Peer to Peer Computing

Partially based on Nelson Minar's article at http://www.openp2p.com/pub/a/p2p/2002/01/08/p2p_topologies_pt2.html

What is Peer-to-Peer?

- 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.

Advantages of P2P Computing

- No central point of failure
  - E.g., 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 does have 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 of P2P Computing

- 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.

P2P Computing Applications

- File sharing
- Process sharing
- Collaborative environments

P2P File Sharing Applications

- Improves data availability
- Replication to compensate for failures.
- E.g., Napster, Gnutella, Freenet, KaZaA (FastTrack), your DFS project.

P2P Process Sharing Applications

- For large-scale computations
- Data analysis, data mining, scientific computing
- E.g., SETI@Home, Folding@Home, distributed.net, World-Wide Computer

P2P Collaborative Applications

- For remote real-time human collaboration.
- Instant messaging, virtual meetings, shared whiteboards, teleconferencing, tele-presence.
- E.g., talk, IRC, ICQ, AOL Messenger, Yahoo! Messenger, Jabber, MS Netmeeting, NCSA Habanero, Games

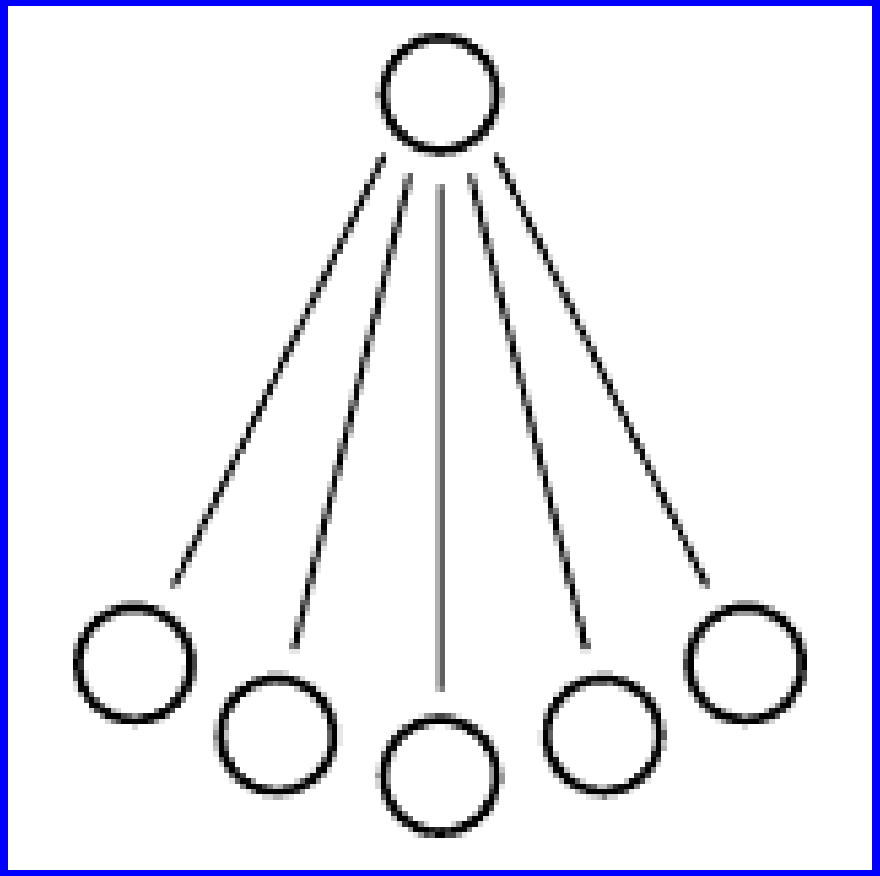
P2P Technical Challenges

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

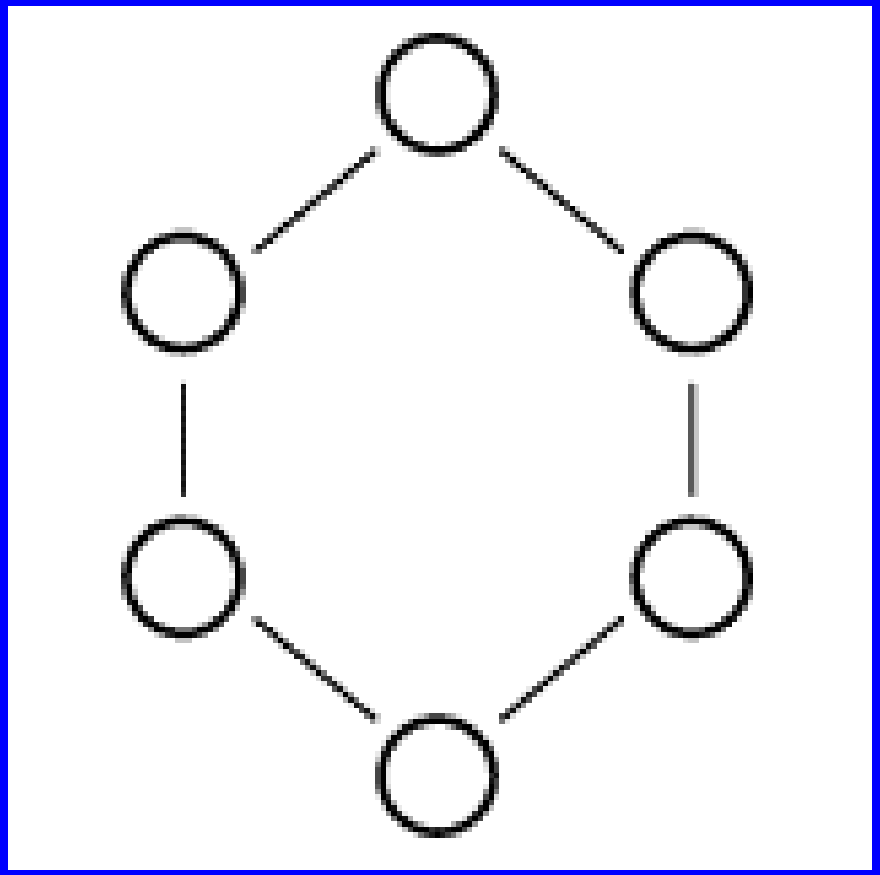
P2P Topologies

- Centralized
- Ring
- Hierarchical
- Decentralized
- Hybrid

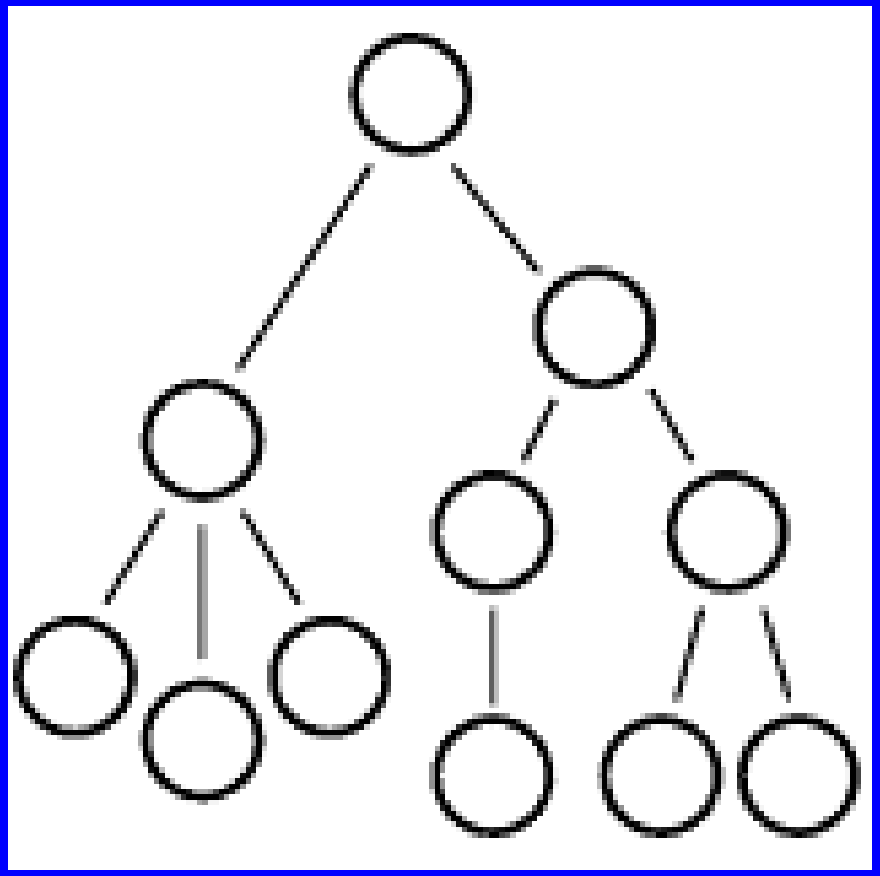
Centralized Topology



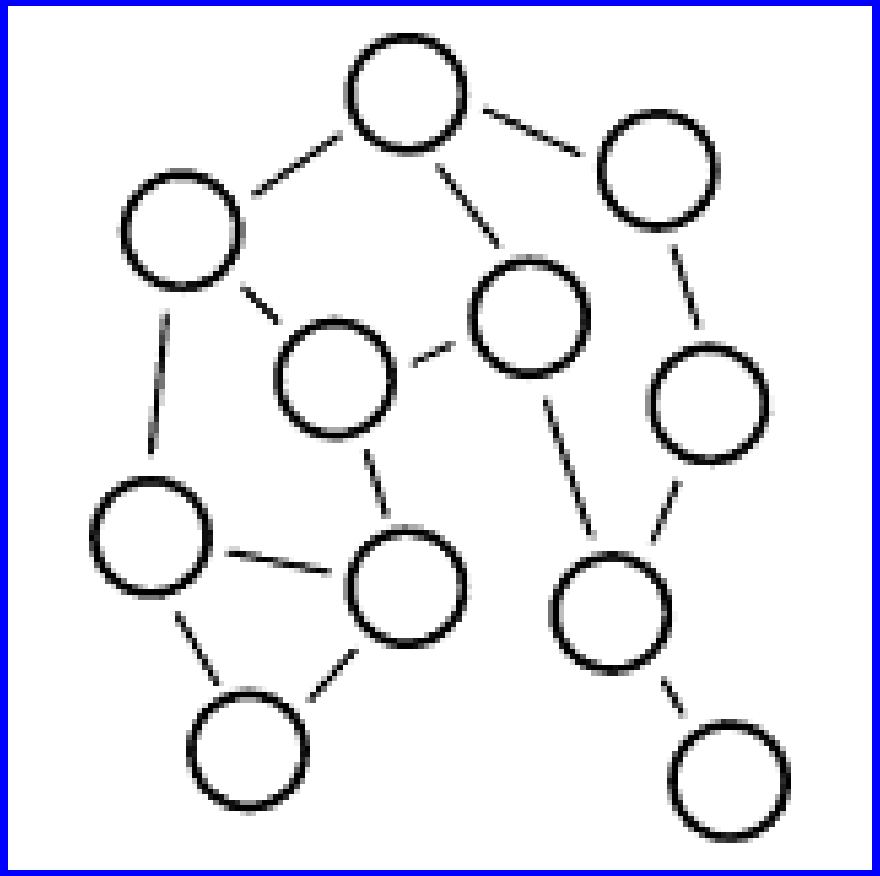
Ring Topology



