BITTORRENT

-

BitTorrent

- The technology has three aspects
  - Many BitTorrent clients are available in open-source
    - Protocol initially written in English (no math, no pseudocode)
    - Attempt of standardize (see bittorrent.org)
  - Many existing clients, mostly open-source
  - A clever idea: using "tit-for-tat" mechanisms to reward good behavior and to punish bad behavior
- This third aspect is especially intriguing!

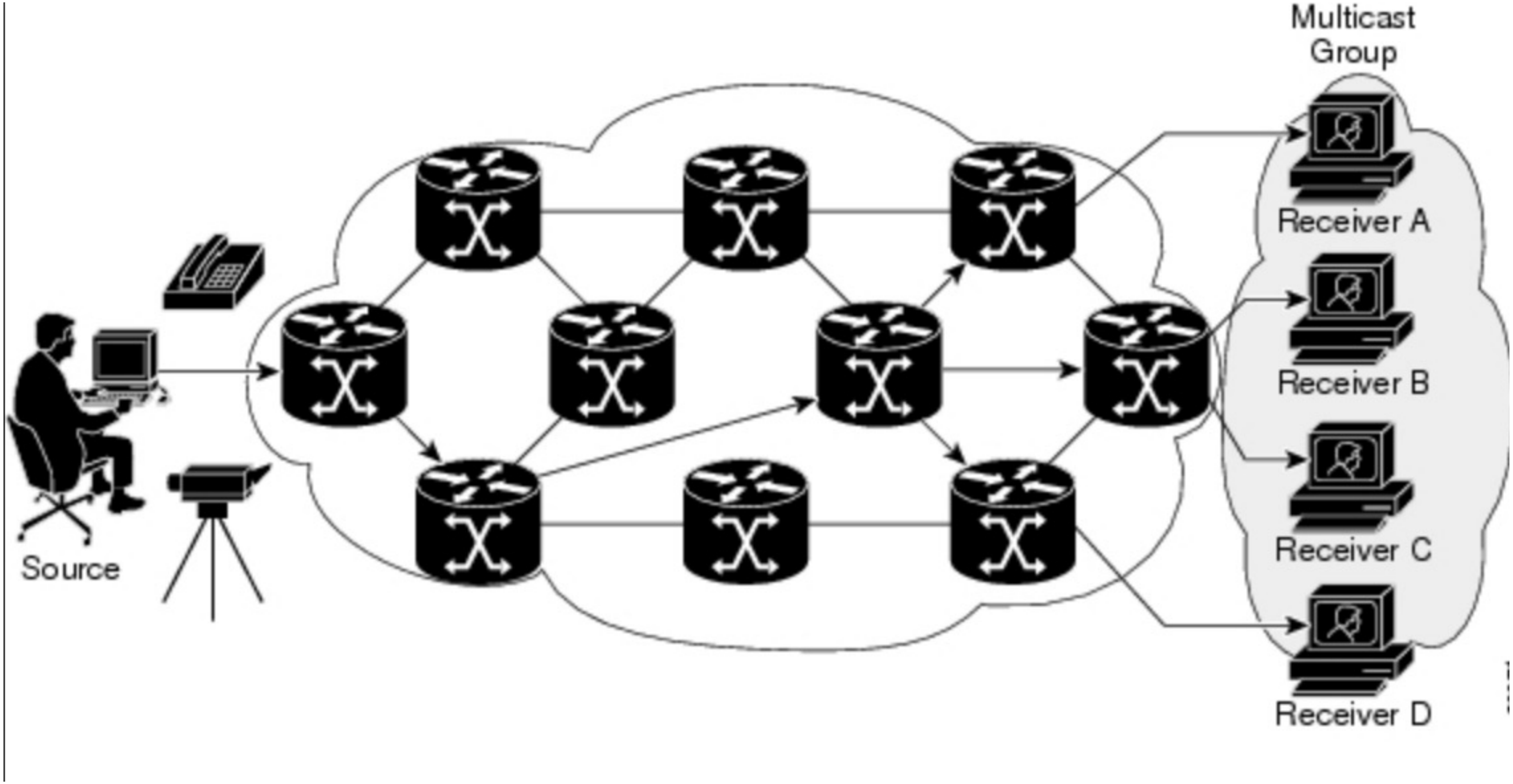
The basic BitTorrent Scenario

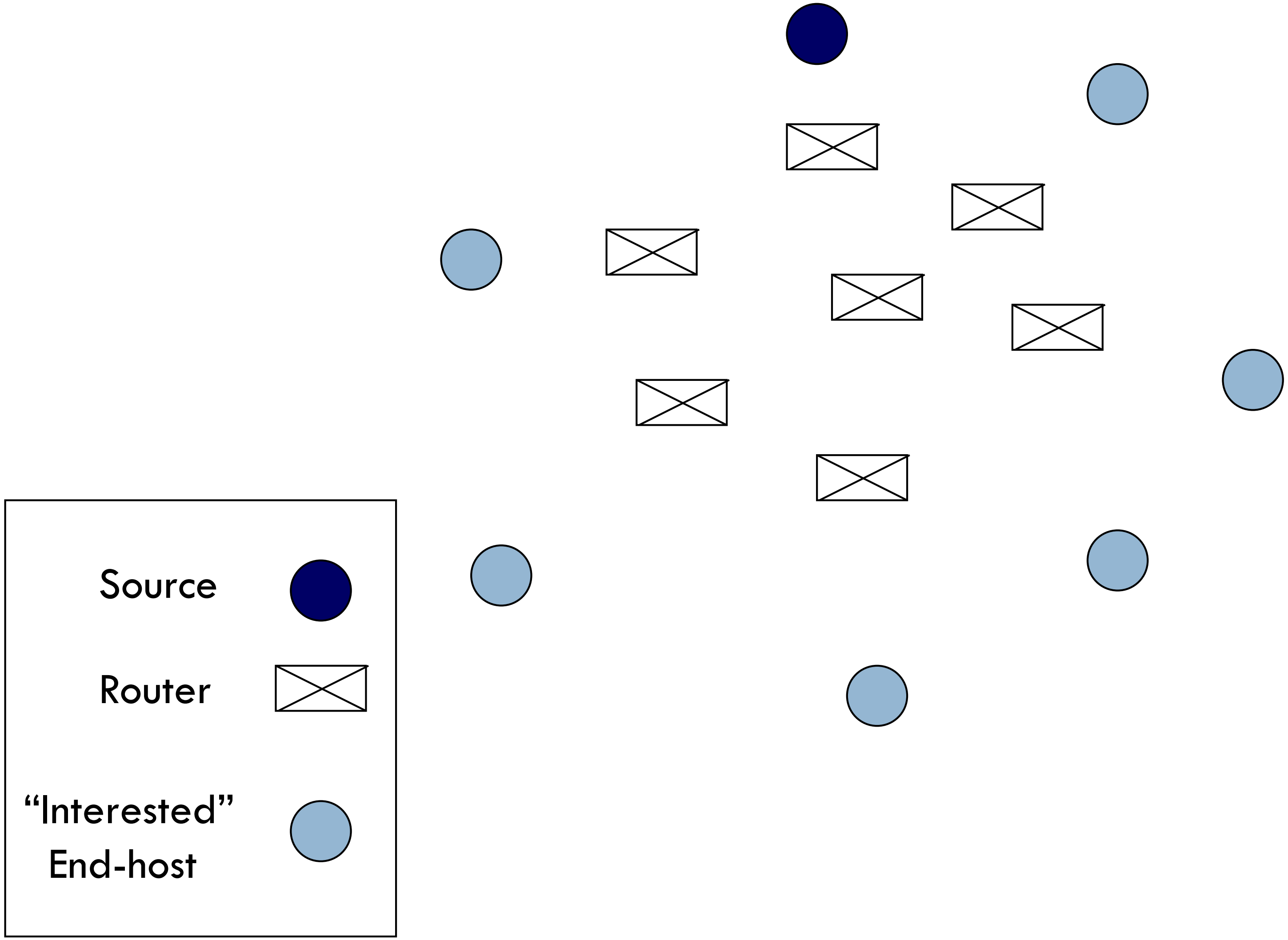
- Millions want to download the same popular huge files (for free)
  - ISO's
  - Media files (the majority !)
- Client-server model fails
  - Single server fails
  - Can't afford to deploy enough servers

Why not use IP Multicast?

- IP Multicast not a real option in general WAN settings
  - Not supported by many ISPs
  - Most commonly seen in private data centers
- Alternatives
  - End-host based Multicast
  - BitTorrent
  - Other P2P file-sharing schemes

IP multicast (only with IPv6)



