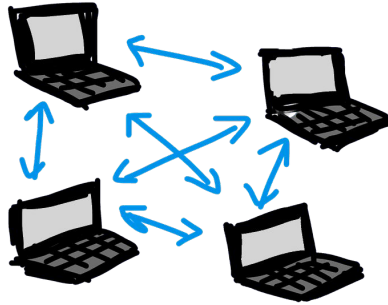


0 Protocolo BitTorrent



CCOS

Centro de Competência
em Open Source

por Guilherme Paixão

GELOS[❄]

Grupo de Extensão em
Livres & Open Source

O GELOS

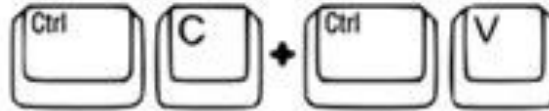
GELOS - *Grupo de Extensão em Livre & Open Source*

- Somos um grupo de extensão com objetivo de reunir pessoas interessadas por [cultura livre](#) e [open source](#).
- Desde software livre e hardware, até Open Science e todos os aspectos sociais e políticos envolvidos!
- O GELOS tem como princípio ser um *agrupamento de entusiastas*. Trocamos conhecimento, e apoiamos (individual e institucionalmente) uns aos outros.
- Você pode, ao seu critério: desenvolver ou idealizar novos projetos, contribuir com projetos de outros membros, ou só socializar. Sendo assim, ser membro *não implica em obrigatoriedade de atividades*, e *não existe hierarquia que “escale membros” para projetos*.



[telegram.gelos.club](https://t.me/telegram.gelos.club)

Esta apresentação está em Domínio Público



"Copy and Paste what thou wilt
shall be the whole of the law."





Sumário

- Motivação
- Funcionamento Modelo Client-Server - HTTP
- O BitTorrent – Surgimento e Funcionamento Básico
- O BitTorrent – Especificação (mais aprofundado)
- Mitos e Questões comuns

Por que?

- BitTorrent é um protocolo totalmente livre – está em domínio público
- Das tecnologias de compartilhamento de arquivos P2P, é a mais utilizada e a que mais “deu certo”
- O BitTorrent, como definido no BitTorrent.org, “*é uma ferramenta de liberdade de expressão*” – Cultura Livre!
- Ótimo caso de estudo: tecnologia livre mantida pela comunidade e usada por milhões de pessoas
- *Sharing is Caring* :)

BitTorrent domina o tráfego mundial de *upstream*!

GLOBAL APP TRAFFIC SHARE			GLOBAL APP TRAFFIC SHARE		
DOWNSTREAM TRAFFIC 			UPSTREAM TRAFFIC 		
	Category	Total Volume		Category	Total Volume
1	YouTube	16.37%	1	BitTorrent	9.70%
2	Netflix	10.61%	2	HTTP	9.05%
3	Facebook	7.67%	3	Google	8.02%
4	Facebook video	4.83%	4	Facebook	5.77%
5	TikTok	4.48%	5	Wordpress	5.01%
6	HTTP Media Stream	4.07%	6	YouTube	4.45%
7	Generic QUIC	4.03%	7	iCloud	4.09%
8	HTTP	2.63%	8	Generic QUIC	3.70%
9	Playstation Download	2.27%	9	Netflix	3.00%
10	iTunes Store	2.12%	10	Facebook Messenger	2.37%

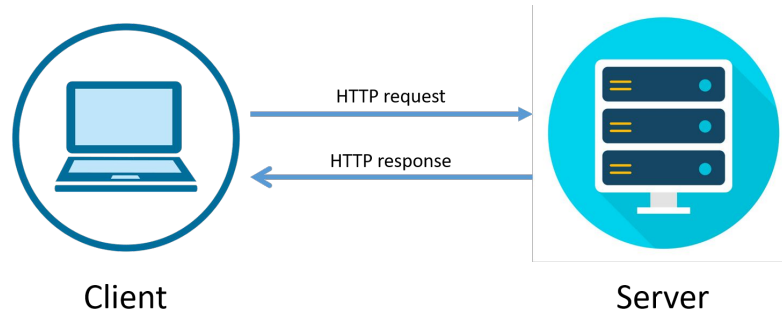
Separando as coisas...

- **BitTorrent:** protocolo *peer-to-peer* de compartilhamento de arquivos.
- **Torrent:** Geralmente se refere ao arquivo *.torrent*, mas também pode ser utilizado informalmente para se referir ao protocolo.
- **uTorrent:** é um *client* **proprietário** mantido por uma empresa com fins lucrativos. (não recomendado)



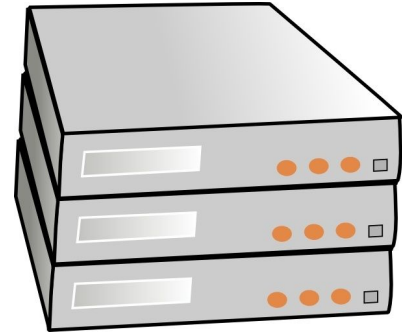
Modelo client-server

Protocolo HTTP – *HyperText Protocol*

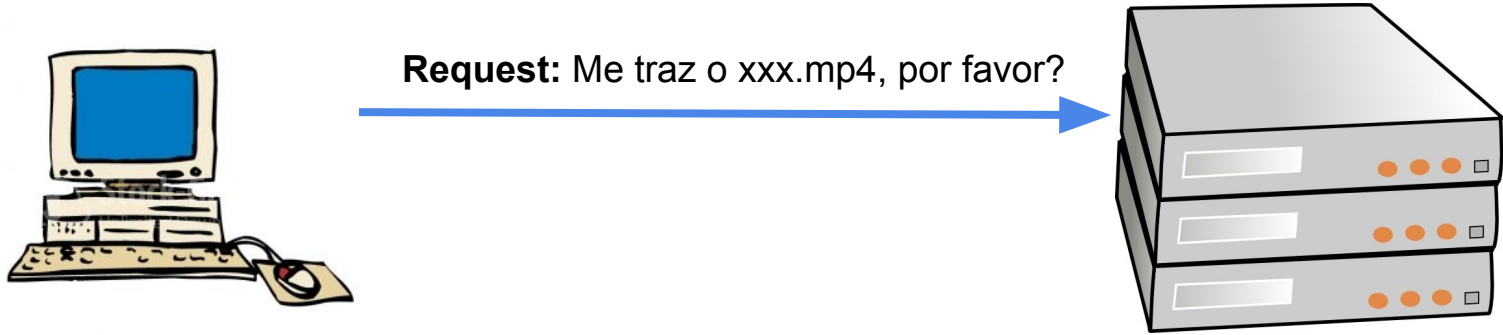


- Protocolo da Camada de Aplicação de Redes
- Utiliza TCP como protocolo da Camada de Transporte
- **Protocolo de requisição/Resposta:** Cliente faz uma requisição ao servidor, o servidor responde ao cliente.
- Possui diferentes métodos -> GET, POST, PUT, DELETE,...

