Distributed Systems

Introduction

Distributed Computing

- Is a field of computer science that studies distributed systems
- A distributed system is a model in which components located on networked computers communicate and coordinate their actions by passing messages
- A computer program that runs in a distributed system is called a distributed program
- Message passing mechanism, including pure HTTP, RPC
- RPC is like a chat system

Distributed System

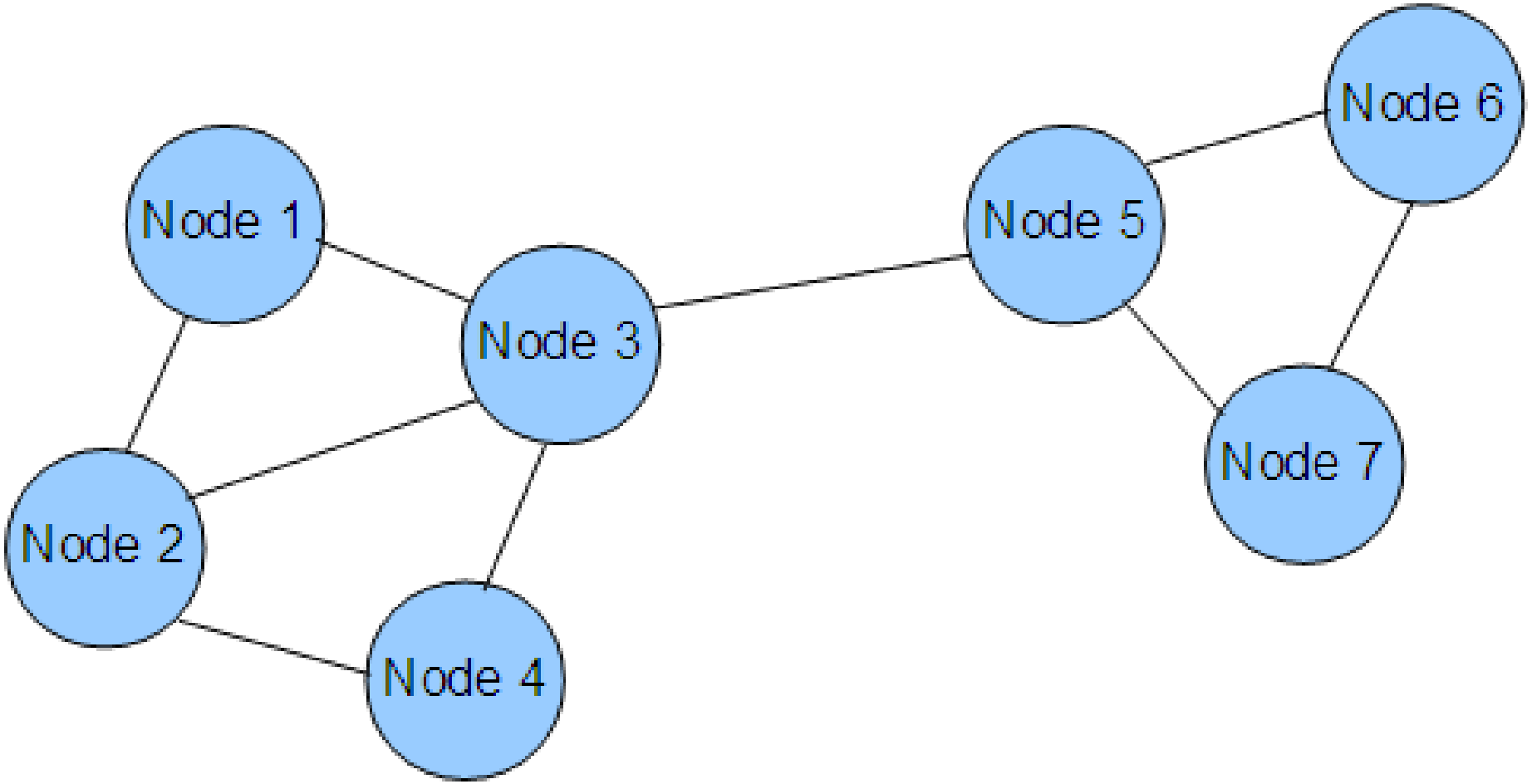
Definition

- A distributed system is a collection of autonomous computing elements that appears to its users as a single coherent system

