Architectural Design of Distributed Applications

Chapter 13

Part of Design Analysis

Designing Concurrent, Distributed, and Real-Time Applications with UML
Hassan Gomaa (2001)

Distributed Application Terminology

- *Subsystems* consist of a collection of tasks executing on a single logical node.
  - Also called “components”
- Subsystems may or may not run on multiple physical nodes.
- *Components* are logical nodes.
- *Nodes* are physical nodes.

Distributed Application Terminology (cont)

- *Distributed Applications* consist of subsystems designed to run on multiple, distributed physical nodes.
  - Whether they are actually on distributed nodes or not is a configuration decision; the system is designed to provide this option.

Configurable Architectures and Software Components (13.1)

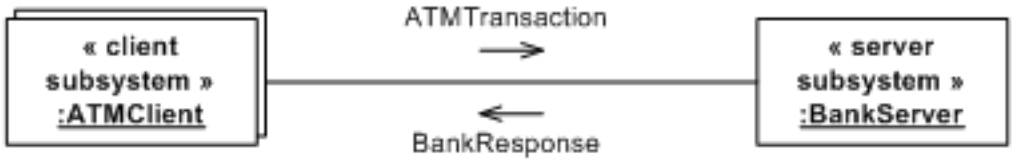
- Distributed applications are intended to be configurable – flexibility is a goal.
- Communication between tasks in different subsystems is limited to messages to achieve this flexibility
  - Recall that messages are *events* with optional *attributes* or *data*.

Steps in Designing Distributed Applications (13.2)

1. System Decomposition
   The overall system is broken into components that could be located on distributed physical nodes.
2. Subsystem Decomposition
   Each subsystem is broken into smaller components as necessary. Because each subsystem operates on a single node, non-distributed techniques may be used.
3. System Configuration
   Instances of the subsystems are defined, connected, and mapped onto a hardware configuration.

Designing Distribute Subsystems (13.3.1)

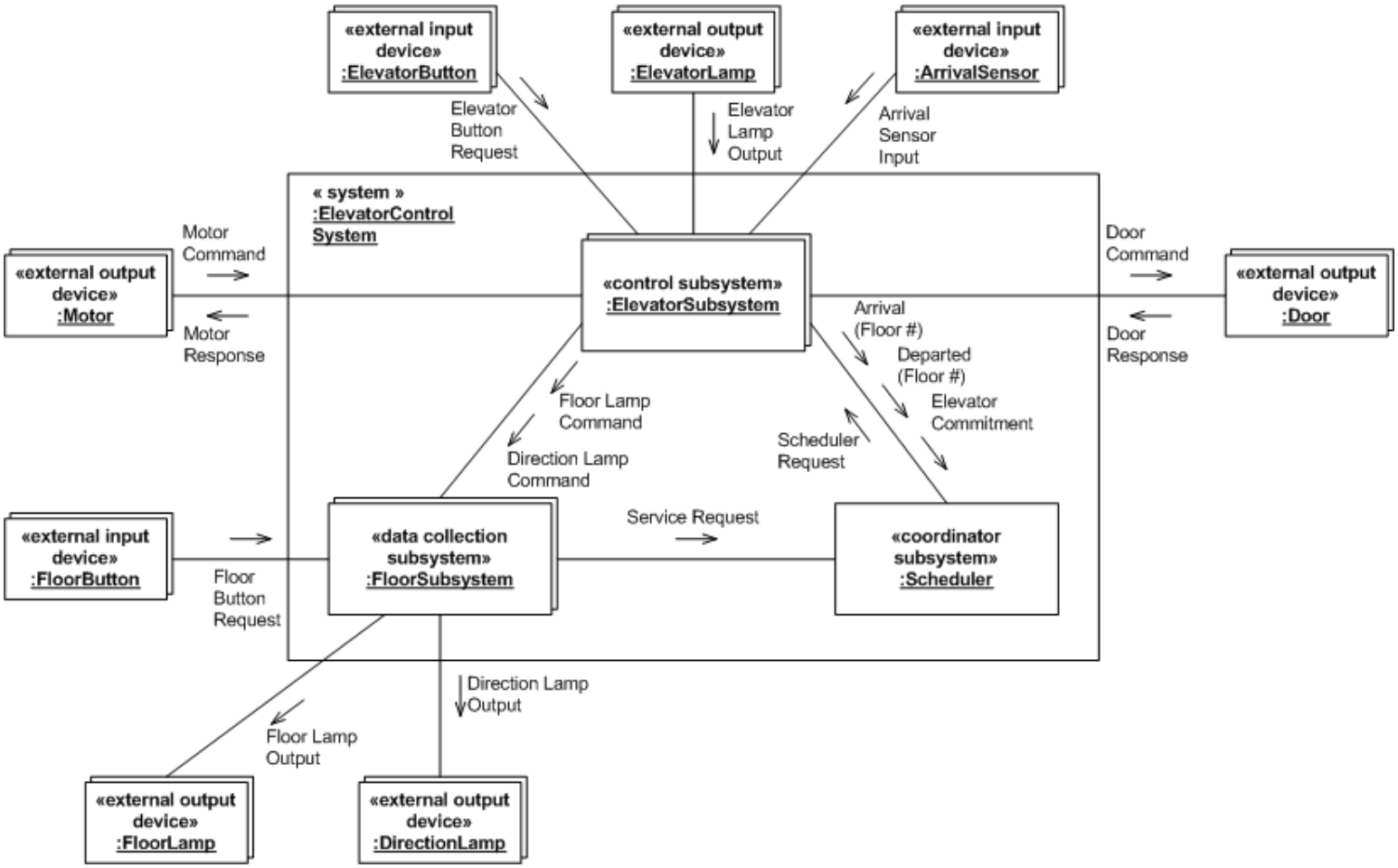
- Places to start system decomposition:
  - Use Cases
    - Which objects must collaborate frequently?
  - Geographic constraints
    - Most often leads to client/server relationships, but may be peer-to-peer
    - Example: Bank Servers must be located at the bank.



