Distributed Systems Lecture 1

Professor Louise E. Moser

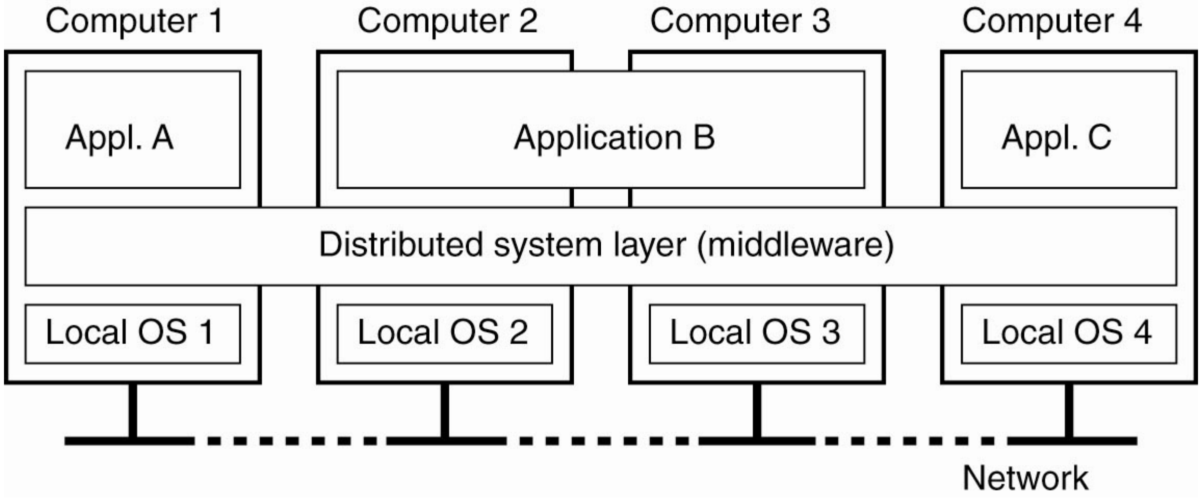
Spring 2014

Course Administrative Details

- **Lectures:** T Th 11am-12:15pm Phelps 2524
- **Discussion Sections:** Th 5-5:50pm Phelps 1444
  F 10-10:50am Phelps 1440
- **Office Hours:**
  - Prof Louise Moser: T Th 2-3:30pm HFH 2164
  - TA Miguel Lastras Montano: M W 3-4pm, Th 4-5pm Phelps 1435
    TA Harichandan Pulagam: M W 1-2pm, F 11-12noon Phelps 1435
- **Text:** Distributed Systems: Principles and Paradigms
  A. S. Tanenbaum and M. van Steen, Prentice Hall, Second Edition
- **Midterm:** Tue, May 6, 11-12:15pm, **Final:** Wed, June 11, 12-3pm
- **Grading:** Midterm 15%, Final 35%, Assignments 35%
  Class Attendance / Participation 15%
  (Lectures and Discussion Sections)

Definition of a Distributed System

- A **Distributed System** is a collection of two or more independent computers that appears to its users as a single coherent system
- A distributed system is typically organized as middleware that extends across multiple computers and that offers each application the same interface



Goals of a Distributed System

- Overall Goal: Connecting resources and users
- Additional Goals:
  - **Transparency**
  - **Openness**
  - **Scalability**

Different Forms of Transparency

Transparency	Description
Access	Hide differences in data representations and how a resource is accessed
Location	Hide where a resource is located
Relocation	Hide that a resource may be moved to another location
Migration	Hide that a resource may be moved to another location while it is being used
Replication	Hide that a resource may be available on several distinct computers
Concurrency	Hide that a resource may be shared by several competing users at the same time
Failure	Hide the failure and recovery of a resource

