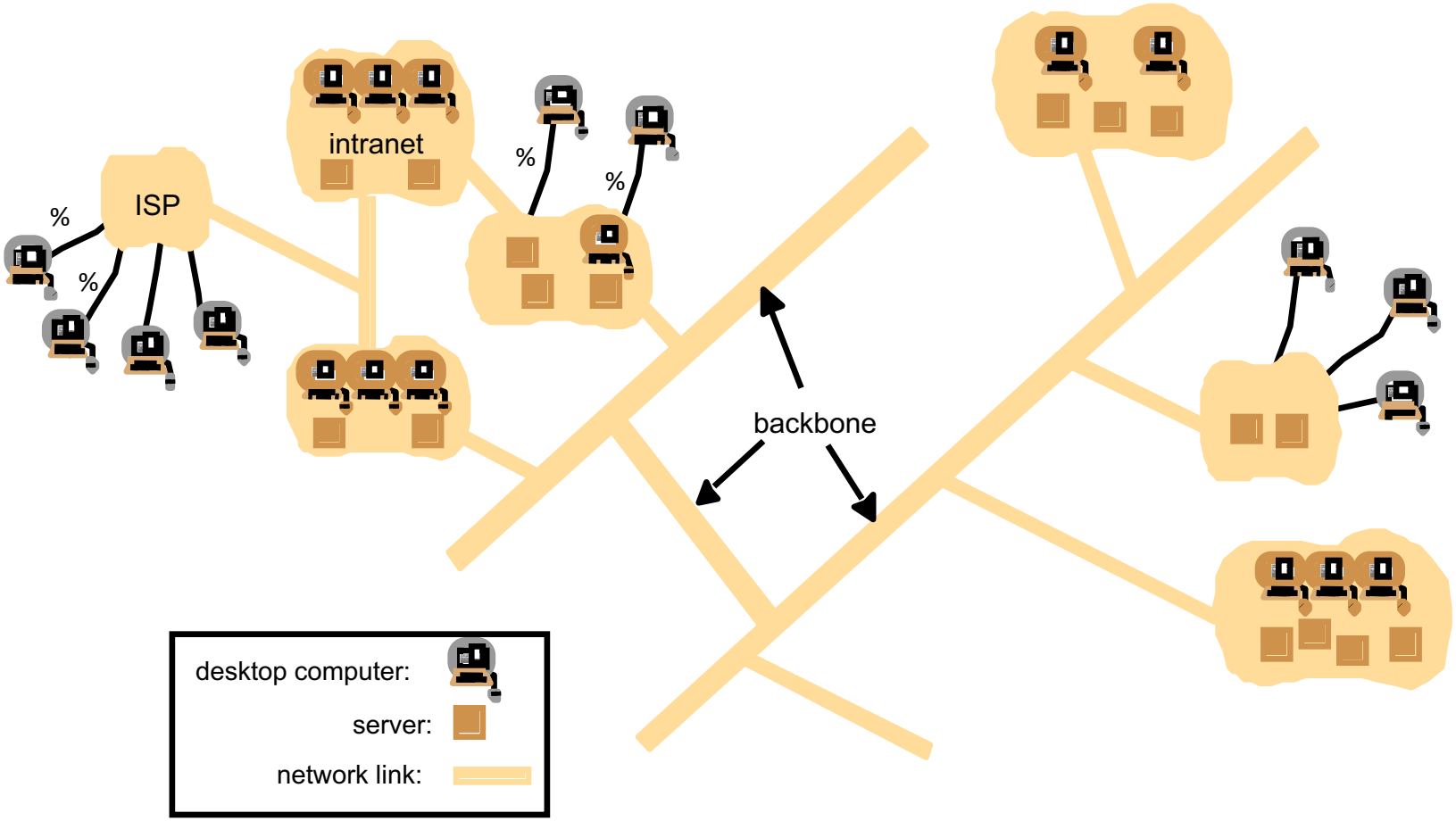
Introduction to Distributed Systems

Yao Liu

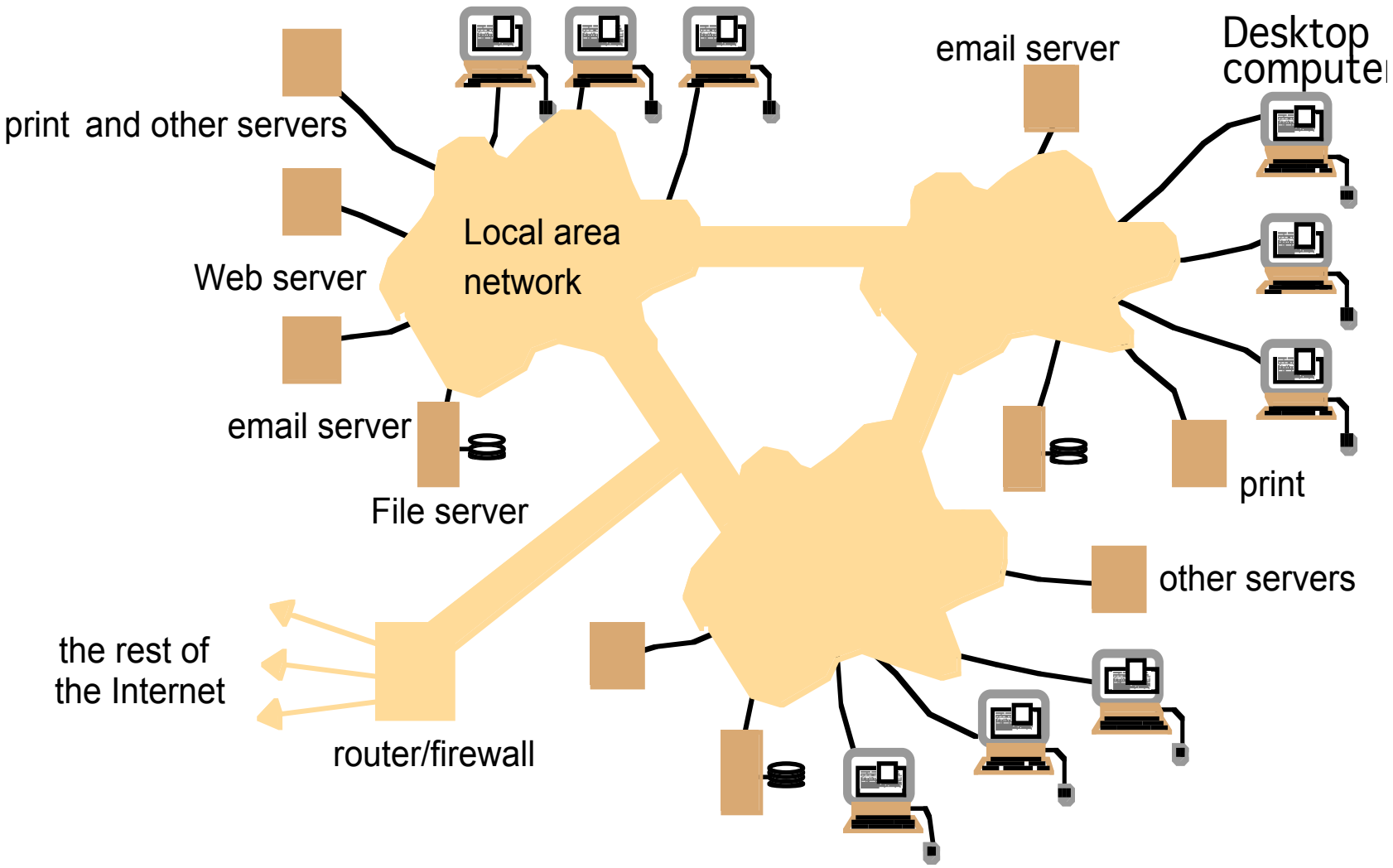
Example distributed systems

- Internet
- Intranets/workgroups
- ATM (bank) machines
- Computer clusters
- Computational cloud
- Pervasive distributed systems
  - Wireless/Cellular networks
  - Mobile ad hoc networks (MANETs)
  - Vehicular networks (VANETs)
  - Wireless sensor networks
- “Internet of Things”

