CSE 5306
Distributed Systems

Introduction

Jia Rao

http://ranger.uta.edu/~jrao/

Outline

- Why study distributed systems?
- What to learn?
- Course structure
- Course policy
- An overview of distributed systems

Why study distributed systems?

- Most computer systems today are a certain form of distributed systems
  - ✓ Internet, datacenters, super computers, mobile devices
- To learn useful techniques to build large systems
  - ✓ A system with 10,000 nodes is different from one with 100 nodes
- How to deal with imperfections
  - ✓ Machines can fail; network is slow; topology is not flat

What to learn

- Architectures
- Processes
- Communication
- Naming
- Synchronization
- Consistency and replication
- Fault tolerance and reliability
- Security
- Distributed file systems

Expected Outcomes

- Familiar with the fundamentals of distributed systems
- The ability to
  - ✓ Evaluate the performance of distributed systems
  - ✓ Write simple distributed programs
  - ✓ Understand the tradeoffs in distributed system design

Course Structure

- Lectures
  - ✓ T/Th, 3:30-4:50pm, in-person lecture at COBA 245
- Homework
  - ✓ 2 written assignments
- Projects
  - ✓ 3 programming assignments
  - ✓ 2 students team up
- Exams (close-book, close-note, one-page cheat sheet)
  - ✓ No midterm exam
  - ✓ Final exam, 2:00-4:30pm, Dec. 12, 2023

Course policy

- Grading scale
  - ✓ A [90, 100], B [80, 90), C [70, 80), D [60, 70), F below 60
- Grade distribution
  - ✓ Discussion 5%
  - ✓ Homework assignments 20%
  - ✓ Projects 40%
  - ✓ Final exam 35%
- Late submissions
  - ✓ 15% penalty on grade for each day after due day
- Makeup exams
  - ✓ No, except for medical reasons

Where to seek help

- Ask questions in class
- Ask questions on Teams
- Go to office hours
  - ✓ Instructor: Jia Rao
    - SEIR 223, email: jia.rao@uta.edu, phone: (817)-272-0770
    - Office hours: T/Th, 2:00-3:00pm or by appointment
  - ✓ TA: Mr. Lingfeng Xiang and Mr. Weishu Deng
    - Email: lingfeng.xiang@mavs.uta.edu
    - Email: weishu.deng@mavs.uta.edu

Textbook and Prerequisites

- Textbook
  - ✓ Andrew S. Tanenbaum and Maarten Van Steen, Distributed Systems: Principles and Paradigms (2nd or 3rd Edition)
- Prerequisites
  - ✓ CSE 3320: Operating Systems
  - ✓ CSE 4344: Computer Networks

CSE 5306
Distributed Systems

Overview

Distributed Systems

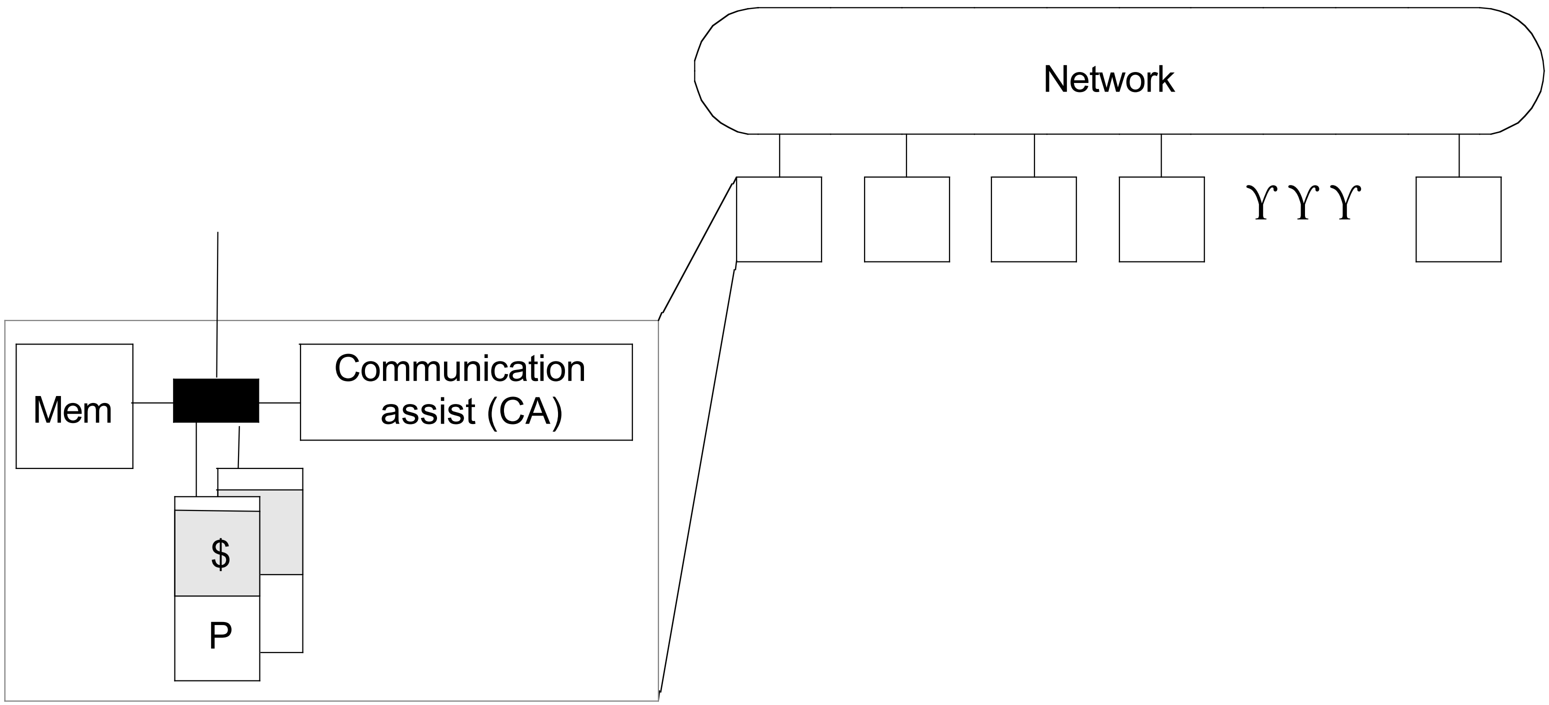
- What is a distributed system?
  - ✓ A collection of independent computers that appear to its users as a single coherent system
- Why distributed systems?
  - ✓ The ever-growing need for highly available and pervasive computing services
  - ✓ The availability of powerful yet cheap “computers”
  - ✓ The continuing advances in computer networks

