

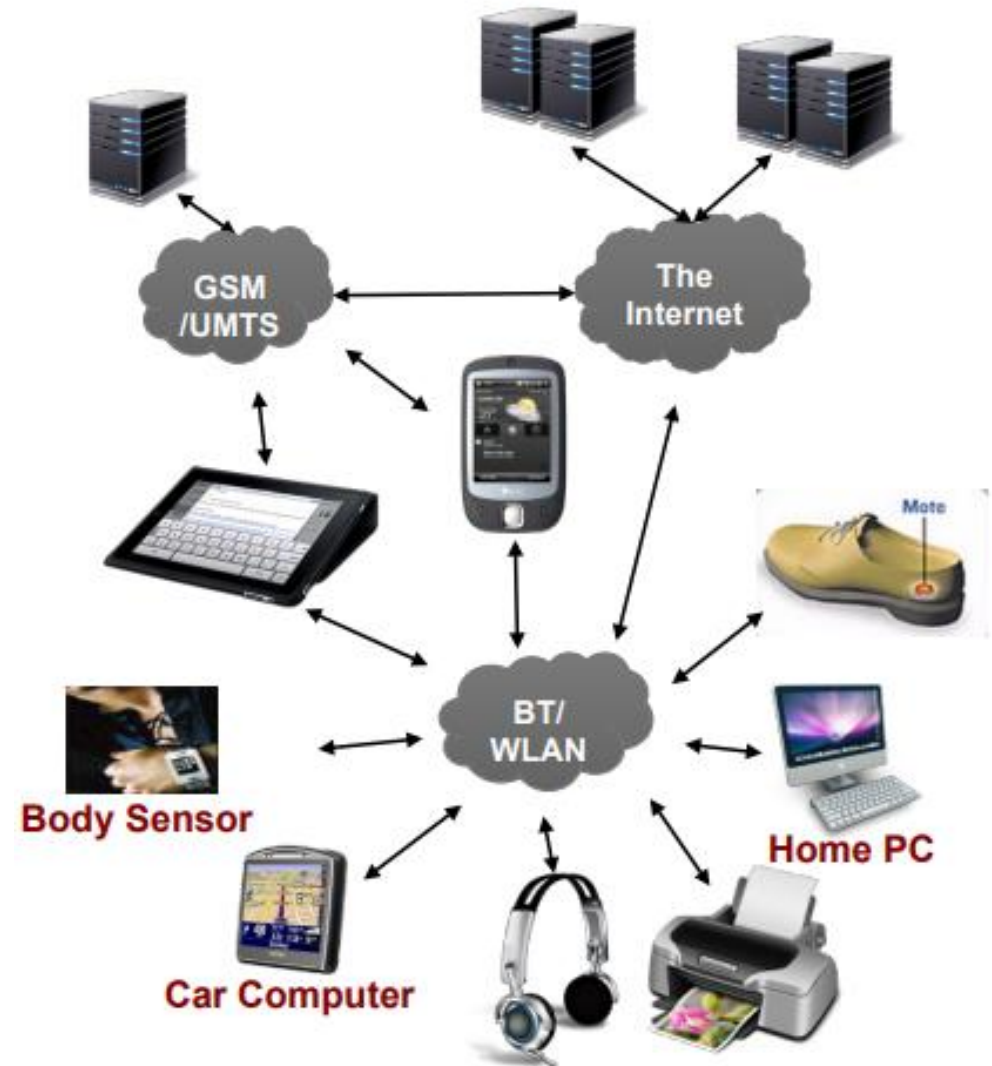
# Mobile and Ubiquitous Computing

CAT3053/N Distributed Computing

Week 11

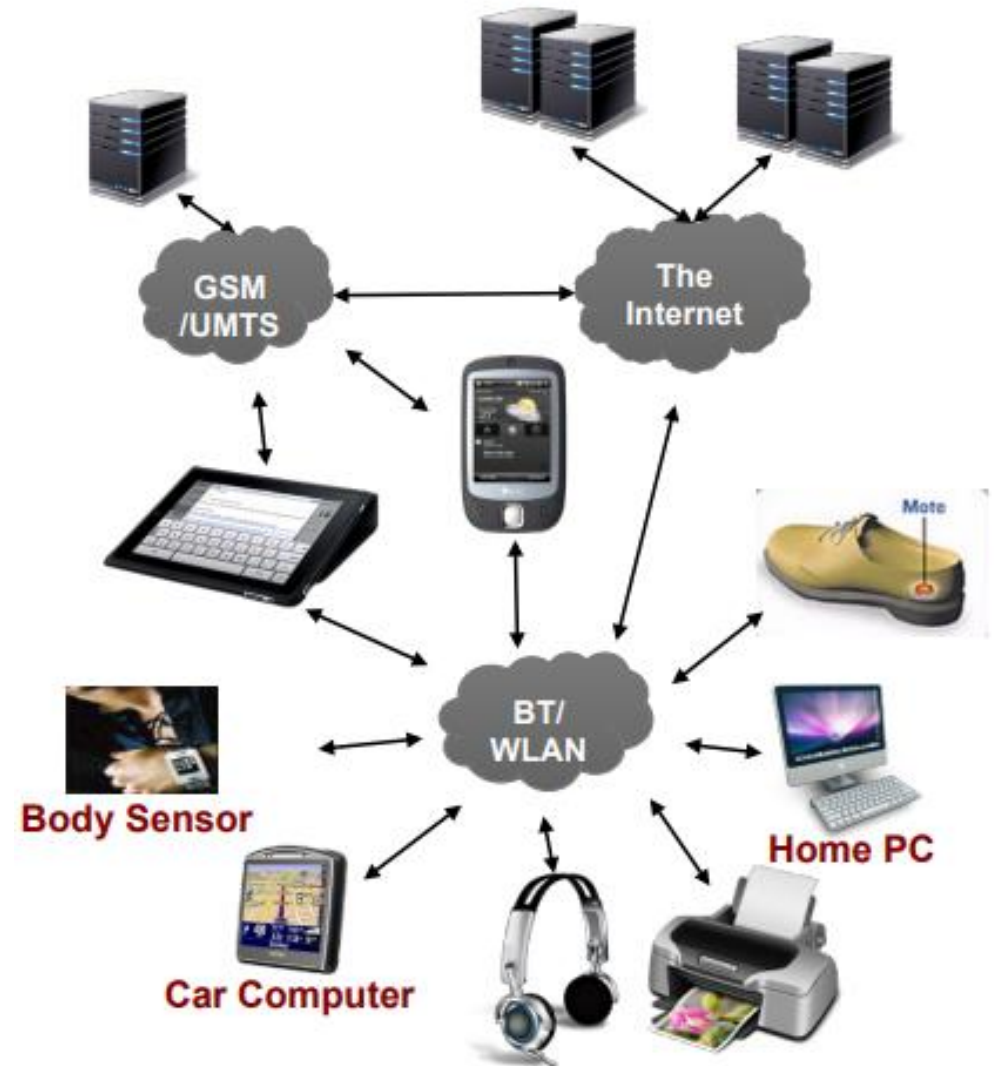
# Motivation

- **Mobile computing:** exploiting the connectedness of portable devices
- **Ubiquitous computing:** exploiting the increasing integration of services and (small/tiny) computing devices in our everyday physical world
