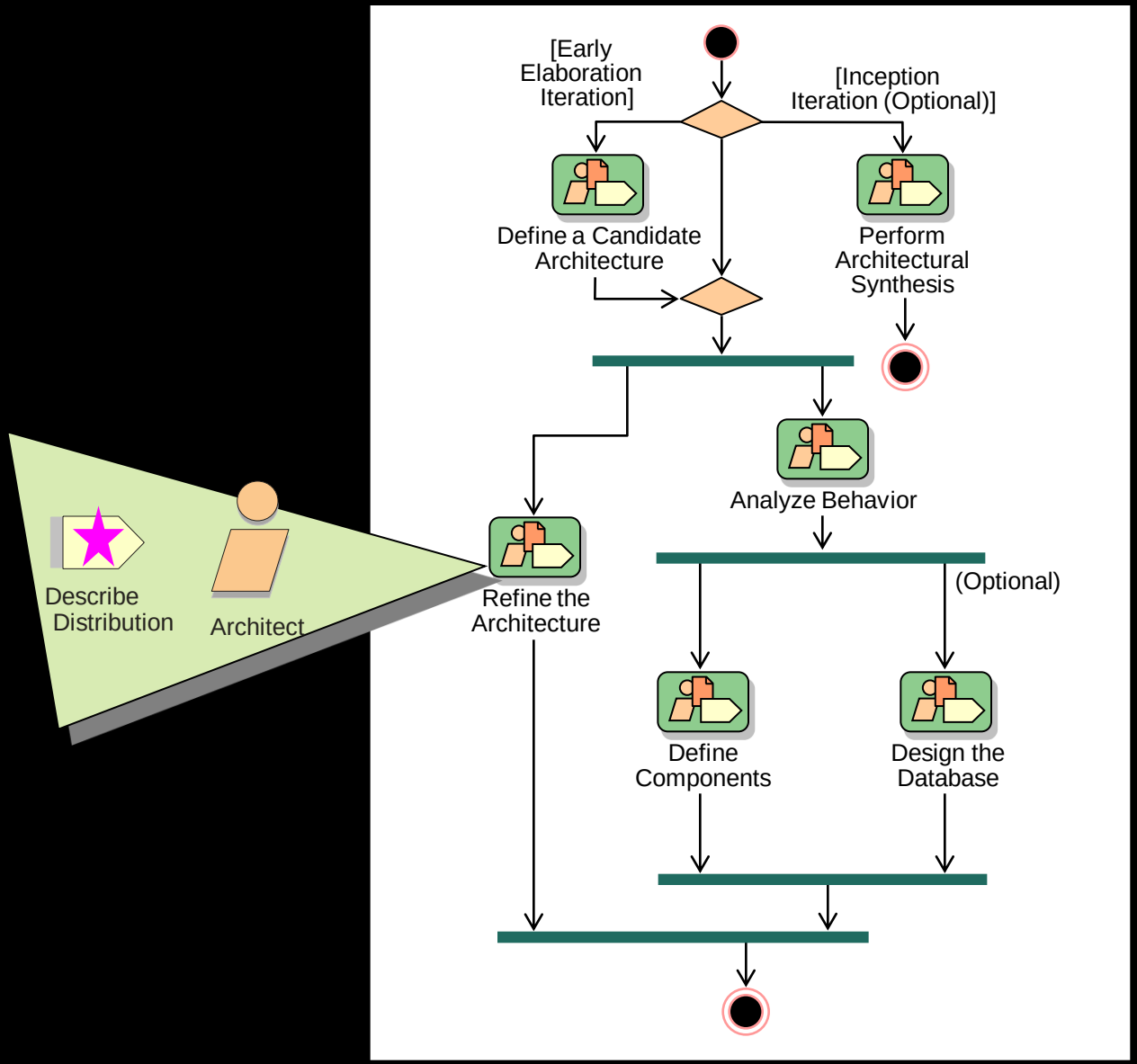
Object-Oriented Analysis and Design

Lecture 10: Describe Distribution

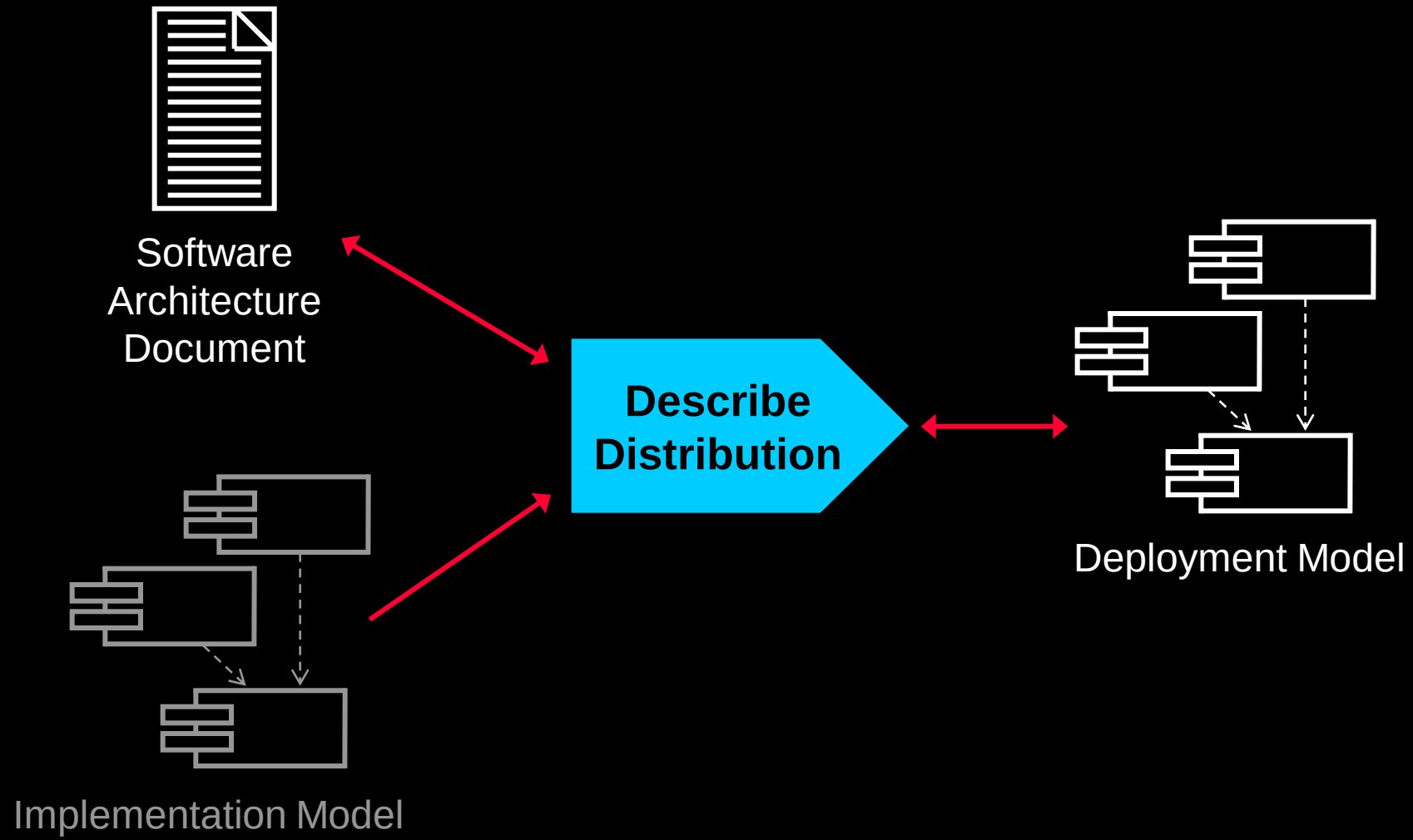
Objectives: Describe Distribution

- Explain the purpose of the Describe Distribution activity and when in the lifecycle it is performed
- Describe how the functionality of the system can be distributed across physical nodes
- Model the distribution decisions of the system in the Deployment Model
- Articulate the rationale and considerations that support the architectural decisions

