DISTRIBUTED SYSTEMS
Principles and Paradigms
Second Edition
ANDREW S. TANENBAUM
MAARTEN VAN STEEN

Bearbeitung von Matthias Wallnöfer

Introduction and Architectures

Definition of a Distributed System (1)

A distributed system is:

A collection of **independent computers** that appears to its users as a **single coherent system**.

Definition of a Distributed System (2)

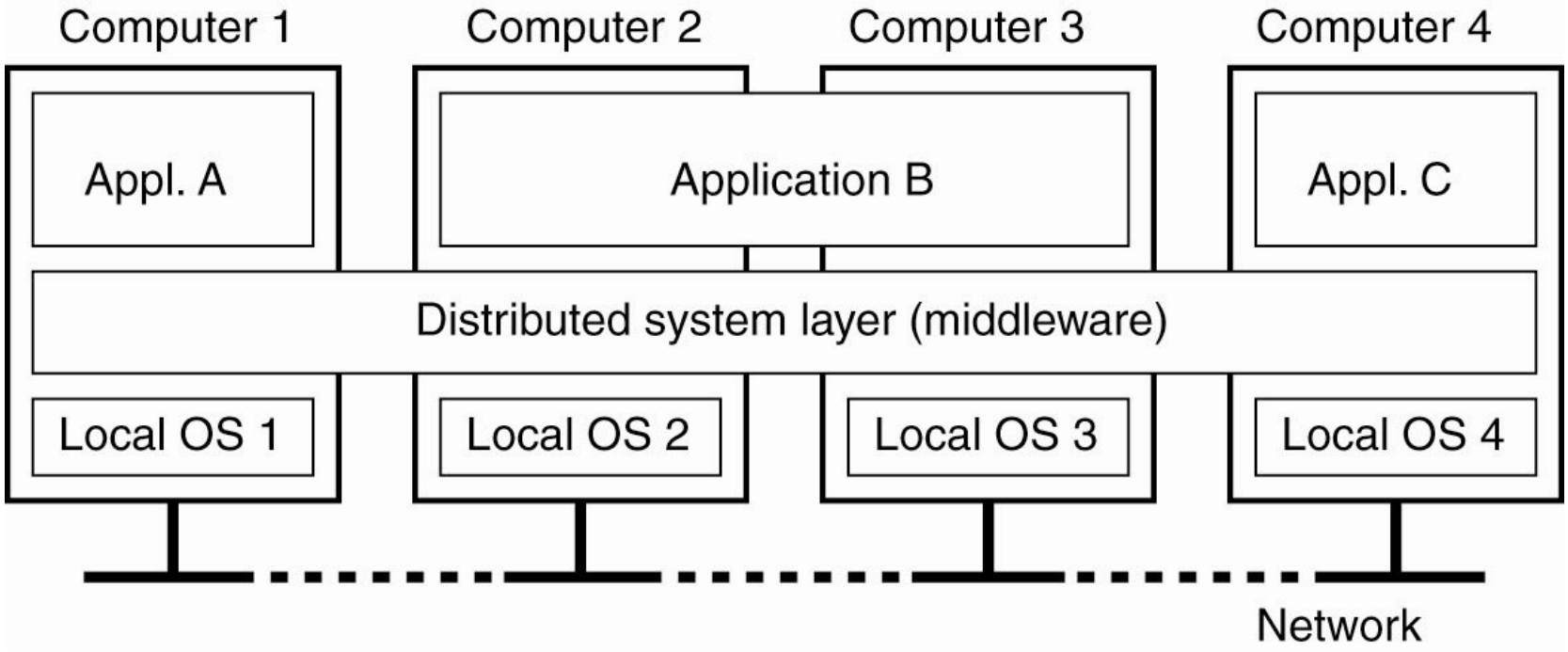

Figure 1-1. A distributed system organized as middleware. The middleware layer extends over multiple machines, and offers each application the same interface.

Cluster Computing Systems

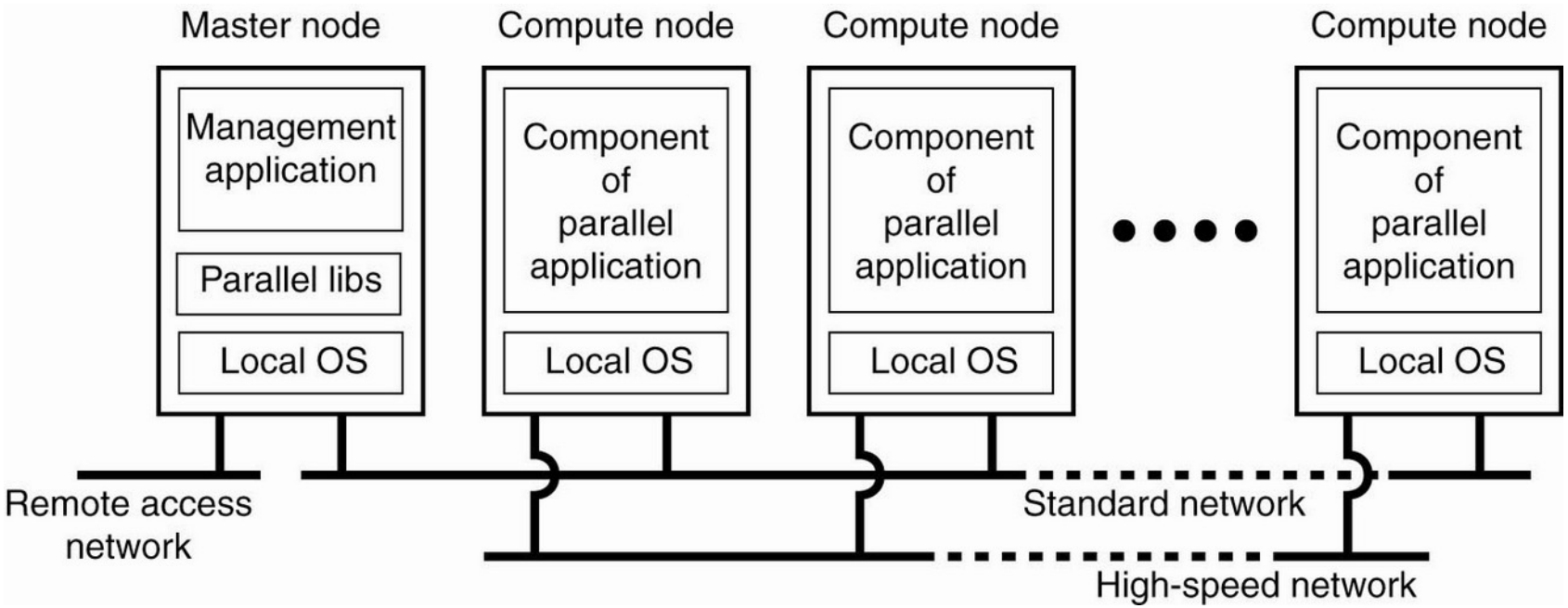

Figure 1-6. An example of a cluster (grid) computing system. Nodes are *homogeneous*: same OS, near-identical hardware, Single managing (= master) node

Transparency in a Distributed System

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

Figure 1-2. Different forms of transparency in a distributed system (ISO, 1995).

Scalability Problems

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

Figure 1-3. Examples of scalability limitations.

Scalability Problems

Characteristics of *decentralized* algorithms:

- No machine **has complete information** about the system state.
- Machines make decisions **based only on local information**.
- Failure of one or some machine(s) **does not ruin** the algorithm (*Byzantine fault tolerance*).
- There is no implicit assumption that **a global clock exists**.

