6. Peer-to-peer (P2P) networks I.

PA159: Net-Centric Computing I.

Eva Hladká

Faculty of Informatics Masaryk University

Autumn 2010

Lecture Overview I

1. Client-Server vs. Peer-to-Peer
   - Client-Server Systems
   - P2P Systems
   - Comparison
2. Generic P2P Architecture
   - Overlays and Peer Discovery
   - Service/Resource Discovery
3. Taxonomy of P2P Systems
   - Centralized P2P Systems
   - Decentralized P2P Systems
   - Hybrid P2P Systems

Lecture Overview I

Distributed Applications I.

- a distributed application consists of multiple software modules located on different computers
  - the modules interact with each other over a communication network connecting the different computers
    - the communication network is used for synchronisation and communication between the modules
  - it is possible that multiple users may use the application concurrently on different computers
- to build a distributed application, it is necessary to decide:
  - how to place those software modules on the different computers in the network
  - how each software module discovers the other modules it needs to communicate with

Distributed Applications II.

- two basic approaches:
  - Client-Server architecture
  - Peer-to-Peer (P2P) architecture
- hybrids are possible and indeed useful

Client-Server Architecture I.

A client-server system comprises of two types of software modules:

- *server module*
  - one centralized instance
    - but might be internally replicated for scaling purposes
  - passively listens for connections from clients
  - multiple client requests may be handled:
    - sequentially
    - concurrently (multithreaded servers)
    - by several replicated servers at different locations
  - pending clients' requests may be queued up
  - servers are assumed to be reliable, often running in a data centre (dedicated/virtualized hardware)

Client-Server Architecture II.

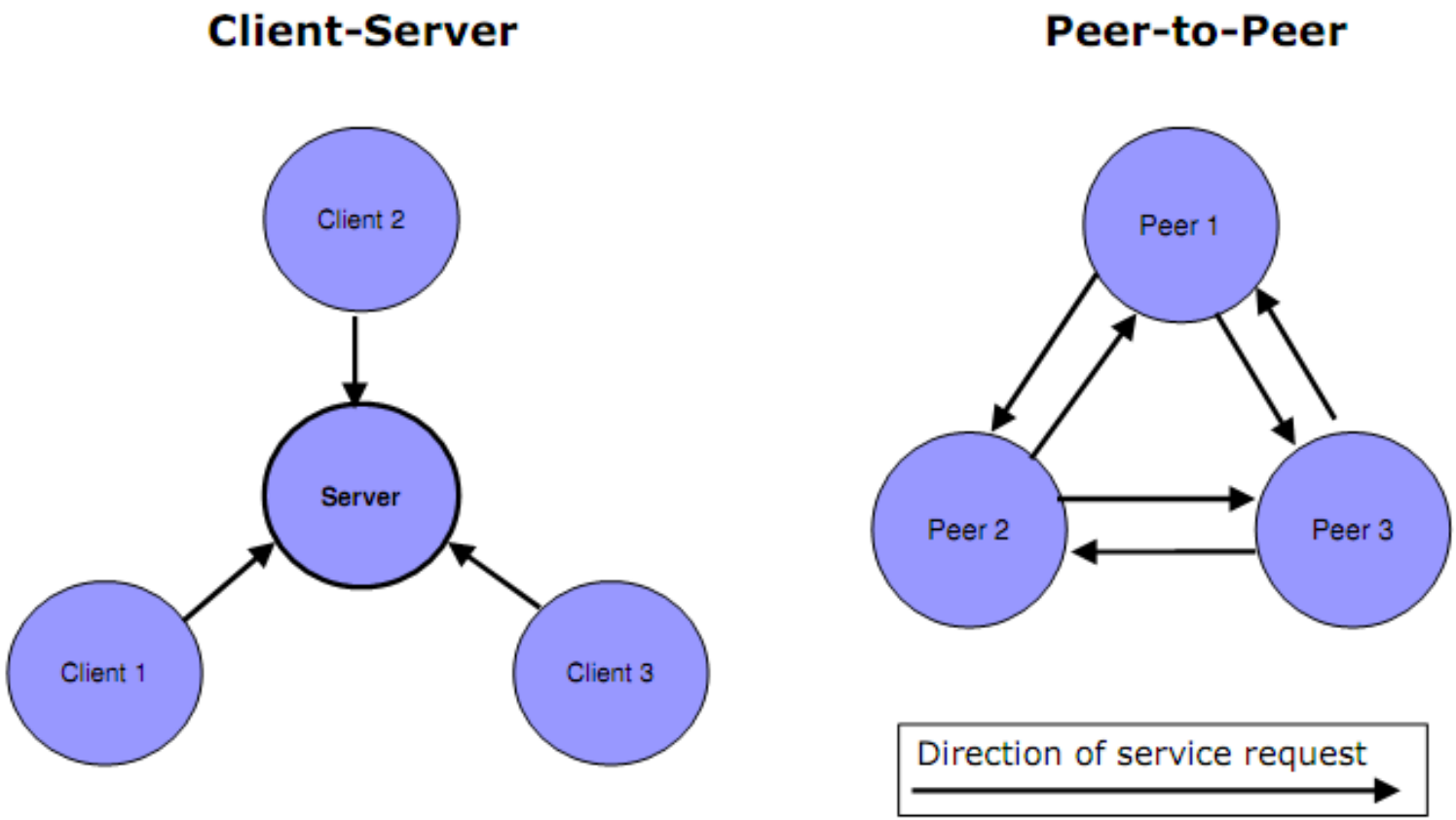
- *client module*
  - multiple distributed instances, possibly controlled by different users
  - actively initiates a connection to a server
  - no direct communication between clients
  - clients need to know the network address and port number of a server
    - service discovery is typically performed through client configuration
  - clients may be unreliable without affecting overall system stability

- examples of client-server systems:
  - web server/web browsers
  - web server/client applications (web services)
  - SSH/Telnet/FTP server/clients
  - NFS/SMB server/clients
  - ...

P2P Architecture

- a P2P system consists of many identical software modules (peers) running on different computers
- peers communicate directly with each other
- each peer is a server as well as a client:
  - provides services to other peers
  - requests services from other peers
- unlike dedicated servers, peers tend to be *unreliable*
- service discovery is more complicated since there are many servers continuously appearing and disappearing at different network locations
- provide natural scalability due to multiple servers
- can work without allocating dedicated server machinery

Communication Structure Comparison



Peer-to-Peer Systems Definition

Peer-to-Peer Systems

Peer-to-peer (P2P) systems are distributed systems consisting of interconnected nodes able to self-organize into network topologies with the purpose of sharing resources such as content, CPU cycles, storage and bandwidth, capable of adapting to failures and accommodating transient populations of nodes while maintaining acceptable connectivity and performance, without requiring the intermediation or support of a global centralized server or authority.

P2P Properties

- *Symmetric role*
  - each participating node typically acts both as a server and as a client
    - however, in many designs this property is relaxed by the use of special peer roles ("super peers" or "relay peers")
- *Scalability*
  - P2P systems can scale to thousands of nodes
    - the P2P protocols cannot require "all-to-all" communication or coordination
- *Heterogeneity*
  - a P2P system is (usually) heterogeneous in terms of the hardware capacity of the nodes
- *Distributed control (Decentralization)*
  - ideally, no centralized structures should exist in P2P systems
- *Dynamism*
  - the topology of P2P systems may change very fast due to joining of new nodes or leaving existing ones
- *Resource sharing*
  - each peer contributes system resources (computing power, data, bandwidth, presence, etc.) to the operation of the P2P system
- *Self-organization*
  - the organization of the P2P system increases over time using local knowledge