- **Mobile and ubiquitous computing:** requires particular solutions in many areas caused by dynamically changing computing environment: users, devices, and software components



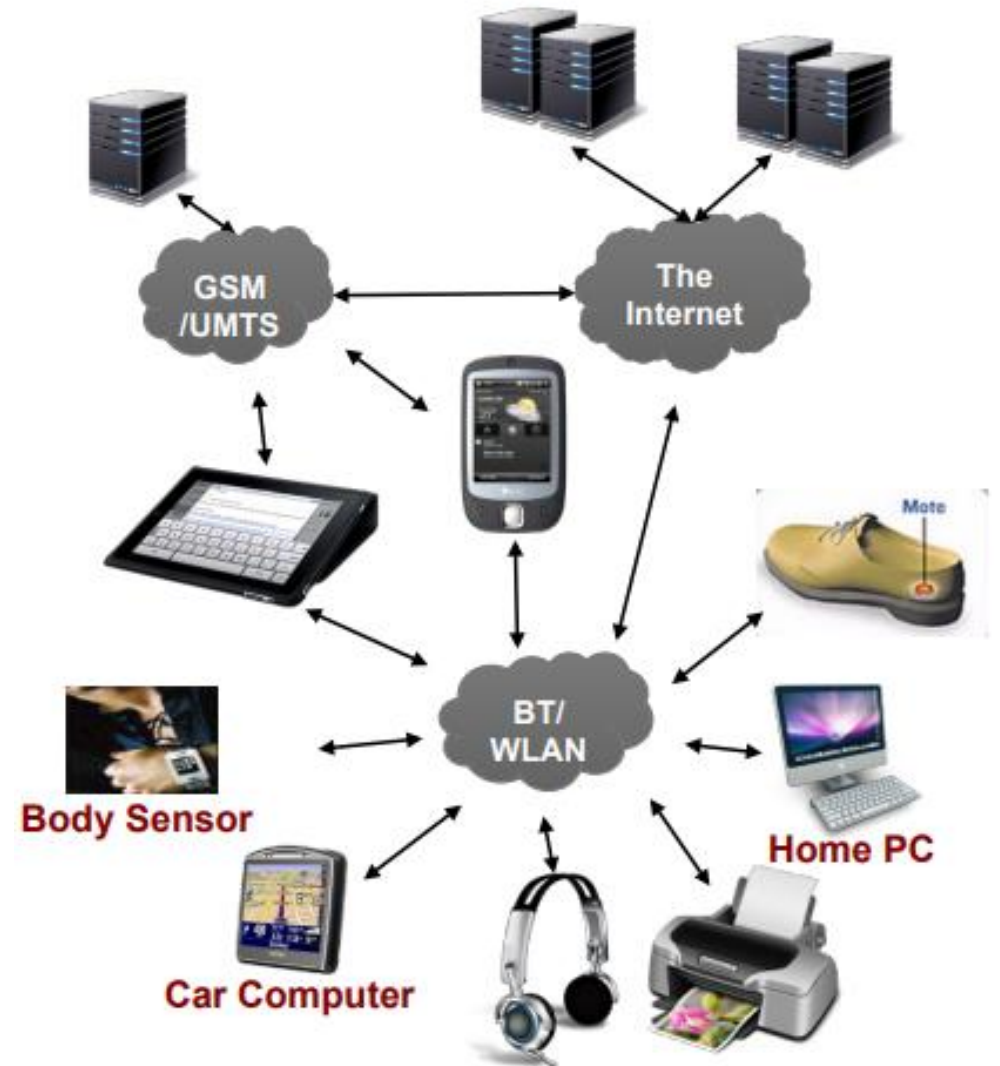
# Open Questions

- How can software components associate and interoperate with one another while devices move, fail or spontaneously appear?
- How can systems become integrated with the physical world?
- How to adapt to small devices' lack of computation and I/O resources?