Distributed v.s. Parallel Systems

- Design objectives
  - ✓ Fault-tolerance v.s. Concurrent performance
- Data distribution
  - ✓ Entire file on a single node v.s. striping over multi nodes
- Symmetry
  - ✓ Machines act as server and client v.s. service separated from clients
- Fault-tolerance
  - ✓ Designed for fault-tolerance v.s. relying on enterprise storage
- Workload
  - ✓ Loosely coupled, distributed apps v.s. coordinated HPC apps

The boundary is blurring

The Convergence of Distributed and Parallel Architectures



A generic parallel architecture

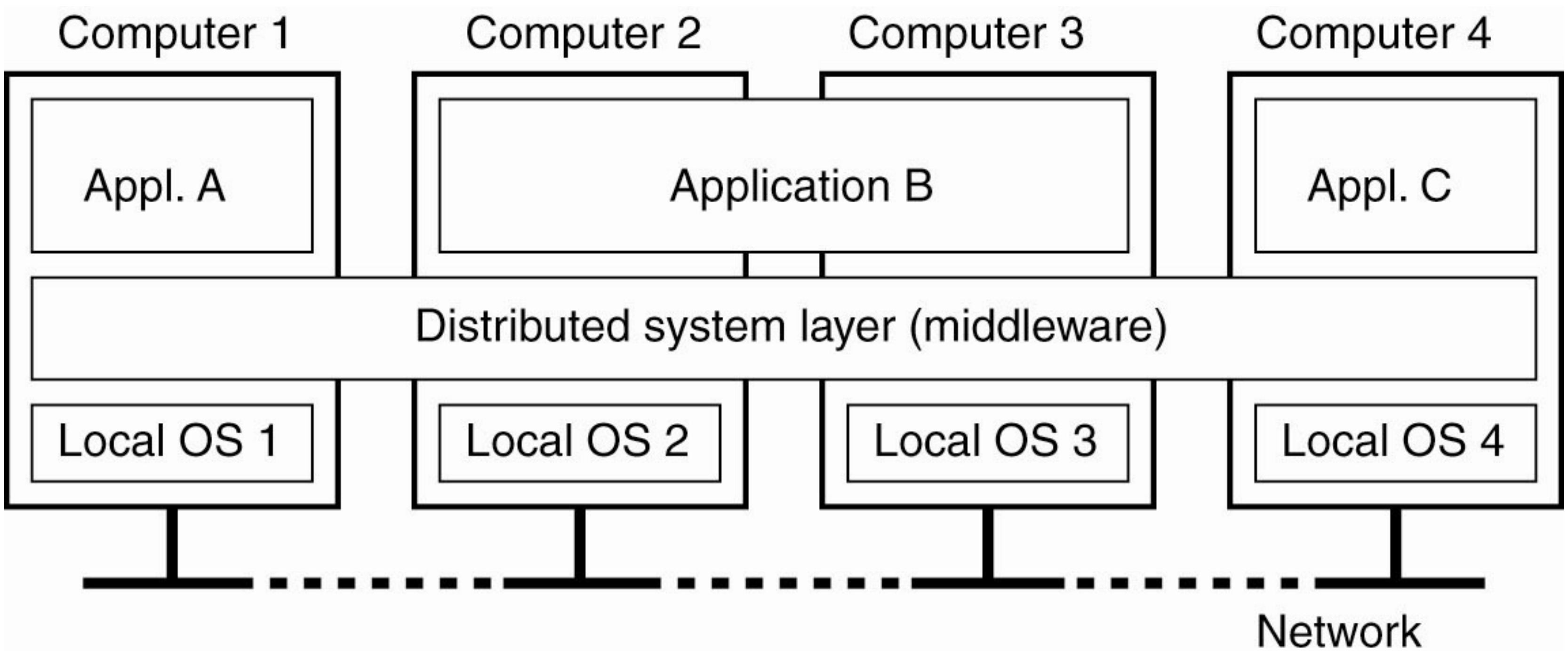
Characteristics

- Autonomous components (i.e., computers)
- A single coherent system
  - ✓ The difference between components as well as the communication between them are hidden from users
  - ✓ Users can interact in a uniform and consistent way regardless of where and when interaction takes place
- Easy to expand and replace

Advantages and disadvantages

- Advantages
  - ✓ Economics
  - ✓ More computing power, more storage space
  - ✓ Reliability
  - ✓ Incremental growth
- Disadvantage
  - ✓ Software design
  - ✓ Network
  - ✓ Failure
  - ✓ Security

Distributed System as a Middleware



The middleware layer extends over multiple machines, and offers each application the same interface

Goals of Distributed Systems

- Resource accessibility
  - ✓ Easy to access and share resources
- Distribution transparency
  - ✓ Hide the fact that resources are across the network
- Openness
  - ✓ Standard interface for interoperability and easy extension
- Performance and reliability
  - ✓ More powerful and reliable than a single system
- Scalability
  - ✓ Size scalable, geographically scalable, administratively scalable

Resource accessibility

- Benefits
  - ✓ Make sharing remote and expensive resources easily and efficiently, e.g., sharing printers, computers, storage, data, files
- Challenges
  - ✓ Security, e.g., eavesdropping, spam, DDoS attacks
  - ✓ Privacy, e.g., tracking to build preference profile

Distribution Transparency

- Access
  - ✓ Hide the difference in data representation and how a resource is accessed
- Location
  - ✓ Hide where a resource is physically located
- Migration
  - ✓ Hide that a resource may be moved to another location
- Relocation
  - ✓ Hide that a resource may be moved during access
- Replication
  - ✓ Hide that a resource may be replicated at many locations
- Concurrency
  - ✓ Hide that a resource may be shared by several competitive users
- Failure
  - ✓ Hide the failure and recovery of a resource

Openness

- Interoperability
  - ✓ Implementations from different vendors can work together by following standard rules
- Portability
  - ✓ Applications from one distributed system can be executed, without modification, on another distributed system
- Extensibility
  - ✓ Easy to add or remove components in the system
- Flexibility
  - ✓ Separating policy from mechanism

Performance and Reliability

- Performance
  - ✓ Combine multiple machines to solve the same problem
  - ✓ Transparently access more powerful machines
- Reliability
  - ✓ Use redundant hardware
  - ✓ Use software design for reliability

Scalability

- Size scalable
  - ✓ Can easily add more users or resources to the system
- Geographically scalable
  - ✓ Can easily handle users and resources that lie apart
- Administratively scalable
  - ✓ Can easily manage a system that spans many independent administrative organizations

Size Scalability

- Centralized services
  - ✓ A single server for all users
- Centralized data
  - ✓ A single database
- Centralized algorithms
  - ✓ Doing routing based on complete topology information

Size scalability problem is also faced by parallel systems but with different issues

Decentralized Algorithms

- No machine has complete information about the system state
- Machines make decisions based only on local information
- Resilient to machine failures
- No implicit assumption about a global clock

