AOT LAB

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

Distributed and Agent Systems

Peer-to-Peer Systems & JXTA

Prof. Agostino Poggi

AOT LAB

What is Peer-to-Peer Computing?

- A class of decentralized, self-organizing distributed systems, in which all or most communication is symmetric
- A model of communication where every node in the network acts alike as opposed to the client-server model, where one node provides services and other nodes use the services
- A class of applications that take advantage of resources (e.g., storage, cycles, content) available at the edge of the Internet

Advantages

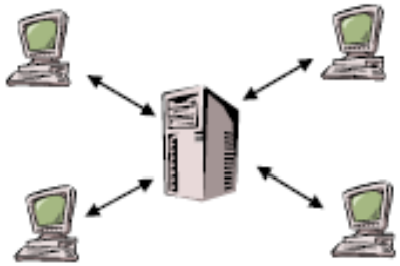
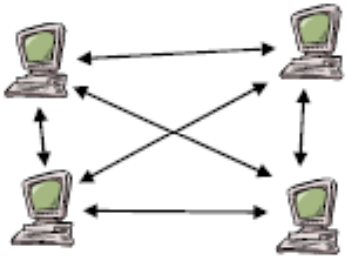
- No central point of failure
  - The Internet and the Web do not have a central point of failure
  - Most internet and web services use the client-server model (e.g. HTTP), so a specific service has a central point of failure
- Scalability
  - Since every peer is alike, it is possible to add more peers to the system and scale to larger networks

Disadvantages

- Decentralized coordination
  - How to keep global state consistent?
  - Need for distributed coherency protocols
- All nodes are not created equal
  - Computing power, bandwidth have an impact on overall performance
- Programmability
  - As a corollary of decentralized coordination

AOT LAB

Peer-to-Peer Vs. Client-Server

	Benefits	Drawbacks
Client-to-Server 	❑ Trust / Security ❑ Manageability ❑ Consistency	❑ Single Point of Failure ❑ Scalability ❑ Costs
Peer-to-Peer 	❑ Exensibility / Scalability ❑ Fault tolerance ❑ Resistance to lawsuits	❑ Missing incentives ❑ Maliciousness ❑ Overhead

AOT LAB

What are Peer-to-Peer Challenges?

- Peer identification
- Routing protocols
- Network topologies
- Peer discovery
- Communication/coordination protocols
- Quality of service
- Security
- Fine-grained resource management

Peer-to-Peer Applications

- File sharing (Gnutella, Kazaa, Napster)
- Content streaming (SplitStream, CoopNet, PeerCast)
- Backup storage (HiveNet, OceanStore)
- Collaborative environments (Groove Networks)
- Web serving communities (uServ)
- Instant messaging (Yahoo, AOL)
- Anonymous email
- Censorship-resistant publishing (Ethernity, Freenet)
- Spam filtering

Peer-to-Peer Computing

- Peers
  - End-systems (typically, user machines)
  - Interconnected through an overlay network
  - Are similar or behave in similar manner
- Cooperate
  - Share resources
    - Data, CPU cycles, storage, bandwidth
  - Participate in protocols
    - Routing, replication, …
- Functions
  - File-sharing, distributed computing, communications, content distribution, …

Peers

- Quite heterogeneous
  - Several order of magnitudes difference in resources
  - Compare the bandwidth of a dial-up peer versus a high-speed LAN peer
- Unreliable
  - Failure is the norm!
- Offer limited capacity
  - Load sharing and balancing are critical

Peer-to-Peer Mechanisms

- To obtain a resilient system:
  - Integrate multiple components with uncorrelated failures
  - Use data and service replication
- To improve delivered QoS:
  - Move service delivery closer to consumer
  - Integrate multiple providers
- To generate meaningful statistics, to detect anomalies:
  - Provide views from multiple vantage points
- To improve scalability:
  - Use decentralized (local) control, unmediated interactions

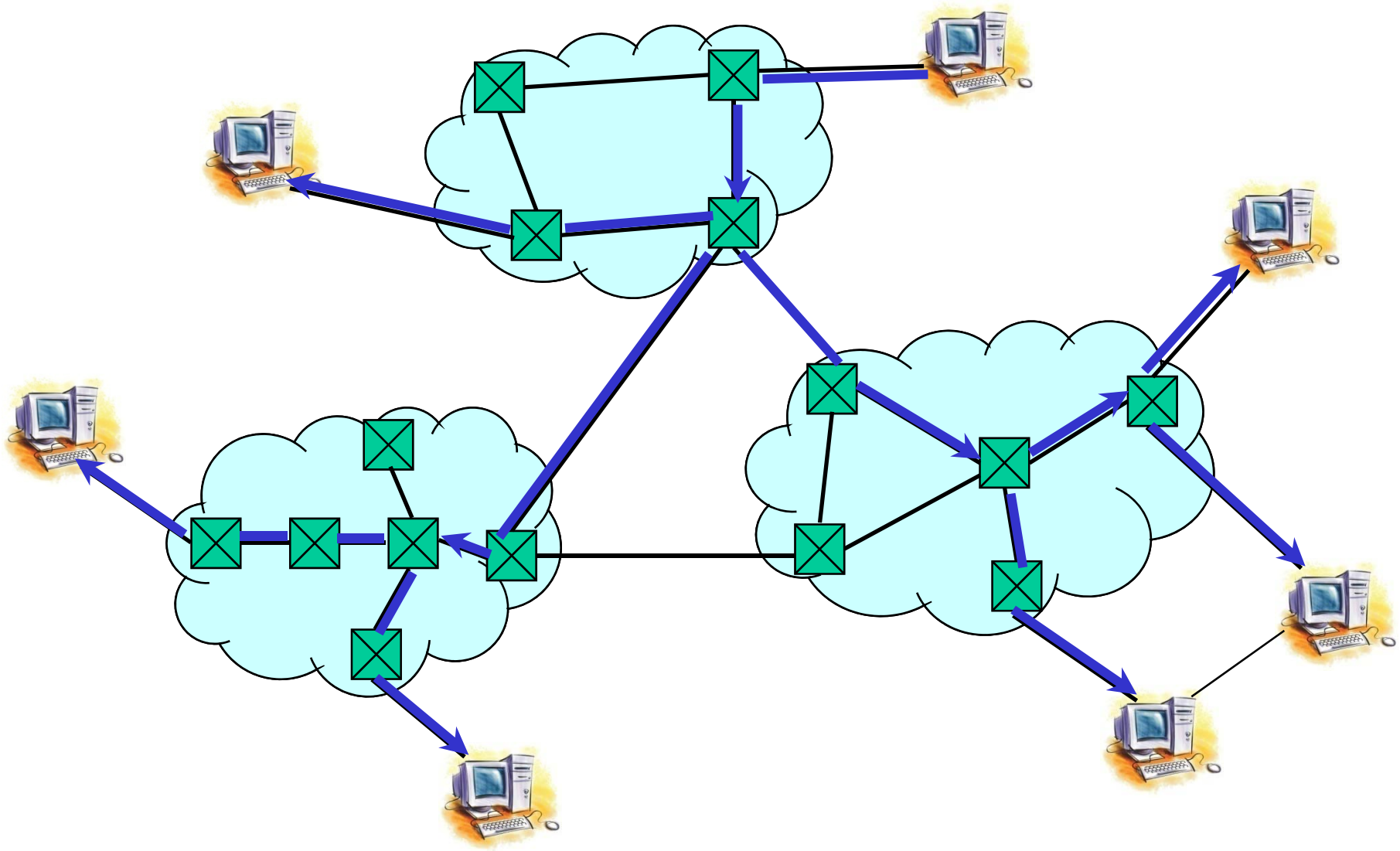
Network

- Scale
  - Numerous number of peers (millions)
- Structure and topology
  - Ad-hoc: No control over peer joining/leaving
  - Highly dynamic
- Membership/participation
  - Typically open
- More security concerns
  - Trust, privacy, data integrity, …
- Cost of building and running
  - Small fraction of same-scale centralized systems