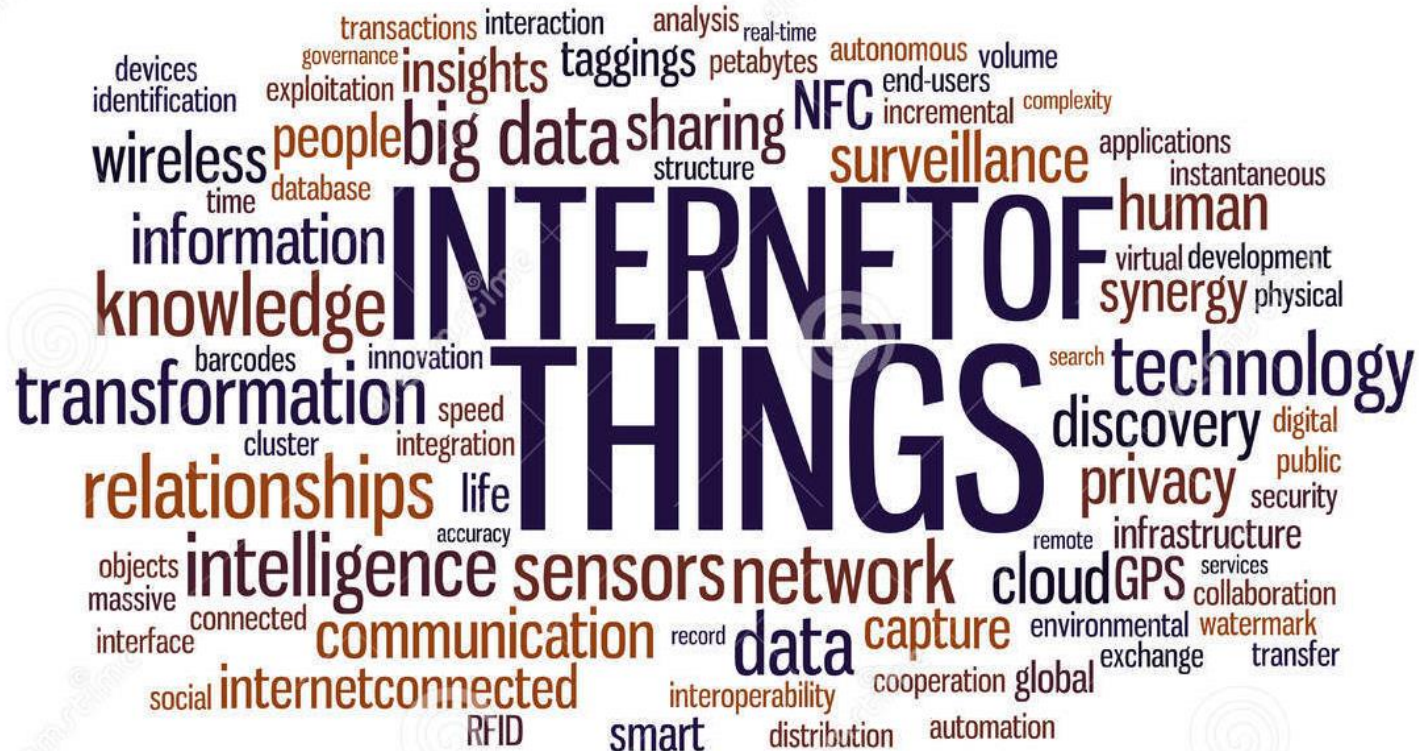
# Fields and Subfields

- Mobile computing
- Ubiquitous computing
- Wearable computing
- Context-aware computing



# Internet of Things

- the worldwide network of interconnected objects uniquely addressable based on standard communication protocols




# Volatile Systems

- Common **system model** for mobile and ubiquitous computing (and their subfields)
- **Changes** (or failures) are considered **common** rather than exceptional (in contrast to other types of systems where changes or failures are considered to be exceptions)
- Forms of volatility
  - **failures** of devices and communication links
  - **changes** in the characteristics of communication such as bandwidth
  - the **creation and destruction of associations** – logical communication relations – between software components resident on the devices
- Mobile and ubiquitous computing exhibit **all** of the above forms of volatility.

# Modeling Elements

- smart spaces
- device model
- volatile connectivity
- spontaneous interoperation

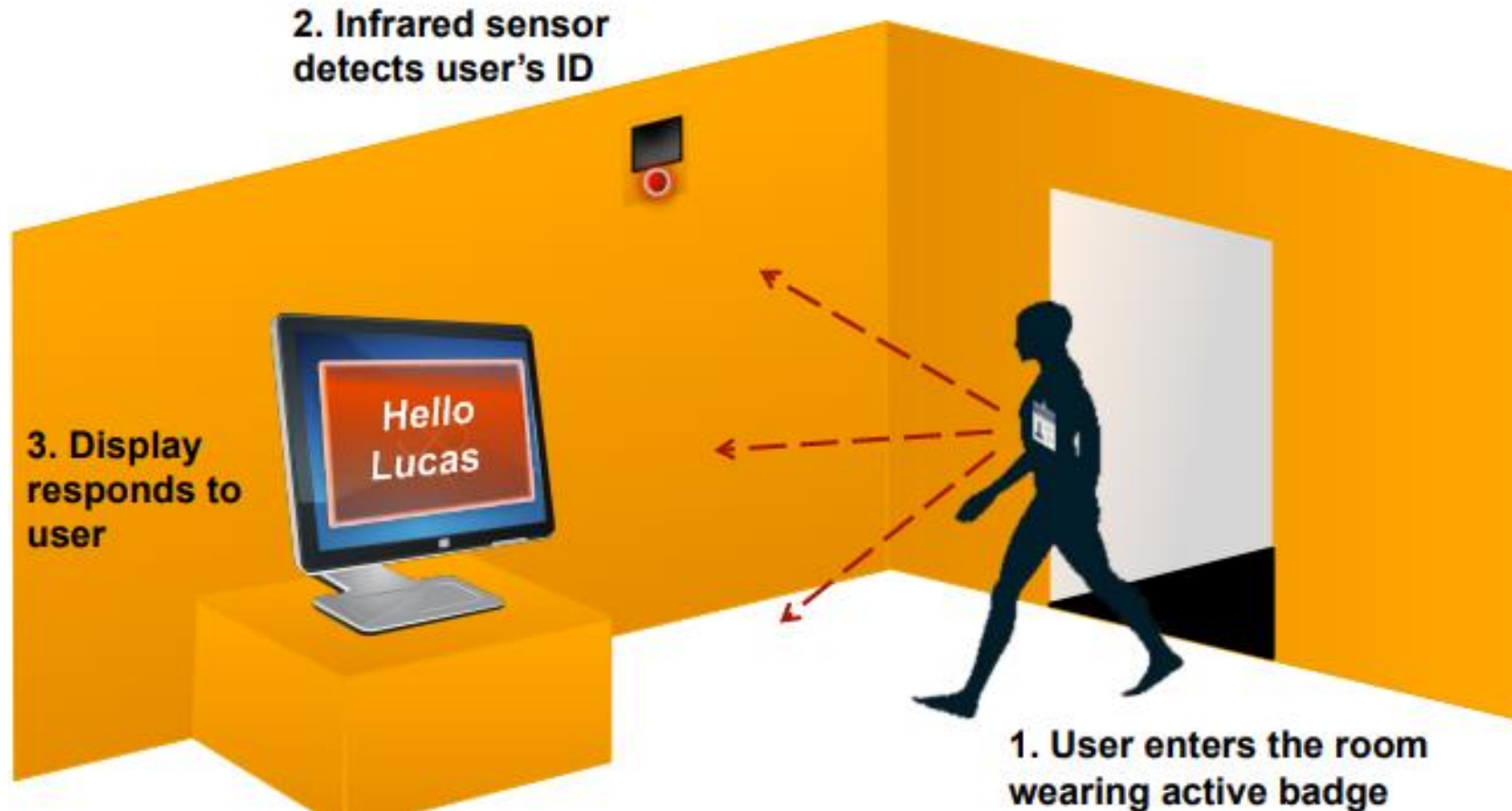
# Smart Spaces

- Environments within which volatile systems subsist
- A physical place/room with **embedded services**.
  - The services are provided only or principally within that space
- Movements or 'appearance or disappearance' in a smart space:
  - Physical mobility
  - Logical mobility 
  - Service/device appearance
  - Service/device disappearance



