Peer-to-Peer (P2P) Networks

Dr. Yingwu Zhu
Overview

• CS Architecture
• P2P Architecture
• Unstructured P2P Networks
  – Napster, Gnutella, KaZza, Freenet
  – BitTorrent
• Structured P2P Networks
  – Chord, Pastry, Tapestry, CAN
  – Won’t be covered here!
Client/Server Architecture

- Well known, powerful, reliable server is a data source
- Clients request data from server
- Very successful model
  - WWW (HTTP), FTP, Web services, etc.

As more clients are added, the demand on the server increases!!!
Client/Server Limitations

• Scalability is hard to achieve
• Presents a single point of failure
• Requires administration
• Unused resources at the network edge
  – CPU cycles, storage, etc.
• P2P systems try to address these limitations
Why Study P2P?

• Huge fraction of traffic on networks today >=50%!
• Exciting new applications
• Next level of resource sharing
  – Vs. timesharing, client-server, P2P
  – E.g. Access 10’s-100’s of TB at low cost.
Users and Usage

• 60M users of file-sharing in US
  – 8.5M logged in at a given time on average
• 814M units of music sold in US last year
  – 140M digital tracks sold by music companies
• As of Nov, 35% of all Internet traffic was for BitTorrent, a single file-sharing system
• Major legal battles underway between recording industry and file-sharing companies
Share of Internet Traffic

Traffic, by network protocol:
- HTTP (Web pages)
- Other non-peer-to-peer
- BitTorrent, e-Donkey, FastTrack, Gnutella & other peer-to-peer

250 megabits per second
Number of Users

Others include BitTorrent, eDonkey, iMesh, Overnet, Gnutella.

BitTorrent (and others) gaining share from FastTrack (Kazaa).
P2P Computing

• P2P computing is the sharing of computer resources and services by direct exchange between systems.

• These resources and services include the exchange of information, processing cycles, cache storage, and disk storage for files.

• P2P computing takes advantage of existing computing power, computer storage and networking connectivity, allowing users to leverage their collective power to the “benefit” of all.
P2P Architecture

- All nodes are both clients and servers
  - Provide and consume data
  - Any node can initiate a connection
- No centralized data source
  - “The ultimate form of democracy on the Internet”
  - “The ultimate threat to copyright protection on the Internet”
What is P2P?

- A distributed **system architecture**
  - No centralized control
  - Typically many nodes, but unreliable and heterogeneous
  - Nodes are symmetric in function
  - Take advantage of distributed, shared resources (bandwidth, CPU, storage) on peer-nodes
  - Fault-tolerant, self-organizing
  - Operate in dynamic environment, frequent join and leave is the norm
P2P Network Characteristics

• Clients are also servers and routers
  – Nodes contribute content, storage, memory, CPU
• Nodes are autonomous (no administrative authority)
• Network is dynamic: nodes enter and leave the network “frequently”
• Nodes collaborate directly with each other (not through well-known servers)
• Nodes have widely varying capabilities
P2P vs. Client/Server

• Pure P2P:
  – No central server
  – For certain requests any peer can function as a client, as a router, or as a server
  – The information is not located in a central location but is distributed among all peers
  – A peer may need to communicate with multiple peers to locate a piece of information

As more peers are added, both demand and capacity of the network increases!
P2P Benefits

• Efficient use of resources
  – Unused bandwidth, storage, processing power at the edge of the network

• Scalability
  – Consumers of resources also donate resources
  – Aggregate resources grow naturally with utilization

• Reliability
  – Replicas
  – Geographic distribution
  – No single point of failure

• Ease of administration
  – Nodes self organize
  – No need to deploy servers to satisfy demand (c.f. scalability)
  – Built-in fault tolerance, replication, and load balancing
P2P Traffics

- P2P networks generate more traffic than any other internet application
- 2/3 of all bandwidth on some backbones
P2P Data Flow

CacheLogic P2P file format analysis (2005)
Streamsight used for Layer-7 Deep Packet Inspection
Category of P2P Systems

• Unstructured
  – No restriction on overlay structures and data placement
  – Napster, Gnutella, Kazza, Freenet, Bittorrent

• Structured
  – Distributed hash tables (DHTs)
  – Place restrictions on overlay structures and data placement
  – Chord, Pastry, Tapestry, CAN
Napster

- Share Music files, MP3 data
- Nodes register their contents (list of files) and IPs with server
- Centralized server for searches
  - The client sends queries to the centralized server for files of interest
  - Keyword search (artist, song, album, bitrate, etc.)
- Napster server replies with IP address of users with matching files
- File download done on a peer to peer basis
- Poor scalability
- Single point of failure
- Legal issues → shutdown
Napster: Publish

I have X, Y, and Z!

123.2.21.23

insert(X, 123.2.21.23)
Napster: Search

Where is file A?