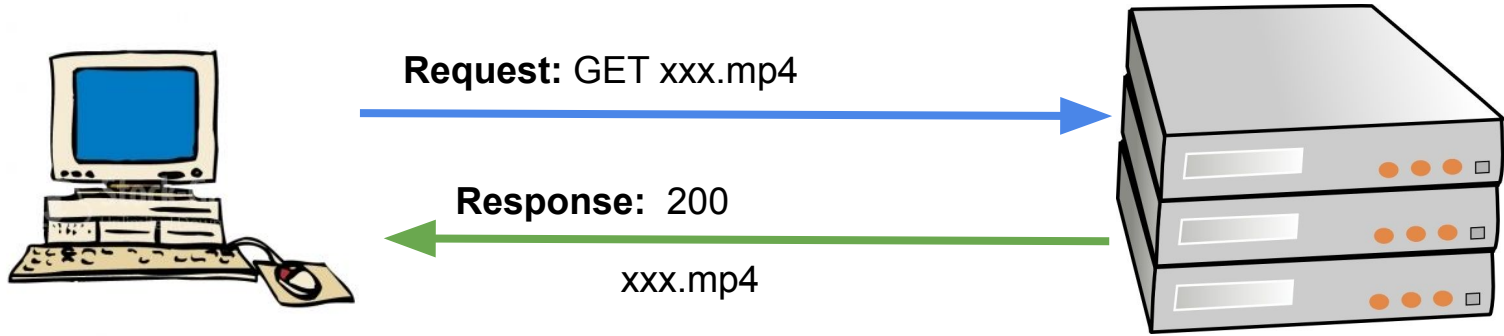
Download de Arquivos por HTTP



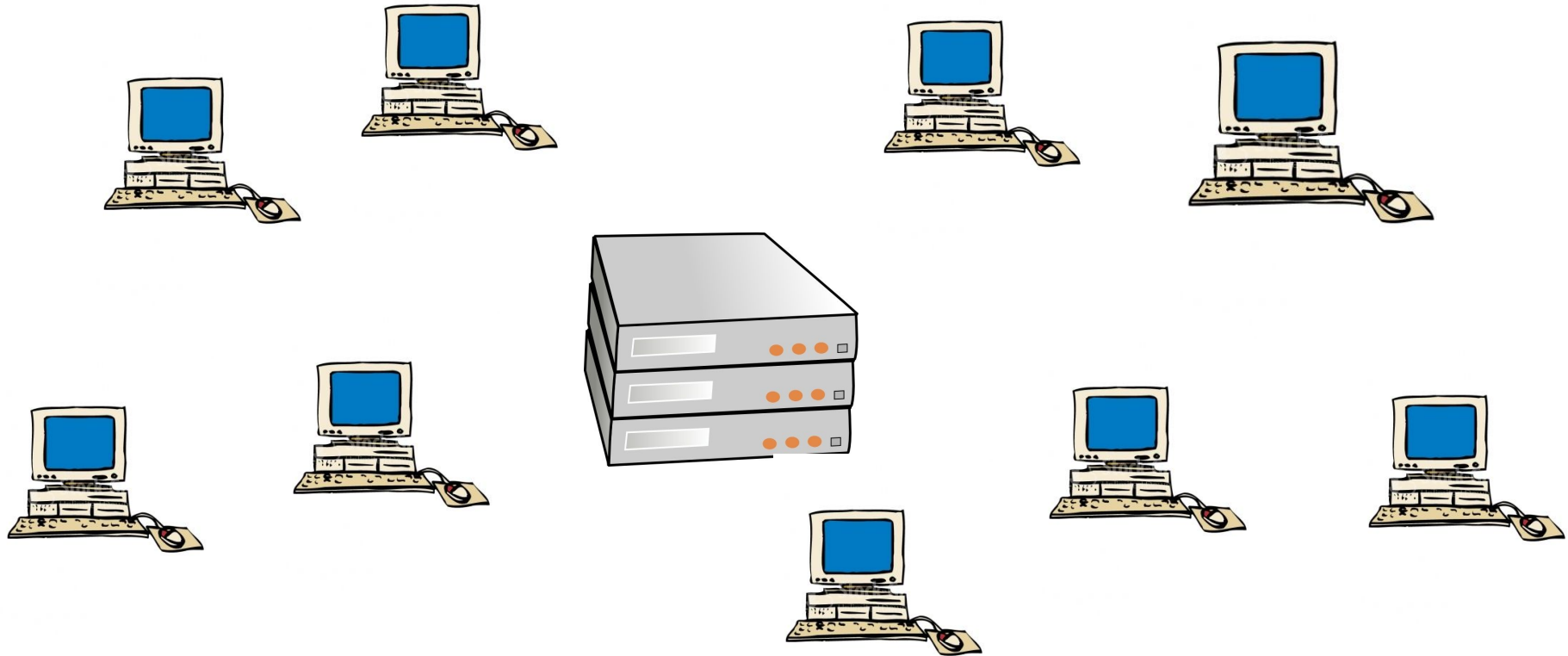
Download de Arquivos por HTTP



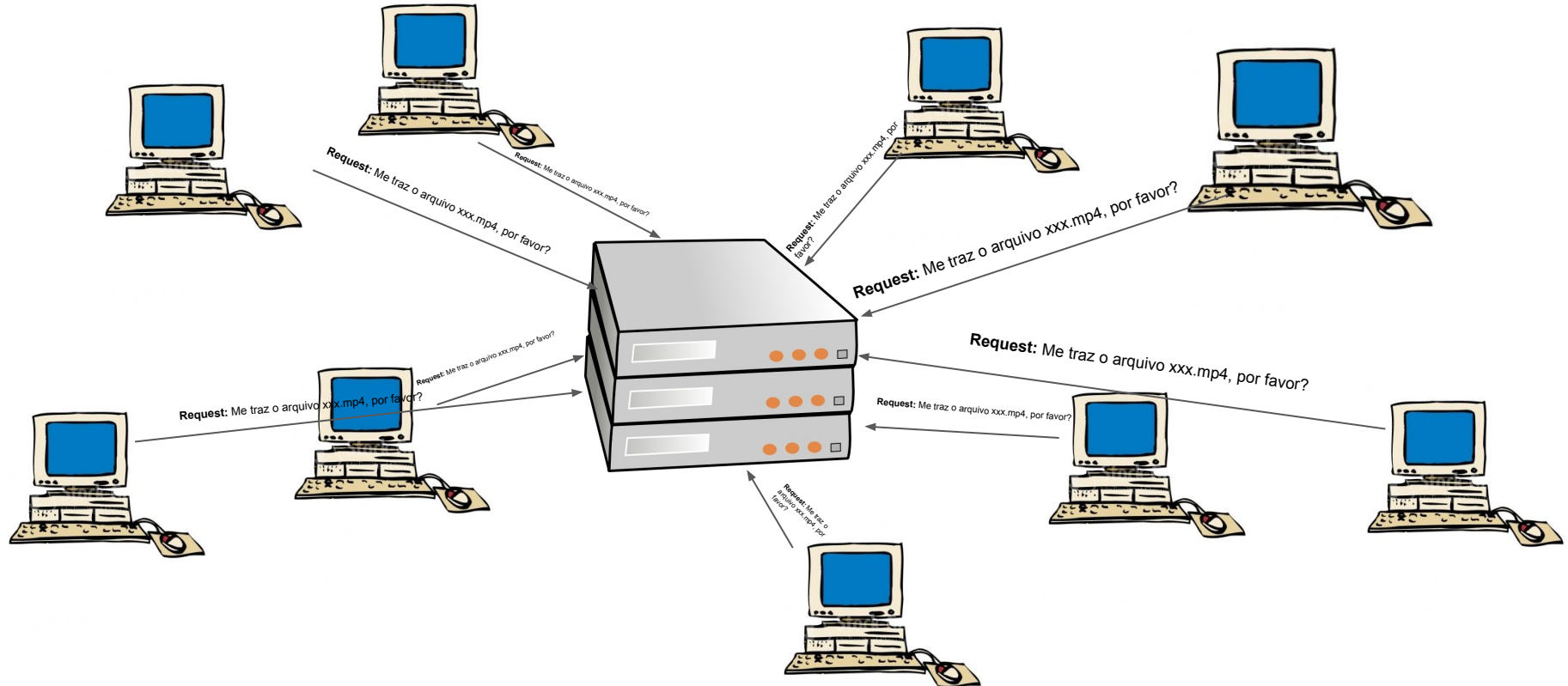
Download de Arquivos por HTTP



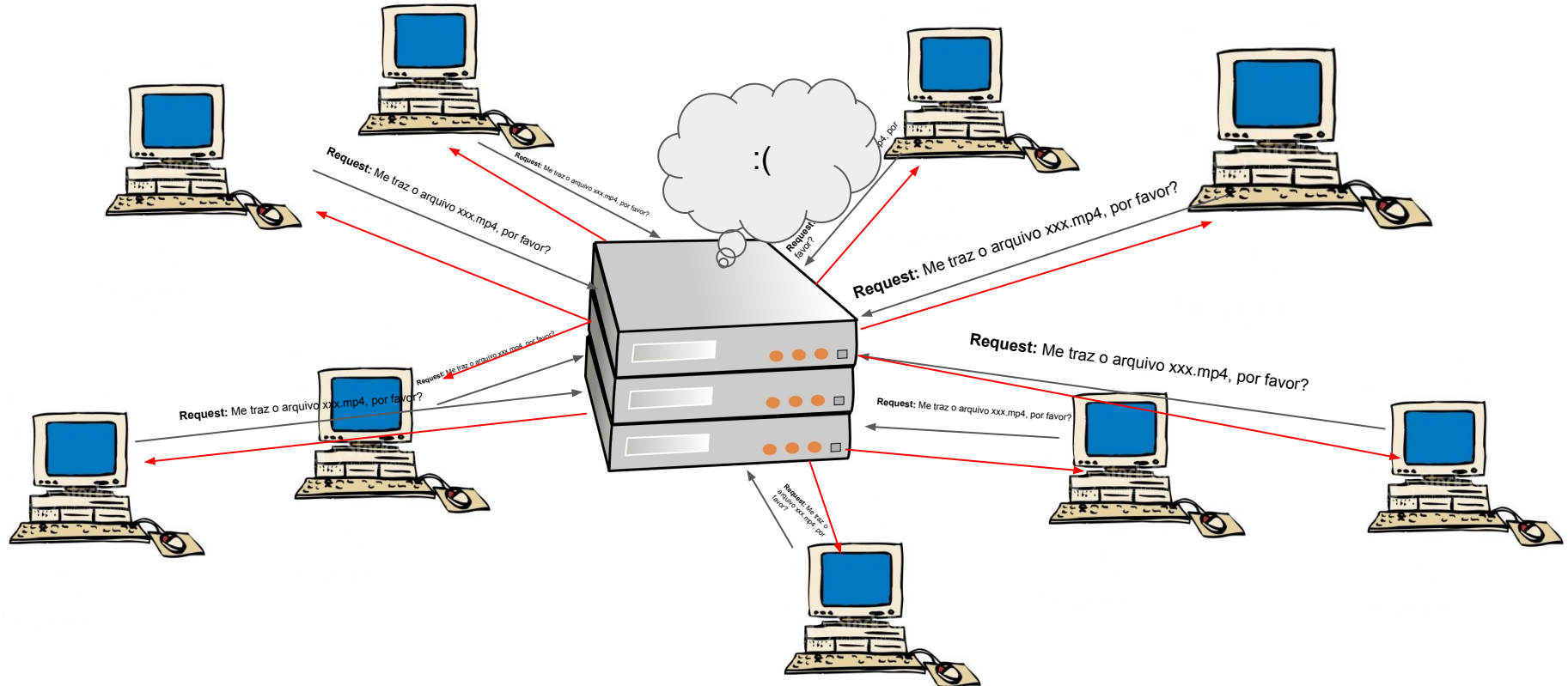
E se tiver muita gente requisitando o mesmo arquivo?



