BitTorrent Choking Algorithm

Game Theory

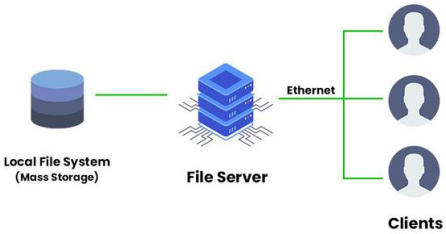
Juswanth T - EE21B063

November 29, 2024

Outline

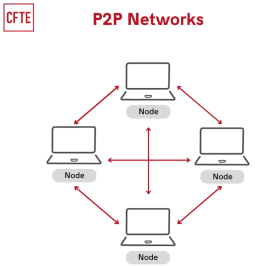
- File sharing mechanisms?
- Introduction to BitTorrent
- Potential Challenges in P2P Networks
- Fluid Model of BitTorrent
- Steady-State Performance
- Choke algorithm that Powers BitTorrent
- Mechanism of the game
- Why Nash Equilibrium May Not Exist?
- Nash Equilibrium in Homogeneous Groups
- Simulation
- Optimistic Choking

File sharing mechanisms



- **Server Based:** A centralized file-sharing system uses a central server to store files, manage access, and facilitate file transfers. Clients requests a file from the server and the server responds.
- Things become interesting when there are large number of request. Since server's bandwidth is limited, more clients will slow down the system.

File sharing mechanisms



- **Peer - to - Peer:** In P2P, there is no central server. It is a decentralized network where each user (peer) acts as both a client (downloader) and a server (uploader)
- No Single Point of Failure: If one peer goes offline, the network remains functional. No single server delivers the entire file. Instead, parts of the file come from multiple peers, distributing the load. Ex : BitTorrent, eDonkey and Napster.

Introduction to BitTorrent

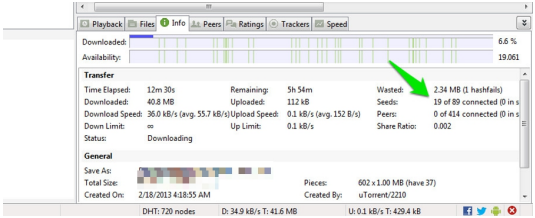
What is it?

BitTorrent is a peer-to-peer (P2P) file-sharing protocol designed to distribute large files efficiently by using the combined bandwidth of all participants in the network, rather than relying on a centralized server.

Key Features

- **File Splitting:**A file is divided into small pieces (chunks), which can be downloaded from different peers simultaneously.
- **Parallel Downloads:** Peers can download different chunks of the same file from multiple sources at the same time, speeding up the process.
- **Fault Tolerance:** As long as some peers have the file, it can still be shared even if others go offline.

How BitTorrent Works?



- **Tracker:**A centralized server that keeps track of peers in the network and helps them connect. Due to the presence of the tracker BitTorrent is called hybrid p2p network instead of pure p2p.
- **Peers:** Users who participate in the network, acting as both uploaders and downloaders.
- **Seeders and Leechers:** Peers who have the entire file and share it. Peers who are downloading the file but haven't yet completed it.

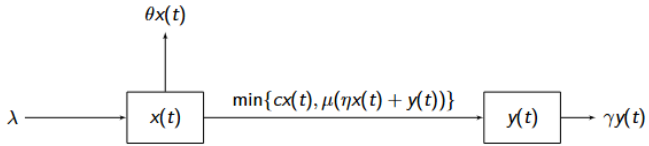
BitTorrent Workflow

- **File Preparation:**The file to be shared is split into chunks and distributed as a .torrent file, which contains metadata like chunk hashes, tracker information, and file structure.
- **Peer Connection:** A user opens the .torrent file in a BitTorrent client, and contacts the tracker using the tracker url. It provides the client with the IP addresses of other peers in the network.
- **Downloading:** The client downloads chunks of the file from multiple peers simultaneously.
- **Uploading:** While downloading, the client uploads chunks it has already received to other peers, contributing to the swarm.
- **Seeder:** Once all chunks are downloaded, the user becomes a seeder, helping others complete their downloads.