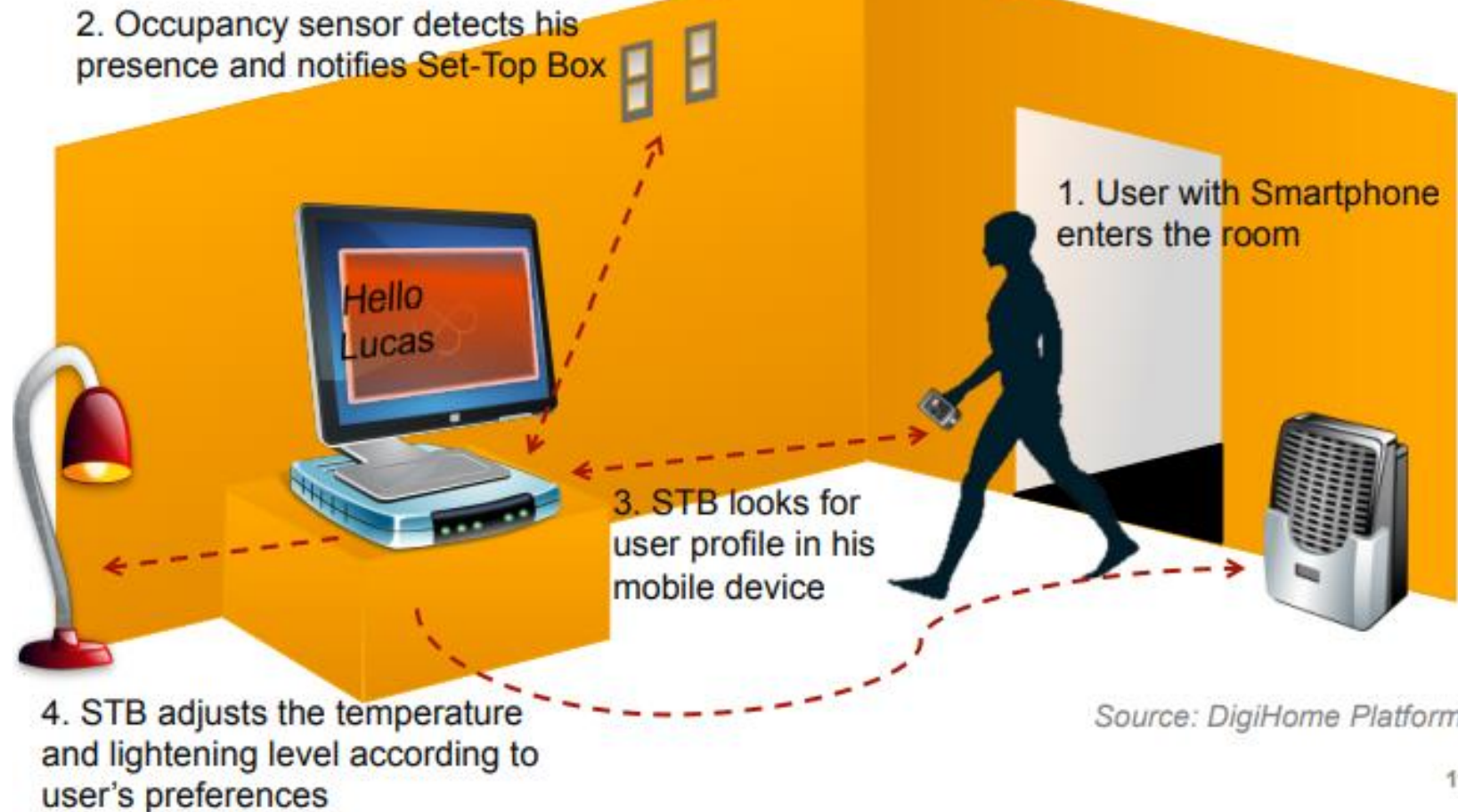
# Smart Space: Example 1

- Smart rooms: responding to a user with an active badge



# Smart Space: Example 2

- Smart homes: respond to a user with a Smartphone



# Device Model

- Limited energy
  - typically use battery
  - energy-preserving algorithms
- Resource constraints
  - relative resource poor (CPU, memory, ...)
  - algorithmic challenge
- Sensor and actuators
  - to make a device context-aware
- Examples: Motes, smart phones, etc