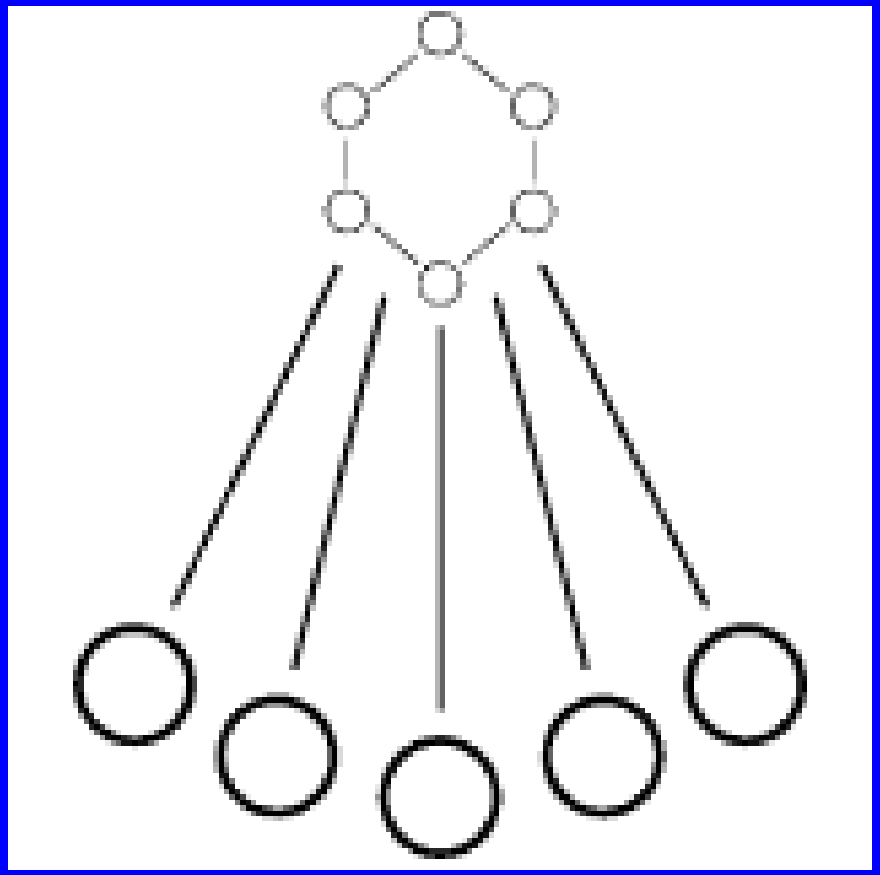
Hierarchical Topology



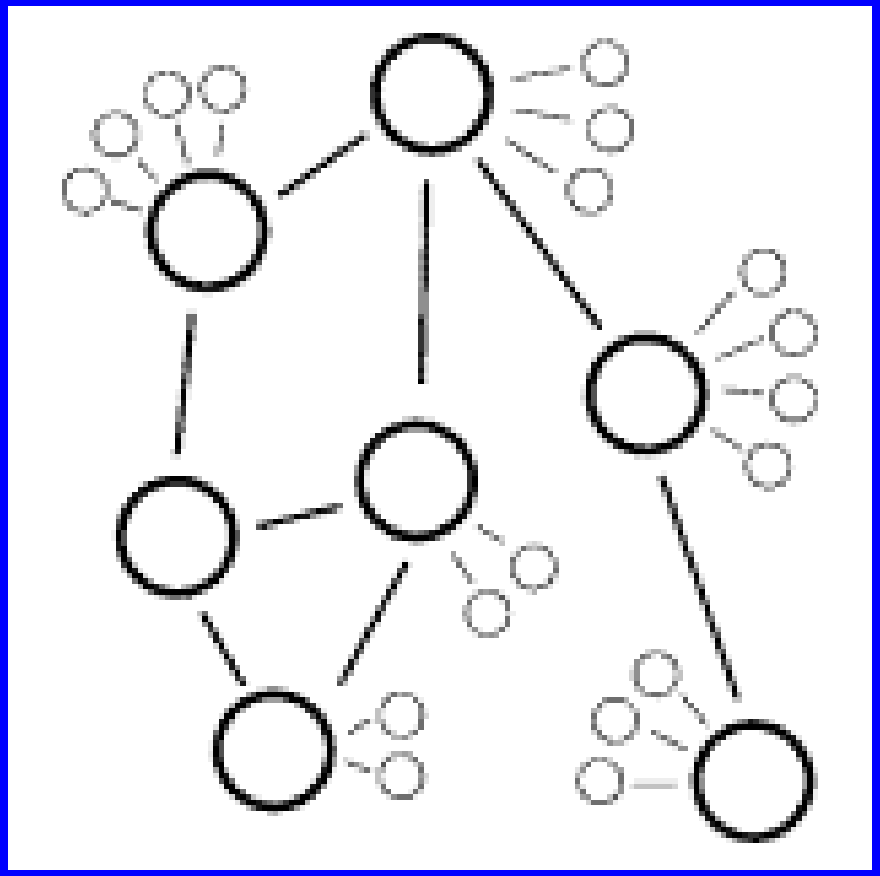
Decentralized Topology



Hybrid Topology
Centralized + Ring



Hybrid Topology
Centralized + Decentralized



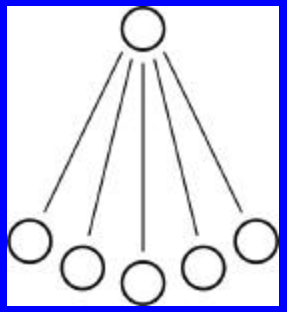
Evaluating topologies

- Manageability
  - How hard is it to keep working?
- Information coherence
  - How authoritative is info? (Auditing, non-repudiation)
- Extensibility
  - How easy is it to grow?
- Fault tolerance
  - How well can it handle failures?