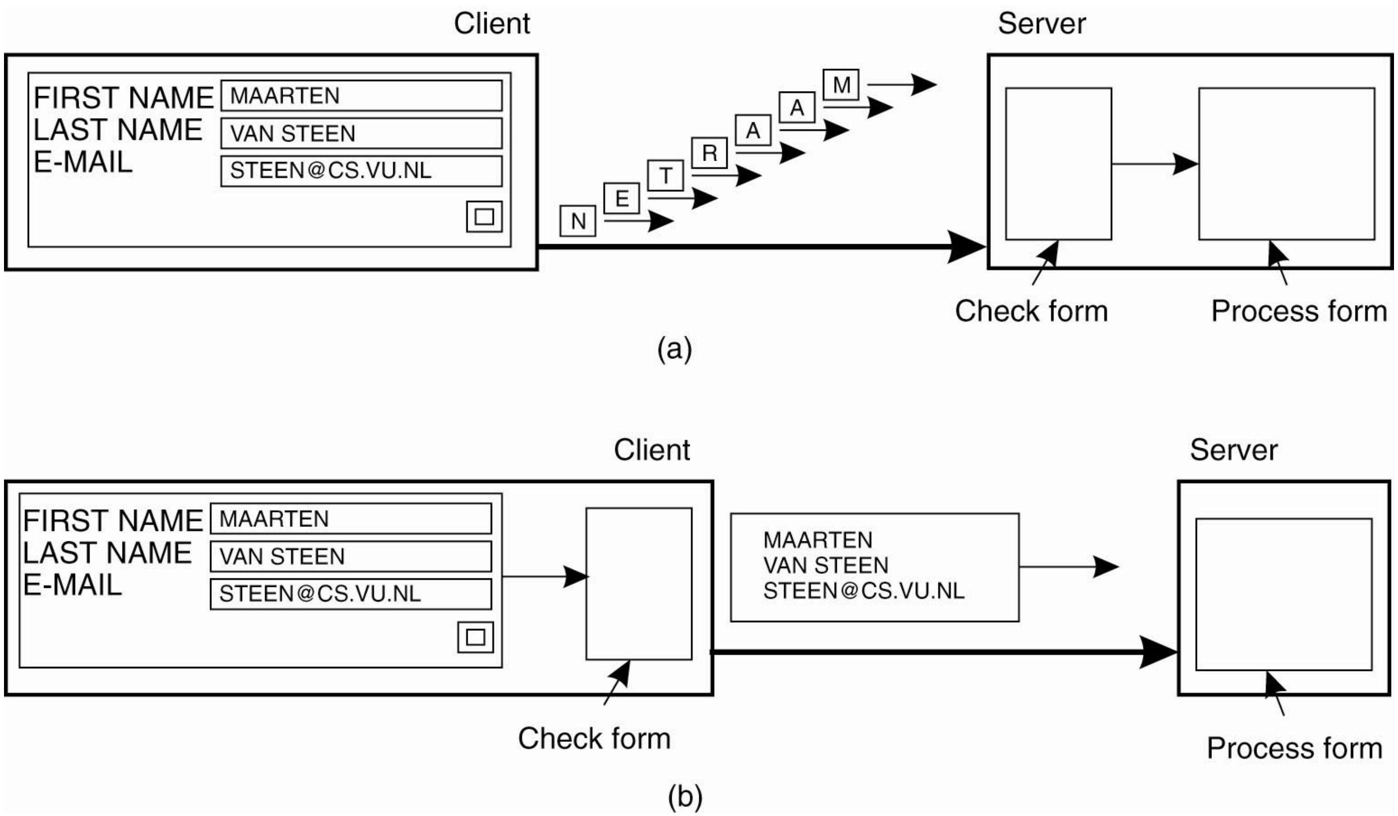
Geographical Scalability

- Challenges in scaling from LAN to WAN
  - ✓ Synchronous communication
    - Large network latency in WAN
    - Building interactive application is non-trivial
  - ✓ Assumption of reliable communication
    - WAN is not reliable
    - E.g., locating a server through broadcasting is difficult

Administrative Scalability

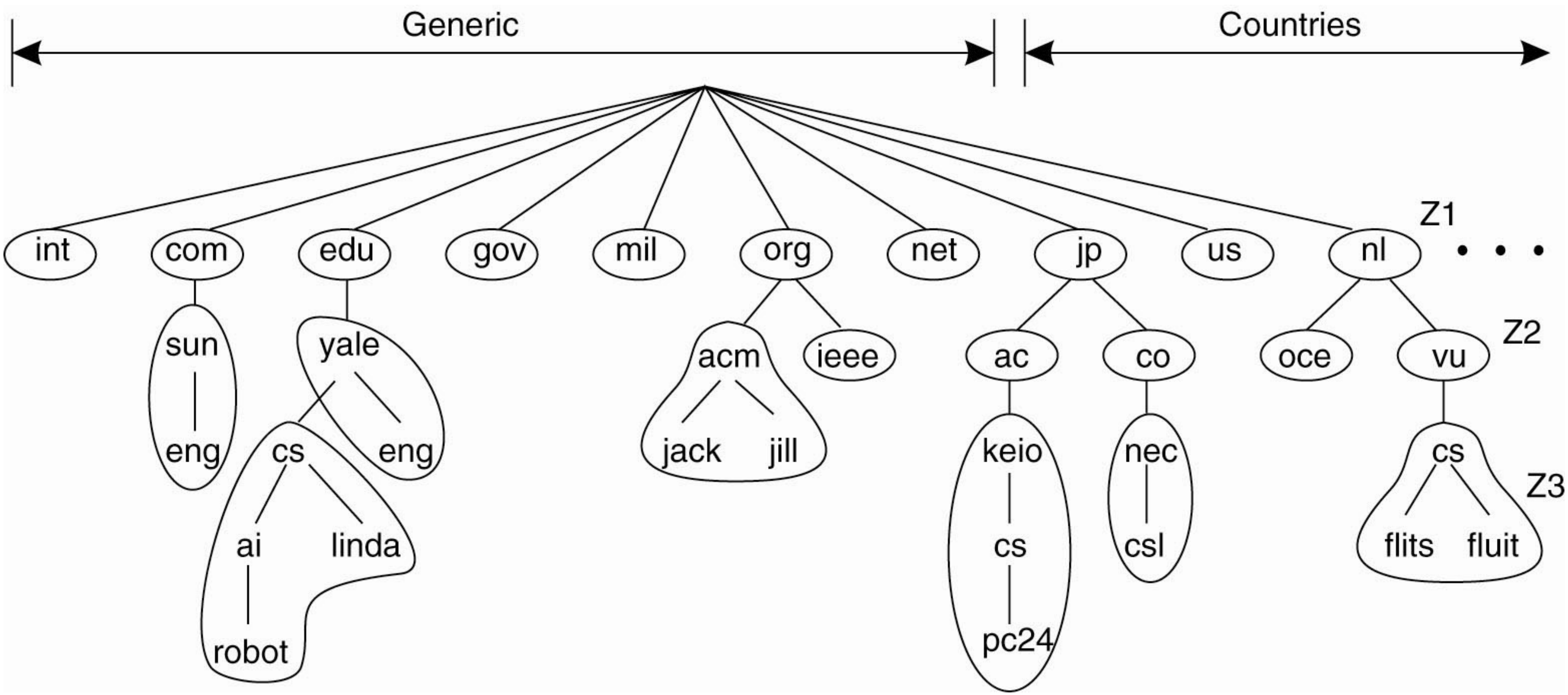
- Conflicting policies with respect to
  - ✓ Resource usage and accounting
  - ✓ Management
  - ✓ Security