Degree of Transparency

Complete transparency might not be desirable or possible

- Users might be located in different countries - distribution is apparent and might not be something you want to hide
- Complete transparency can result in extra performance costs
  - **Keeping Web caches *exactly* up-to-date with the master copy**
  - **Immediately copying the results of write operations to disk**
- Hiding faults in hardware and software might not be theoretically or practically possible
  - **You cannot distinguish a slow computer from one that has failed**
  - **You cannot be sure that a computer has actually performed an operation before it crashed**

Openness

- An **open** distributed system is one that interacts with services from other open systems and requires
  - **Well-defined interfaces and protocols**
  - **Portability of applications**
  - **Interoperability of systems**
- An open distributed system is one that is independent of the heterogeneity of the underlying environment
  - **Languages**
  - **Operating systems**
  - **Hardware**

Implementing Openness

- **Policies** specified by the applications and the users
  - **What level of consistency do we require for client-cached data?**
  - **Which operations do we allow downloaded code to perform?**
  - **Which QoS requirements do we adjust for different applications?**
  - **What level of security/privacy do we require for communication?**

- **Mechanisms** implemented by the middleware and the operating system
  - **Allow dynamic setting of caching policies, preferably per cached item**
  - **Support different levels of trust for downloaded code**
  - **Provide adjustable QoS parameters per data stream**
  - **Offer different encryption algorithms**

Scalability

- Limitations of scalability due to centralization

Concept	Example
Centralized services	A single server for all users
Centralized data	A single on-line telephone book
Centralized algorithms	Doing routing based on complete information

Scalability

- Characteristics of **decentralized** algorithms
  - **No machine has complete information about the state of the distributed system**
  - **Machines make decisions based on local information**
  - **No implicit assumption that a global clock exists**
  - **Failure of one machine does not affect the operation of the algorithm**

Scalability Metrics

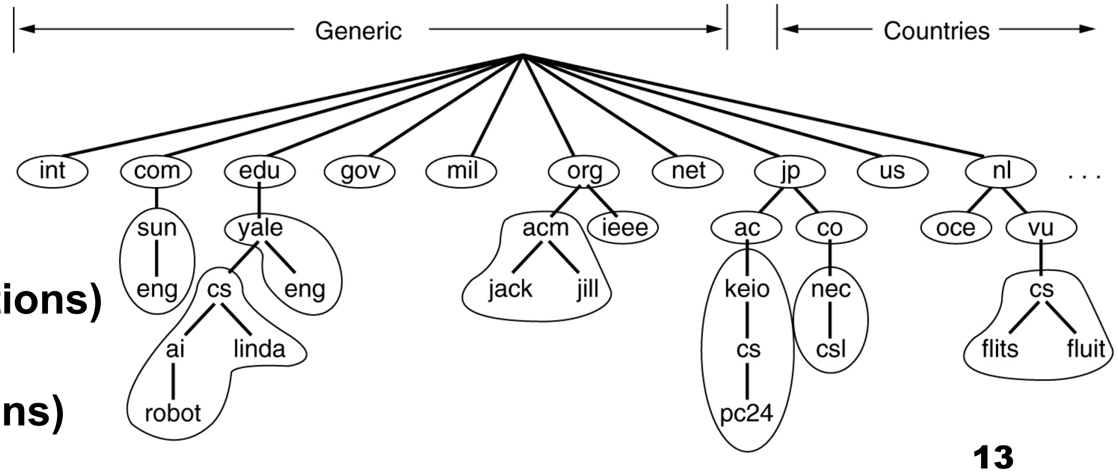
- Many developers of distributed systems use the term "scalable" without making clear *how* their systems scale
- At least three scalability metrics exist
  - **Size scalability – number of users or processes or machines**
  - **Geographical scalability – maximum distance between nodes**
  - **Administrative scalability – number of administrative domains**