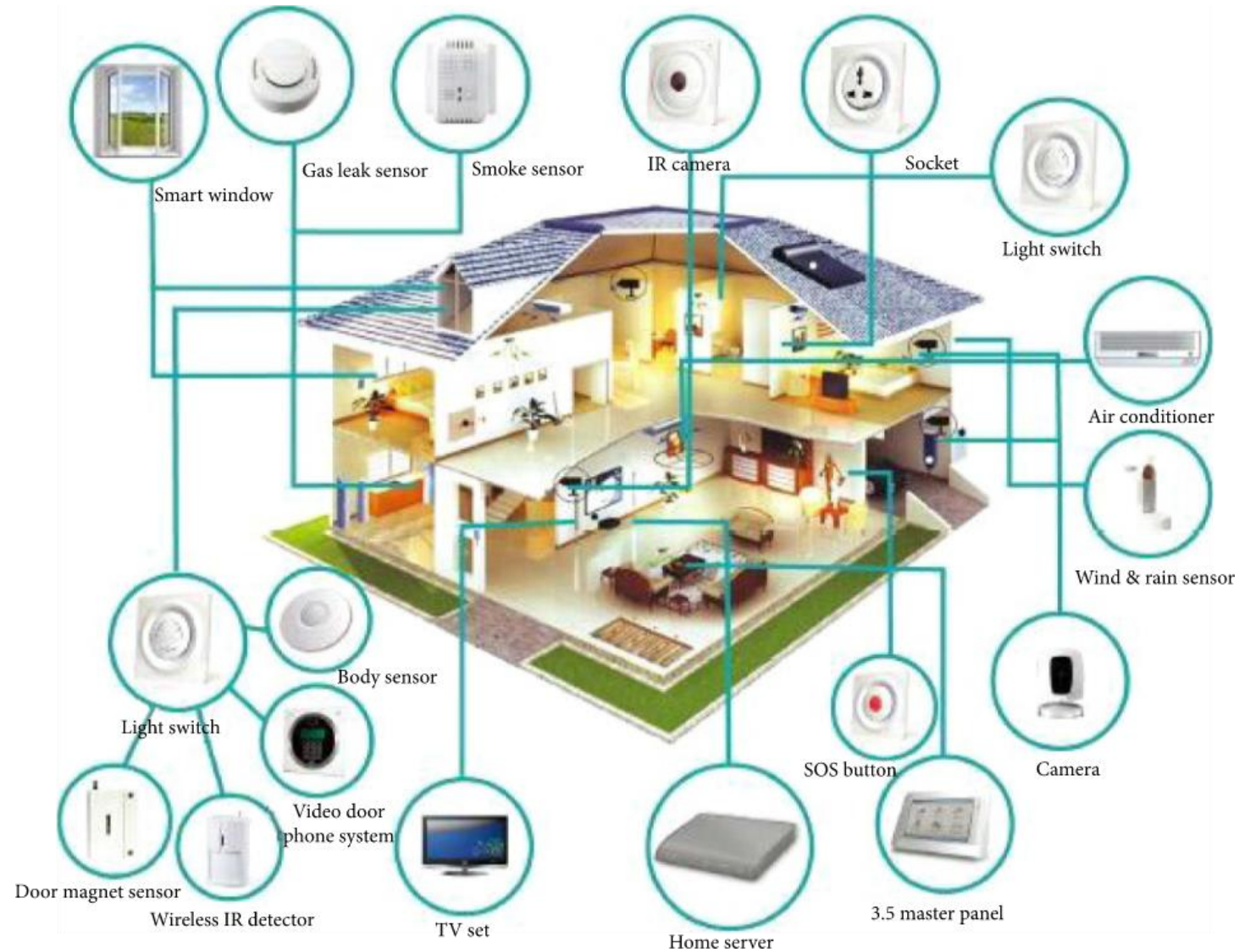
Car computer



Body sensor



# Smart Home





# Volatile Connectivity

- Variation between different technologies (Bluetooth, WiFi, 3G, 4G, 5G etc.)
  - Bandwidth, latency
  - Energy costs
  - Financial costs to communicate
- Disconnection
  - More likely in wireless networks
- Variable bandwidth and latency
  - Packet loss due to weak signal
  - Signal strength varies
  - Difficult to determine timeout-values in higher layer protocols due to varying conditions

# Spontaneous interoperation

- In volatile systems, components routinely change the set of components they communicate with
  - take advantage of possibility to communicate with local components in a smart space, or a device may want to offer services to clients in its local environment
- **Association:** a logical relationship formed when at least one of a given pair of components communicates with the other over some well-defined period of time
- **Interoperation:** interaction during an association
- **Spontaneous** interoperation: interoperation that is not planned or designed in!

# Pre-configured vs Spontaneous Associations

Preconfigured	Spontaneous
Service-driven: <i>email client and server</i>	Human-driven: <i>web browser and web servers</i> Data-driven: <i>P2P file-sharing applications</i> Physically driven: <i>mobile and ubiquitous systems</i>

---

# Association

- **Requirement:** a device that appears in a smart space needs to bootstrap itself in the smart space
- Two steps for **bootstrapping** itself:
  - Network bootstrapping (DHCP-server)
  - Zero Configuration Networking (Apple's Bonjour)
  - Establish associations between components on the device and services in the smart space
- The **association** problem
  - With which components of the many devices in the space should the components on the appearing device interoperate?
  - How to constrain the scope to services in the smart space only (e.g., the hotel room)?
  - 'Boundary principle':
    - smart spaces need to have system boundaries that correspond accurately to meaningful spaces as they are normally defined (territorially or administratively)