Scaling techniques – hide and reduce latency



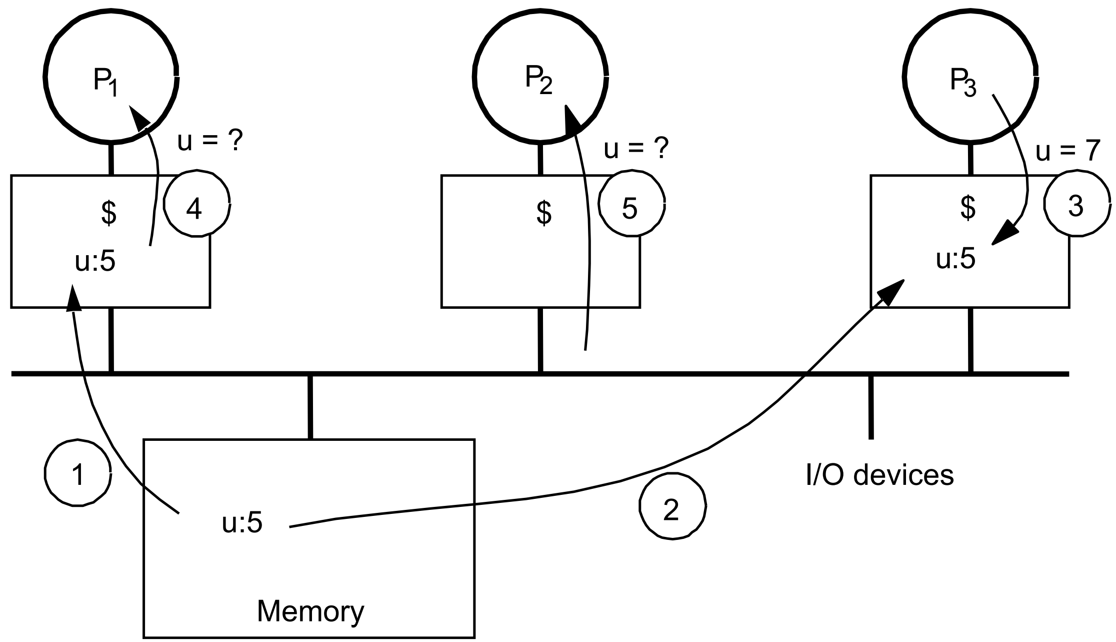
1. Use asynchronous communication
2. Move part of the computation to the client if applications can't use asynchronous communications efficiently

Scaling techniques - distribution



An example of dividing the DNS name space into zones, e.g., locating nl.vu.cs.flits

Scaling techniques - replication



Replication not only increases availability, but also helps to balance the load, leading to better performance

Key issue: how to keep replicas coherent?

Pitfalls

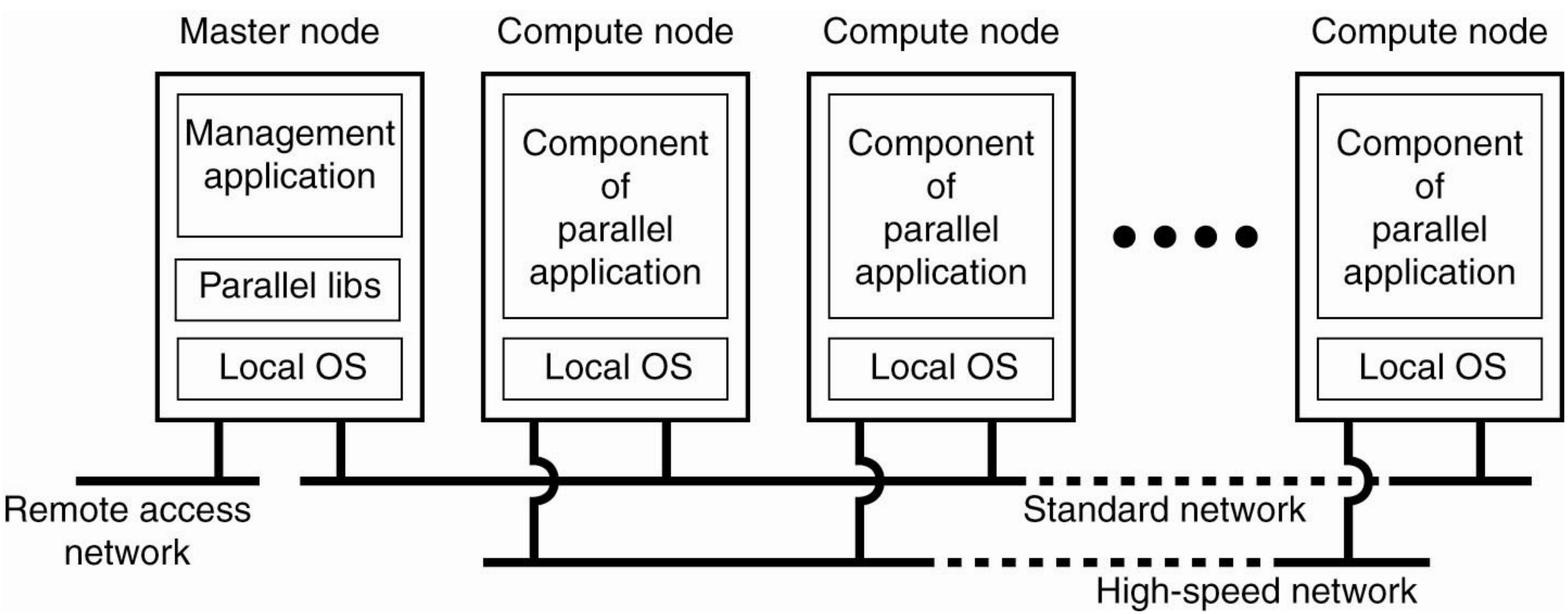
- Network is reliable
- Network is secure
- Network is homogeneous
- Topology does not change
- Latency is zero
- Bandwidth is infinite
- Transport cost is zero
- There is one administrator