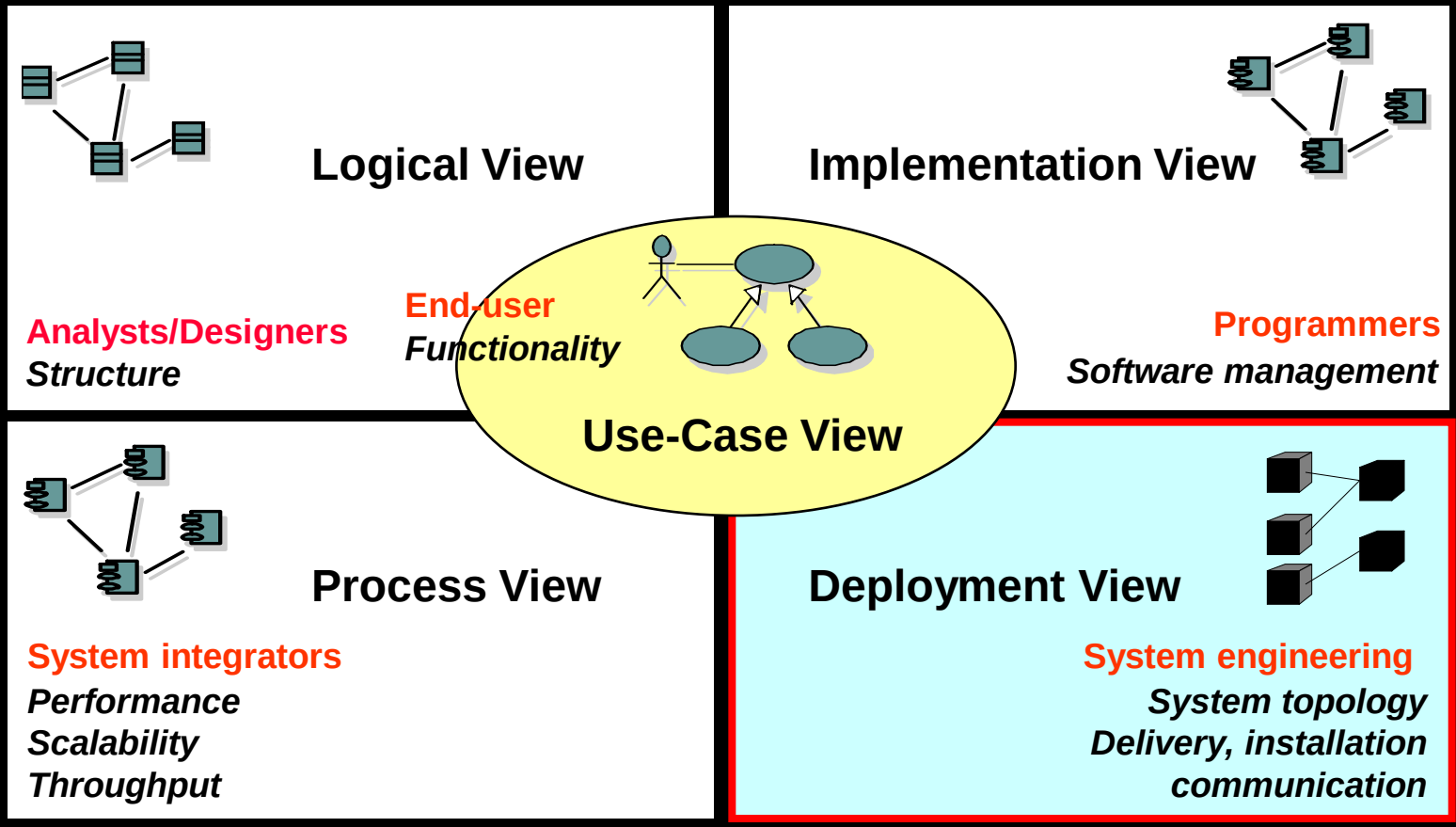
Describe Distribution in Context



Describe Distribution Overview



Key Concepts: The Deployment View



The Deployment View is an "architecturally significant" slice of the Deployment Model.

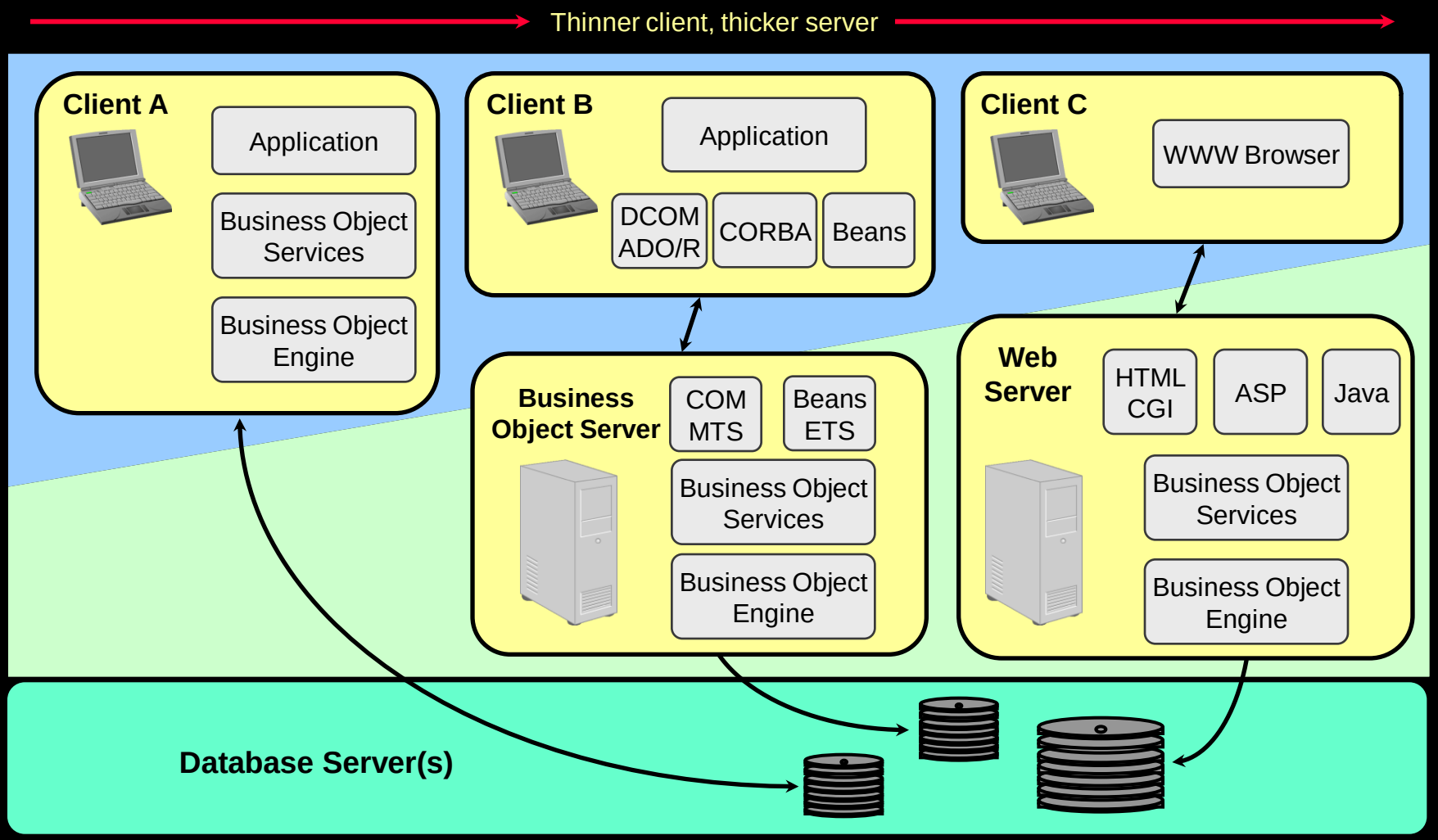
Why Distribute?