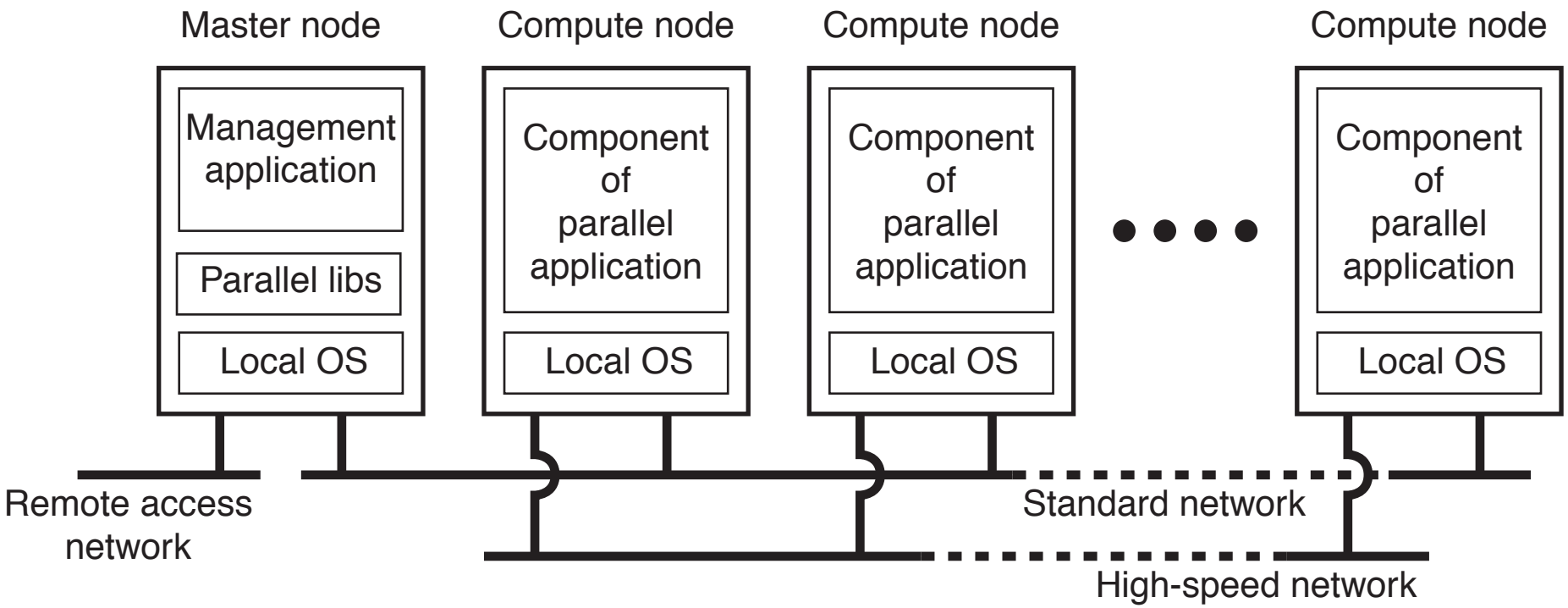
A typical portion of the Internet



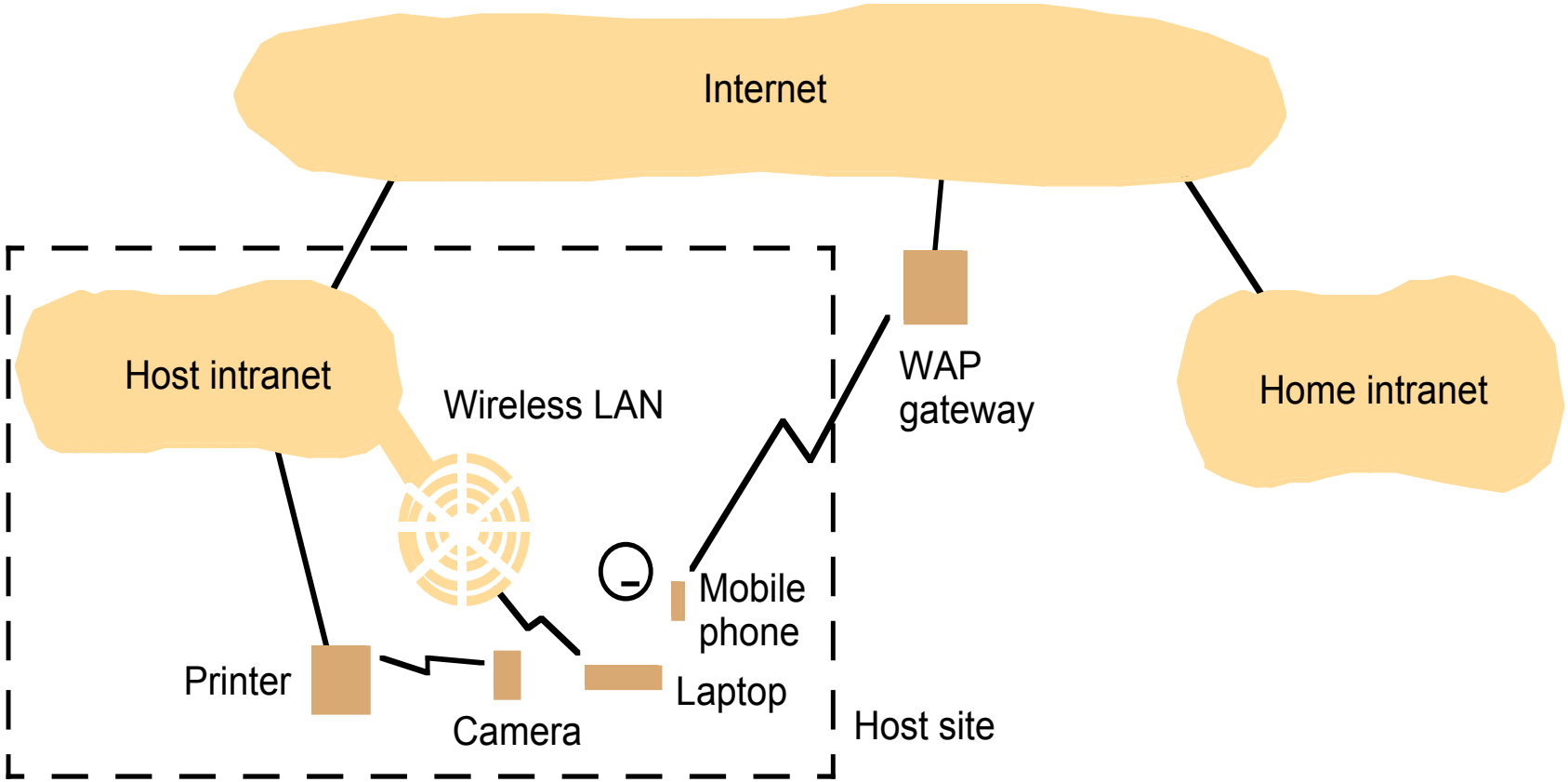
A typical intranet



Cluster computing systems



Portable and handheld devices in a distributed system



Distributed applications

- Applications that consist of a set of processes that are distributed across a network of machines and work together as an ensemble to solve a common problem
- In the past, mostly “client-server”
  - Resource management centralized at the server
- Peer-to-Peer computing represents a movement towards more “truly” distributed applications

Definition

- Tannenbaum: “A distributed system is a collection of independent computers that appears to its users as a single coherent system”
