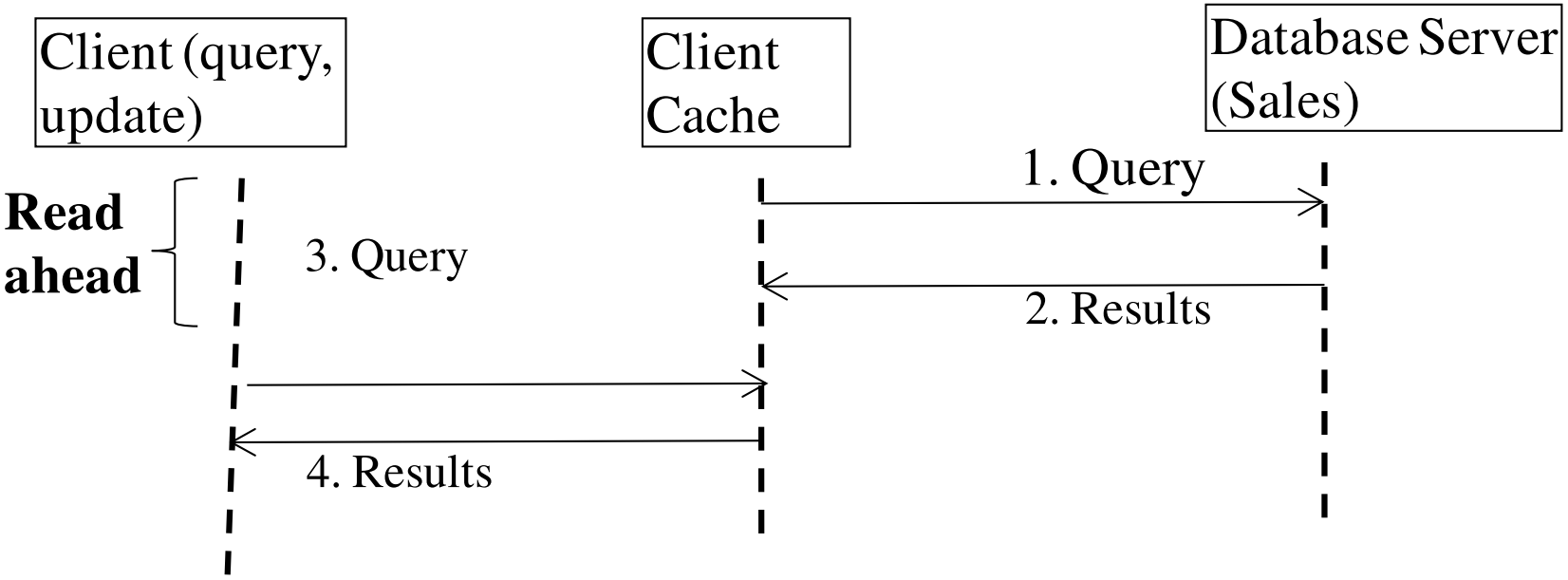


Volatile Service invocation: read ahead caches

- Read ahead is one design option to deal with volatile service invocation
- Information is pre-cached in devices when the network is available.
- Cache-hit
- Cache-miss
- Design decisions?

- Support frequent caching?
  - .
- Support less frequent caching?

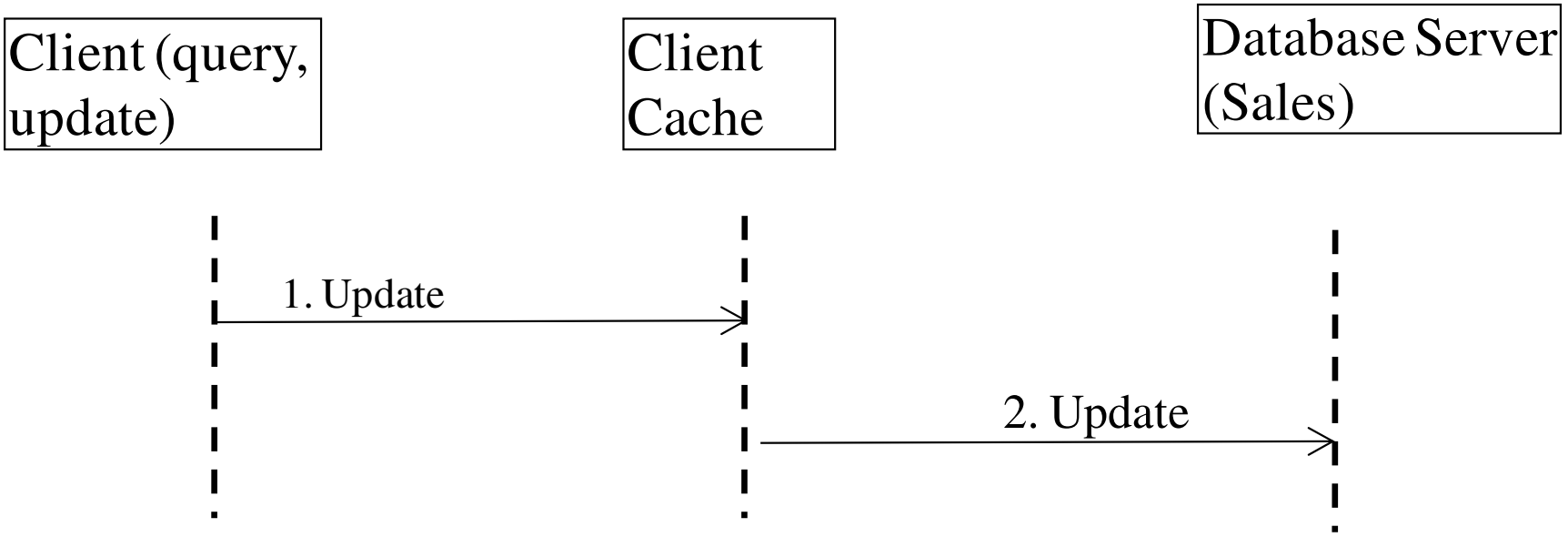
Volatile Service Invocation: Read Ahead



Volatile Service Invocation: Delayed Writes

- With delayed writes, updates are made to the local cache whilst services are unreachable which must be later reintegrated upon reconnection.
- Concurrent local and remote updates may need to be synchronised.
- Write conflicts need to be detected when the same data has been modified locally and remotely.
- Techniques to handle cache misses & cache resynchronisation?

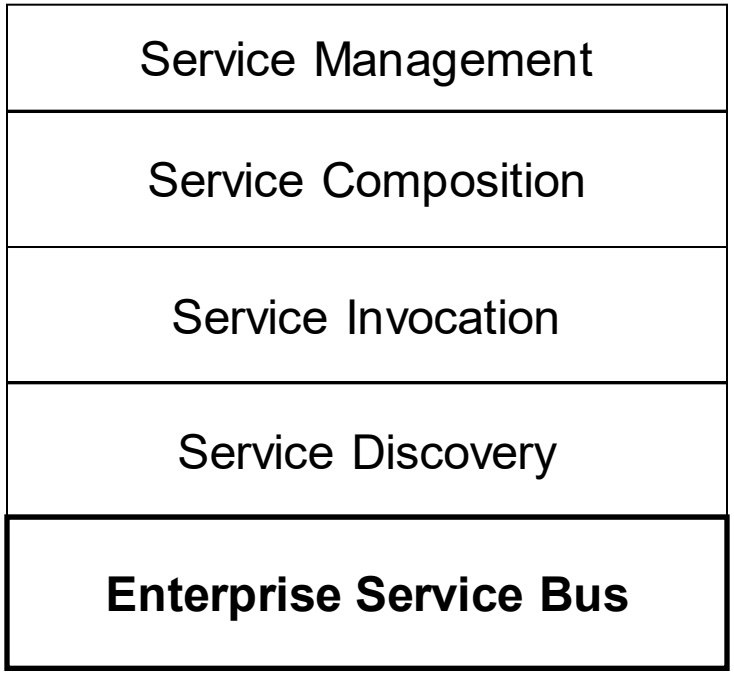
Volatile Service Invocation: Delayed Writes



Overview

- Service Provision Lifecycle
- Service Discovery
- **Service Invocation √**
  - Data Models
  - On demand Service invocation
  - Volatile Service Invocation
  - **ESB versus MOM √**
- Service Composition

Service Oriented Computing (SOC)



Service Invocation: ESB Model

- Enterprise Service Bus or ESB supports messaging, Web Service integration, data transformation and intelligent routing for SOC
  - Decouples service provision from service access.
- 2 main types of design
  - MOM ESB
  - SOC ESB

Service Invocation: MOM ESB Model

- MOM (Message-Oriented Middleware) are examples of asynchronous messaging systems:
  - Other examples., ??
- Often use mediators have facilities to store, route and transform messages.
- Lack of agreed standards for MOM
  -

Service Invocation: MOM ESB Model

- Where to support asynchronous messaging?
  - .
- MOM solutions tend to be more robust to failures than RPC
- Enables service requesters to continue to process other requests and not block.
- But MOM-based applications are complex to design. Why?
  -

Service Invocation: MOM ESB Model

- MOM designS for ESB support
  - ???
- MOMs may mandate a specific application level / transport protocol
  - .
- MOM ESB may itself not be modelled as a direct Web service or first-class service
  - .