123.2.0.18

Search

Fetch

Query

Reply

search(A)

123.2.0.18
Napster

• Central Napster server
  – Can ensure correct results
  – Bottleneck for scalability
  – Single point of failure
  – Susceptible to denial of service
    • Malicious users
    • Lawsuits, legislation

• Search is centralized

• File transfer is direct (peer-to-peer)
Gnutella: Query Flooding

Breadth-First Search (BFS)
Gnutella: Query Flooding

• A node/peer connects to a set of Gnutella neighbors
• Forward queries to neighbors
• Client which has the Information responds.
• Flood network with TTL for termination

+ Results are complete
- Bandwidth wastage
Gnutella vs. Napster

• Decentralized
  – No single point of failure
  – Not as susceptible to denial of service
  – Cannot ensure correct results

• Flooding queries
  – Search is now distributed but still not scalable
Gnutella: Random Walk

- Improved over query flooding
- Same overly structure to Gnutella
- Forward the query to random subset of its neighbors
  - Reduced bandwidth requirements
  - Incomplete results
  - High latency
Kazza (Fasttrack Networks)

- Hybrid of centralized Napster and decentralized Gnutella
- Super-peers act as local search hubs
  - Each super-peer is similar to a Napster server for a small portion of the network
  - Super-peers are automatically chosen by the system based on their capacities (storage, bandwidth, etc.) and availability (connection time)
- Users upload their list of files to a super-peer
- Super-peers periodically exchange file lists
- You send queries to a super-peer for files of interest
  - The local super-peer may flood the queries to other super-peers for the files of interest, if it cannot satisfy the queries.
- Exploit the heterogeneity of peer nodes
Kazza

- Uses supernodes to improve scalability, establish hierarchy
- Uptime, bandwidth
- Closed-source
- Uses HTTP to carry out download
- Encrypted protocol; queuing, QoS
KaZaA: Network Design

“Super Nodes”
insert(X, 123.2.21.23)

I have X!

123.2.21.23

Publish
KaZaA: File Search

Where is file A?

search(A) --> 123.2.0.18

search(A) --> 123.2.22.50

query

replies

123.2.22.50

123.2.0.18
Freenet

- Data flows in reverse path of query
  - Impossible to know if a user is initiating or forwarding a query
  - Impossible to know if a user is consuming or forwarding data

“Smart” queries

Requests get routed to correct peer by incremental discovery

Figure 1. Typical request sequence. The request moves through the network from node to node, backing out of a dead-end (step 3) and a loop (step 7) before locating the desired file.
BitTorrent

Dr. Yingwu Zhu
Bittorrent

• A popular P2P application for file exchange!
Problems to Address

• Traditional Client/Server Sharing
  – Performance deteriorates rapidly as the number of clients increases

• Free-riding in P2P network
  – Free riders only download without contributing to the network.
Basic Idea

• Chop file into many pieces
  – A piece is broken into sub-pieces ... typically 16KB in size
  – Policy: Until a piece is assembled, only download sub-pieces for that piece
  – This policy lets complete pieces assemble quickly

• Replicate DIFFERENT pieces on different peers as soon as possible

• As soon as a peer has a complete piece, it can trade it with other peers

• Hopefully, we will be able to assemble the entire file at the end
File Organization

1 2 3 4

File

Piece 256KB

Block 16KB

Incomplete Piece
Overall Architecture

Web Server

Tracker

Peer
[Seed]

Peer
[Leech]

Peer
[Leech]

Downloader
“US”

A

Web page with link to .torrent

torrent

B

C
Overall Architecture

Peer [Leech]
Downloader “US”

Web Server

Tracker

Web page with link to .torrent

Get-announce

Peer [Seed]

Peer [Leech]

Peer
Overall Architecture

Web Server

Tracker

Response-peer list

Peer [Leech]

Downloader “US”

Peer [Leech]

Peer [Seed]
Overall Architecture

- Web Server
- Tracker
- Peer [Leech]
- [Leech]
- Downloader “US”
Overall Architecture

Web Server

Tracker

Peer [Leech]

Peer [Seed]

Downloader “US”

Peer [Leech]

A

B

C
Overall Architecture

Web Server

Tracker

Web page with link to .torrent

Peer [Leech]

Downloader "US"

Peer [Leech]

Peer [Seed]
Overall Architecture

A

Web Server

Get-announce

Response-peer list

pieces

Peer

[Leech]

Downloader

“US”

B

Peer

[Leech]

C

Tracker

Peer

[Seed]
Critical Elements

• 1 A web server
  – To provide the ‘metainfo’ file by HTTP
  – For example:
    • http://bt.btchina.net
    • http://bt.ydy.com/

The Lord of Ring.torrent

Web Server

Troy.torrent
Critical Elements

• 2 The .torrent file
  – Static ‘metainfo’ file to contain necessary information:
    • Name
    • Size
    • Checksum
    • IP address (URL) of the Tracker
    • Pieces <hash₁,hash₂,….hashₙ>
    • Piece length
Critical Elements

- 3 A BitTorrent tracker
  - Non-content-sharing node
  - Track peers
  - For example:
    - [http://bt.cnxp.com:8080/announce](http://bt.cnxp.com:8080/announce)