Aggregate and Composite Subsystems in COMET (13.3.2)

- *Composite Subsystems* comprise objects that share geographic location, but not necessarily functionality.
- *Aggregate Subsystems* comprise objects that share functionality, but not necessarily geographic location.
- Systems may have both composite and aggregate subsystems.

Aggregate and Composite Subsystems in COMET (13.3.2, example)



Criteria for Designing Configurable Distributed Subsystems (13.3.3-4)

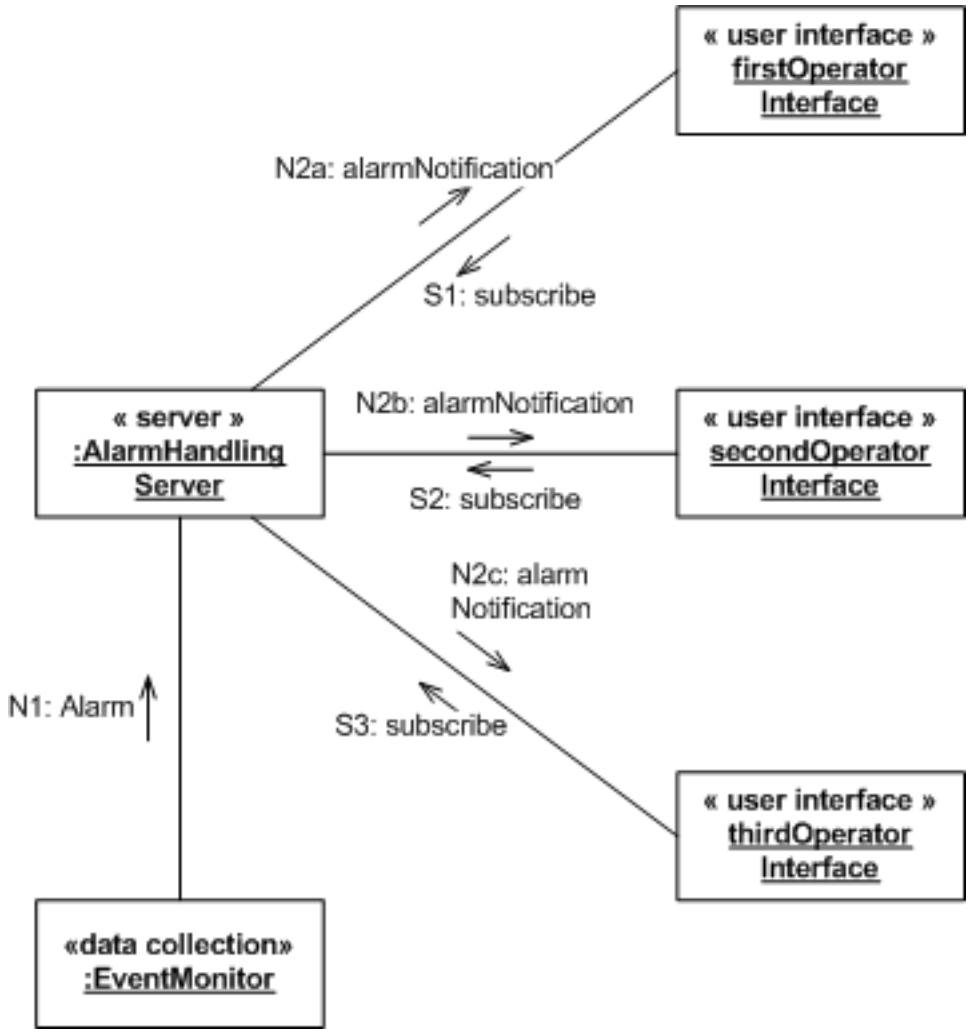
- Proximity to the source of the physical data.
- Localized autonomy.
- Performance.
- Specialized hardware.
- User Interface.
- Server.