- Reduce processor load
- Special processing requirements
- Scaling concerns
- Economic concerns
- Distributed access to the system

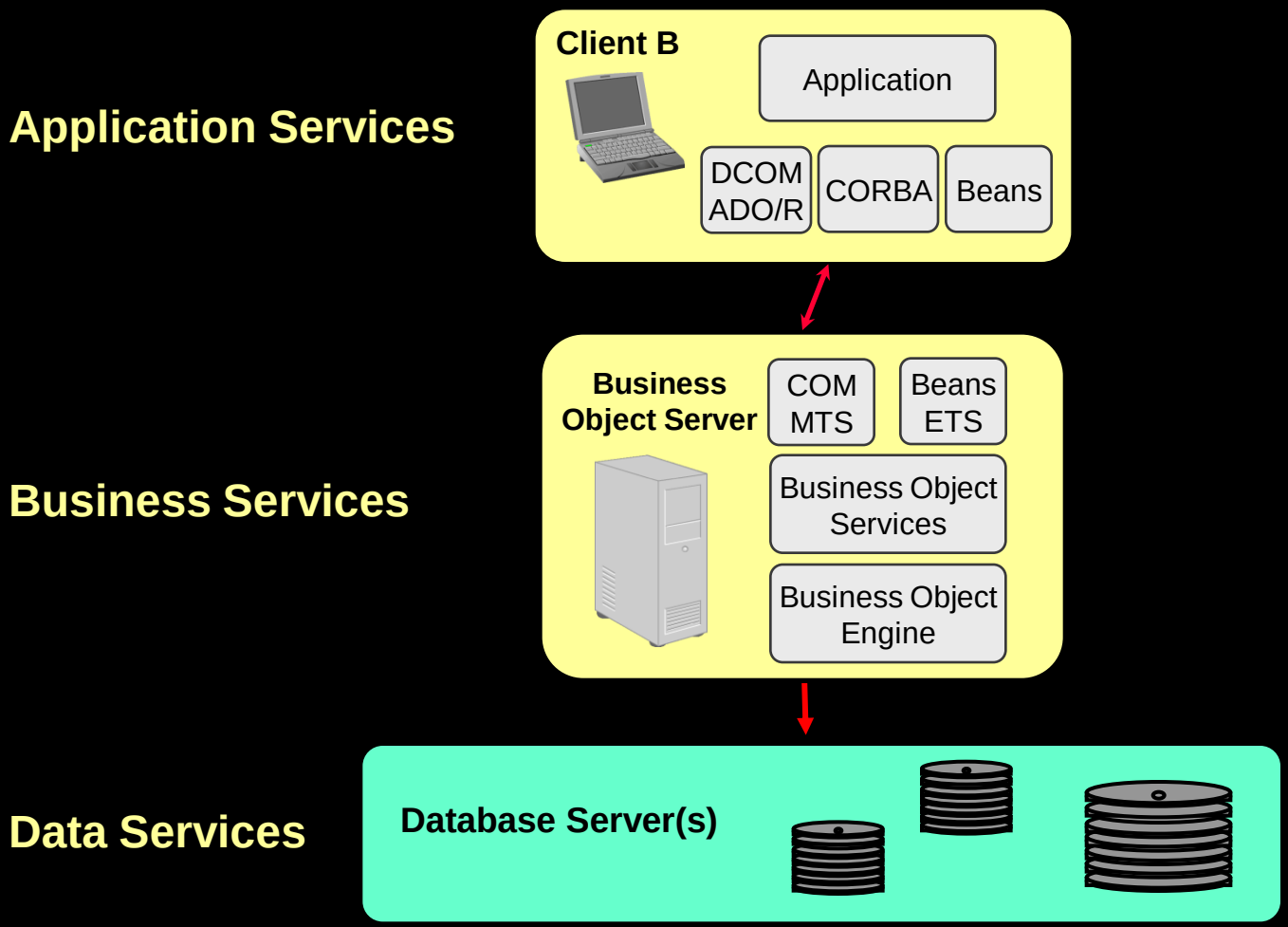
Distribution Patterns

- Client/Server
  - 3-tier
  - Fat Client
  - Fat Server
  - Distributed Client/Server
- Peer-to-peer