Overlay Network

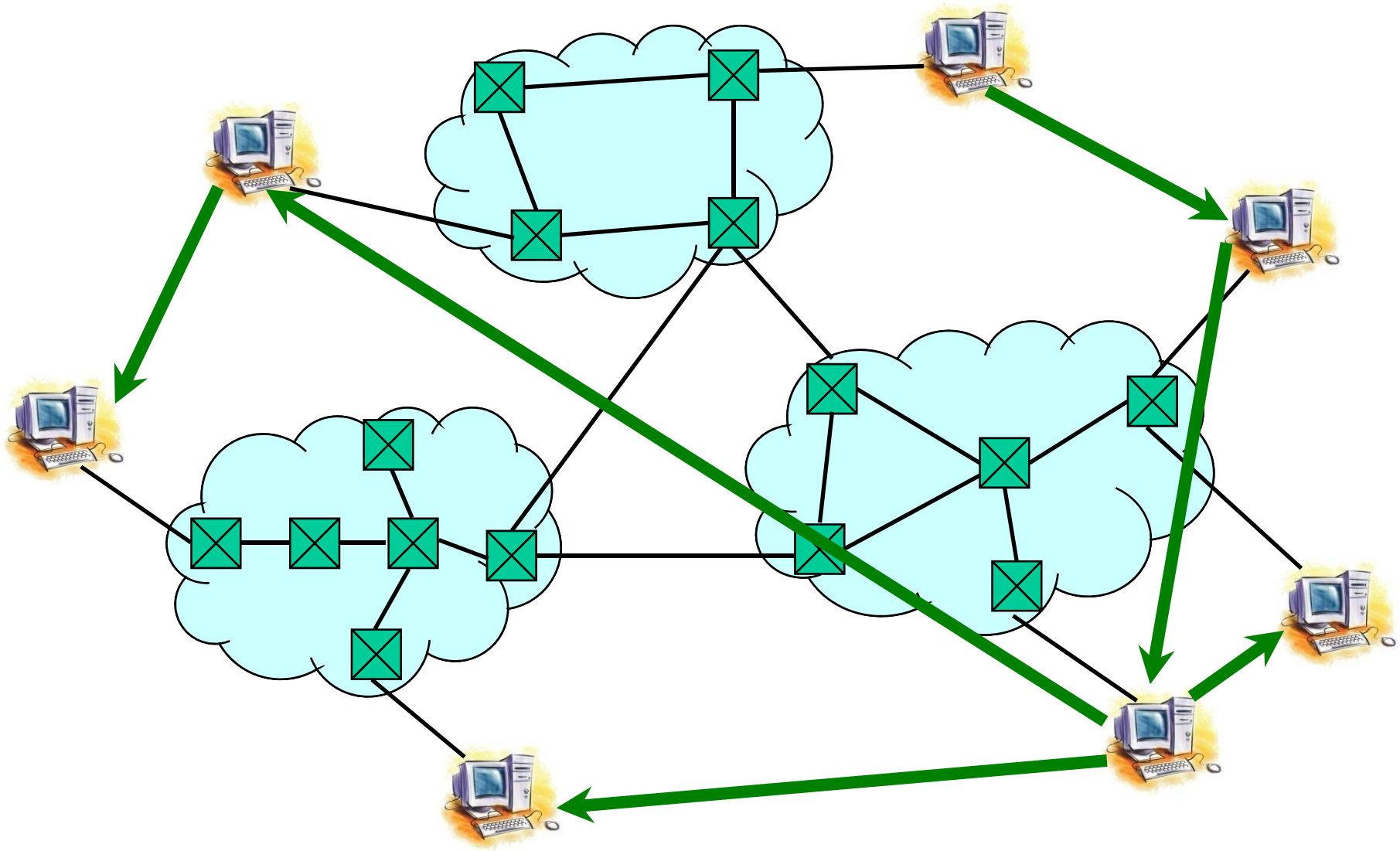
- An abstract layer built on top of the physical network
- Neighbors in the overlay can be several hops away in the physical network
- An overlay network offers flexibility in:
  - Choosing neighbors
  - Forming and customizing topology to fit application needs (e.g., short delay, reliability, high bandwidth, …)
  - Designing communication protocols among nodes
  - Get around limitations in legacy networks
  - Enable new network services

