peer-to-peer and agent-based computing

P2P Algorithms & Issues

P2P: algorithms

- P2P systems have algorithms to control activities among peers.
- The algorithm specifies the workings of the system: what happens and when.
- Let's look at 3 models used in algorithms:
  - Centralised directory model
  - Flooded request model
  - Document routing model

P2P: models

Centralised directory model

1. Peers connect to the central directory and inform the contents they offer for sharing
2. Upon request from a peer, the central index will find the best peer in the directory that matches the request

   (best = cheapest, nearest, fastest, or most reliable)
3. The requesting peer, in possession of the best match for a peer, will contact this peer and, if agreed, a file exchange will take place

P2P: models (Cont'd)

Centralised directory model

- Requires infrastructure for directory server
- If we have lots of peers
  - A more powerful server must be used, AND
  - More storage is needed for the directory

that is, scalability issues…

- This is the model used by Napster
  - In spite of the theoretical limitations above, the model is robust and efficient.
  - It scaled up very well, with 6 million peers at some points.

P2P: models (Cont'd)

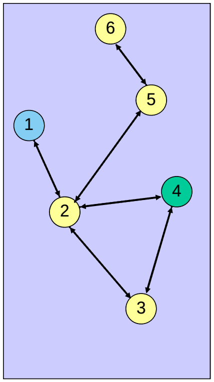
Flooded requests model

- Pure P2P:
  - No central servers
  - No advertisements of resources

P2P: models (Cont'd)

Flooded requests model

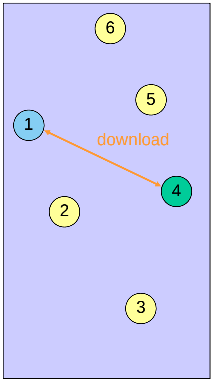
- A peer's request is flooded, that is, broadcast to its directly connected peers
- A request is flooded until
  - it is answered OR
  - a maximum number of flooding steps (5 to 9, typically) is reached



P2P: models (Cont’d)

Flooded requests model

- A peer’s request is flooded, that is, broadcast to its directly connected peers
- A request is flooded until
  - it is answered OR
  - a maximum number of flooding steps (5 to 9, typically) is reached



P2P: models (Cont’d)

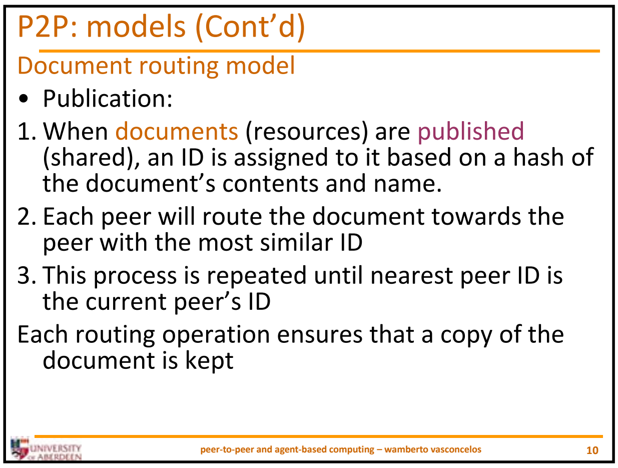
Flooded requests model

- Advantages:
  - No central point of failure
  - Limited per-node state
- Drawbacks:
  - Requires a lot of bandwidth
- Efficient in small communities:
  - A company Intranet, for instance.
- This is the model used by Gnutella

P2P: models (Cont’d)

Document routing model

- Most recent approach
- Also “pure” P2P – no central servers
- Each peer that joins the system is assigned a random ID number
- Each peer knows a number of peers



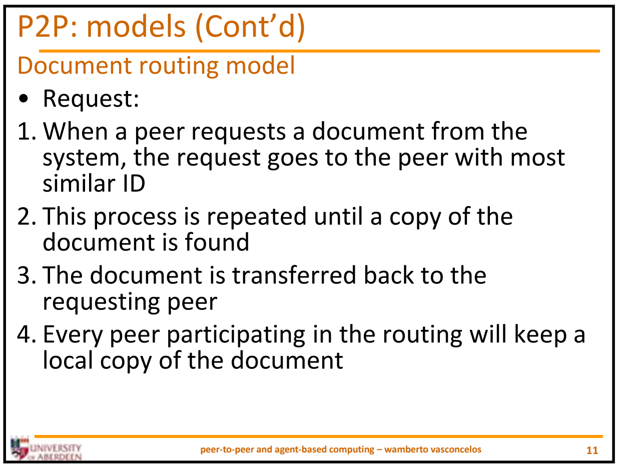
P2P: models (Cont'd)