Client/Server Architectures



Client/Server: Three-Tier Architecture

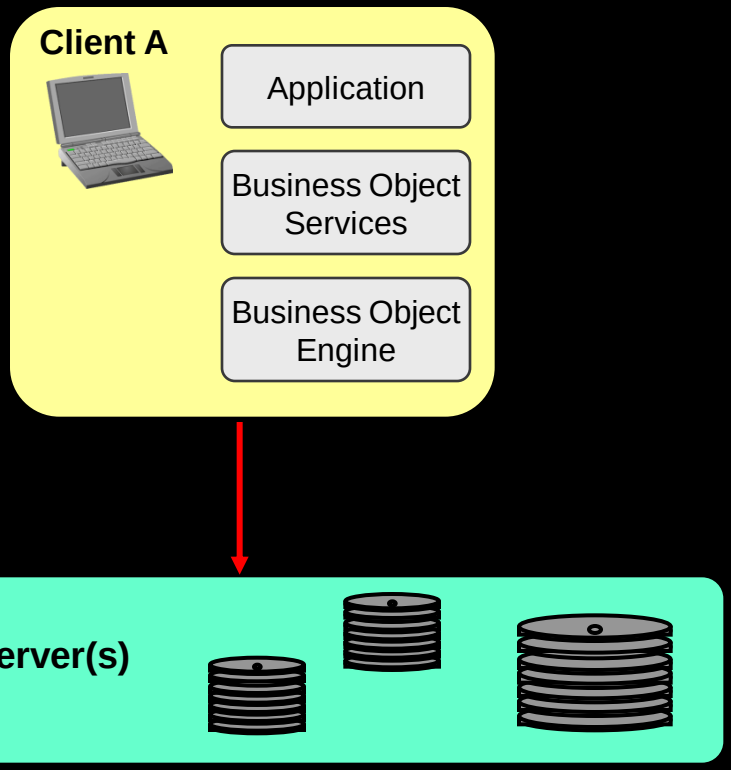


Client/Server: “Fat Client” Architecture

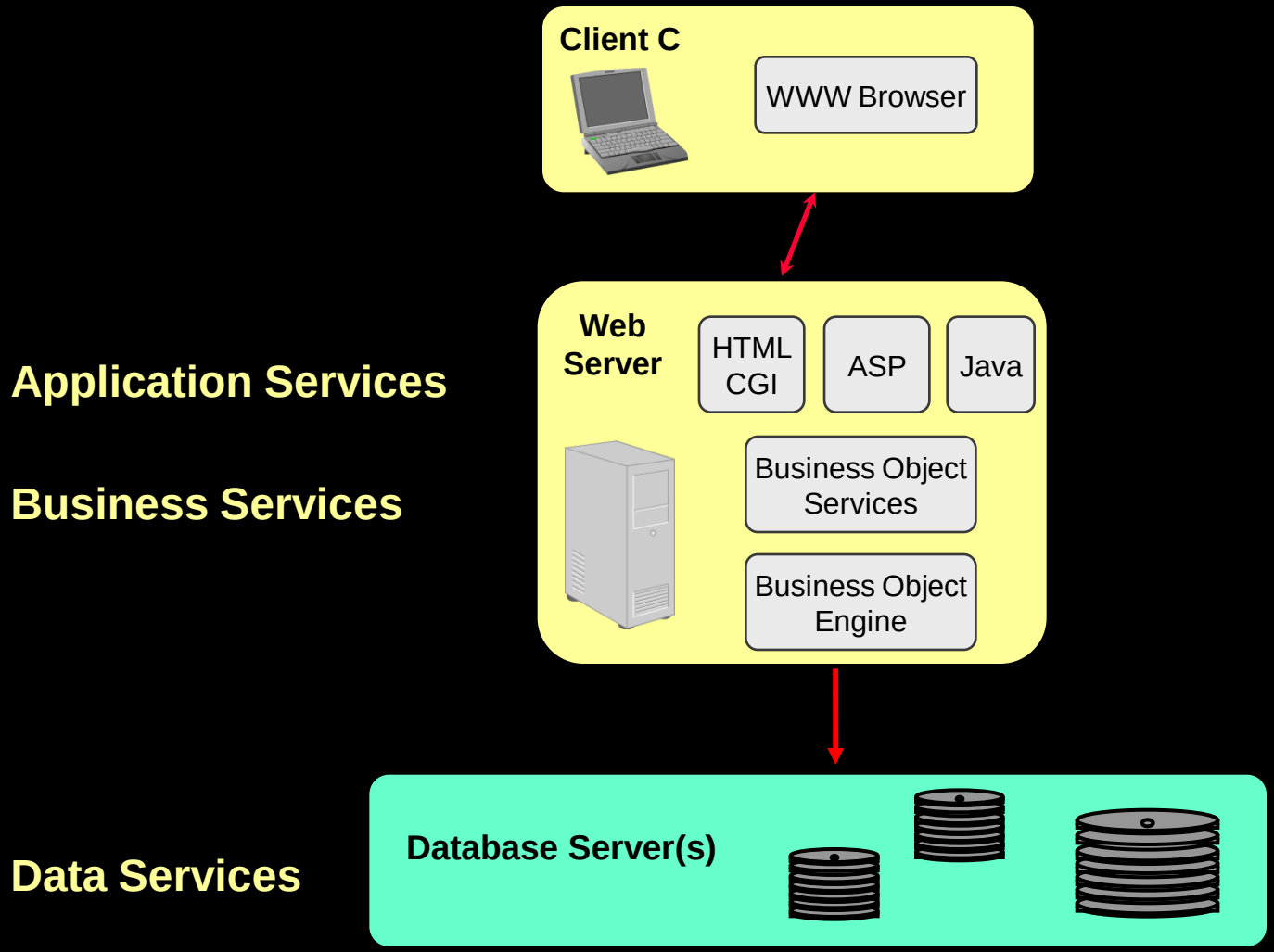
Application Services

Business Services

Data Services



Client/Server: Web Application Architecture

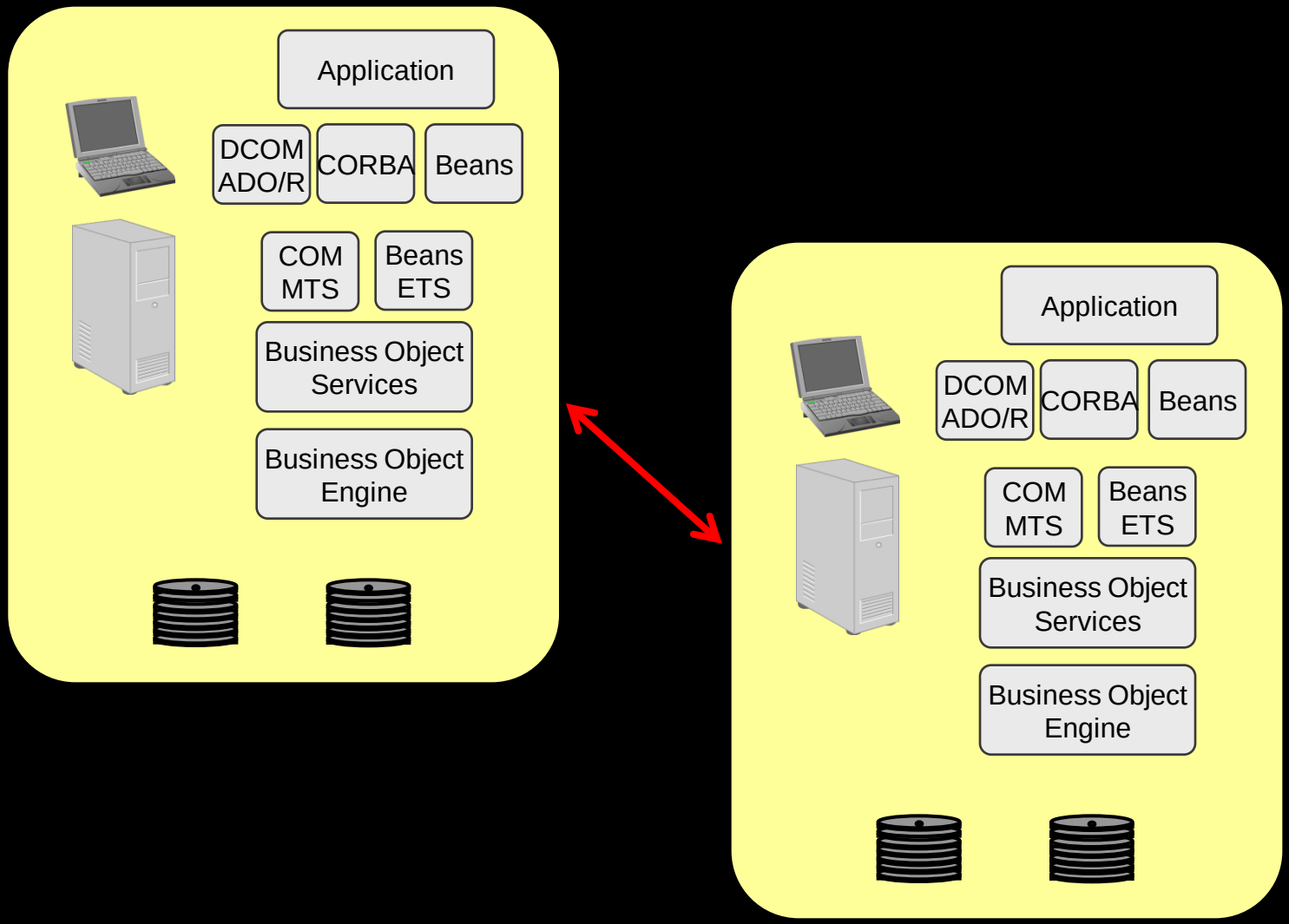


Peer-to-Peer Architecture

Application Services

Business Services

Data Services

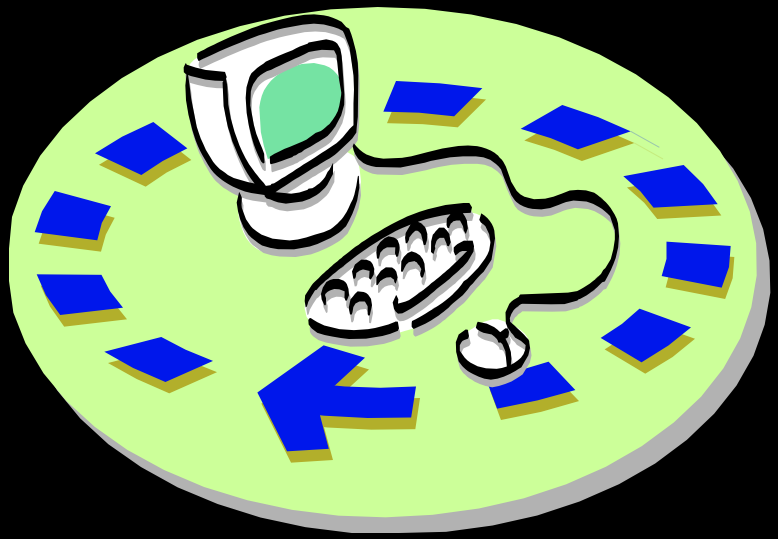


Describe Distribution Steps

- Define the network configuration
- Allocate processes to nodes
- Define the distribution mechanism