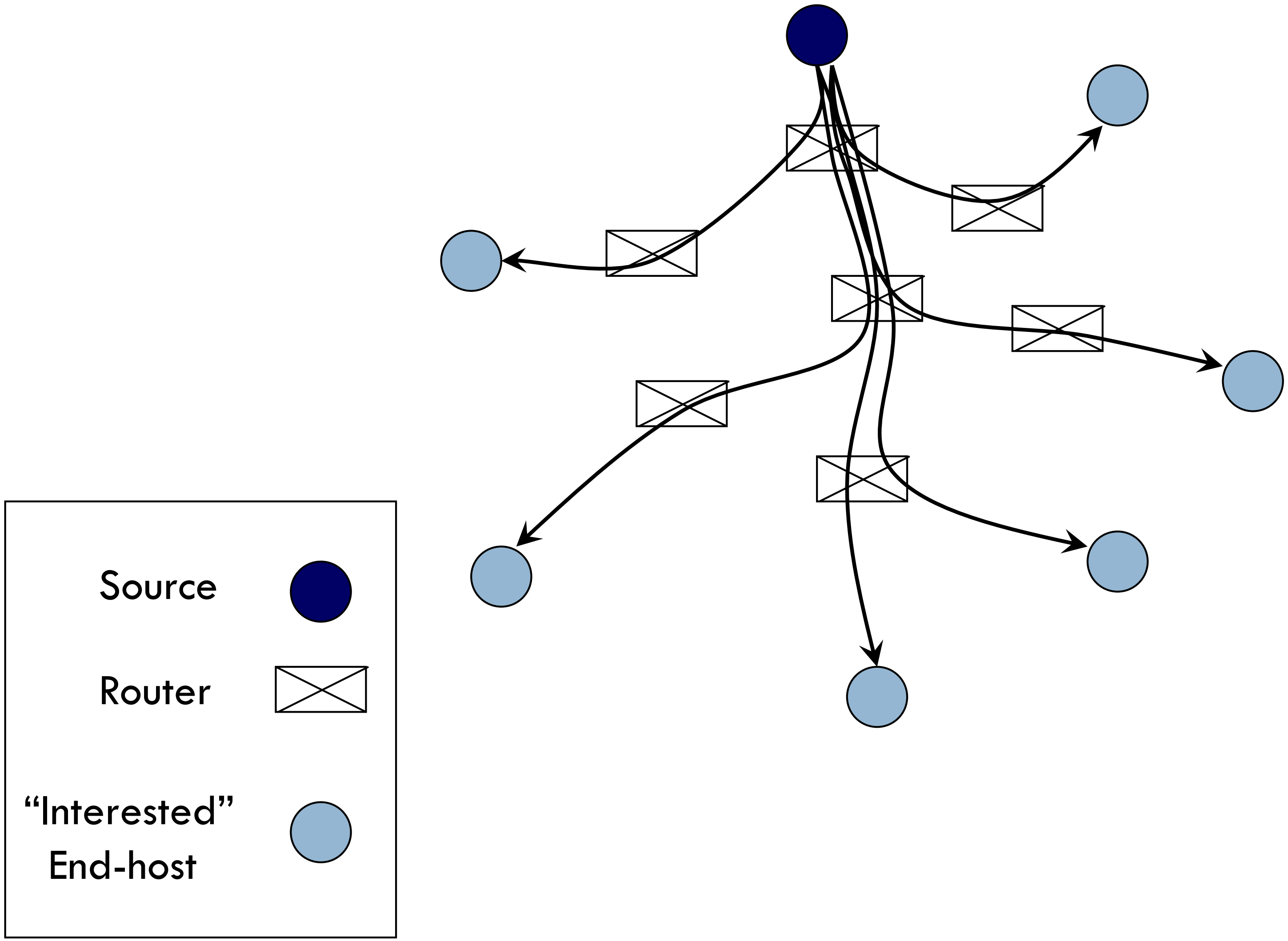


Source ●

Router ☒

"Interested" End-host ●

Client-Server

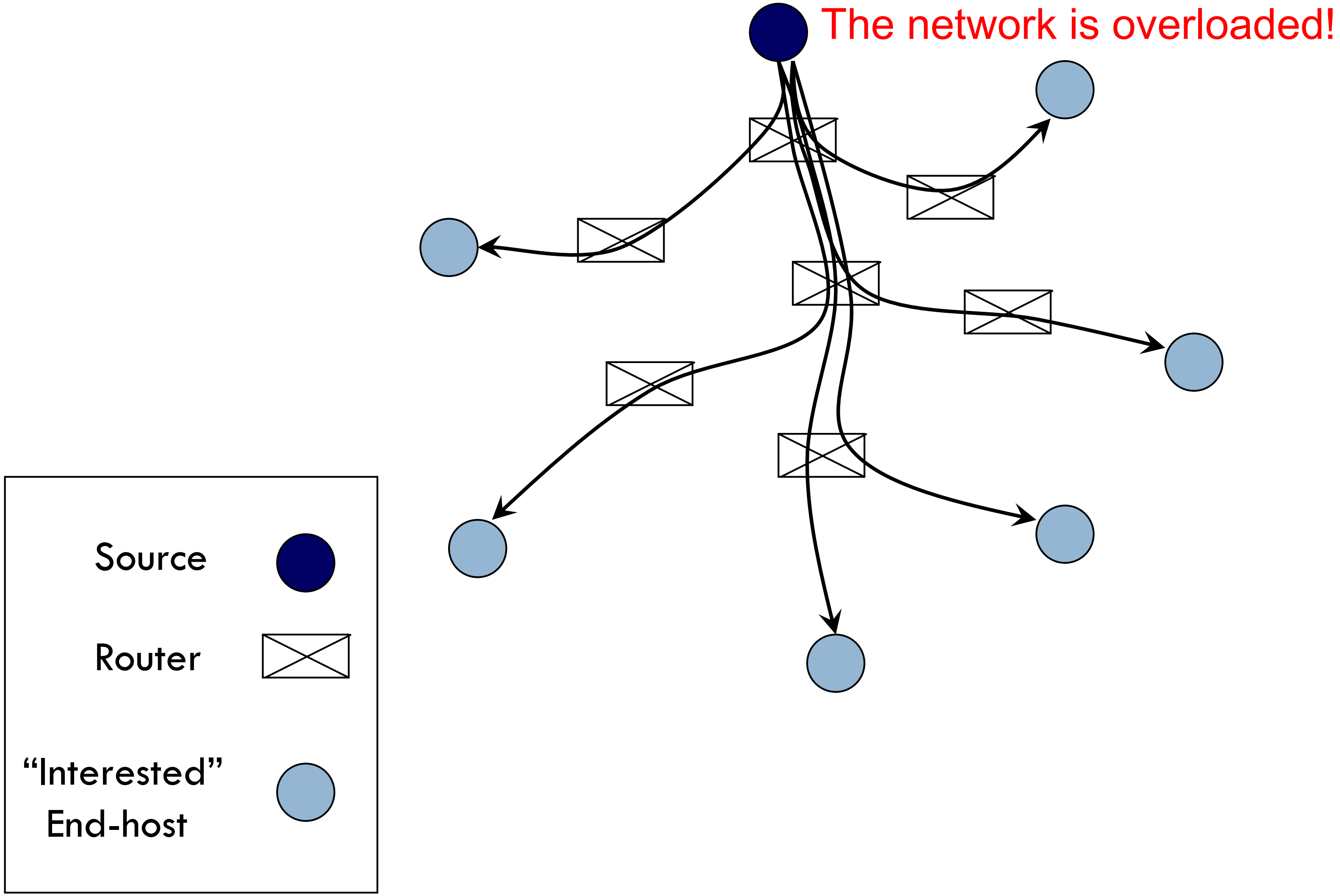


Source ●

Router ☒

"Interested" End-host ●

Client-Server



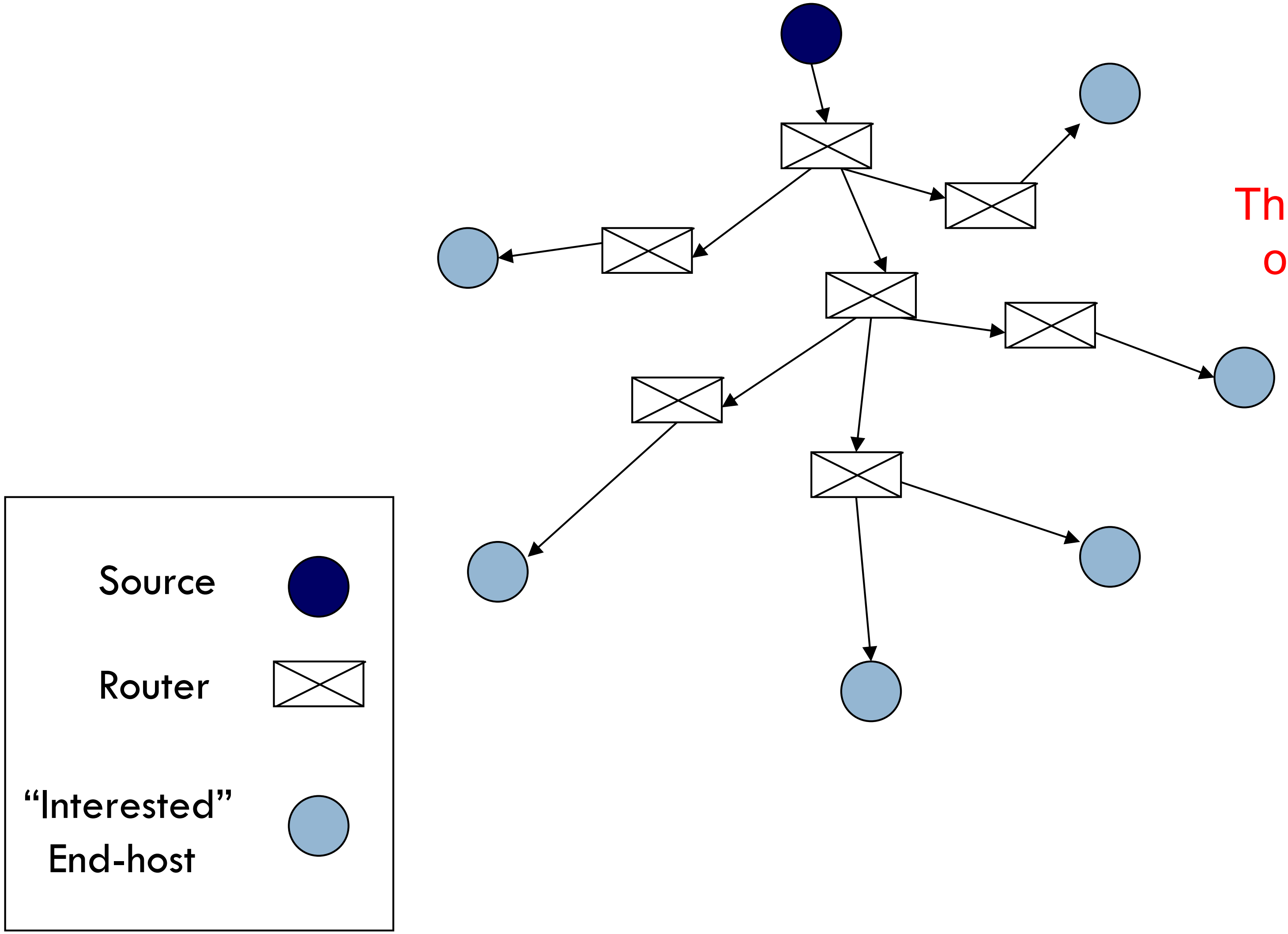
The network is overloaded!