Scaling Techniques

- **Distribution:** Partition the data and computations across multiple machines
  - **Move computations to clients (Java applets)**
  - **Decentralized naming services (DNS)**
  - **Decentralized information systems (WWW)**
- **Caching:** Allow client processes to access local copies
  - **Web caches (browser or proxy)**
  - **File caching (client or server)**
- **Replication:** Make copies of the data available on different machines (mainly for fault tolerance)
  - **Replicated file servers**
  - **Replicated databases**
  - **Mirrored Web sites**

Example Scaling Technique

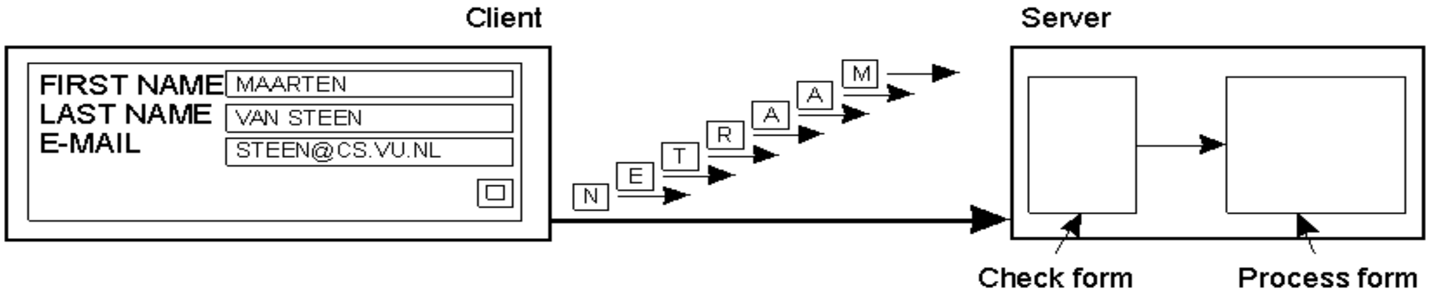
- **Domain Name System (DNS)** in the Internet is hierarchically structured into several hundred top-level domains
  - **Domains correspond to organizational boundaries, *not* physical boundaries**
  - **Domains are partitioned hierarchically in a tree structure into subdomains, etc.**
  - **To create a new domain, need permission from the manager above in the tree**
  - **Domain is named by a path, e.g., ece.ucsb.edu**
- Generic DNS names
  - **com (commercial)**
  - **edu (educational)**
  - **gov (US government)**
  - **mil (US military)**
  - **int (international organizations)**
  - **net (network providers)**
  - **org (non-profit organizations)**



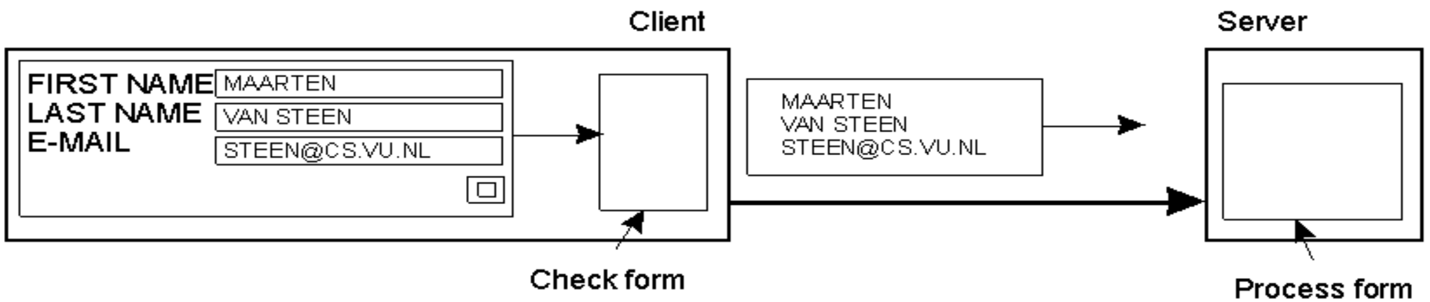
Example Scaling Technique

(a) Server checks the form for each client, or

(b) Client checks the form, which is more scalable because server can then serve more clients