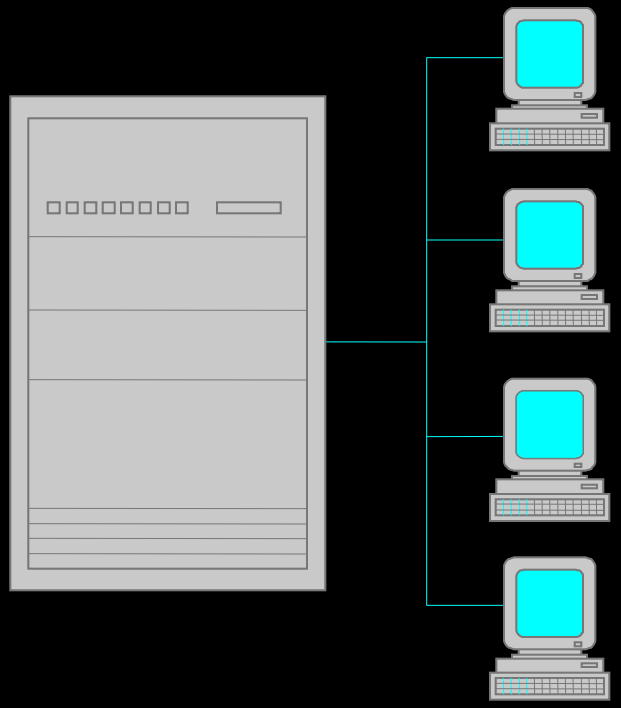
Describe Distribution Steps

- Define the network configuration
  - Allocate processes to nodes
  - Define the distribution mechanism



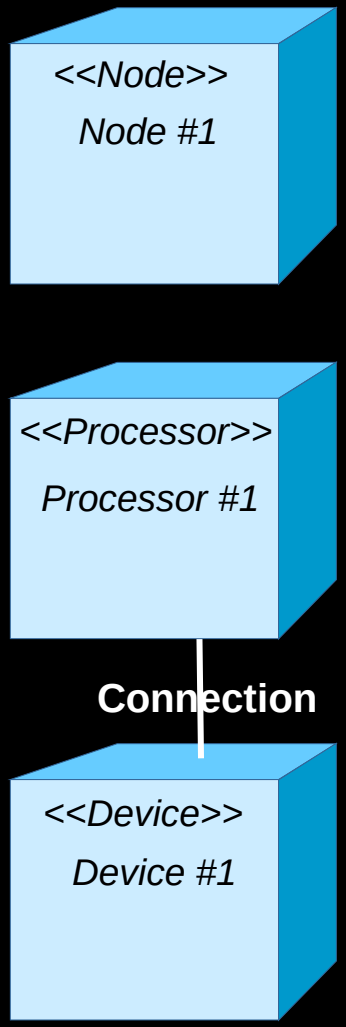
The Network Configuration

- End-user workstation nodes
- ”Headless" processing server nodes
- Special configurations
  - Development
  - Test
- Specialized processors

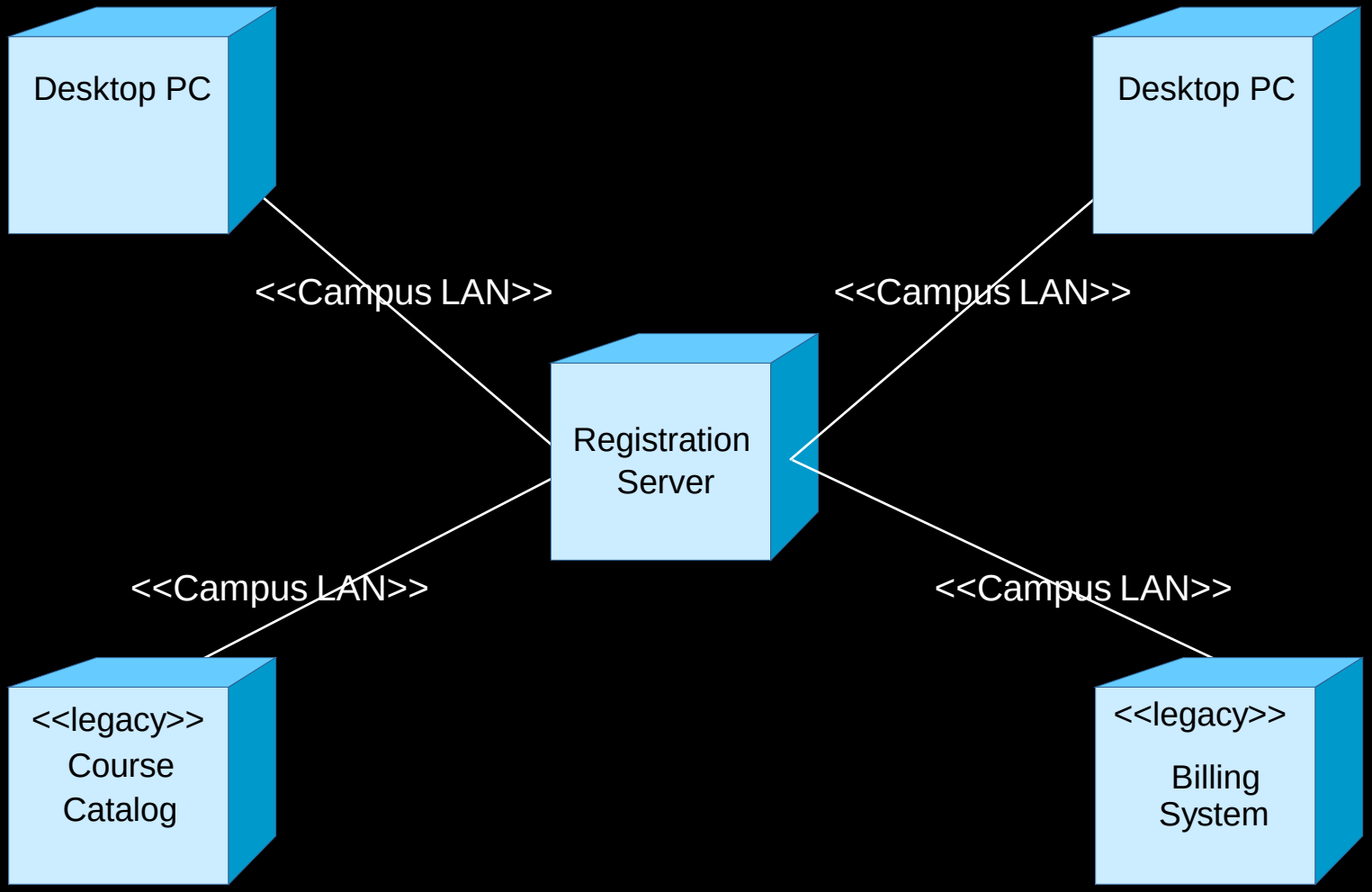


Deployment Model Modeling Elements

- Node
  - Physical run-time computational resource
  - Processor node
    - Executes system software
  - Device node
    - Support device
    - Typically controlled by a processor
- Connection
  - Communication mechanism
  - Physical medium
  - Software protocol