Characteristic features

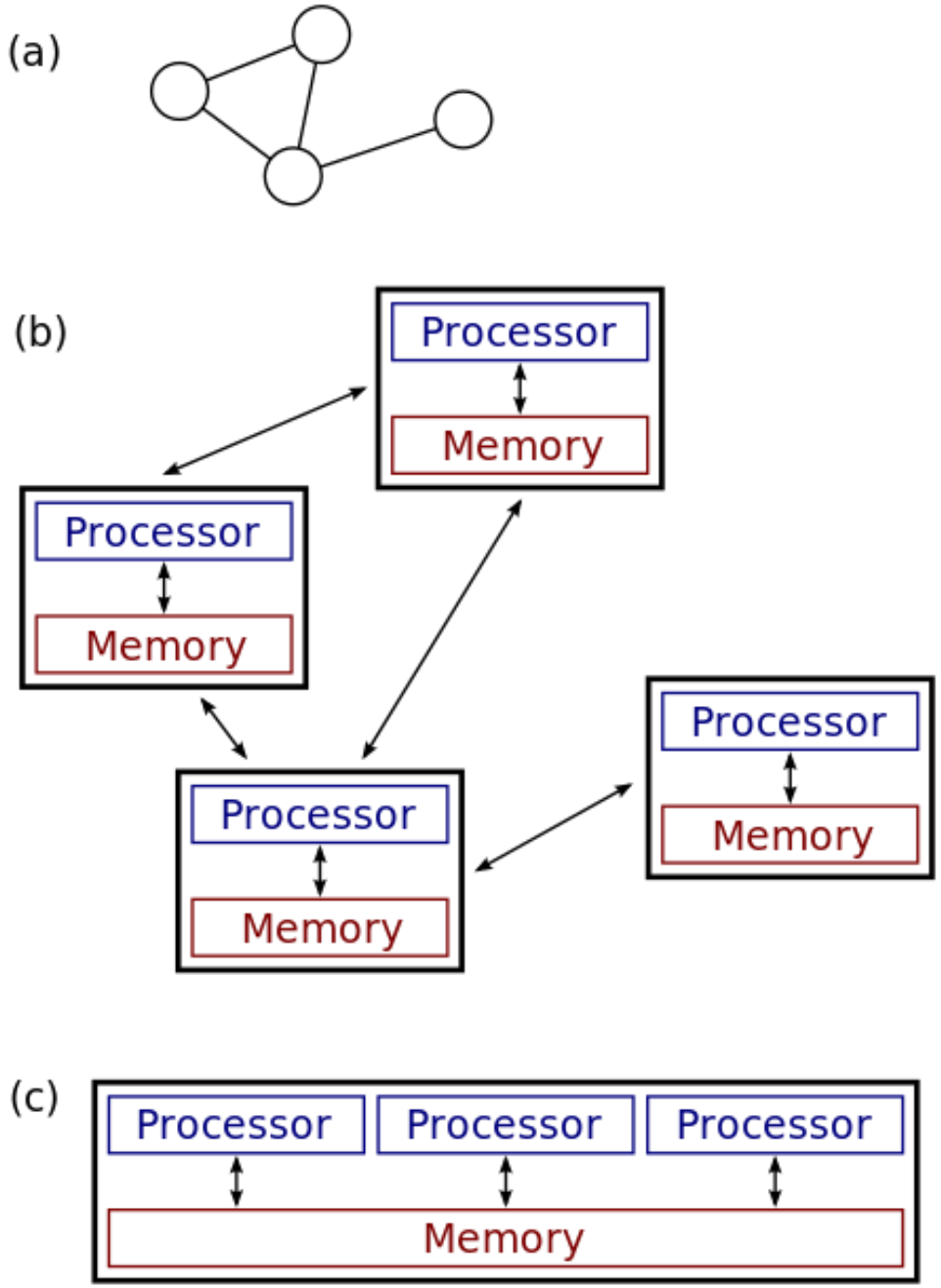
- Autonomous computing elements, also referred to as nodes, be they hardware devices or software processes.
- Single coherent system: users or applications perceive a single system ⇒ nodes need to collaborate.

Distributed System Node



(a), (b): a distributed system.

(c): a parallel system.



Collection of autonomous nodes

Node

- Autonomous
- Time

Collection of nodes

- How to manage group membership?
- How to know that you are indeed communicating with an authorized (non)member?