(a)



(b)

Pitfalls in Developing Distributed Systems

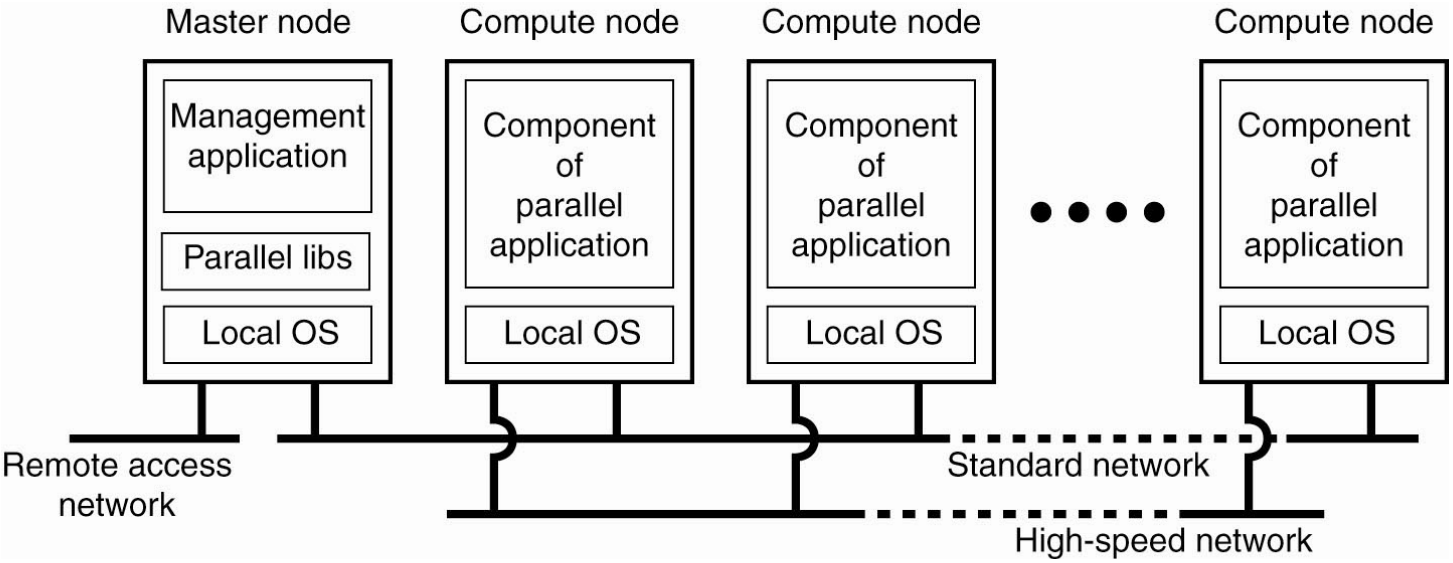
- **False assumptions** made by a first-time developer of a distributed system
  - Network is homogeneous
  - Network is secure
  - Network is reliable
  - Topology does not change
  - Latency is zero
  - Bandwidth is infinite
  - Transport cost is zero
  - There is one administrator

Types of Distributed Systems

- Distributed Computing Systems
  - Cluster Computing Systems
  - Grid Computing Systems
- Distributed Information Systems
  - Transaction Processing Systems
  - Enterprise Application Integration (EAI)
- Pervasive Distributed Systems
  - e-Healthcare Systems
  - Sensor Networks