E se tiver muita gente requisitando o mesmo arquivo?



Servidor Sobrecarregado!



E se um(a) corporação/governo não estiver muito contente com o conteúdo do servidor?



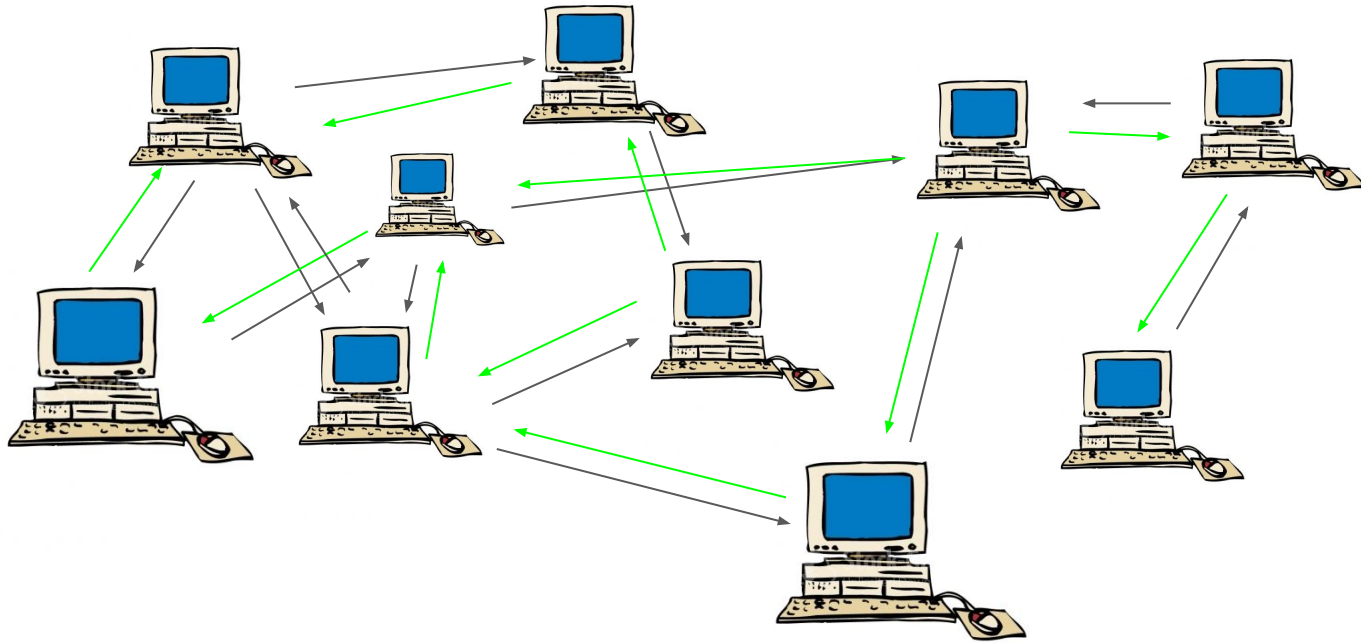
information
wants
to be free



Uma solução para estes problemas:

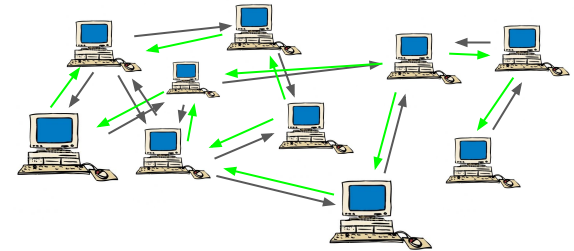
Redes *Peer-to-peer* (P2P)