P2P Applications

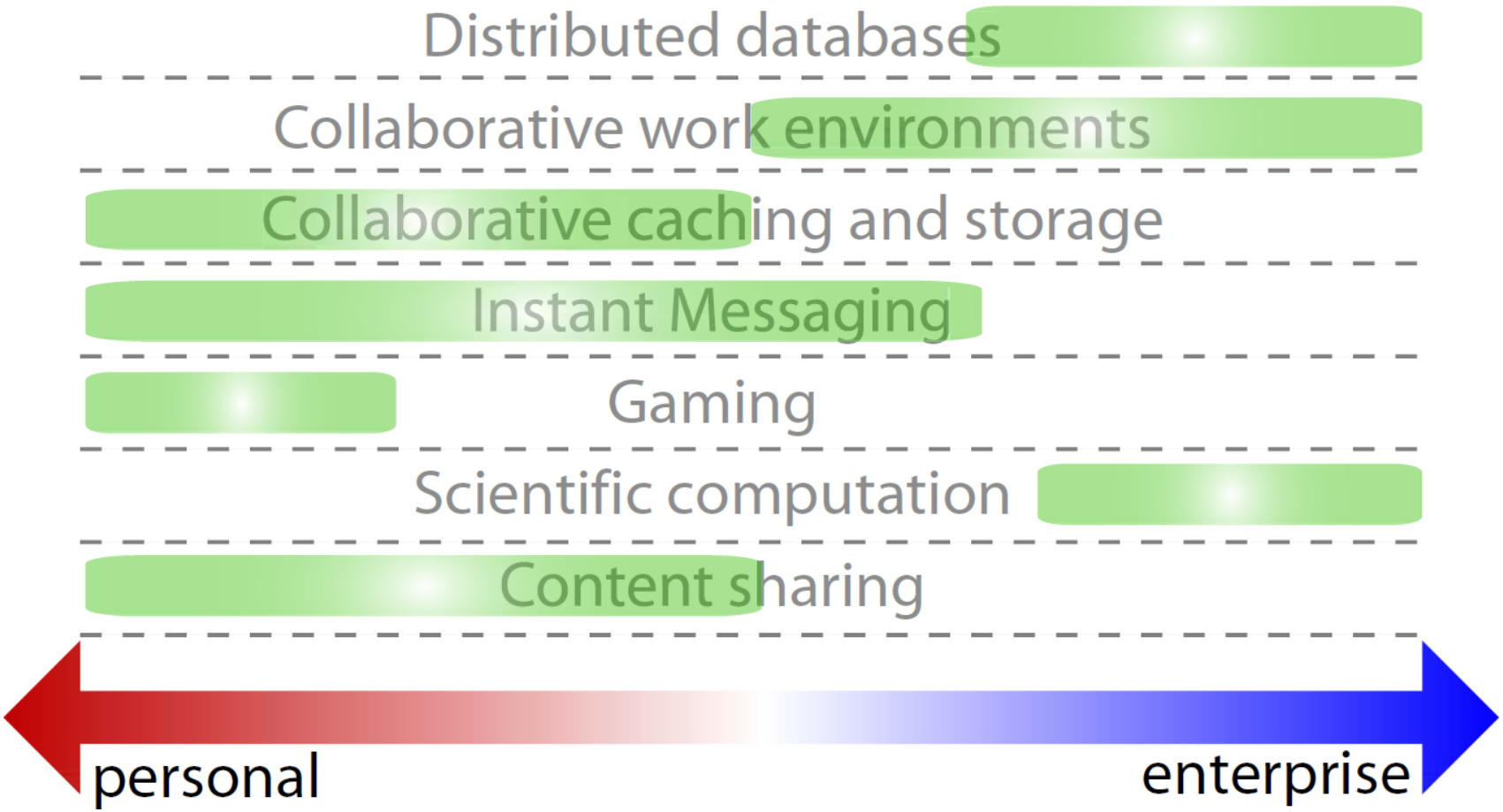


Figure: P2P Applications.

Client-Server vs. Peer-to-peer

Comparison I.

The systems can be compared from several points of view:

- *Ease of development*
  - C-S is more established and familiar than P2P
  - C-S exhibits simple interaction patterns for clients and server, while P2P involves more complex interaction patterns between peers
- *Manageability*
  - it is easier to maintain a centralized server in a C-S environment than keeping a track of and maintaining several distributed peers in a P2P system
- *Scalability*
  - C-S scalability is limited by fixed server hardware, though scaling can be achieved through load balancing over multiple servers at increased cost
  - P2P is scalable by nature, since as the number of peers grows, so does the "server" capacity

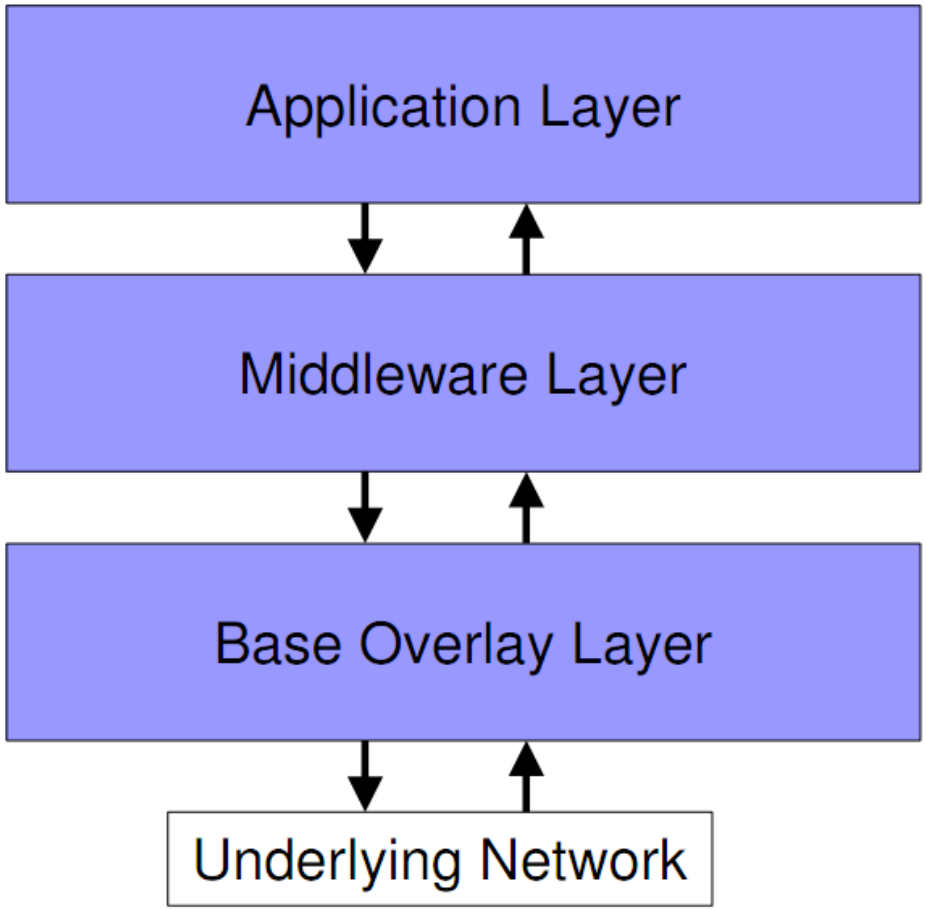
Client-Server vs. Peer-to-peer

Comparison II.

- *Security*
  - responsibility for the C-S security lies within the server, which is centrally hosted in a secure environment
  - responsibility for P2P security is distributed across peers in different administrative domains, some of which might be compromised
- *Reliability*
  - the C-S's reliability is achieved through the use of multiple redundant servers (possibly hosted at different locations) with automatic fail-over, at additional cost
  - with P2P, resilience comes free of charge, since multiple peers are usually able to provide the same service in the case that some peers fail

Lecture Overview I

P2P Architecture



- libraries exist that provide reusable P2P functionality (e.g. JXTA)
- some applications integrate all of the above (e.g., Gnutella, Bittorrent, etc.)

P2P Architecture

Base Overlay Layer I.

- the *base overlay layer* is responsible for:
  - discovering new peers
  - maintaining the P2P overlay (virtual) network
  - forwarding messages between peers
- the *overlay network* is a virtual network laid over the "physical" network (e.g. TCP/IP)
  - overlay network "wires" are implemented using underlying network facilities (e.g. TCP connections or UDP messages)
  - overlay network distance is measured in the number of hops from peer to peer
    - peers, that are distant in the physical network may be neighbours in the overlay network, and vice-versa
  - the performance of the P2P system is influenced by the structure of the overlay network

P2P Architecture

Base Overlay Layer II.