Scaling Techniques (1)

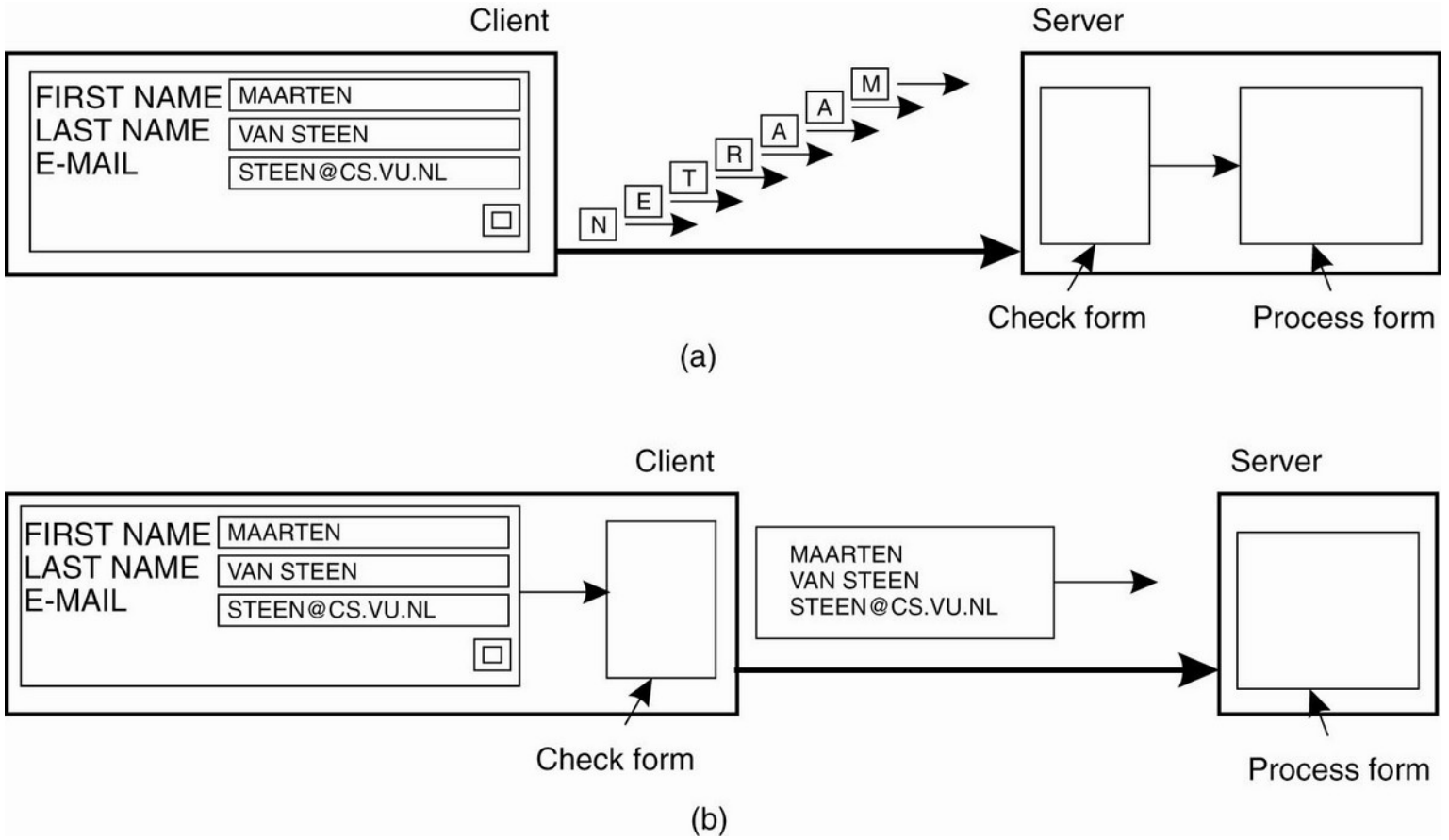

Figure 1-4. The difference between letting (a) a server or (b) a client check forms as they are being filled.

Scaling Techniques (2)

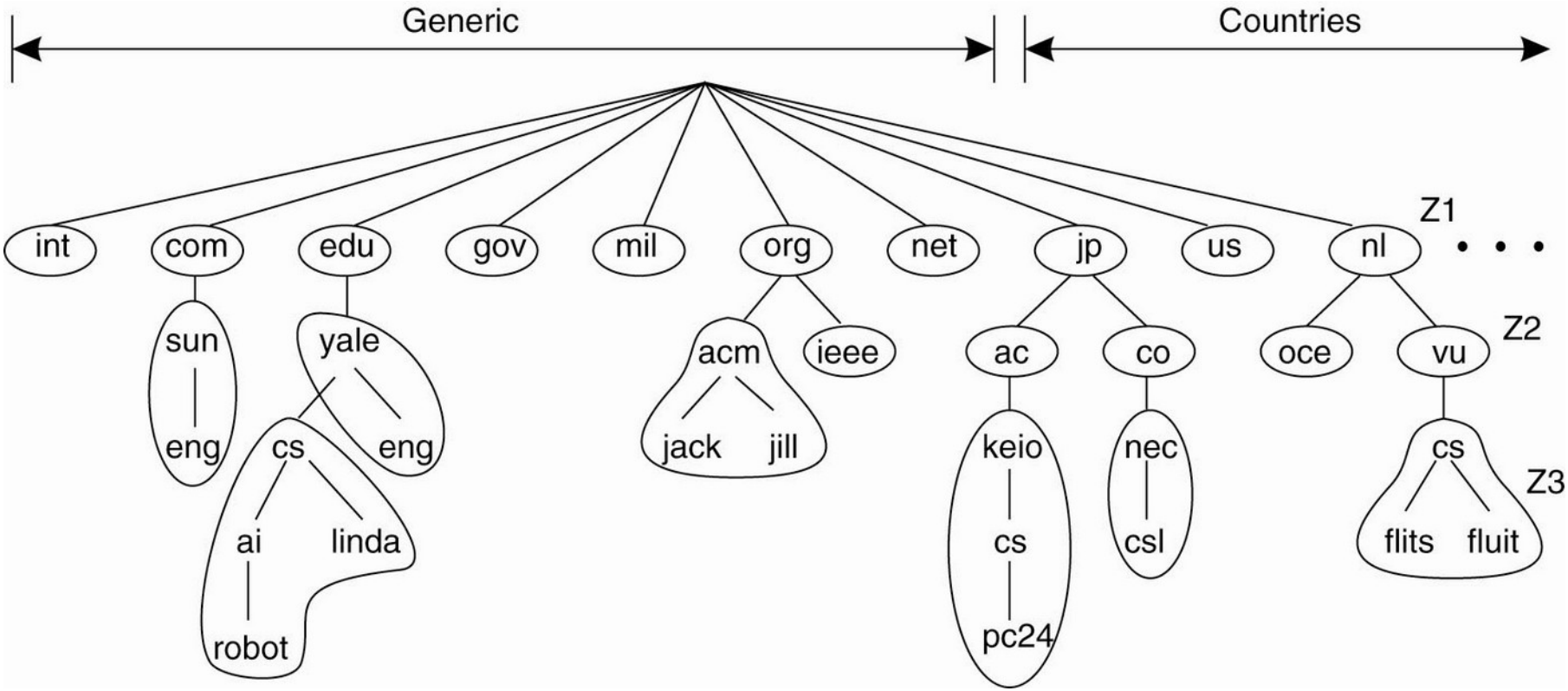

Figure 1-5. An example of dividing the DNS name space into zones.

What if a *single DNS server* would handle everything?

Pitfalls when Developing Distributed Systems

False assumptions made by first time developer:

- The **network** is **reliable**.
- The **network** is **secure**.
- The **network** is **homogeneous**.
- The **topology** does **not change**.
- **Latency** is **zero**.
- **Bandwidth** is **infinite**.
- **Transport cost** is **zero**.
- There is **one administrator**.

Transaction Processing Systems (1)

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

Figure 1-8. Example primitives for transactions (usually on databases).

Transaction Processing Systems (2)

Characteristic 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.