Service Invocation: SOA ESB Model

- SOC model for ESB as opposed to a message-oriented model offers fuller support for three types of integration:

1. integrating multiple service access, e.g., behaving as a portal;
2. integrating multiple application service processes,
3. supporting work-flows, brokerage and propagation; supporting data translation.

Overview

- Smart Device and Service Characteristics
- Distributed System Viewpoints
- System Abstraction
- Partitioning and Distribution of System Components
- Proxies & Middleware
- Service Oriented Computing (SOC)
- Peer-to-Peer Systems
- Service Provision Lifecycle
- Service Discovery
- Service Invocation
- **Service Composition √**
- MTOS, BIOS & VM

Service Composition: Service Interoperability

- The ability to communicate at the network level is insufficient to interoperate at service level. Why?

- Difference between integrated services versus interoperable (or federate) services?

Service Composition: Service Interoperability

- Distributed systems that interoperate can exchange data in a variety of data formats, encodings, etc.

2 methods for heterogeneous data exchange:

- common or canonical exchange data format is used
- receiver or sender makes it right scheme
  - .
- Which is best?

Service Composition

- Service processes are sequences of individual service actions that are scheduled for execution.
- Service processes may involve one or more entities, one or more actions and involve one or more processes.
- Composition concerns?
  - .
- Statically organising services design issues.

- Dynamic service composition?

Service Composition

- Composition can occur incrementally
- Why / Benefits?
  - .
- Composition Management
- See later, chapter 9

Service Composition: *Orchestration*

- Term *orchestration* in music
  - .
- Orchestrator tends to hold a global viewpoint of:
  - Service actions
  - Service constraints for participants
- Service composition can be specified
  - manually or
  - automatically.

Service Composition: *Choreography*

- Derived from the Greek words for “dance” and “write”

- Participants are usually more responsive to:
  - local viewpoints of the service actions of self
  - the adjacent service action of others
  - less use of global view.
- More about Choreography designs in chapter 9

Service Orchestration versus Service Choreography?

- Service orchestration is simpler to design & manage
- Service orchestration is more commonly used in SOAs
- etc

Service Composition: dynamic

- Several approaches exist to automate service composition:
  - business processes,
  - Work-flow,
  - Semantic Web
  - MAS planning.
- See Chapter 9

Overview

- Smart Device and Service Characteristics
- Distributed System Viewpoints
- System Abstraction
- Partitioning and Distribution of System Components
- Proxies & Middleware
- Service Oriented Computing (SOC)
- Peer-to-Peer Systems
- Service Provision Lifecycle
- Service Discovery
- Service Invocation
- Service Composition
- **MTOS, BIOS & VM √**

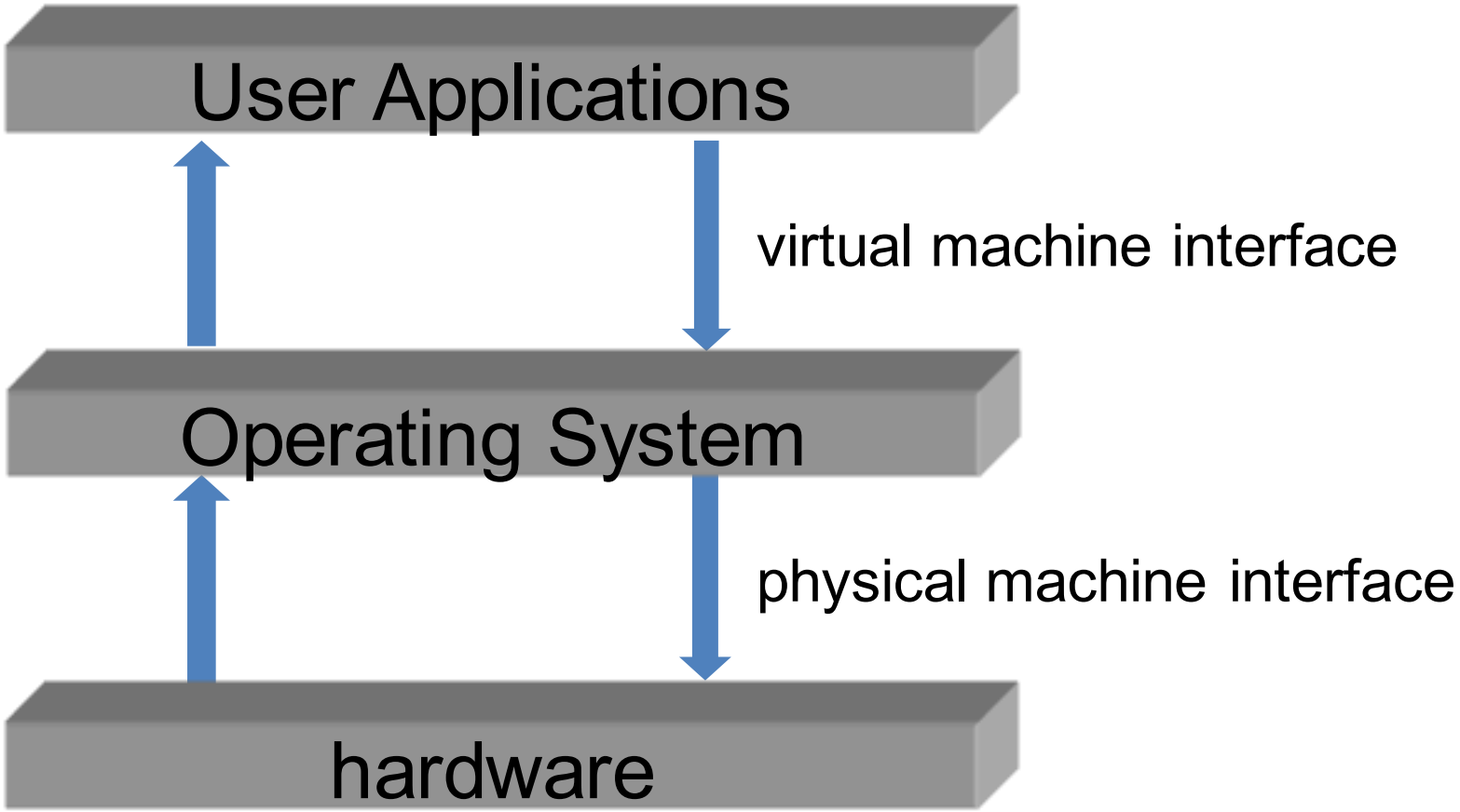
Operating System (OS)

- OS: system software that:
  - *Controls/abstracts hardware*
  - *Manages resources and processes* to support different applications
- OS enables user applications to be ↑ simpler & device-independent
  - Applications use API to access hardware and OS
- 3 main resources of system are Managed. What?