  - Sort of a “classical” definition of distributed systems

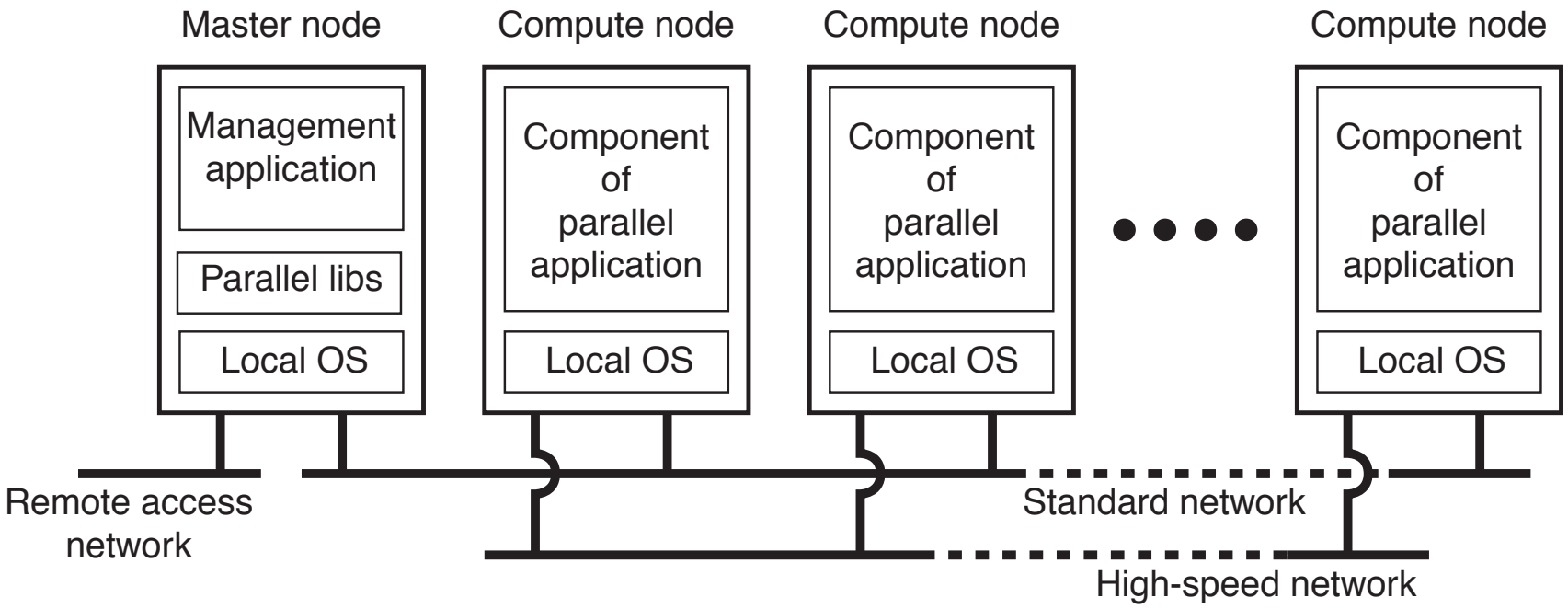
- Lamport: “You know you have one when the crash of a computer you’ve never heard of stops you from getting any work done”

Types of distributed systems

- Distributed computing systems
  - Cluster computing: homogeneous
  - Grid computing: heterogeneous
  - Cloud computing: everything as a service
- Distributed information systems
  - Transaction processing systems
  - Enterprise application integration
  - Publish/subscribe systems
- Distributed pervasive systems
  - Mobile computing systems
  - Sensor networks