Transaction Processing Systems (3)

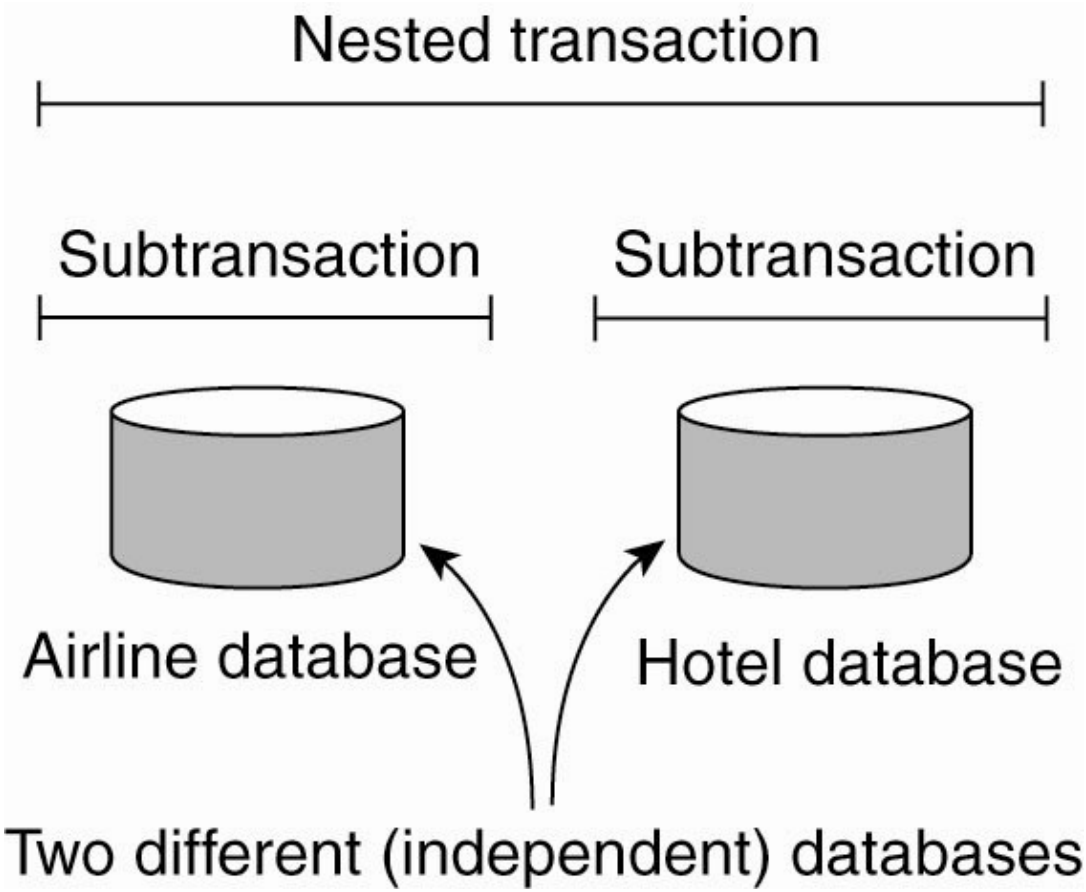

Figure 1-9. A nested (distributed) transaction.

Transaction Processing Systems (4)

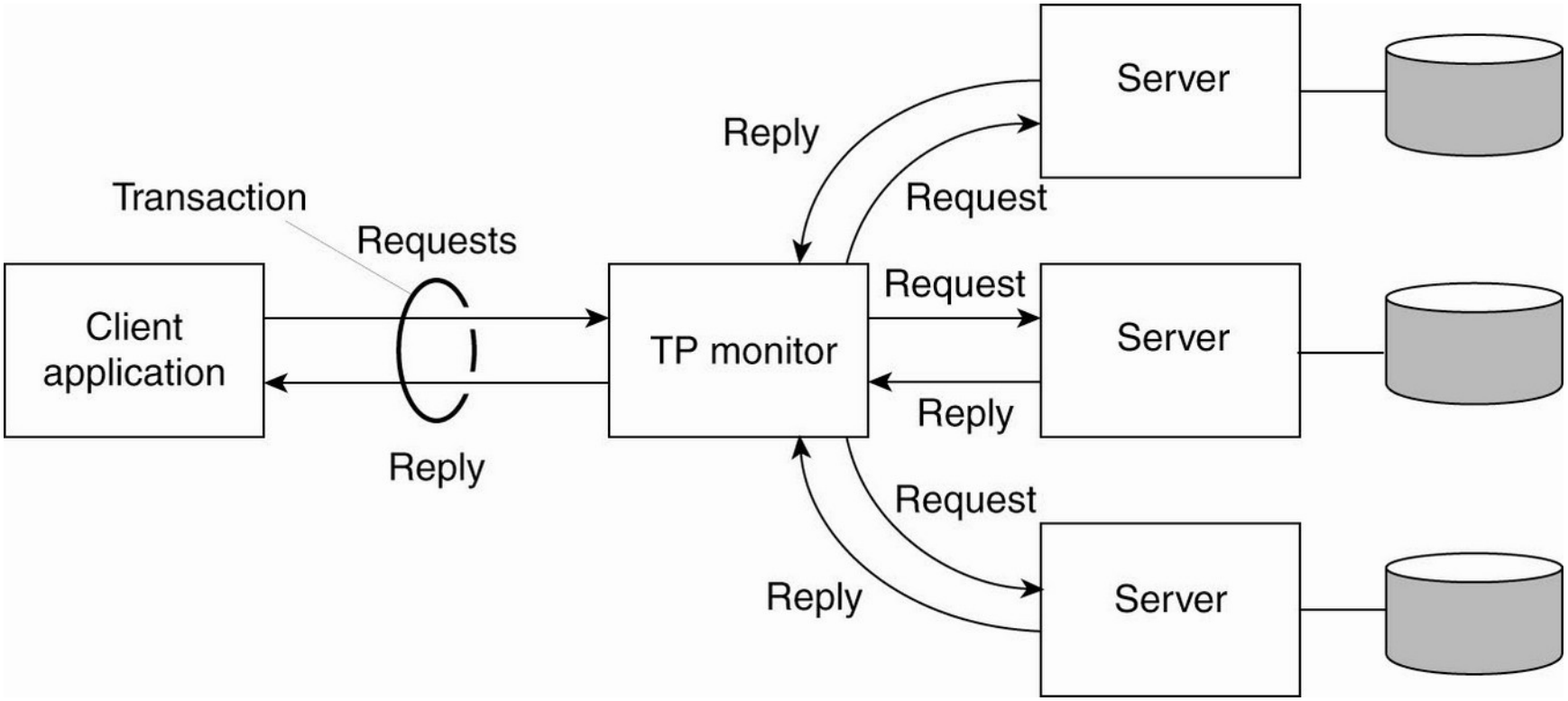

Figure 1-10. The role of a TP monitor in distributed systems.

Distributed Pervasive Systems

Requirements for pervasive (=ad hoc) systems

- Embrace **contextual changes**.
- Encourage **ad hoc composition**.
- Recognize **sharing as the default**.

Electronic Health Care Systems (1)

Questions to be addressed for health care systems:

- Where and how should monitored **data be stored**? How can we prevent loss of data?
- What **infrastructure** is needed to generate and propagate alerts?
- How can physicians (= doctors) **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 Health Care Systems (2)