Organization

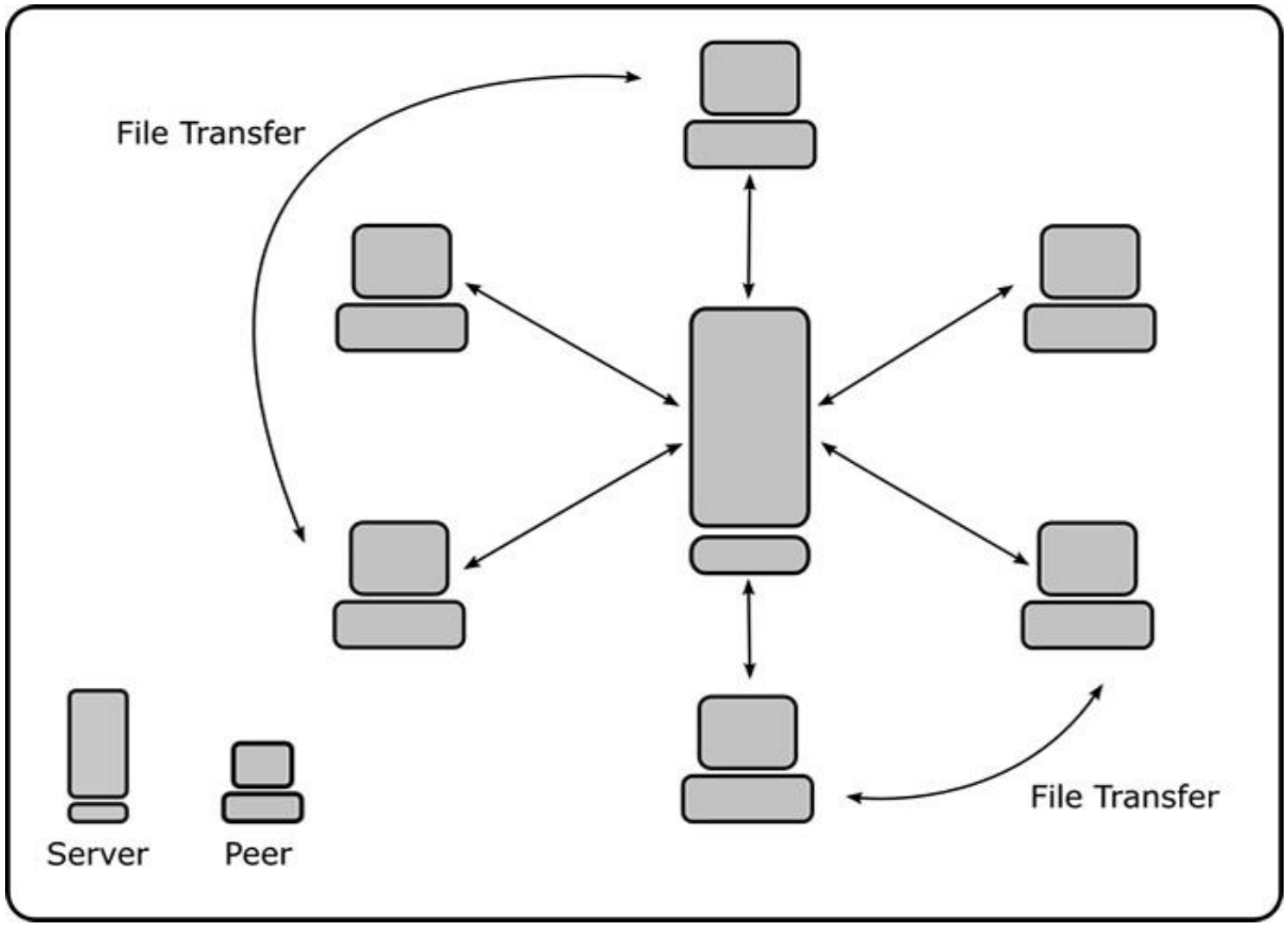
Overlay network

- Each node in the collection communicates only with other nodes in the system, its neighbors.
- The set of neighbors may be dynamic, or may even be known only implicitly (i.e., requires a lookup).

Overlay types

- Well-known example of overlay networks: peer-to-peer systems.
- Structured: each node has a well-defined set of neighbors with whom it can communicate (tree, ring).
- Unstructured: each node has references to randomly selected other nodes from the system.