IP Multicast

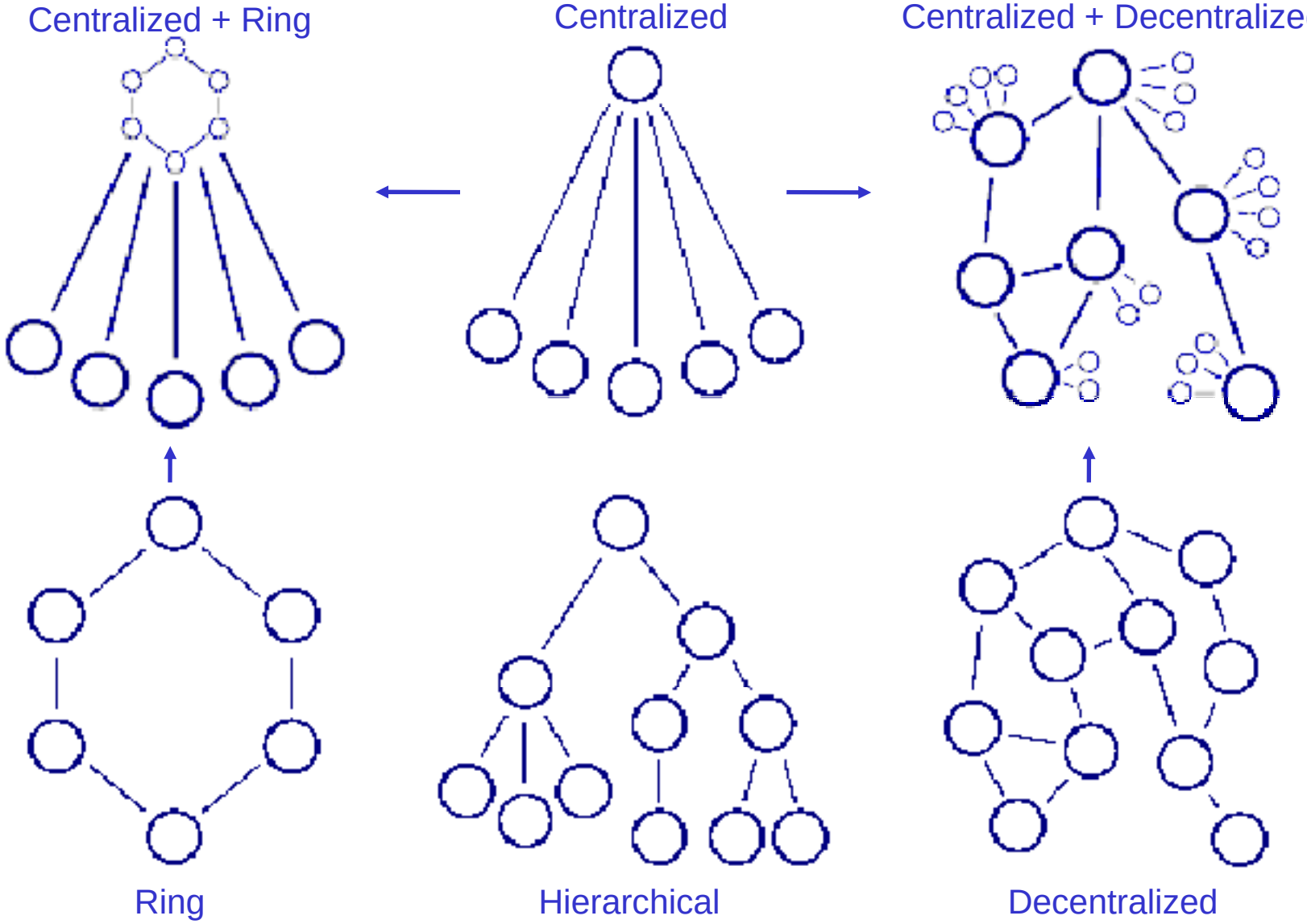


AOT LAB

Application Level Multicast



Topology


Centralized + Ring
Centralized
Centralized + Decentralized
Ring
Hierarchical
Decentralized

AOT LAB

Topology Evaluation

Manageable	How hard is it to keep working?
Coherent	How authoritative is info?
Extensible	How easy is it to grow?
Fault Tolerant	How well can it handle failures?
Secure	How hard is it to subvert?
Lawsuit-proof	How hard is it to shut down?
Scalable	How big can it grow?

Centralized

Manageable	Yes	System is all in one place
Coherent	Yes	All information is in one place
Extensible	No	No one can add on to system
Fault Tolerant	No	Single point of failure
Secure	Yes	Simply secure one host
Lawsuit-proof	No	Easy to shut down
Scalable	?	One machine

Ring

Manageable	Yes	Simple rules for relationships
Coherent	Yes	Easy logic for state
Extensible	No	Only ring owner can add
Fault Tolerant	Yes	Fail-over to next host
Secure	Yes	As long as ring has one owner
Lawsuit-proof	No	Shut down owner
Scalable	Yes	Just add more hosts

Hierarchical

Manageable	½	Chain of authority
Coherent	½	Cache consistency
Extensible	½	Add more leaves, rebalance
Fault Tolerant	½	Root is vulnerable
Secure	No	Too easy to spoof links
Lawsuit-proof	No	Just shut down the root
Scalable	Yes	Hugely scalable

Decentralized

Manageable	No	Very difficult, many owners
Coherent	No	Difficult, unreliable peers
Extensible	Yes	Anyone can join in!
Fault Tolerant	Yes	Redundancy
Secure	No	Difficult, open research
Lawsuit-proof	Yes	No one to sue
Scalable	?	Theory – yes : Practice – no

Centralized + Ring

Manageable	Yes	Just manage the ring
Coherent	Yes	As coherent as ring
Extensible	No	No more than ring
Fault Tolerant	Yes	Ring is a huge win
Secure	Yes	As secure as ring
Lawsuit-proof	No	Still single place to shut down
Scalable	Yes	Ring is a huge win

AOT LAB

Centralized + Decentralized

Manageable	No	Same as decentralized
Coherent	½	Better than decentralized
Extensible	Yes	Anyone can still join!
Fault Tolerant	Yes	Plenty of redundancy
Secure	No	Same as decentralized
Lawsuit-proof	Yes	Still no one to sue
Scalable	?	Looking very hopeful

Information Retrieval

- Centralized directory model (Nasper)
  - High efficiency
  - Low scalability
- Flooded requests model (Gnutella)
  - Low efficiency
  - High traffic
- Document routing model (FreeNet, Chord, CAN, Tapestry, Pastry)
  - Good efficiency

Centralized Directory Model

- The peers of the community connect to a central directory where they publish information about the files they offer for sharing
- Upon request from a peer, the central index will match the request with the best peer in its directory that matches the request
- If such a peer is found, then a file exchange will occur directly between the two peers

Flooded Requests Model

- Each request from a peer is broadcast to directly connected peers
- Peers propagate the request until either a peer that owns the requested information is found or the maximum number of flooding steps is reached
- If such a peer is found, then
  - An answer is back propagated to the source of the request
  - A file exchange will occur directly between the two peers

Document Routing Model

- Each peer from the network is assigned an ID and each peer also knows a given number of peers
- When a document is published on the community, an ID is assigned to the document based on a hash of the document's contents and its name
- Each peer will then route the document towards the peer with the ID that is most similar to the document ID
- This process is repeated until the nearest peer ID is the current peer's ID

Document Routing Model

- Each routing operation also ensures that a local copy of the document is kept
- When a peer requests the document, the request will go to the peer with the ID most similar to the document ID
- This process is repeated until a copy of the document is found
- Then the document is transferred back to the request originator, while each peer participating the routing will keep a local copy

Document Routing Implementation

- The primary goals are to reduce:
  - The number of hops that must be taken to locate a document of interest
  - The amount of routing state that must be kept at each peer
- Different implementations exist that
  - Either guarantee logarithmic bounds with respect to the size of the peer community
  - Or argue that logarithmic bounds can be achieved with high probability