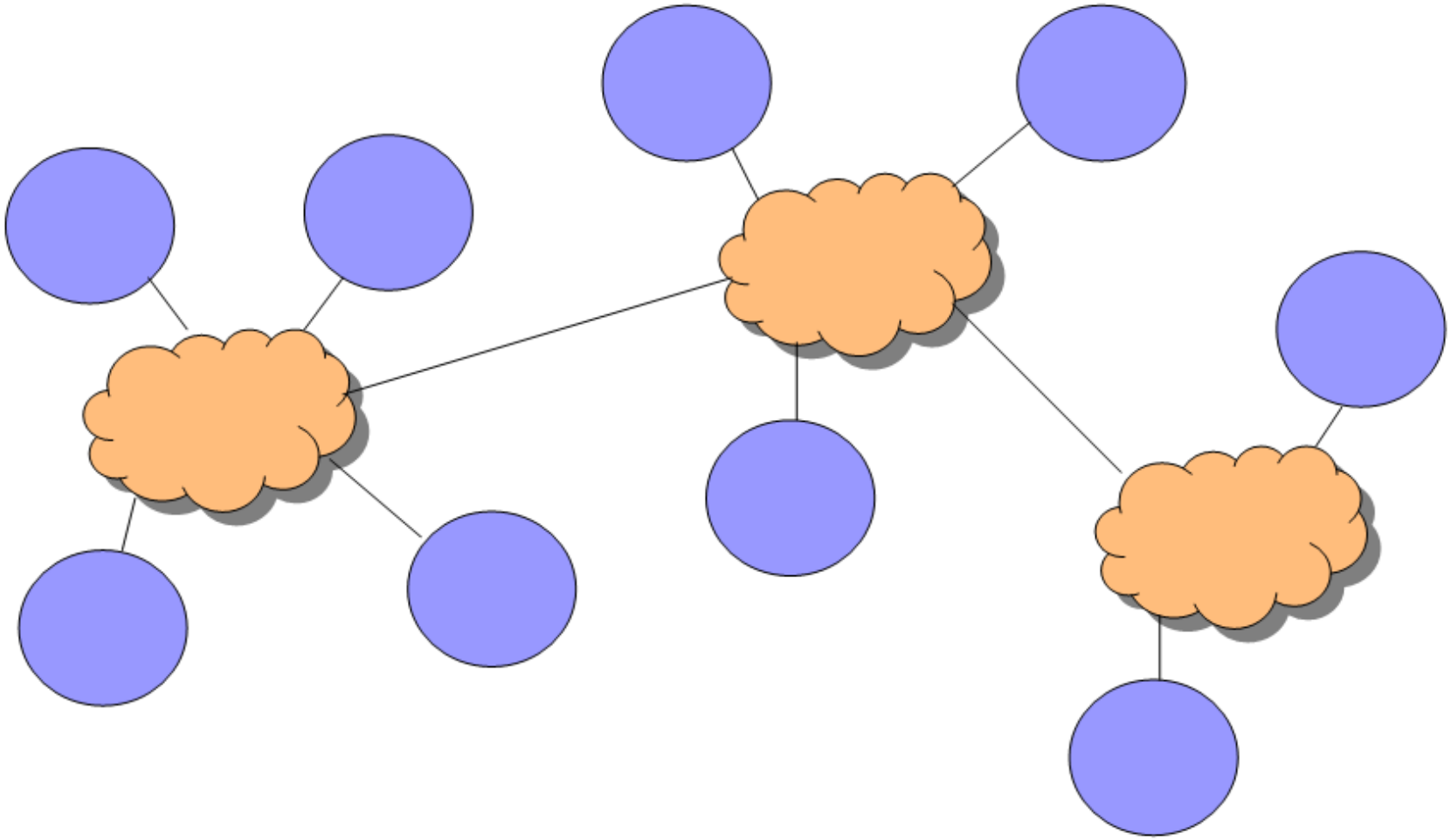


Figure: Overlay vs. Underlying Network.

P2P Architecture

Base Overlay Layer II.

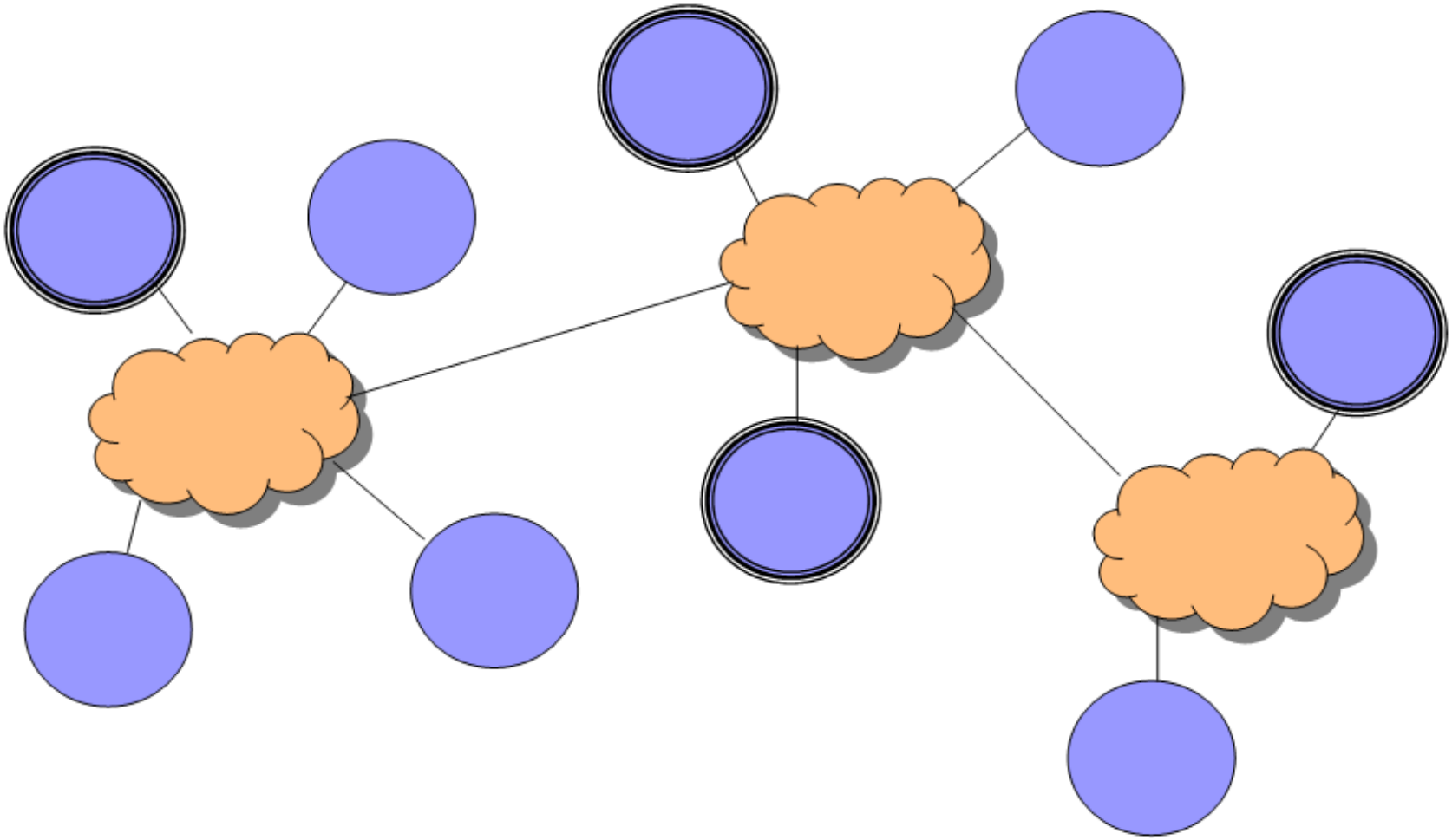


Figure: Overlay vs. Underlying Network.

P2P Architecture

Base Overlay Layer II.