Types of Distributed Systems

- Distributed computing systems
  - ✓ Cluster computing systems
  - ✓ Grid computing systems
  - ✓ Cloud computing systems
- Distributed information systems
  - ✓ Transaction processing systems
  - ✓ Enterprise application integration
- Distributed pervasive systems
  - ✓ Smart-home systems
  - ✓ Electronic healthcare systems, body area network (BAN)
  - ✓ Wireless sensor networks

Cluster Computing Systems

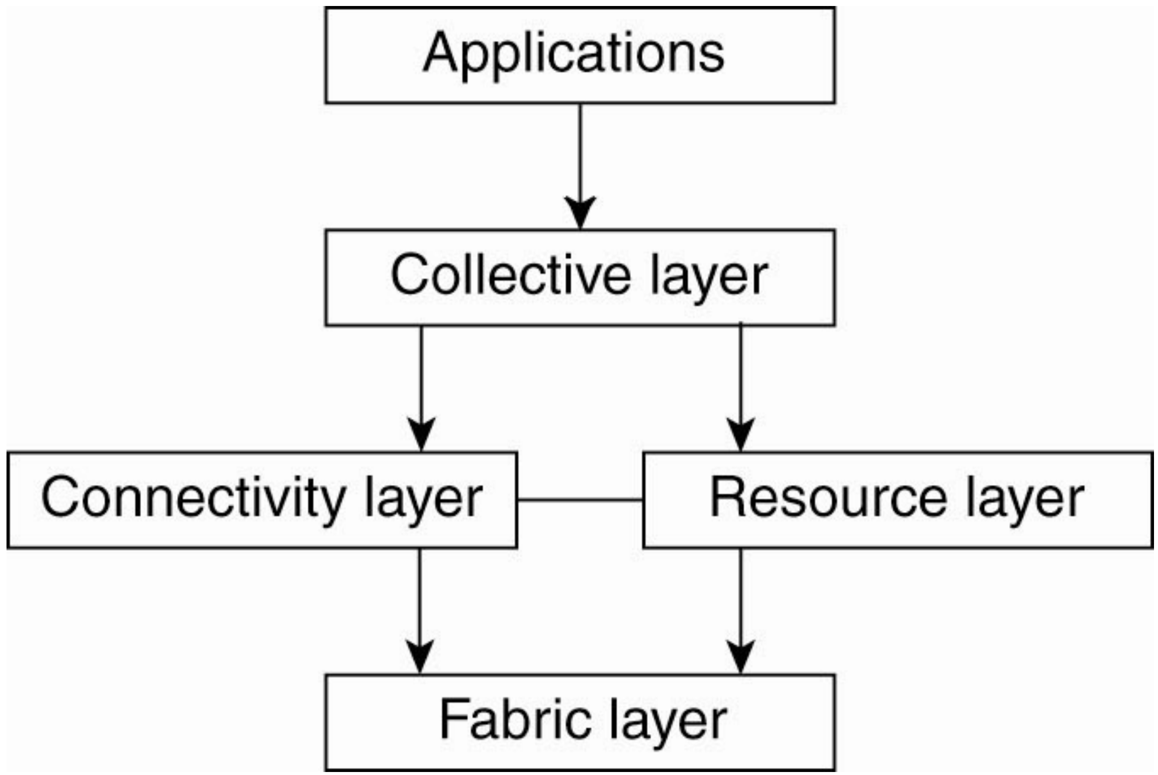
- A collection of simple (mostly homogeneous) computers via high-speed network
- Example: Linux-based beowulf architecture



Grid Computing Systems

- Grid computing
  - ✓ Has a high degree of heterogeneity
  - ✓ Has no assumption of hardware, OS, security, etc.
- Users and resources from different organizations are brought together to allow collaboration
  - ✓ Virtual organization (VO)
- Software design focus
  - ✓ Provide access to resources to users that belong to a specific VO

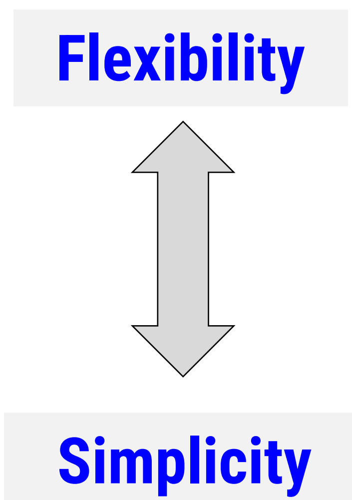
Grid Computing System Architecture



A layered architecture for grid computing systems.

Cloud Computing Systems

- Computing resources (hardware and software) are delivered as a service over the network
- Cloud computing models
  - ✓ Infrastructure as a service (IaaS)
    - Amazon EC2, Microsoft Azure
  - ✓ Platform as a service (PaaS)
    - Salesforce, Google App engine
  - ✓ Software as a service (Saas)
    - Microsoft Office 365, Gmail



Why Clouds?

- Pay as you go
  - ✓ No upfront cost
- On-demand self service
  - ✓ Convenience, no need to worry about maintenance
- Rapid elasticity
  - ✓ Virtually infinite resources
- Economy of scale
  - ✓ Cheap!

Distributed Information Systems

- Deal with interoperability between networked applications
  - ✓ Transaction processing system (TPS)
    - Distributed transaction: all or nothing happened
  - ✓ Enterprise application integration (EAI)

