AOT LAB

Agent and **O**bject **T**echnology **Lab**
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma

Distributed and Agent Systems

Distributed Systems

Prof. Agostino Poggi

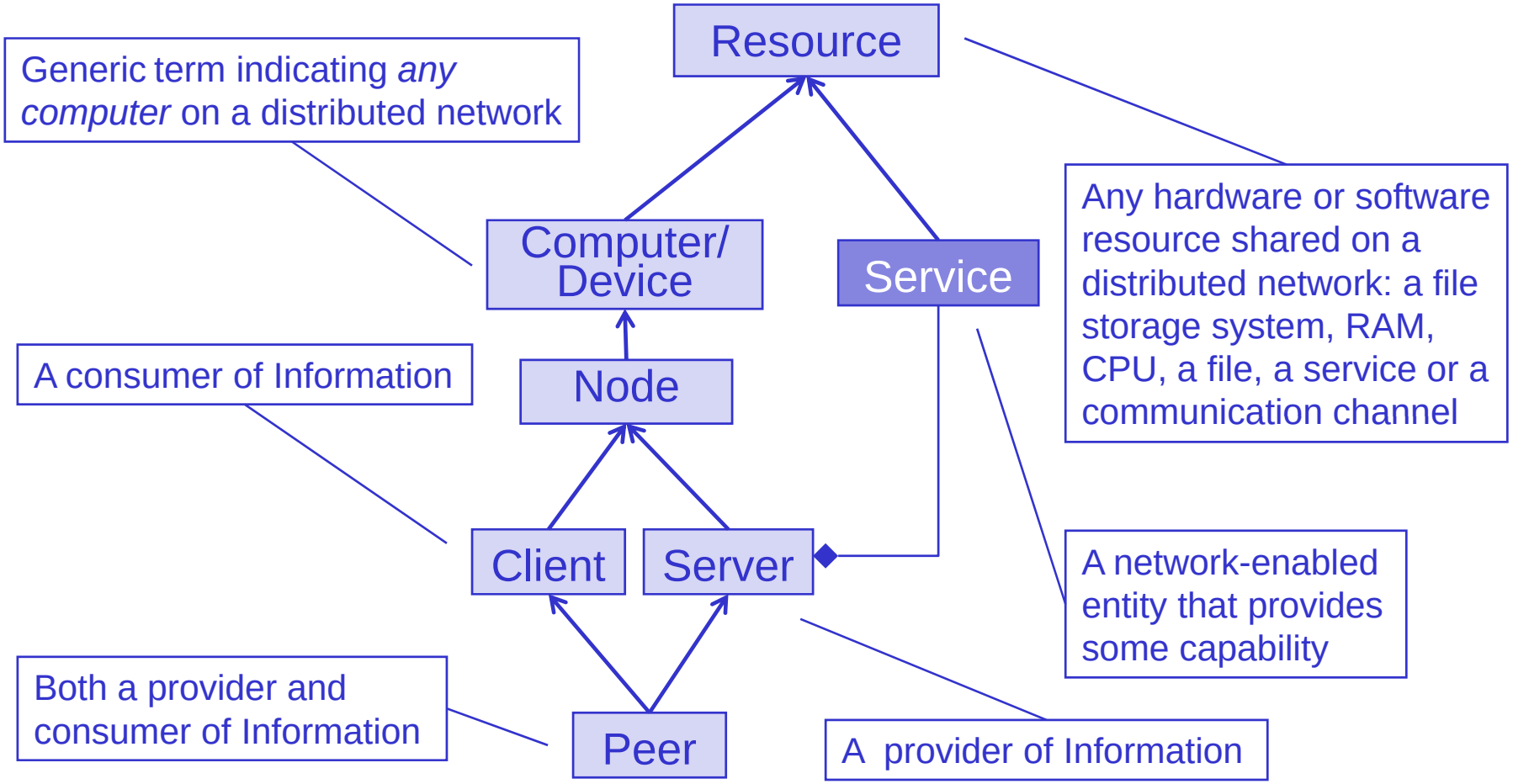
What is a Distributed System?

- A distributed system is a collection of independent computers that appears to its users as a single coherent system

- Relevant features of distributed systems should be:
  - Use of heterogeneous computers
  - Transparent communication
  - Easy to expand and scale
  - Permanently available (even though parts of it are not)

AOT LAB

Terminology



AOT LAB

Centralized Vs. Distributed Systems

Centralized System	Distributed System
Homogeneity	Heterogeneity
Non-autonomous components	Autonomous components
Shared resources	Unshared resources
Single point of control and failure	Concurrency Multiple points of failure

Distributed Systems Reasons

- Scalability
- Distribution of resources
- Advantages of availability through replication
- Often, inherent distribution in the range of applications

Distributed Systems Limits

- Safety risk
- Dependence of the usage and reliability of the network
- Complexity of software

Distributed Computing Issues

- Heterogeneity
  - Networks, hardware, OS, programming language, data representations, …
- Interoperability
  - Middleware, mobile code, protocols, …
- Openness
  - Access, extendibility, …
- Security
  - Confidentiality, integrity, availability, …
- Scalability
  - Data, users, …
- Failure Handling
  - Detecting failures, masking failures, tolerating failures, recovery from failures, redundancy, …
- Concurrency
  - Consistency, causality, mutual exclusion, ...

Goal of Technologies for Distributed Systems

- Distributed systems should be perceived by users and application programmers as a whole rather than as a collection of cooperating components
- Distributed systems have to be seen by developers as a simple abstract machine
- Therefore, the main goal of the technologies for distributed systems is to provide transparency

Transparency Types

- Access transparency
  - A components interacts with another component by requesting execution of a service
  - The interface to a service request is the same, regardless of whether the communication is local or remote
  - Access transparent components can be easily moved from one host to another
- Location transparency
  - Service requesters do not need to know the physical location of components

Transparency Types

- Migration transparency
  - A component can be relocated without users or clients noticing it
  - Information objects can move within a system without affecting the operations
- Replication transparency
  - Users and programs do not know whether a replica or a master provides a service
  - Replicas remain synchronized with their original