Evaluating topologies

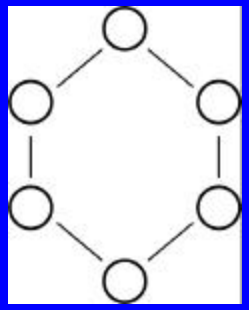
- Resistance to legal or political intervention
  - How hard is it to shut down? (Can be good or bad)
- Security
  - How hard is it to subvert?
- Scalability
  - How big can it grow?

Centralized



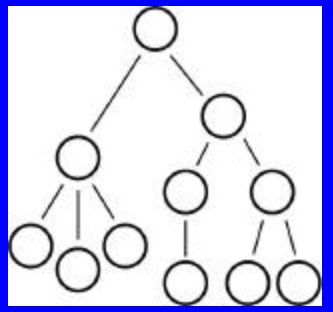
Manageable	✓	System is all in one place
Coherent	✓	All information is in one place
Extensible	X	No one can add on to system
Fault Tolerant	X	Single point of failure
Secure	✓	Simply secure one host
Lawsuit-proof	X	Easy to shut down
Scalable	?	One machine. But in practice?

Ring



Manageable	✓	Simple rules for relationships
Coherent	✓	Easy logic for state
Extensible	X	Only ring owner can add
Fault Tolerant	✓	Fail-over to next host
Secure	✓	As long as ring has one owner
Lawsuit-proof	X	Shut down owner
Scalable	✓	Just add more hosts

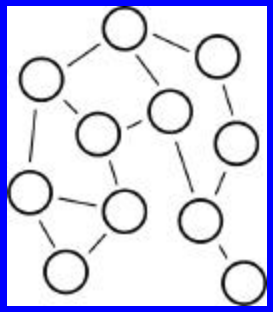
Hierarchical



Manageable
Coherent
Extensible
Fault Tolerant
Secure
Lawsuit-proof
Scalable

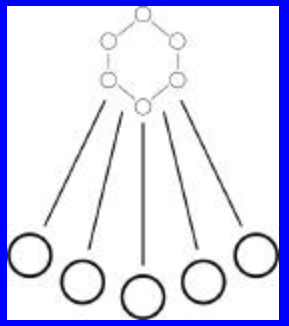
½ Chain of authority
½ Cache consistency
½ Add more leaves, rebalance
½ Root is vulnerable
X Too easy to spoof links
X Just shut down the root
✓ Hugely scalable – DNS

Decentralized



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

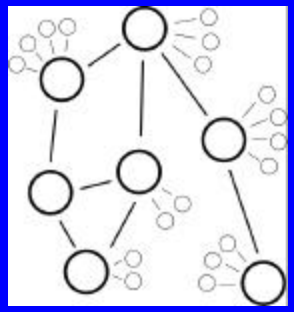
Centralized + Ring



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

Common architecture for web applications

Centralized + Decentralized



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

Best architecture for P2P networks?

Napster

- The P2P revolution is started.
- Central indexing and searching service
- File downloading in a peer-to-peer point-to-point manner.

Gnutella

- Peer-to-peer indexing and searching service.
- Peer-to-peer point-to-point file downloading using HTTP.
- A gnutella node needs a server (or a set of servers) to “start-up”… gnutellahosts.com provides a service with reliable initial connection points

But introduces a new single point of failure!

The Gnutella protocol (v0.4)

- PING – Notify a peer of your existence
- PONG – Reply to a PING request
- QUERY – Find a file in the network
- RESPONSE – Give the location of a file
- PUSHREQUEST – Request a server behind a firewall to push a file out to a client.

Freenet

- Peer-to-peer indexing and searching service.
- Peer-to-peer file downloading.
- Files served use the same route as searches (not point-to-point)
  - Provides for anonymity.

KaZaA/Morpheus

- Hybrid indexing/searching model
  - Not centralized like Napster, not decentralized like Gnutella.
- Peer-to-peer file downloading using HTTP.
  - “SmartStream” for incomplete file downloads.
  - “FastStream” for partial file downloads.
- “SuperNodes” elected dynamically if sufficient bandwidth and processing power – hybrid topology model.
- A central server keeps user registrations, logs usage, and helps bootstrapping peer discovery.