Transaction Processing Systems

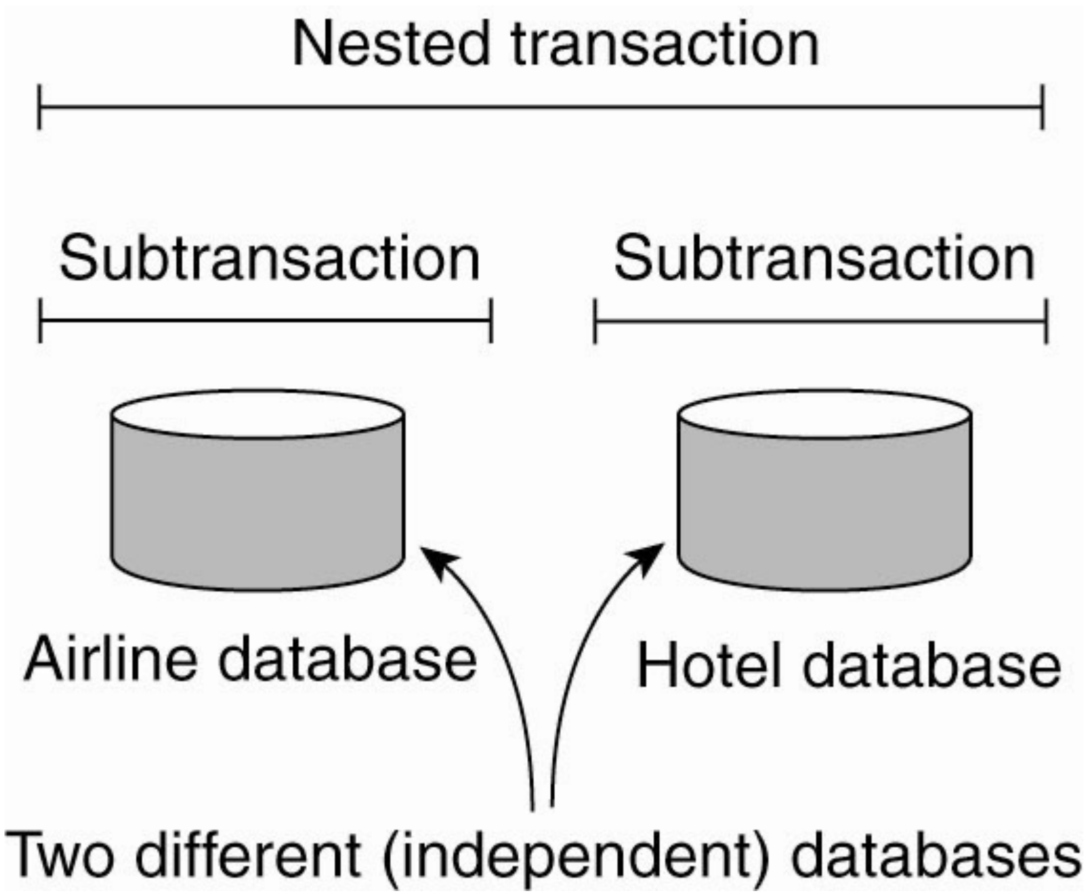
- Primitives for transactions.

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

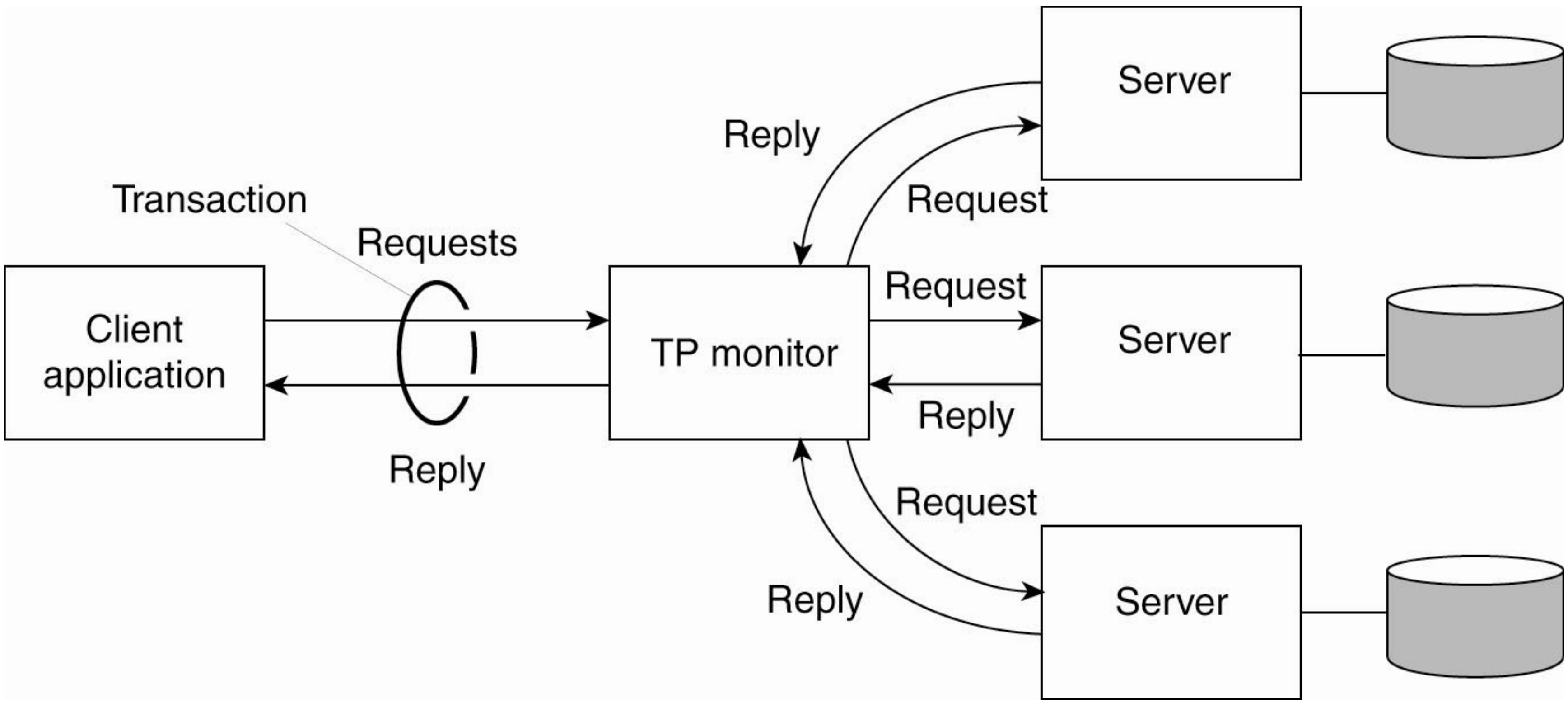
Properties of Transactions

- Atomic: to the outside world, the transaction happens indivisibly.
- Consistent: the transaction does not violate system invariants.
- Isolated: concurrent transactions do not interfere with each other.
- Durable: once a transaction commits, the changes are permanent.