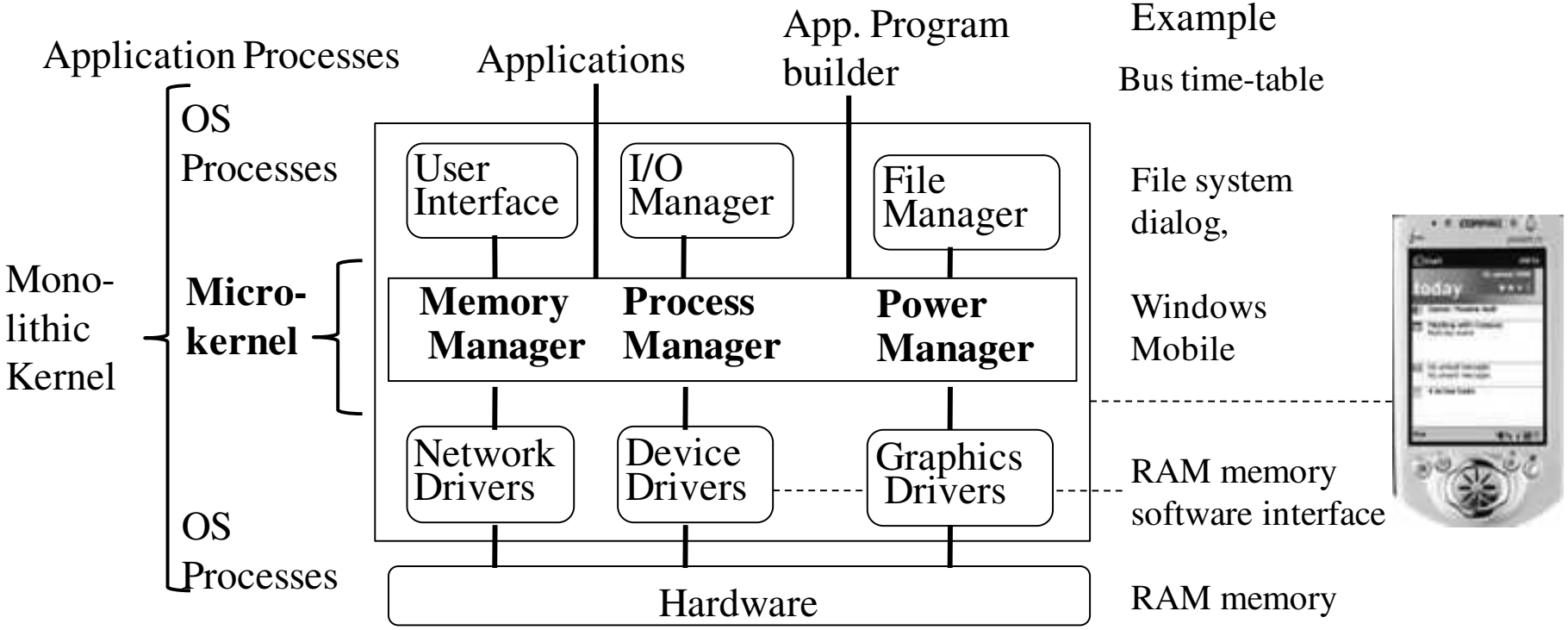
- In mobile, resource constrained devices additional resources are managed. What?
  - Power (See Section 4.3)
  - UI & Content (See Section 7.6.1.2)

Operating System

- Typically a 3-tier architecture



Operating System



OS: Macro kernel

- Macro-Kernel (Monolithic Kernel)
- Everything in One Single Large Kernel
  - i.e., Hardware related, i.e., device drivers, MM, process support, Scheduling, Network protocol stack, File system
  - e.g., versions of Linux
- Benefits?:

- Drawbacks?:

OS: Micro-Kernel

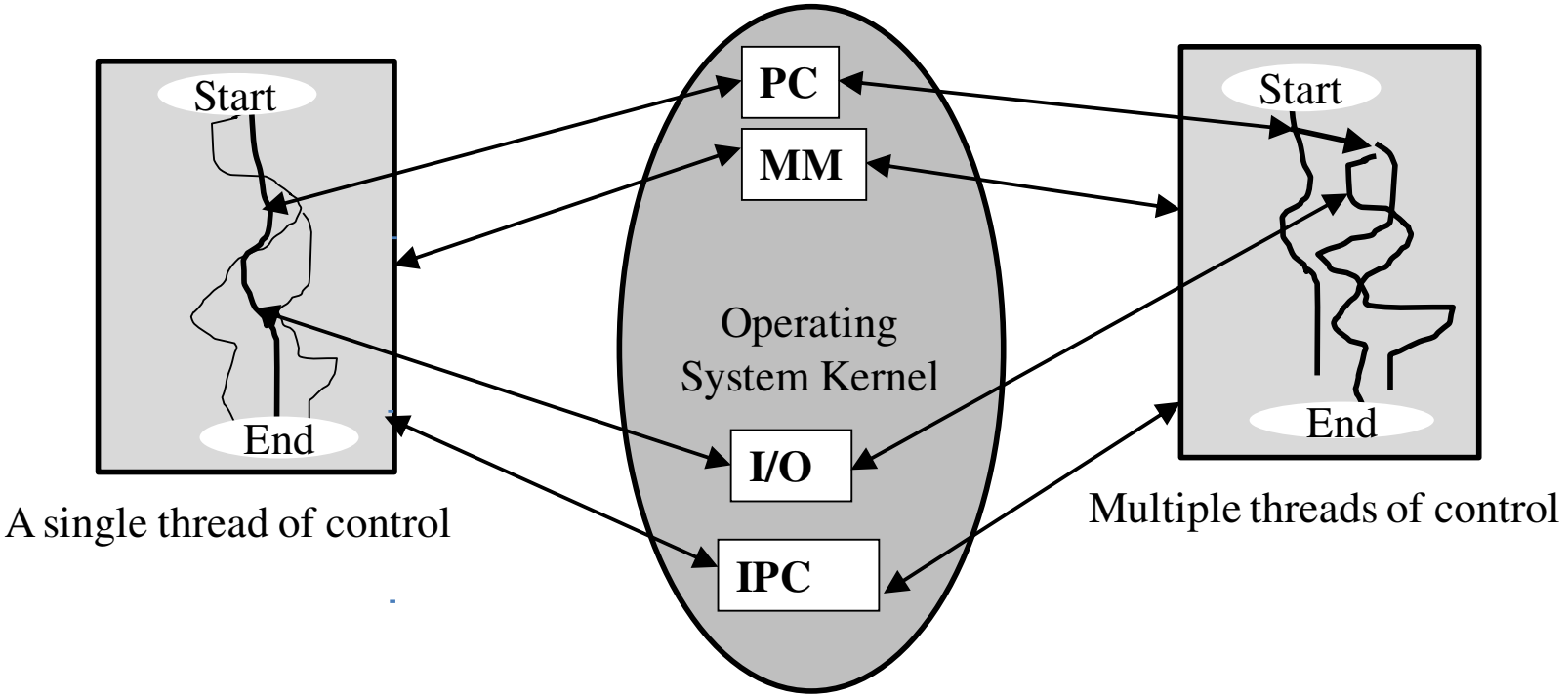
- Only fundamental parts in kernel. Which?
- Benefits?
- Drawbacks?

OS: Process Management

- OS Kernel is a special process that ….

- Has full access rights to all physical resources
- Has its own protected address space for its data memory
- It runs the CPU in a special mode called the supervisor mode.
- Creates an execution environment for processes to safely run in memory (outside the kernel space).

MTOS: Process Management



MTOS: Process vs. Thread Management

- ???

Operating System: Process Management

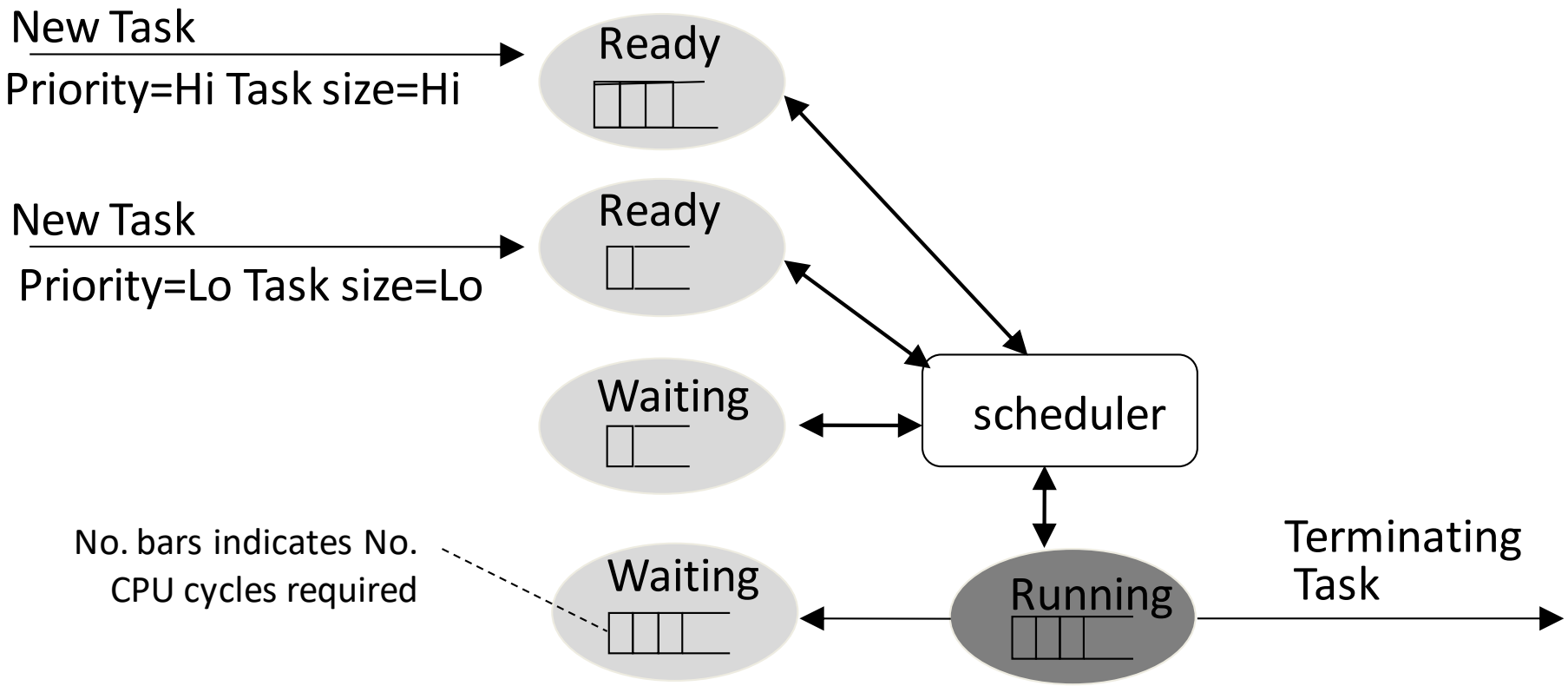
- OS kernel coordinates multiple processes use of CPU
- OS manages inter-process communication.
- Sometimes No. of *executable processes* > No. CPUs
- Executing processes can block waiting for resources. Why?
  - Solution?