- Peer cache
  - IP, port, peer id

- State information
  - Completed
  - Downloading

- Returns random list
Critical Elements

• 4 An end user (peer)
  – Guys who want to use BitTorrent must install corresponding software or plug-in for web browsers.
  – **Downloader (leecher)**: Peer has only *a part (or none)* of the file.
  – **Seeder**: Peer has the *complete* file, and chooses to stay in the system to allow other peers to download
Messages

- Peer – Peer messages
  - TCP Sockets
- Peer – Tracker messages
  - HTTP Request/Response
BitTorrent – joining a torrent

Peers divided into:
- **seeds**: have the entire file
- **leechers**: still downloading

1. Obtain the *metadata file*
2. Contact the *tracker*
3. Obtain a *peer list* (contains seeds & leechers)
4. Contact peers from that list for data
BitTorrent – exchanging data

- Verify pieces using hashes
- Download sub-pieces in parallel
- Advertise received pieces to the entire peer list
- Look for the rarest pieces
BitTorrent - unchoking

- Periodically calculate data-receiving rates
- Upload to *(unchoke)* the fastest downloaders
- *Optimistic unchoking*
  - periodically select a peer at random and upload to it
  - continuously look for the fastest partners
Demo

HTTP GET  MYFILE.torrent

webserver

Tracker

User

Peer 1

Peer 2

Peer 40

list of peers

ID1  169.237.234.1:6881
ID2  190.50.34.6:5692
ID3  34.275.89.143:4545
...  
ID50 231.456.31.95:6882

http://mytracker.com:6969/
S3F5YHG6FEB
FG5467HGF367 “register”
F456JI9N5FF4E
...

Peer 40
Swarming Pieces and Sub-pieces

- A piece, typically 256KB is broken into 16KB sub-pieces.
- Until a piece is assembled, only sub-pieces for that piece is downloaded.
- This ensures that complete pieces assemble quickly.
- When transferring data over TCP, it is critical to always have several requests pending at once, to avoid a delay between pieces being sent.
- At any point in time, some number, typically 5, are requested simultaneously.
- On piece completion, notify all (neighbor) peers.
Piece Selection

• The order of pieces is very important for good performance.
• A bad algorithm could result in all peers waiting for the same missing piece.
• Random Piece First policy
  – Initially a peer had no pieces to trade, thus important to get a piece ASAP.
  – Policy: Peer starts with a random piece to download.
• Rarest Piece First policy
  – Policy: Download the pieces which are most rare among your peers.
  – Ensures most common pieces are left for last.
Rarest First Policy

HAVE <12,7,36>

HAVE <12,7,14>

HAVE <14>

HAVE <12,7,14>
End Game mode

- When all the sub-pieces that a peer doesn’t have are requested, a request is sent to every peer.
- When the sub-piece arrives, duplicate requests are canceled.
- This ensures, completion is not prevented due to a slow peer.
Tit-for-Tat Strategy

“Give and yet shall receive”

• Cooperate if the other peer cooperates.
• Chocking mechanism.
• Choke all peers except top 4 up loaders.
• Optimistic Un-choke for eventual cooperation and recovery.
Tit-for-Tat

Peer 1 Un-choked
Peer 2 Choked

Peer 1 Choked
Peer 2 Un-choked

Peer

Peer 1

Peer 2

Slow Upload
Choking

- Ensures every nodes cooperate and prevents free-riding problem.
- Goal is to have several bidirectional connections running continuously.
- Choking is temporary refusal to upload, downloading occurs as normal.
- Connection is kept open so that setup costs are not borne again and again.
- At a given time only 4 best peers are un-choked.
- Evaluation on whom to choke/un-choke is performed every 10 seconds.
- Optimistic Un-choke every 30 seconds.
  - Give a chance for newly joined peer to get data to download (bootstrapping newcomers!)
  - Hope to find faster upload peers
Choking Algorithm

• Goal is to have several bidirectional connections running continuously
• Upload to peers who have uploaded to you recently
• Unutilized connections are uploaded to on a trial basis to see if better transfer rates could be found using them
Choking Specifics

• A peer always unchokes a fixed number of its peers (default of 4)
• Decision to choke/unchoke done based on current download rates, which is evaluated on a rolling 20-second average
  – This prevents wastage of resources by rapidly choking/unchoking peers
  – Supposedly enough for TCP to ramp up transfers to their full capacity
• Which peer is the optimistic unchoke is rotated every 30 seconds
Anti-Snubbing

• Policy: When over a minute has gone by without receiving a single sub-piece from a particular peer, do not upload to it except as an optimistic unchoke

• A peer might find itself being simultaneously choked by all its peers that it was just downloading from

• Download will lag until optimistic unchoke finds better peers

• Policy: If choked by everyone, increase the number of simultaneous optimistic unchokes to more than one
Up-load only or Seeding mode

• Once the download is complete, has no download rates to compare, nor requires them.

• Which node to upload?

• Policy: Upload to top 4 peers with maximum download rate.
  – Ensures faster replication.
  – Threat: manipulation by faster downloading peers