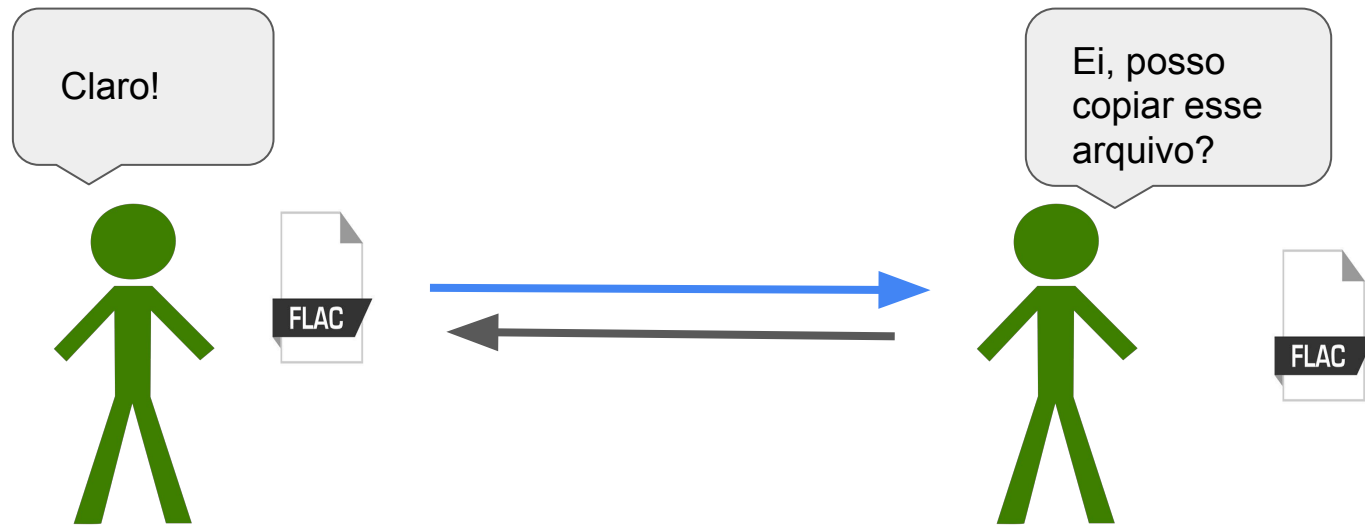
Rede *Peer-to-Peer* (P2P)

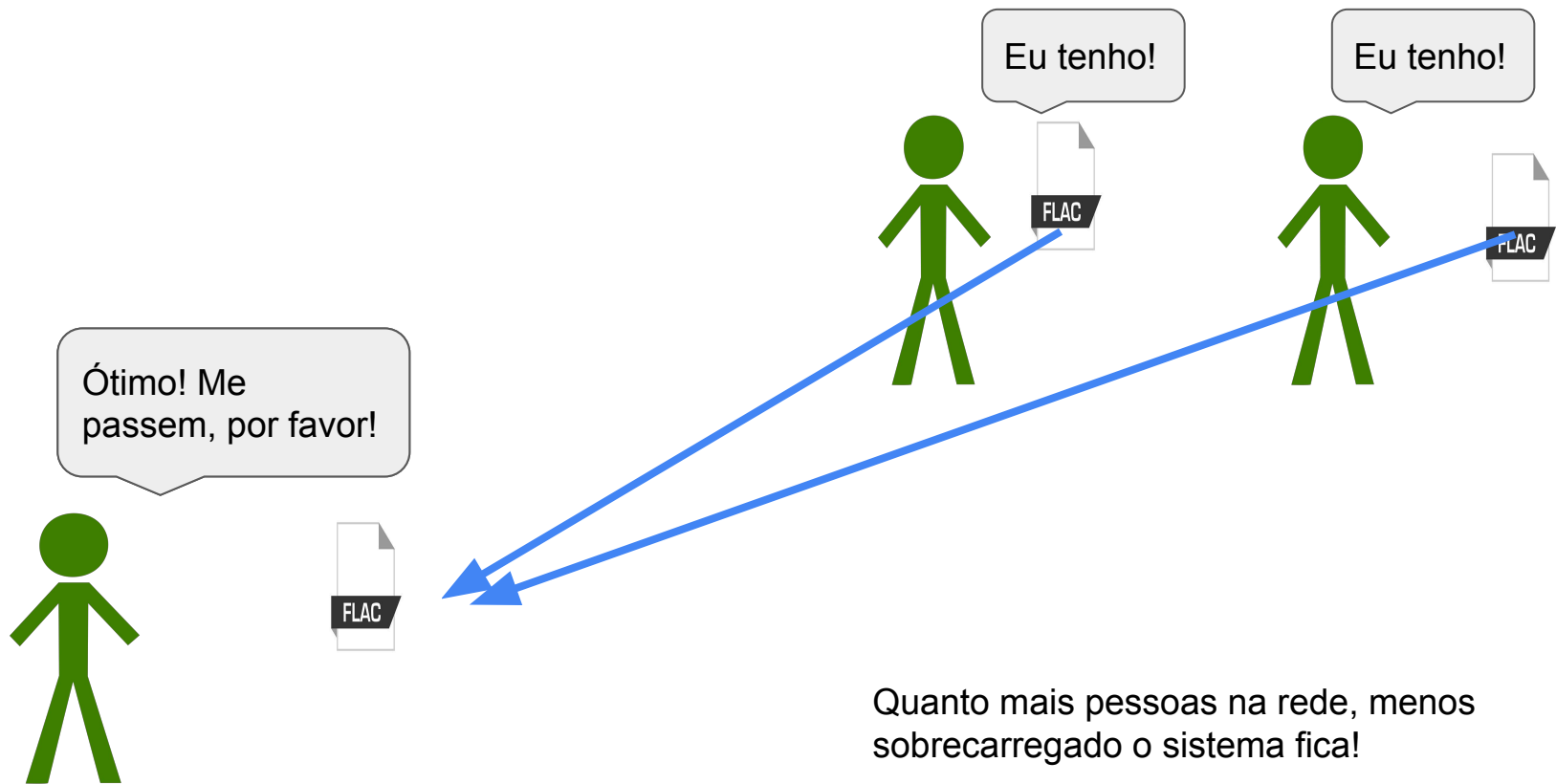


Redes *Peer-to-Peer* (P2P)

Uma rede P2P, diferente de uma conexão centralizada, como a anterior, é distribuída, onde todos os clientes são servidores também..