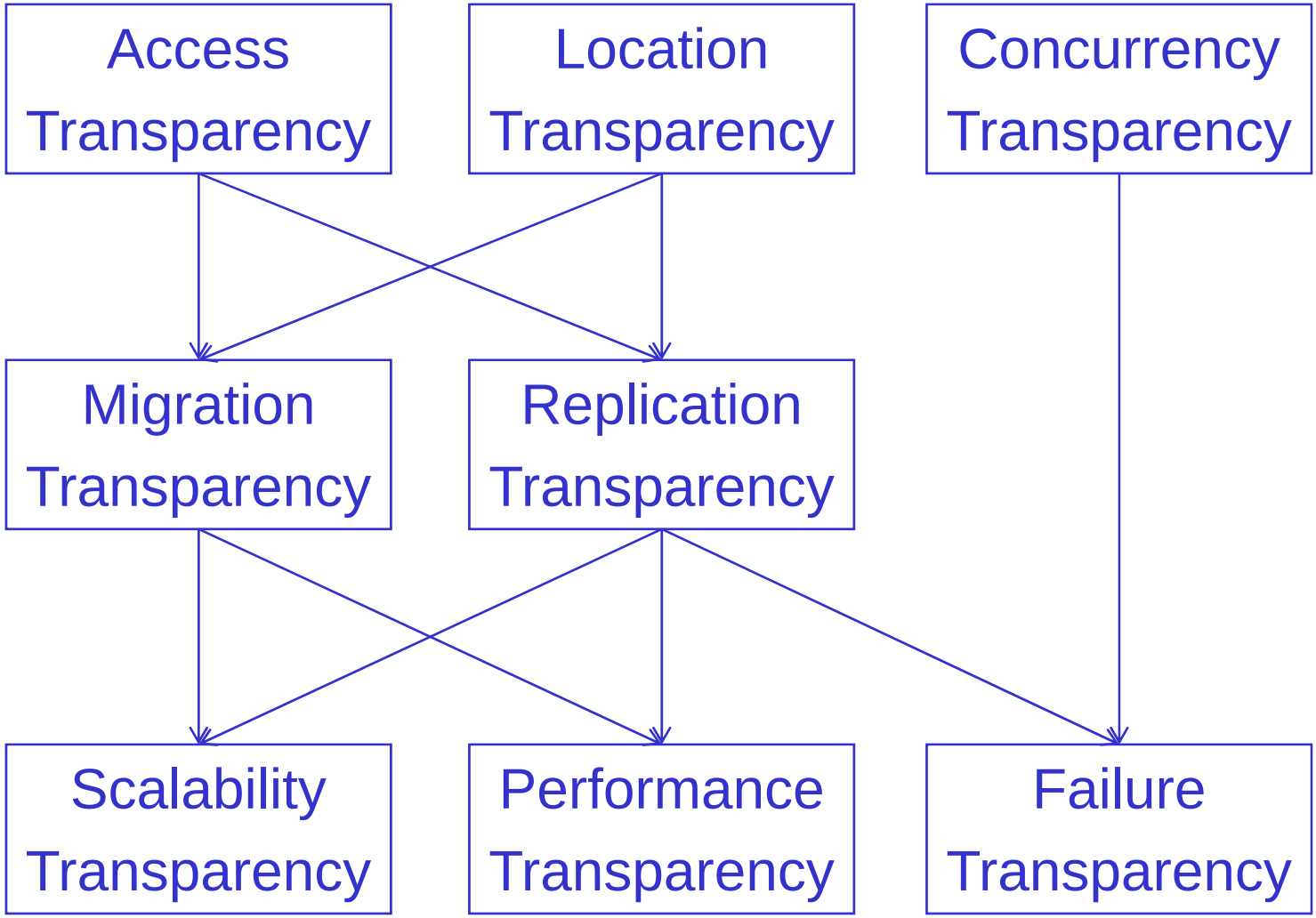
Transparency Types

- Concurrency transparency
  - Users and programmers are unaware that components request services concurrently
  - Enables several processes to use shared information objects without interference between them
    - Do not see what other users do
    - Avoids error-prone programming
- Scalability transparency
  - More components and more concurrent requests can be introduced without change to the system structure or the application algorithms
  - Users and programmers do not know how scalability is achieved

Transparency Types

- Performance transparency
  - Users and programmers are unaware how good system performance is maintained, but from a single request perspective
- Failure transparency
  - Users and programmers are unaware of how the system conceals faults
  - Allows users and applications to complete their tasks despite the failure of other components

Transparency Types



How Is Transparency Reached?

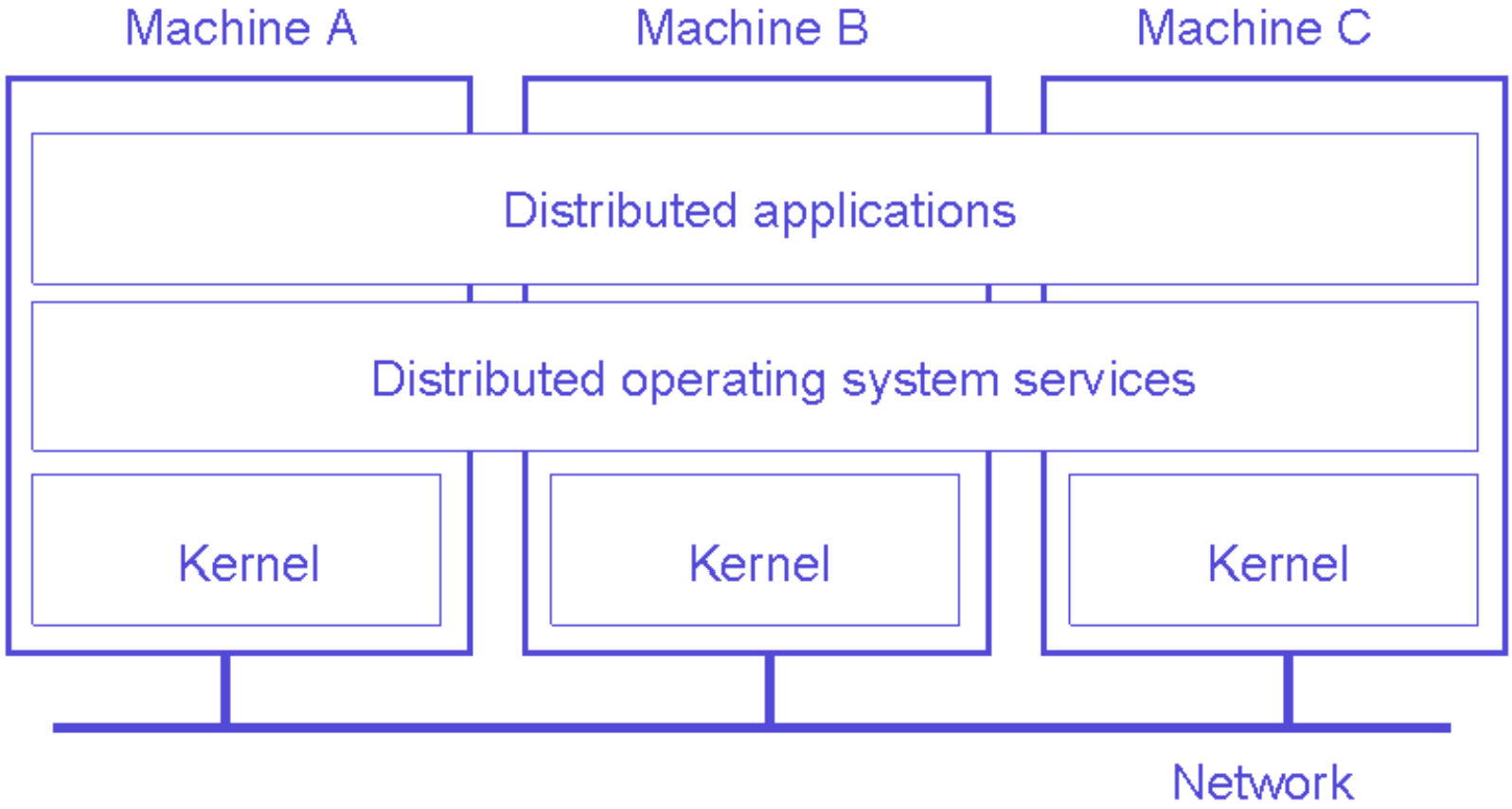
- Distributed Operating Systems
- Standard network protocols
  - TCP/IP (... HTTP...)
- Platform independent languages
  - Java (... C# ... XML …)
- Convenient programming environment
  - Middleware platforms

Transparency Solutions

System	Description	Main Goal
DOS	Tightly-coupled operating system for multi-processors and homogeneous multi-computers	Hide and manage hardware resources
NOS	Loosely-coupled operating system for heterogeneous multi-computers (LAN and WAN)	Offer local services to remote clients
Middleware	Additional layer atop of NOS implementing general-purpose services	Provide distribution transparency