Source ●

Router ☒

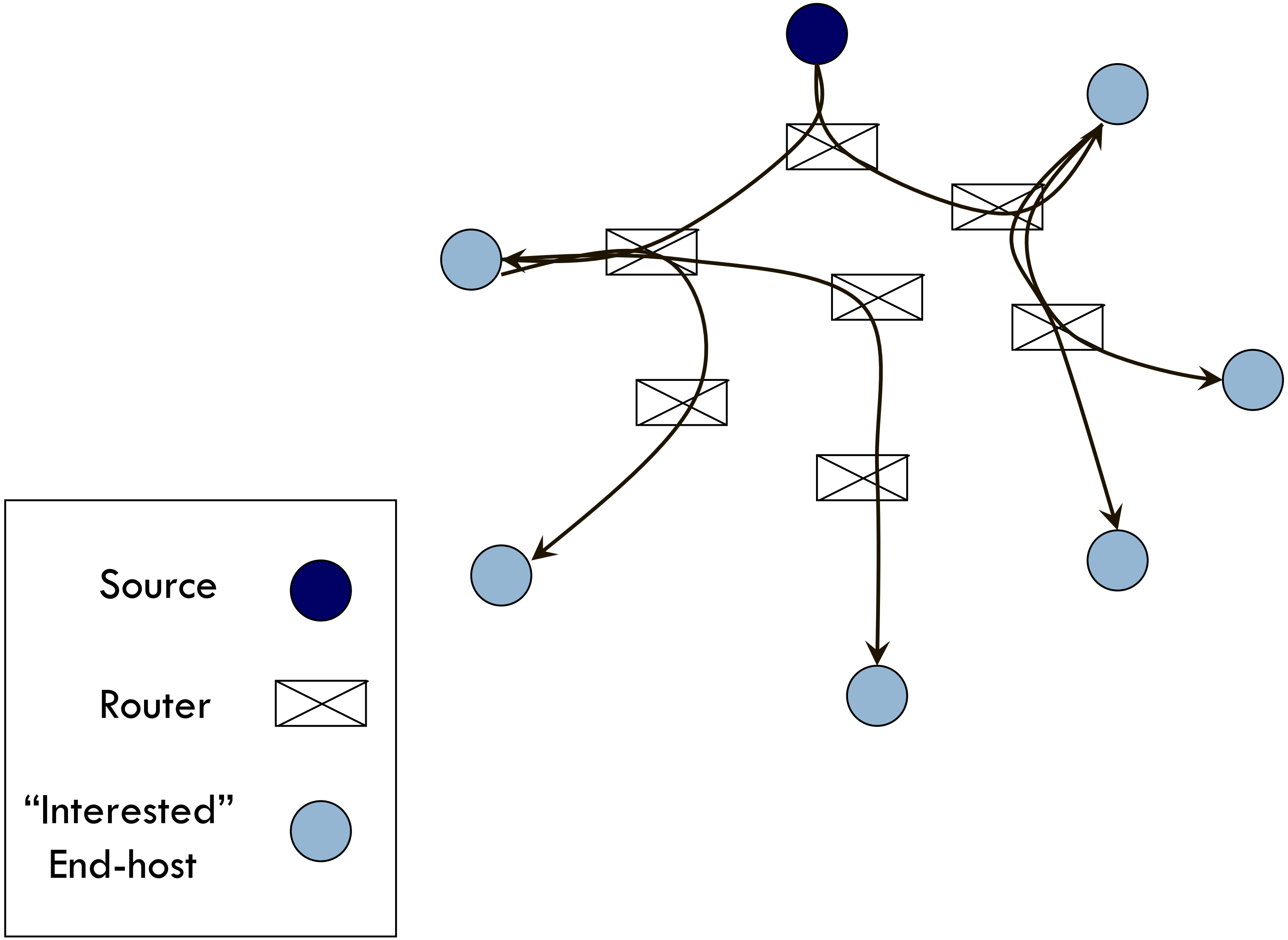
“Interested” End-host ●

(plain) IP multicast



The network is overloaded!

End-host based multicast



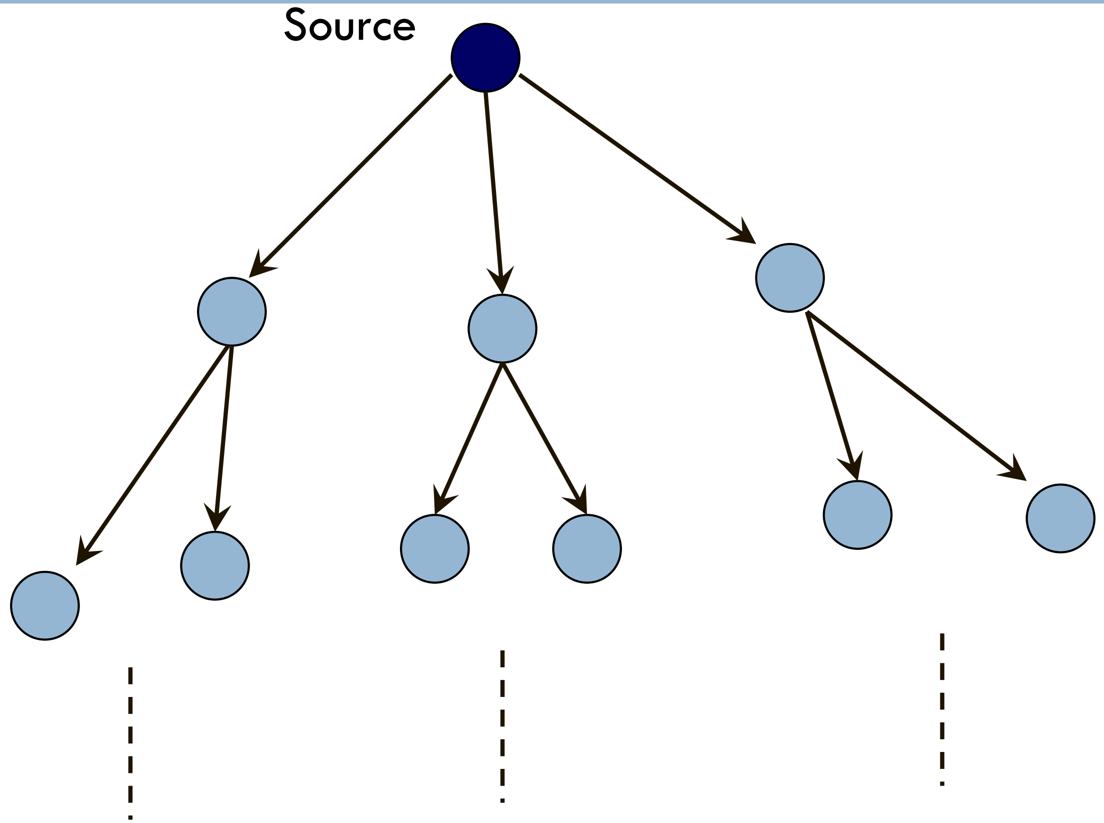
End-host based multicast

- Single-uploader versus Multiple-uploaders
  - Lots of nodes want to <u>*download*</u>
  - Make use of their <u>*uploading*</u> abilities as well
  - Node that has downloaded (part of) file will then upload it to other nodes
  - ➢ Uploading costs amortized across all nodes

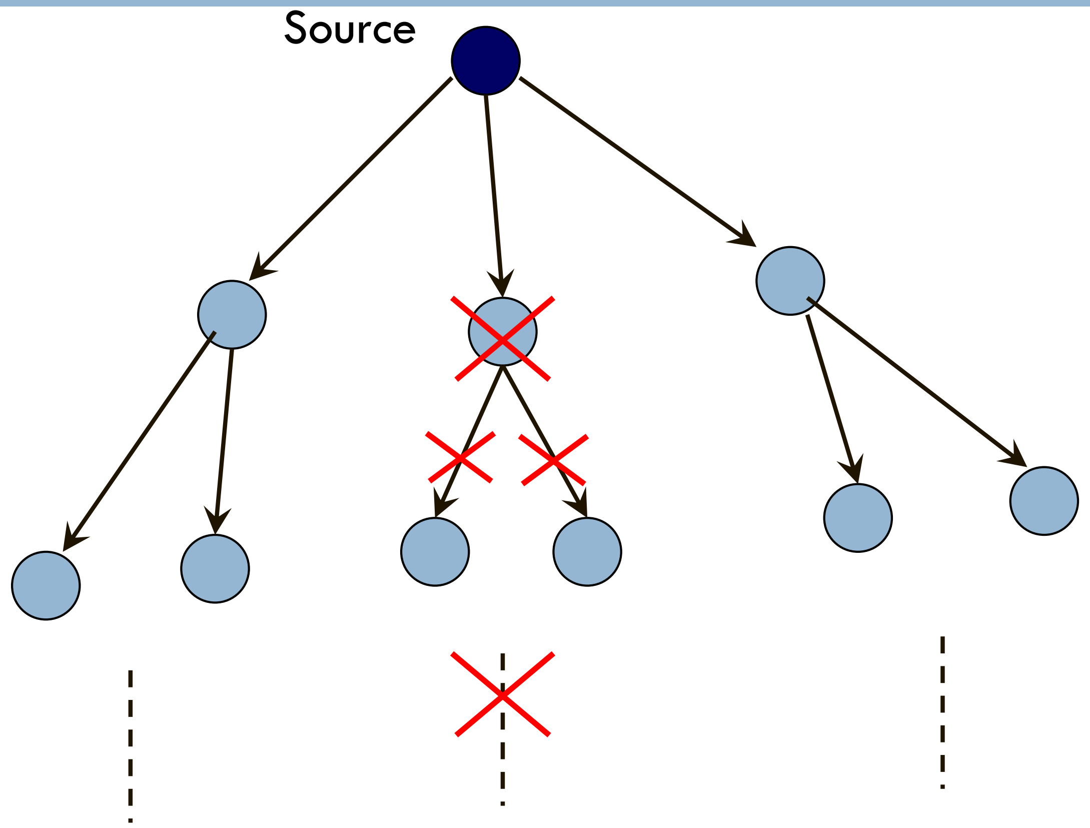
End-host based multicast

- Also called Application-level Multicast
- Many protocols proposed this two decades ago !
- All use single trees
  - Problem with single trees ? Unfortunately yes !