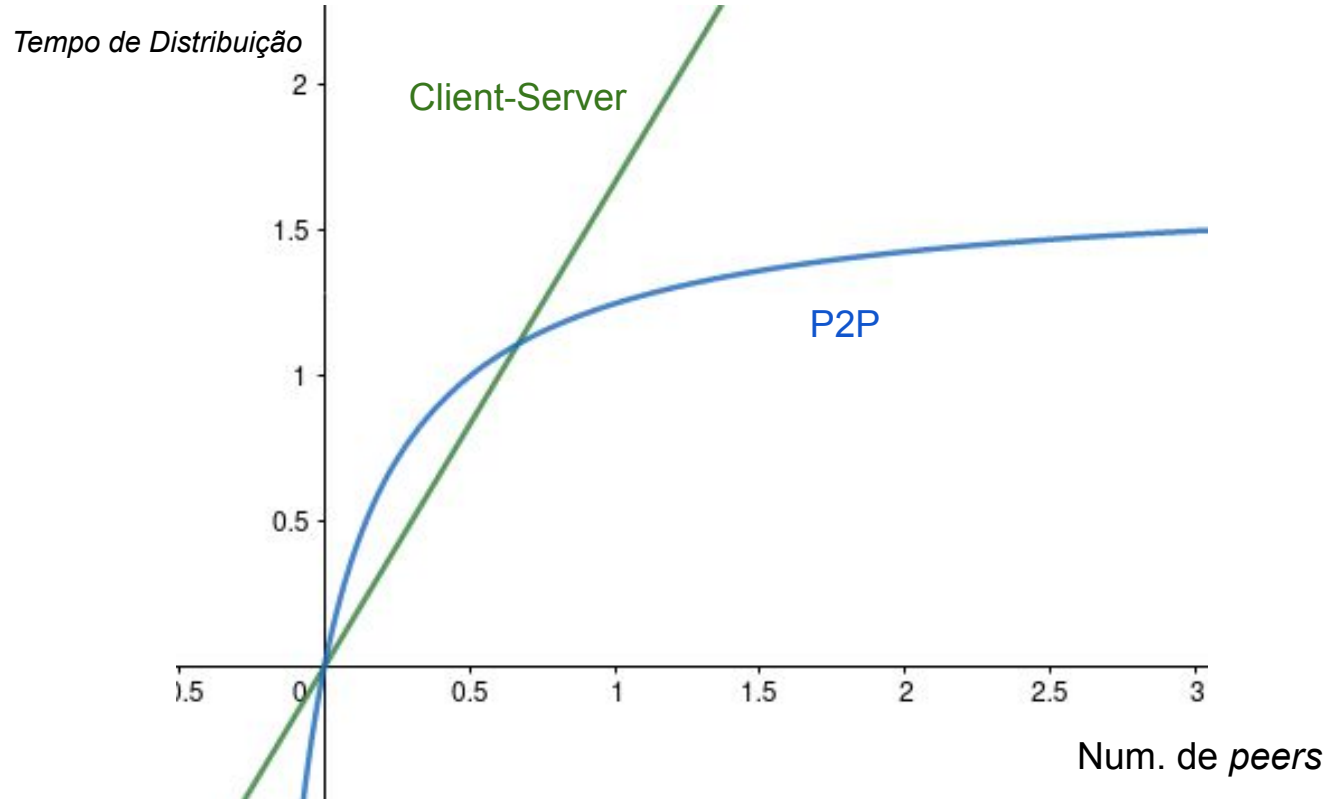




Análise das Distribuições de Arquivos

- Seja um arquivo de tamanho F armazenado num servidor com taxa de *upload* u_s e N clientes.
- Cada cliente possui uma taxa de *upload* u_i e uma taxa de *download* d_i .
- O **tempo de distribuição** é o tempo necessário para que **todos** os N pares obtenham uma cópia do arquivo.
- Suponhamos que o núcleo da Internet tenha largura abundante e que o servidor e os clientes não participam de nenhuma outra aplicação de rede.

Comparando os Tempos de Distribuições CS e P2P



Cliente-Servidor

- Nenhum dos pares auxilia na distribuição do arquivo.
- O servidor deve transmitir uma cópia do arquivo a cada um dos N pares. Assim, ele deve transmitir DF bits no total.
- A taxa de *upload* do servidor é u_s , logo, o tempo de distribuição do arquivo (D_{cs}) deve ser, pelo menos, NF/u_{cs} bits/s.

Cliente-Servidor

- Sendo d_{min} a menor taxa de *download* entre os clientes, sabemos que o cliente com essa taxa finalizará o *download* num tempo de, no mínimo, F/d_{min} segundos.
- Juntando as informações, conseguimos generalizar um limite inferior para o tempo de distribuição:
$$D_{cs} \geq \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$$
- Para N suficientemente grande, no pior cenário, podemos dizer que:

$$D_{cs} = \frac{NF}{u_s}$$

Cliente-Cliente (P2P)

- Apenas o servidor inicial começa com o arquivo.
- Cada par auxilia o servidor (ou nó inicial) distribuindo partes do arquivo, à medida em que vão recebendo.
- Diferente do esquema Cliente-Servidor, um bit enviado pelo servidor pode não precisar ser enviado novamente. Logo, o tempo mínimo de distribuição será dado por F/u_s .
- Assim como no Cliente-Servidor, o par com menor taxa de *download* obtém o arquivo no tempo de, no mínimo, F/d_{min} . Portanto, no mínimo, os arquivos serão distribuídos em F/d_{min} .

Cliente-Cliente (P2P)

- A capacidade total de *upload* do sistema é dada pela soma de todas as taxas de *upload* dos pares e do servidor. Assim:

$$u_{total} = u_s + u_1 + \dots + u_N = u_s + \sum_{i=1}^N u_i$$

- O sistema deve entregar F bits para cada um dos pares, dando um total de NF bits.
- Essa entrega não é possível de ser feita a uma taxa acima de u_{total} .
- Assim, sabemos que o tempo de distribuição também não pode estar abaixo de NF/u_{total} .

Cliente-Cliente (P2P)

Juntando os três limites inferiores deduzidos:

$$D_{P2P} \geq \frac{NF}{u_s + \sum_{i=1}^N u_i} \quad D_{P2P} \geq \frac{D}{d_{min}} \quad D_{P2P} \geq \frac{F}{u_s}$$

Concluimos que o limite inferior do tempo de distribuição numa rede P2P, é dado por:

$$D_{P2P} \geq \max\left\{\frac{F}{u_s}, \frac{NF}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^N u_i}\right\}$$

Cliente-Cliente (P2P)

Como cada par redistribui também os arquivos recebidos e, num sistema bem implementado, isso acontece com uma boa constância, podemos dizer apenas que

$$D_{P2P} = \max\left\{\frac{F}{u_s}, \frac{NF}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^N u_i}\right\}$$

é uma boa aproximação para seu tempo real de distribuição!

Comparando os Tempos de Distribuições CS e P2P

$$D_{CS} = \frac{NF}{u_s} \rightarrow \text{Cresce linearmente em função de } N, \text{ ou seja, do número de pares, no pior cenário.}$$

$$D_{P2P} = \frac{NF}{u_s + \sum_{i=1}^N u_i} \rightarrow \text{No por cenário (} N \rightarrow \infty \text{), sua variação diminui conforme o número de pares aumenta e ela é limitada em } F/u_{\text{medio}}.$$

$$\lim_{N \rightarrow \infty} \frac{NF}{u_s + Nu_{\text{medio}}} = \frac{F}{u_{\text{medio}}}$$

O Protocolo *BitTorrent*

- Permite o compartilhamento de arquivos de forma “descentralizada”!
- Arquivos são compartilhados de forma paralela entre os *peers*
- Cada *peer* envia/recebe um “pedaço” do arquivo, com o tamanho variando de acordo com a divisão escolhida
- *Peers* que fazem download de um arquivo, também podem, de forma concorrente, fazer upload dos “pedaços” que já possuem.
- Domínio público :)

Alguns Conceitos do BitTorrent

- **Peer** -> Qualquer nó da rede, ou seja, máquina que esteja baixando/subindo.
- **Leecher** -> *Peer* que está fazendo download.
- **Seeder** -> *Peer* que já finalizou o download, estando apenas fazendo upload.
- **Ratio** -> Razão entre o quanto foi recebido e o quanto foi enviado.
- **Client** -> Programa que gerencia os *torrents* no lado do usuário.
- **Seeding** -> Estar realizando **apenas** upload (*seeder*)
- **tracker** -> Servidor que armazena informações dos *peers* e dos *torrents*, para que estes se conectem.

