Peer-to-Peer Systems

Ruixuan Li
School of Computer Science and Technology, HUST
http://idc.hust.edu.cn/~rxli/

Outline
- What is P2P Computing
- P2P Topologies
- P2P Systems and Search
- P2P Issues

Rise and Fall of P2P in Media
- “Peer-to-peer is the next great thing for Internet” Stanford Law Net Guru – Lawrence Lessig, 2000
- “Peer-to-peer computing is leading us into the 3rd age of the internet” Bob Knighton – Inter Corp, Fall 2000
- “Is P2P plunging off the deep end” Wall Street Journal, April 2, 2001
- “Does Peer-to-Peer Suck?” Jon Katz – Slashdot, April 4, 2001

P2P Streaming in China

P2P in USA
What Do We Believe About P2P?
- Is it a new technology?
- Is it about getting movie for free?
- Is it only hype?
- Is it a good way to make money?
- Is it against the law?

What’s P2P Computing

Recap Client/Server Definitions
- **Server:**
  - An entity that serves requests from other entities but does not initiate requests.
- **Client:**
  - An entity that initiates requests but is not able to serve requests.

The Contrast: Server-Centric
- The Client/Server paradigm
  - The client is basically a glorified I/O device
    - Information, Control, Computation is kept at the Server
    - Simpler to build centralized systems

Client/Server ‘Problems.COM’
- A Typical Client-Server architecture:
  - Server is a “Super Computer”
  - Addresses/Ports of servers are known
  - MANY clients to ONE server
  - Client is just a “Monitor”
  - Server is down = Network is down
  - Server is Expensive
  - Scalability: More clients=More Servers

Client/Server Solutions
- **Replication** (Many Servers)
  - **Drawbacks:** Expensive, Synchronization and much more…
- **Brute Force** (Faster Server)
  - **Drawbacks:** Expensive, Scalability, Single point of failure
Beyond Client/Server World: Internet

- The rapid growing Internet services are provided by an increasing number of peers.
- Variety of devices: from smartphones, pads, laptops, PCs to supercomputer centers.
- Pervasive computing: access information and services anytime and anywhere.

Client/Server Model is Being Challenged

- No single server or search engine can sufficiently cover increasing Web contents.
- \(2 \times 10^{18}\) Bytes/year generated in Internet.
- But only \(3 \times 10^{12}\) Bytes/year available to public (0.00015%).
- Google only searches \(1.3 \times 10^8\) Web pages (4\(\times 10^{10}\), 2008).
  
  (Source: IEEE Internet Computing)

Client/Server (continued)

Client/server model seriously limits utilization of available bandwidth and services.
- Popular servers and search engines become traffic bottlenecks.
- But high speed networks connecting many clients become idle.
- Computing cycles and information in clients are ignored.

Content Delivery Networks (CDN): A Transition Model

- Servers are decentralized (duplicated) throughout the Internet.
- The distributed servers are controlled by a centralized authority (headquarters).
- Examples: Internet content distributions by Akamai, Overcast, and FFnet.
- Both Client/Server and CDN models have single point of failures.

A New Paradigm: Peer-assisted Systems

- Both client (consumer) & server (producer).
- Has the freedom to join and leave any time.
- Huge peer diversity: service ability, storage space, networking speed, and service demand.
- A widely decentralized system opening for both opportunities and new concerns.

Peer-assisted Systems

- Client/server
- Content Delivery Networks
- Pure P2P
- Hybrid P2P
- e.g. Freenet & Gnutella
- e.g. Akamai, Overcast, FFnet
- e.g. Napster
So, What Is a “Peer”?

- Merriam-Webster:
  - one that is of equal standing with another

- Cambridge:
  - a person who is the same age or has the same social position or the same abilities as other people in a group

What Is a “Peer”?

- Computer and Communication:
  - any end system at the edge of the Internet

What’s P2P Computing?

- Peer-to-Peer Computing – Computing between equals

Resource Sharing

- Exploit idle resources available in the edges
  - E.g. CPU idle cycles, unused storage space,…

- Exploit plentiful resources among network edges
  - E.g. spare network bandwidth

- Federated cooperation among companies
  - Sharing unavailable resources (e.g. databases)

- …

First Definition: Peer-to-Peer Network

- A distributed network architecture may be called a Peer-to-Peer network, if the participants share a part of their own hardware resources (processing power, storage capacity, network link capacity, printers).
  - These shared resources are necessary to provide the Service and content offered by the network (e.g. file sharing or shared workspaces for collaboration).
  - They are accessible by other peers directly, without passing intermediary entities.
  - The participants of such a network are thus resource (Service and content) providers as well as resource (Service and content) requestors (Served).