Designing Subsystem Interfaces (13.4)

- Synchronous or Tightly Coupled Communication
  - Wait for a reply.
- Asynchronous or Loosely Coupled Communication
  - "Fire and forget" messaging.
- Servers may support both, even to the degree of allowing each client to select its communication style.

Subscription/Notification Group Communication (13.4.4)

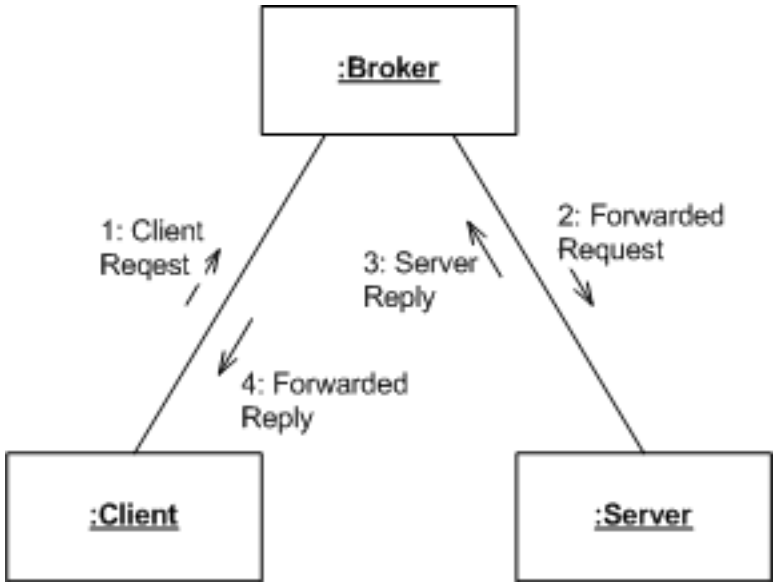
- Objects subscribe to groups
- Messages are sent to all members of groups.