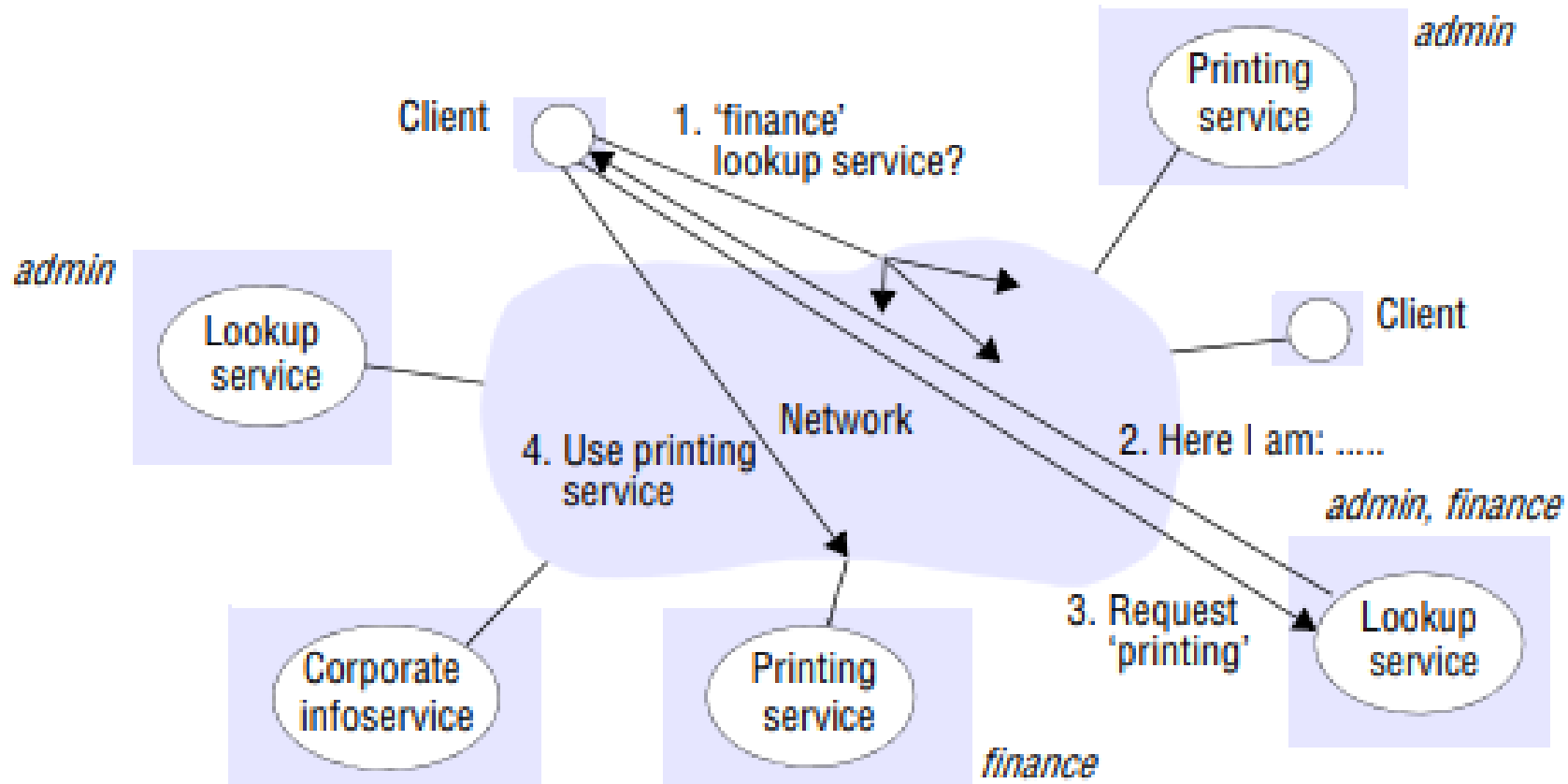
# Discovery services

- A directory service that is used to register and look up services in a smart space
- Requirements to discovery services
  - Service **attributes** is determined at runtime (hard!)
  - Service discovery must be possible in a smart space **without infrastructure** to host a service discovery service
  - Registered services may **spontaneously disappear**
  - The protocols used for accessing the directory need to be sensitive to the **energy** and **bandwidth** they consume (cf. device model)

# Interface to a Discovery Service

Methods for service registration and deregistration	Explanation
<i>lease := register(address, attributes)</i>	Register the service at the given address with the given attributes; a lease is returned
<i>refresh(lease)</i>	Refresh the lease returned at registration
<i>deregister(lease)</i>	Remove the service record registered under the given lease
Method to look up a service	
<i>serviceSet := query(attributeSpecification)</i>	Return a set of registered services whose attributes match the given specification

# Service Discovery in Jini



# Interoperation

- How can components that want to associate determine **what protocol they can use** to communicate?
- Main problem is **incompatibility between software interfaces** (components need not have been designed together)
- Two approaches:
  - **Adapt interface** to each other (interface adaptation): difficult
  - Constrain interfaces to be **identical in syntax across as wide a class of components** as possible
    - Example: Unix pipes (read, write)
    - Example: The set of methods defined in HTTP (GET, POST, ...)
    - Such systems are called data oriented
    - Require additional mechanisms to describe type and value of data exchanged, such as the processing semantics of the server (difficult!)
    - Data oriented programming models: event-systems (pub/sub), tuple spaces, direct device interoperation (devices brought into direct association)

# Sensing and Context Awareness

- Systems can be integrated with the physical world through **sensing and context awareness**
- Sensing: use sensors to collect data about the environment
- **Context aware systems**: can respond to its (sensed) physical environments (location, heat, light intensity, device orientation, presence of a device, etc.) and the context can determine its (further) behaviour
- **Context** of an entity (person, place or thing): an aspect of its physical circumstances of relevance to system behaviour

# Sensors

- Combination of hardware and/or software
- Sensors are the basis for determining contextual values
  - Location, velocity, orientation, ...
  - Temperature, light intensity, noise, ...
  - Presence of persons or things (e.g., based on RFID – electronic labels - or Active Badges)
- An important aspect of a sensor is its **error model**
  - Some are simple (e.g., a thermometer often has known error bounds and distribution)
  - some are complicated (e.g., accuracy of satellite navigation units depend on dynamic factors)

# Sensing Architectures

- Four functional challenges to be overcome in designing context-aware systems
  - Integration of idiosyncratic (peculiar) **sensors**
    - Specialized knowledge to correctly deploy sensors in the physical scenario of interest
  - **Abstracting** from sensor data
    - Agreement on the meaning of contextual attributes, and software to infer those attributes from raw sensor values
  - Sensor output may need to be **combined**
    - Require output from sensors of different types in order to gather several contextual attributes that it needs to operate
  - **Context** is dynamic
    - Response to changes in context, and not simply to read a snapshot

# Sensor Architectures

- Applications normally operate on more abstract values than sensors can produce
- Sensor **abstractions are important** to avoid application level concerns with the peculiarities of individual sensors
- Therefore common **to build a software architecture** for sensor data as hierarchies
  - Nodes at a low hierarchical level provide sensor data at a low level of abstraction (longitude/latitude of a device)
  - Nodes at higher hierarchical levels (closer to the root node) provide sensor data at higher levels of abstraction (the device is in Frank's Cafe)
- Nodes at higher levels combine sensor data from lower levels both **to abstract and to increase reliability**