Second: Pure Peer-to-Peer

- A distributed network architecture has to be classified as a Pure Peer-to-Peer network, if it is firstly a Peer-to-Peer network and secondly if any single, arbitrary chosen Terminal Entity can be removed from the network without having the network suffering any loss of network service.

Third: Hybrid Peer-to-Peer

- A distributed network architecture has to be classified as a Hybrid Peer-to-Peer network, if it is firstly a Peer-to-Peer network and secondly a central entity is necessary to provide parts of the offered network services.


P2P Application

Application is P2P if:
- Allows for variable connectivity & temporary network addresses
- Gives the nodes at the edges of the network significant autonomy

Another Point Of View

- In P2P, peers in relation to each other act as:
  - Clients AND
  - Servers AND
  - Routers AND
  - Caches AND... EVERYTHING

First Question

- Is Peer-to-Peer a new Technology?
  - ABSOLUTELY... NOT!

History of P2P

- P2P has been seen as the most natural approach in early distributed applications
- E-mail system built on top of SMTP
- Usenet News
- FTP, even if a server/client application, was running as P2P

History of P2P

- Origin of P2P dates back to ARPANET
- Early P2P applications/servers is Usenet and DNS
- 1990s: Shift in paradigm to client-server
- 1999: Napster => explosion of P2P usage
- 2000s: Gnutella, KaZaA, Audiogalaxy, etc.
Objectives and Benefits of P2P

- As long as there is no physical break in the network, the target file will always be found.
- Adding more contents to P2P will not affect its performance. (information scalability).
- Adding and removing nodes from P2P will not affect its performance. (system scalability).

What makes it so hyped?

- Industry looking for something positive after .Com death
- Has large social consequences
  - The Internet has already changed society
  - We can expect further changes
- Some very interesting applications became widely known and used
  - Napster, iMesh, Gnutella, FreeNet, KaZaA, Morpheus, CuteMX, Scour, BitTorrent, PPLive, ...

Terminology Related to P2P

Distributed Computing
- A computer system in which several interconnected computers share the computing task assigned to the system

Grid Computing
- An infrastructure for globally sharing compute-intensive resources

Ad-hoc system
- A system that enables communication to take place without any preexisting infrastructure in place

Edge Computing
- adaptively, timely, and (temporally) move contents and computing resources from centralized centers to sites (edge) close to the end-users

Terminology Related to P2P

And the usual...
- Centralized system
- Distributed system
- Client
- Server

P2P System Taxonomy
P2P Systems - Distributed Computing

- The idea is to achieve **processing scalability** by aggregating the resources of large number of individual computers.
  - The computational problem to be solved is **split** into several small independent part.
  - The processing of a single part is done on an individual workstation and the results are **collected** in a central server.
- Examples: SETI@Home, distributed.net

P2P Systems – File Sharing

- **Content storage and exchange** is one of the most successful area of P2P. The (legal) advantages of file sharing are:
  - **Potentially unlimited storage area**
    - A file is stored on a node but all the P2P community can retrieve it (Gnutella, Freenet, KaZaA)
  - **Highly available safe storage**
    - The file can be replicated multiple times ensuring a high level of availability (OceanStore, Chord)
  - **Anonymity**
    - Some systems ensure the anonymity of the authors and publishers of documents (Publius)

P2P Systems – Collaboration

- Collaborative P2P applications allows ** applications-level collaboration** between users.
  - Peers form a group of two or more users and begin a task.
  - When a change occurs at one peer (for example, one peer enters the group), an event is generated and sent to the rest of the peers.
- Examples of applications: instant messaging and chat, online games

P2P Systems – Platforms

- Many **middleware** solutions are increasingly becoming more important for applications development and deployment.
  - Java and Web browsers are examples.
  - P2P systems have a natural support for such environments, like naming, discovery and security.
- One example of P2P Platform is JXTA.