Potential Challenges in P2P Networks

- **Free Riding:** Now it is very obvious that anyone who enters the network would just download the file without contributing back. If everyone does that, the network dies as there won't be anyone to upload the files.
- Here is where BitTorrent Choking algorithm comes into play. It helps to maintain fairness and provide dynamic incentive to contribute and keeps the network healthy and efficient.

Fluid Model of BitTorrent



Variables and parameters used in this model:

- $x(t)$ - number of downloaders (also known as leechers) in the system at time t.
- $y(t)$ - number of seeds in the system at time t.
- λ - the arrival rate of new requests.
- θ - the rate at which downloaders abort the download.
- γ - the rate at which seeds leave the system.
- μ - the uploading bandwidth of a given peer.
- c - the downloading bandwidth of a given peer.
- η - the effectiveness factor of the file sharing (η takes values in $[0, 1]$).

Simple Fluid Model Equations

- **Total uploading rate:**

$$\text{Total uploading rate} = \min\{cx(t), \mu(nx(t) + y(t))\},$$

where

- $cx(t)$ captures the *constraint due to downloading bandwidth*, and
- $\mu(nx(t) + y(t))$ captures the *constraint due to uploading bandwidth*.

Fluid Model of BitTorrent

- **Rate of change of number of downloaders (leechers) in the system is given by:**

$$\frac{dx}{dt} = \lambda - \theta x(t) - \min\{cx(t), \mu(nx(t) + y(t))\}.$$

- **Rate of change of number of seeds in the system is given by:**

$$\frac{dy}{dt} = \min\{cx(t), \mu(nx(t) + y(t))\} - \gamma y(t).$$

Steady-state Performance

At steady state, $\frac{dx}{dt} = \frac{dy}{dt} = 0$. ($x$ and y are the equilibrium values of $x(t)$ and $y(t)$, respectively.)

Scenario 1: $cx \leq \mu(\eta x + y)$ **(Downloading bandwidth is the constraint)**

$$x = \frac{\lambda}{c(1 + \frac{\theta}{c})}$$

$$y = \frac{\lambda}{\gamma(1 + \frac{\theta}{c})}$$

Scenario 2: $cx \geq \mu(\eta x + y)$ **(Uploading bandwidth is the constraint)**

$$x = \frac{\lambda}{\nu(1 + \frac{\theta}{\nu})}$$
$$y = \frac{\lambda}{\gamma(1 + \frac{\theta}{\nu})}$$

Where:

$$\frac{1}{\nu} = \frac{1}{\eta}\left(\frac{1}{\mu} - \frac{1}{\gamma}\right)$$

Let $\frac{1}{\beta} = \max\left(\frac{1}{\eta}\left(\frac{1}{\mu} - \frac{1}{\gamma}\right), \frac{1}{c}\right)$. Then, the steady state values can be written as:

$$x = \frac{\lambda}{\beta(1 + \frac{\theta}{\beta})}, \quad y = \frac{\lambda}{\gamma(1 + \frac{\theta}{\beta})}$$

Steady-State Performance

Using Little's law, the average download time (T) can be written as :

$$T = \frac{1}{\theta + \beta}, \quad \text{where} \quad \frac{1}{\beta} = \max\left(\frac{1}{\eta}\left(\frac{1}{\mu} - \frac{1}{\gamma}\right), \frac{1}{c}\right)$$

Insights from the above expression:

- T is independent of λ, the request arrival rate. Hence, the system scales well.
- When η increases, T decreases.
- When γ increases, T increases.

Choke Algorithm that Powers BitTorrent

What is Choking?

- Temporary refusal to upload to a peer while still downloading from them. It encourages cooperation in data sharing.