End-host multicast using single tree

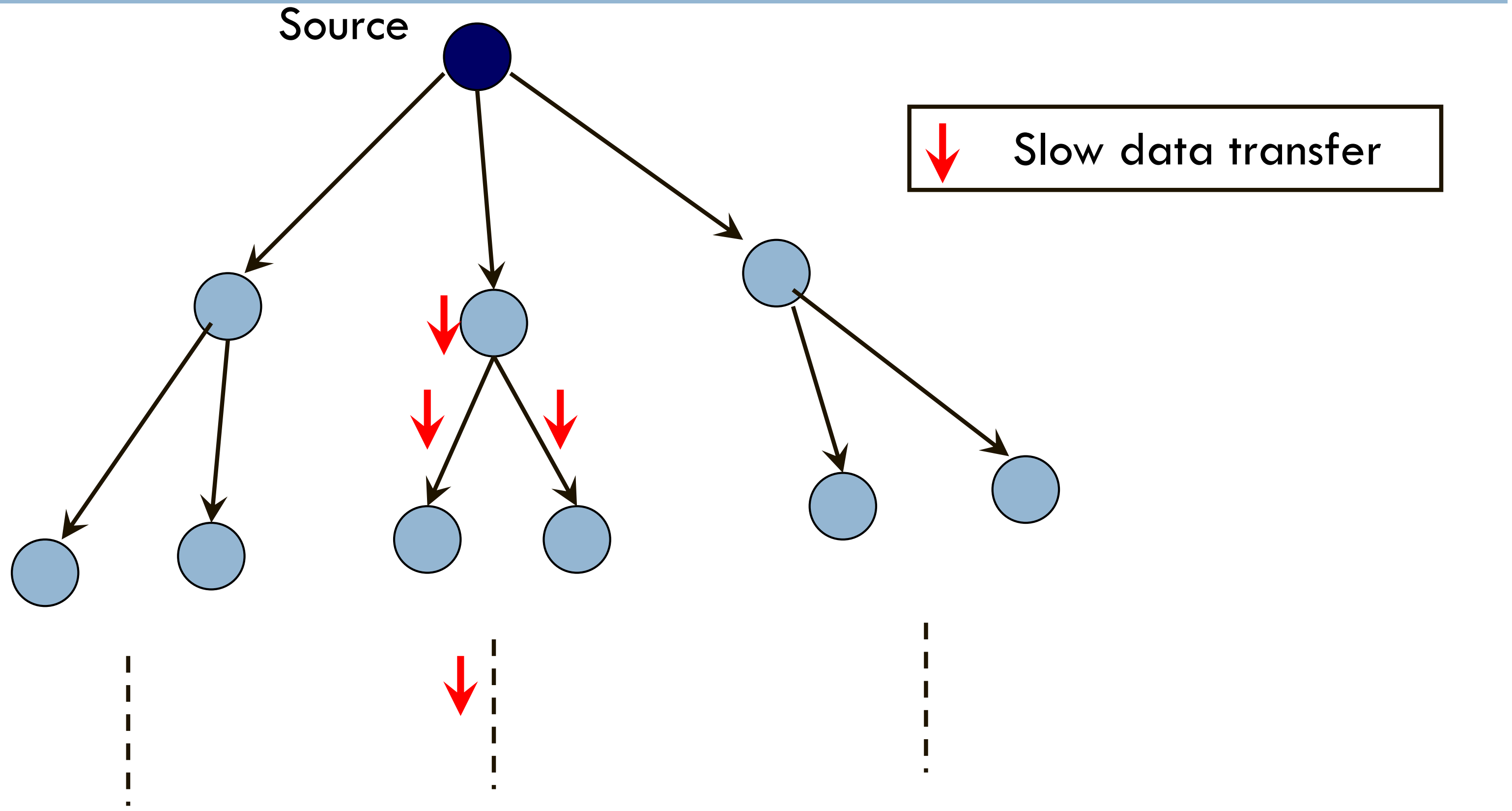


End-host multicast using single tree



A supernode failure
implies that all
subtree nodes fails

End-host multicast using single tree



End-host multicast using single tree

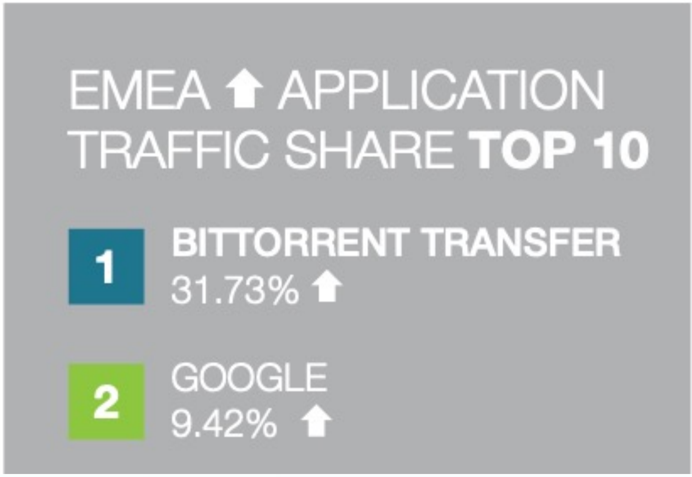
- □ Tree structure is push-based – node receives data, pushes data to children
- □ Failure of interior-nodes affects downloads in entire subtree rooted at node
- □ Slow interior-nodes similarly affects entire subtree
- □ Also, leaf-nodes don’t do any sending!
- □ Though later multi-tree / multi-path protocols mitigate some of these issues, tree are not a good topology !!!

BitTorrent

Give and ye shall receive!

- Written by Bram Cohen (in Python) in 2001
- “Pull-based” & “Swarming” approach
  - Each file split into smaller pieces
  - Nodes request desired pieces from neighbors
    - As opposed to parents pushing data that they receive
  - Pieces not downloaded in sequential order
  - Previous multicast schemes aimed to support “streaming”; BitTorrent does not (but new bittorrent dialects supports streaming … e.g. <201X Spotify used P2P tech)
- Encourages contribution by all nodes