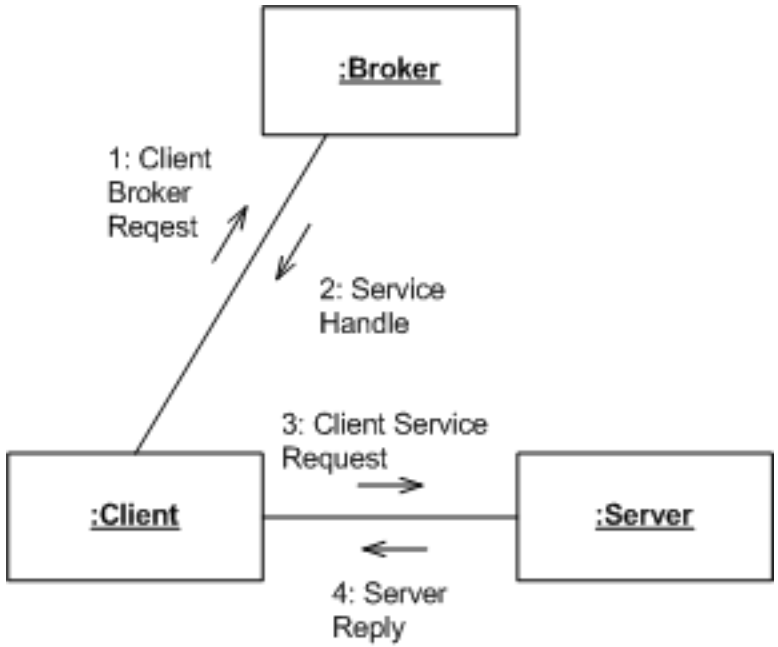
Brokered Communication (13.4.5)

- An *Object Broker* facilitates communication between clients and servers.
- Advantages:
  - Location information is only stored in one place.
  - Multiple servers may be used for a single type of request.

Brokered Communication (13.4.5, examples)



Forwarding Design



Handle-driven Design

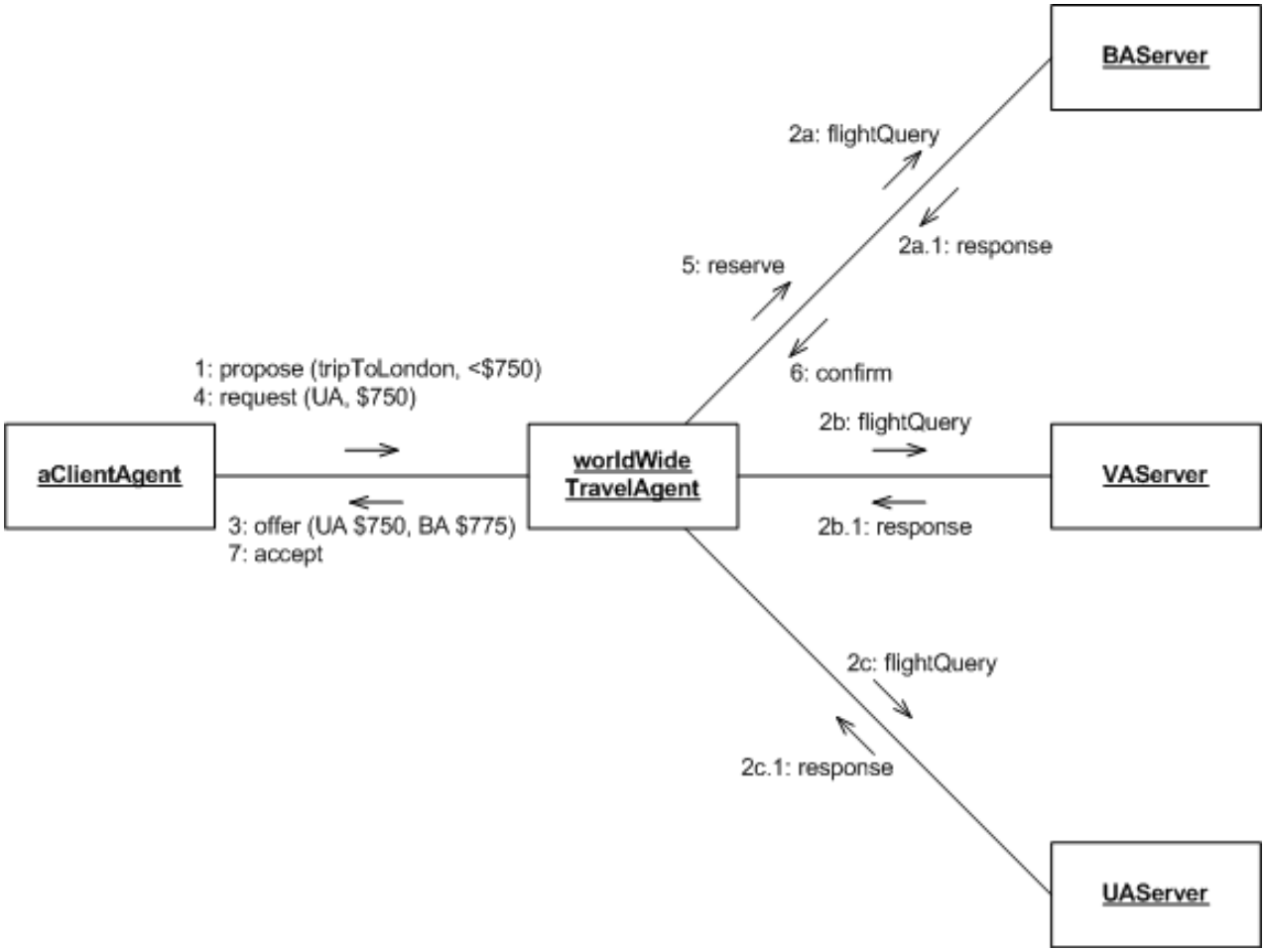
Negotiated Communication (13.4.6)