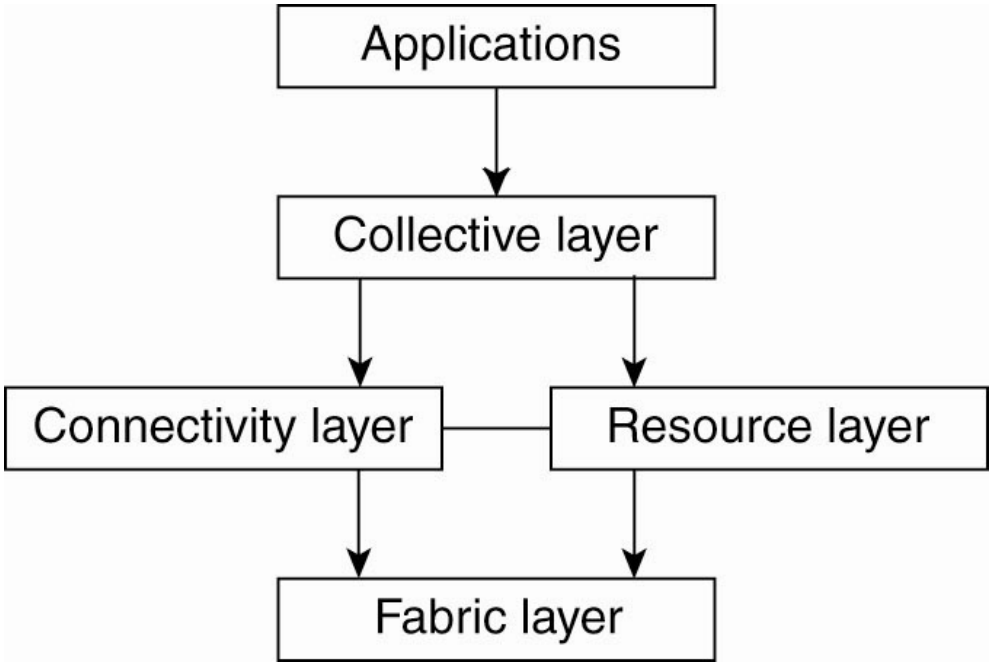
Cluster computing systems

- Essentially a group of high-end systems connected through a LAN:
  - Homogeneous: same OS, near-identical hardware
  - Single managing node



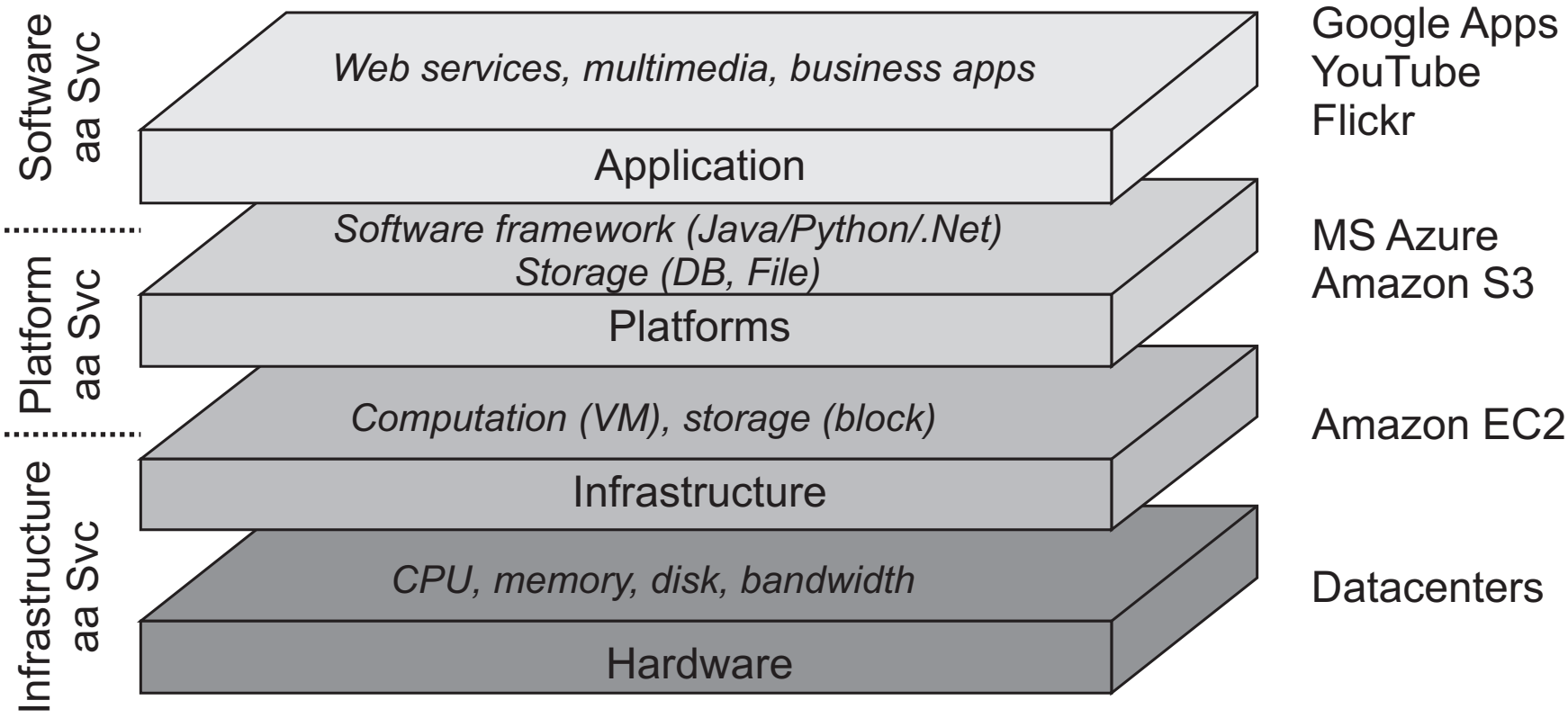
Grid computing systems

- Heterogeneous nodes
- Dispersed across several organizations
- Can easily span a wide-area network



Cloud computing systems

- Everything as a service:



Transaction processing systems

- Example primitives for transactions:

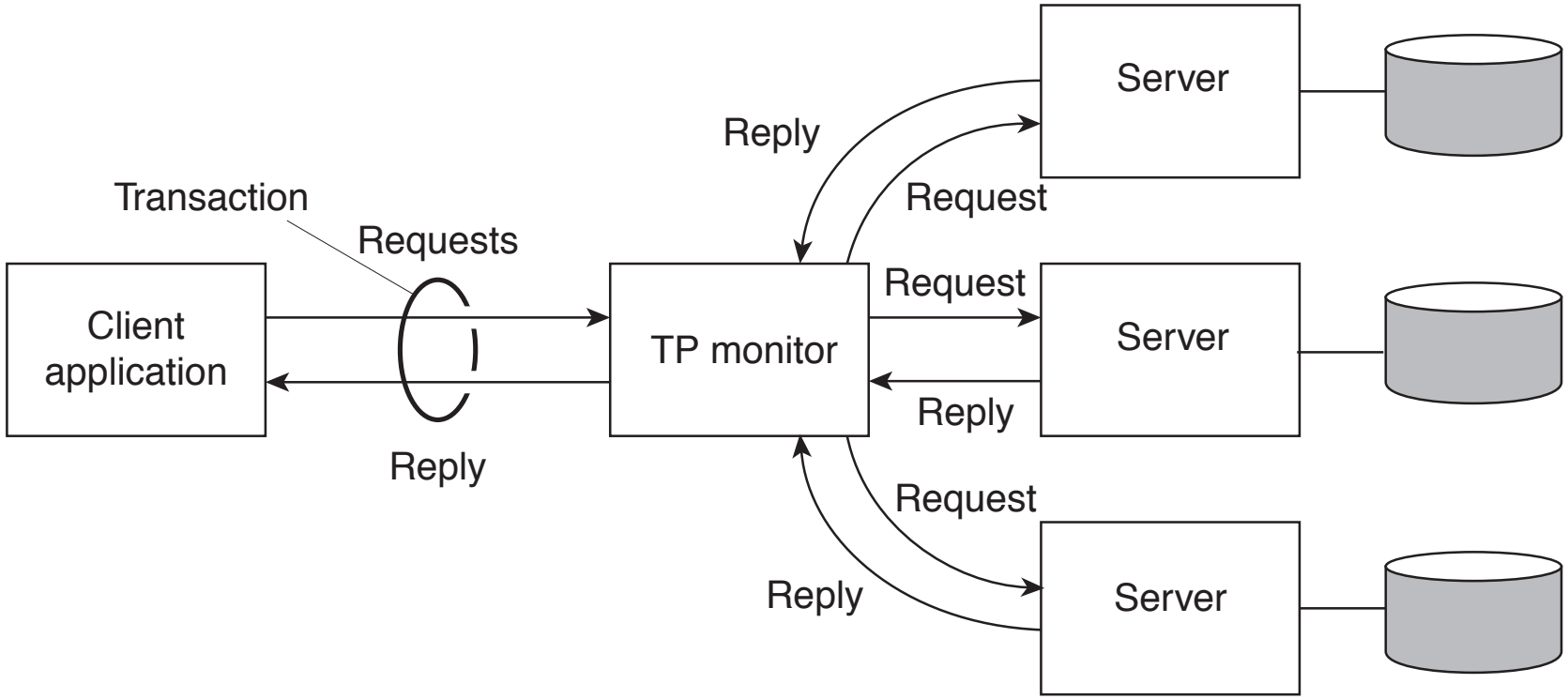
Primitive	Description
BEGIN_TRANSACTION	Mark the start of a transaction
END_TRANSACTION	Terminate the transaction and try to commit
ABORT_TRANSACTION	Kill the transaction and restore the old values
READ	Read data from a file, a table, or otherwise
WRITE	Write data to a file, a table, or otherwise

Transaction processing systems (2)

- A transaction is a collection of operations on the state of an object (database, object composition, etc.) that satisfies the following properties (ACID):
- **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.