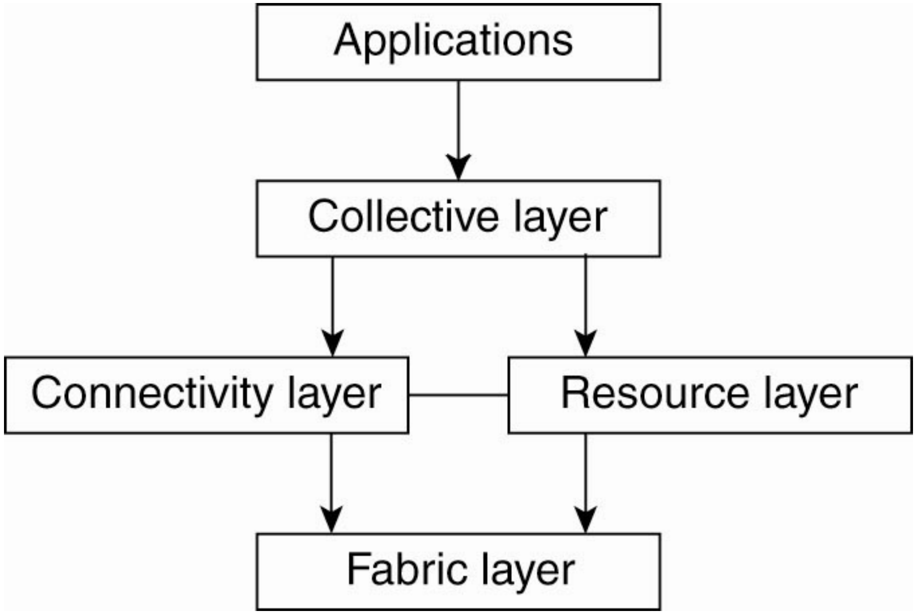
Cluster Computing Systems

- A **cluster** is a collection of workstations or PCs connected by a local-area network (LAN)
- Each node runs the same operating system (OS)
- Example: Master/slave system where the slaves (compute nodes) operate in parallel



Grid Computing Systems

- A **grid** is a federation of computing systems
  - **Each system might be in a different administrative domain**
  - **Each system might use different software, hardware, network technology**
- A layered architecture for a grid computing system



Transaction Processing Systems

- A **transaction** is a group of program statements that are intended to be executed together (i.e., atomically)
- Standard primitives for transactions

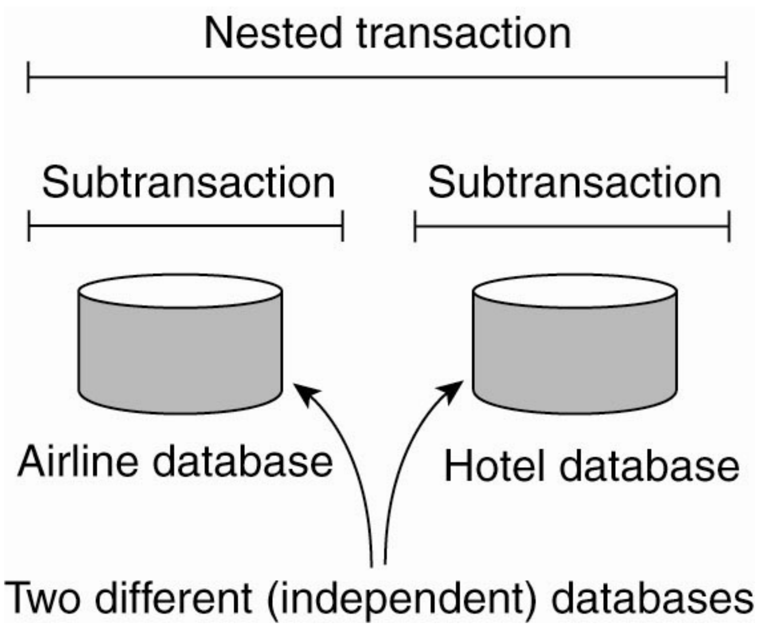
Primitive	Description
BEGIN_TRANSACTION	Mark the start of a transaction
END_TRANSACTION	Terminate the transaction and try to commit
ABORT_TRANSACTION	Kill the transaction and restore the old values
READ	Read data from a file, a table, or otherwise
WRITE	Write data to a file, a table, or otherwise