Document routing model

- Publication:

1. When documents (resources) are published (shared), an ID is assigned to it based on a hash of the document's contents and name.
2. Each peer will route the document towards the peer with the most similar ID
3. This process is repeated until nearest peer ID is the current peer's ID

Each routing operation ensures that a copy of the document is kept

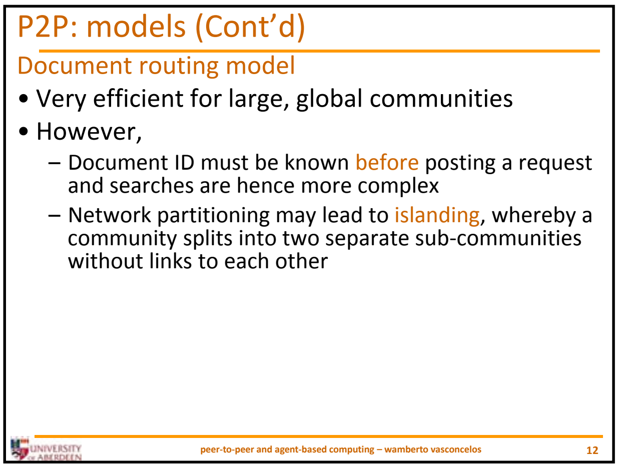


P2P: models (Cont'd)

Document routing model

- Request:

1. When a peer requests a document from the system, the request goes to the peer with most similar ID
2. This process is repeated until a copy of the document is found
3. The document is transferred back to the requesting peer
4. Every peer participating in the routing will keep a local copy of the document



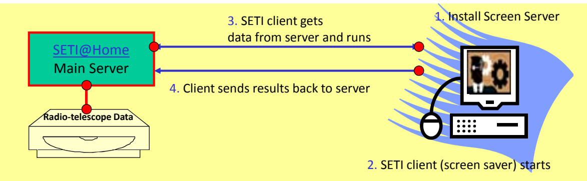
P2P: models (Cont'd)

Document routing model

- Very efficient for large, global communities
- However,
  - Document ID must be known before posting a request and searches are hence more complex
  - Network partitioning may lead to islanding, whereby a community splits into two separate sub-communities without links to each other

SETI@HOME

- Launched 1996, aims at scientific experiment
- Uses idle processors of connected computers to “Search for Extraterrestrial Intelligence” (SETI)
- Distributes a screen saver–based application to users
- Applies signal analysis algorithms different data sets to process radio-telescope data.
- Over 3 million users, provided over a million years of CPU time to date



3. SETI client gets data from server and runs

1. Install Screen Server

SETI@Home Main Server

4. Client sends results back to server

Radio-telescope Data

2. SETI client (screen saver) starts

P2P: issues

- Decentralisation
- Scalability
- Anonymity
- Self-organisation
- Cost of ownership
- Ad-hoc connectivity
- Performance
- Security
- Transparency and Usability
- Fault resilience
- Interoperability

P2P: issues

Decentralisation

- DIS still retain some degree of centralisation
  - Servers that receive requests from clients
- Centralised systems are ideal for some scenarios:
  - Management of access rights and security is easier
- However,
  - Topology leads to inefficiencies, bottlenecks
  - Centralised resources must be set up and managed, thus requiring human expertise and time

P2P: issues

Decentralisation

- In pure P2P systems
  - Emphasis is on users' ownership and control of resources
  - As there is no central server the implementation of a model is more complex
  - No global view of the system
- "Hybrid" P2P systems like Napster try to balance two opposite design forces:
  - Decentralised, distributed model, BUT with
  - Global view of resources

P2P: issues

Scalability

- How many components can be catered for?
- Decentralisation improves scalability:
  - Synchronisation/coordination not necessary
  - No global states to look after
- Pure P2P copes with larger systems, BUT
  - No guarantees about the time it takes for a search to complete
  - No guarantees about unsuccessful searches
- A certain amount of centralisation goes a long way…