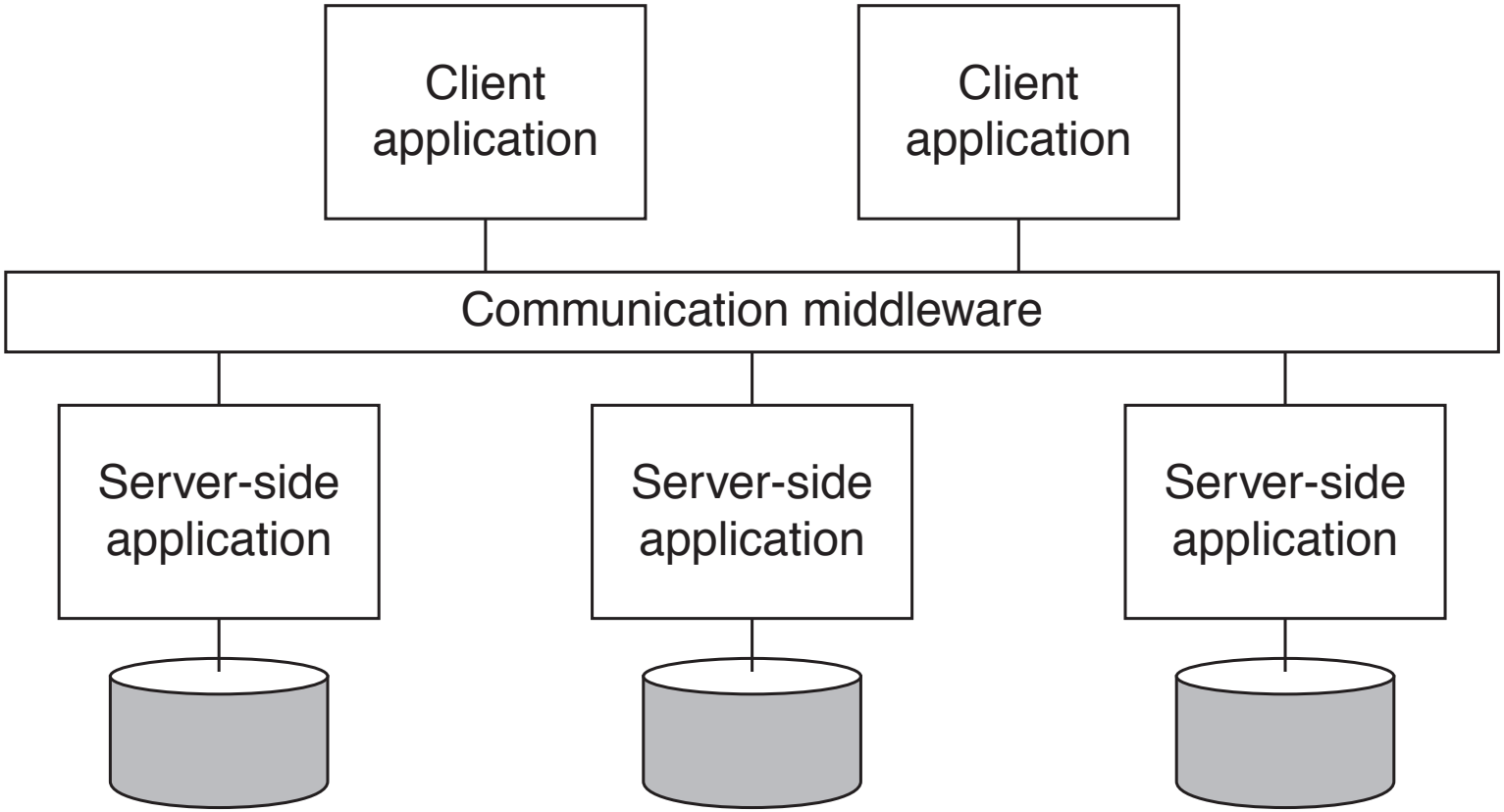
Transaction processing monitor

- A TP Monitor is responsible for coordinating the execution of a transaction



Enterprise application integration

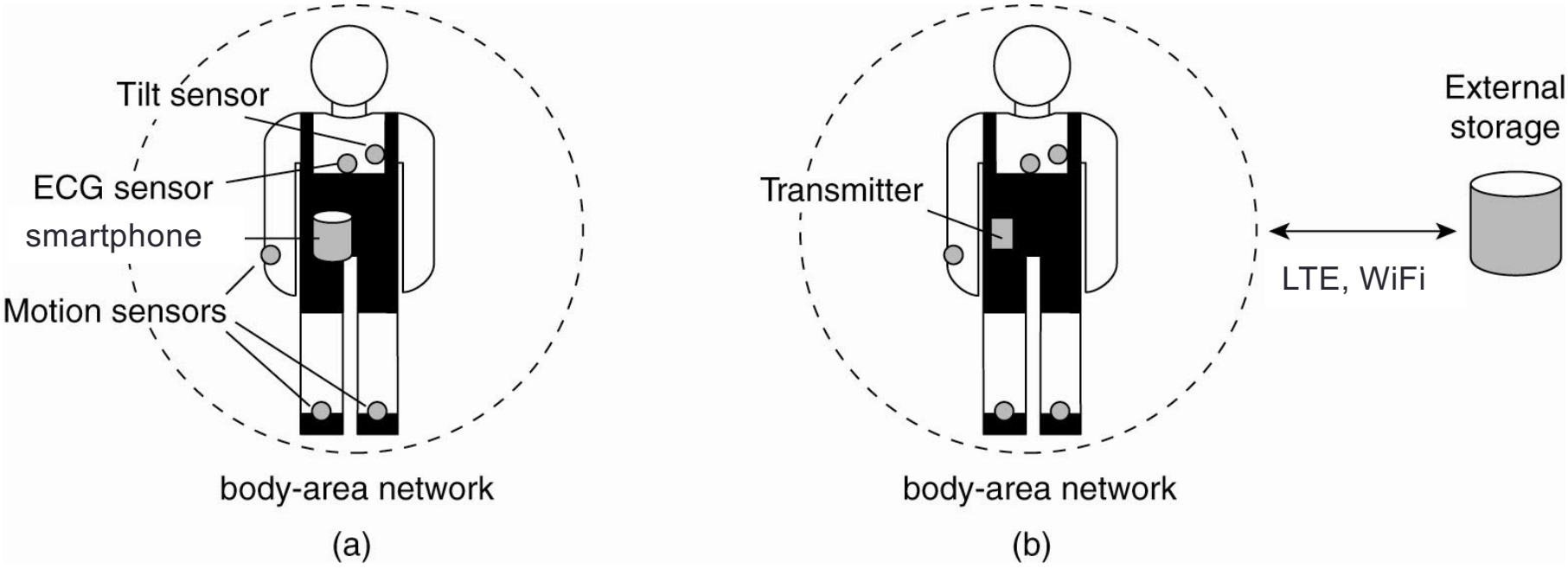
- Middleware as a communication facilitator in enterprise application integration.



Mobile computing systems

- Many different types of mobile devices: smartphones, tablets, remote controls, and so on
- Battery powered
- Wireless communication
- Devices may continuously change their location:
  - setting up a route may be problematic, as routes can change frequently
  - devices may easily be temporarily disconnected

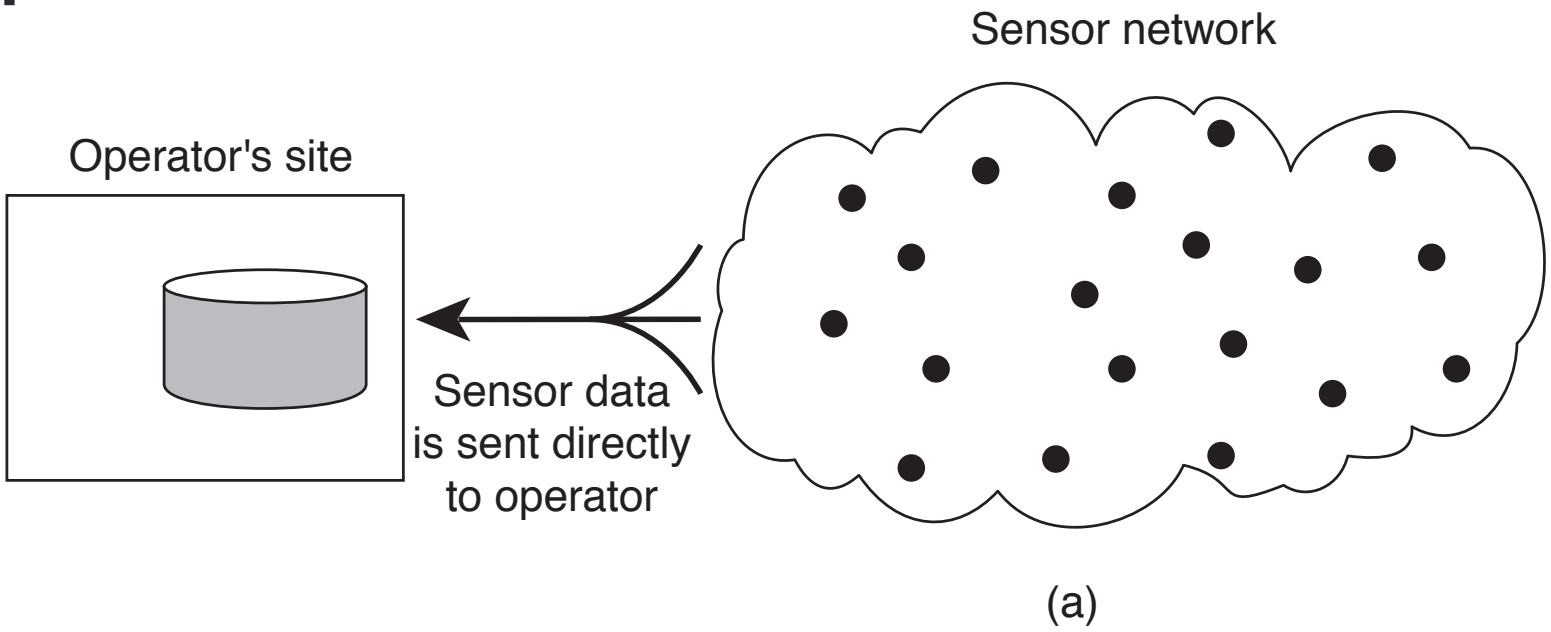
Electronic health care systems



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

Sensor networks (1)