Operating System: Process Scheduling

- *Pre-emptive scheduler (*Simplest):

- *Non pre-emptive* scheduler:
  - –
- *Priority scheduling*:

OS: Process Management or Control



Memory Management

- Processes associated with Virtual Memory / address space
  –
- OS kernel defines a separate region of address space for each process
- Each process has separate (primary) memory area
- What if more processes than memory?

Distributed System OS

- Much ICT use is inherently distributed
- Devices are distributed
- Distributed Service invocation (review Section 3.3.3)

BIOS

- Often when a computer is started or booted, also called bootstrapped, software is loaded in stages.

1. BIOS or Basic Input/Output System, a type of firmware, is loaded.
2. BIOS initialises several motherboard components and peripherals, e.g., ?
3. BIOS loads & transfers control to OS

Service Virtualisation

- In practice, complete service abstraction is hard to achieve. Why?
- Physical resources such as hardware resources can be virtualised
- Combination of the virtualisation and abstraction

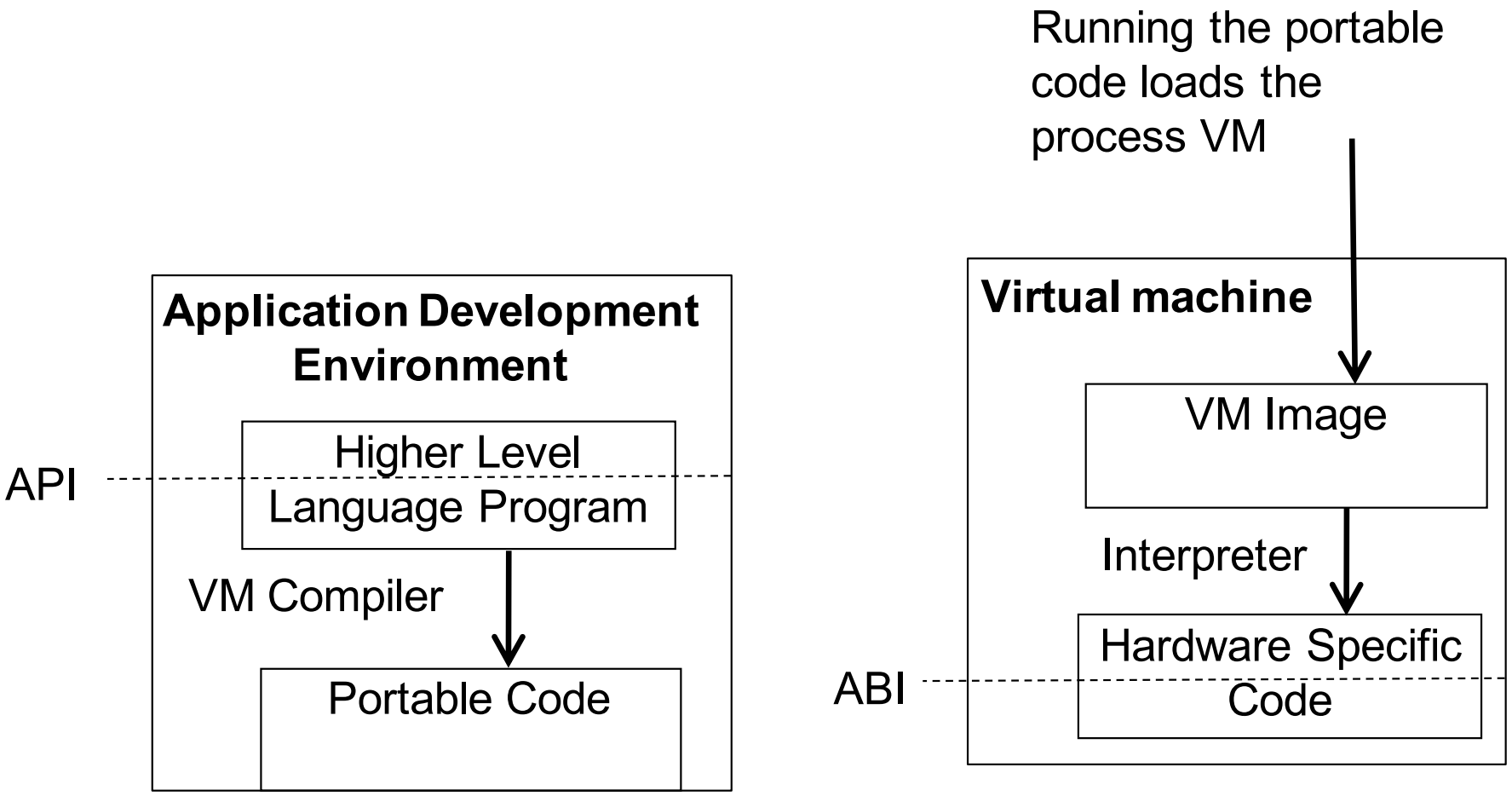
Virtual Machines (VM)

- Important uses of virtualisation. What?
  –
- Virtual Machines support. What?
  –

VM: Process vs. System

- 2 different types of VM
- (application) process viewpoint : Process VM
- (operating) system viewpoints: System VM

Process Virtual Machines



Virtual Machine: System VM

- System VM - the original type of VM developed in the 1960s and 1970s for mainframes
- System VM enable multiple OS systems and application to be run on the same hardware
- If one system VM fails, others are isolated and keep running.
- Still a useful technique in modern servers and server farms that need to support multiple users and their applications and share hardware resources amongst them.

Overview

- Smart Device and Service Characteristics √
- Distributed System Viewpoints √
- System Abstraction √
- Partitioning and Distribution of System Components
- Middleware √
- Service Oriented Computing (SOC)
- Peer-to-Peer Systems √
- Service Provision Lifecycle √
- Service Discovery √
- Service Invocation √
- Service Composition √
- MTOS, BIOS & VM √

Summary & Revision

For each chapter

- See book web-site for chapter summaries, references, resources etc.
- Identify new terms & concepts
- Apply new terms and concepts: define, use in old and new situations & problems
- Debate problems, challenges and solutions
- See Chapter exercises on web-site

Exercises: Define New Concepts

- Service discovery etc

Exercise: Applying New Concepts

- What is the difference between service push and service pull?