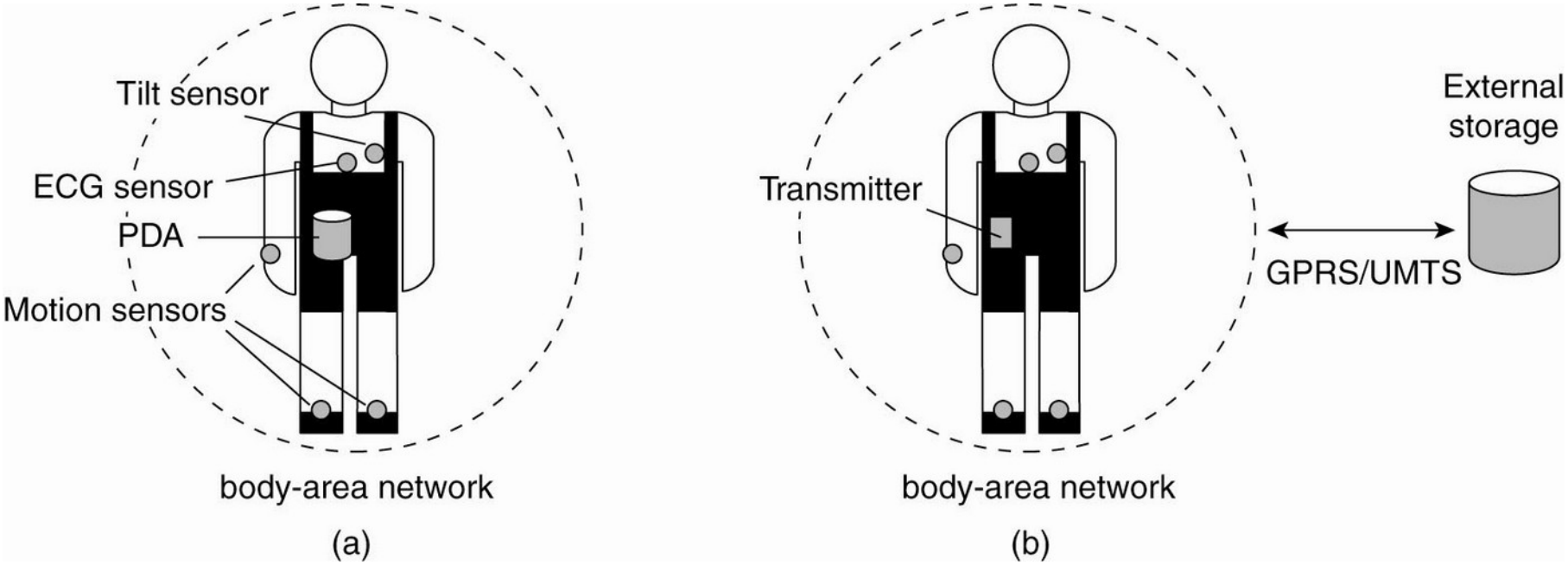

Figure 1-12. Monitoring a person in a pervasive electronic health care system, using (a) a local hub or (b) a continuous wireless connection.

Nowadays usually by means of smartphone/tablets and Bluetooth LE.

Architectural Styles (1)

Important styles of architecture for distributed systems

- **Layered** architectures
- **Object-based** architectures
- **Event-based** architectures
- **Data-centered** architectures

Architectural Styles (2)

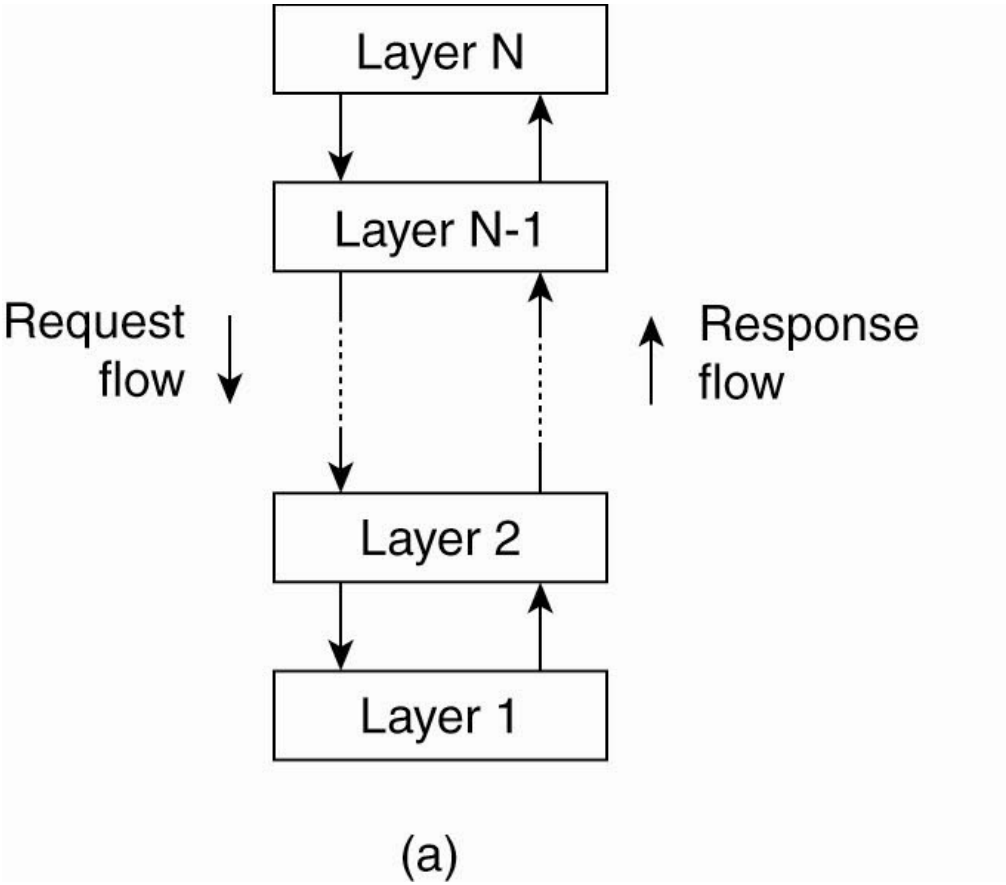

Figure 2-1. The (a) **layered** architectural style (*RPC*) and …

Architectural Styles (3)

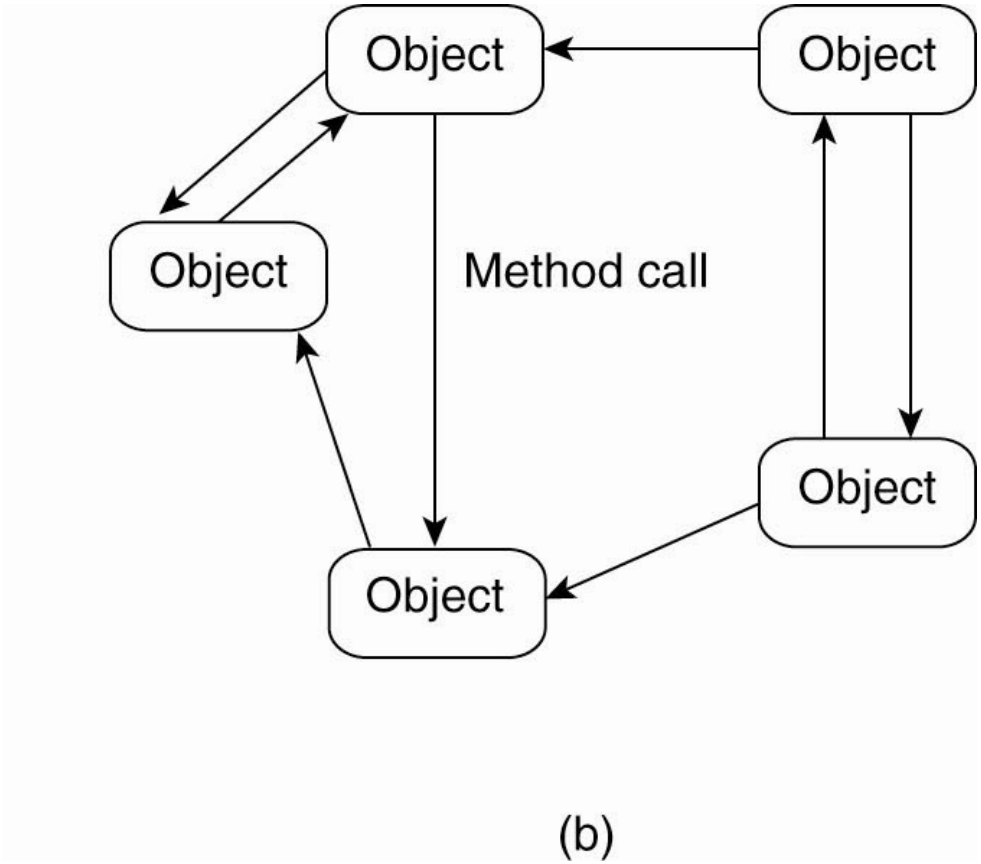

Figure 2-1. (b) The **object-based** architectural style (*RMI*).