P2P Topologies

- Centralized
- Hierarchical
- Decentralized
- Decentralized with Super Nodes
- Hash Circle
Topologies - Centralized

- Like Client-Server, many clients and one server entity (one server/group of servers)
- Used in Napster
- Server acts like "114", just helps to initiate the communication
- Simple to design

Topologies - Centralized

- Ways of action:
  - Client sends server the query, server asks everyone and responds to client
  - Client gets list of clients from server
  - All Clients send ID's of the data they hold to the server and when client asks for data, server responds with specific addresses

Topologies - Hierarchical

- Servers are organized in a tree
- Suits for communication between "hierarchical objects" like companies, organizations – Inside P2P, Outside Client-Server
- Suits for security architectures like Certificate Authority

Topologies - Hierarchical

- Ways of action:
  - Much like the Centralized topology
  - Can set policy rules at the level of servers
  - Server sends the queries to his ancestor when needed
Topologies - Decentralized

- It’s the “Pure” P2P topology
- No servers (well, maybe just one !)
- Topology changes as peers are joining/leaving the network
- Mainly, the topology is really based on the “logical” behavior of the peers

Ways of action:
- Peer sends requests to his “neighbors”
- Neighbors route the requests to their neighbors - Flooding
- Many message could drop since “weak” peers might not work as fast as needed
- In future, special algorithms will dictate the behavior of this topology

Topologies – Decent’ + Super Nodes

- New topology
- Still – no servers (not expensive ones at least)
- Used in iMesh, KaZaA
- Slow peers do not slow the search

Ways of action:
- A super node is a normal node that’s elected to act as a local server
- Usually super nodes are elected for their bandwidth
- Requests are forwarded from slow peers to super nodes
Topologies – Hash Circle

- Mainly for file-sharing, storage-distribution
- All resources are represented by a hash value
- Only “Exact” searches are allowed

Ways of action:
- When a peer joins, it gets responsibility for part of the hash space
- Each peer knows his neighbors in the hash space and a few other randomly chosen peers
- Requests are forwarded to the node closest to the hash query
- Requires O(logN) forwards = low bandwidth

P2P Network Infrastructure

- Overlay networks: peers communicate to each other in the application layer.
  - Making friends with an IP address globally without considering distance, message types, low level protocols used.
  - Peers are not required to understand physical networks, creating a new domain of development opportunities.

More on Overlay Networks

- Overlay Graph: each edge is a TCP connection or a pointer to an IP address.
- Overlay Maintenance:
  - (1) periodically ping to verify liveness of peers.
  - (2) delete the edge with a dead peer.
  - (3) new peer needs to bootstrap.
- Overlay Problems:
  - (1) topology-unaware.
  - (2) duplicated messages.
  - (3) inefficient network usage.

P2P Types and Operations

- Directory-based P2P: a centralized index server makes a direct map between a pair of requesting and serving peers, e.g. Napster.
- Unstructured P2P: peers are randomly connected in overlay graph, flooding for queries/retrievals, e.g. Gnutella, and KaZaA.
- Structured P2P: peers are objectively connected in overlay graph by a Distributed Hash Table for registrations and queries/retrievals, e.g. Chord, CAN, FAN
P2P Systems and Search

Example P2P Systems
- Napster (Directory-based)
- Gnutella (Unstructured)
- KaZaA (Unstructured & Super Node)
- Skype (Unstructured & Super Node)
- CAN, Chord, FAN (Structured: DHT)
- SETI@home (Distributed Computing)

Directory-based P2P of Sharing Music: Napster
- Centralized
- MP3 file sharing
- Clients/Peers hold the files
- Servers hold catalog and broker relationships
  - Clients upload IP address, music file shared, and requests
  - Clients request locations where requests can be met
- File transfer is P2P-proprietary protocol

Napster (cont)

How does Napster Work (very simple!)
- Application-level:
  - client/server protocol over point-to-point TCP/IP.
  - central directory server.
- User operation steps:
  - connect to Napster server (www.napster.com)
  - upload a request list and the IP address in the server.
  - Index server searches the list and returns results to the IP.
  - User pings the music hosts, looking for best transfer rate.
  - User chooses a music provider for data transfer.
- The index server does not scale its P2P system.

Brief History and Implication of Napster
- 1999/1: Shawn Fanning (freshman, Northeastern), dropped out and started it.
- 1999/6: Napster began operations for swapping music among peers.
- 1999/12: lawsuit on copyright violation (RIAA), asking for $100K of each.
- 2000/3: universities ban it due to heavy traffic, e.g., 25% traffic in Uwisc.
- 2000/5: VC firm Hummer Winblad invested $15 million to Napster.
- 2000/7/26: US District judge orders to stop Napster’s operations in 2 days.
- 2000/7/28: 9th US Circuit Appeals Court rules it is allowed to continue.
- 2000/12: Federal Appeals Court rules it must stop trading copyrighted music.
- 2001/9: It reaches a settlement with music writers/publishers: pay $30 M for the past damage and a % to them as it starts as a paying service in 2002.
- 2008: Best Buy
- 2011/10: Rhapsody
Unstructured P2P: Gnutella

- Completely decentralized – no servers with catalogs
- Shares any files
- Gnutella node ---- SERVENT
  - Issue the query and view search result
  - Accept the query from other SERVENTs and check the match against its database and response with corresponding result

Gnutella (cont)

- Gnutella broadcasts its messages.
- To control flooding - TTL is introduced.
- To prevent forwarding same msg twice - each servent maintains a list of recently seen msgs.