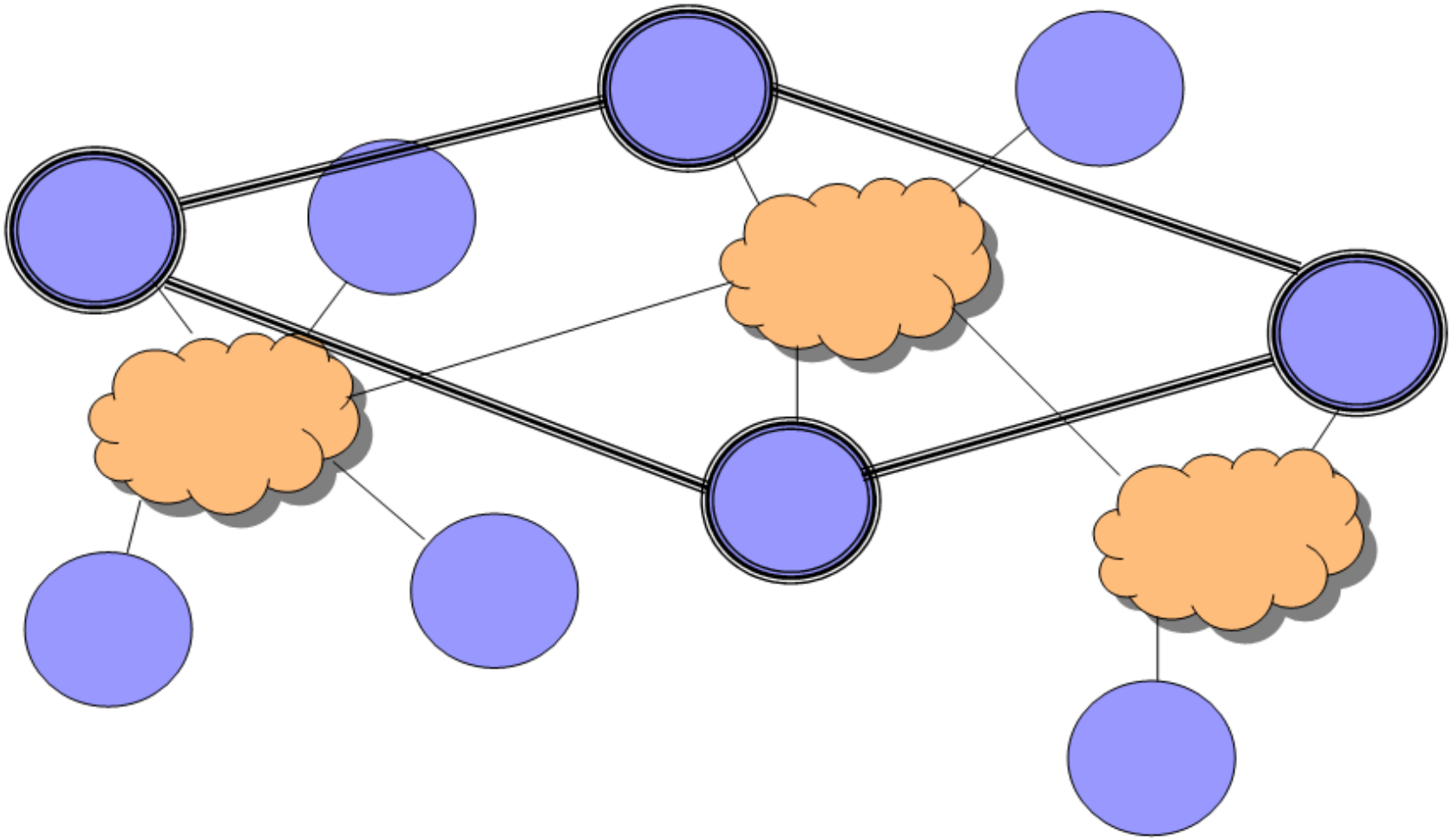


Figure: Overlay vs. Underlying Network.

P2P Architecture

Base Overlay Layer II.

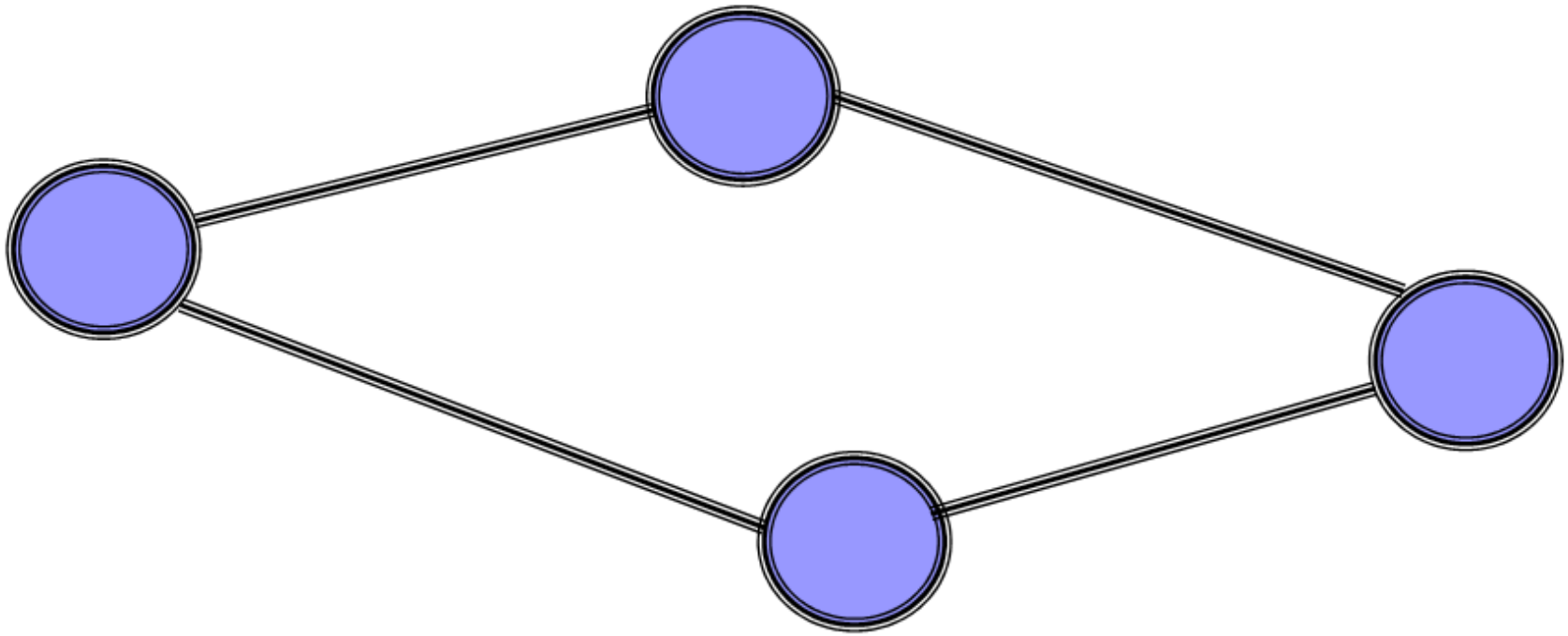
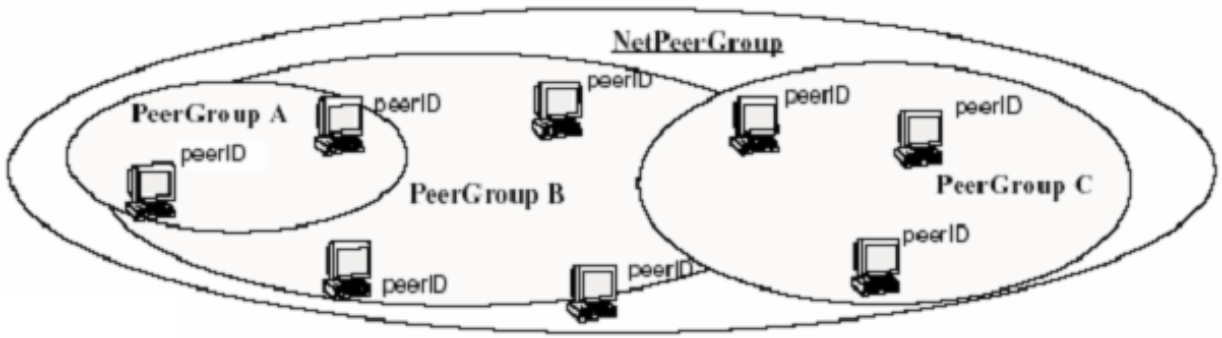


Figure: Overlay vs. Underlying Network.