in 2018

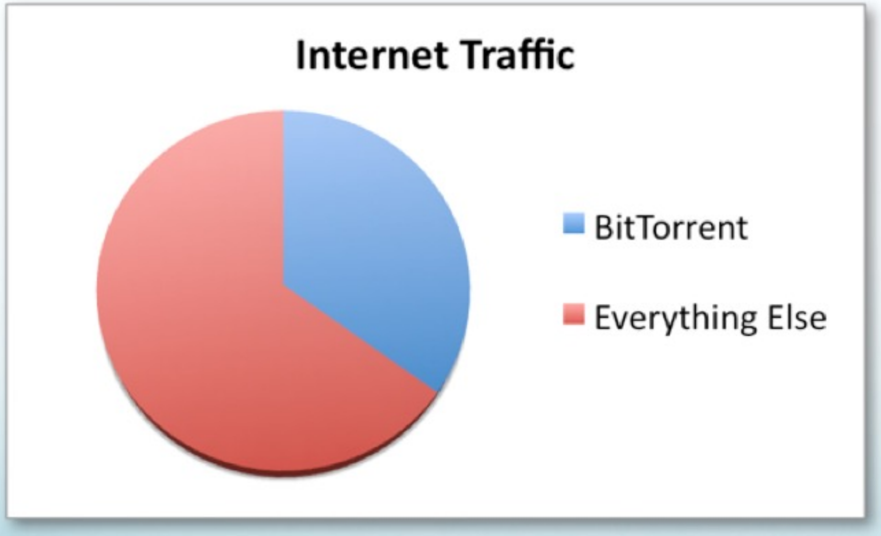


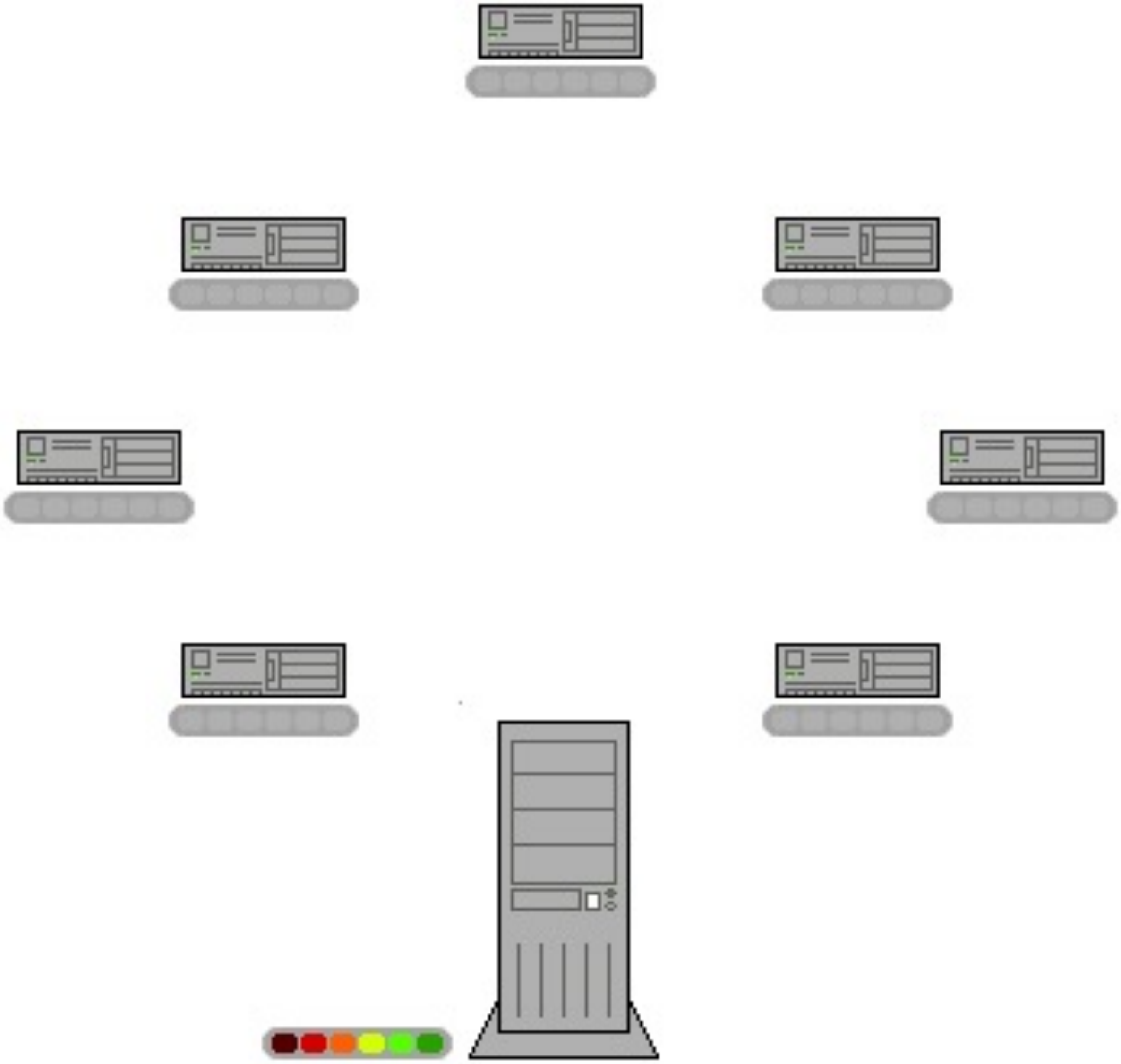
EMEA ⬆ APPLICATION TRAFFIC SHARE **TOP 10**