Transaction Processing Systems

- Traditionally, transactions operate on a single local database
- Such transactions are expected to satisfy the following **ACID** properties:
  - **Atomic:** **To the outside world, the transaction happens indivisibly, i.e., all of the operations of the transaction are performed, or none of them is performed**
  - **Consistent:** **The transaction does not violate system invariants**
  - **Isolated:** **Concurrent transactions do not interfere with each other; they can be serialized**
  - **Durable:** **Once a transaction commits, the changes are made permanent, by writing them to disk**
- Note: It is very difficult to guarantee the ACID properties for distributed transactions that operate on multiple databases and that span multiple enterprises

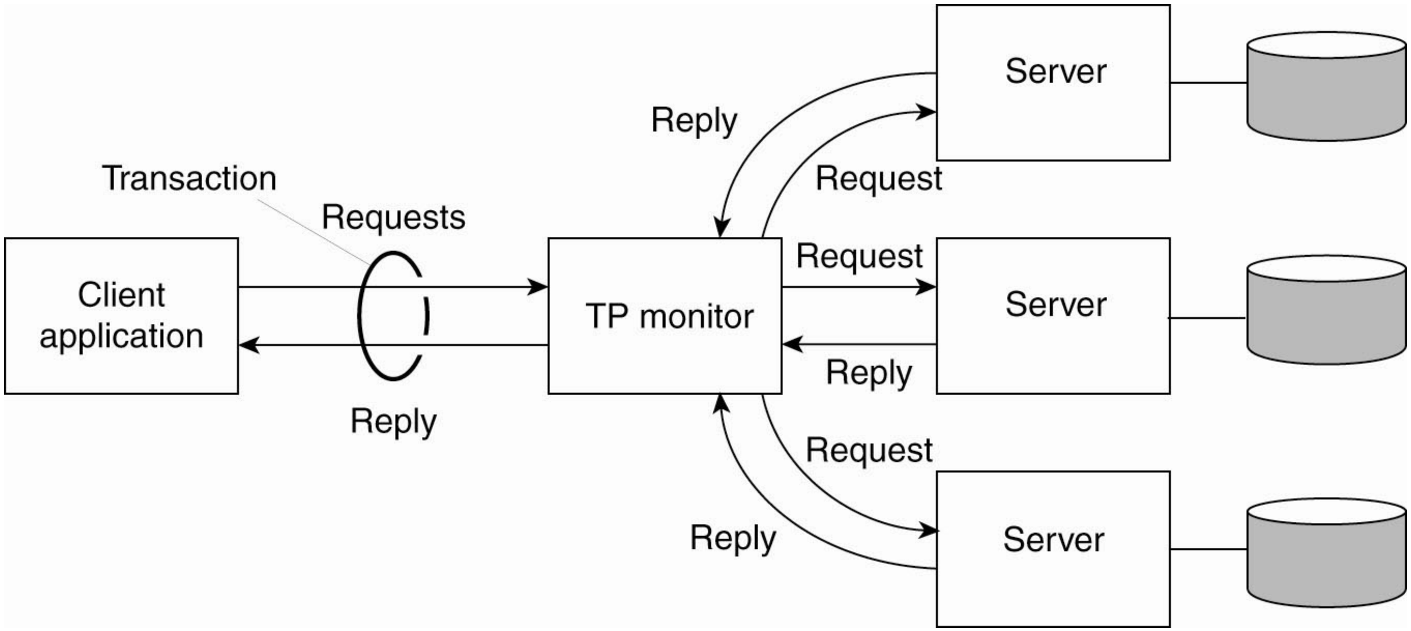
Transaction Processing Systems

- **Nested transactions** with **subtransactions**
- Subtransactions typically operate in parallel on different databases
- Durability applies only to the top-level transactions



