Data distribution in P2P systems

Data distribution: the P2P approach(es)

Andrea Bianco
(Thanks to Michela Meo)
Telecommunication Network Group
firstname.lastname@polito.it
http://www.telematica.polito.it/

Peer-to-peer architecture

• Peers (hosts running the app) contribute to service provisioning
• All peers have the same role
• Peers are at the same time servers and clients, i.e., they both use and provide service
• The resources needed to provide service are at the periphery of the network, in the hosts
• Resources can be:
  – contents
  – computation/storage
  – bandwidth
The overlay network

- The overlay network among peers allows to put together the resources at the network periphery (in the hosts)

Nodes in the overlay network are peers: hosts running the application

Links in the overlay network are logical links at the application level

A logical link at the application level requires that two peers know each other:
  - Both are running the application
  - Know their contact information: IP address and port number
  - If the logical links use TCP at the transport layer, they must have opened the TCP connection

Two peers with a logical link are neighbors
P2P systems: motivations

- **Scalability:**
  - P2P approaches scale well with respect to the number of users, i.e., they work and are efficient even under extremely large number of users
  - When the number of peers grows, both the amount of work and the service provisioning grow

- **Cost reduction:**
  - Resources are (partially) deployed by users
  - No (or limited) need for infrastructure

Peer-to-peer systems

Examples of possible applications

- **File sharing:**
  - Peers share their contents, the P2P system allows to retrieve contents that are in the peers

- **Content distribution:**
  - Peers contribute to the distribution of contents (of big size) of interest to a large population of users
  - Peers use their bandwidth for the content distribution

- **Distributed computing:**
  - Peers use their computational power for a common goal
Issues at the application level

• Some issues are related to churning:
  – on/off unpredictable behavior of users
• System resources are highly variable (depend on the users’ participation):
  – total amount varies
  – position in the overlay varies
• Resource discovery is not easy
• Connectivity varies in time
• NAT traversal and firewalling obstacles

Issues at the ISP level

• Need to adequate network design
  – from asymmetric traffic profiles (more capacity on the downlink than on the uplink) to more symmetric
• Potentially very large amounts of traffic, often difficult to control
• Protection of the network from systems that bypass firewall/NAT control
• Competitive services
Issues at the user level

- Risks for the user’s system that related to opening the system (malware, spyware, viruses, ...)
- Content availability
- Privacy issues
- Some legal aspects can arise for applications distributing contents that are covered by copyright

Peer-to-peer systems

- Based on the overlay network topology, we distinguish P2P systems in
  - Unstructured systems:
    - The overlay topology is not regular, it is randomly created according to rules for the overlay creation and maintenance
  - Structured systems:
    - The overlay topology has a regular topology that is predefined (grid, ring, tree, ...)
- P2P architectures can be
  - flat: all peers are in charge of the same functions
  - hierarchical: different functions for the peers
File sharing applications

- Users share their contents
- When many peers participate, many contents are shared: demand for service grows with the number of users, but the availability of contents also grows
- File sharing is the first case of P2P system
- Started with very successful music sharing applications (Napster)
  - Operated in 1999-2001
  - Reached 80 millions of users
  - Sued by the Recording Industry Association of America (RIAA), Napster had to close

Napster

- Client connects to Napster with login and password
- Transmits current listing of shared files
- Napster registers username, maps username to IP address and records song list
Data distribution in P2P systems

Napster

- Client sends song request to Napster server
- Napster checks song database
- Returns matched songs with usernames and IP addresses (plus extra stats)

Napster

- User selects a song, download request sent straight to user
- Machine contacted if available
Data distribution in P2P systems

Napster: assessment

- Scalability, fairness, load balancing
  - Replication to querying nodes
    - Number of copies increases with popularity
  - Large distributed storage
  - Unavailability of files with low popularity (no guarantee)
- Content location
  - Simple, centralized search/location mechanism
- Failure resilience
  - No dependencies among normal peers
  - Index server as single point of failure