P2P: issues

Anonymity

- People should have access to resources without fears (legal, social, etc.)
- Censorship of digital content should not be possible
- Three kinds of anonymity:
  - Sender anonymity
  - Receiver anonymity
  - Mutual anonymity
- Different techniques to ensure anonymity:
  - E.g., identity spoofing, covert paths

P2P: issues

Self-organisation

- Systems that change their organisation without any previously agreed policy.
- P2P: self-organisation is essential for
  - Scalability
  - Fault resilience
  - Cost of ownership
- Self-organisation in P2P occurs when
  - Peers appear/disappear continuously
  - The topology (who knows who) changes to reflect this
  - Resources appear/disappear or are copied

P2P: issues

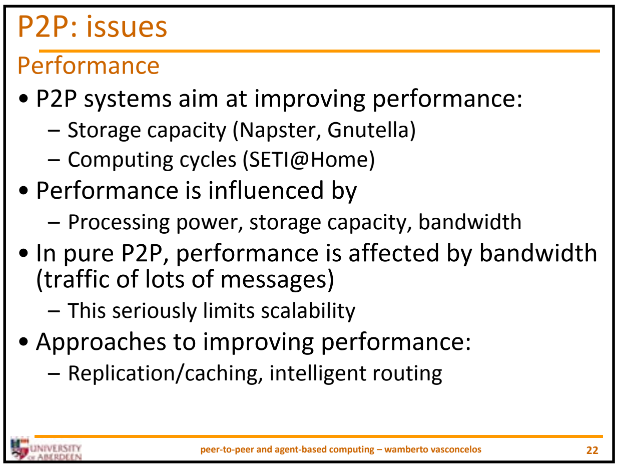
Cost of ownership

- It is expensive to own a resource:
  - File space, CPU time, bandwidth
- In P2P systems ownership is shared:
  - Peers offer file space and time for downloading
  - Peers inform about updates
  - Peers offer computing time to process tasks

P2P: issues

Ad-hoc connectivity

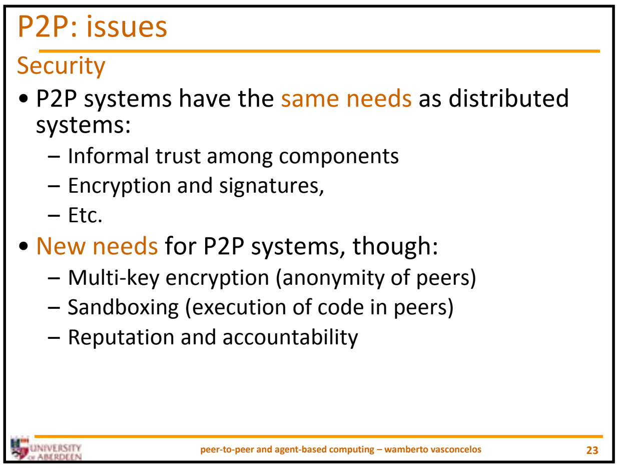
- Some components of a large system may not be available at (crucial) times…
- In P2P, this is the usual scenario:
  - In content sharing systems, users expect resources to be available intermittently
- Ad hoc effect can be reduced: redundancy



P2P: issues

Performance

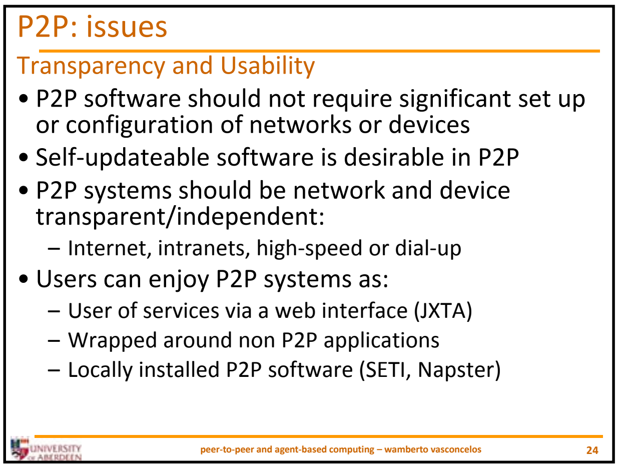
- P2P systems aim at improving performance:
  - Storage capacity (Napster, Gnutella)
  - Computing cycles (SETI@Home)
- Performance is influenced by
  - Processing power, storage capacity, bandwidth
- In pure P2P, performance is affected by bandwidth (traffic of lots of messages)
  - This seriously limits scalability
- Approaches to improving performance:
  - Replication/caching, intelligent routing