Peer-to-peer systems



Coherent system

Essence

- The collection of nodes as a whole operates the same, no matter where, when, and how interaction between a user and the system takes place

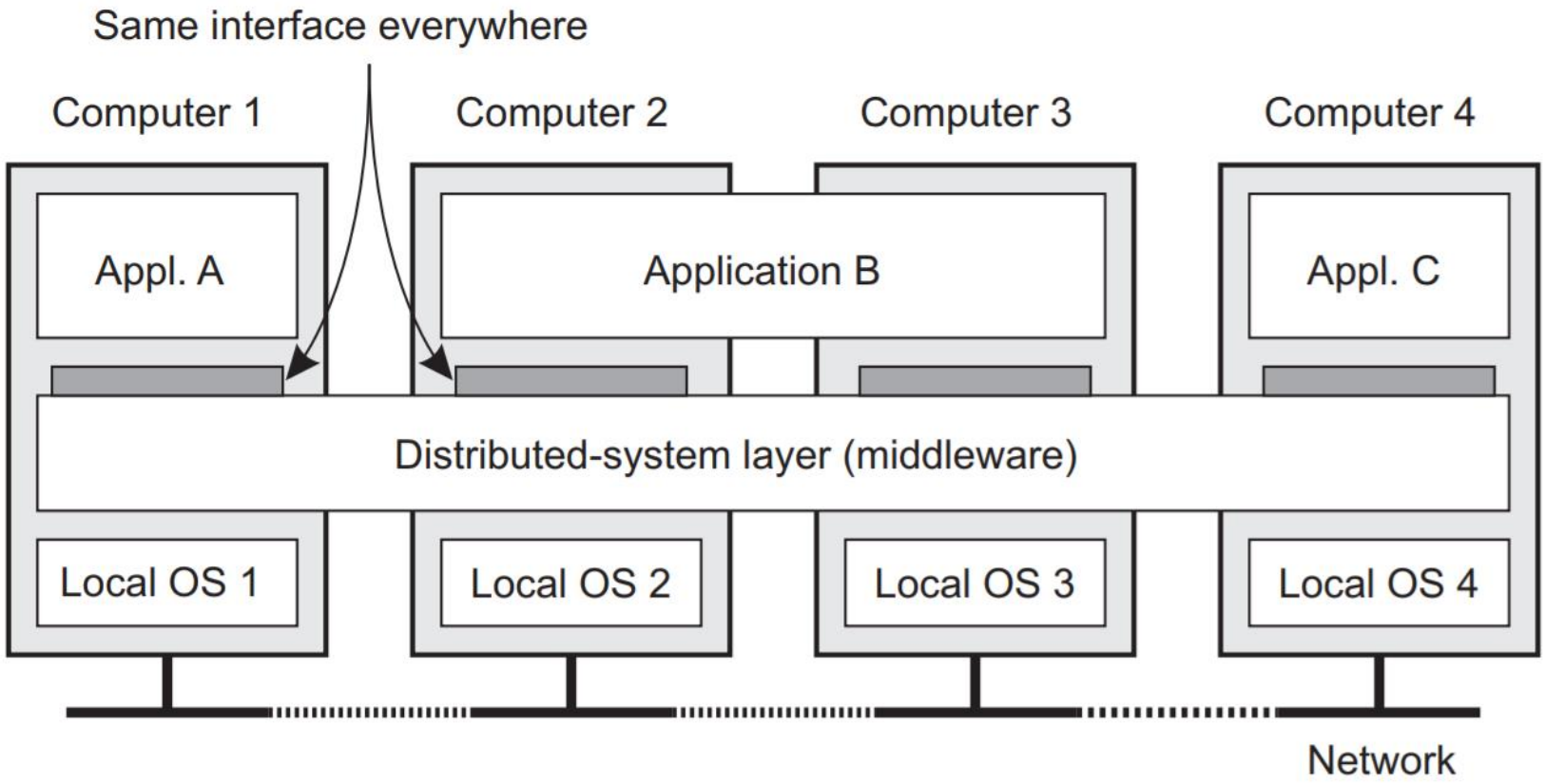
Examples

- An end user cannot tell where a computation is taking place
- Where data is exactly stored should be irrelevant to an application
- If or not data has been replicated is completely hidden

It is inevitable that at any time only a part of the distributed system fails.

Hiding partial failures and their recovery is often very difficult and in general impossible to hide.

Middleware: the OS of distributed systems



What to achieve?

- Support sharing of resources
- Distribution transparency
- Openness
- Scalability

Sharing resources

Examples

- Cloud-based shared storage and files
- Peer-to-peer assisted multimedia streaming
- Shared mail services (Mail systems)
- Shared Web hosting (Distribution networks)

Distribution 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

Transparency (Drawback)

Aiming at full distribution transparency may be too much:

- There are communication latencies that cannot be hidden
- Completely hiding failures of networks and nodes is (theoretically and practically) impossible
  - You cannot distinguish a slow computer from a failing one
  - You can never be sure that a server actually performed an operation before a crash

Transparency

Aiming at full distribution transparency may be too much:

- Full transparency will cost performance, exposing distribution of the system
- Keeping replicas exactly up-to-date with the master takes time
- Immediately flushing write operations to disk for fault tolerance