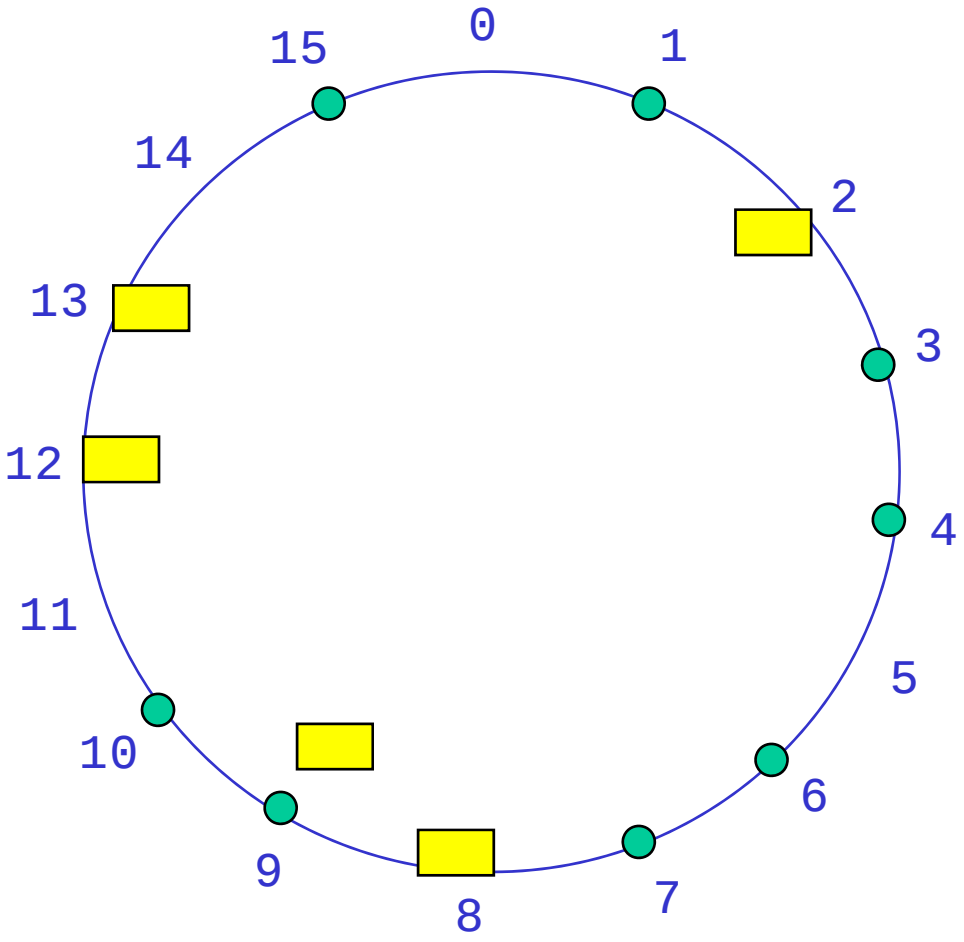
Chord Algorithm

- The identifier space is a uni-dimensional, circular identifier space
- IDs are assigned to the peers on the based on a hash on their IP address
- The routing table at each peer *n* contains $\log_2(N)$ entries for other peers where the i-th peer succeeds *n* by at least 2^{i-1}
- To route to another peer, the routing table at each hop is consulted and the message is forwarded toward the desired peer

Chord Algorithm

- When a peer joins the network, it contacts a gateway peer and routes toward its successor
- When the successor of the new peer is found, the new peer takes responsibility for the set of documents that have identifiers less than or equal to its identifier and establishes its routing table
- It then updates the routing state of all other peers in the network that are affected by the insertion
- To increase the robustness of the algorithm, each document can be stored at some number of successive peers

Chord Algorithm



CAN Algorithm

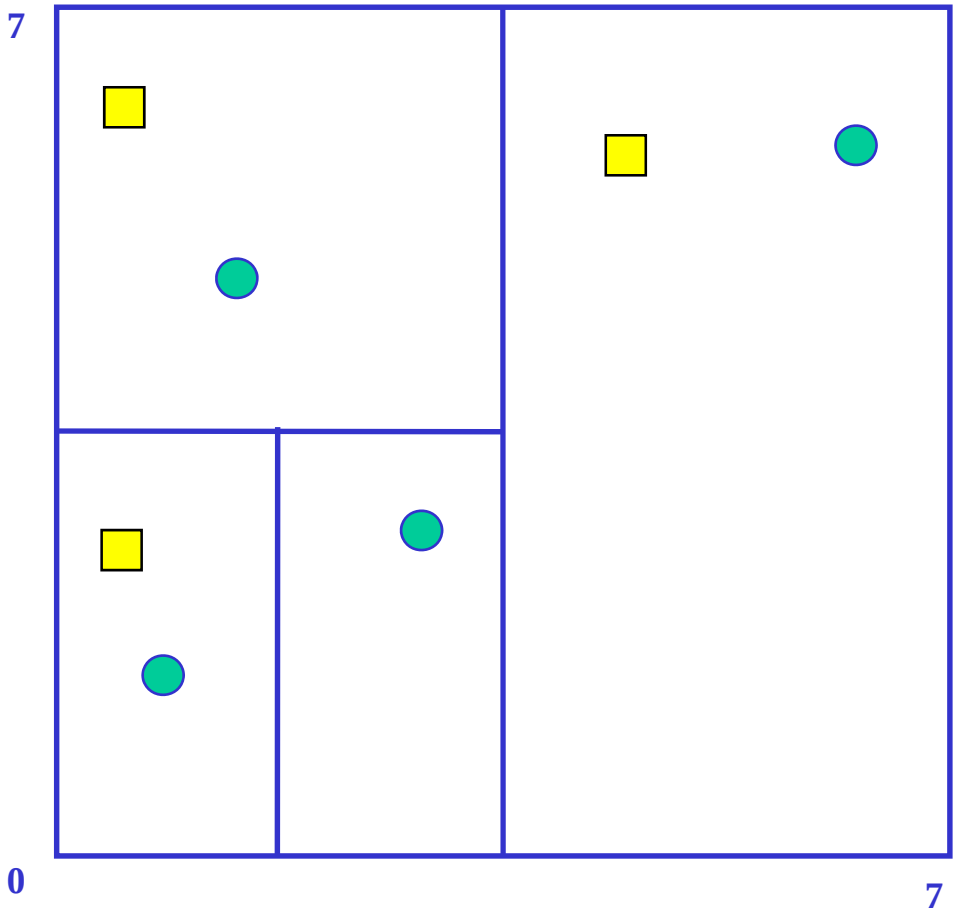
- CAN models the identifier space as multidimensional
- Each peer keeps track of its neighbors in each dimension
- When a new peer joins the network, it randomly chooses a point in the identifier space and contacts the peer currently responsible for that point
- The contacted peer splits the entire space for which it is responsible into two pieces and transfers responsibility of half to the new peer

CAN Algorithm

- The new peer also contacts all of the neighbors to update their routing entries
- To increase the robustness of this algorithm, the entire identifier space can be replicated to create two or more “realities” where each peer is responsible for a different set of information
- Therefore, if a document cannot be found in one reality, then a peer can use the routing information for a second reality to find the desired information