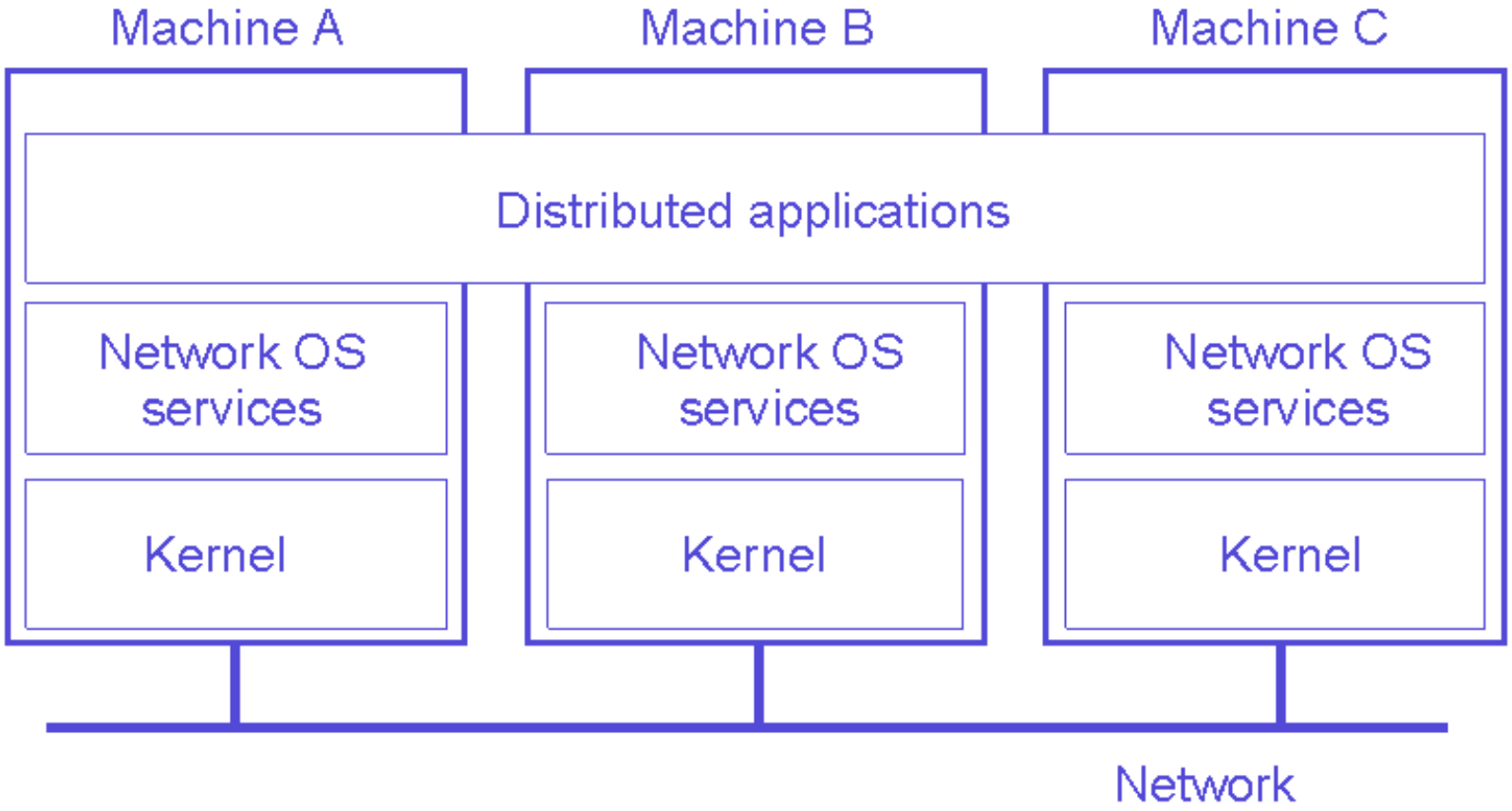
AOT LAB

Distributed Operating Systems

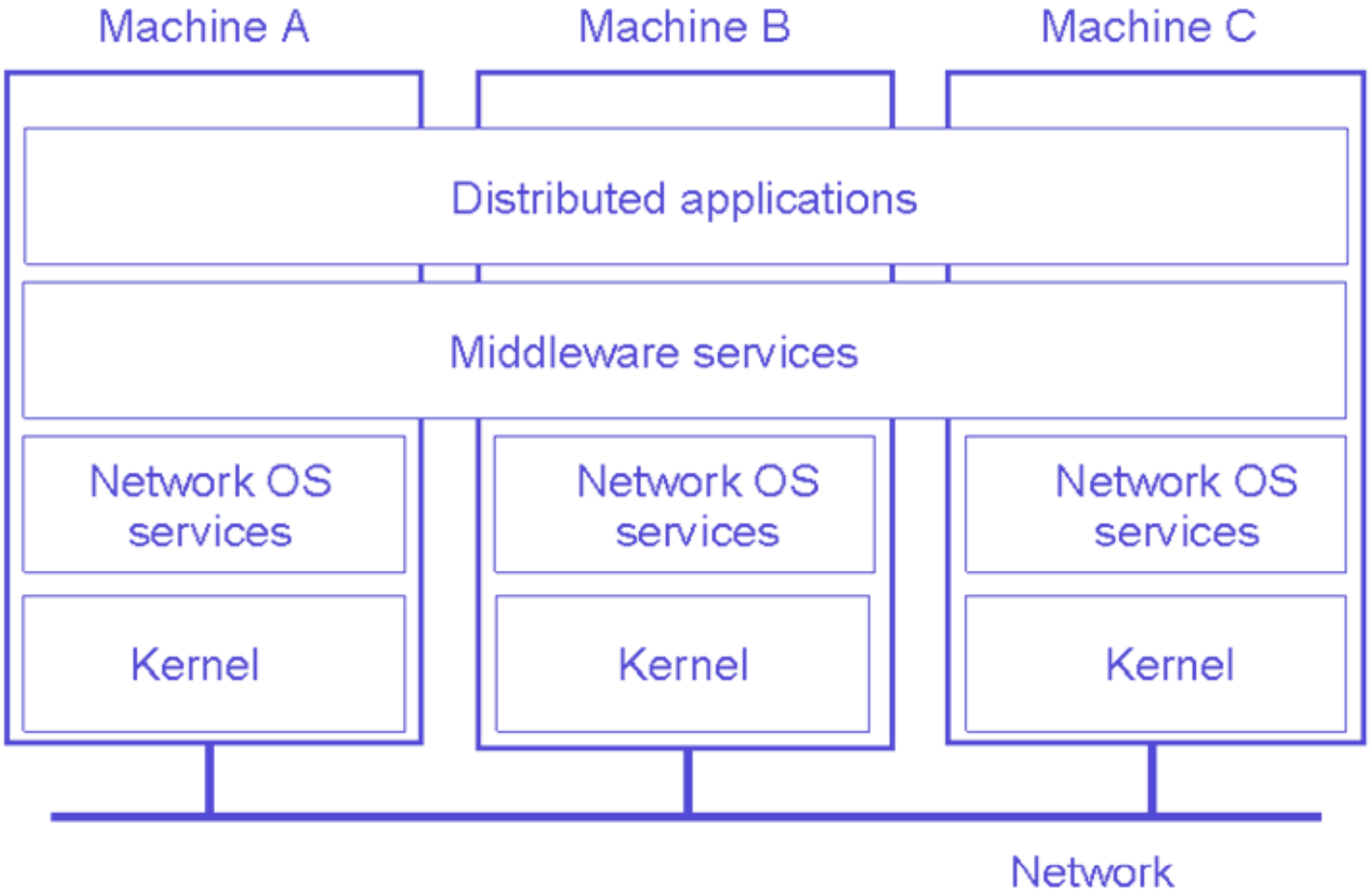


AOT LAB

Network Operating System



Middleware


Machine A
Machine B
Machine C
Distributed applications
Middleware services
Network OS services
Network OS services
Network OS services
Kernel
Kernel
Kernel
Network