P2P Architecture

Middleware Layer

- the *middleware layer* facilitates P2P application development by hiding overlay and service discovery issues
- it provides access to the services/resources provided by peers, and may be responsible for functions such as:
  - **security:** controlling access to services/ resources
  - **service/resource discovery:** searching and indexing services/resources distributed across peers
  - **peer groups:** coordinating peers that provide or consume a particular service/resource
    - may provide fault tolerance and persistent state



- e.g., JXTA (Java P2P platform), Windows P2P Networking, P2P.NET, etc.

P2P Architecture

Application Layer

- the middleware services can be used to build complete applications:
  - file sharing – e.g., Napster, Gnutella, Kazaa, ...
  - routing protocols
  - instant messaging, videoconferencing applications – e.g., Skype
  - distributed file systems
  - distributed backup systems
  - distributed computing – e.g., grid computing, SETI@Home, ...
  - and many many more...

Overlays and Peer Discovery

- a P2P network is typically a "virtual" network overlaid on an existing network (e.g. the Internet)
  - the overlay is used for indexing and peer discovery and make the P2P system independent from the physical network topology
  - content is typically exchanged directly over the underlying IP network
- a new peer needs to discover at least one existing peer in order to join a P2P network
  - network location information: IP address, listening port number, etc.
- if no peers are found immediately, the new peer either
  - passively waits for new participants, or
  - proactively looks for potential new participants
- it is hard to locate existing peers in a large network such as the Internet

Overlays and Peer Discovery

Initial Peer Discovery I.

Static configuration:

- each peer is preconfigured with a list of the network locations (IP address and port number) of every other peer in the system
- on startup (and possibly periodically) each peer attempts to connect to some other peers in its list, some of which may be running
- due to the manual configuration, this is only suitable for P2P networks with a small number of peers which do not change frequently
- can alternatively be used to initially contact a small number of "well-known" peers that are guaranteed to be online

Overlays and Peer Discovery

Initial Peer Discovery II.

Centralized directory:

- each peer is preconfigured with the network location of a centralized server
- each peer contacts the server on startup (and possibly periodically) to:
  - obtain an updated list of currently active peers
  - indicate to the server that it is active
- most subsequent communications bypass the server, using the P2P overlay network to route messages instead
  - occasionally, other services are also provided by the server (e.g. a list of files hosted by each peer)
- peers may go offline
  - cleanly, the peer's shutdown procedure contacts the server to remove it from the active peer list
  - without warning (crash, network or power failure), making the server's active peer list obsolete (it's necessary to use active peer list item expiry and periodic liveness checks)
- usually, a peer only needs to connect to a few peers on the overlay network
  - the other members can be discovered by the *member propagation techniques*
- centralized directory server is a single point of failure

Overlays and Peer Discovery

Initial Peer Discovery III.

Member Propagation Techniques with Initial Member Discovery:

- in general, it is not necessary to discover all of the participating members in the network
  - in many cases, discovering a subset of the participating members is adequate
- after discovering just one existing peer, information about the rest of the P2P network can be obtained from it
  - if each peer maintains a full member list $\rightarrow$ easy for any new peer to obtain a full member list from any other peer
  - alternatively, each peer can maintain a partial member list, replacing offline peers with new ones from neighbouring peers' lists

Overlays and Peer Discovery – Overlay Network Topology

- intermediate peers in the overlay network forward messages between indirectly connected peers
- the overlay topology significantly affects P2P system performance
- two key properties determine the effectiveness of the overlay mesh:
  - **Diameter:** longest distance between any two peers (overlay hops or latency)
    - should be minimized
  - **Average Degree:** average number of links per peer (high AD increases message load, but improves fault tolerance)
    - should be kept at a moderate level
- it is necessary to avoid linear formations and splits in the mesh
- common topologies:
  - *Random Mesh*
  - *Tiered*
  - *Ordered Lattice*

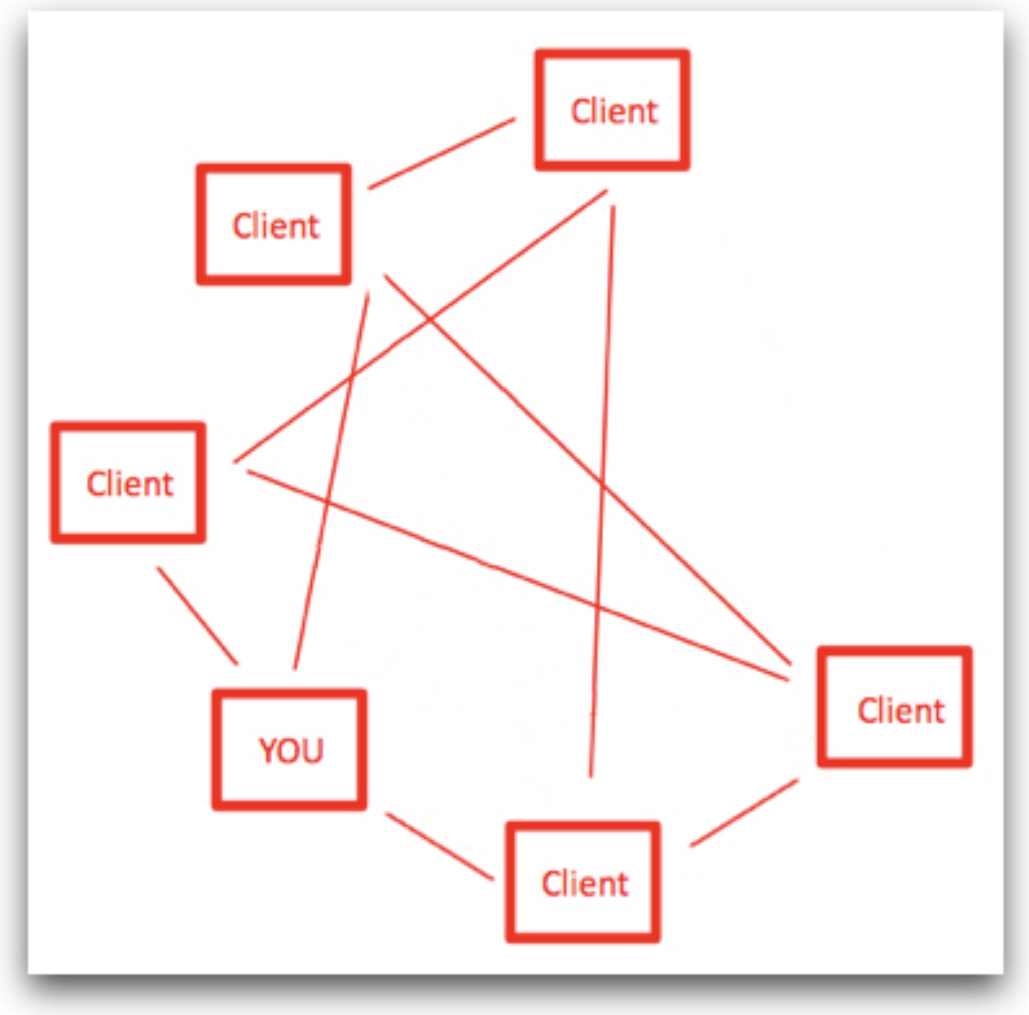
Overlays and Peer Discovery – Overlay Network Topology

Random Mesh

- each peer discovers a number of other peers and attempts to connect to them indiscriminately
- this (hopefully) results in a random structure with uniform degree
- distant peers on underlying network could be overlay neighbours
  - *solution:* connect to peers with lowest latency
- random mesh is suitable for linking a large number of peers with uniform resources and connectivity
- search message flooding can easily be used to discover resources/services on other peers
  - but generates a lot of traffic