Architectural Styles (4)

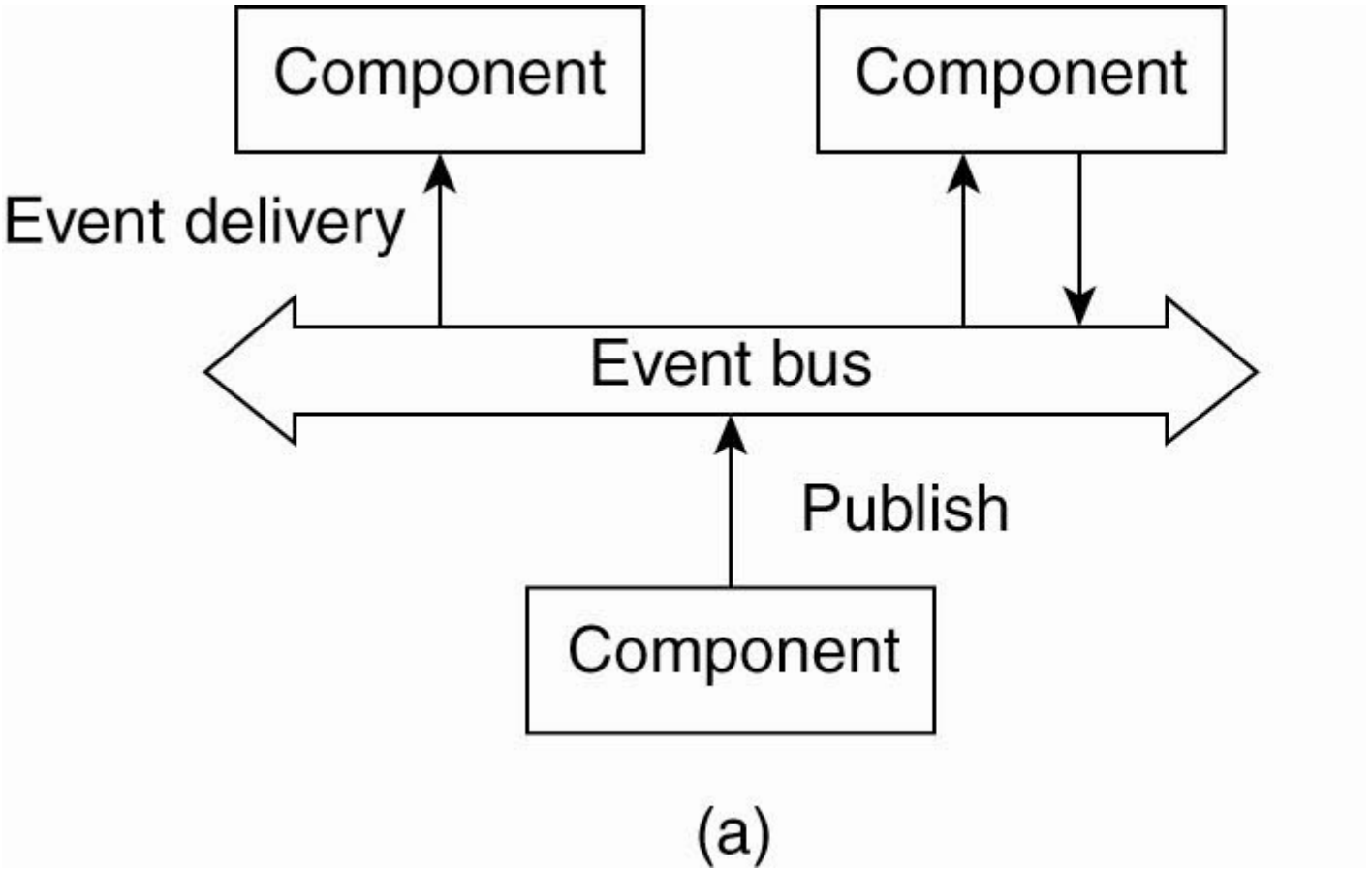

Figure 2-2. (a) The **event-based** architectural style (*MoM, ESB*) and …

Architectural Styles (5)

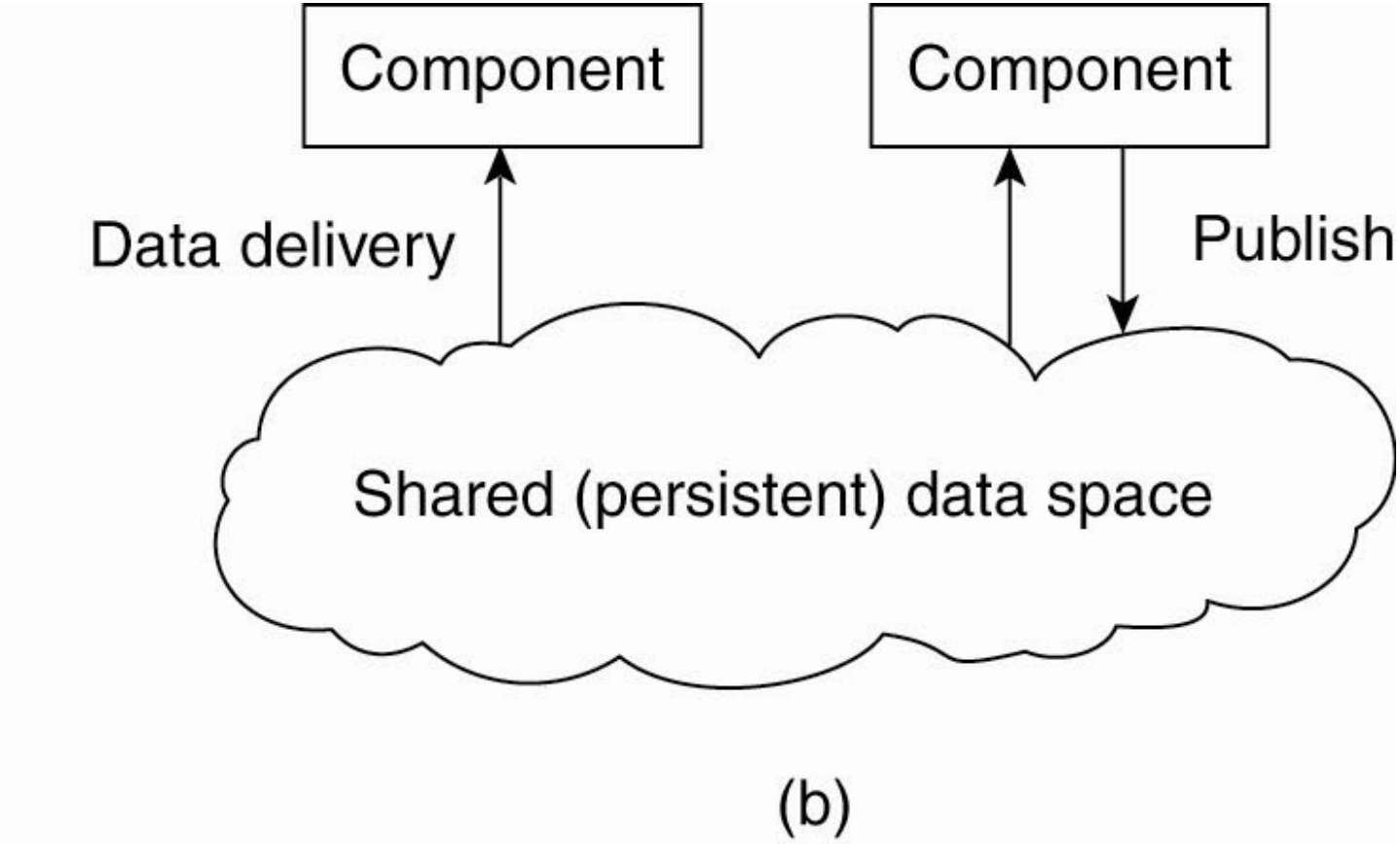

Figure 2-2. (b) The **shared data-space** architectural style (TupleSpace, InMemory-Store, *DB*).