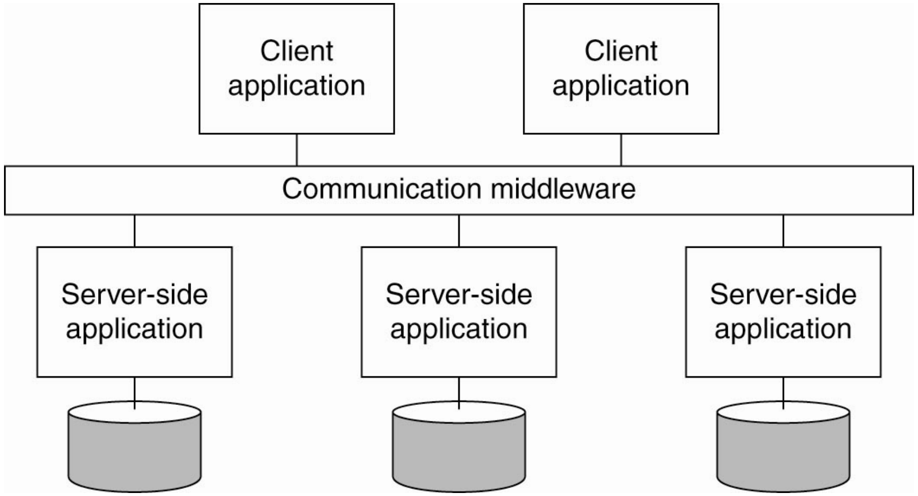
Transaction Processing Systems

- **Transaction Processing (TP) monitor** allows an application to access multiple servers/databases, using the transaction model



Enterprise Application Integration

- In **Enterprise Application Integration (EAI)**, a distributed application is separated into independent components
  - **In particular, processing components are separated from database components**
- These components need to be integrated and to communicate with one another
  - **Middleware facilitates this integration and communication in EAI**



Pervasive Distributed Systems

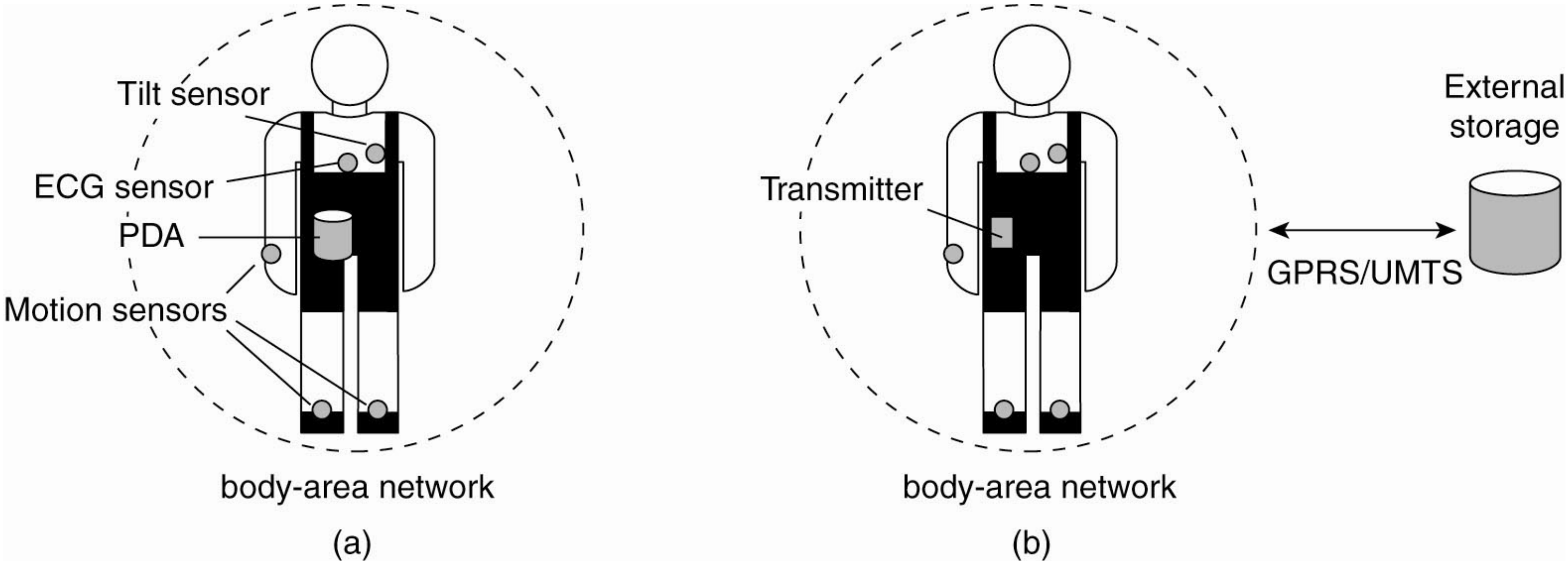
- A **pervasive distributed system** typically comprises small, battery-powered, mobile computing devices with wireless connections
  - **Part of our surroundings, lacks human administrative control**
  - **Devices must automatically discover their environment and other devices within it**
- Pervasive distributed systems requirements:
  - **Adapt to contextual changes**
  - **Enable ad hoc composition**
  - **Recognize sharing as the default**