Example: Network Configuration

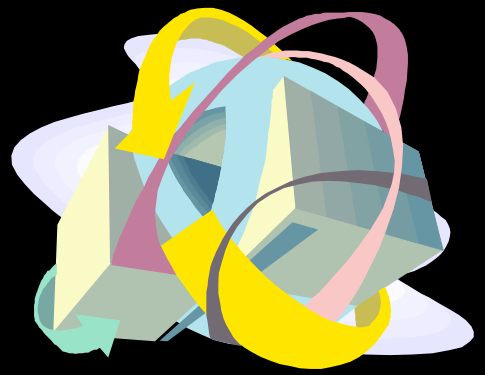


Describe Distribution Steps

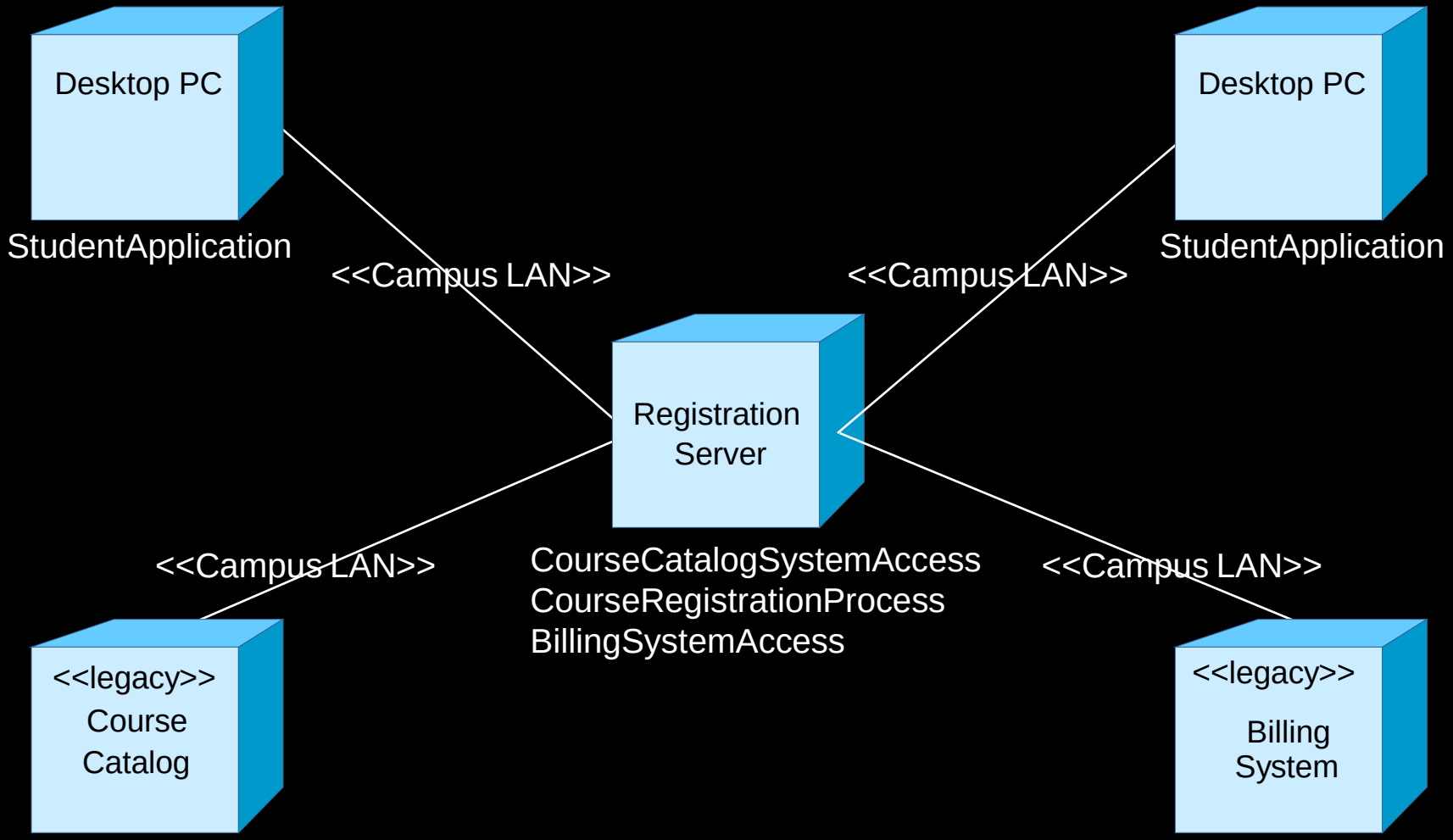
- Define the network configuration
- ★ Allocate processes to nodes
- Define the distribution mechanism

Process-to-Node Allocation Considerations

- Distribution patterns
- Response time and system throughput
- Minimization of cross-network traffic
- Node capacity
- Communication medium bandwidth
- Availability of hardware and communication links
- Rerouting requirements



Example: Process-to-Node Allocation

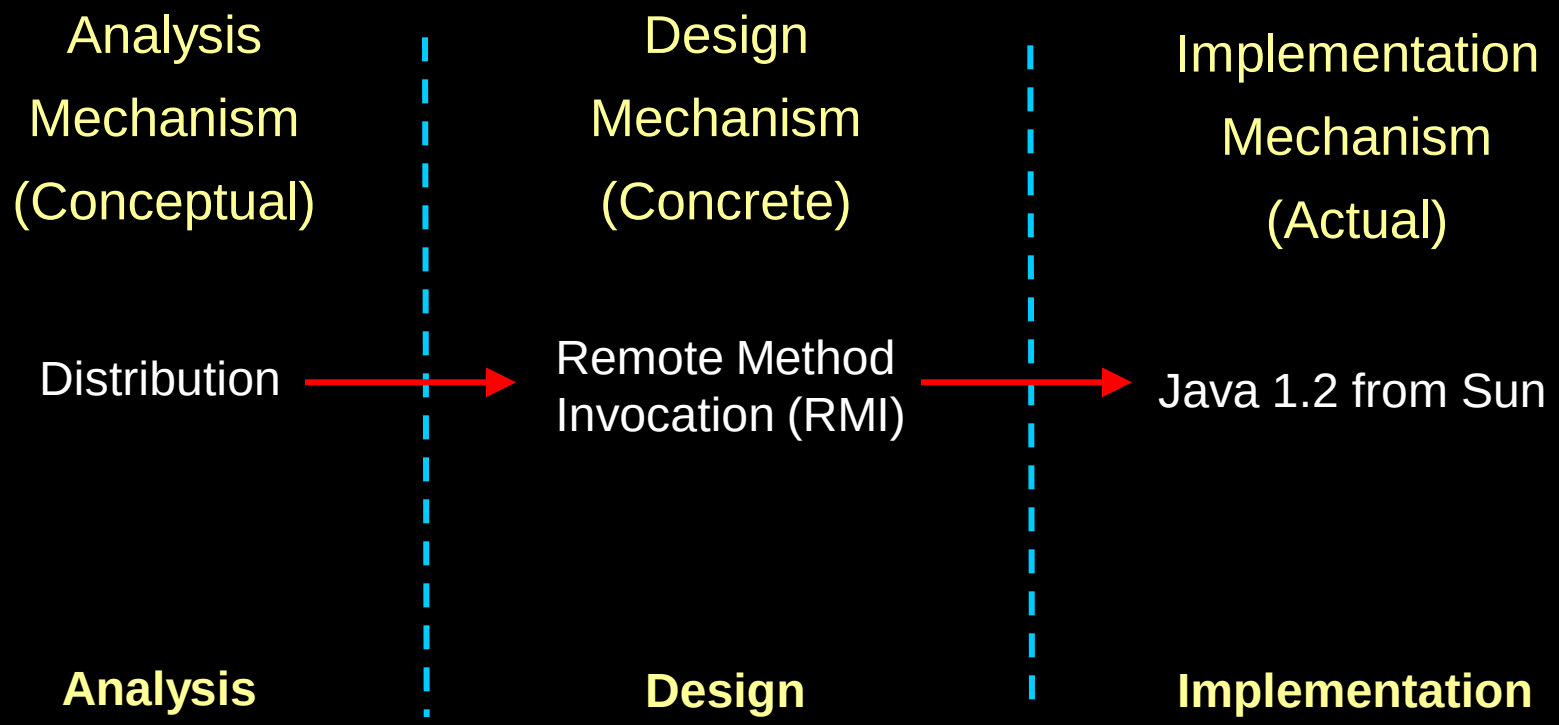


Describe Distribution Steps

- Define the network configuration
- Allocate processes to nodes
- ★ Define the distribution mechanism