Functions in P2P file sharing

- **Join**: a peer enters the overlay network and starts participating to the system
- **Overlay maintenance**: take care that the overlay is properly connected so as to guarantee the properties that are needed for the correct working of the system
- **Query**: a peer queries for a content and retrieves information on the peers holding it
- **Download**: a peer downloads the content it was looking for
Gnutella

- Program for sharing files over the Internet
  - Peers share their file
- Purely distributed approach, no centralized point, no infrastructure → get rid of the central index (see Napster)
- The overlay network is used to implement the query function
- Download is done on a point-to-point basis, once the content is found through the query function

Joining

1. A requests a list of active peers
2. A receives the list, it contains B
3. A chooses B from list and sends a JOIN request
4. Confirm JOIN
Overlay maintenance

- After JOIN, peer A is connected to the overlay
- Its connectivity is very limited (only 1 connection to B)
- Overlay update and maintenance is needed
Overlay maintenance

- Once A discovers new peers, it can choose which one to connect to

![Diagram of peer network with overlay maintenance](image)

- PING forward continues up to H hops away from the peer that initiated the process
  - Implemented with a TTL field, decremented at each forwarding
- Messages have an ID to
  - Avoid reacting to duplicates of the same request
  - Duplicates are dropped
- PONG messages follow the reverse path of the corresponding PING
  - They can cross only logical links of the overlay network
Overlay maintenance

• PING/PONG messages exchange allows to:
  – Verify connectivity of neighbors
  – Receive contact information of other peers that are in the overlay

• Connectivity can be updated/adjusted once PONG messages are received

• Peer discovery is
  – very effective: in a short time many PONGs are received
  – very costly for the network: huge number of PING and PONG messages

Overlay maintenance

• Assuming
  – k neighbors (constant) per peer
  – up to H forwarding of PING messages

• Number of PING messages (and contacted peers):

\[ N = \sum_{i=0}^{H-1} k(k-1)^i \]

• Number of PONG messages

\[ M = \sum_{i=0}^{H-1} (i+1)k(k-1)^i \]

• Average time to contact N peers: Number of PONG messages:

\[ T = T_c \sum_{i=0}^{H-1} k(k-1)^i \]

with \( T_c \), average contact time

\( k=4, H=7 \rightarrow N=4372 \)

\( k=4, H=7 \rightarrow M=28K \)
Data distribution in P2P systems

Query

• By flooding
• Similar to overlay maintenance (PING/PONG)

1. A sends a QUERY(C) message for content C
2. B checks if it has C
3. If not, B forwards QUERY to its neighbors
4. D forwards QUERY(C)
5. QUERY_HIT

Query

• Peers may receive several positive replies to a QUERY and choose where to download from
• QUERY has ID and TTL (like PINGs)

• The searching mechanism is
  – very effective: in a short time many peers are contacted
  – probability to find the content depends on popularity and it is not guaranteed for little popular contents
  – flooding is very costly for the network, requires many messages
Data distribution in P2P systems

Query

• Assuming that
  – peer A queries for content C
  – k neighbors (constant) per peer
  – up to H forwarding of QUERY message
  – popularity of C is p (probability that a peer holds content C)

• Prob. that C cannot be found:

\[ P = (1 - p)^L \]

with L equal to the number of contacted peers:

\[ L = \sum_{i=0}^{H-1} k(k-1)^i \]

Download

• QUERY_HIT contains information for contacting the peer
• Direct download
• No logical link is established on the overlay network
Gnutella: Assessment

- Scalability, fairness, load balancing
  - Replication to querying nodes
    - Number of copies increases with popularity
  - Large distributed storage
  - Unavailability of files with low popularity
  - Bad scalability, uses flooding approach
  - Network topology is not accounted for at all, latency may be increased

- Content location
  - No limits to query formulation
  - Less popular files may be outside TTL

- Failure resilience
  - No single point of failure
  - Many known neighbors
  - Assumes quite stable relationships

BitTorrent objectives

- Download
  - Large contents (movies, OS updates,...)
  - To large populations of users
  - “Flash crowd” scenario

- Users’ contribute by becoming content distributors while downloading the content