Overlays and Peer Discovery – Overlay Network Topology

Random Mesh



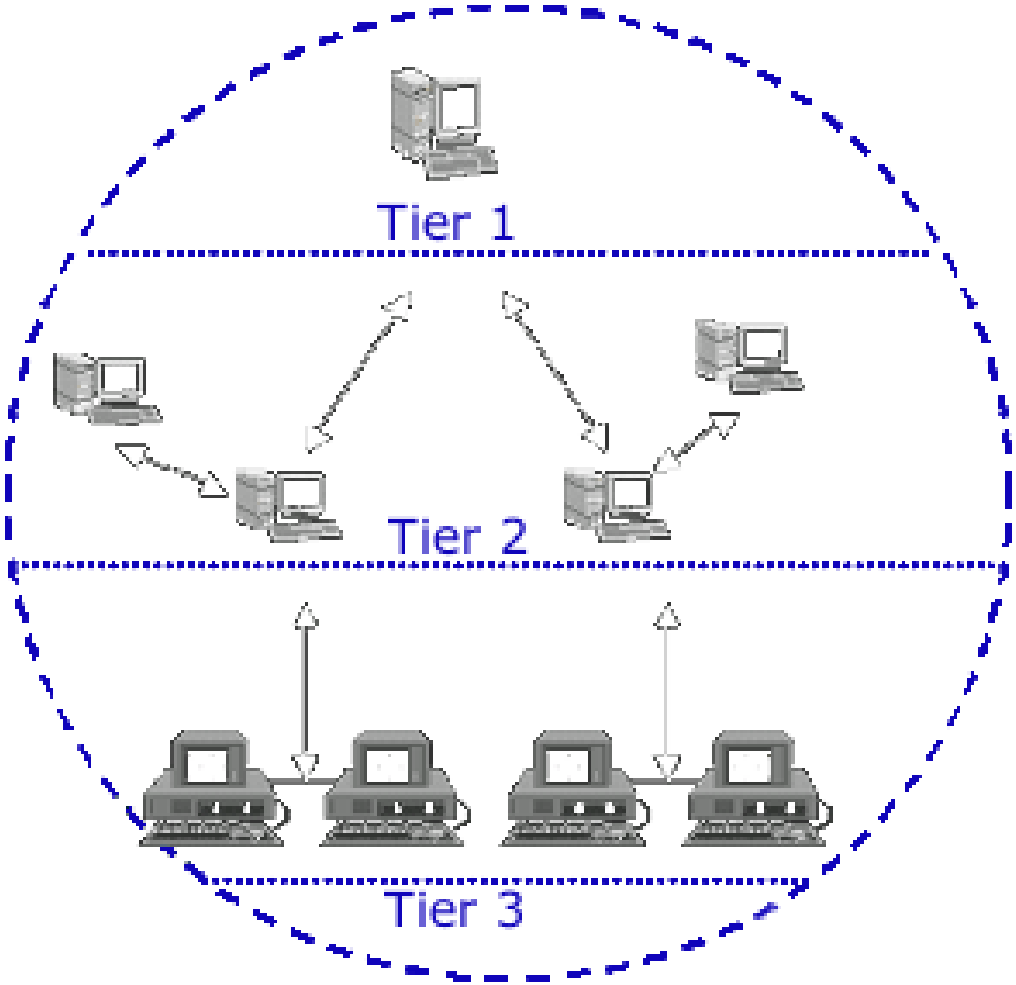
Overlays and Peer Discovery – Overlay Network Topology

Tiered Structure

- peers are ordered into tiers of a tree depending on their advertised resources and connectivity (e.g. Kazaa's nodes and supernodes, 2-tier)
  - tier 0 is the foundation tier containing (possibly well-known) reliable peers with adequate resources and message forwarding capacity
  - at each tier, every peer is linked to a number of peers of a lower tier and forwards messages up and down
  - poorly-resourced leaf peers only link to their 'super-peer' and do not forward other peers' messages; they are omitted from peer discovery
- the system needs to recover from peers leaving abruptly and disrupting the tree structure
- the hierarchy may be optimized to follow the underlying network's structure (e.g. P2P video streaming)

Overlays and Peer Discovery – Overlay Network Topology

Tiered Structure



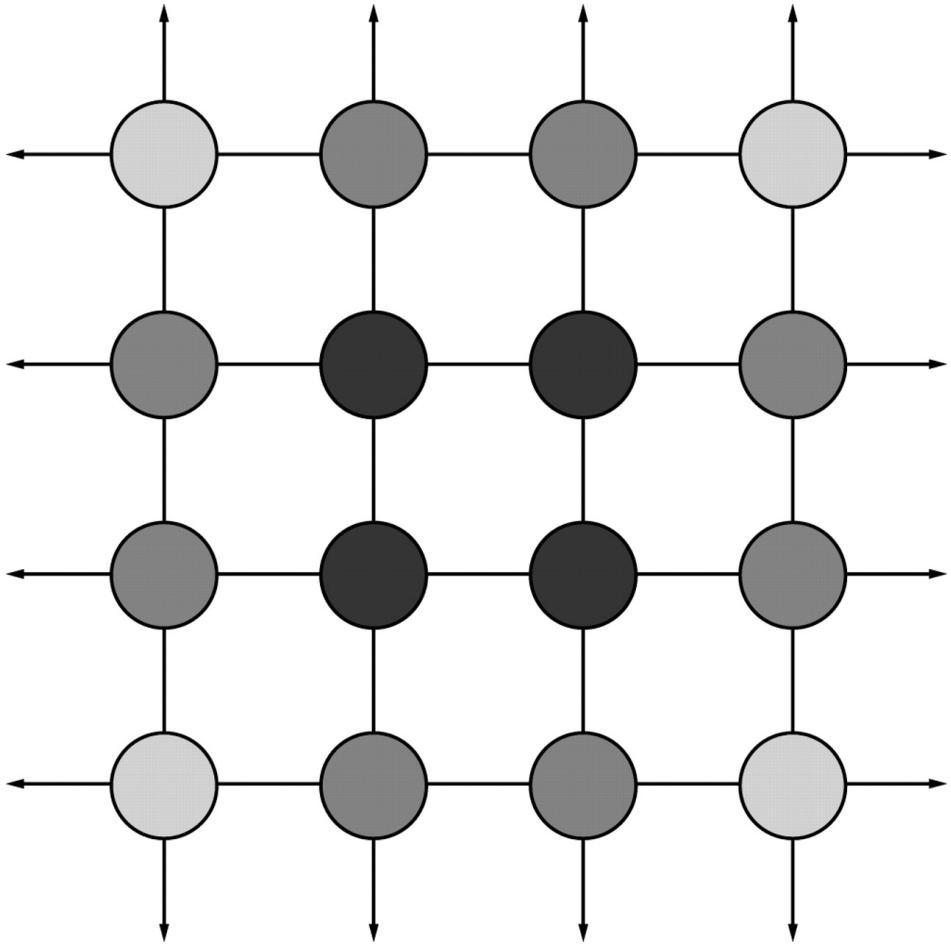
Overlays and Peer Discovery – Overlay Network Topology

Ordered Lattice

- in a two dimensional lattice, peers organize themselves in a rectangular grid:
  - each node maintains direct connections to 4 neighboring peers (except edge peers)
    - peers on opposite edges can also link to form a torus
  - can be extended to n dimensions
- messages are routed parallel to the lattice axes
- peer additions and deletions must be handled on the fly, possibly distorting the structure
  - insertions and deletions of nodes imply that different rows/columns have different numbers of members between themselves
- peer coordinates in a multi-dimensional lattice may be used as a key to locate resources in *content addressable networks (CAN)*
  - sometimes also denoted as *Distributed Hash Table (DHT)*

Overlays and Peer Discovery – Overlay Network Topology

Ordered Lattice



Service/Resource Discovery

- a peer must advertise its services to enable their discovery and subsequent use by other peers
  - e.g., in file sharing applications, the "service" is a shared file/block
- service discovery is itself a service
  - centralized – a server is asked for service location
    - Napster, UDDI for web services
  - pure P2P – a request is flooded or hashed through the peers
    - flooding, overlay multicast, CAN/DHT
- when a search message reaches a matching advertisement on a peer, the server's location is returned to the originator
- actual service messages are either routed through the overlay or directly via underlying network by the application
- can be optimized by caching advertisements/data (e.g. file/block) along search/return path on the overlay

Lecture Overview I

1. Client-Server vs. Peer-to-Peer
   - Client-Server Systems
   - P2P Systems
   - Comparison
2. Generic P2P Architecture
   - Overlays and Peer Discovery
   - Service/Resource Discovery
3. Taxonomy of P2P Systems
   - Centralized P2P Systems
   - Decentralized P2P Systems
   - Hybrid P2P Systems

Taxonomy of P2P Systems I.