- *Negotiated Communication* involves clients, or client agents, and server or server agents.
  - *Agents* are objects that act on behalf of other objects.
- Negotiated communication allows clients and servers to cooperatively make decisions and thus better use their resources.

Negotiated Communication (13.4.6, cont)

- *Clients*, or *Client Agents*, may:
  - Propose Service
  - Request Service
  - Reject Service Offer
- *Servers*, or *Server Agents*, may:
  - Offer a service
  - Reject client request/proposal
  - Accept client request/proposal

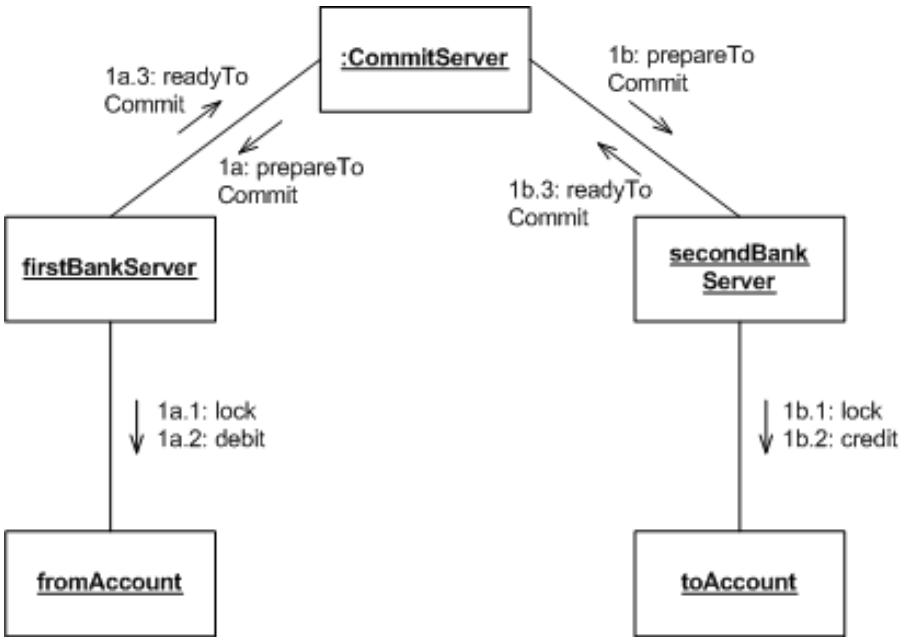
Negotiated Communication (13.4.6, example)