UbiCom Book Slides

Chapter 4
Smart Mobile Devices, Networks & Cards

Stefan Poslad
http://www.eecs.qmul.ac.uk/people/stefan/ubicom

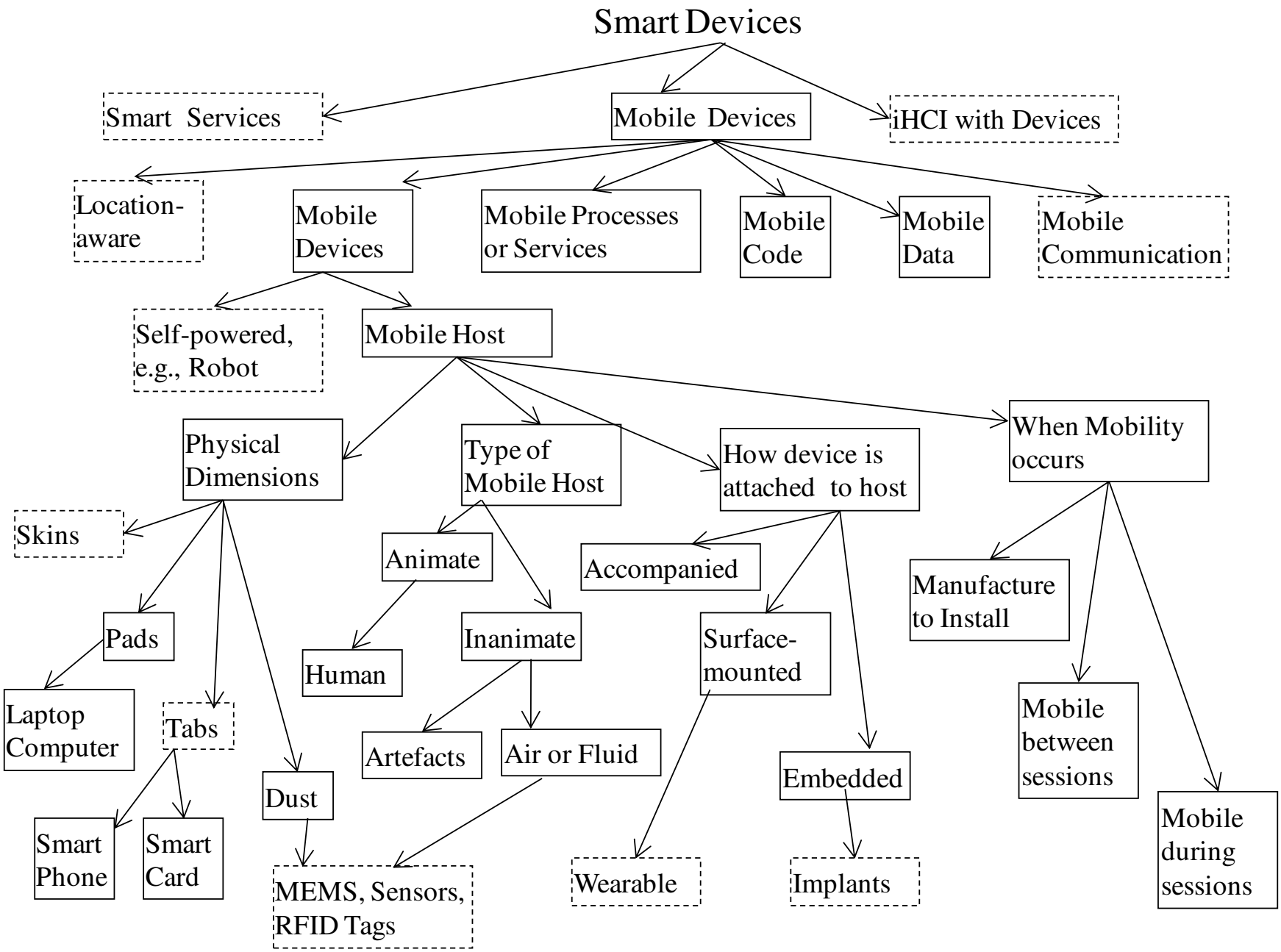
Introduction

Chapter 4 focuses on:

- Internal system properties: distributed system & on sub-property of mobility
- External interaction with other ICT systems in its virtual computing environment.

Links to Related Sections

- Mobile human devices, e.g., wearables and implants (Chapter 5)
- Robots as a mobile host (Chapter 6)
- Mobile devices in form of dust (Chapter 6)
- Location-awareness often quoted as a key killer app for mobile devices (Chapter 7)
- Management of mobile devices (Chapter 12)
- Future for mobile devices (Chapter 13)


Smart Devices
Smart Services
Mobile Devices
iHCI with Devices
Location-aware
Mobile Devices
Mobile Processes or Services
Mobile Code
Mobile Data
Mobile Communication
Self-powered, e.g., Robot
Mobile Host
Physical Dimensions
Type of Mobile Host
How device is attached to host
When Mobility occurs
Skins
Animate
Accompanied
Manufacture to Install
Pads
Human
Inanimate
Surface-mounted
Laptop Computer
Tabs
Artefacts
Air or Fluid
Embedded
Mobile between sessions
Dust
Smart Phone
Smart Card
Wearable
Implants
Mobile during sessions
MEMS, Sensors, RFID Tags

Introduction

The slides for this chapter are also expanded and split into several parts in the full pack

- **Part A: Mobility Dimensions & Design**
- **Part B: Mobile Services**
- **Part C: Mobile device OS**
- **Part D: Cards Devices and Device Networks**

Overview

- **Smart Mobile Device Characteristics √**
- **Mobile Code & Service Design Principles**
- **Mobile Service Design**
- **SMS, WAP & I-Mode Mobile Services**
- **Mobile Device OS**
- **Smart Card Devices**
- **Device Networks**

Smart Mobile Devices

- Mobile Smart Device
  - Enable devices to access services ubiquitously
- Smart mobile devices are driven by ↑ capability to embed
- Variety of form factors
- ↑ Wireless? LAN, WAN access to Internet, voice, video etc.
- Mobile devices themselves are often not mobile!

Smart Mobile Device Characteristics

What are these?

Dimensions of Mobility

- Mobility is a very rich concept.
- Some important dimensions of mobility
  - Mobile devices
  - Mobile services (also see Chapter 3)
  - Mobile data
  - Mobile code
  - Mobile communication (also see Chapter 11)
  - Mobile context (also see Chapter 7)

Dimensions of Mobility for Devices

- Sometimes mobile devices are not mobile
  - Mobile host & non-mobile device vs. mobile device
- Type of mobile host?
- Physical dimensions of mobile device?
- How a non-mobile device is attached to a mobile host?
- When the mobility occurs, during the operational life-cycle.

Mobility Dimensions: When Mobility Occurs

- Different degrees of mobility
  –
- Relative (from home) versus absolute (untethered)
- Mobile host versus mobile device
- How is device attached to mobile host?
  – Accompanied, Surface mounted, embedded

Overview

- Smart Mobile Device Characteristics
- **Mobile Code & Service Design Principles √**
- Mobile Service Design
- SMS, WAP & I-Mode Mobile Services
- Mobile Device OS
- Smart Card Devices
- Device Networks

Mobile Code

- Enables providers to maintain, e.g., upgrade and fix, code in consumer devices with a network connection
  –
- Installation requires configuring code on each platform.
- Mobile code languages: Java, C#, Postscript, etc
- Some Mobile Cole Models allow code to move during operation / between sessions

Mobile Code

- Enables providers to maintain, e.g., upgrade and fix, code in consumer devices with a network connection
  –
- Installation requires configuring code on each platform.
  –
- Mobile code languages
  – E.g.,
- Some Mobile Cole Models allow code to move during operation / between sessions

Mobile Code: Pros and Cons

- Benefits?

- Cons?

Mobile Code: Security

Several main approaches to mobile code security:

- Sandboxes
- Code signing
- Firewalls
- Proof-carrying code (PCC)

Mobile Code: Designs

Mobile code design varies according to where code executes and who determines when mobility occurs:

- Client-server / remote evaluation interaction (Chapter 3)
- Code on demand (versus Software as a Service Model (Chapter 3, 12)
- Process migration
- Mobile agents
- Active networks (Chapter 11)

Overview

- Smart Mobile Device Characteristics
- Mobile Code & Service Design Principles
- **Mobile Service Design √**
- SMS, WAP & I-Mode Mobile Services
- Mobile Device OS
- Smart Card Devices
- Device Networks

Mobile Service Design (Overview)

- Transparent Service access
- Data Access
- Data Management on Mobile Device
- Networking
- Volatile Network Links

Mobile Service Design: Transparent Service Access

- To simplify service access whilst mobile, various transparencies are useful. Why? What?
  - –
- Where should Transparency be handled?
- Should Client applications designed to be aware of mobile changes?
  - What are the Pros and cons?

Mobile Service Design: Transparent Service Access

3 kinds of transparency for middleware

- User Virtual Environments (UVE)
- Mobile Virtual Terminals (MVT)
- Virtual Resource Management (VRM)

Mobile Service Design: Data Access

- How do we manage all the content we access on the mobile phone?
- How do service deal with heterogeneous terminal capabilities, Web Browsers?
- Dumb approach to content adaptation
- Access specialised Mobile portal content vs. adapt content on the fly from any Web portal?
  - (See Section 7.6)

Mobile Service Design: Data Management on the Mobile Device

- Mobile devices may create new local data that may be business sensitive or personal.
- Denial of Service (DoS) can occur when mobile device gets stolen or left behind.
- Solutions to handle temporary DoS?