Distribution Mechanism

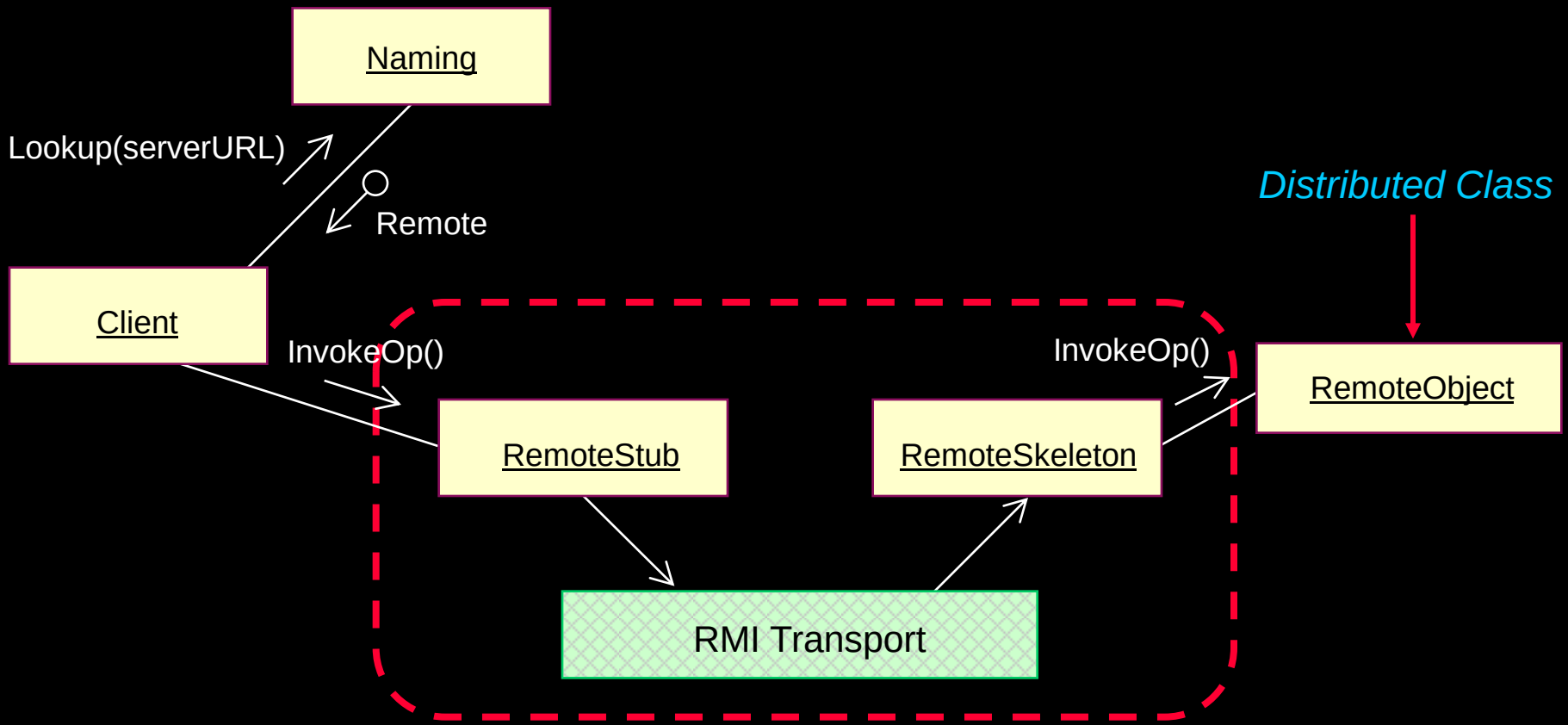
- RMI was chosen as the implementation mechanism for distribution



Design Mechanisms: Distribution: RMI

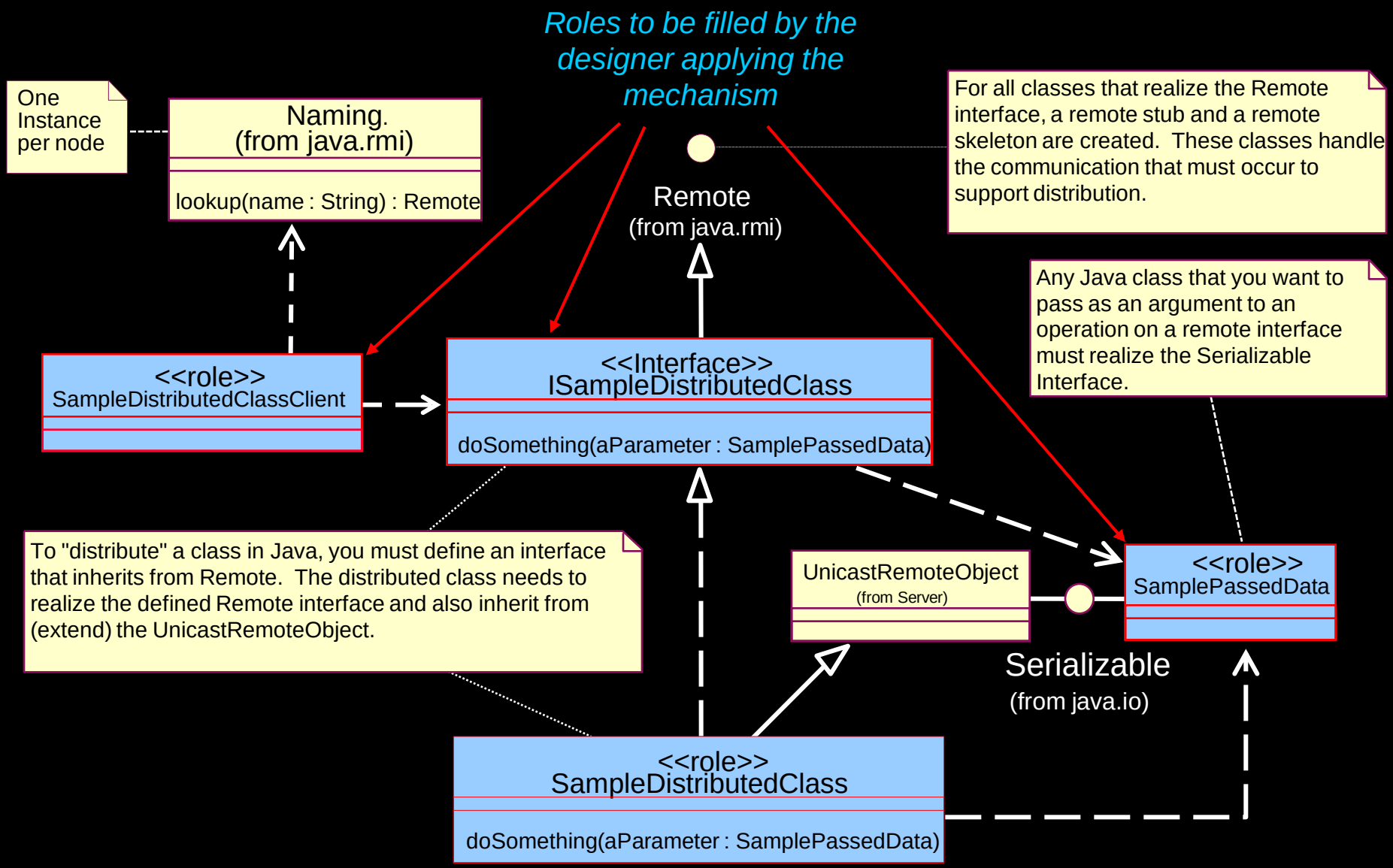
- Distribution characteristics
  - Latency
  - Synchronicity
  - Message Size
  - Protocol

Remote Method Invocation (RMI)



Provided "for free" with RMI for each distributed class

RMI (cont.)