How It Works

1. **Tit-for-Tat Strategy:**
   - Cooperate with peers who cooperate with you.
   - Retaliate briefly, then forgive to restore cooperation.
2. **Choking Process:**
   - Upload to peers offering the **highest recent download rates**.
   - Measure download rates using a **20-second average**.
3. **Unchoking:**
   - A peer unchokes **4 peers** at a time (default).

Downloading Rate Calculation for Peers

Set Definition (D_i) and Aggregate Downloading Rate (d_i)

Let D_i be the set of peers that select peer i. Each peer randomly selects a peer for uploading. Given a large number of peers, the download rate for i can be expressed as:

$$d_i = \frac{1}{n_u} \sum_{k \in D_i} \mu_k$$

Where:

- n_u: Total number of uploading peers.
- μ_k: Uploading bandwidth of peer k.

Handling Peers with the Same Uploading Bandwidth

If multiple peers have the same uploading bandwidth (μ), their downloading rates (d_k) may differ. To address this, we use the following formula to redistribute the rates fairly:

$$d(\mu) = \frac{1}{j - i + 1} \sum_{k=i}^{j} d_k$$

Where:

- i: The first peer with uploading bandwidth μ.
- j: The last peer with uploading bandwidth μ.

Mechanism of the game

The peer strategy in BitTorrent revolves around determining the optimal **uploading bandwidth** (μ_i) that a peer should contribute to maximize their **downloading rate** (d_i) while considering the **cost** of contributing.

1. Aggregate Downloading Rate For a peer i, the downloading rate is defined as:

$$d_i = d_i(\mu_i, \mu_{-i})$$

Where:

- μ_i: The upload bandwidth of peer i.
- μ_{-i}: The upload bandwidths of all other peers.

Key Insight: The downloading rate d_i depends not only on the peer's own upload bandwidth but also on how it compares to other peers' upload bandwidths. Higher upload bandwidths result in better downloading rates.

2. Strategy for Maximizing Gain Each peer has two competing objectives:

1. **Maximizing the Downloading Rate (d_i)**:
   - A higher μ_i leads to better rankings in the peer selection process, increasing d_i.
2. **Minimizing Cost ($c(\mu_i)$)**:
   - Uploading bandwidth comes at a cost, represented as $c(\mu_i)$, which grows as μ_i increases.

Thus, the **utility** for a peer is defined as:

$$u_i(\mu_i) = d_i(\mu_i, \mu_{-i}) - c(\mu_i)$$

The peer strategy involves finding the μ_i that maximizes u_i.

Mechanism of the game

1. Incentive Mechanism Overview The BitTorrent incentive mechanism encourages users to contribute upload bandwidth by linking it to their downloading rate. Each peer i can choose an upload bandwidth μ_i up to its physical limit p_i. The downloading rate of peer i, $d_i(\mu_i, \mu_{-i})$, is a non-decreasing function of μ_i, with d_i being maximized when $\mu_i = p_i$. However, setting $\mu_i = p_i$ may not always be optimal due to associated costs.

2. Optimal Uploading Bandwidth To balance gain and cost, a peer i should select

$$\mu_i = \min\{\inf\{\tilde{\mu}_i | d_i(\tilde{\mu}_i, \mu_{-i}) = d_i(p_i, \mu_{-i})\} + \epsilon, p_i\}.$$

Why Nash Equilibrium May Not Exist?

If all peers have different physical uploading bandwidths (p_i), then no Nash equilibrium exists.

Setup

The physical upload bandwidths (p) of the peers are as follows:

$$p = [9, 7, 8, 5, 4, 3]$$

The initial upload bandwidths (μ) are:

$$\mu = [9, 6, 4, 3, 2, 1]$$

Each peer uploads to a specific set of peers, and the download rate is proportional to the relative upload contributions of the peers in the network.

Why Nash Equilibrium May Not Exist?