P2P: issues

Security

- P2P systems have the same needs as distributed systems:
  - Informal trust among components
  - Encryption and signatures,
  - Etc.
- New needs for P2P systems, though:
  - Multi-key encryption (anonymity of peers)
  - Sandboxing (execution of code in peers)
  - Reputation and accountability



P2P: issues

Transparency and Usability

- P2P software should not require significant set up or configuration of networks or devices
- Self-updateable software is desirable in P2P
- P2P systems should be network and device transparent/independent:
  - Internet, intranets, high-speed or dial-up
- Users can enjoy P2P systems as:
  - User of services via a web interface (JXTA)
  - Wrapped around non P2P applications
  - Locally installed P2P software (SETI, Napster)

P2P: issues

Fault resilience

- Ability to carry on when things break down
- Some P2P systems allow computations to carry on where they were left:
  - Relay nodes in Groove (www.groove.net)
  - This is particularly important when peers are mobile and dis/re-connection is more likely to happen
- Resources may go missing if peer is disconnected:
  - Replication of resources (as in Gnutella)

P2P: issues

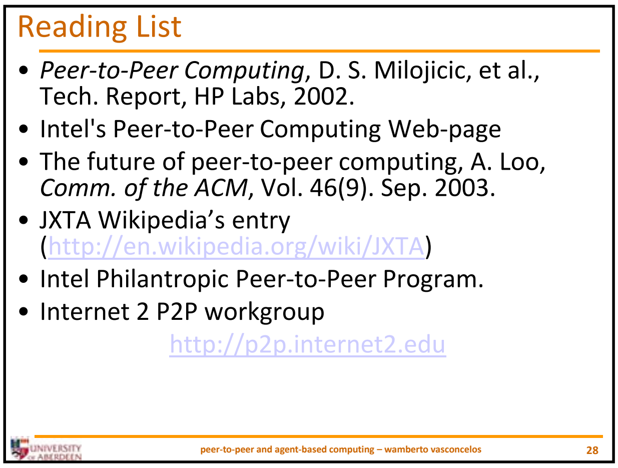
Interoperability

- Many existing P2P systems, but they don’t interoperate
- Some issues that ought to be solved are:
  - Which protocol to be used?
  - How do requests take place?
  - What about models that are too different?

P2P: some existing applications

- JXTA (jxta.kenai.com)
- Gnutella (http://www.gnutellaforums.com/)
- We should look at these from the following perspective:

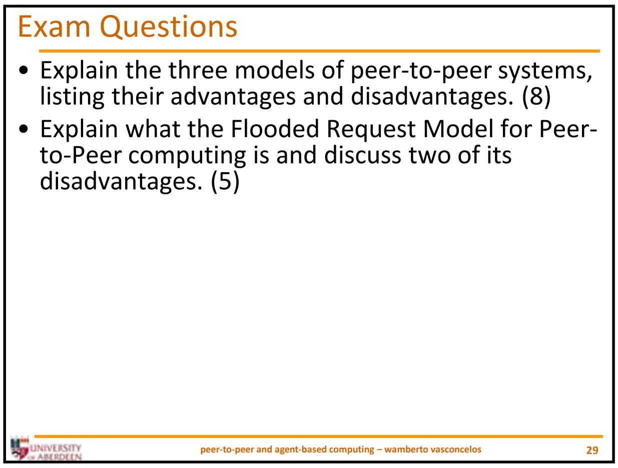
  “how can this technology be used as part of an integrated solution for a distributed information system?”



Reading List

- *Peer-to-Peer Computing*, D. S. Milojicic, et al., Tech. Report, HP Labs, 2002.
- Intel's Peer-to-Peer Computing Web-page
- The future of peer-to-peer computing, A. Loo, *Comm. of the ACM*, Vol. 46(9). Sep. 2003.
- JXTA Wikipedia's entry (http://en.wikipedia.org/wiki/JXTA)
- Intel Philantropic Peer-to-Peer Program.
- Internet 2 P2P workgroup

http://p2p.internet2.edu



Exam Questions

- Explain the three models of peer-to-peer systems, listing their advantages and disadvantages. (8)
- Explain what the Flooded Request Model for Peer-to-Peer computing is and discuss two of its disadvantages. (5)