Centralized Architectures

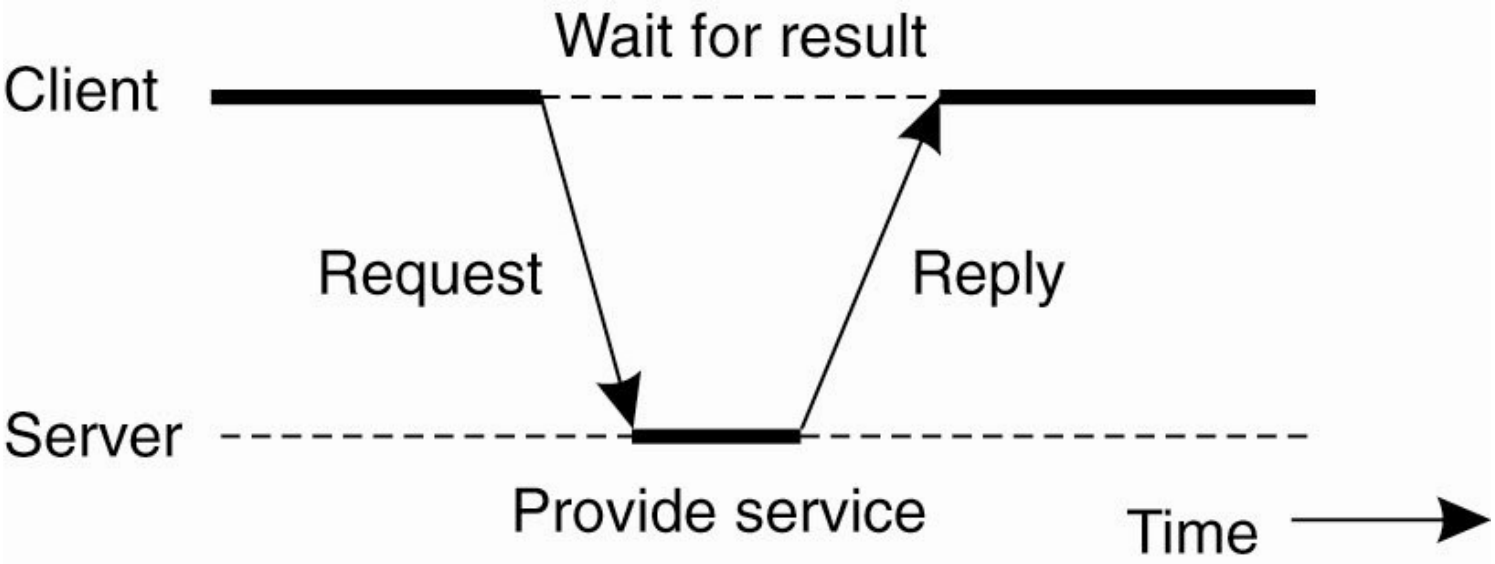

Figure 2-3. General interaction between a client and a server.

Structured Peer-to-Peer Architectures (1)

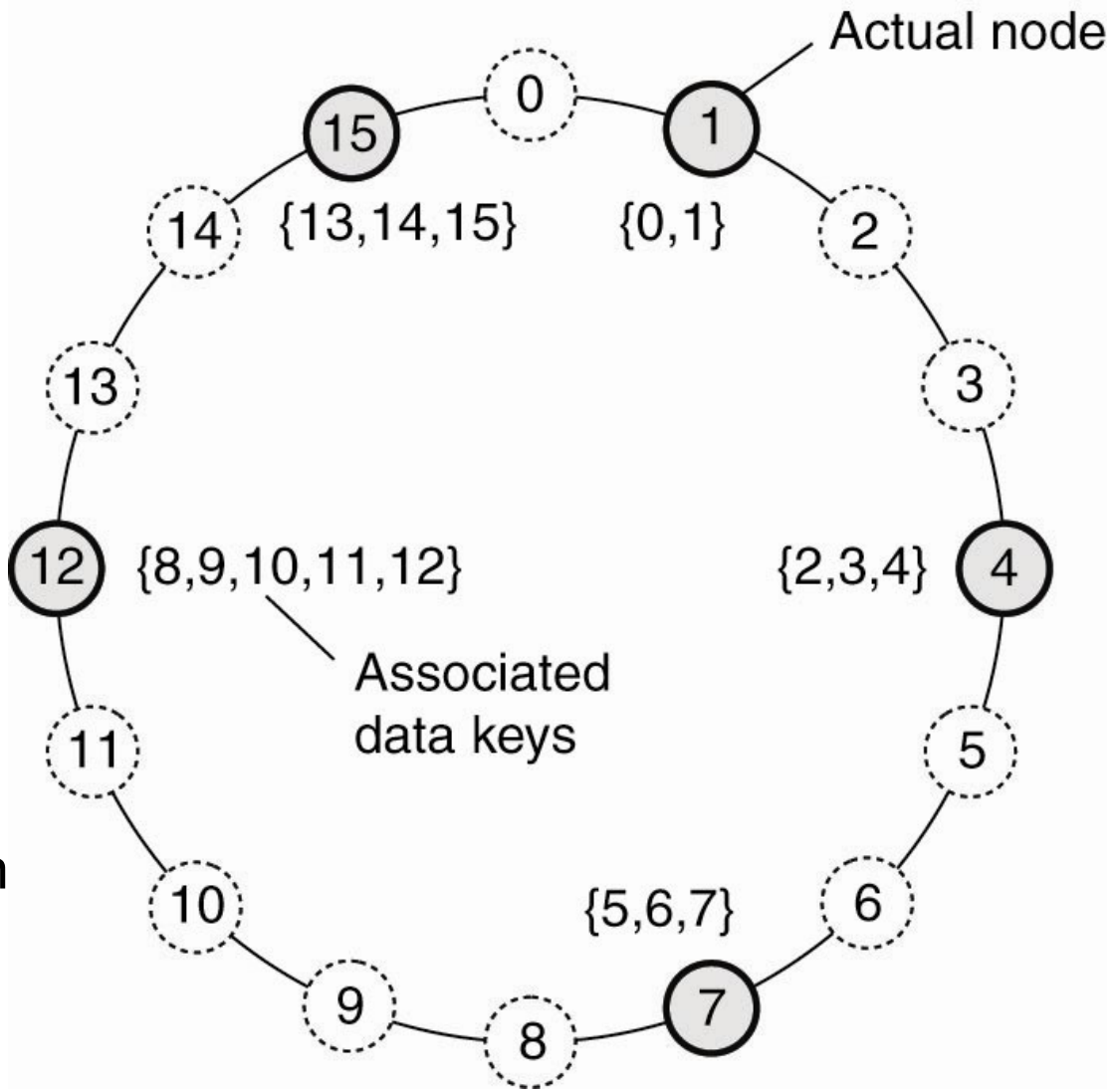

Figure 2-7. The mapping of data items onto nodes in Chord. → a **distributed hash table (DHT)**

Structured Peer-to-Peer Architectures (2)