Nested Transactions



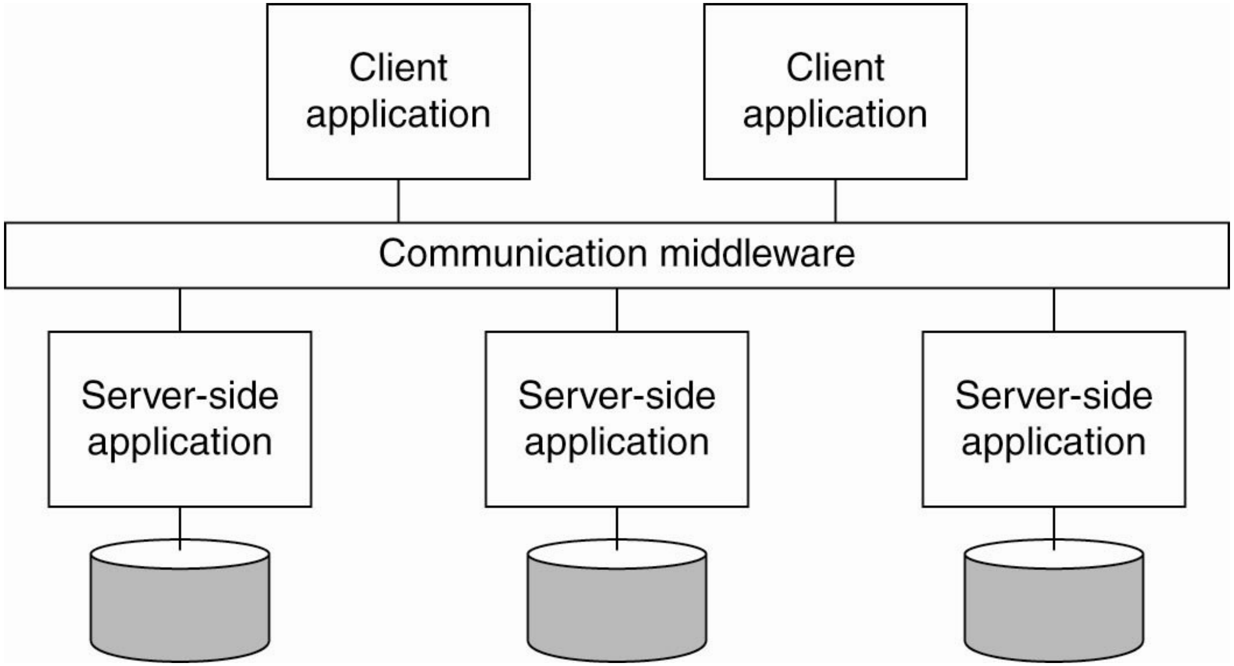
Transaction Processing Monitor



TP monitor offers a transactional programming model to allow an application to access multiple servers/databases

Enterprise Application Integration

- Goal: link applications in a single organization together to simplify or automate the business process
- Middleware as a communication facilitator (RPC, RMI)
  - ✓ Example: Apache ActiveMQ



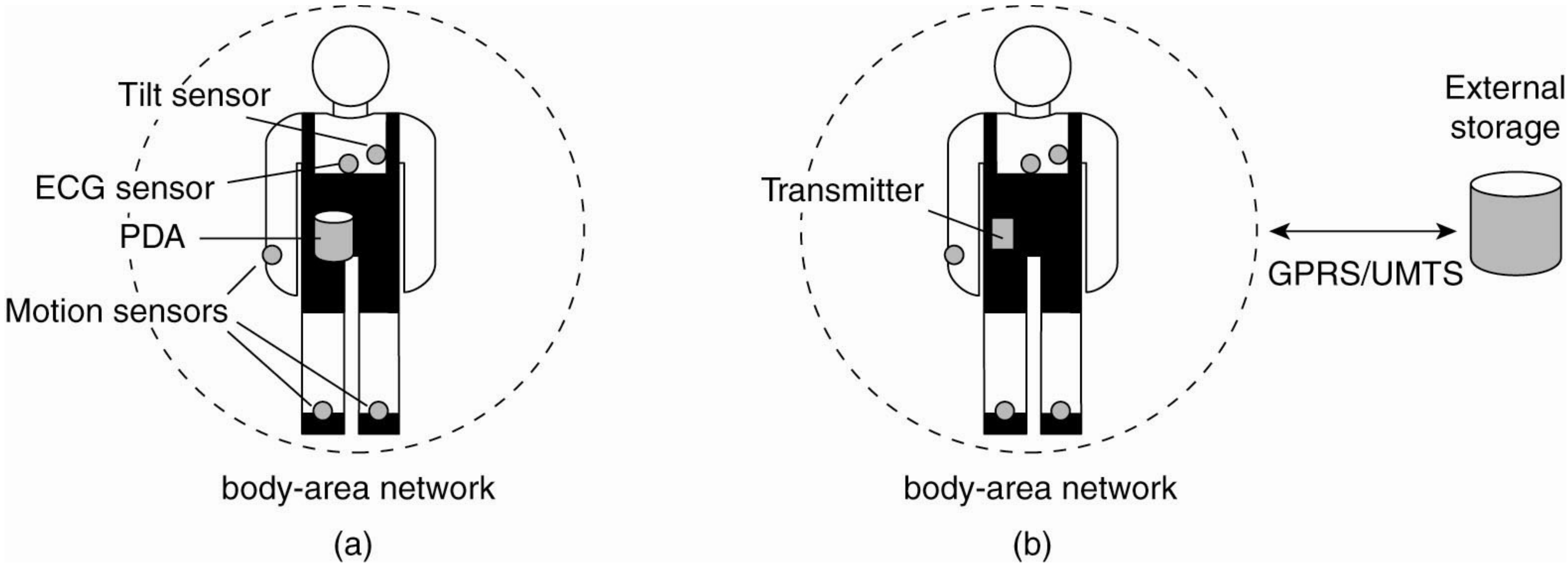
Distributed Pervasive Systems

- Devices in a distributed pervasive system are often
  - ✓ Small, battery-powered, and with limited wireless communication
- Requirements for pervasive systems
  - ✓ Embrace contextual changes
    - Environment changes all the time, e.g., switching wireless base station
  - ✓ Encourage ad hoc composition
    - Devices will be used differently by different users
  - ✓ Recognize sharing as the default
    - Easy to read, store, manage, and share information

Electronic Health Care Systems

- Questions to be addressed for health care systems:
  - ✓ Where and how should monitored data be stored?
  - ✓ How can we prevent loss of crucial data?
  - ✓ What infrastructure is needed to generate and propagate alerts?
  - ✓ How can physicians provide online feedback?
  - ✓ How can extreme robustness of the monitoring system be realized?
  - ✓ What are the security issues and how can the proper policies be enforced?

Electronic Healthcare Systems



Monitoring a person in a pervasive electronic health care system, using (a) a local hub or (b) a continuous wireless connection.