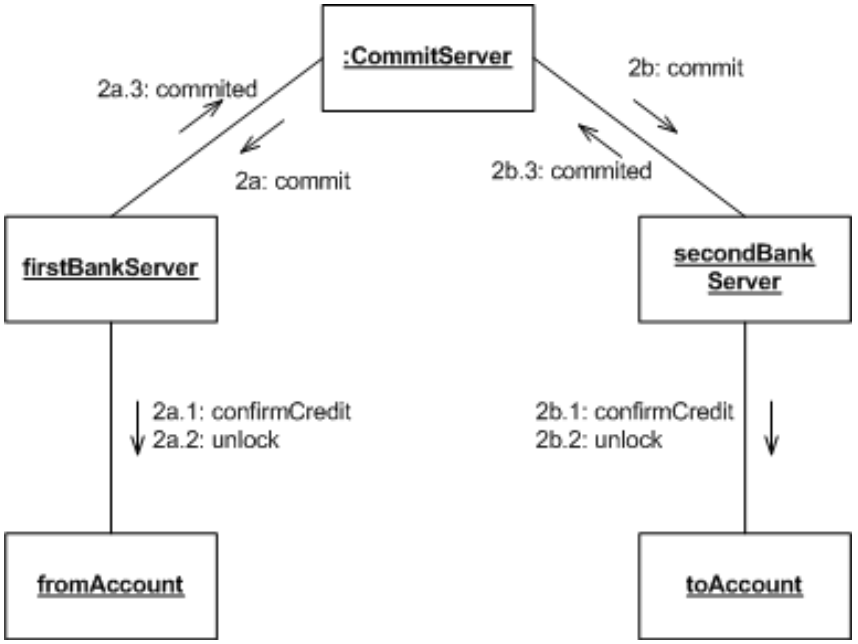
Transaction Management (13.5)

- Two-Phase Commit protocols ensure all desired outcomes of a transaction, or message sequence, occur.
- Consider a banking transfer:
  - Either both the credit and debit operations occur;
  - Or neither the credit or debit operation should occur.

Transaction Management (3.5.1, example)