Each peer uploads to a specific set of peers as follows:

- Peer 0 uploads to $[1, 2, 3, 4]$
- Peer 1 uploads to $[0, 2, 3, 4]$
- Peer 2 uploads to $[0, 1, 3, 4]$
- Peer 3 uploads to $[0, 1, 2, 4]$
- Peer 4 uploads to $[0, 1, 2, 3]$
- Peer 5 uploads to $[0, 1, 2, 3]$

Why Nash Equilibrium May Not Exist?

Iteration 1: Initial Download and Upload Rates

Peer ID	Upload BW (μ_i)	Download Rate (d_i)	Uploaders
0	9.00	4.00	$[1, 2, 3, 4, 5]$
1	6.00	4.75	$[0, 2, 3, 4, 5]$
2	4.00	5.50	$[0, 1, 3, 4, 5]$
3	3.00	5.70	$[0, 1, 2, 4, 5]$
4	2.00	5.00	$[0, 1, 2, 3]$
5	1.00	0	[]

Explanation:

- Peer 0 uploads the maximum $\mu_0 = 9.00$ and downloads from peers $[1, 2, 3, 4, 5]$.
- Peer 5 receive the lowest download rate (0), as no one allows it to download from them. It is been choked by all.

Iteration 2: Adjustments and New Rates

In this iteration:

- Peer 5 increases its upload bandwidth to just greater than 5th hightest , that is 2.1
- All other peers tries to reduce their bandwidth hoping that their download rate will still be maintained.

Peer ID	Upload BW (μ_i)	Download Rate (d_i)	Uploaders
0	7.15	3.59	[1, 2, 3, 4, 5]
1	4.48	4.26	[0, 2, 3, 4, 5]
2	2.53	4.75	[0, 1, 3, 4, 5]
3	2.5	4.8	[0, 1, 2, 4, 5]
4	1.1	0	[]
5	2.10	4.10	[0, 1, 2, 3]

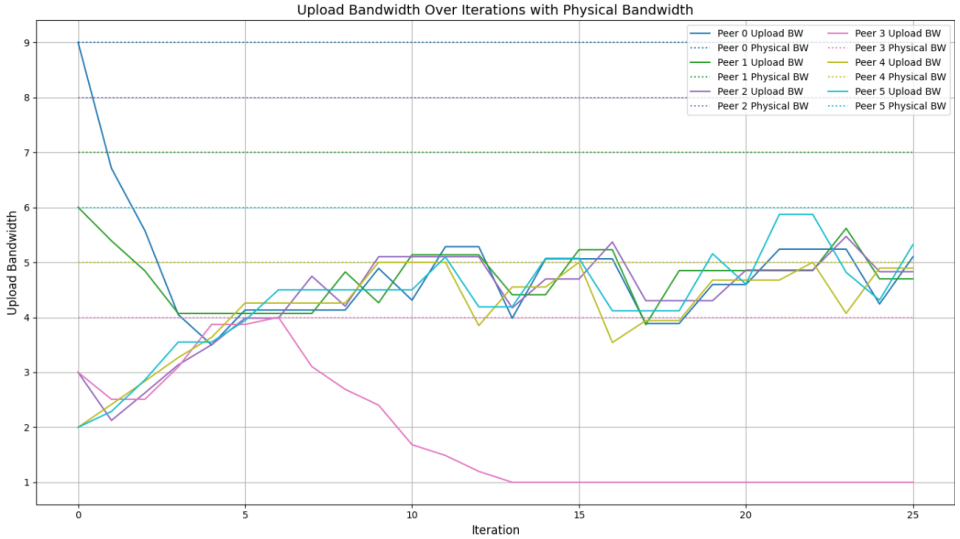
Oscillation Begins: Why Nash Equilibrium Does Not Exist

In the next iterations:

- Peer 4 increases its upload bandwidth slightly above Peer 5's physical bandwidth threshold, setting $\mu_4 = 3.1$.
- Peer 5 responds by reducing its upload bandwidth to $\mu_5 = 0.5 = \epsilon$, to minimize costs while maintaining its download rate which is 0. ϵ is the minimum possible rate one can keep.
- Peer 4 then decreases its upload bandwidth to $\mu_4 = 0.6$, and Peer 5 counters with $\mu_5 = 0.7$ in the next interation.
- This back-and-forth adjustment continues indefinitely.