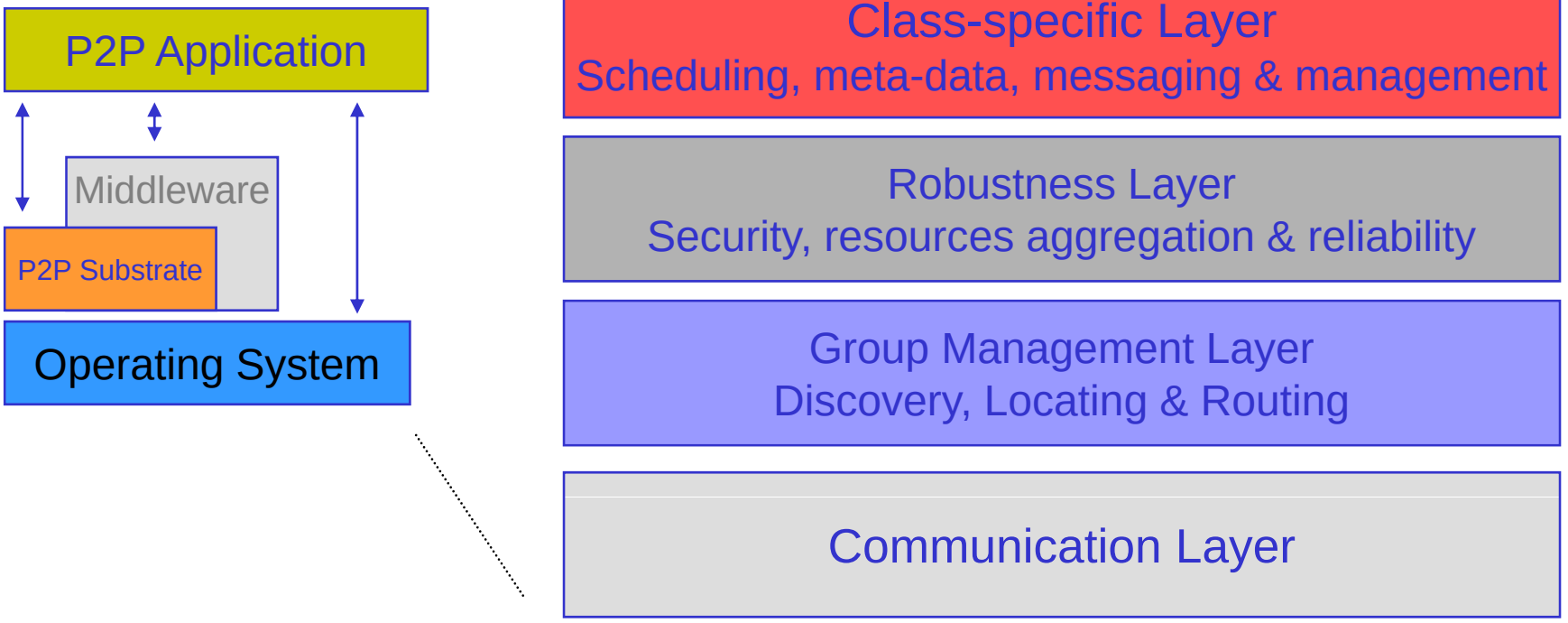
AOT LAB

CAN Algorithm



AOT LAB

Peer Software Stack



Peer-to-Peer Substrate

- Overlay management
  - Construction
  - Maintenance (peer join/leave/fail and network dynamics)
- Resource management
  - Allocation (storage)
  - Discovery (routing and lookup)

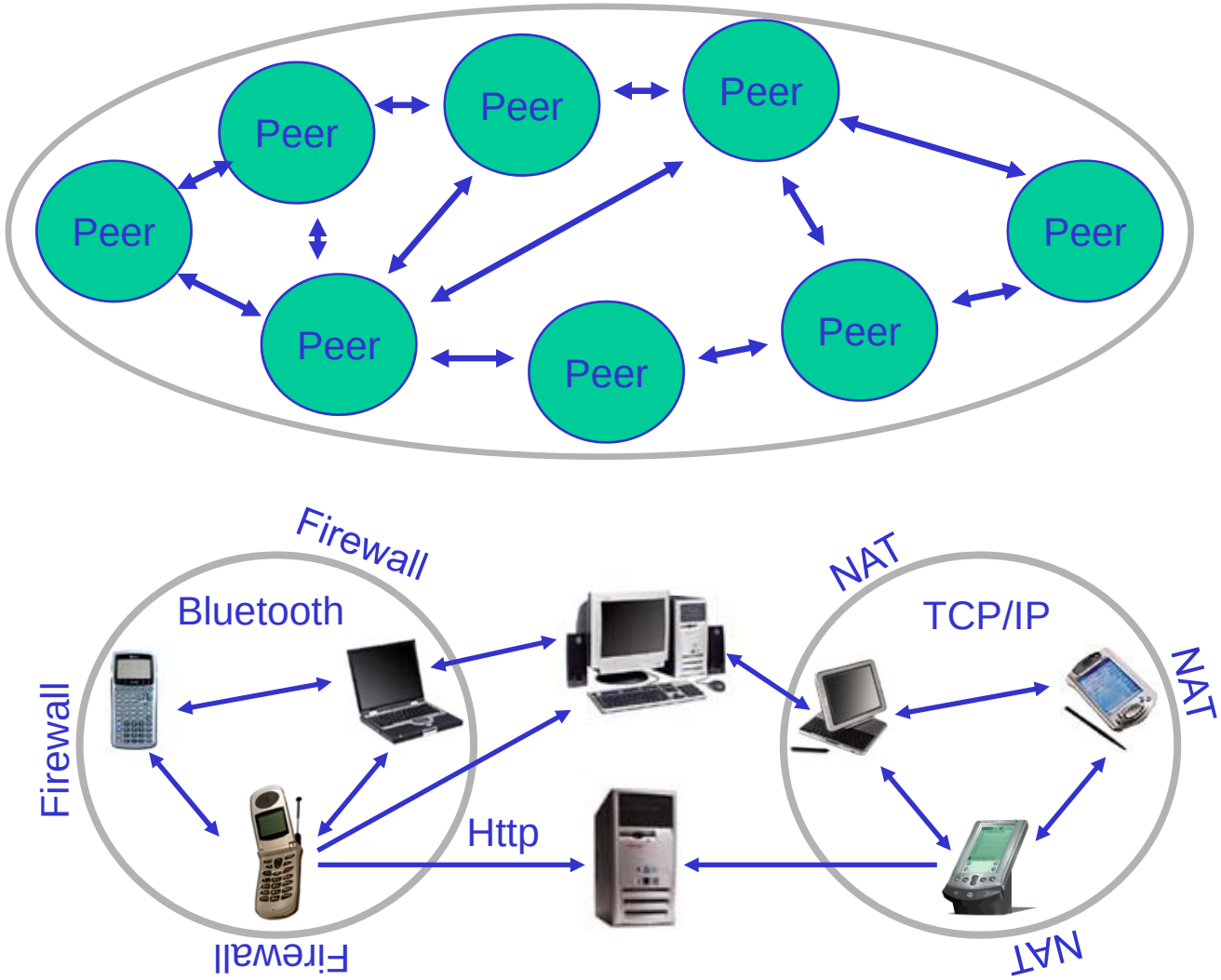
What is JXTA?

- Is an open platform designed for P2P computing based on a set of open, generalized protocols that allow any connected device on the network to communicate and collaborate as peers
- Its name is derived from the word *juxtapose,* meaning to place two entities side by side or in proximity
- By choosing this name, the development team recognized that P2P solutions would always exist alongside the current client/server solutions rather than replacing them completely

What is JXTA?

- Is not an application and does not define the type of the applications
- Is programming language and transport protocol independent
- Its goal is to develop basic building blocks and services to enable applications for peer groups
- Originally conceived by Sun Microsystems was designed with the participation of experts from academia and industry
- Implementations for Java SE, C/C++/C# and Java ME