Prepare the transaction

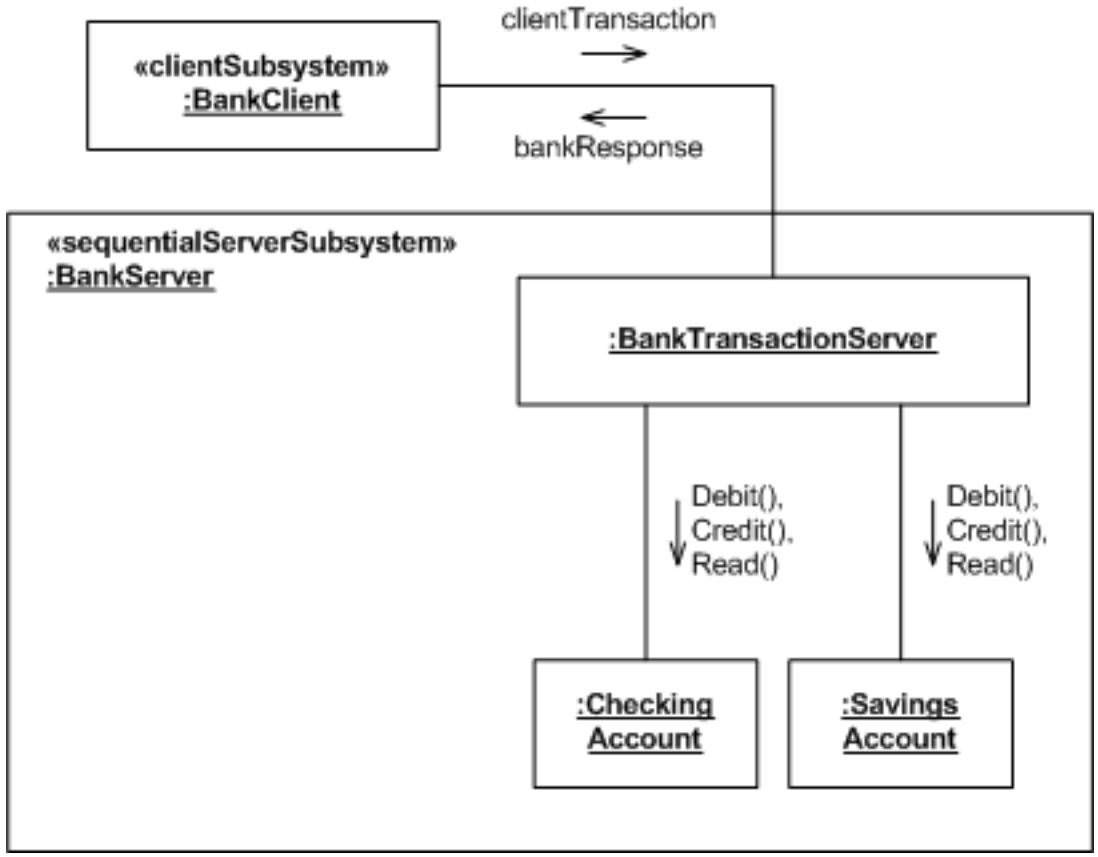


Commit the transaction

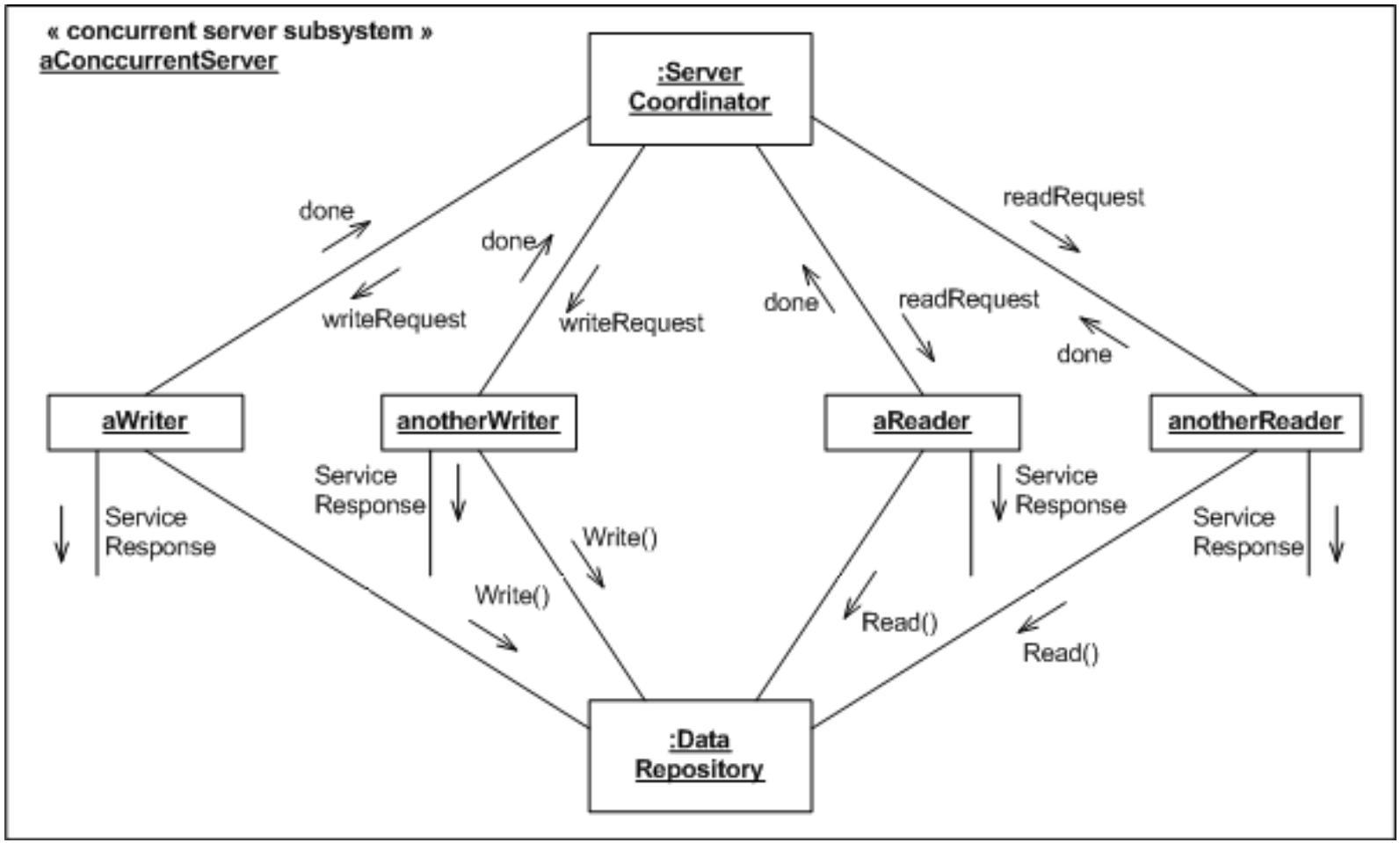
Design of Server Subsystems (13.6)