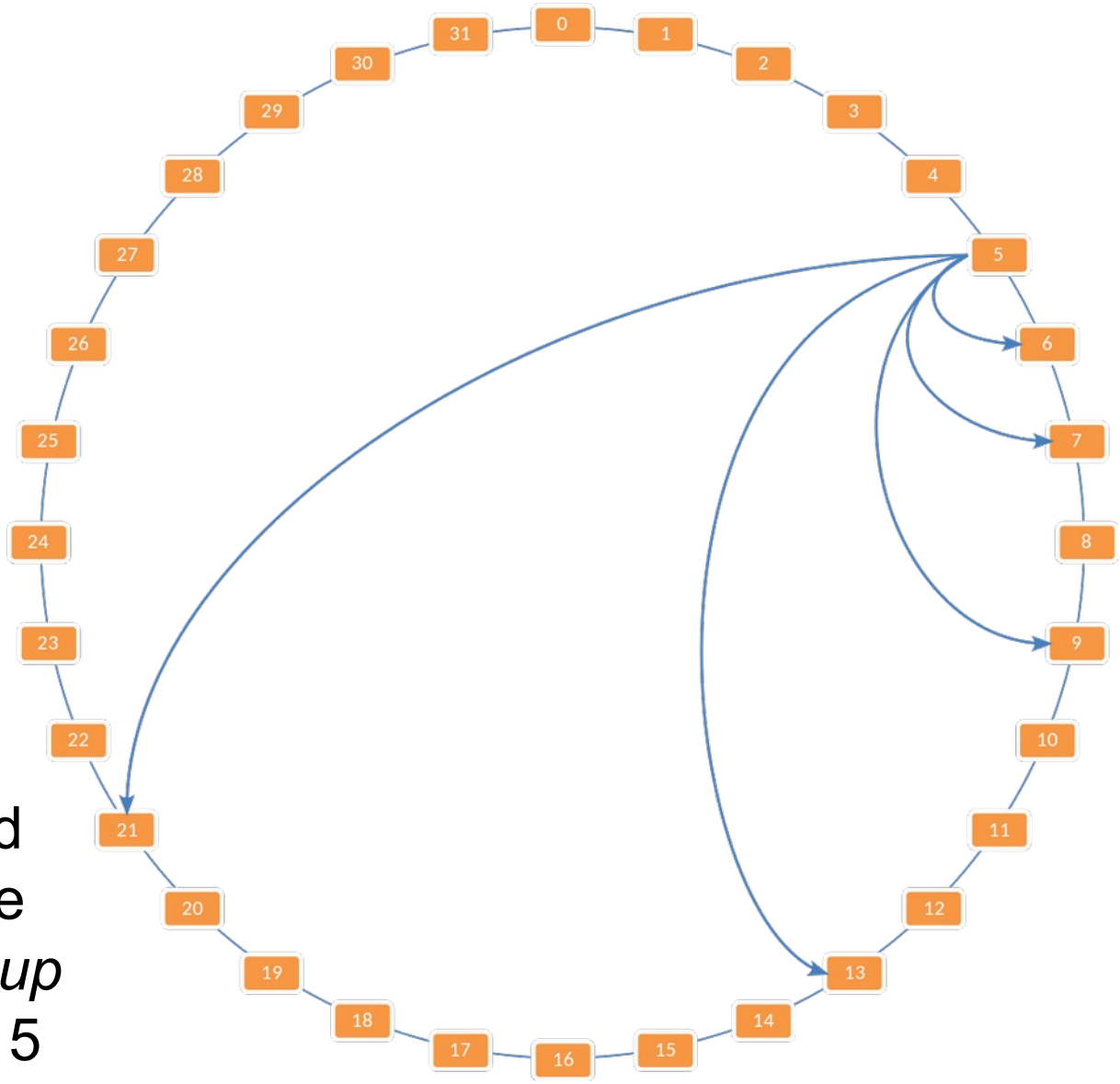

Figure 2-8. A Chord system with the *finger table lookup entries* of node 5

Unstructured Peer-to-Peer Architectures

Observation: Many unstructured P2P systems attempt to maintain a **random graph**:

Basic principle: Each node is required to be able to contact a randomly selected other node:

- Let each peer maintain a **partial view** of the network, consisting of c other nodes
- Each node P periodically selects a node Q from its partial view
- P and Q exchange information **and** exchange members from their respective partial views

Observation: It turns out that, depending on the exchange, randomness, but also **robustness** of the network can be maintained.

Topology Management of Overlay Networks (1)

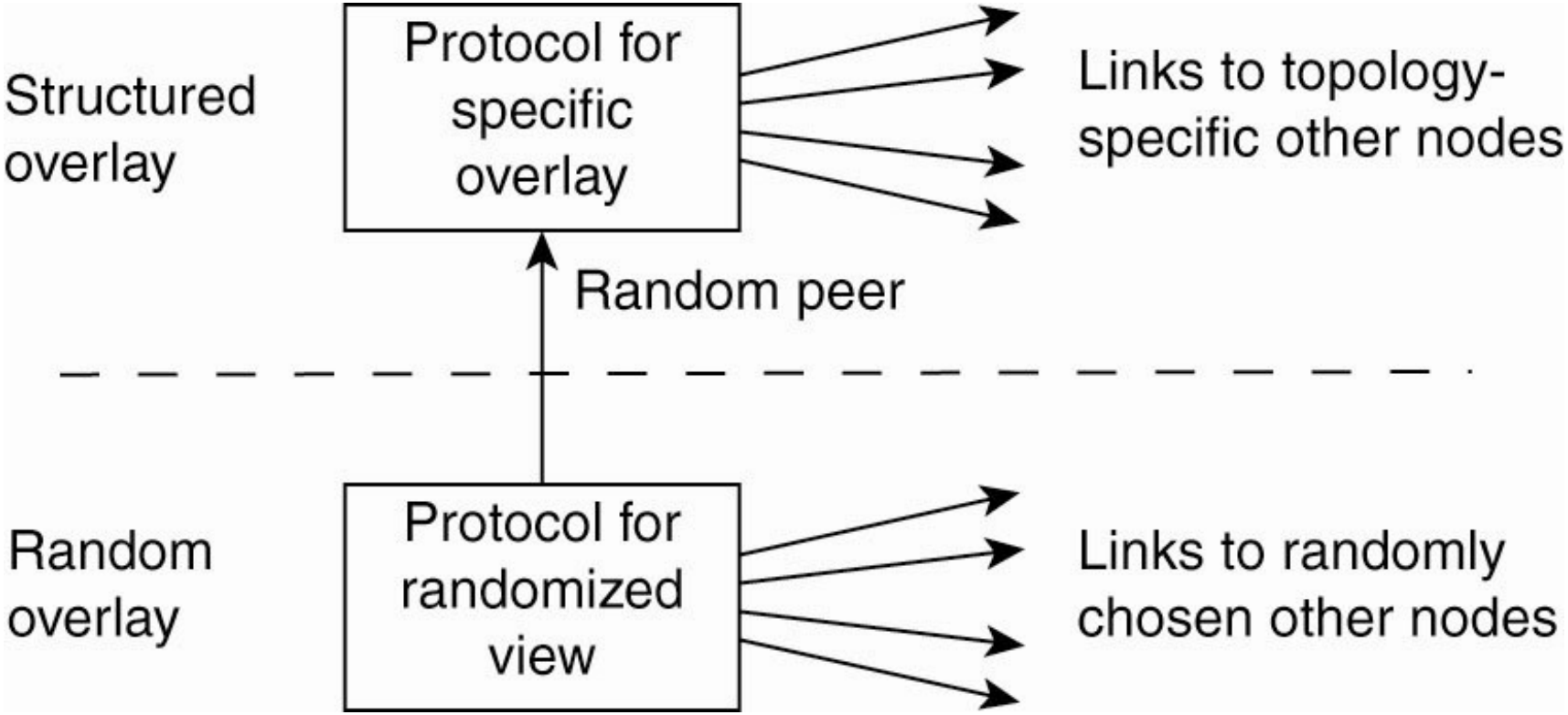

Figure 2-10. A two-layered approach for constructing and maintaining *specific overlay topologies* using techniques from *unstructured peer-to-peer systems*.