- Data synchronisation is needed
  –
- Solutions to handle temporary DoS, e.g., permanently lost

Mobile Service Design: Wireless Networking

- Do we need new protocols for wireless?
- Or do we need to specialize/optimize existing protocols?
- Protocols optimised for wireless
  - e.g., cellular devices
  - Data exchange protocols for wireless end-loops
  - Data presentation for mobile terminals (Chapter 7)
  - Design? thin client-server model, terminal only does presentation (Chapter 3)
  - Management and security? (Chapter 12)

Mobile Service Design: Volatile Network Links

- Wireless network links may be volatile for a variety of reasons (Chapter 11)
- Hence Mobile Services must be designed to be volatile.
  - There are a variety of designs (Chapter 3)

Overview

- Smart Mobile Device Characteristics
- Mobile Code & Service Design Principles
- Mobile Service Design
- **SMS, WAP & I-Mode Mobile Services √**
- Mobile Device OS
- Smart Card Devices
- Device Networks

Mobile Service Design: SMS

What is SMS?

- SMS (short messaging service) for GSM / 2G.
- Network service characteristics?

- Use of gateways
  –

Mobile Service Design: SMS

- Advantages?

- Limitations?

- Because of limitations, WAP developed but SMS still used more than

Mobile Service Design: WAP

WAP (Wireless Application Protocol) - Advantages over SMS?

- Two WAP versions v1 & v2
- Earlier WAP was a separate standards body but since 2003, part of the OMA (Open Mobility Alliance) which covers everything

Mobile Service Design: WAP V1 vs V2

- ??

Mobile Services: 3-Tier Client-Proxy-Server Model

- See Chapter 3

Mobile Service Design: I-mode

- Mobile information service launched by NTT DoCoMo of Japan in Feb 1999
- Based on proprietary technology
  - Japanese PDC-P (Personal Digital Cellular-Packet)
- Compelling (and profitable) cHTML content
- Single browser (Access) on multiple handsets
- Handsets designed for the service, rather than technology
- Cheap to use (packet-based costs not time-based)
- End to end

Mobile Service Design: c-HTML/i-mode

Android & i-phone

Overview

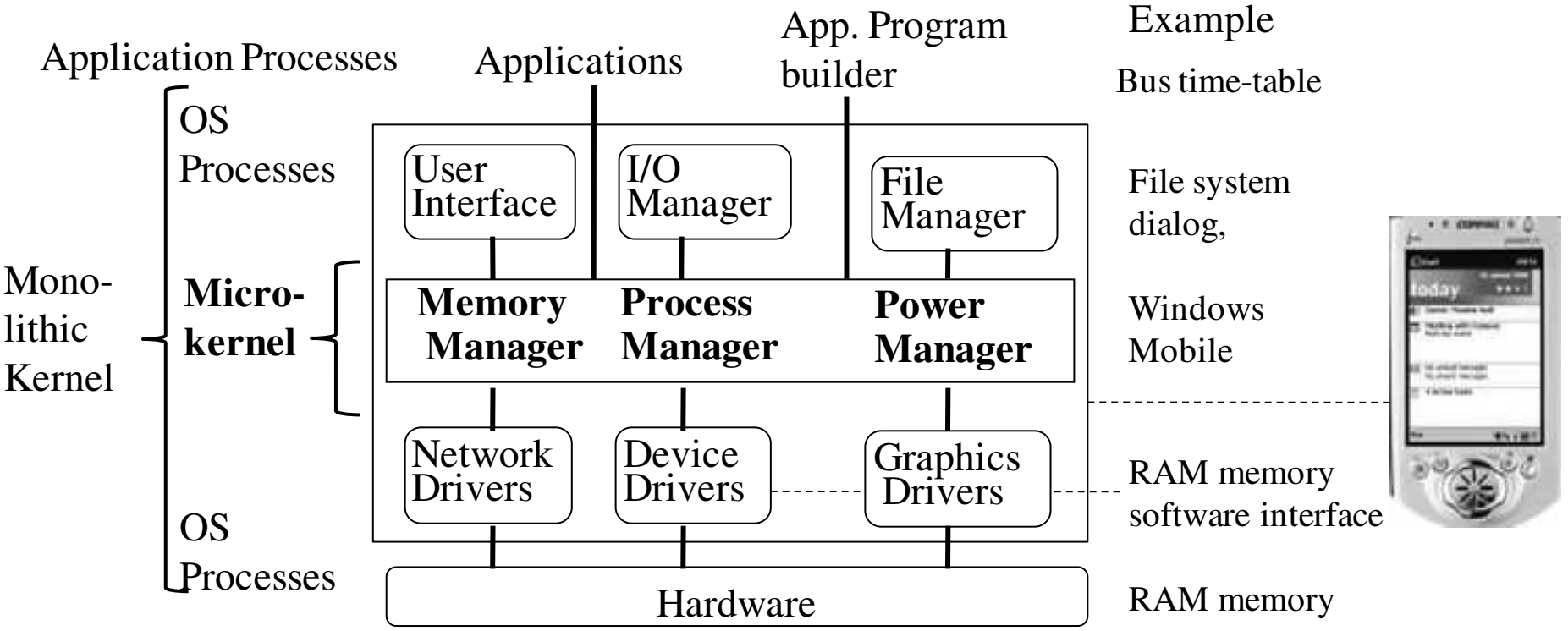
- Smart Mobile Device Characteristics
- Mobile Code & Service Design Principles
- Mobile Service Design
- SMS, WAP & I-Mode Mobile Services
- **Mobile Device OS √**
- Smart Card Devices
- Device Networks

Operating System (OS)

- OS: system software that:
  - *Controls/abstracts hardware*
  - *Manages resources and processes* to support different applications
- OS enables user applications to be ↑ simpler & device-independent
  - Applications use API to access hardware and OS
- 3 main resources of system are Managed. What?

- In mobile, resource constrained devices additional resources are managed. What?
  - Power (See Section 4.3)
  - UI & Content (See Section 7.6.1.2

Operating System



OS: Macro kernel

- Macro-Kernel (Monolithic Kernel)
- Everything in One Single Large Kernel

- Benefits? (for mobile device use):

- Drawbacks? for mobile device use):

OS: Micro-Kernel

- Only fundamental parts in kernel.
  -
- Benefits (for mobile device use):
- Drawbacks (for mobile device use):

Symbian OS

- Specifically designed as an OS for mobile devices.
- Has a multi-tasking kernel
- Has a POSIX compliant interface and a JVM
  - –
- Etc

Mobile OS: Process Control

How to support multi-tasking in a mobile device OS?

Mobile OS Design: Static vs Dynamic Process Scheduling

- Static: all scheduling decisions determined before execution
  - ???
- Dynamic: run-time decisions are used
  - .

Mobile OS design: Scheduling CPU Usage

- *Pre-emptive scheduling*:
  - .
- *Non-pre-emptive scheduling*: vs. run to completion .
  - .

Mobile OS: Memory Management

Memory Management

- Kernel should be small.
- Good resource / Memory management needed
- System resources should be released as soon as they are no longer needed

Mobile OS Design: Memory

- In the past, phone devices retain information in memory as long as the battery held a charge.
  –
- Now, permanent storage in the form of Flash ROM
  –
- Mobile devices boot from ROM & load data more slowly.
- On the other hand, ROM memory uses less power
  –
- N.B. earlier types of ROM such as Compact Flash had a limited lifetime in terms of read/writes.