- Users contribute to the service through their upload bandwidth

- Reduction of cost for the content distributor
Data distribution in P2P systems

Content distribution

- **Client-server**
  - needs significant capacity to serve in short time all the users

- **P2P**
  - alleviates burden on the source, other peers redistribute the content

P2P vs. Client-Server

- **P2P**
  - Capacity of service
    \[ C(t) = O(e^t) \], where \( t \) is time
- **Client-server**
  - Capacity of service
    \[ C(t) = 1 \], where \( t \) is time
Data distribution in P2P systems

Content transfer model

• Simple model
  – Each peer serves only one peer at a time
  – The unit of transfer is the content
  – n peers want the content
  – We assume n=2^k
  – T is the time to complete an upload
    • T=s/b, s content size, b upload capacity (for each peer)
  – Global knowledge, always know which peers need the content

Capacity growth

• Capacity of service C
  – t=0, 2^0 peers, C=b2^0
  – t=T, 2^1 peers, C=b2^1
  – t=2T, 2^2 peers, C=b2^2
  – ...
  – t=iT, 2^i peers, C=b2^i
  – 2^{i/T} peers, C=b2^{i/T}
Completion time

• Finish time
  – Seed has the content at \( t=0 \)
  – \( 2^0 \) peers finish at \( t=T \)
  – \( 2^1 \) peers finish at \( t=2T \)
  – \( \ldots \)
  – \( 2^k \) peers finish at \( t=kT \)
  – We served the \( n \) peers in
    • \( t = kT = \log_2(n)T \)

Model discussion

• Each peer has the same upload capacity
• No network bottleneck
• Idealized peer selection strategy
  – Each peer always knows to which peer \( P \) send the content at a given time
    • Peer \( P \) does not have the content yet
    • Peer \( P \) is not chosen by any other peer
  – Conflict resolution solved with global knowledge
  – No peer churning, i.e., arrival and departure
Data distribution in P2P systems

Capacity growth

- Capacity grows with time
- Effectiveness of the P2P approach grows
- First part of the transfer is the most fragile one
  - few copies of the content
  - only few “servers”
- Service capacity depends on
  - Availability of content
  - Presence of interested peers

Observations

- In this distribution tree, not all the peers contribute in the same way
- Leaves in the distribution tree do not use their upload bandwidth → split the content in pieces so that different distribution trees are created to distribute in parallel the many pieces
- Peers contribute if they don’t leave the system once they have downloaded the content (free riders)
Content transfer model

- What about distributing the content to more than one peer at the same time?
  - Each peer serves two peers at a time
  - The time to complete an upload
    - $T' = \frac{s}{b/2} = 2s/b$, $s$ content size, $b$ upload capacity
    - $T' = 2T$, double time needed to complete the upload with respect to the previous case

Service parallelism

Seed

\[ t=0 \]

\[ t=T \]

\[ t=2T \]

16 peers at $t=4T$

9 peers at $t=4T$

3 peers at $t=2T$

...
Data distribution in P2P systems

Discussion

• The model suggests to
  – Divide the content in pieces
  – Transfer one piece at a time
  – Carefully choose peer and piece selection strategies

• P2P is very efficient when
  – There is always a peer to send data to
  – There is always a piece to send to this peer
• Peer and piece selection strategies are at the core of an efficient P2P protocol

BitTorrent

• It is a P2P system for file sharing:
  – It uses a P2P approach for the download
  – Query is solved outside the P2P distribution process
  – Overlay maintenance is done through a dedicated device (in a distributed way in some cases)
• There exists no single BitTorrent network, but thousands of temporary networks (torrents) consisting of clients downloading the same file
• There exist many different BitTorrent clients:
  – The java based client Azureus is one of the most popular
Terminology