Generally, P2P systems can be divided into two main categories:

- **centralized** – one or more central servers are available providing various services
- **decentralized** – no central servers are employed
  - they have to consider two main design issues:
    - *the structure* – flat (single tier) vs. hierarchical (multitier)
    - *the overlay topology* – unstructured vs. structured
- besides these two, **hybrid** P2P systems also exist
  - they combine both centralized and decentralized approach to leverage the advantages of both architectures

Taxonomy of P2P Systems II.

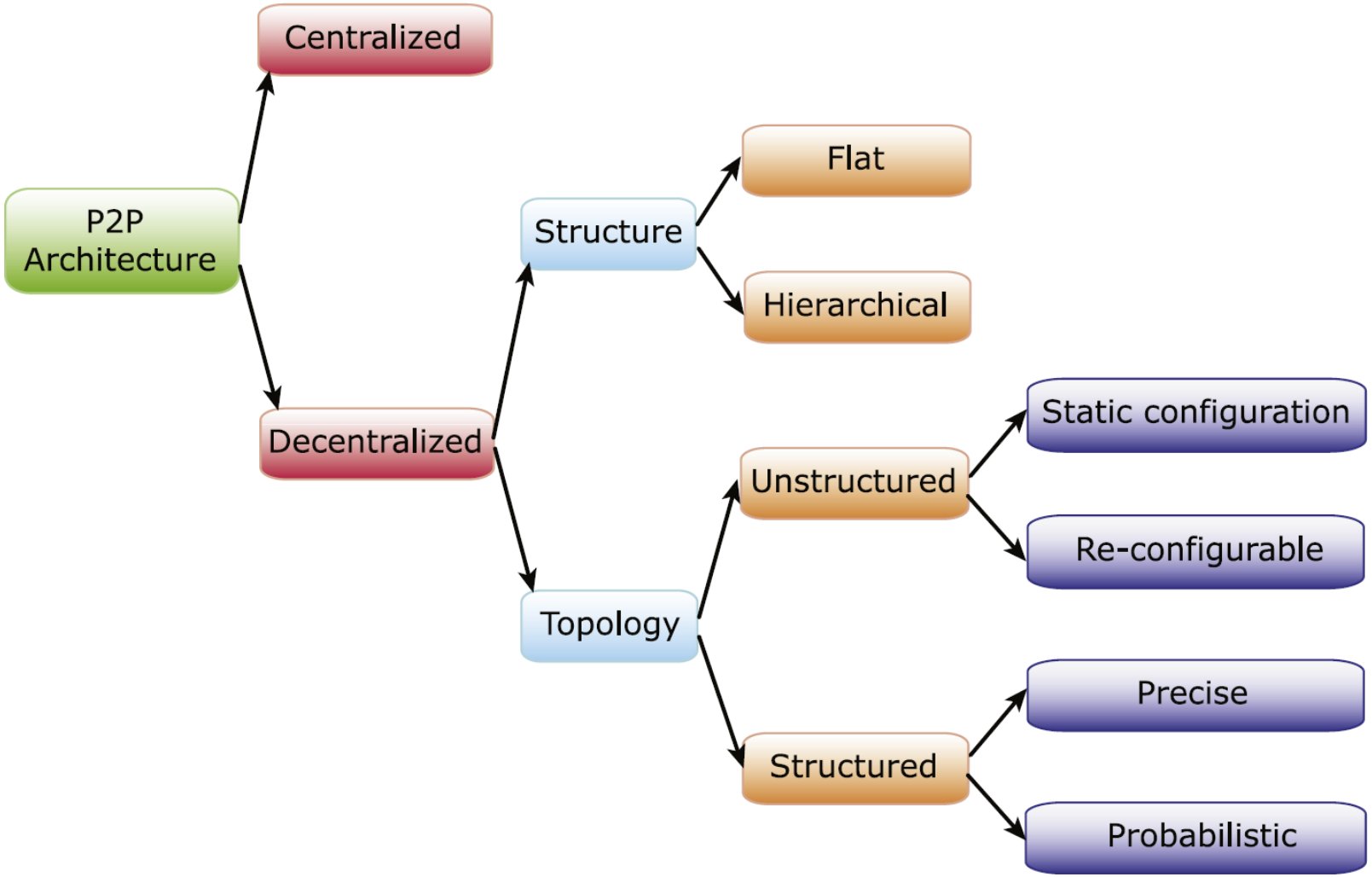


Figure: A taxonomy of P2P systems.

Taxonomy of P2P Systems III.

Centralized P2P Systems I.

Centralized P2P Systems

- combine the features of centralized (client-server) and decentralized systems
  - like a centralized system, there are one or more central servers, which help peers to locate their desired resources or act as task scheduler to coordinate actions among them
    - a peer sends messages to the central server to determine the addresses of peers that contain the desired resources
  - like a decentralized system, once a peer has its information/data, it can communicate directly with other peers
    - i.e., without going through the server anymore
- *drawbacks:*
  - susceptible to malicious attacks and single point of failure
  - a bottleneck for a large number of peers (performance degradation)
  - lacks scalability and robustness
- *examples:*
  - scientific computation – SETI@home, BOINC, Folding@home, Genome@home
  - digital content sharing – Napster, Openext
  - others – Jabber (IM), Net-Z and StarCraft (entertainment), etc.

Taxonomy of P2P Systems III.

Centralized P2P Systems II.

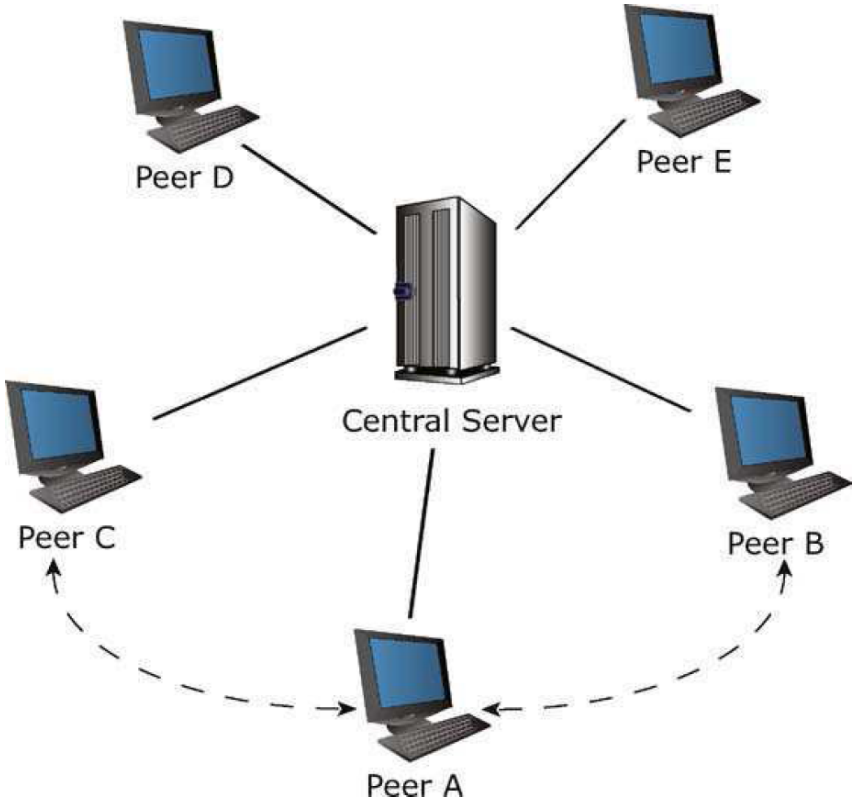


Figure: *Centralized P2P Systems:* Peer A submits a request to the central server to acquire a list of nodes that satisfy the request. Once it obtains the list (which contains Peers B and C), it communicates directly with them.