JXTA Virtual Network



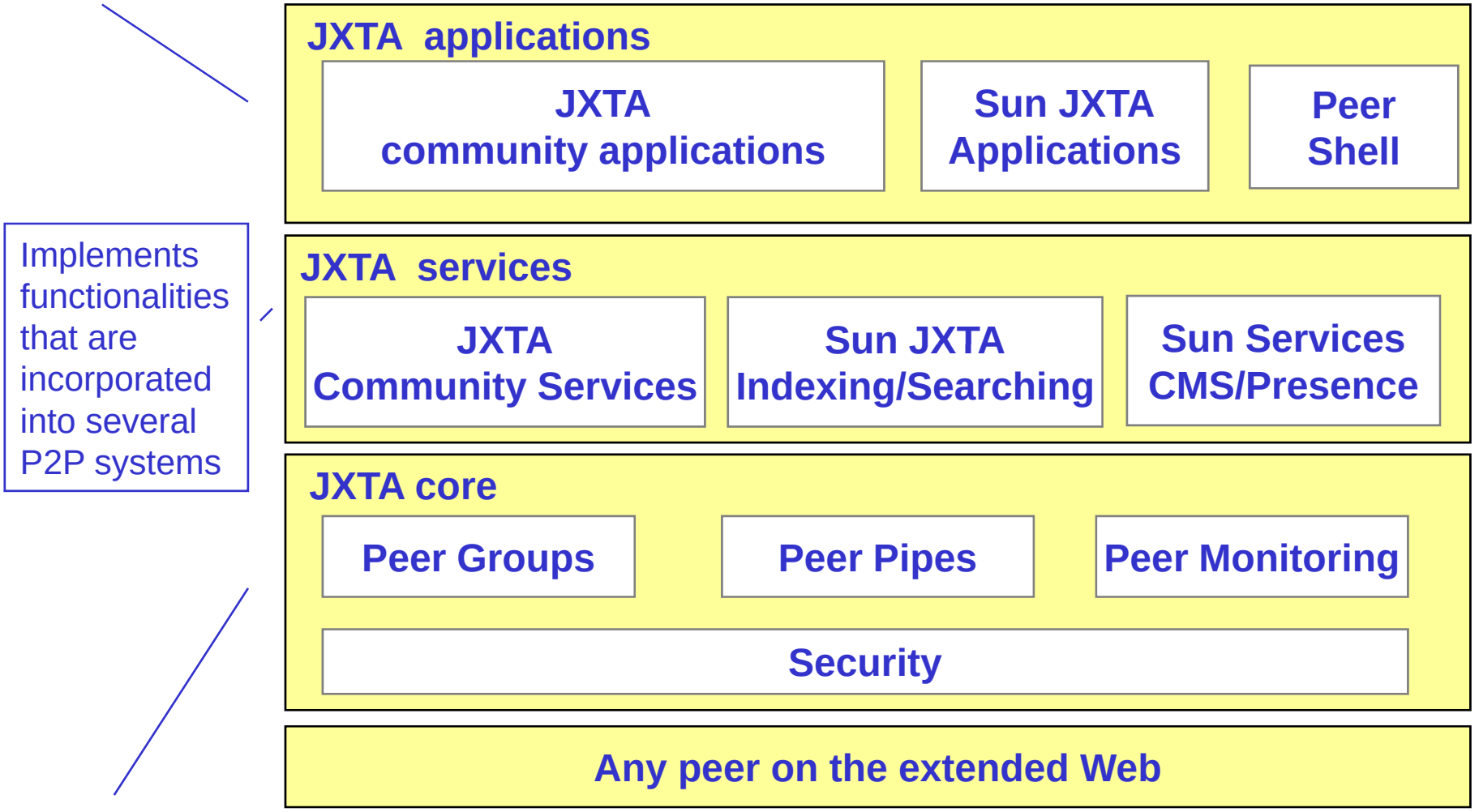
JXTA Layers

- Platform (minimal and essential primitives)
  - Peer and peer group discovery
  - Communication
  - Monitoring
  - Security
- Services (network / Web services)
  - Indexing and searching
  - Directory storage systems
  - Distributed file systems
- Applications
  - Instant messaging
  - Content management
  - Auction systems
  - Email systems

AOT LAB

JXTA Software Architecture

Builds on the capabilities of the services layer to provide the common P2P applications



Implements functionalities that are incorporated into several P2P systems

Provides elements essential to every P2P system

JXTA Components

- Peers
- Peer groups
- Pipes
- Messages
- Modules
- Network services
- Identifiers
- Advertisements
- Protocols

Peer

- May be any networked device
- Implements one or more JXTA protocols
- Operates independently and asynchronously
- Publishes one or more network interfaces for use with JXTA protocols
- May belong to more peer-group

Peer Group

- Is a collection of peers that have agreed upon a common set of services
- Can establish its own membership policy:
  - From open (anybody can join)
  - To highly secure and protected (sufficient credentials are needed to join)
- Peers may belong to more than one peer group simultaneously
- One special group, called the World Peer Group (the default peer group a peer joins) includes all JXTA peers

Pipe

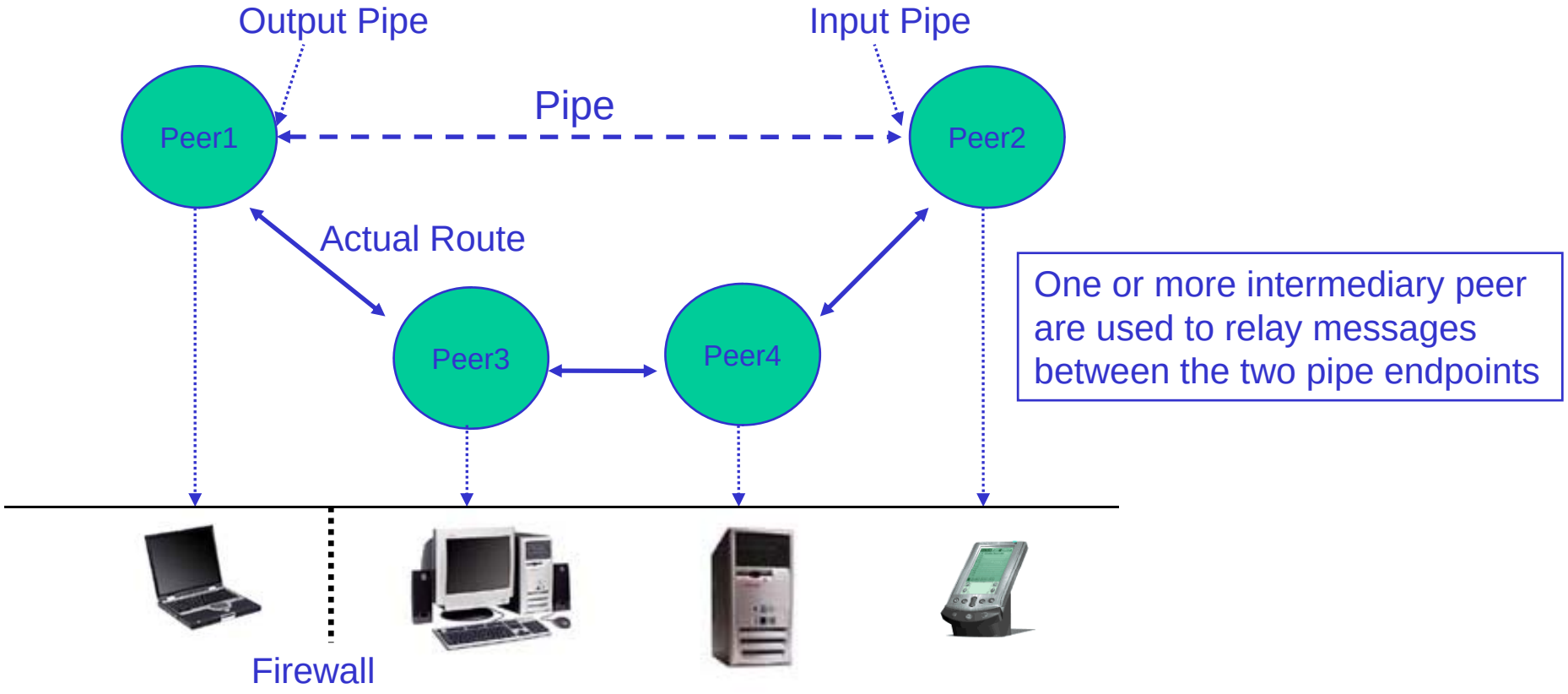
- Is an asynchronous and unidirectional message transfer mechanism used for service communication
- Support the transfer of any object, including binary code, data strings, and Java objects
- Is a virtual communication channel and may connect peers that do not have a direct physical link