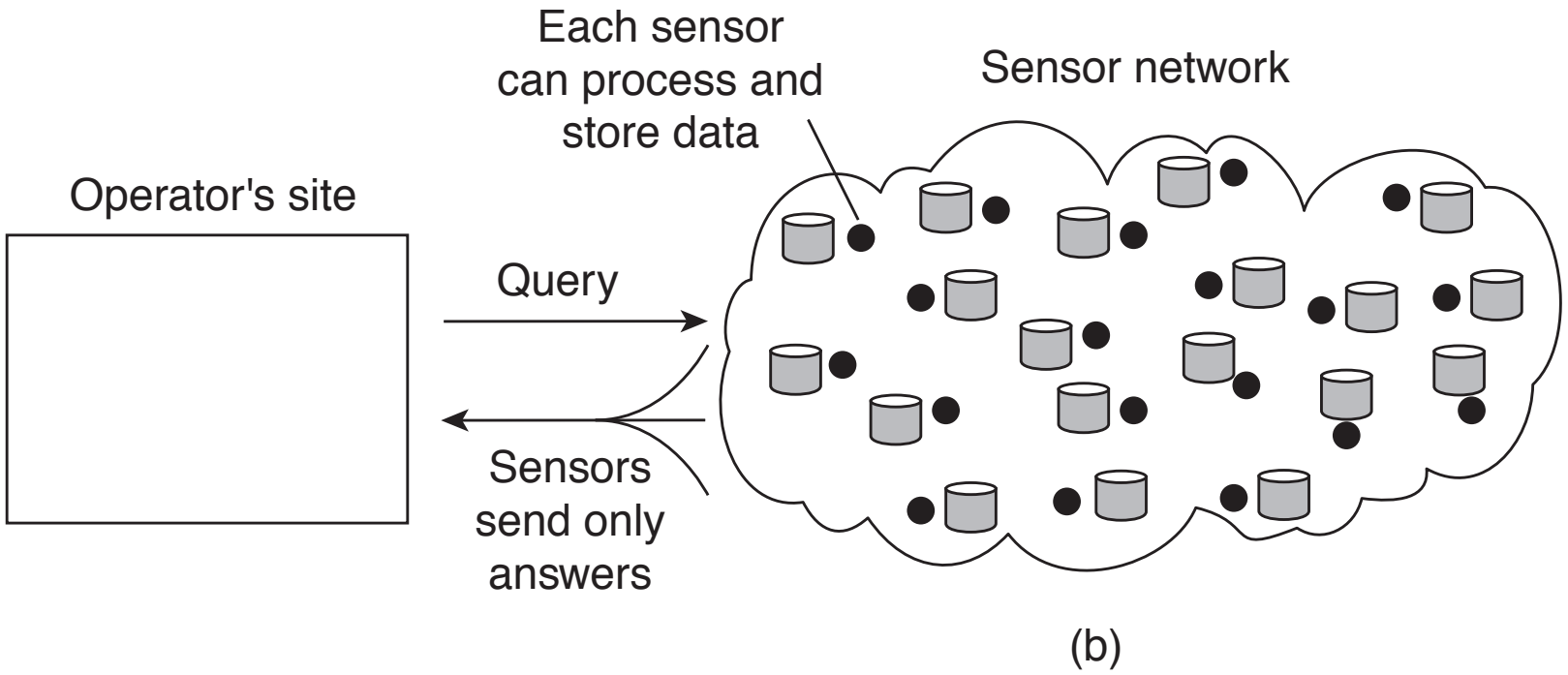
- Organizing a sensor network database, while storing and processing data **(a) only at the operator's site.**



(a)

Sensor networks (2)

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



Goals/Benefits of distributed systems

- Resource sharing
- Distribution transparency
- Scalability
- Fault tolerance and availability
- Performance

Challenges

- Heterogeneity: (need for openness)
  - hardware, platforms, languages
  - interoperability and portability

  key interfaces in software and communication protocols need to be standardized, e.g., Interface Definition Language (IDL)
- Security:
  - Denial of service attacks
  - Mobile code
- Scalability
- Transparency
- Failure handling
- Quality of service

Scalability

- Size scalability
  - Number of users and/or processes
  - Concerning centralized services/data/algorithm
- Geographical scalability
  - Maximum distance between nodes
  - synchronous communication in LAN vs. asynchronous communication in WAN
- Administrative scalability
  - Number of administrative domains
  - Policy conflicts from different orgs (e.g., for security, access control)