Roles to be filled by the designer applying the mechanism
One Instance per node
Naming.
(from java.rmi)
lookup(name : String) : Remote
Remote
(from java.rmi)
For all classes that realize the Remote interface, a remote stub and a remote skeleton are created. These classes handle the communication that must occur to support distribution.
Any Java class that you want to pass as an argument to an operation on a remote interface must realize the Serializable Interface.
<<role>>
SampleDistributedClassClient
<<Interface>>
ISampleDistributedClass
doSomething(aParameter : SamplePassedData)
To "distribute" a class in Java, you must define an interface that inherits from Remote. The distributed class needs to realize the defined Remote interface and also inherit from (extend) the UnicastRemoteObject.
UnicastRemoteObject
(from Server)
<<role>>
SamplePassedData
Serializable
(from java.io)
<<role>>
SampleDistributedClass
doSomething(aParameter : SamplePassedData)

Incorporating RMI: Steps

- Provide access to RMI support classes (e.g., Remote and Serializable interfaces, Naming Service)
  - *java.rmi and java.io package in Middleware layer*
- For each class to be distributed:
  - *Controllers to be distributed are in the Application layer*
  - *Dependency from the Application to the Middleware layer is needed to access java packages*
  - Define interface for class that realizes Remote
  - Have class inherit from UnicastRemoteObject

Deferred

(continued)

Incorporating RMI: Steps (cont.)

- Have classes for data passed to distributed objects realize the Serializable interface
  - *Core data types are in Business Services layer*
  - *Dependency from Business Services layer to the Middleware layer is needed to access java.rmi*
  - Add the realization relationships } *Deferred*
- Run pre-processor } *Out of scope*

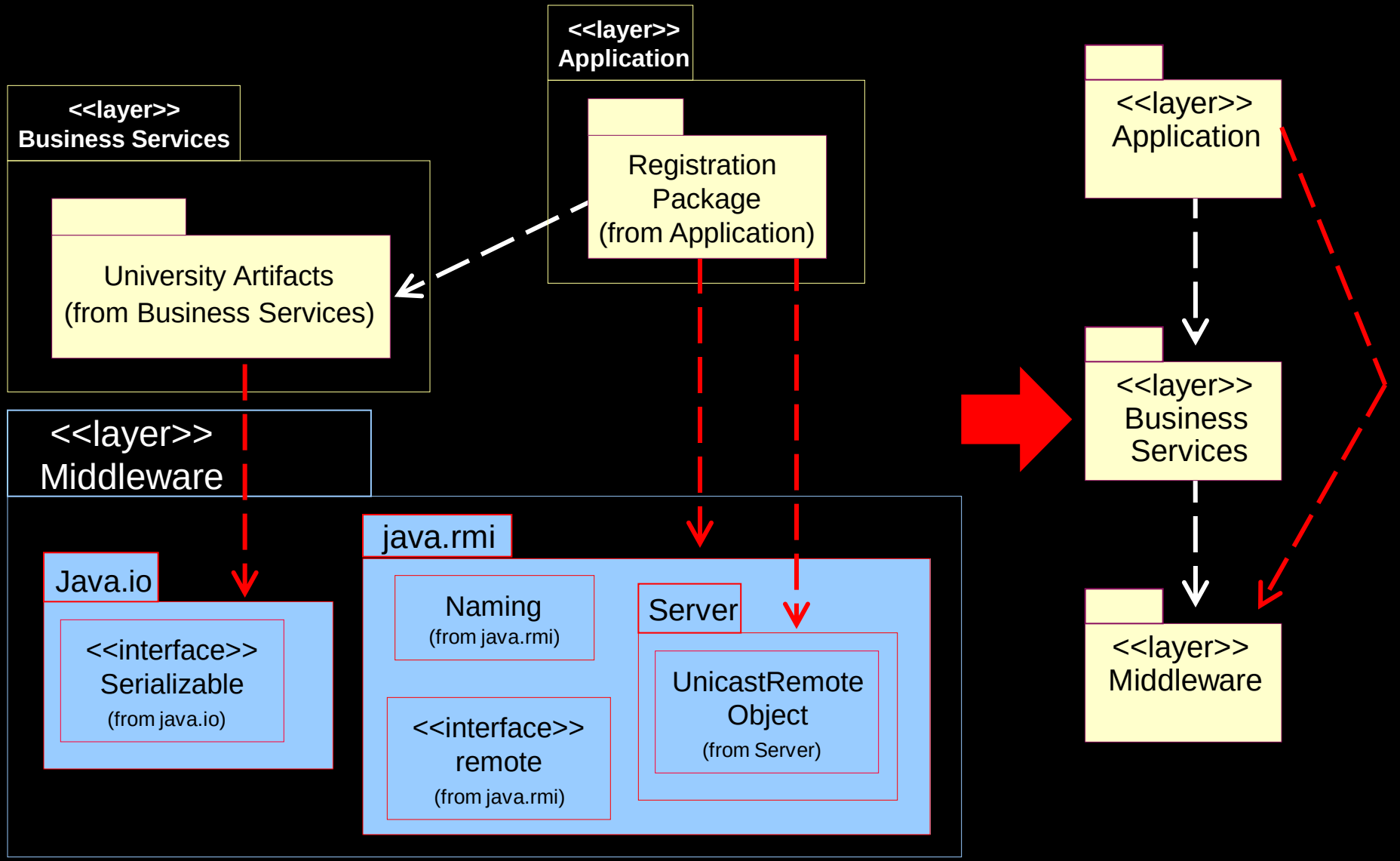
(continued)

Incorporating RMI: Steps (cont.)

- Have distributed class clients lookup the remote objects using the Naming service
  - *Most Distributed Class Clients are forms*
  - *Forms are in the Application layer*
  - *Dependency from the Application layer to the Middleware layer is needed to access java.rmi*
  - Add relationship from Distributed Class Clients to Naming Service
- Create and update interaction diagrams with distribution processing

Deferred

Example: Incorporating RMI



Checkpoints: Deployment View

- Have the distributed data update coordination and synchronization issues been addressed and documented?
- Are services that require more rapid response available locally (LAN vs. WAN)?
- Have all redundant server issues been addressed and documented (primary vs. secondary)?
- Are the failure modes documented?

Review: Describe Distribution

- What is the purpose of the Describe Distribution activity?
- What is a node? Describe the two different “types” of nodes.
- Describe some of the considerations when mapping processes to nodes.
- How do you model the Deployment View? What modeling elements and diagrams are used?

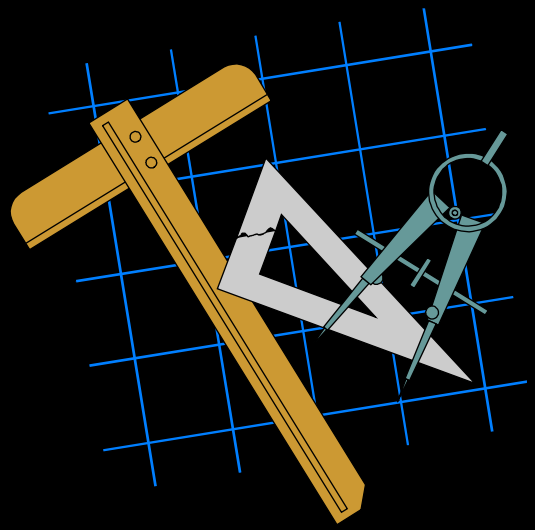
Exercise: Describe Distribution

- Given the following textual information:
  - Network configuration (e.g., nodes and their connections)
    - What processes run on what nodes?

(continued)

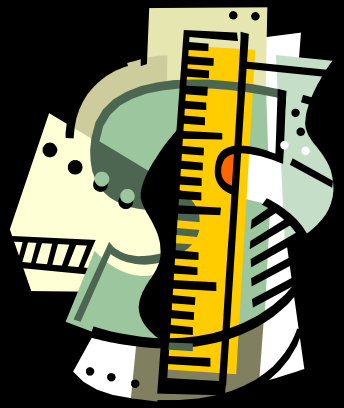
Exercise: Use-Case Analysis

- Produce the following:
  - Deployment diagram depicting:
    - Nodes
    - Connections
    - What processes run on what nodes



Exercise: Review

- Compare your Deployment Model with those developed by the rest of the class.
  - Have nodes and node connections been modeled?
  - Have processes been identified and assigned to nodes? Do the allocations make sense?
  - Are the processes listed beneath the nodes in the Deployment diagram?



Payroll System