Pipe

- Its endpoints are referred to as input and output pipes and are dynamically bound to peer endpoints at runtime
  - Peer endpoints correspond to available peer network interfaces (e.g., a TCP port and associated IP address)
- Can have endpoints that are connected to different peers at different times, or may not be connected at all

Pipe

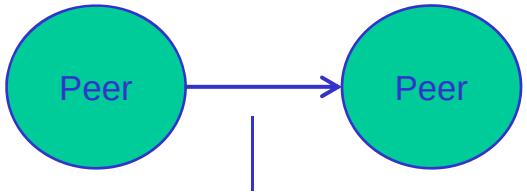
The output pipe and all input pipes must belong to the same peer group



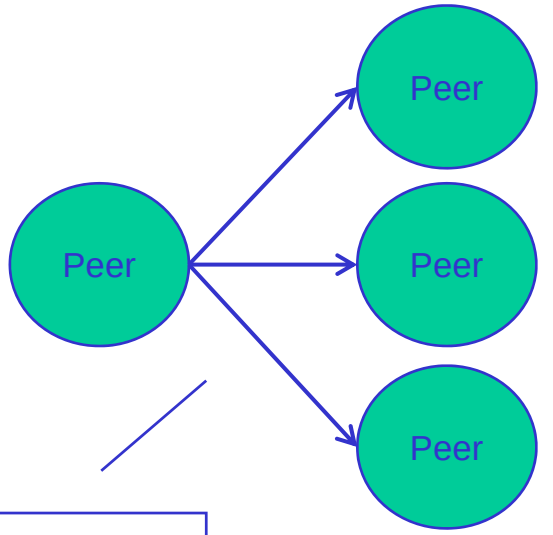
One or more intermediary peer are used to relay messages between the two pipe endpoints

AOT LAB

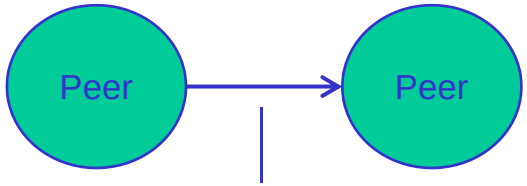
Pipe Communication Modes



A **point-to-point pipe** connects exactly two pipe endpoints together



A **propagate pipe** connects one output pipe to multiple input pipes



A **secure unicast pipe** is a point-to-point pipe that provides a secure communication channel

Additional types of pipe services can be built using the basic core pipes

Message

- Is an object that is sent between JXTA peers
  - It is the basic unit of data exchange between peers
- Is sent and received by the Pipe Service and by the Endpoint Service
- Is an ordered sequence of named and typed contents called message elements, i.e., is a set of name/value pairs
- Its contents type can be arbitrary

Message

- The JXTA protocols are specified as a set of messages exchanged between peers
- Each software platform binding describes how a message is converted to and from a native data structure such as a Java technology object or a C structure
- There are two representations for messages:
  - XML
  - Binary

Module

- Is an abstraction used to represent any piece of "code" used to implement a behavior in the JXTA World

- There are three types of module:
  - Module class
  - Module specification
  - Module implementation

Module

- Module class
  - Primarily used to advertise the existence of a behavior
- Module specification
  - Used to access the module
  - There can be multiple specifications for a given class (similar to interface part of WSDL)
- Module implementation
  - There can be multiple implementation for a given specification (similar to implementation part of WSDL)

Network Service

- Can be either pre-installed onto a peer or loaded from the network
- The process of finding, downloading and installing a service from the network is similar to performing a search on the Internet for a Web page, retrieving the page, and then installing the required plug-in
- JXTA protocols recognize two levels of network services:
  - Peer services
  - Peer Group services

Peer Service

- Is accessible only on the peer that is publishing that service

- Multiple instances of the service can be run on different peers, but each instance publishes its own advertisement

Peer Group Service

- Is composed of a collection of service instances running on multiple members of the peer group

- Peer group services are published as part of the peer group advertisement

Peer Group Service Types

- **Discovery** service searches for peer group resources, such as peers, peer groups, pipes and services
- **Membership** service rejects or accepts a new group membership application
- **Access** service validates requests made by one peer to another
- **Pipe** service creates and manages pipe connections between the peer group members
- **Resolver** service sends generic query requests to other peers
- **Monitoring** service allows one peer to monitor other members of the same peer group

Identifier

- Peers, peer groups, pipes and other JXTA resources need to be uniquely identifiable
- A JXTA ID uniquely identifies an entity and serves as a canonical way of referring to that entity
- URNs are used to express JXTA IDs
- URNs are a form of URI that "... are intended to serve as persistent, location independent, resource identifiers"
- Like other forms of URI, JXTA IDs are presented as text

Advertisement

- All JXTA network resources, such as peers, peer groups, pipes, and services, are represented by an advertisement
- An advertisement is language-neutral metadata structures represented as XML documents
- The JXTA protocols use advertisements to describe and publish the existence of a peer resources
  - Peers discover resources by searching for their advertisements
  - Peers may cache any discovered advertisements locally

Advertisement

- Each advertisement is published with a lifetime that specifies the availability of its associated resource
  - Lifetimes enable the deletion of obsolete resources without requiring any centralized control
  - An advertisement can be republished to extend the lifetime of a resource

AOT LAB

Module Advertisement Example

```
<jxta:MIA xmlns:jxta="http://jxta.org">
    <MSID>urn:jxta:uuid-
    DEADBEEFDEAFBABAFEEDBABE000000010206</MSID>
    <Comp>
        <Efmt>JDK1.4</Efmt>
        <Bind>V1.0 Ref Impl</Bind>
    </Comp>
    <Code>net.jxta.impl.peergroup.ShadowPeerGroup</Code>
    <PURI>http://www.jxta.org/download/jxta.jar</PURI>
    <Prov>sun.com</Prov>
    <Desc>
        Default NetPeerGroup reference implementation.
    </Desc>
</jxta:MIA>
```