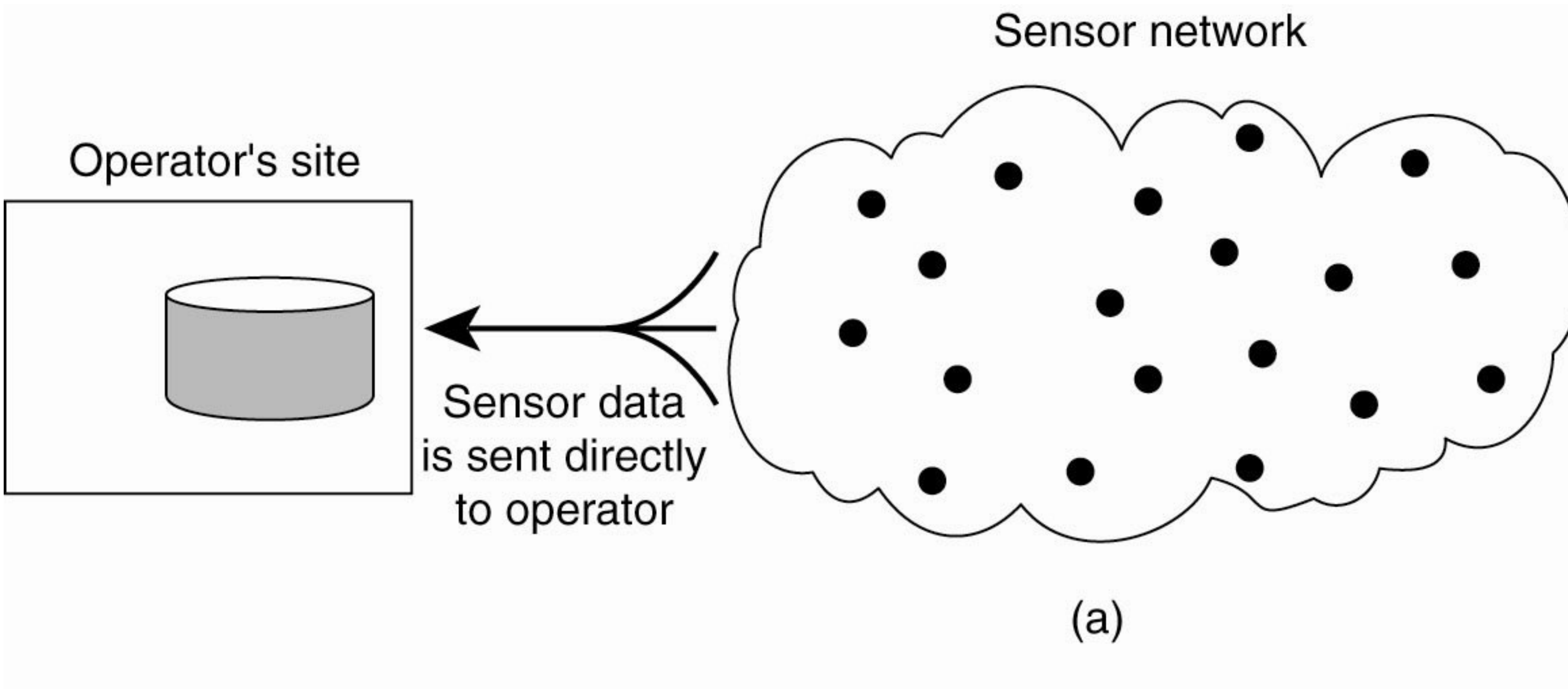
Wireless Sensor Network (WSN)

- A network that consists of a large number of low-end sensor nodes, each can sense the environment and talk to other sensors
- Applications
  - ✓ Military surveillance
  - ✓ Environment monitoring
  - ✓ Smart home/cities
  - ✓ Vehicular network

Key Design Questions of WSN

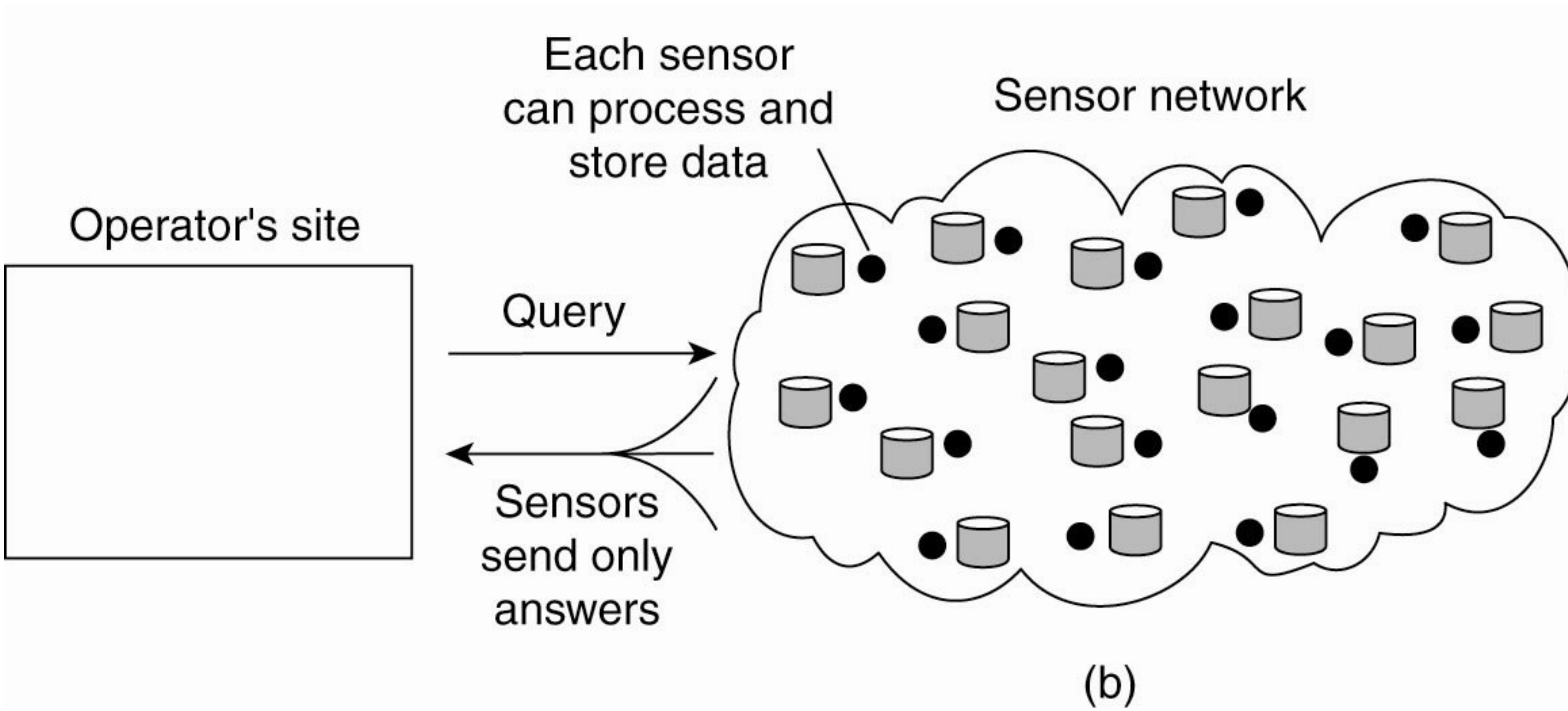
- How do we (dynamically) set up an efficient tree in a sensor network?
- How does aggregation of results take place? Can it be controlled?
- What happens when network links fail?

Wireless Sensor Network – cont'd



Organizing a sensor network database, while storing and processing data (a) only at the operator's site

Wireless Sensor Network – cont'd



Organizing a sensor network database, while storing and processing data (b) only at the sensors.