Example: e-Healthcare Systems

- Questions to be addressed in an e-healthcare system
  - **Where and how should patient data from monitoring devices be stored?**
  - **How can we prevent loss of crucial patient data?**
  - **What infrastructure is needed to generate and propagate alerts to the patient, caregivers, healthcare professionals?**
  - **How can physicians provide online feedback?**
  - **What are the security and privacy issues and how can the proper policies be enforced?**
  - **How can robustness of the e-healthcare system be realized?**

e-Healthcare Systems

- Health monitoring of a patient using
  - (a) A local hub
  - (b) A continuous wireless connection



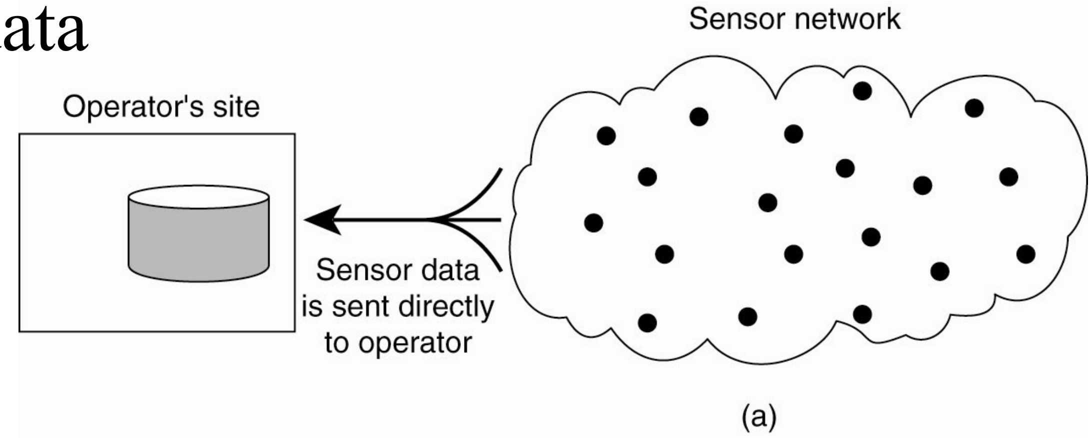
Example: Sensor Networks

- A **sensor network** typically comprises hundreds to thousands of small battery-powered nodes (each with a sensor) that use wireless communication
  - **Typically, used for measurement and surveillance**
  - **Efficiency is a prime design criterion due to limited processing and memory resources, communication capabilities, and power supplies**
- Questions concerning data processing within a sensor network
  - **How do we set up an efficient means of communication?**
  - **How do we aggregate and consolidate results?**
  - **What happens when a network link fails?**

Sensor Networks

- A sensor network database with storage and processing of data

(a) Only at the operator's site



(b) Only at the sensors