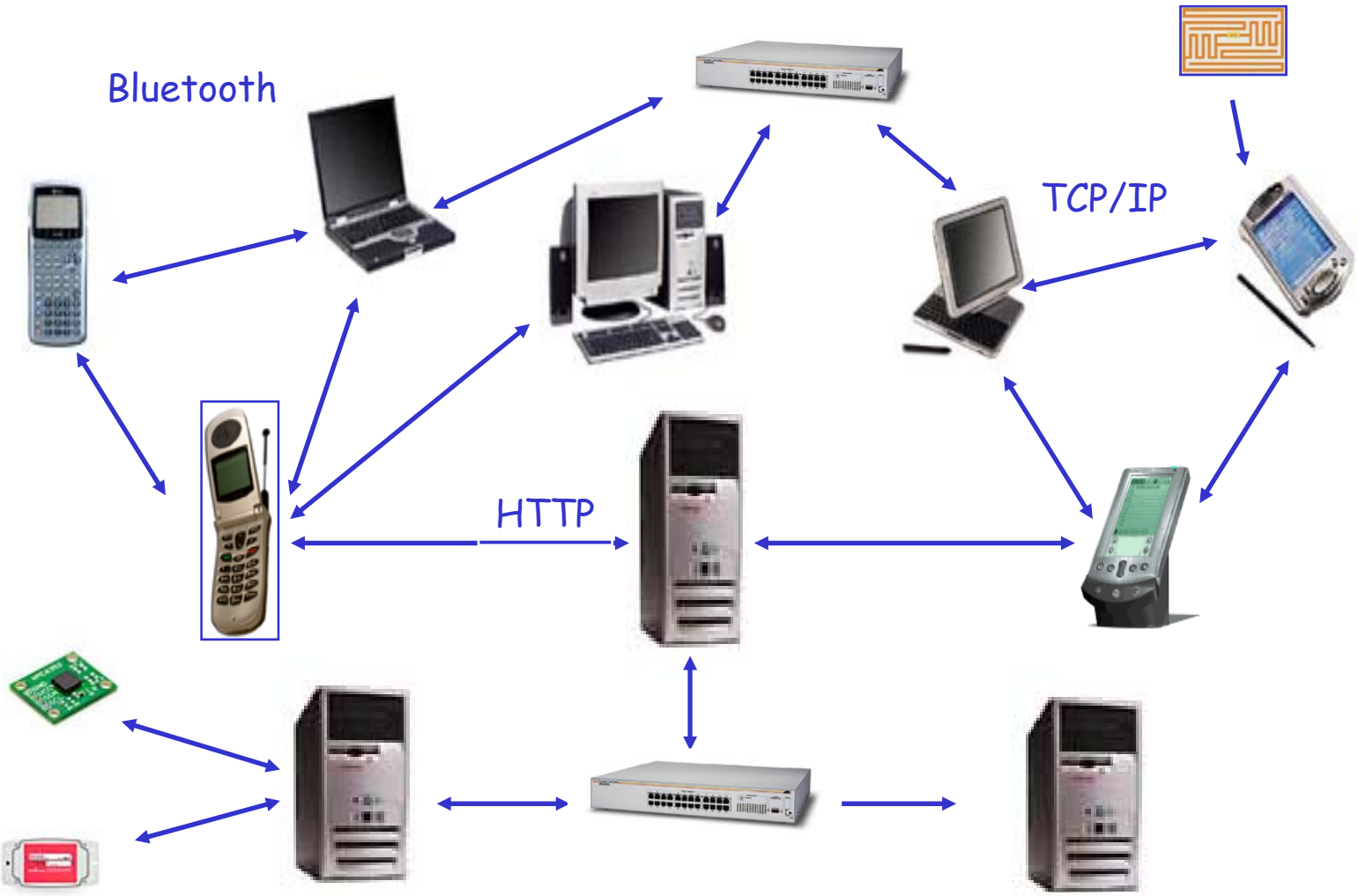
AOT LAB

Solutions Comparison

Feature	DOS		NOS	Middleware
	Multi processors	Multi computers		
Degree of transparency	Very High	High	Low	High
Same OS on nodes	Yes	Yes	No	No
Number of OS copies	1	N	N	N
Basis for communication	Shared memory	Messages	Files	Model specific
Resource management	Global, central	Global, distributed	Per node	Per node
Scalability	No	Moderately	Yes	Varies
Openness	Closed	Closed	Open	Open

AOT LAB

Distributed Systems Evolution



AOT LAB

Forms of Distributing Computing

- Distributed object computing
- Service-oriented computing
- Peer-to-peer computing
- Mobile or nomadic computing
- Pervasive, ubiquitous or invisible computing
- Grid computing
- Autonomic computing
- Cloud computing
- Space based computing

Distributed Object Computing

- Distributed object computing is a computing paradigm that allows objects to be distributed across a heterogeneous network

- In distributed object computing, objects can
  - Be located across a variety of platforms and in different processes
  - Communicate transparently with each other as if they were located on a single machine

Service Oriented Computing

- Service oriented computing is an evolution of object-oriented and component based computing that allows the construction of flexible, dynamic systems
- Systems are assembled from a loosely coupled collection of services, which
  - Have a published interface
  - Can communicate with each other
- Systems are implemented on the basis of widely accepted standards and protocols
  - Based on XML
  - Operate above internet standards

Peer-to-Peer Computing

- Peer-to-peer computing is a model in which each party
  - Has the same capabilities
  - Can initiate a communication session
- Peer-to-peer computing has the goal of simplifying the sharing of resources, including bandwidth, storage space, and computing power, among the parties involved in the peer-to-peer network

Mobile Computing

- Mobile or nomadic computing is the use of portable computing devices and of mobile communications technologies to enable users to access the Internet and data on their home or work computers from anywhere in the world
- Mobile computing mainly copes with the problems of network connectivity and device constraints
- Mobile computing copes with the problems of physical, logical and personal mobility

Pervasive Computing

- Pervasive, ubiquitous or invisible computing has the goal of enhancing everyday objects and makes them smarter without requiring extra human interaction
- Pervasive computing devices are not personal computers as we tend to think of them, but very tiny - even invisible - devices, either mobile or embedded in almost any type of object imaginable, including cars, tools, appliances, clothing and various consumer goods - all communicating through increasingly interconnected networks

Grid Computing

- Grid computing has the goal of organizing the resources of many computers in a network to be use to solve a single problem

- Grid computing is usually applied to scientific or technical problems that require a great number of computer processing cycles or access to large amounts of data

Autonomic Computing

- Autonomic computing has the goal of creating computer systems capable of self-management, to overcome the rapidly growing complexity of computing systems management, and to reduce the barrier that that complexity poses to further growth

- Autonomy computation may use artificial systems imitating collective behaviors of social animals to solve hard computational problems

Cloud Computing

- Cloud computing is Internet (cloud) based development and use of computer technology (computing)
- It is a style of computing in which dynamically scalable and often virtualized resources are provided as a service over the Internet
- Users need not have knowledge of, expertise in, or control over the technology infrastructure "in the cloud" that supports them
- The Cloud appears as a single point of access for all the computing needs of consumers

Space Based Computing

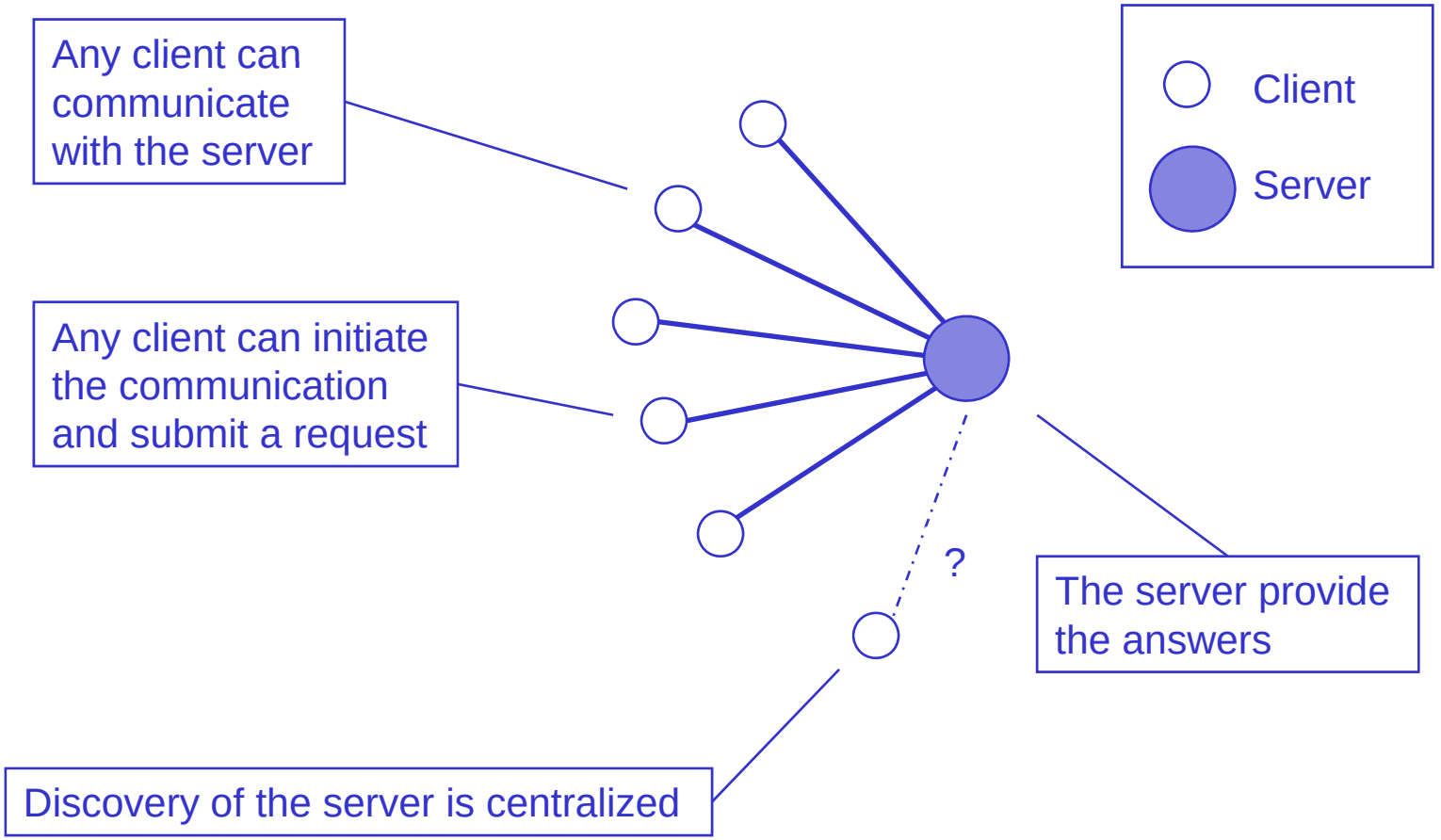
- Space based computing has the goal of coordinating applications through a virtual shared memory
- Space based computing decouples interaction in three dimensions
  - Time: applications can read and write data whenever they want to
  - Space: applications only need to access the same space in order to communicate
  - Reference: applications communicating with each other do not need to know explicitly from each other