Gnutella (cont)

- Problems
  - Broadcast msgs congest the network
  - Lost of reply packets (dynamic environment)

The fundamental problem

- Fundamental problem: search for a particular item in a network of peers without centralized servers
- Question: Where do you send your queries?
- Strategies:
  - Controlled flooding
  - Gossip
  - Random walker

Controlled Flooding

- Send the query for a particular item to known peers (i.e., neighbours)
  - The query will be forwarded by these neighbours to their own neighbours
  - Until the query finds itself to the peer node hosting the item, or until a particular TTL (Time-to-Live) has been reached
  - Those who have seen the original query will not forward it again
  - The query will not be sent back to where it came from (controlled flooding)
Controlled Flooding

- **Pros**
  - the search is fast, since it is an expanded ring search. For a network of size $n$, it takes approximately $\log n$ hops to reach the item

- **Cons**
  - too much messaging overhead for one search

- **Future work**
  - How do we reduce such messaging overhead?

Gossip

- **Pros**
  - Modify the forwarding mechanism in controlled flooding, so that each peer node will only forward the query with a certain probability

- **Cons**
  - reduced message overhead
  - may not guarantee that the item can be found, if it exists

Random walker

- **Pros**
  - reduced message overhead

- **Cons**
  - may not guarantee that the item can be found, if it exists

- **Future work**
  - The problem can be mitigated by using $k$ parallel random walkers, rather than just one random walker
  - The problem can also be mitigated by replicating the item to more peer nodes, increasing its exposure to random walkers

Super Node based P2P: KaZaA

- **Pros**
  - Hybrid between centralized and decentralized
  - Has 2 tiers of control

- **Cons**
  - Ordinary nodes that connect to super nodes in a centralized fashion
  - Super nodes that connect to each other in a decentralized manner

KaZaA (cont)
KaZaA (cont)

- Joining the network? - **Bootstrapping** node
- Querying?
- Problems (Like Gnutella)
  - Broadcast msg between **Super Nodes**
  - Lots of reply packets

Super Node based P2P: Skype

- Inherently P2P: pairs of users communicate.
- Proprietary application-layer protocol
- Hierarchical overlay with SNs
- Index maps usernames to IP addresses; distributed over SNs

Structured P2P: Distributed Hash Table (DHT)

- Nodes and names have keys, which are large integers. You get the key from the name by hashing. Nodes get keys (called IDs) by some way, usually just random.
- Given a name, hash it to the key.
- Now find the node that is responsible for that key. It is the node that has an ID >= key.

Standard Hashing

![Standard Hashing Diagram](image)

- Hashing function: \( H(a) = a \mod M \)
  - \( a \): numerical ID
  - \( M \): Hashing table size

Basic Hashing Operations

- **Insert** \((a, S)\): insert object \(a\) to Set \(S\).
  - Compute \(h(a)\);
  - Search the list pointed by Table \([h(a)]\); if \(a\) is not on the list, it is appended in the list.
- **Delete** \((a, S)\): delete object \(a\) from set \(S\).
  - Search the list pointed by Table \([h(a)]\); if found, delete object \(a\) in the list.
- **Find** \((a, S)\): find object \(a\) in Set \(S\).
  - Search the list pointed by Table \([h(a)]\); if \(a\) is on the list, returns its location, otherwise returns Null.

Distributed Hash Table (DHT)

- Problem: given an object stored in a node or multiple nodes, find it.
- The **Lookup** problem (Find \((a, S)\)): \(S\) is distributed and stored in many nodes.
  - Returns the network location of the node currently responsible for the given key.
- Take a 2-d CAN as an example
From Hash Table to DHT

Distributed Hash Table (DHT)

Insert(k1, v1)
Retrive(k1)
"Core" questions when introducing "distributed":
- How to divide a whole hash table to multiple distributed hash tables?
- How to reach the hash table who has the key I want, if I cannot find it from the local hash table?