Mobile OS: Power Management

Requirements

- If Mobile devices' hardware resources are fully powered up all the time, often only a fraction of power is being used:
- Device needs to be responsive in all situations. How?
- Devices should not be powered down completely. Why?
- Competing processes/ users scheduled to receive, a fair share of battery (power) resources rather than CPU resources,

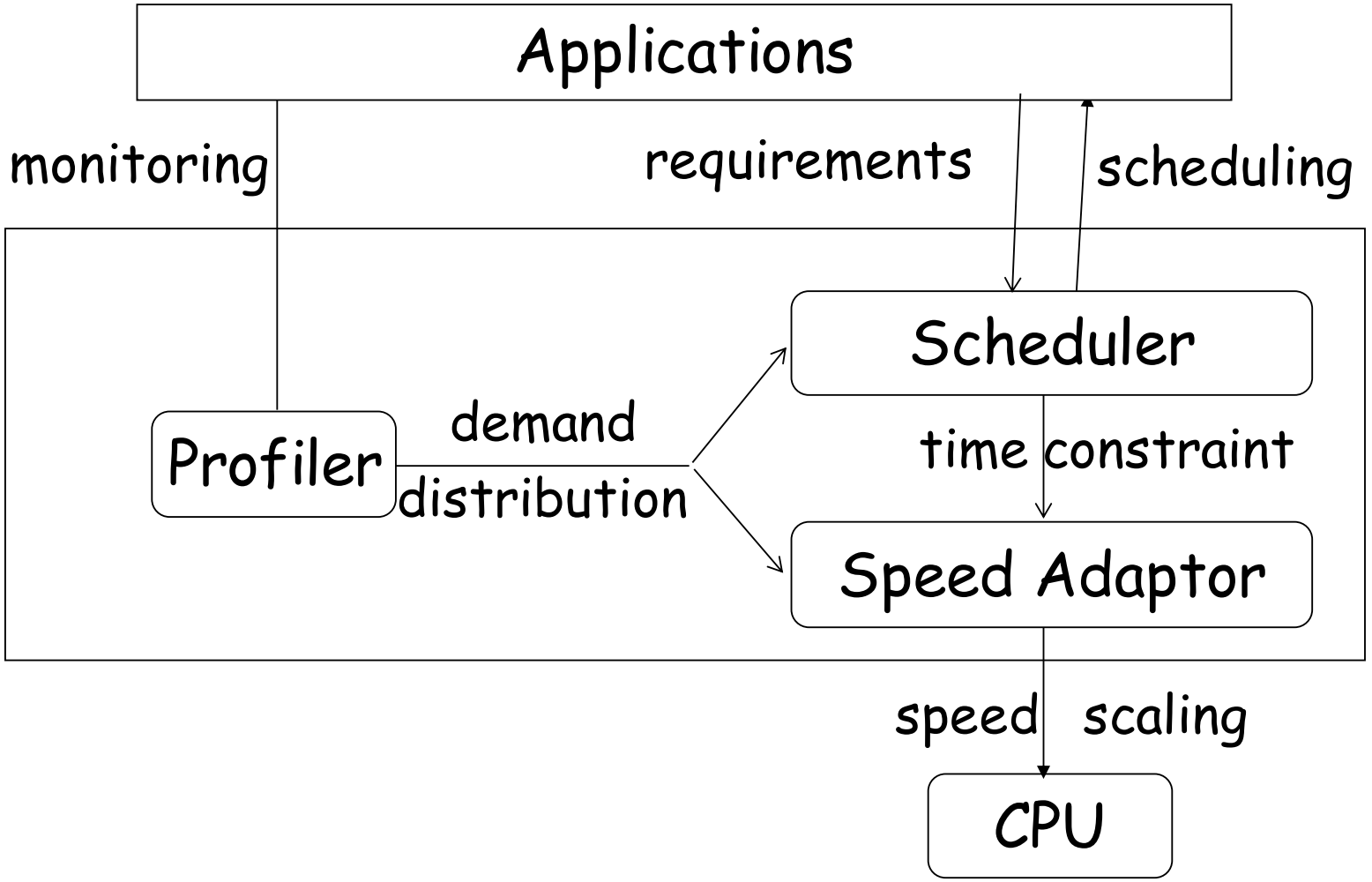
Mobile OS: Power Management

Design

- Dynamic Voltage Scaling (DVS)

- DVS+SRT

Mobile OS: Power Management



Mobile Device Power Management

Miscellaneous Issues

- Focus here (Chapter 4) has been on power management of Tab and pad type devices.
- Power management for dust type mobile devices (Chapter 6)
- Low power, eco-friendly issues are (Section 13.5.2)

Mobile OS: Combined Process & Power Scheduling

Overview

- Smart Mobile Device Characteristics
- Mobile Code & Service Design Principles
- Mobile Service Design
- SMS, WAP & I-Mode Mobile Services
- Mobile Device OS
- **Smart Card Devices √**
- Device Networks

Smart Cards

- Type of chip card constructed out of substrate, e.g., plastic
  - .
- Just about anything found in a person's wallet has the potential to be stored on a smart card, what?
  -

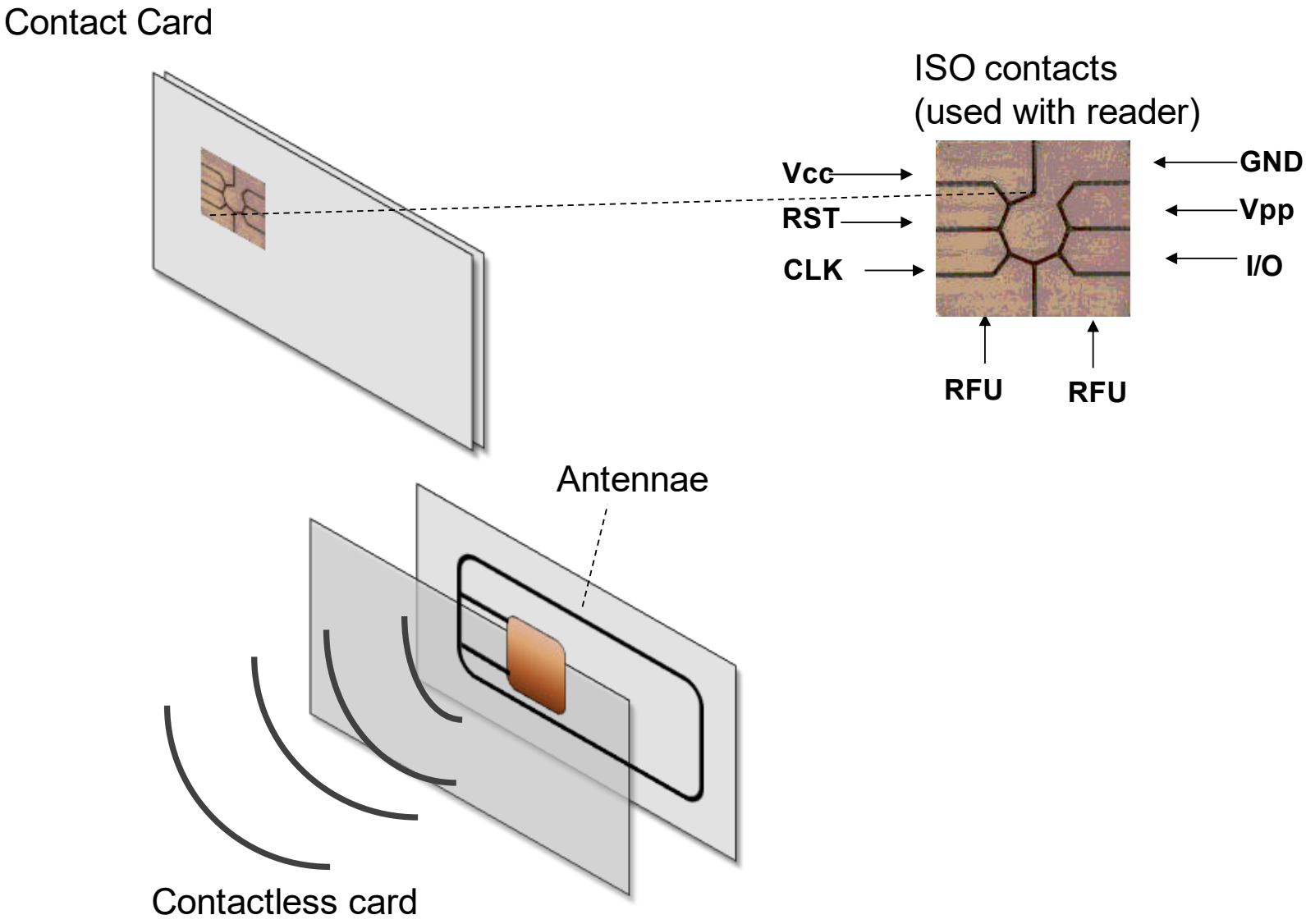
Smart Cards vs. Smart Phones

???