Architectural Models

- Client-Server Model
- Peer-to-Peer Model
- Message Oriented Models
- Mobile Models
- Space Based Model

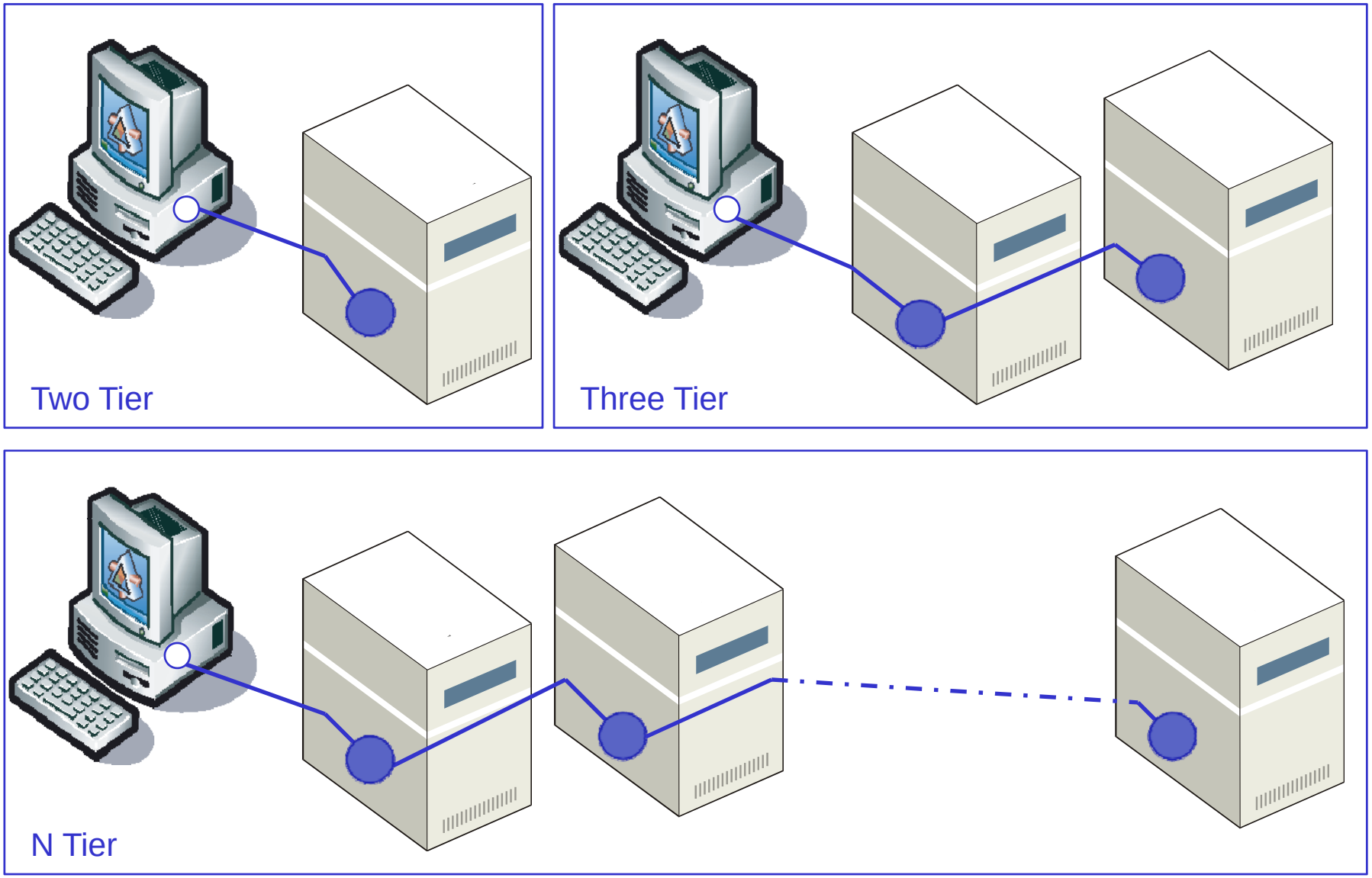
AOT LAB

Client-Server Model



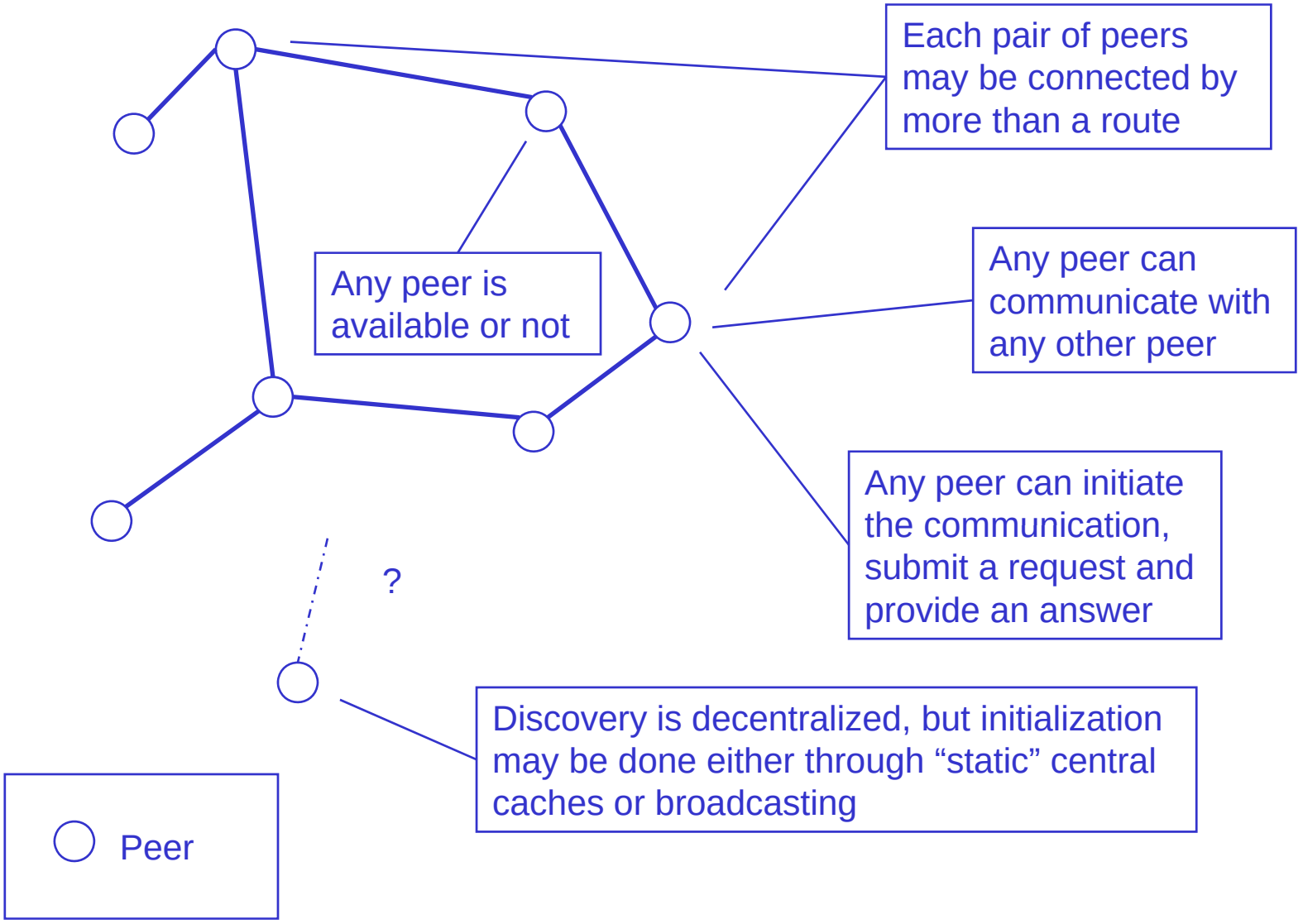
AOT LAB

Two, Three, ... , N Tier System