Topology Management of Overlay Networks (2)

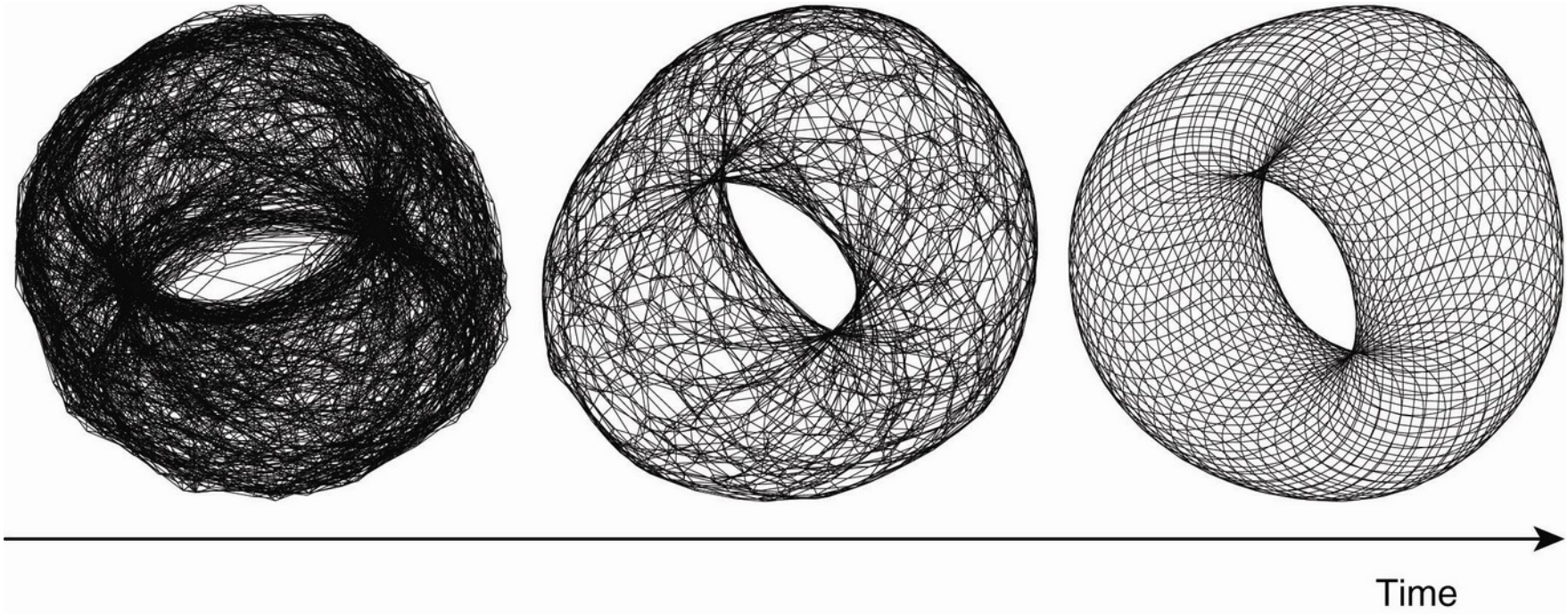

Figure 2-11. Generating a specific overlay network using a two-layered unstructured peer-to-peer system [adapted with permission from Jelasity and Babaoglu (2005)].

Superpeers

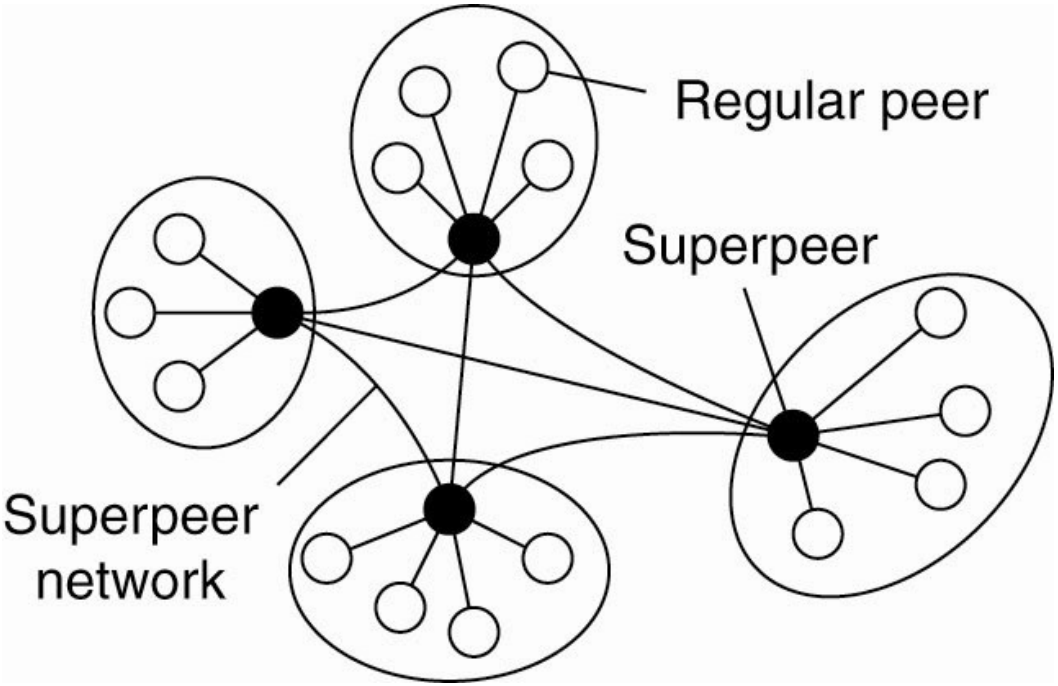

Figure 2-12. A hierarchical organization of nodes into a superpeer network.

Superpeers a) **maintain** an index, b) **monitor** the network's state, c) take care of **direct connection setups** between nodes

Collaborative Distributed Systems

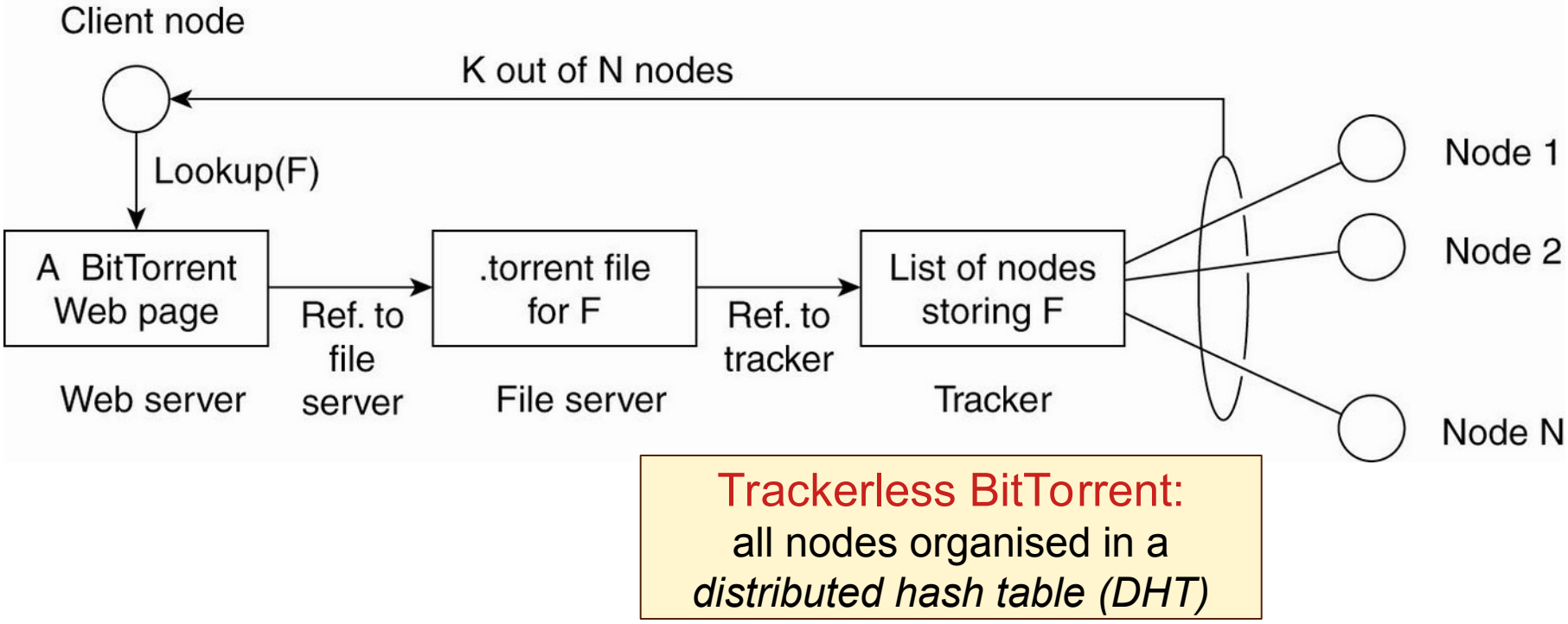

Figure 2-14. The principal working of Tracker-based BitTorrent [adapted with permission from Pouwelse et al. (2004)].