Smart Cards

- 1977: Motorola, Bull produced 1st smart card microchip
- 2009: Add latest figures about Smart Card use today
- 2010+: Multiple plastic cards could “meld into 1 universal, multifunctional. smart card
- Chip may be memory only or CPU & memory
- Data associated with either value or information or both is stored and processed within the card
- No inbuilt user interface for I/O
- ard data is transacted via a card reader or wireless base-station

Smart Card: Hardware Interface



Smartcard Types: Contact vs. Contactless

'Contact' cards

- Card inserted in reader
- Physical contact made

Contactless' card

- Uses RF transceiver / transponder chip
- Card is 'waved' in immediate vicinity of a reader or base-station

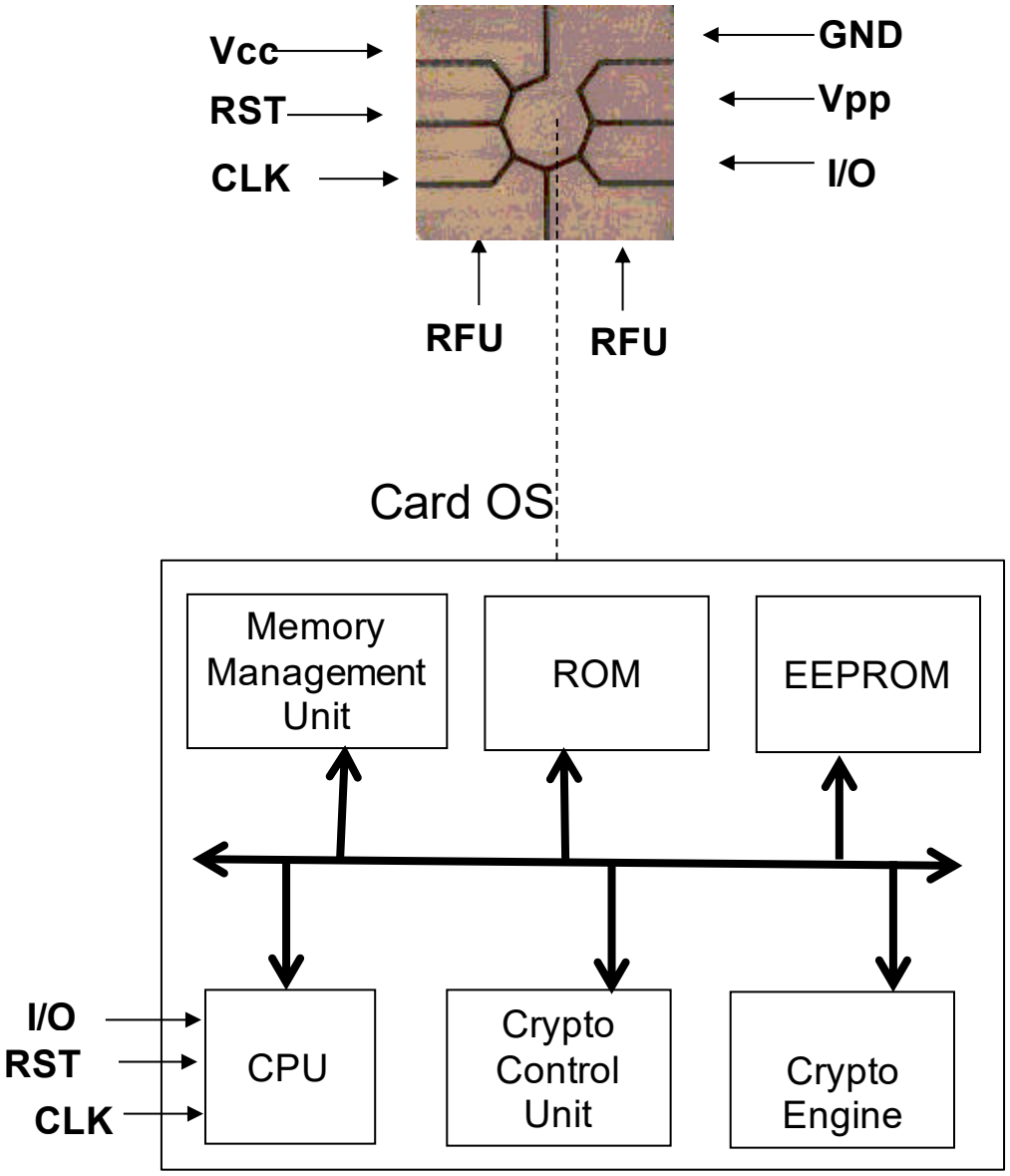
Smartcard Types: Contact vs. Contactless

- ???

Smart cards: SIM cards

- Subscriber Identity Module (SIM
- Used in GSM mobile phones to authorise subscriber access to cellular network
- SIM cards securely identify a subscriber.
- SIM card allows users to change phones by swapping SIM cards
- 1st larger, credit-card sized) SIM Card was made in 1991
- Later, miniature-versions appeared

Smart Card: Operating System



Smart Card Application Design: System & Environment Requirements

- Upgradable
- Security?
- Contactless vs. Contact Card design
- Wireless RF: operating range & data rate

Smart Card: Application Requirements

- Primary tasks of smart card operating system on behalf of an application:
- Transferring data to and from the smart card
- Controlling the execution of commands
- Managing files
- Managing and executing cryptographic algorithms
- Managing and executing program code.

Smart Card APIs: Java Card

- Java Card refers to a technology that allows small Java-based applications (applets) to be run securely on smart cards and similar small memory footprint devices.
- Java Card is the tiniest of Java targeted for embedded devices. Java Card gives developers the ability to program the device and make them application specific.
- Widely used in SIM cards (used in GSM mobile phones) and ATM cards.
- First Java Card was introduced in 1997

Smart Cards: Java CardVM

- Java Card applications are Java Card bytecode run in a Java CardVM
- Uses a different encoding optimized for size.
- Java Card applet typically uses less bytecode than the hypothetical Java applet obtained by compiling the same Java source code.
- Conserves memory
- Techniques exist for overcoming the size limitation of 64 KB limit
- Java CardVM runs in many smart cards even a GSM phone SIM card

Development of Java Smart Card Application (applet)

There are 4 steps comprise the Java Card applet development:

1. Specify the functions of the applet
   - E.g., security function requires the user to enter a PIN, card locks after three unsuccessful attempts to enter the PIN.
2. Request and assign AIDs to both the applet and the package containing the applet class
3. Design the class structure of the applet programs
4. Define the interface between the applet and the terminal application

- Develop using the JavaCard development kit
  - See http://java.sun.com/products/javacard/dev_kit.html

More about Smart Card Application Development

Advances in Smart Cards

- ??

Overview

- Smart Mobile Device Characteristics
- Mobile Code & Service Design Principles
- Mobile Service Design
- SMS, WAP & I-Mode Mobile Services
- Mobile Device OS
- Smart Card Devices
- **Device Networks √**

Device Networks

- Objective of a device network is to enable a wide variety of devices to interoperate.
- Applications:
  - home automation, e.g., light and climate control,
  - person-aware systems
  - home security, care in the community
  - pervasive AV content access (Section 2.3.2.1).
  - etc

Devices versus Services

- Can a device simply be abstracted & modelled as a service?

Device Networks: Characteristics & Challenges

Characteristics

- ????

Challenges

- ??

Device Networks: Network Technologies

- InfraRed
- BlueTooth
- X10
- HAVi
- HES
- UPnP
- Jini
- OSGi
- WiFi,
- DECT
- 3G mobile phone networks

Device Networks: Device Discovery

- Network Discovery:
  - ????
- Device / service discovery:
  - *Jini*,
  - *UPnP*
  - IETF's SLP,
  - DNS Service Discovery
  - Bluetooth's SDP
- Device / service execution / management
  - OSGi

Open Services Gateway Initiative (OSGi)

- *OSGi* promotes open specifications for the delivery of managed services into networked environments such as homes & automobiles.
- Initial market for OSGi was home services gateways, e.g., in video broadcast set top boxes, broadband modems.
- These then act as a gateway, between the end user (and owner) of the devices on a LAN, and the service providers that could be accessible over the Internet who want to providers (i.e., sellers) of services

Open Services Gateway Initiative (OSGi)

- Core OSGi platform specification:
  - ???.
- OSGi in turn uses underlying Java VM (Section 4.2.1.2) and OS services (Chapters 3 and 4)
- Application services are encapsulated, deployed in bundles
  -
- Event-driven mgt. mechanisms support installation, activation, deactivation, update, and removal of bundles.
- Typical OSGi service framework implementation
  - .

Overview

- **Smart Mobile Device Characteristics √**
- **Mobile Code & Service Design Principles √**
- **Mobile Service Design √**
- **SMS, WAP & I-Mode Mobile Services √**
- **Mobile Device OS √**
- **Smart Card Devices √**
- **Device Networks √**

Summary & Revision

For each chapter

- See book web-site for chapter summaries, references, resources etc.
- Identify new terms & concepts
- Apply new terms and concepts: define, use in old and new situations & problems
- Debate problems, challenges and solutions
- See Chapter exercises on web-site

Exercises: Define New Concepts

- Mobile code, etc

Exercise: Applying New Concepts

- What are the main design challenges in developing mobile code?
- etc