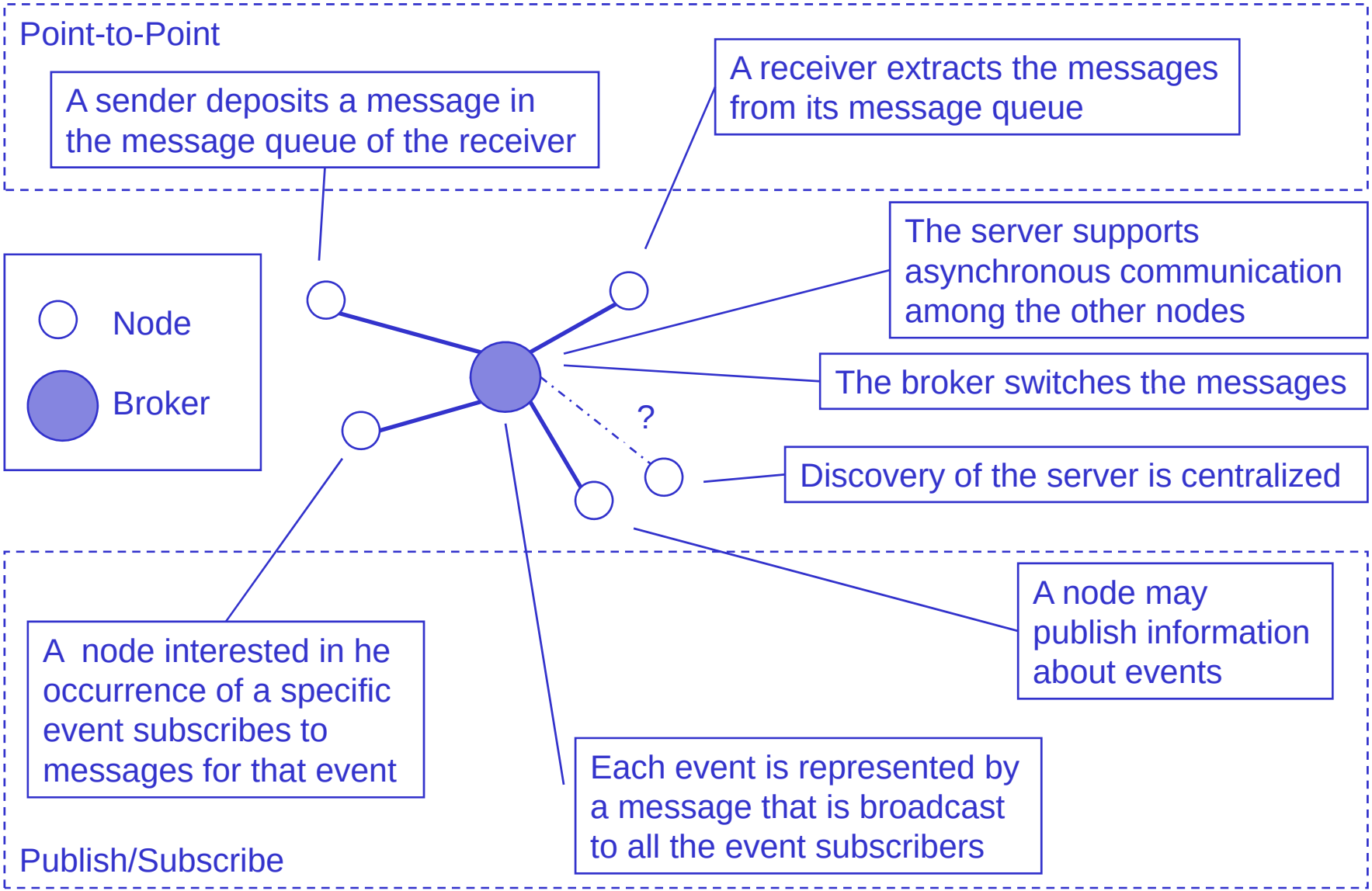


AOT LAB

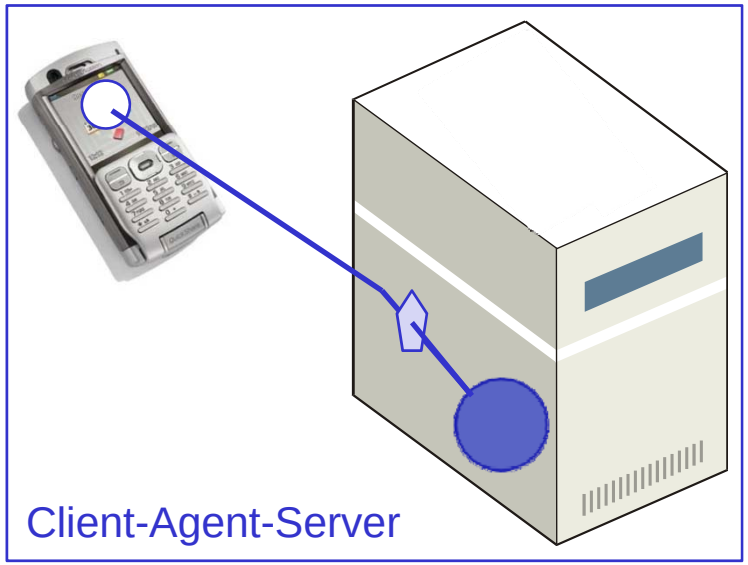
Peer-To-Peer Model

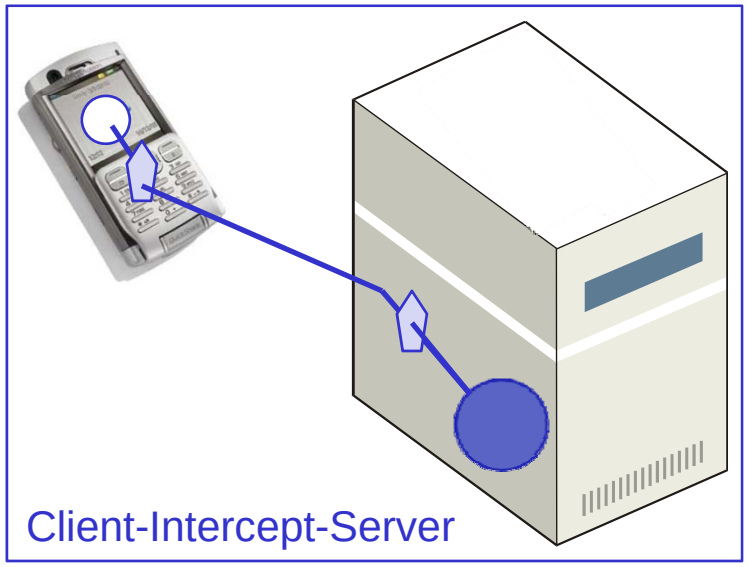


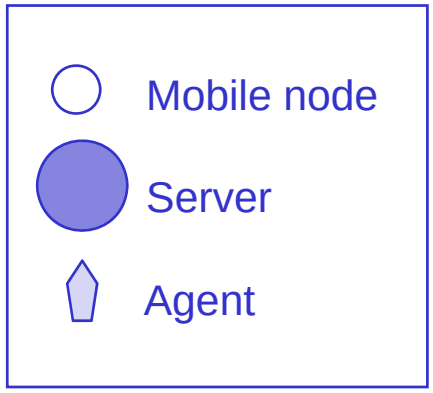
Message Oriented Models

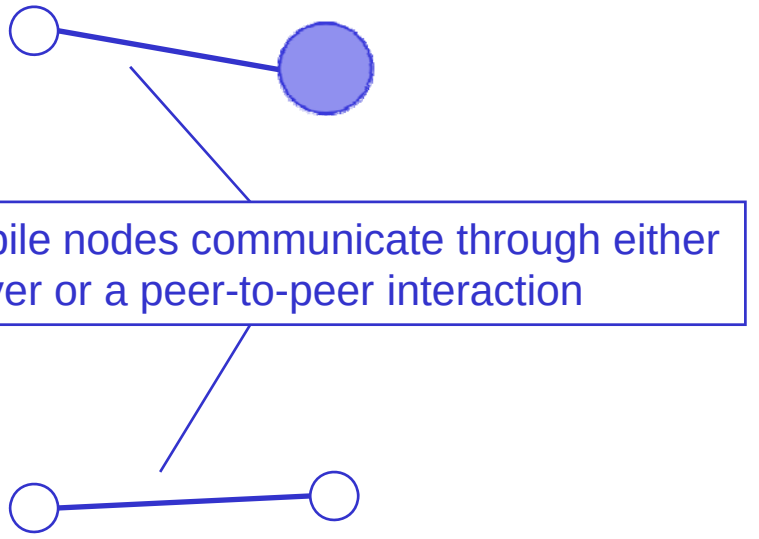


Mobile Oriented Models

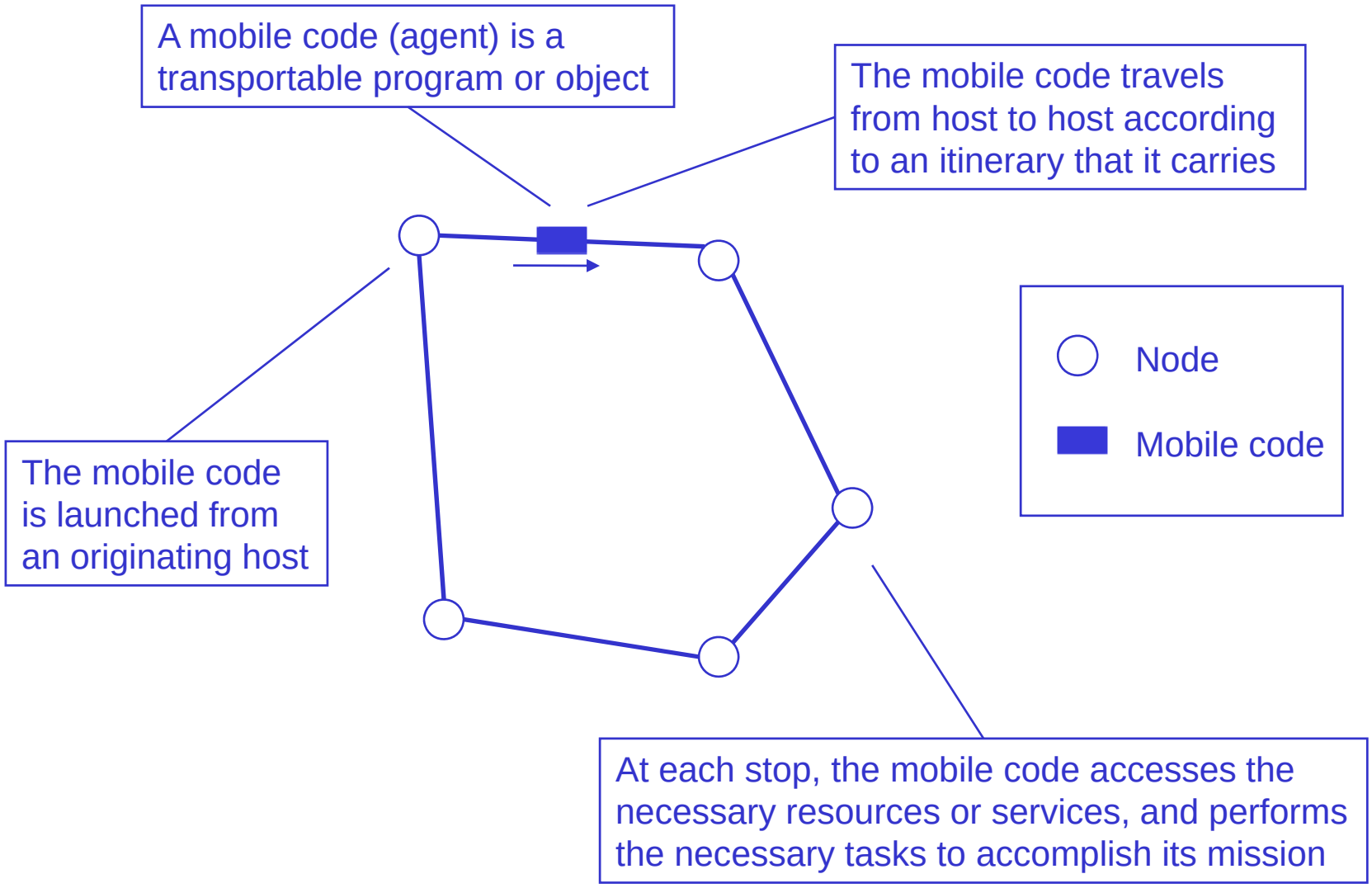