Thus, a Nash equilibrium does not exist in this system.

Why Nash Equilibrium May Not Exist?

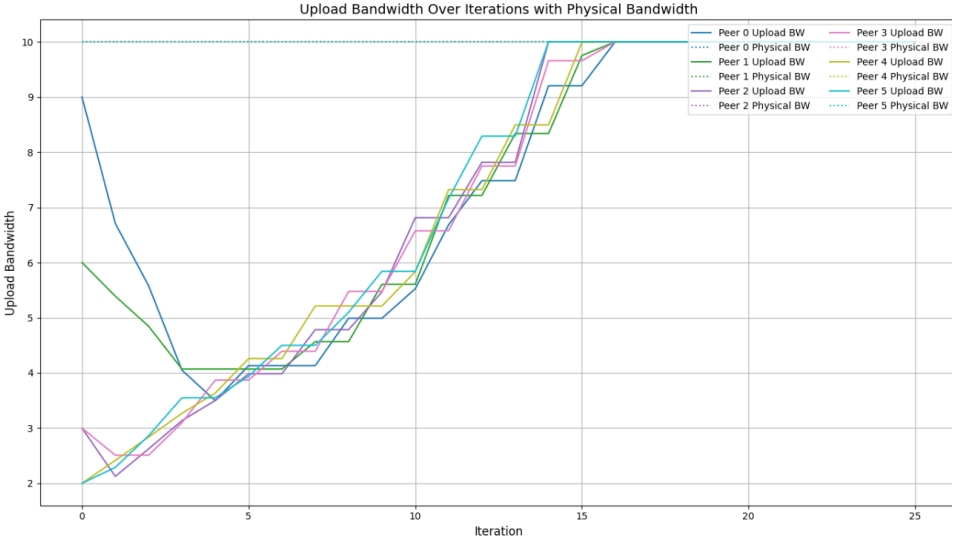


Nash Equilibrium in Homogeneous Group

When the physical bandwidth of all of them are same, nash converges.

- It is easy to see it because, consider the 5th and 6th highest. The last one that is 6th highest can always keep his upload bandwidth above 5th highest unless the 5th highest is the physical bandwidth, in which case everyone would have kept their bandwidth to physical bandwidth.

Nash Equilibrium in Homogeneous Group



Optimistic Choking

- A process where each peer randomly selects an additional peer periodically from which it has received a downloading request and uploads to this peer.
- This process is used to explore the network and obtain information about other peers.
- Optimistic unchoking is attempted periodically (usually every 30 seconds) and to allow optimistic unchoking while keeping the maximum number of uploads, an upload to the peer with the least downloading rate is dropped.
- With optimistic unchoking, the rate of free-riding is a small quantity. Although free-riding can be completely eliminated by not using this process, it cannot be done since it plays a major role in network exploration.

Conclusion

The choking algorithm in BitTorrent plays a crucial role in optimizing the file-sharing process. It determines how peers select other peers to upload to, thereby balancing fairness, speed, and resource utilization. Some of the advantages of using choking algorithm in BitTorrent are:

1. It ensures that bandwidth is fairly distributed among peers, rewarding those who contribute.
2. It ensures that the peer maintains connections with fast peers while also exploring potential new high-speed connections, maximizing overall download speeds.(optimistic unchoking)
3. It minimizes network congestion and allows each connection to be more efficient
4. It leaves the rational decision makers with no choice other than to keep the uploading rate as large as possible.

Thank You