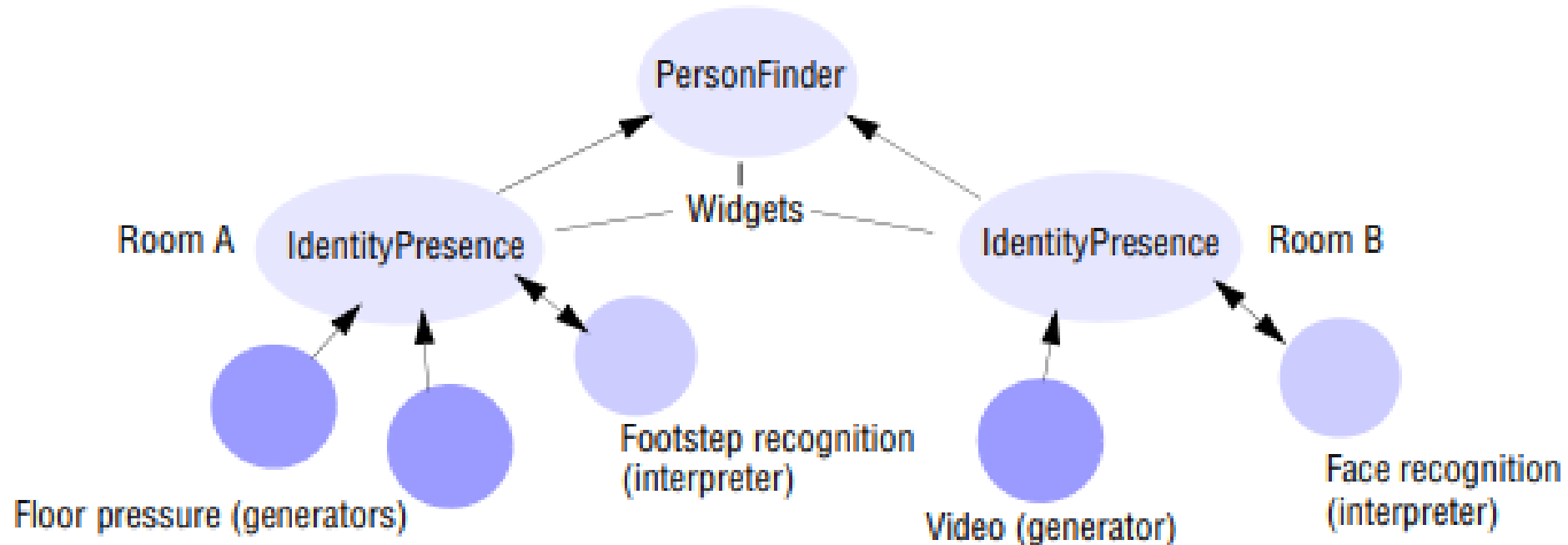
# Example: Context Toolkit

- System architecture for general context aware applications
- Based on 'context-widgets': reusable components that abstract over some types of context attributes (hide low level sensor details)
- Example: Interface to a IdentityPresence widget class

Attributes (accessible by polling)	Explanation
<i>Location</i>	Location the widget is monitoring
<i>Identity</i>	ID of the last user sensed
<i>Timestamp</i>	Time of the last arrival
Callbacks	
<i>PersonArrives(location, identity, timestamp)</i>	Triggered when a user arrives
<i>PersonLeaves(location, identity, timestamp)</i>	Triggered when a user leaves

# Example: Context Toolkit

- Example of use of IdentityPresence widget
- A PersonFinder widget constructed by using IdentityPresence widgets
- 



# Wireless Sensor Networks (WSNs)

- Network consisting of a (typically high) number of **small, low-cost units or nodes** that are more or less arbitrary arranged (e.g., “thrown out” in high numbers in a certain geographical area)
- Self-organising (ad-hoc network), functions independently of an infrastructure
- The nodes have sensing and processing capacity, can communicate wirelessly with a limited range (save energy), and act as routers for each other
- Are volatile systems because nodes can fail (battery exhaustion or otherwise destroyed (e.g., fire)), connectivity can change due to node failures

# WSNs: Architectural Features

- Features: driven by requirements of **energy conservation** and **continuous operation**
- **In-network processing**: The nodes have processing capabilities because processing is much less costly in energy consumption than (wireless) communication. Can be exploited to reduce the need for communication (only communicate when there is a need for it)
- **Disruption-tolerant networking**: based on store-and-forward transfer of data (not end-to-end)
- **Data oriented programming of nodes**: since nodes can fail, we can not rely on programming techniques for sensor nodes that refer to single nodes

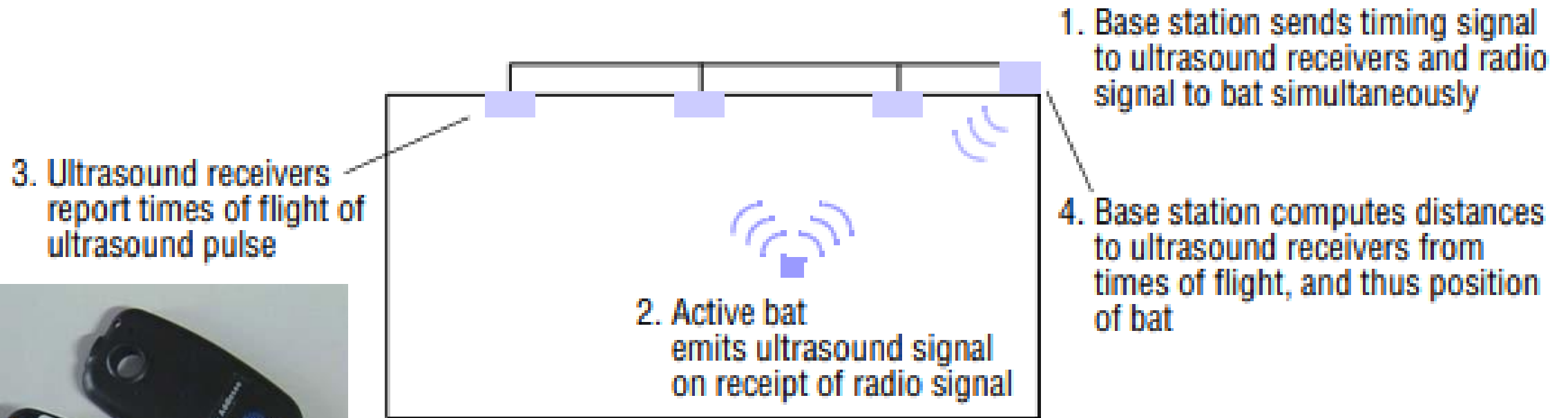
# Location-Sensing

- Location is an obvious parameter for mobile, context-aware computing
- It seems natural to make applications and devices behave in a way that depends on where the user is, such as the context-aware phone
- Location-sensing systems
  - Designed to obtain data about the **position** of **entities** (objects and humans) within some type of region of interest