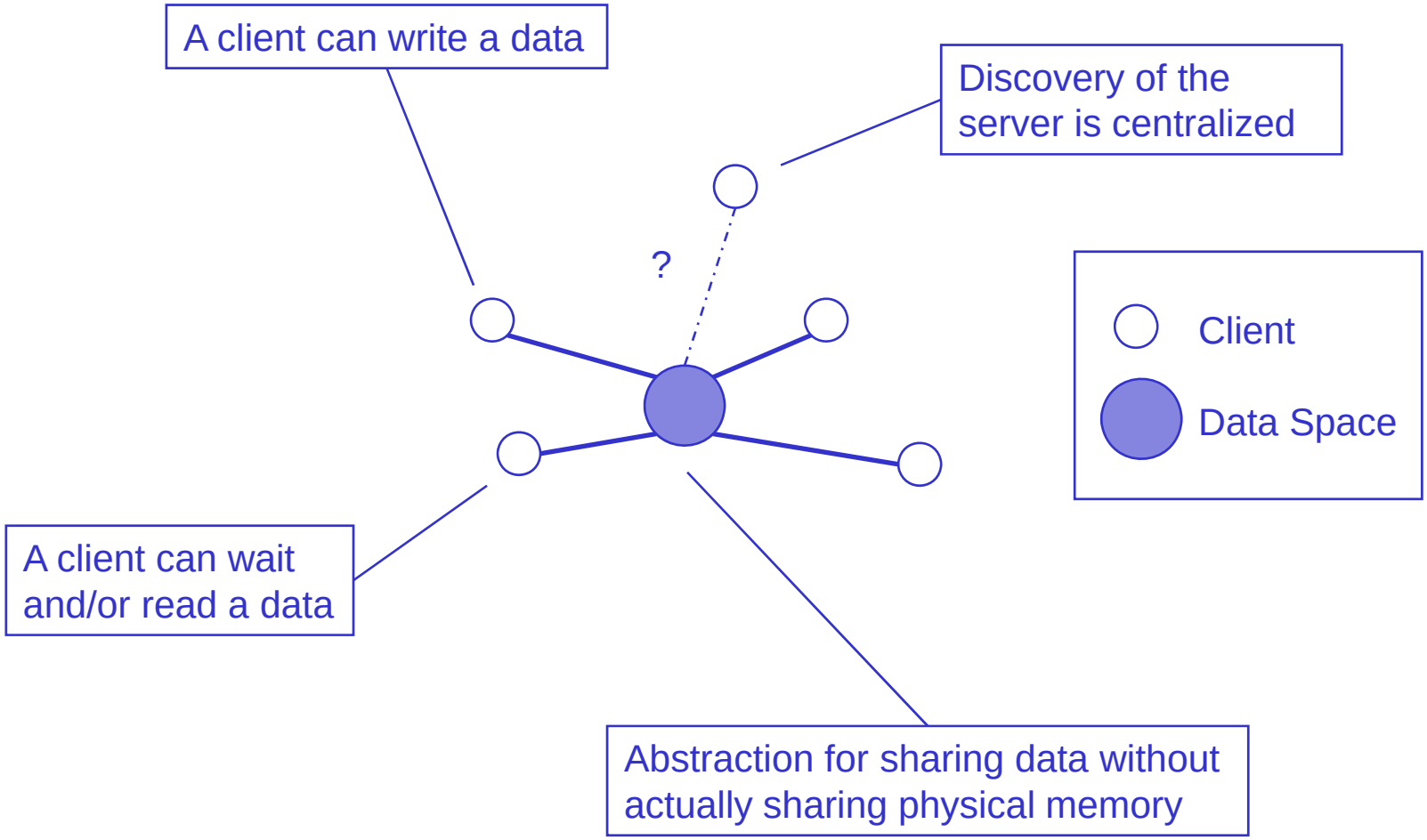




Mobile Oriented Models



Space-Based Model



Interoperability

- According to the Oxford Dictionary
  - Interoperable means able to operate in conjunction
  - Interoperate implies that one system performs an operation on behalf of another system