Taxonomy of P2P Systems III.

Decentralized P2P Systems I.

Decentralized (Pure) P2P Systems

- peers have equal rights and responsibilities
  - each peer has only a partial view of the P2P network and offers data/services that may be relevant to only some queries/peers
  - $\Rightarrow$ locating peers offering services/data quickly is a critical and challenging issue
- *advantages:*
  - immune to single point of failure
  - (usually) provide high performance, scalability, robustness, and other desirable features
- *examples:* Gnutella, Crescendo, PAST, FreeNet, Canon, etc.

Taxonomy of P2P Systems III.

Decentralized P2P Systems II.

Two dimensions in the design of decentralized P2P systems:

- **flat (single-tier)** vs. **hierarchical (multi-tier)** network structure
  - *flat structure* $\rightarrow$ the functionality and load are uniformly distributed among the participating nodes
  - *hierarchical structure* $\rightarrow$ multiple layers of routing structures
    - example: national level (interconnecting states), states level (interconnecting universities), universities level (interconnecting departments), etc.
    - offers certain advantages (fault isolation and security, effective caching and bandwidth utilization, hierarchical storage, etc.)

Taxonomy of P2P Systems III.

Decentralized P2P Systems II.

- **structured** vs. **unstructured** logical topology
  - *unstructured P2P system* → each peer is responsible for its own data, and keeps track of a set of neighbors that it may forward queries to
    - no strict mapping between the identifiers of objects and those of peers
    - ⇒ locating data is a challenge (its difficult to precisely predict which peers maintain the queried data)
    - ⇒ there is no guarantee on the completeness of answers (unless the entire network is searched)
    - ⇒ there is no guarantee on response time (except for the worst case where the entire network is searched)
  - *structured P2P system* → data placement is under the control of certain predefined strategies (generally, a *distributed hash table – DHT*)
    - there is a mapping between data and peers
    - ⇒ these systems can provide a guarantee (precise or probabilistic) on search cost
    - ⇒ however, typically at the expense of maintaining certain additional information
  - (systems employing a mix between structured and unstructured topology also exist)

Taxonomy of P2P Systems III.

Decentralized P2P Systems III.

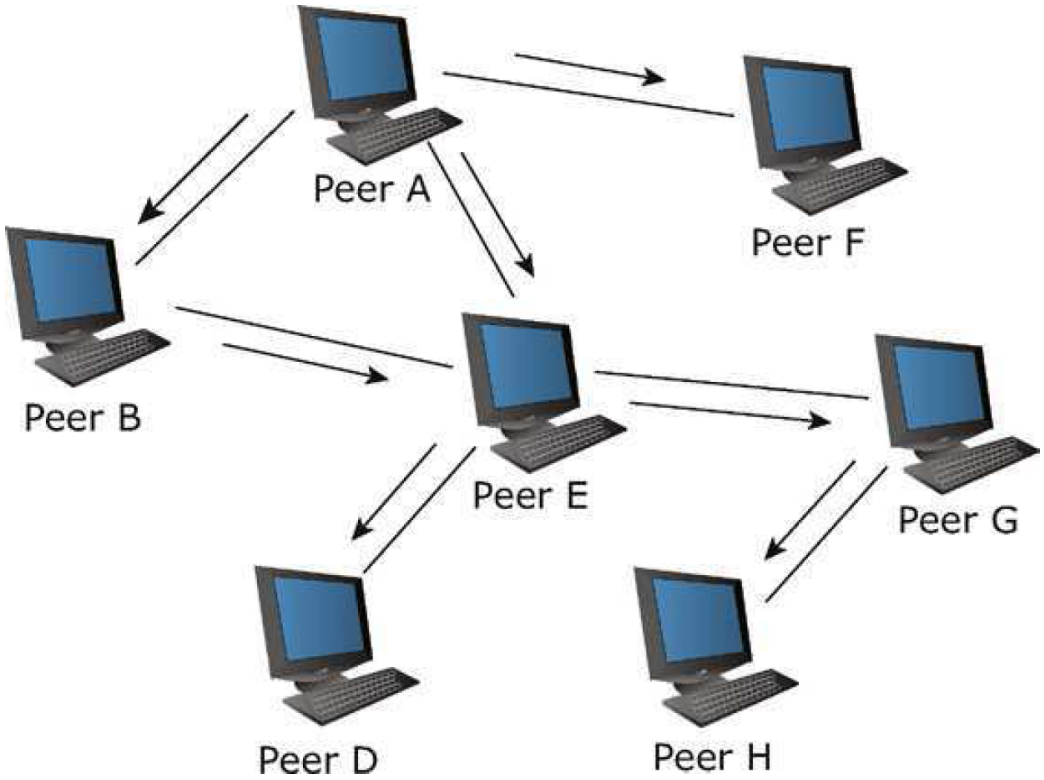


Figure: *Decentralized P2P Systems:* Peer A requests for some data that Peer D and Peer H have. The query will be broadcasted to the neighbors of Peer A, and gradually, to the other peers in the whole network (Gnutella).

Taxonomy of P2P Systems III.

Hybrid P2P Systems

Hybrid P2P Systems

- the main advantage of centralized P2P systems: quick and reliable resource locating
  - BUT with the limitation in terms of scalability
- the main advantage of decentralized P2P systems: scalability
  - BUT with the limitation in terms of longer time necessary for resource locating
- $\Rightarrow$ *Hybrid P2P systems*:
  - to maintain the scalability, there are no central servers
  - however, more powerful peer nodes are selected to act as servers to serve others
    - = *super peers*
  - $\Rightarrow$ resource locating can be done by both decentralized and centralized search techniques (asking super peers)

Taxonomy of P2P Systems III.

Hybrid P2P Systems III.

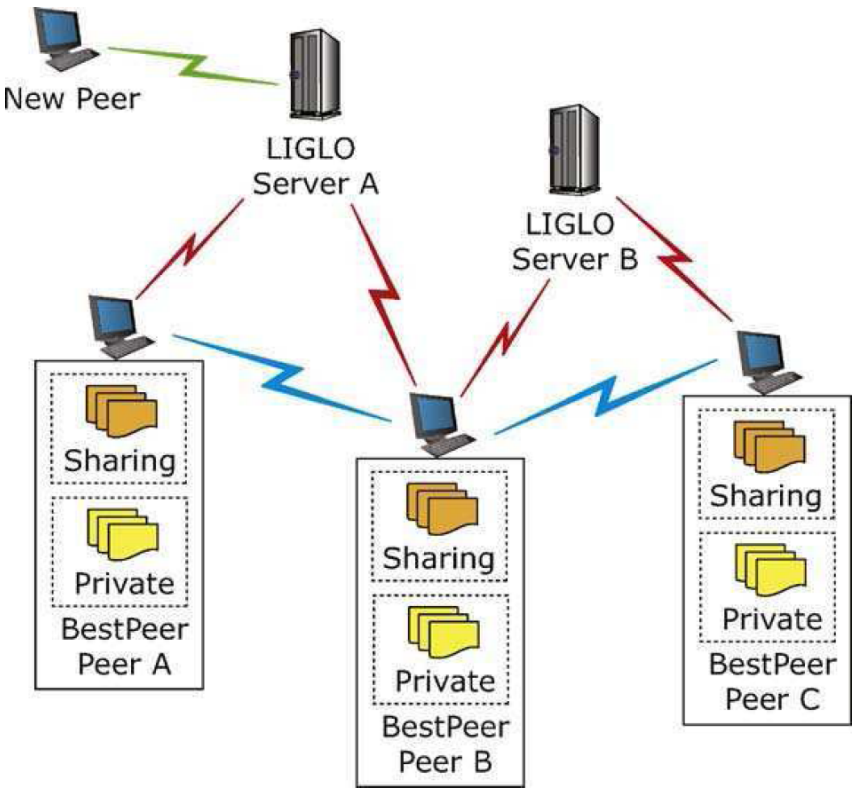


Figure: *Hybrid P2P Systems:* LIGLO servers are used to identify peers independently of their IP address (thus, even though a peer changes its IP address, the system still recognizes it as a unique peer) using a global and unique identifier. (BestPeer)