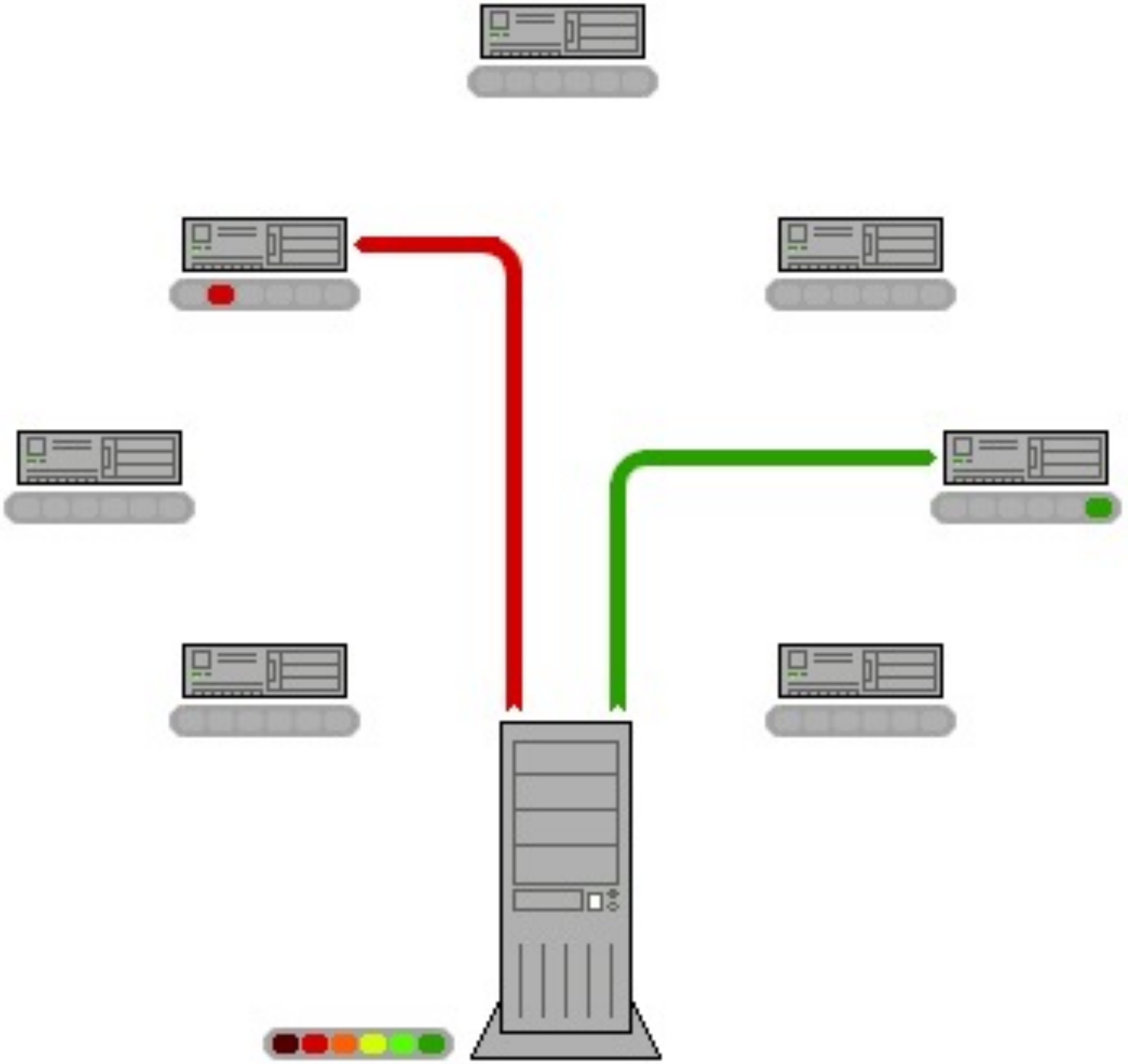
1 **BITTORRENT TRANSFER**
31.73% ⬆

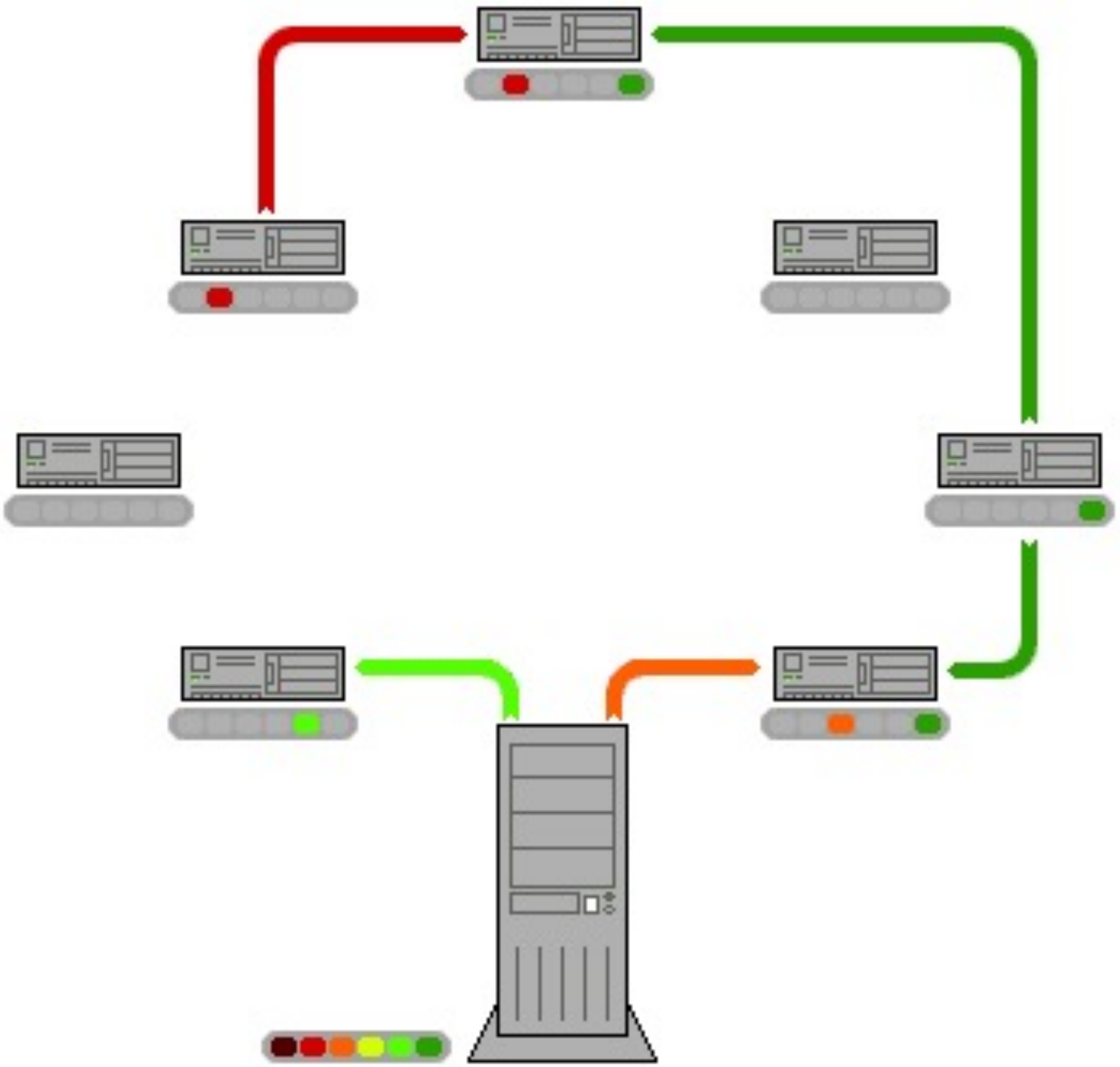
2 GOOGLE
9.42% ⬆

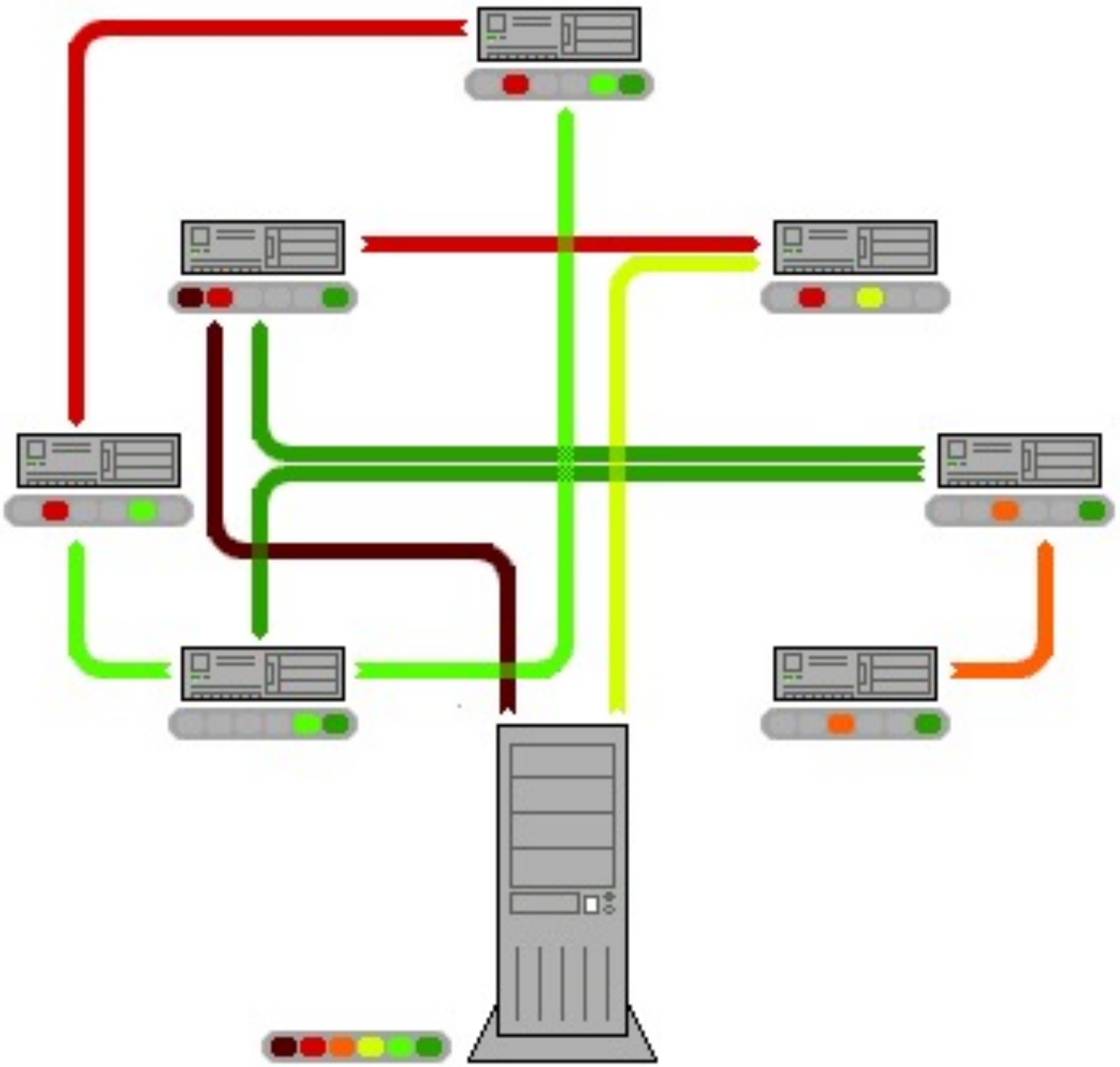
in 2010

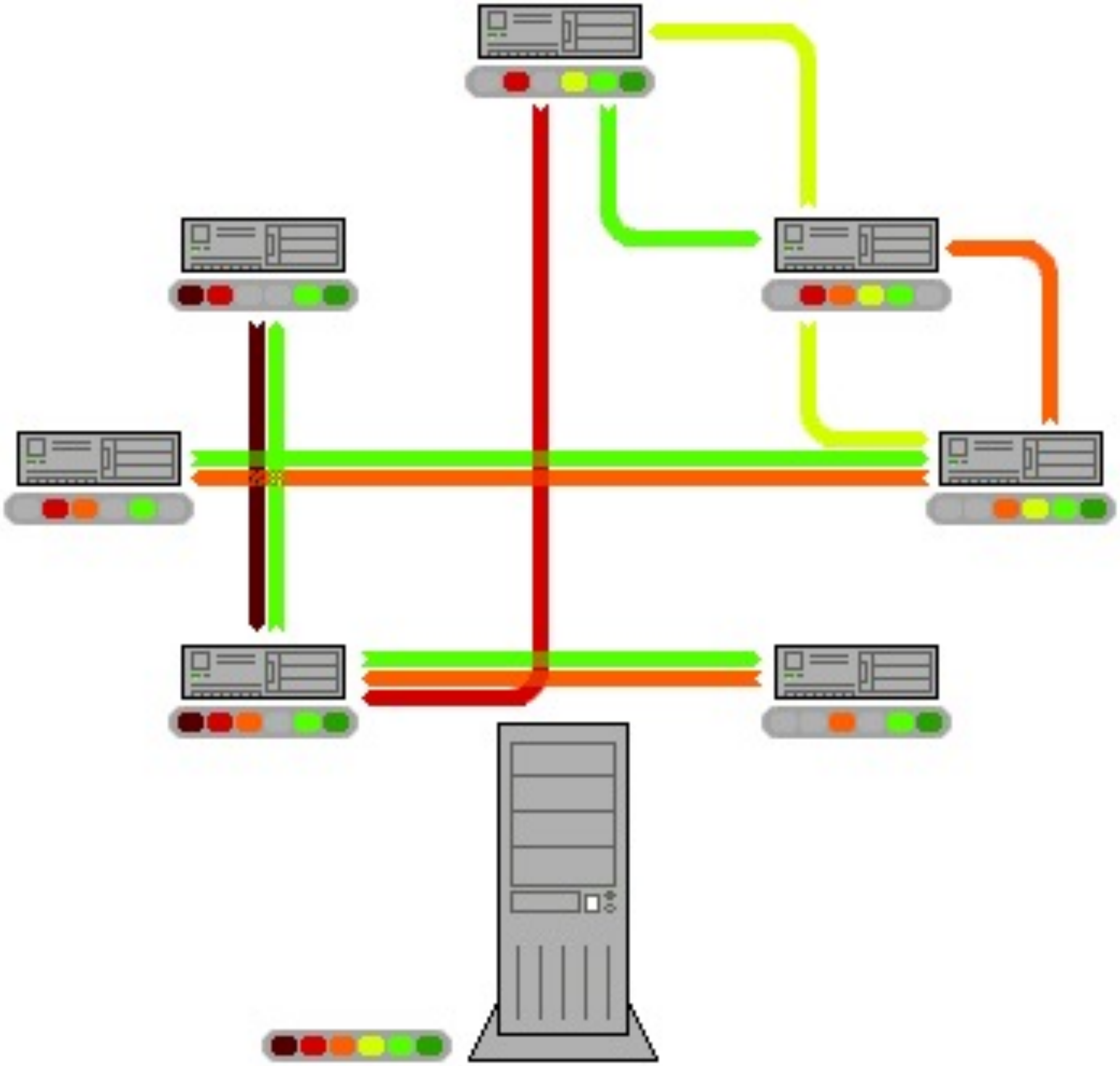


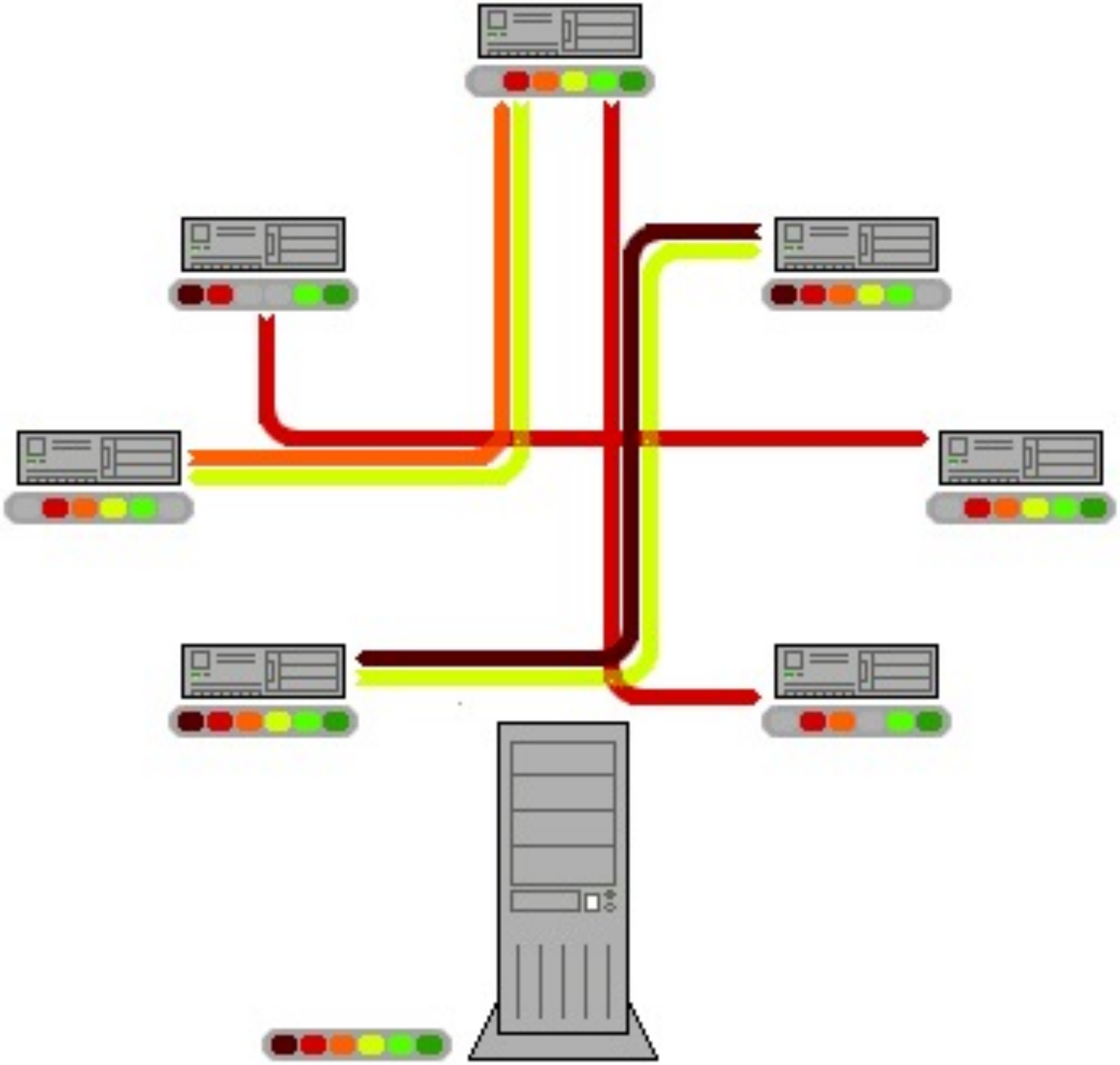


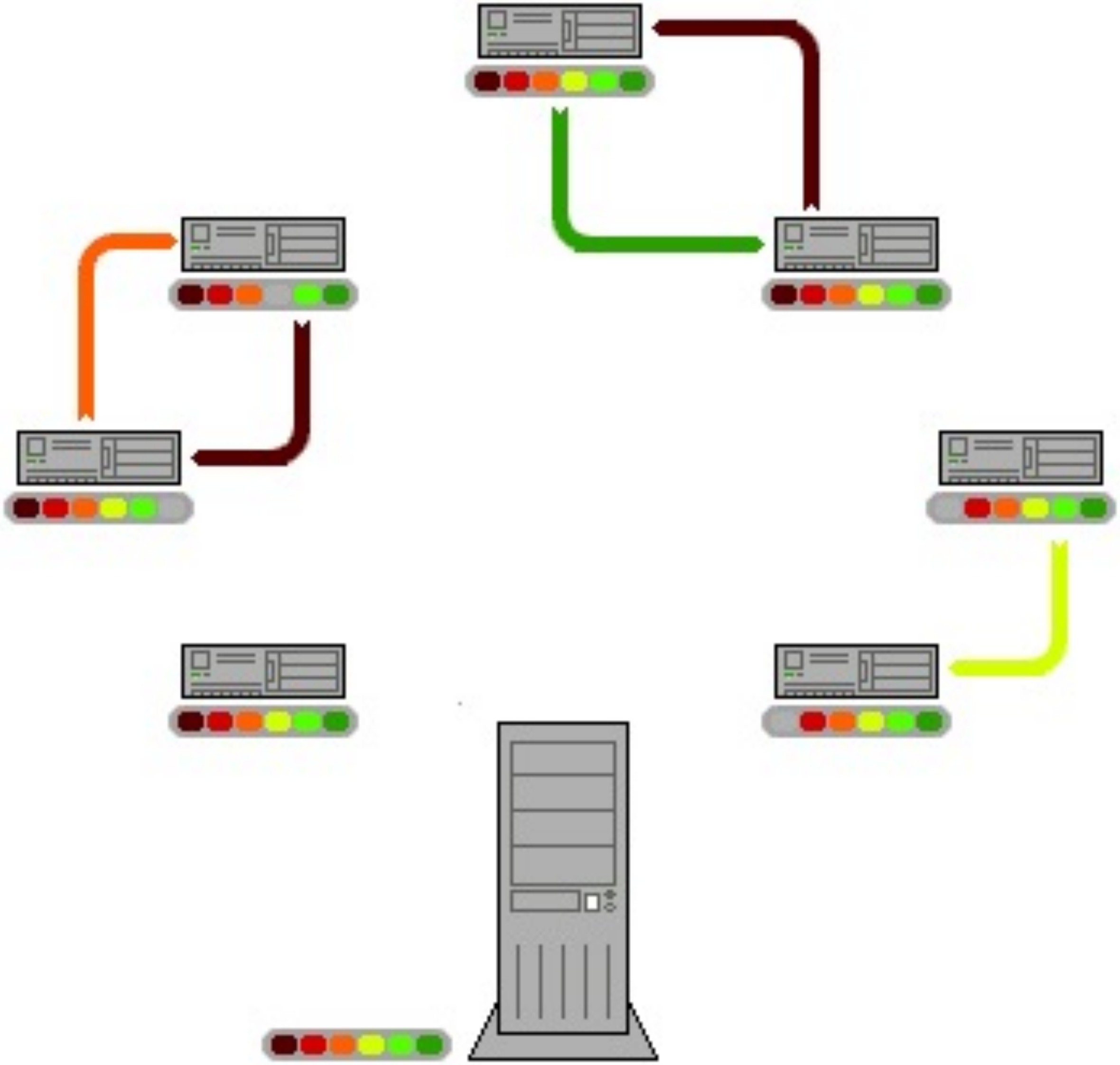


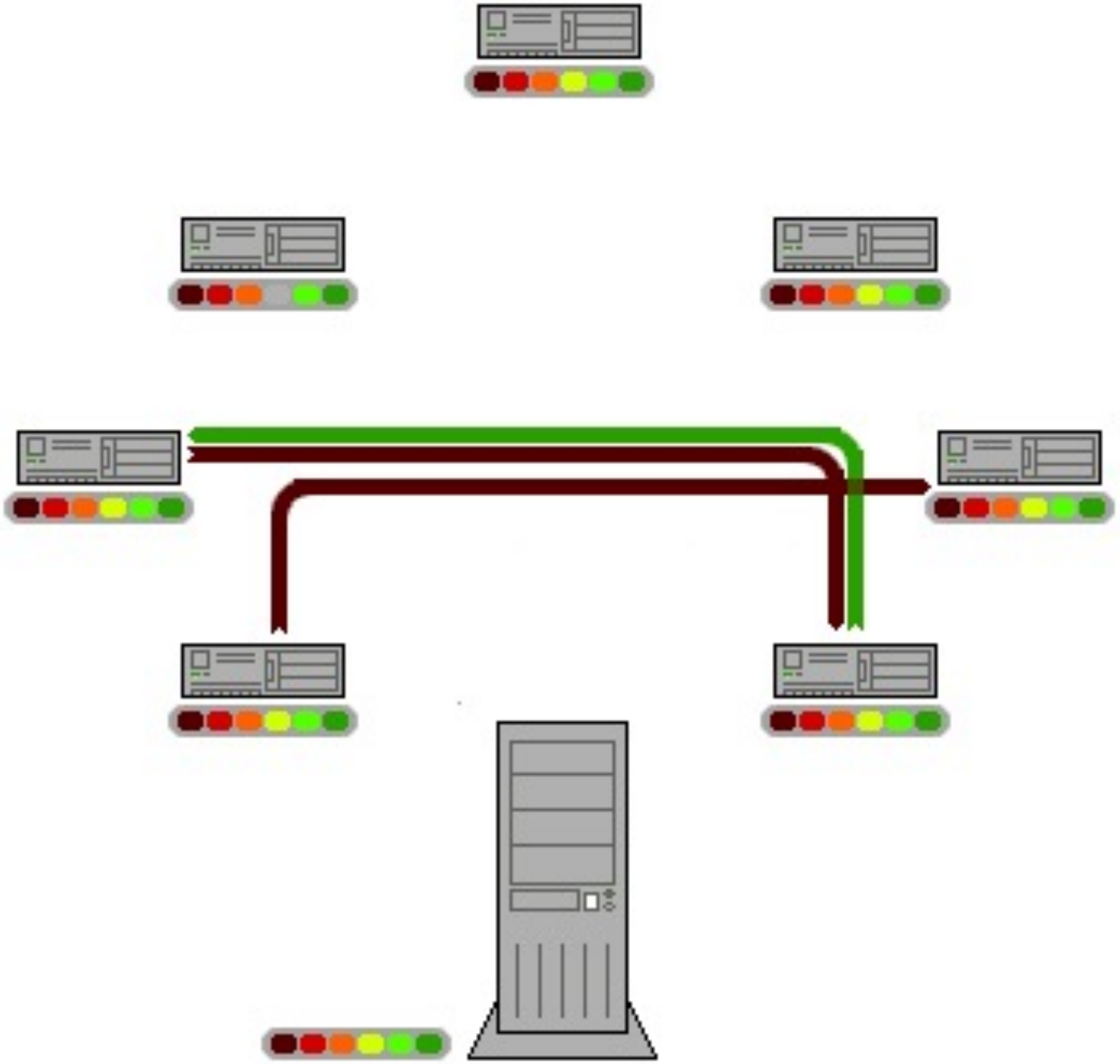


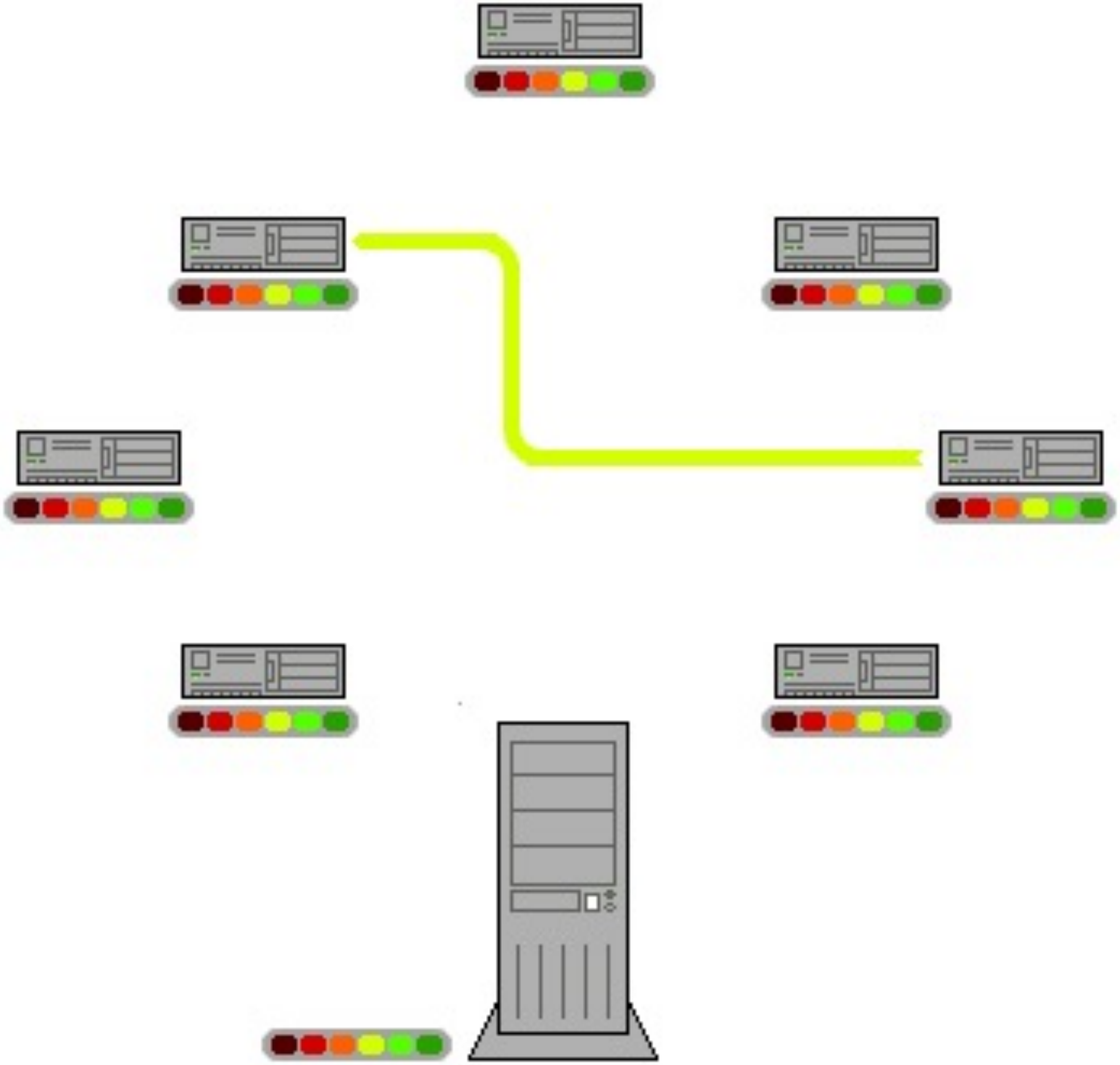


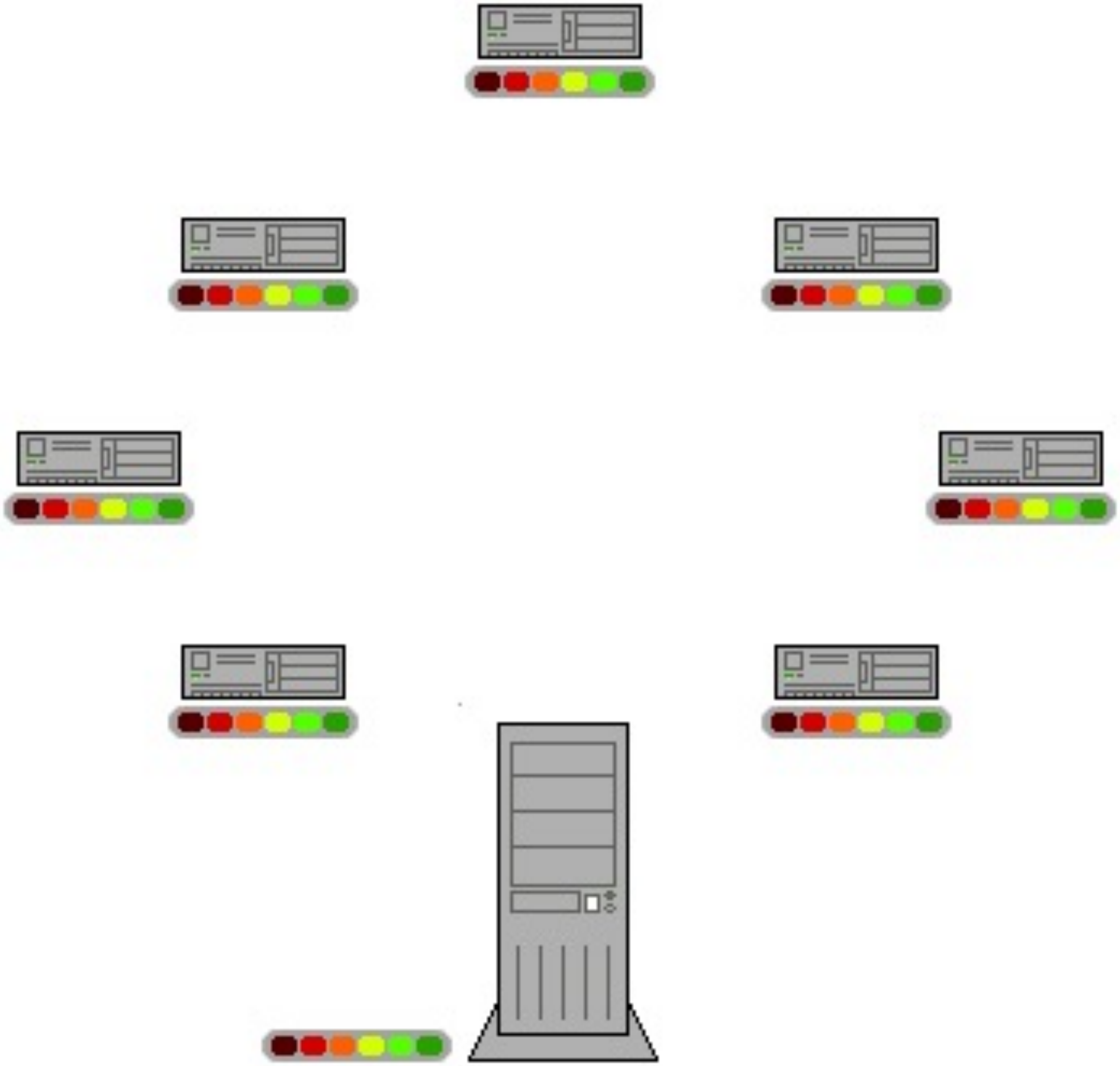












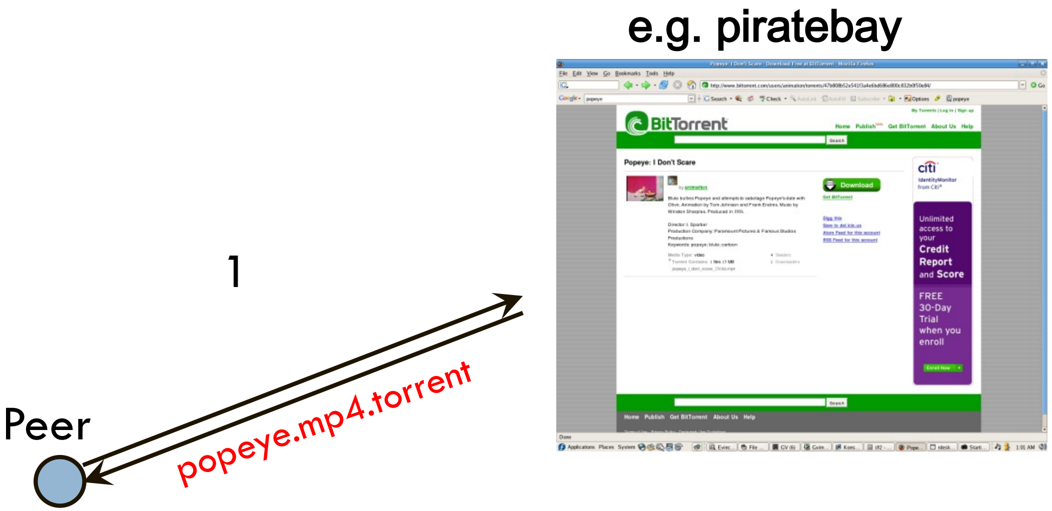
BitTorrent Swarm

- Swarm (Essaim/Enjambre/Sciame/埃賽姆)
  - Set of peers all downloading the same file
  - Organized in a random mesh topology