- **Seeder**
  - A peer who has all the blocks in a torrent
- **Leecher**
  - A client who is downloading from the seeders
- **Chunk**
  - A piece of a file typically 64 KB to 256 KB in size
- **Tracker**
  - A middleman who informs the peers of the other peers in the network
- **Torrent**
  - A group of peers that are connected to the same tracker and downloading the same file
- **Torrent file (.torrent)**
  - A file which provides a URL to the tracker and contains a list of SHA1 hashes for the data being transferred
- **Choked**
  - A connection is choked if no file data is passed through it
  - Control data may flow but the transmission of actual blocks will not
- **Interest**
  - indicates whether a peer has blocks which other peers want

Operation summary

- **The original file distributor**
  - publishes details of the file on a web server, and
  - creates a tracker that allows peers interested in the file to find each other
- **To download the file, peers access the tracker and join the torrent**
- **The file is divided into equal-sized blocks (typically 32-256 KB) and nodes download concurrently the blocks from multiple peers**
- **The blocks are further subdivided into sub-blocks to enable pipelining of requests to mask the request-response latency**
- **As a peer downloads blocks of the file, it also uploads to other peers in the torrent blocks that it has previously been downloaded**
Data distribution in P2P systems

**Detailed operation**

- Nodes in the system are either
  - *seeders*: nodes that have a complete copy of the file and are willing to serve it to others
  - *leechers*: nodes that are still downloading the file but are willing to serve the blocks that they already have to others
- When a new node joins a torrent, it contacts the tracker to obtain a list containing a random subset of the nodes currently in the system
  - both seeds and leechers
- The new node then attempts to establish connections to many (about 40) existing nodes, which become its neighbors
- If the number of neighbors of a node ever dips below a threshold (e.g., 20) due to churning, the node contacts the tracker again to obtain a list of additional peers it could connect to

---

**Overall architecture**

![Diagram of P2P system architecture](attachment:diagram.png)
Overall architecture

Data distribution in P2P systems

Andrea Bianco – TNG group - Politecnico di Torino

Computer Networks Design and Management - 47

Overall architecture

Computer Networks Design and Management - 48

Andrea Bianco – TNG group - Politecnico di Torino

Pag. 24
Data distribution in P2P systems

Overall architecture

Web Server

Tracker

A

B

C

Peer [Leech]

Peer [Leech]

Peer

Shake-hand

Shake-hand

Shake-hand

[See d]

[See d]

Overall architecture

Web Server

Tracker

A

B

C

Peer [Leech]

Peer [Leech]

Peer [Leech]

piece

piece

piece
Data distribution in P2P systems

Overall architecture

A [Leech]  
Peer Download  

B  
Peer [See d]  

C  
Peer [See d]  

Web Server  

Tracker  

Get announce Peer list  

piece s  

piece s  

piece s
The Torrent file

- The torrent file has all necessary information for a peer to download a file
  - URL of the tracker
  - Fileinfo (considering only one file)
    - Name of the file
    - Piece length/size
    - File size
    - SHA1 hashes of each piece
  - File ID is generated as SHA1 hash of the fileinfo

Tracker

- The tracker receives information of all peers
- The tracker provides random lists of peers, when needed (join, increase of connectivity)
- Single point of failure
  - New versions of BitTorrent can use a DHT for receiving other peers information (trackerless)
- Request consists of:
  - File ID
  - Peer ID
  - Peer IP
  - Peer Port
- Tracker response contains:
  - Interval: number of seconds between normal requests
  - List of peers (i.e., 40 peers) containing ID, IP and Port of each peer
- Peers may re-request on nonscheduled times, if they need more peers
Requirements for the Tracker

- The requirements from the Web hosting end are not too much
- To transmit a torrent, it is needed only a standard HTTP Web server and a free program called a "tracker"
- The tracker's job is:
  - to keep track of which clients can serve which files to other clients
- At the tracker traffic load is relatively light
- Offering a tracker to customers can make using BitTorrent to distribute contents a much simpler process for both the content distributor and the customers

Pieces and blocks

- Content is split into pieces, which are split into blocks

```
<table>
<thead>
<tr>
<th>Piece 1</th>
<th>Piece 2</th>
<th>Piece m-1</th>
<th>Piece m</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block 1</td>
<td>Block 2</td>
<td>Block k-1</td>
<td>Block k</td>
</tr>
</tbody>
</table>
```