Transferência de Arquivo pelo BitTorrent

1. É criado um arquivo `.torrent` contendo metadados do Torrent.
2. Ao abrir o arquivo no *client*, o *seeder* realiza um anúncio ao tracker.
3. O tracker armazena a *hash* do *torrent* e relaciona com o ip e porta do *seeder*.
4. O usuário que quer baixar o torrent, abre o arquivo e faz um anúncio ao *tracker*.
5. Obtendo os outros *peers* do *torrent*, o download se inicia.
6. A medida que o *peer* vai baixando os pedaços, ele vai fazendo upload destes aos *peers* que não os possuem

Aprofundando no BitTorrent

BEP - BitTorrent Enhancement Proposals

- O BitTorrent é mantido pelo *BitTorrent Community Forum*.
- Qualquer pessoa pode fazer uma proposta de modificação/incrementação no protocolo.
- O único requisito é que sua proposta esteja em domínio público.
- Quando uma proposta é submetida, um dos editores do *bittorrent.org* atribui um número BEP.
- Todos os BEPs estão indexados em www.bittorrent.org/beps/bep_0000.html
- Essa apresentação possui o BEP 3 como base.

Criação do Torrent

- O arquivo é dividido em pedaços iguais (exceto pelo último que pode ser truncado)
- Utiliza-se um tamanho em potência de 2 para cada pedaço (geralmente 256KB ou 1MB)
- Para cada pedaço é feito um *checksum* que gera uma *hash*, a fim de calcular a integridade dos dados posteriormente.
- No arquivo .torrent as *hashs* de cada pedaço serão salvas em ordem.
- Sabendo o tamanho e o índice de cada pedaço = recuperação dos dados

Arquivos .torrent

Os arquivos .torrent são dicionários codificados em *bencode*.

info: Dicionário que descreve o(s) arquivo(s) do torrent.

announce: A url de anúncio do *tracker*. (ex: *http://tracker.share.net/announce*)

creation date (opcional): A data de criação do torrent.

comment (opcional): Texto livre para comentários do autor do .torrent.

created by (opcional): Nome e versão do programa usado para criar o .torrent.

Dicionário *info*

name: Nome do arquivo/diretório raiz dos arquivos.

piece-length: Número de *bytes* em cada pedaço

pieces: Contém as hashes em SHA1 de cada pedaço no *index* correspondente

length: Tamanho do *download* em *bytes* caso tenha apenas **um** arquivo.

files: Lista de dicionários com os tamanhos e caminhos de cada arquivo caso haja **mais de um arquivo**.

Arquivo

a
b
c
d
e
f
g
h

Dicionário *info*

(representado em JSON)

```
{  
  "name":path,  
  "piece-length":256000,  
  "length":1800000,  
  "files":[  
    checksum(a),  
    checksum(b),  
    ...  
    checksum(h)  
  ]  
}
```

Peer Protocol

- Protocolo do BitTorrent usado entre os *peers*.
- Opera em TCP ou uTP ([BEP 29](#))
- Conexões entre os *peers* são simétricas
- *Peers* compartilham os pedaços do(s) arquivo(s) entre si.
- Cada pedaço é identificado pelos seus índices.
- Quando o *download* de um pedaço é finalizado, é feito um anúncio a todos os *peers* conectados.

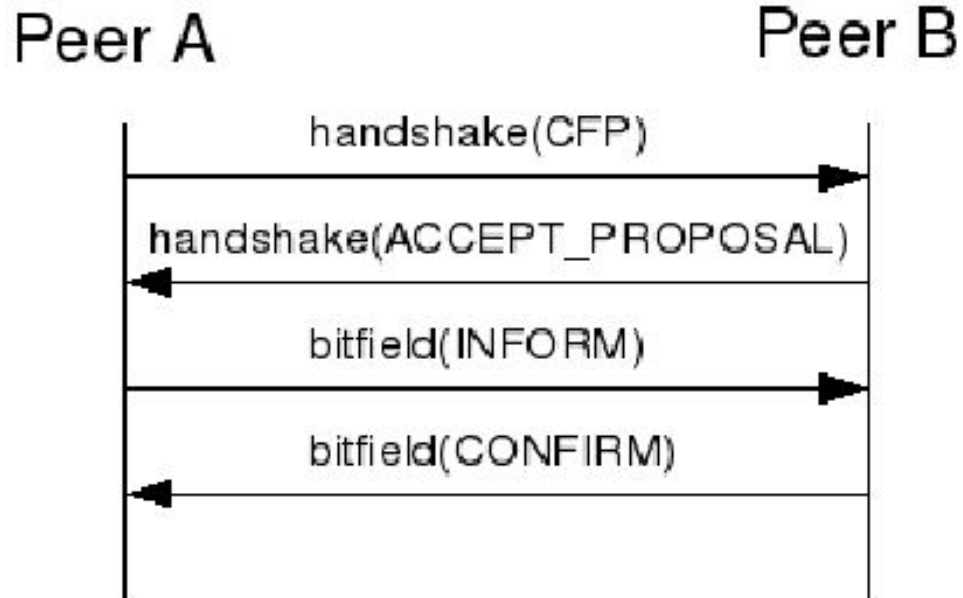
Peer Protocol - Handshake

A conexão começa com um *handshake* entre os *peers*:

- ***Header***: Inteiro 19, seguido por pela *string* “BitTorrent Protocol”.
- Após o *header*, é enviado 8 bytes de valores nulos;
- 20 *bytes* da *info_hash*, que representa o *checksum* do torrent;
- 20 *bytes* do *peer id* que deve estar presente no *tracker*;

Após o *handshake*, se estabelece a conexão entre os *peers*.

Peer Protocol - Handshake



Peer Protocol – Comunicação

- As mensagens entre os *peers* são trocadas no formato *tamanho:string*.
- Mensagens de tamanho zero são *keepalives*, sendo ignoradas.
- Cada *peer* tem um estado representado por 2 *bits*:
 - ***chocked*** -> Enquanto verdadeiro, define que nenhum dado será enviado.
 - ***interested*** -> Define se um *peer* está “interessado” em uma transferência.
- Todas as conexões começam como ***chocked*** e ***not interested***.

Peer Protocol – Esquema das Mensagens

A mensagem começa com um *byte*, que representa o tipo:

- | | | |
|------------------|----------------------|---------------|
| - 0 - choke | - 3 - not interested | - 6 - request |
| - 1 - unchoke | - 4 - have | - 7 - piece |
| - 2 - interested | - 5 - bitfield | - 8 - cancel |

choke, unchoke, interested e not interested não têm *payload* – são mudanças de estado

Peer Protocol – Tipos de Mensagens

bitfield:

- usado apenas como a primeira mensagem.
- ***Payload:*** um *bitfield* com cada índice que o *downloader* enviou setado para 1 e o resto 0.

have:

- ***payload:*** um número contendo o índice do pedaço que o *downloader* acabou de completar, já tendo verificado o checksum

Peer Protocol – Tipo de Mensagens

Request:

- ***index:*** índice correspondente do pedaço
- ***begin, length:*** *byte offsets* representando o início do pedaço e seu tamanho

piece:

- ***index, begin***
- ***piece:*** bytes do pedaço a ser transferido

cancel:

- Mesmo *payload* que o ***Request***.
- É enviado no fim do *download*

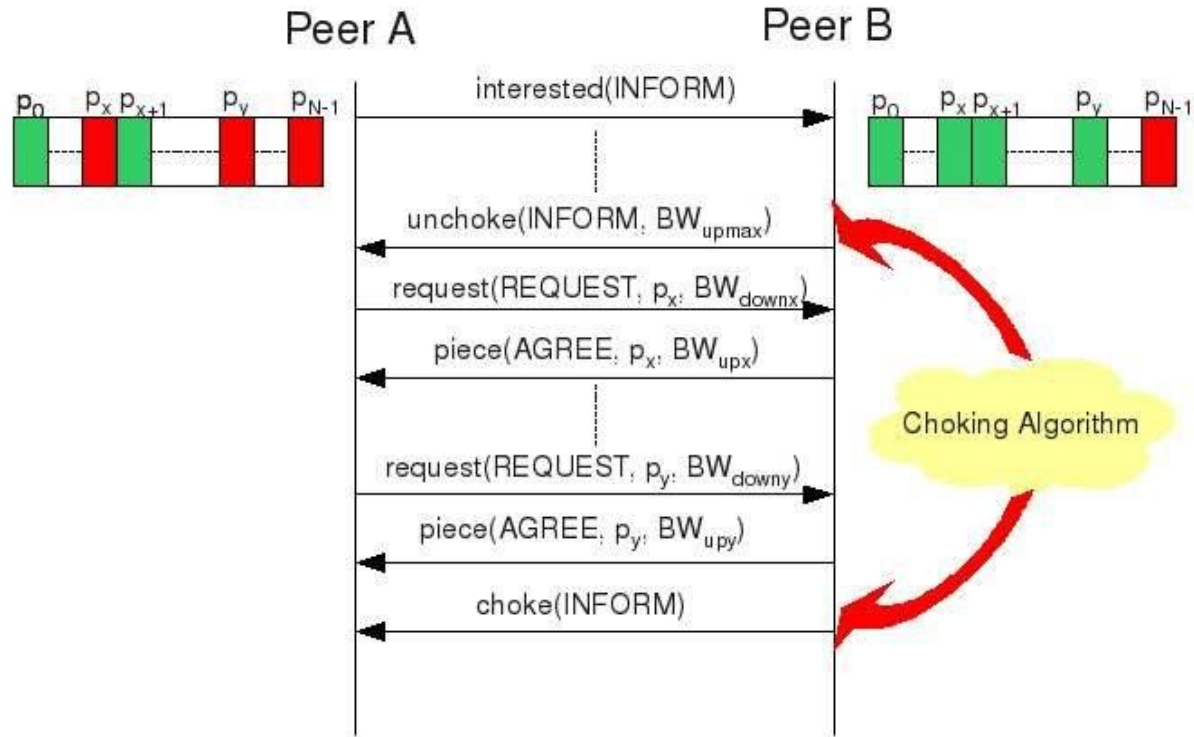
Peer Protocol – choking

- ***Choking***: Parar, temporariamente, de fazer *upload*.
- O *peer* decide de quais outros *peers* ele irá baixar cada pedaço.
- **Algoritmo de *choking*** :
 - Busca utilizar todos os recursos disponíveis na rede
 - Prover taxas de *download* consistentes a todos os *leechers*
 - Penalizar *leechers* que apenas fazem *download*

Peer Protocol – choking

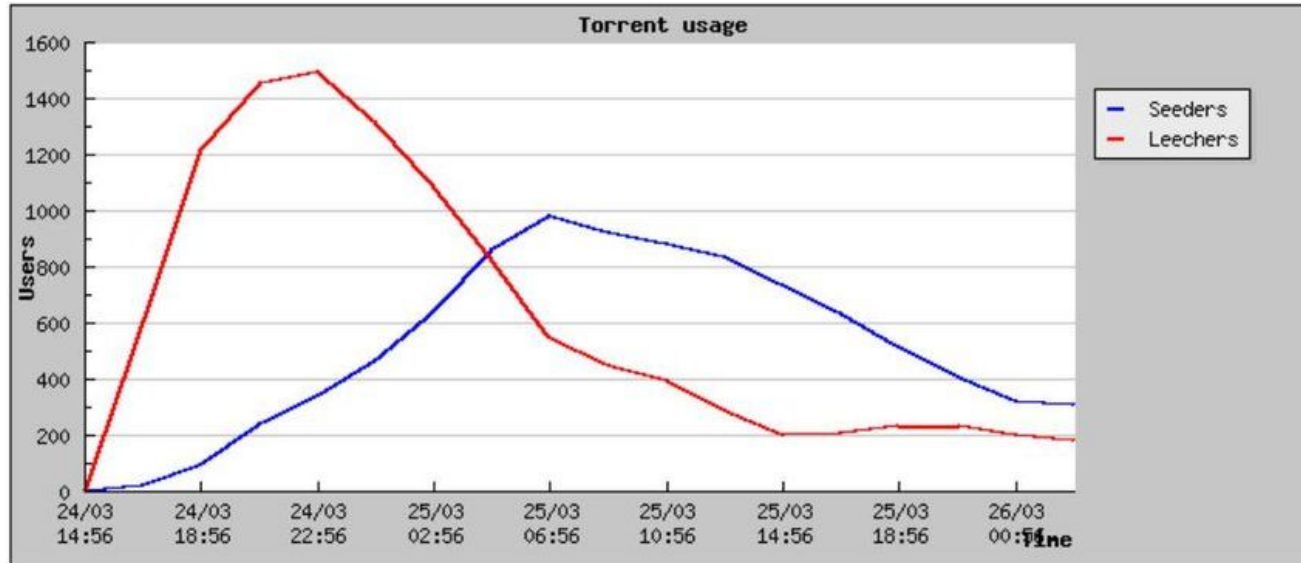
- **Algoritmo de *Chokin* do BitTorrent:**
 - Quanto maior a taxa de download do *peer*, maior sua penalidade
 - Quanto maior a taxa de *upload*, maior sua prioridade
 - Quando um *peer* finaliza o download, ele permanece fazendo *upload* (*seeder*).
 - O *upload* dos *seeders* são priorizados aos *leechers* com maiores taxas de *upload*.

Peer Protocol – Comunicação



Peer Protocol

Na Prática:



<https://www.bittorrent.org/bittorrentecon.pdf>

Entidades da Distribuição do Torrent

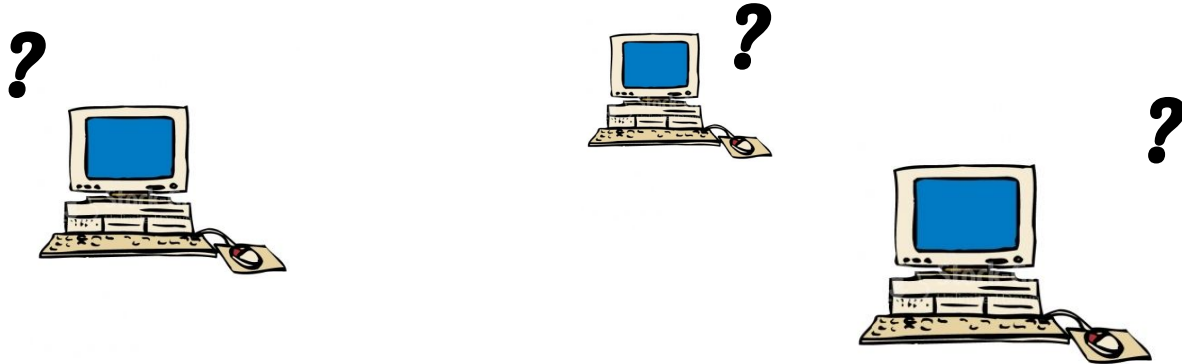
- Servidor Web
- Um arquivo *[.torrent]* com metadados do *torrent* (ou um *link magnético*)
- *Peers*
- Um *tracker* BitTorrent (Não precisa caso haja DHT)
- Um *client* BitTorrent para os *peers*

Servidor Web

- Servidor Web comum, usa HTTP
- Armazena e disponibiliza aos clients **APENAS** os arquivos .torrent.
- Servidores públicos geralmente possuem ferramentas de buscas e indexação dos .torrents



Como um *peer* consegue encontrar outros
peers do mesmo Torrent?