Interoperability

- From a software engineering point of view
  - Interoperability means that two co-operating software systems can easily work together without a particular interfacing effort
  - It also means establishing communication and sharing information and services between software applications regardless of hardware platforms
  - Interoperability is the ability to integrate data, functionality and processes with respect to their semantics
  - From a technology point of view, interoperability needs the integration among operating systems, programming language, network protocol and data representation

AOT LAB

Interoperability Levels

The assumptions and constraints of the meaningful abstraction of reality are aligned

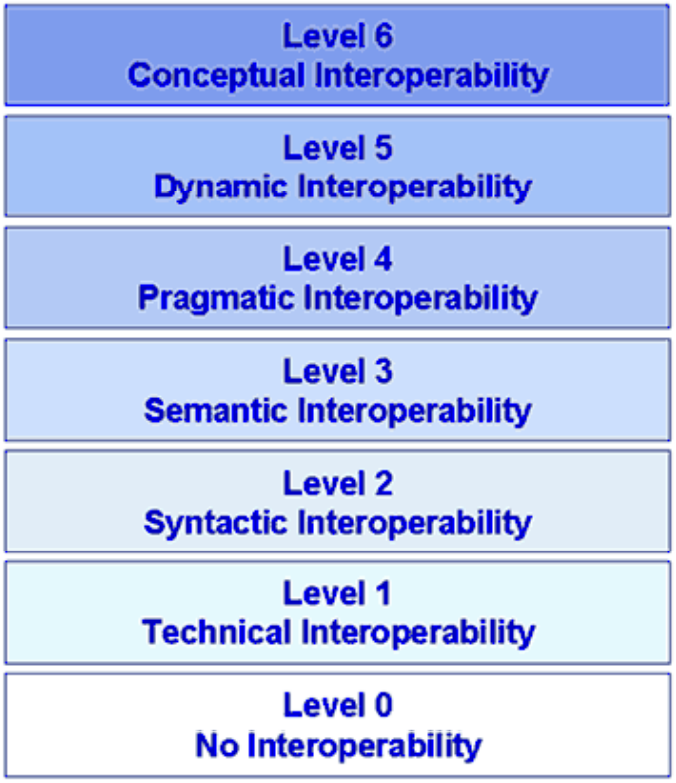
The state changes, that occur in the assumptions and constraints that each system is making over time, are comprehended

The use of the data or the context of its application is understood

A common information exchange reference model is used, i.e., the meaning of the data is shared

A common data format is applied

A communication protocol exists for exchanging data between participating systems

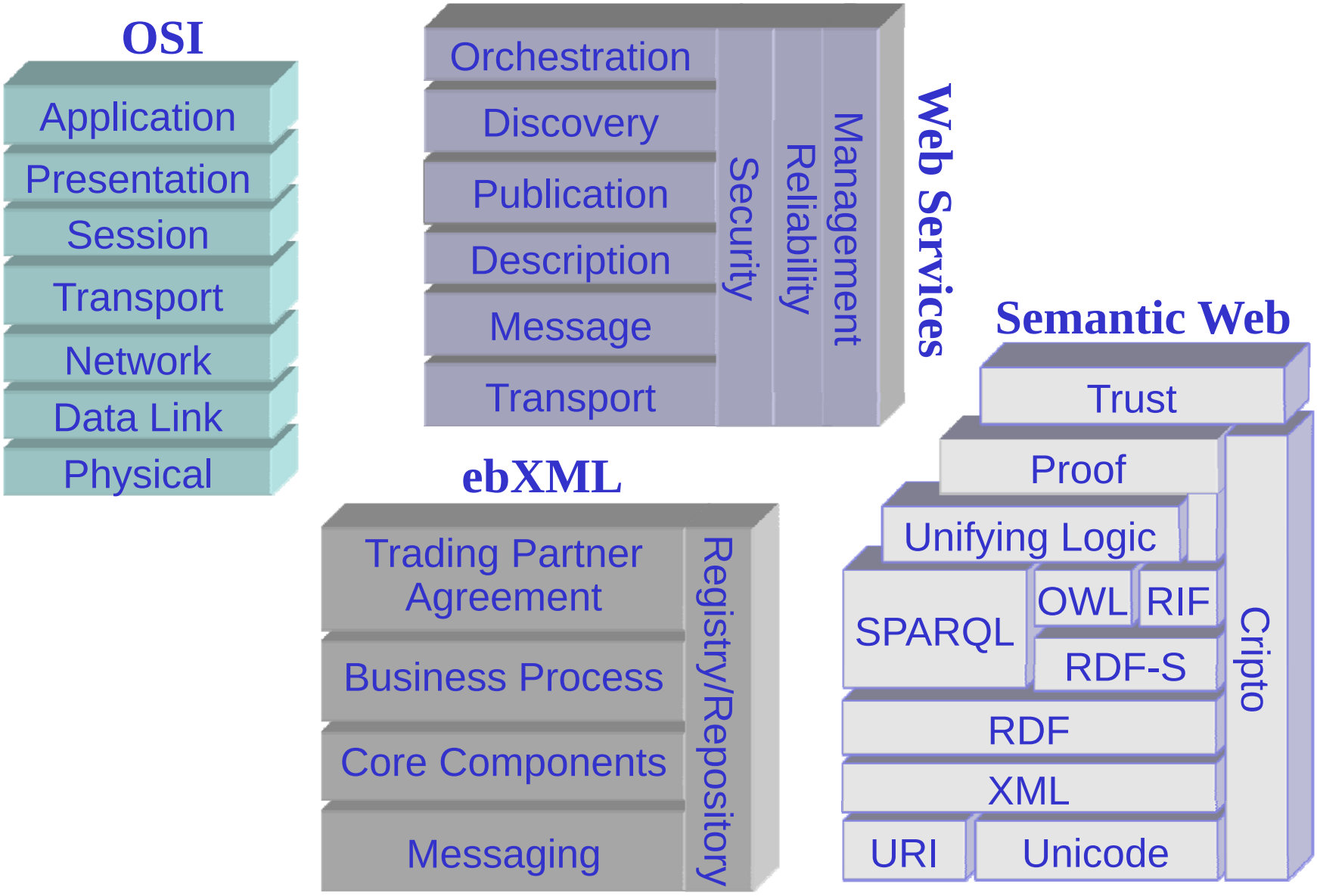


Interoperability Achievement

- Data level
  - Exchanged data are correctly interpreted
- Application level
  - Entities can exchange functionality and interpretable data
- Business process level
  - Business process can be automatically realized utilizing essential human labor only
  - Typically, It happens when a process conforms to standards that enable it to achieve its objective regardless of ownership, location, make, version or design of the computer systems used

AOT LAB

Standards for Interoperability



Interoperability Challenges

- Development of systems on the basis of interoperable components
  - Component based development
- Integration of existing (legacy) systems in the context of business to business and enterprise integration
  - Service oriented development

Integration Types

- Static Integration
  - Web services API
  - Workflow technologies (e.g., WS-BPEL)
  - New choreography languages (e.g., WSCDL)
- Dynamic integration
  - Planners
  - Agents