# Location-Sensing Technologies

<i>Type</i>	<i>Mechanism</i>	<i>Limitations</i>	<i>Accuracy</i>	<i>Type of location data</i>	<i>Privacy</i>
GPS	Multilateration from satellite radio sources	Outdoors only (satellite visibility)	1–10 m	Absolute geographic coordinates (latitude, longitude, altitude)	Yes
Radio beaconing	Broadcasts from wireless base stations (cellular, 802.11, Bluetooth)	Areas with wireless coverage	10 m–1 km	Proximity to known entity (usually semantic)	Yes
Active Bat	Multilateration from radio and ultrasound	Ceiling-mounted sensors	10 cm	Relative (room) coordinates	Bat identity disclosed
Ultra Wide Band	Multilateration from reception of radio pulses	Receiver installations	15 cm	Relative (room) coordinates	Tag identity disclosed
Active Badge	Infrared sensing	Sunlight or fluorescent light	room size	Proximity to known entity (usually semantic)	Badge identity disclosed
Automatic identification tag	RFID, Near Field Communication, visual tag (e.g. barcode)	Reader installations	1 cm–10 m	Proximity to known entity (usually semantic)	Tag identity disclosed
Easy Living	Vision, triangulation	Camera installations	Variable	Relative (room) coordinates	No

# Locating an Active Bat within a Room



# Adaptation

- The devices in the volatile systems are heterogeneous
  - processing power
  - input/output capabilities such as screen size
  - network bandwidth
  - memory
  - energy capacity
- Heterogeneity is unlikely to ease significantly because of the multiple purposes we have for devices.
- Runtime conditions prone to dramatic changes
  - available bandwidth
  - energy



# Adaptive System

- A model of varying resources
- Adapt runtime behavior to the current resource availability
- To accommodate heterogeneity by allowing software reuse across contexts
  - Device capabilities
  - User preferences
- To accommodate changing runtime resource conditions by adapting application behavior without sacrificing crucial application properties

# Context-aware adaptation of content

- Multimedia applications operate by exchanging or streaming multimedia data such as images, audio and video.
- Simple approach
  - send the same content regardless of the content-consuming device
  - device to render the content appropriately for its needs and limitations
- Bandwidth limitations and device heterogeneity make that approach impractical in general
- The capabilities of devices in volatile systems to receive, process, store and display multimedia content vary widely
  - screen sizes, keyboards, microphones, audio output, etc.

# Context-aware adaptation of content (Cont.)

- the content that a service needs to deliver to a given device is a function of the context
- the media producer should take account not only of the consuming device's capabilities, but also of such factors as the preferences of the device's user and the nature of the user's task
  - one user might prefer text to images on a small screen
  - another might prefer audio output to visual output
- The items that the service delivers within a piece of content may need to be a function of the user's task
  - features required in a map of a given region: a tourist looking for attractions or a worker looking for infrastructure access points

# Context-aware adaptation of content (Cont.)

- Adaptation Processes:
  - Adapt the original data programmatically into a suitable form
  - Generating content
  - Transforming content
- Adaptation can occur within media types
  - selecting from map data
  - reducing the resolution of an image
- Adaptation can occur across media types
  - converting text to speech or vice versa, according to the user's preferences or whether the consuming device has a display or supports audio output

# Context-aware adaptation of content (Cont.)

- adaptation for bandwidth-limited devices is type-specific compression
- proxies perform compression on the fly between services and clients
- To accommodate limited bandwidth, compression should be lossy but specific to the media type, so that semantic information can be used to decide which media features it is important to retain.
  - an image can be compressed by throwing away colour information.
- Transcoding should be performed on the fly
  - statically pre-prepared content forms will not provide sufficient flexibility to cope with dynamic data and an increasing set of permutations of clients and services.
- Transcoding should be performed in proxy servers so that both clients and services are transparently separated from transcoding concerns.

# Adapting to changing system resources

- Hardware resources such as screen size are heterogeneous across devices are at least stable and well known
- Applications also rely on resources that are subject to change at runtime and that may be hard to predict
  - available energy and network bandwidth

# OS support for adaptation to volatile resources

- Three approaches to adaptation
- Applications to request and obtain resource reservations
  - satisfactory QoS guarantees are sometimes difficult to achieve in volatile systems and are impossible in cases such as energy depletion
- Notify the user of changed levels of resource availability
  - if bandwidth becomes low, the user of a video player could operate a slider in the application to switch the frame rate or resolution
- Notify the application of changing resource conditions
  - application to adapt according to its particular needs

# Taking advantage of smart space resources

- Cyber foraging
  - a processing-limited device discovers a compute server in a smart space and offloads some of its processing load
  - increase application responsiveness for the user
  - Along with *energy-aware adaptation*
- Challenging requirements associated with cyber foraging
  - Application needs to be decomposed
  - Part of it can be processed efficiently on a compute server
  - Application should still function correctly if no compute server is available
  - Compute server should run a part of the application that involves relatively little communication with the portable device



# Adaptation Summary

- Adapt the behaviour of mobile applications (applications running on mobile devices) to a dynamically varying context
  - Varying capabilities of different devices
  - Varying resource availability
  - User needs and wishes
- Need for adapting the application to a dynamically varying context
  - Adapt application to resource situation (battery, bandwidth, memory)
    - – Dynamically adapt media quality (e.g., video) to available bandwidth and/or to user preferences, and/or to device capabilities (scaling and/or transcoding within same media type or between media types)
  - Dynamically adapt user interface to situation of user or the orientation of the device
  - Adapt application to availability of devices and services in the environment (ubiquitous services)

# Summary

- Most challenges to mobile and ubiquitous systems are caused by their volatile nature
- In such environments applications need to be context aware and adaptive
  - Integrated with the physical world through sensing and context awareness
  - Adapt to changes in the physical circumstances by changing behavior (e.g. component reconfiguration)
- There are many challenges, but yet only few (comprehensive) solutions