- Server subsystems may be:
  - Sequential – processing only one request at a time.
  - Concurrent – processing several requests simultaneously

Sequential Server Subsystem (13.6.1, example)



Concurrent Server Subsystem (13.6.2, example)



Distribution of Data (13.7)

- In single server configurations, the server may become a bottleneck for performance issues.

- Data can be distributed to prevent performance bottlenecks using two methods:
  - Distributed Server
  - Data Replication
- Distributed Server systems involve keeping data at the location it is collected, for local access.
- Data Replication systems duplicate data to multiple locations, ensuring

Distributed Server (13.7.1)

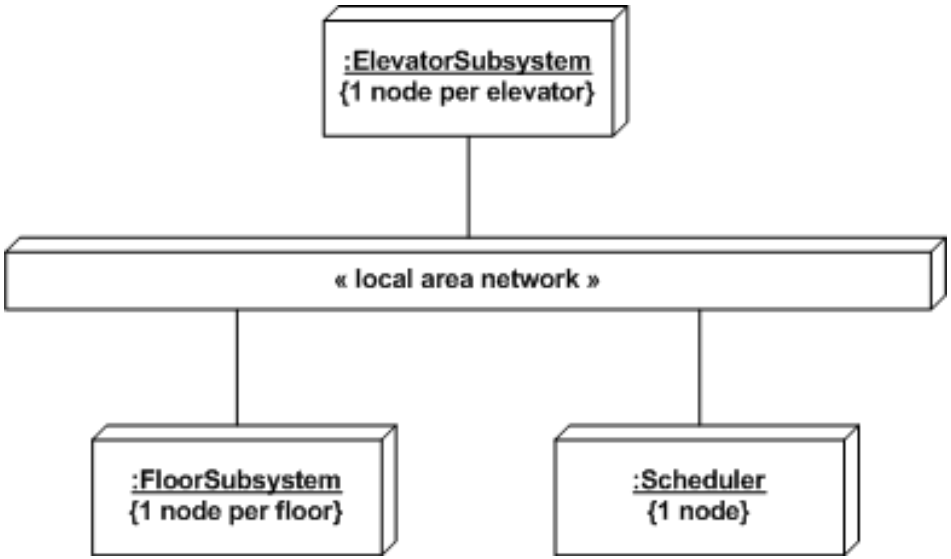
- *Distributed Server* systems involve storing data at the location it is collected.
- The local server then responds to local requests
- If the data is needed elsewhere, a mechanism must be designed to facilitate transfer. This is an example of data replication.

Data Replication (13.7.2)

- *Data Replication* involves duplicating data to more than one location.
- Procedures for ensuring all the copies of data remain updated must be designed.

System Configuration (13.8)

- System configuration involves three activities:
  - Defining instances of component types.
  - Interconnecting the defined instances.
  - Mapping the component instances to physical nodes.



Summary (13.9)

- Designing a system that can be configured onto distributed physical nodes requires special considerations.
- Subsystems may be Aggregate or Composite.
- Subsystem interfaces may be synchronous or asynchronous.
- Client/Server Communication may be:
  - Subscription/Notification
  - Brokered
  - Negotiated

Summary (13.9, cont)

- Two-phase commit transactions ensure “all-or-none” transactions.
- Server Subsystems may be:
  - Sequential
  - Concurrent
  - Distributed
- System Configuration involves defining, connecting, and mapping component instances onto physical nodes.