Transparency

Conclusion

Distribution transparency is a nice a goal

but achieving it is a different story

Openness

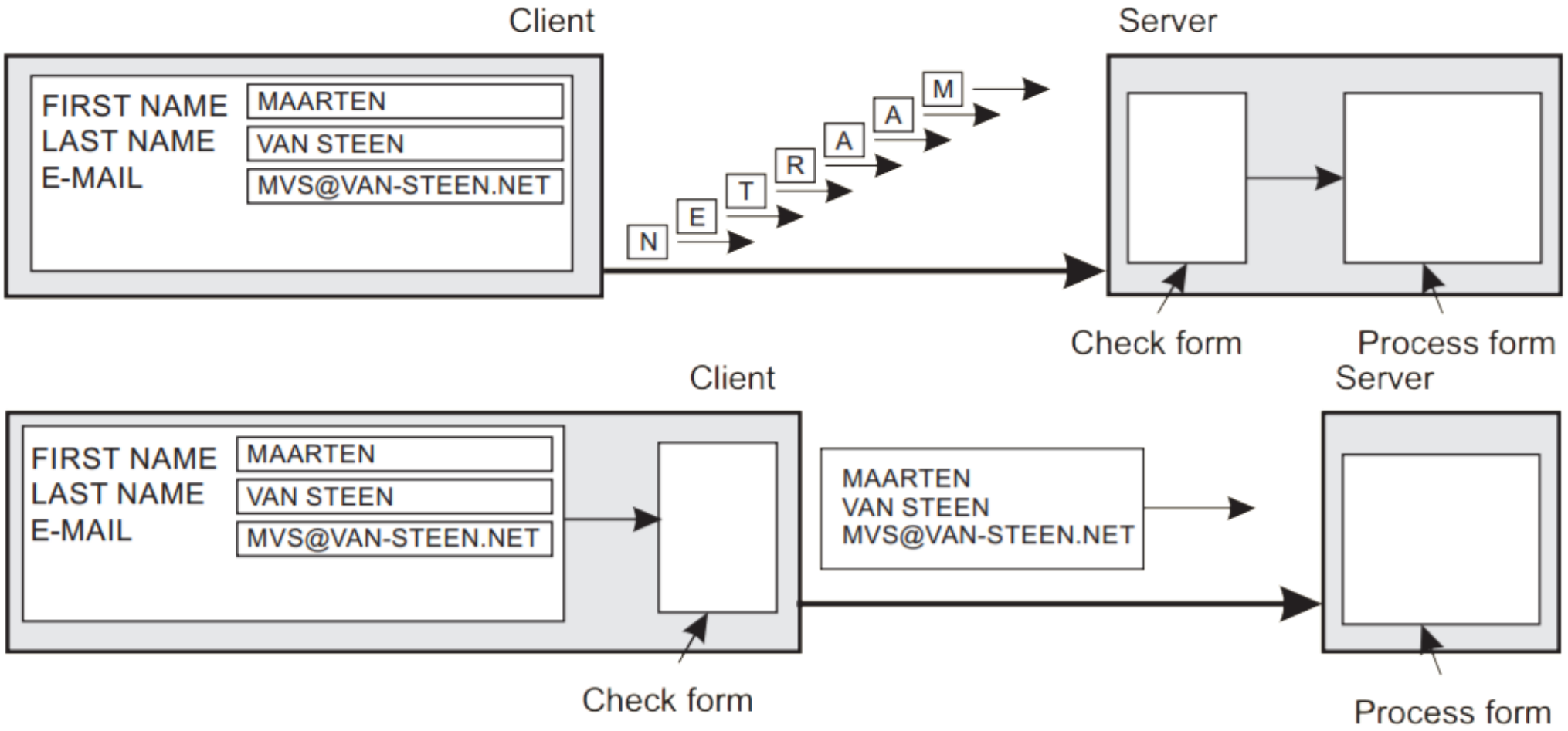
Be able to interact with services from other open systems, irrespective of the underlying environment:

- Systems should conform to well-defined interfaces
- Systems should easily interoperate
- Systems should support portability of applications
- Systems should be easily extensible

Scale

- Number of users and/or processes
  (size scalability)
- Maximum distance between nodes
  (geographical scalability)
- Number of administrative domains
  (administrative scalability)

Techniques for scaling



Techniques for scaling

Partition data and computations across multiple machines

- Move computations to clients
  (Java applets)
- Decentralized naming services
  (DNS)
- Decentralized information systems
  (WWW)