Requirements:
- Data should be identified using unique numeric keys using hash function such as SHA-1 (Secure Hash Algorithm)
- Nodes should be willing to store keys for each other

Structured P2P: Chord
  
  Google Scholar: cited by 10009 times
  Microsoft Academic: cited by 5690 times

Simple Key Location Scheme
- The mapping of data items onto nodes in Chord.

Scalable Lookup Scheme
- Resolving key 26 from node 1 and key 12 from node 28 in a Chord system.

Distributed Hash Tables
- Finger [k] = first node that succeeds \((n+2^k) \mod 2^m\)
**FAN: Flabellate Overlay Network**


**Improved FAN construction**

- Peer stores the routing information of its adjacent subspace
- Use the super-peer to manage the routing information for its subspace

**Routing table at peers**

<table>
<thead>
<tr>
<th>Peer coordinates</th>
<th>Peer address</th>
<th>Subspace range</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,0.5,1,1.5,1,1)</td>
<td>211.69.192.70:9705</td>
<td>(5,10)</td>
</tr>
<tr>
<td>(0,0.5,2,1.5,1)</td>
<td>211.69.192.80:9705</td>
<td>(5,10)</td>
</tr>
<tr>
<td>(1,1,1,1.4,0,1.5)</td>
<td>211.69.202.18:9706</td>
<td>(5,10)</td>
</tr>
<tr>
<td>(0.5,1,1.5,0,1.5)</td>
<td>211.72.101.54:9706</td>
<td>(5,10)</td>
</tr>
<tr>
<td>(1,0,0.5,0.5,1)</td>
<td>211.80.102.79:9705</td>
<td>(3,5)</td>
</tr>
<tr>
<td>(2,2.5,1.5,0,2)</td>
<td>211.82.101.78:9705</td>
<td>(10,18)</td>
</tr>
</tbody>
</table>

**Example of FAN routing**

- Routing efficiency: $O(N/k)$
- Routing efficiency: $O(\log(N/k))$

**P2P Issues**
Some key issues

- **Scalability**
  - Networks can grow to millions of nodes
  - Challenge in achieving efficient peer and resource discovery
  - High amount of query/response traffic

- **Availability**
  - Potential for commercial content provision
  - Such services require high availability and accessibility

- **Anonymity**
  - What is the right level of anonymity?

**Some key issues (cont)**

- **Security**
  - Due to open nature, have to assume environment is hostile
  - Concerns include:
    - Privacy and anonymity
    - File authenticity
    - Threats like worms and virus

- **Fault Resilience**
  - The system must still be able to function even though several important nodes go offline.

**Some key issues (cont)**

- **Standards and Interoperability**
  - Lack of standards lead to poor interoperability between applications
  - Can be improved by using common protocols

- **Copyright / Access Control**
  - Classic case of Napster being shut down
  - Other applications have learned to get around the law
  - Possibility of paid access in future

**Some key issues (cont)**

- **Quality of Service (QoS)**
  - Metrics to be used is not clearly defined
  - Tradeoff between achieving QoS and costs

- **Complexity of Queries**
  - Must be able to support query languages of varying degree of expressiveness
  - Simple keywords to SQL-like searches

- **Search Mechanism**
  - Different search algorithms are used to reduce search time and maximize search space

**Some key issues (cont)**

- **Load Balancing**
  - Existence of hot-spots (overloaded nodes) due to:
    - Uneven node distribution throughout logical space
    - Uneven object distribution among nodes
    - Uneven demand distribution among objects
    - Query and routing hot-spots

- **Self-organization**
  - Ability to adapt itself to the dynamic nature of the Internet
  - Depends on the architecture of the system

**P2P: Lessons Learned**

- Servers are absolutely required, with or without peer assistance
  - In order to provide a satisfactory level of quality
- **Key challenge**: how to use peer assistance in a complementary fashion without relying upon it

- **New research areas**
  - ISP-friendly overlay topologies: localize the traffic
  - Incentives and pricing models for non-cooperative users
  - Scalability: protocols that work well in a testbed may not scale
  - Availability, robustness to failures, workload characterization
  - More in-depth analysis of large-scale P2P systems
  - New application scenarios: one-click hosting services
Useful Links

- JXTA: http://www.jxta.org
- Gnutella: http://www.gnutella.com
- Napster: http://www.napster.com
- Freenet: http://www.freenet.de
- BitTorrent: http://www.bitwonder.com
- P2P China: http://www.ppcn.net
- Open Source P2P: http://openp2p.com

THANK YOU !!!

For not falling asleep : - )