Tracker

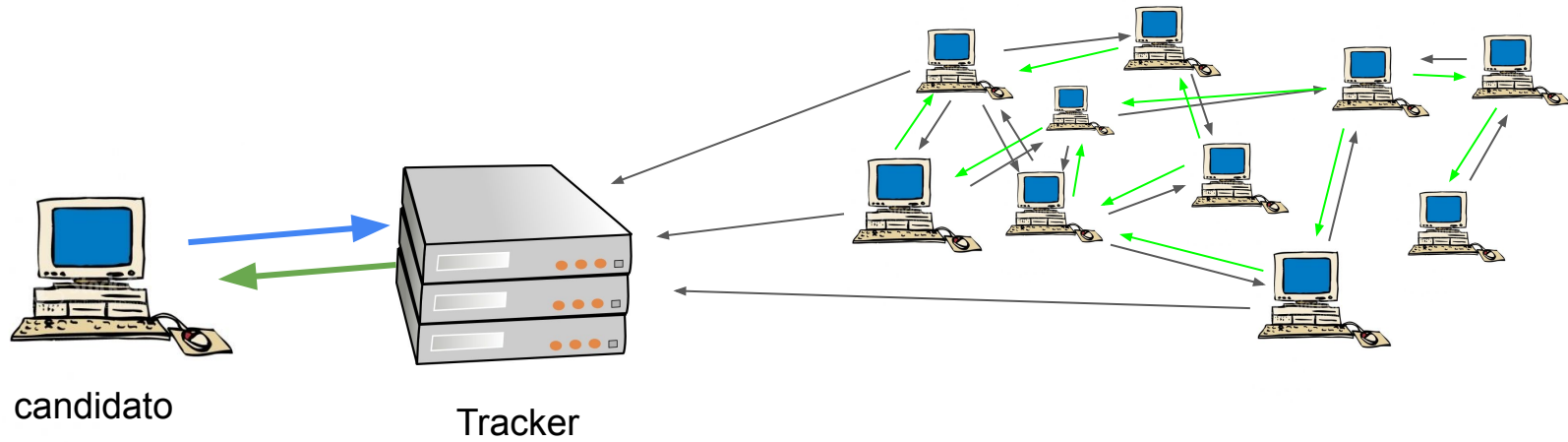
- Servidor HTTP ou UDP ([BEP 15](#))
- Armazena e distribui informações sobre os *peers* de um torrent, permitindo que eles se “encontrem”.
- Os principais dados que o *Tracker* necessita armazenar para o BitTorrent funcionar são:
 - *info_hash* do Torrent
 - IP de cada *peer* e a porta onde o *client* está hospedado
- A maioria dos *trackers* hoje são UDP

Tracker HTTP

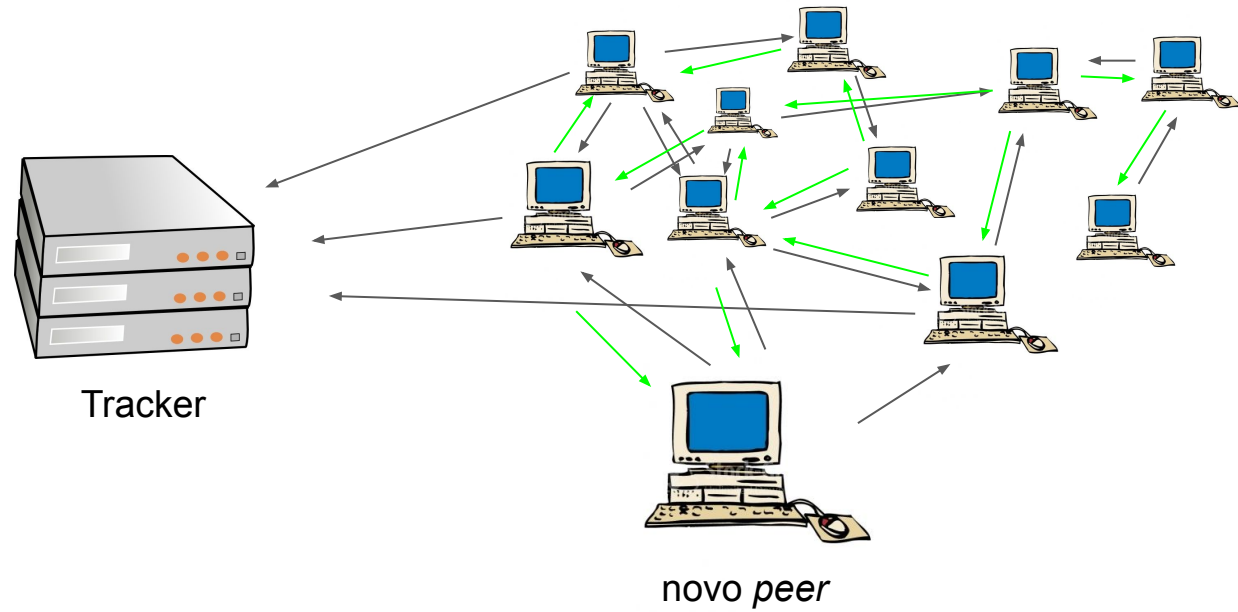
Requisições de Anúncio (GET):

- ***info_hash***: *checksum* do *torrent*
- ***peer_id***: um id gerado pelo *peer* aleatoriamente
- ***IP***: Endereço IP do *peer* (*opcional*)
- ***port***: A porta em que o *peer* está hospedando o *client*.
- ***uploaded***: O quanto já foi feito de *upload*
- ***downloaded***: O quanto já foi feito de *download*
- ***left***: Quanto falta para o *download* finalizar
- ***event***: Informa o estado do *peer* → *started*, *completed*, *stopped* (*opcional*)

Rede de *Peers* do torrent



Rede de *Peers* do *torrent*



Problema...

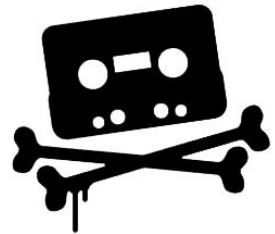
- Ter apenas um servidor *tracker* acaba centralizando mais a rede
- Mas sem o *tracker*, *peers* não têm como adivinhar quem possui o *torrent*...

DHT - *Distributed Hash Table*

- Permite que cada *peer* armazene informações dos outros *peers*
- Cada *peer* se torna um *tracker*!
- Implementado no [BEP 5](#), mas como uma extensão
- Utiliza o algoritmo ***Kademlia***

Obrigado :)

guip.me
gelos.club



Bibliografia

- Redes de computadores e a Internet - Kurose F, James
- [The BitTorrent Protocol Specification - BEP 3 - bittorrent.org](#)
- [Extension for Peers to Send Metadata Files - BEP 3 - bittorrent.org](#)
- [https://wiki.theory.org/BitTorrentSpecification](#)
- [Kademlia: A Peer-to-Peer Information System Based on the XOR Metric](#), Petar Maymounkov and David Mazières