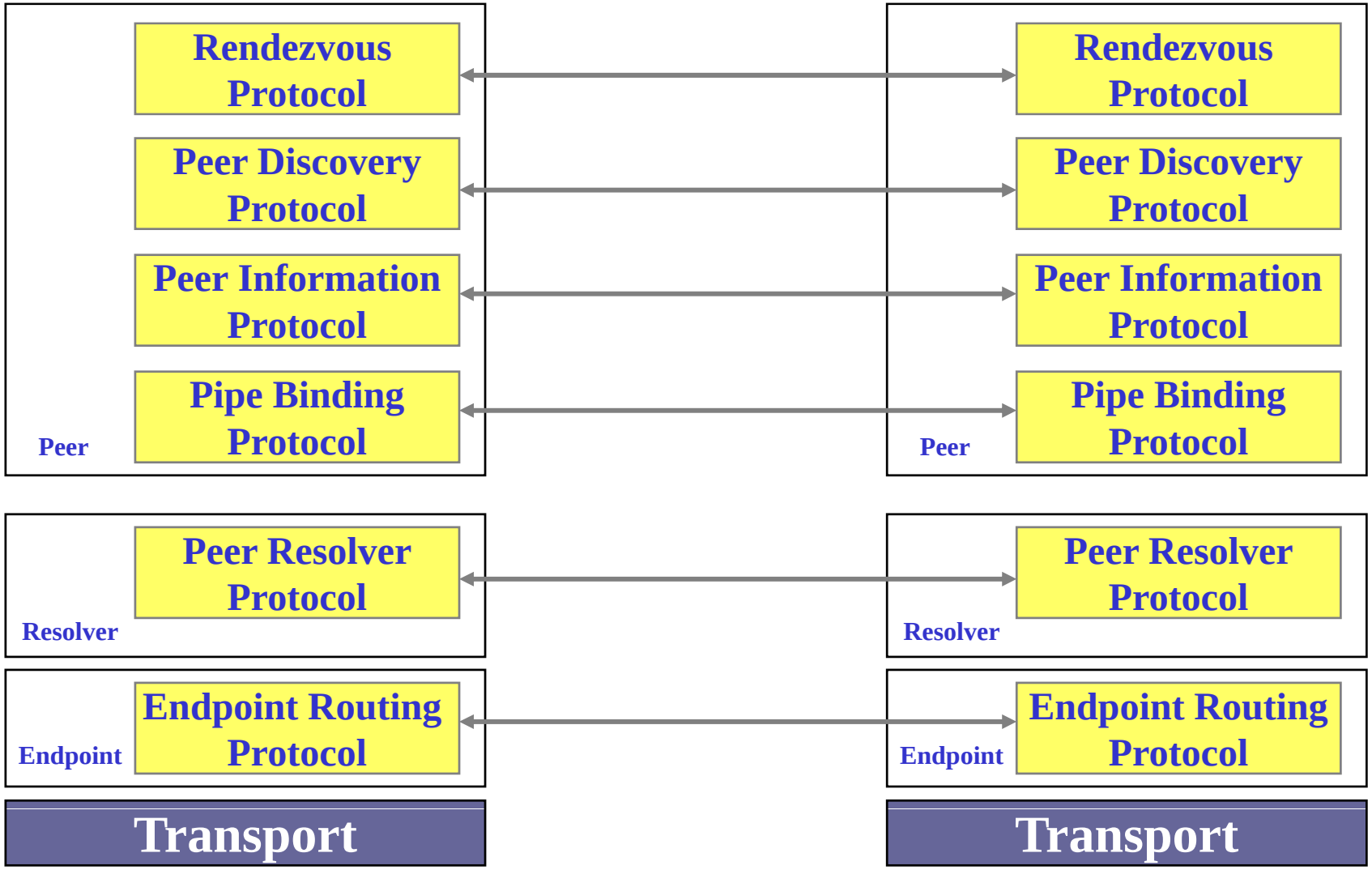
JXTA Protocols

- Set of six protocols
- Designed for ad hoc, pervasive and multi-hop peer-to-peer network computing
- Objectives
  - Interoperability
  - Platform independence
  - Ubiquity

JXTA Protocols

- Using these protocols, peers can cooperate to form self-organized and self-configured peer groups
  - Advertise resources
  - Discover resources
  - Route messages/communicate

JXTA Protocols



JXTA Protocols

- Each protocol is semi-independent of the others
- Possible implementation of a subset of the protocols
- Not entirely independent of each other
- Must be granted connectivity to other peers
- The JXTA specification reccomends to implement all protocols

JXTA Protocols

Sends and receives message, caches ads, replies to discovery request if it is in cache, usually does not forward the discovery request (simple peer), but can be configured to do it (rendezvous peer)

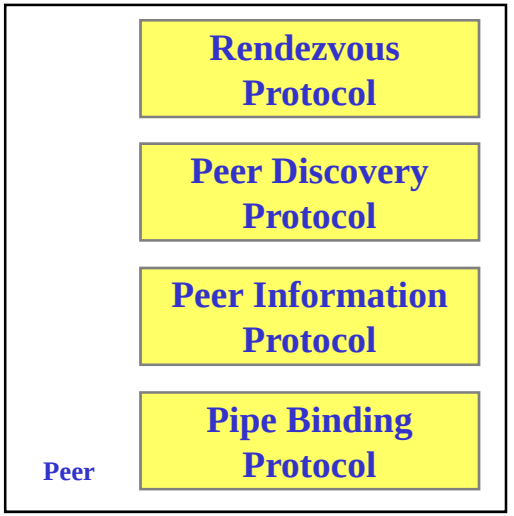
Sends information about the routes to other peers, routes messages to other peers and spools messages for unreachable peers

Super Peer

Peer

Sends and receives message, does not cache ads, does not route messages

Micro Peer



AOT LAB

Endpoint Routing Protocol

- Allows peers to find information about the available routes for sending a message to a destination peer
  - Allows the implementation of routing algorithms into JXTA
- Peers implementing the endpoint routing protocol respond to queries with available route information giving a list of gateways along the route
  - Route information includes an ordered sequence of relay peer IDs that can be used to send a message to the destination
- Router peers
  - Cache route information
  - Bridge different logical and physical networks

Peer Resolver Protocol

- Enables a peer to implement high-level search capabilities through the sending and reception of generic queries to find or search for peers, peer groups, pipes, and other information
- Its behavior is similar to the one of the publish-and-subscribe software pattern
- Queries can be directed to all peers in a peer group or to specific peers within the group
- A good metaphor for this protocol is TV broadcasting
  - A TV only processes the signal from the channel it is currently tuned to
  - A peer only processes a message of a type it is looking for

Pipe Binding Protocol

- Allows a peer to establish a virtual communication channel (i.e., a pipe) between peers
- Allows the binding of the two or more ends of the pipe endpoints forming the connection
- A peer binds a pipe advertisement to a pipe endpoint thus indicating here messages actually go over the pipe
- Supports two operations:
  - Bind (occurs during the open operation)
  - Unbind (occurs during the close operation)

Peer Information Protocol

- Provides a set of messages to obtain peer status information allowing a peer to learn about the capabilities and status of other peers
- For example, to ping a peer to determine if it is alive or to ask for the time during which a system is working without failure
- This protocol is useful for implementing monitoring

Peer Discovery Protocol

- Enables peers to discover peer resources
  - Peers, peer groups, pipes, services, etc
- Default discovery protocol
- Based on web-crawling and rendezvous peers
- Rendezvous peers
  - Cache advertisements
  - Forward requests
- Discovery request messages
  - Unicast
  - Propagate

Rendezvous Protocol

- Is responsible for propagating messages within a peer group
- While different peer groups may have different means to propagate messages, this protocol defines a simple protocol that allows:
  - Peers to connect to service (be able to propagate messages and receive propagates messages)
  - Control the propagation of the message (TTL, loopback detection, etc.)
- This protocol is used by the Peer Resolver Protocol and by the Pipe Binding Protocol in order to propagate messages

Security

- Dynamic P2P networks such as the JXTA network need to support different levels of resource access

- JXTA peers operate in a role-based trust model, in which an individual peer acts under the authority granted to it by another trusted peer to perform a particular task

Security

- Five basic security requirements must be provided:
  - Confidentiality
  - Authentication
  - Authorization
  - Data integrity
  - Refutability

Basic Security Requirements

- Confidentiality: guarantees that the contents of a message are not disclosed to unauthorized individuals
- Authentication: guarantees that the sender is who she or he claims to be
- Authorization: guarantees that the sender is authorized to send a message
- Data integrity: guarantees that the message was not modified accidentally or deliberately in transit
- Refutability: guarantees that the message was transmitted by a properly identified sender and is not a replay of a previously transmitted message