Content
Pieces and blocks

- Pieces
  - The smaller unit of retransmission
  - Typically 256/512/1024/2048 kByte
  - Size adapted to have a reasonably small .torrent file
    - One SHA-1 hash per piece in the .torrent file

- Blocks
  - 16kB (hard coded)
  - Used for pipelining
    - Always 5 requests pending

Pieces exchange

- A peer exchanges buffer maps of pieces with its neighbors
- A new downloaded piece is notified immediately
Data distribution in P2P systems

Peer protocol

- Each downloader reports to all of its neighbors what pieces of the file it has
- Peers download pieces from all peers they can
- Peers upload to other peers accordingly to the Tit-for-tat (choking) algorithm
  - peers are selected based on their contribution to file download
- Piece selection: local rarest first
  - peer downloads the piece which the fewest of its peers has first
- To avoid delays between pieces that lower transfer rates
  - splits pieces into sub-pieces
  - always having some number of sub-pieces requests pipelined
  - completes a piece before requesting sub-pieces from other pieces

BitTorrent algorithms

- Two components in BitTorrent downloading algorithms:
  - Peer Selection – determines from whom to download the piece
  - Piece Selection – determines which piece to download
**Tit-for-tat algorithm**

- **Objectives:**
  - Limit the number of concurrent uploads
  - Reduce free riding
  - Incentivate peers to contribute to content upload

- A neighboring peer can either be:
  - *Choke* (blocked): cannot download from the peer
  - *Unchocked* (unblocked): download from the peer is allowed

- A peer always unchoke a fixed number of peers (typically 4)
  - which peers to unchoke is based on current download rate from that peer

- A peer recalculates which peers to choke or unchoke every 10 seconds by
  - creating an ordered list of its neighbors based on the download rate from them
  - the 3 peers that contributed the most are unchoked (upload to them is possible)
  - 10 s is:
    - enough time for TCP to achieve full transfer capacity
    - avoids oscillations (no rapid change of choke and unchoke)

- In addition, every 40 seconds: *optimistic unchoke*
  - unchokes a random peer, regardless of its current download rate
  - which peer to optimistic unchoke is rotated every third rechoke
    - enough time for upload and download to achieve full transfer capacity
    - enough time for the unchoked peer to reciprocate
Data distribution in P2P systems

Tit-for-tat algorithm

• Seeders, that do not need to download any piece, choose to unchoke the fastest downloaders

• The choking algorithm is the main driving factor behind BitTorrent’s fairness model:
  – a free-rider will eventually get low download rates
  – lack of cooperation results in being choked from most other peers

• Choking algorithm penalizes peers at the beginning of the content download
  – They cannot contribute because they have no pieces to upload

BitTorrent - Piece selection

• Local rarest first policy
  – Determine the piece that is the most rare among neighbors and download that one first
  – Ensures that the most common pieces are left till the end to download
  – Rarest first also reduces the possibility that pieces disappear

• Rationale
  – Cannot maintain the state for all peers
  – The initial seed should send as fast as possible a first copy of the content
Local Rarest First

- Improve the entropy of the pieces
  - Peer selection is not biased
  - Better survivability of the torrent
    - Even without a seed the torrent is not dead
- Increase the speed at which the initial seed delivers a first copy of the content
  - The seed can leave early without killing the torrent

Random first piece

- Random first piece makes more likely to complete the first piece faster
- Not optimal, but a good tradeoff between simplicity and efficiency (the random piece may be a rarest one)
- Only impacts the startup phase of a peer
- Then switches to local rarest first
Sub-blocks

- BitTorrent uses TCP and it is thus crucial to always transfer data or else the transfer rate will drop because of the slow start mechanism.
- The pieces are further broken into sub-pieces, often about 16kb in size (very small).
- The protocol makes sure to always have some requests (typically five) for sub-pieces pipelined at any time.
- When a new sub-piece is downloaded, a new request is sent.
- Sub-pieces can be downloaded from different peers.
- A new piece is requested only when all sub-pieces of another piece are downloaded.