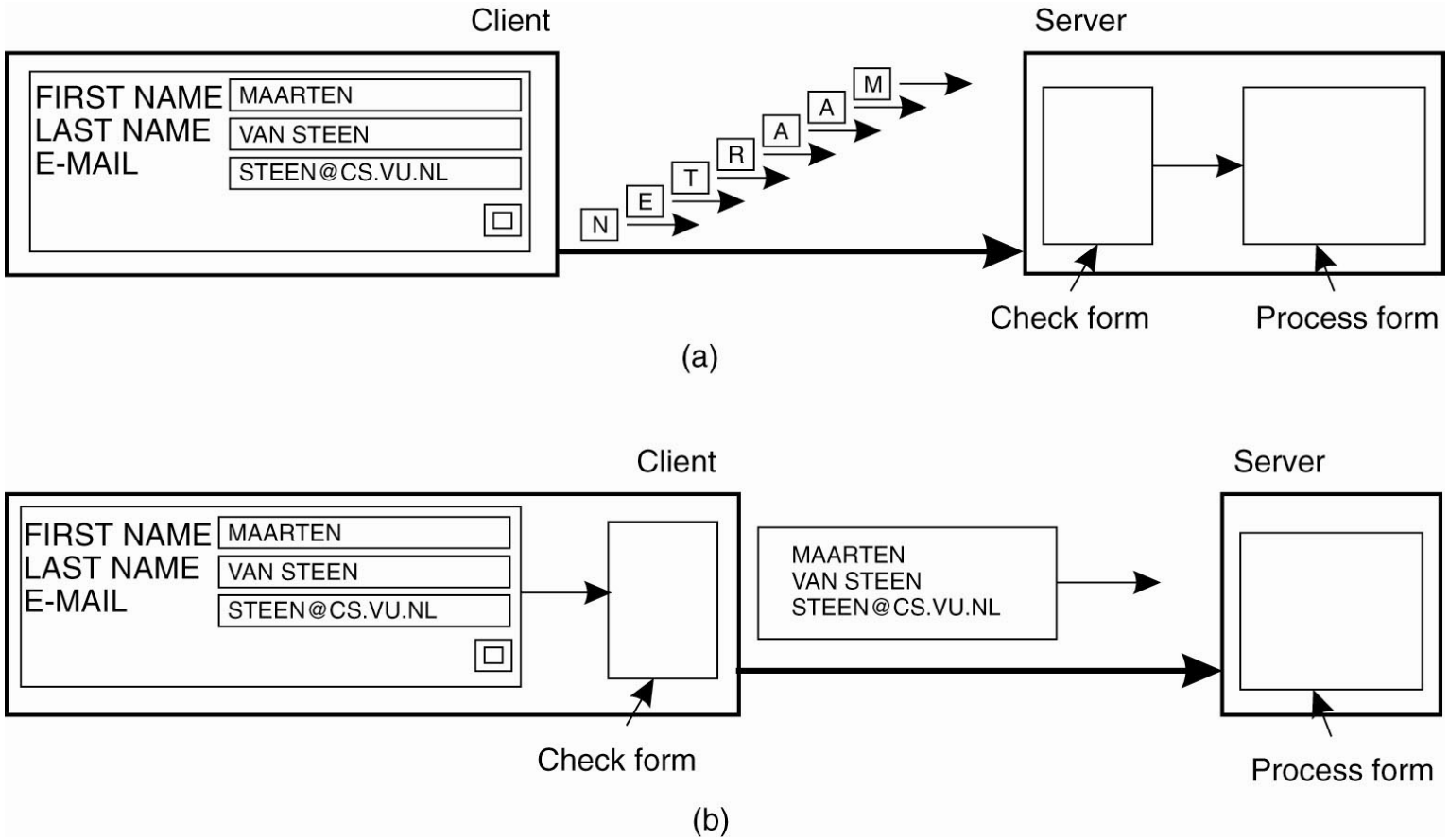
Scalability

- Key to scalability: decentralized algorithms and data structures
  - No machine has complete information about the state of the system
  - Machines make decisions based on locally available information
  - Failure of one machine does not ruin the algorithm
  - There is no implicit assumption that a global clock exists

Scalability techniques

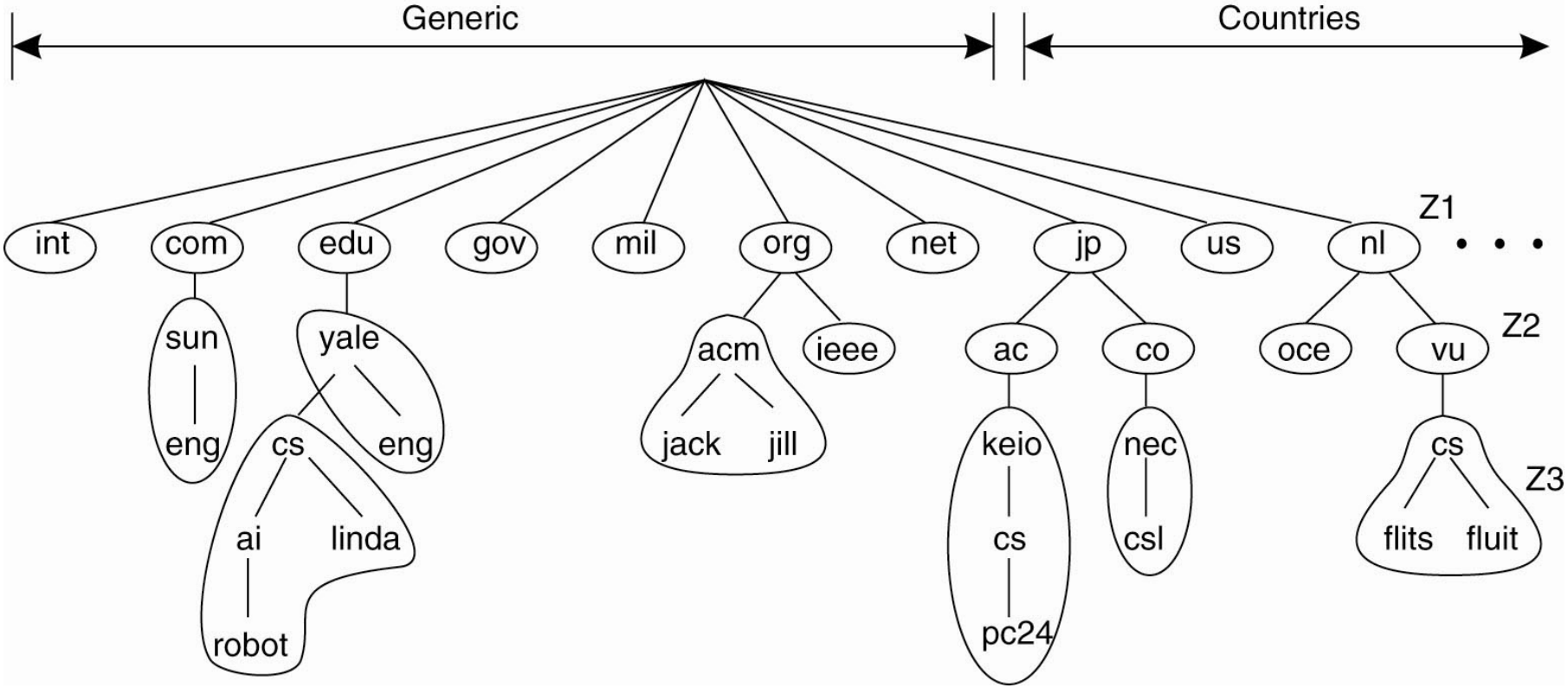
- Hiding communication latency
  - Asynchronous communication
  - Code migration (to client)
- Distribution
  - Splitting a large component to parts (e.g., DNS)
- Replication
  - Caching (decision of clients vs. of the server)
  - On demand (pull) vs. planned (push)

Scaling techniques (1)



The difference between letting (a) a server or (b) a client check forms as they are being filled

Scaling techniques (2)



An example of diving the DNS namespace into zones

Find the IP address of **robot.ai.cs.yale.edu**

Transparency

Transparency	Description
Access	Hide differences in data representation and how an object is accessed
Location	Hide where an object is located
Relocation	Hide that an object may be moved to another location while in use
Migration	Hide that an object may move to another location
Replication	Hide that an object is replicated
Concurrency	Hide that an object may be shared by several independent users
Failure	Hide the failure and recovery of an object

Achieving transparency is a very difficult problem.

Reading

- Chapter 1 of TBook
- Chapters 1 and 2 of CBook
- "A Note on Distributed Computing"
  - J. Waldo, G. Wyant, A. Wollrath, S. Kendall
- "Above the Clouds: A Berkeley View of Cloud Computing"
  - M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, M. Zaharia