- Each node knows list of pieces downloaded by neighbors
- First intuition: a node requests pieces it does not own from neighbors

How a node enters a swarm for file "popeye.mp4.torrent"

- File popeye.mp4.torrent hosted at a (well-known) webserver
- The .torrent has the address of the TRACKER for the file
- The tracker, which runs on a webserver as well, keeps track of all peers downloading the file. It refresh the track list continuously (approx 30s)

How a node enters a swarm for file “madonnaCD.mp4”


e.g. piratebay
1
Peer
popeye.mp4.torrent
Popeye: I Don't Scare - Download Free at BitTorrent - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.bittorrent.com/users/animation/torrents/47b808b52e541f3a4e6bd686e800c832b0f50e84/
BitTorrent
Home Publish Get BitTorrent About Us Help
Search
Download
Get BitTorrent
Digg this
Save to del.icio.us
Atom Feed for this account
RSS Feed for this account
citi
IdentityMonitor from Citi®
Unlimited access to your Credit Report and Score
FREE 30-Day Trial when you enroll
Enroll Now
Done

Popeye: I Don't Scare

by animation

Bluto bullies Popeye and attempts to sabotage Popeye's date with Olive. Animation by Tom Johnson and Frank Endres. Music by Winston Sharples. Produced in 1936.

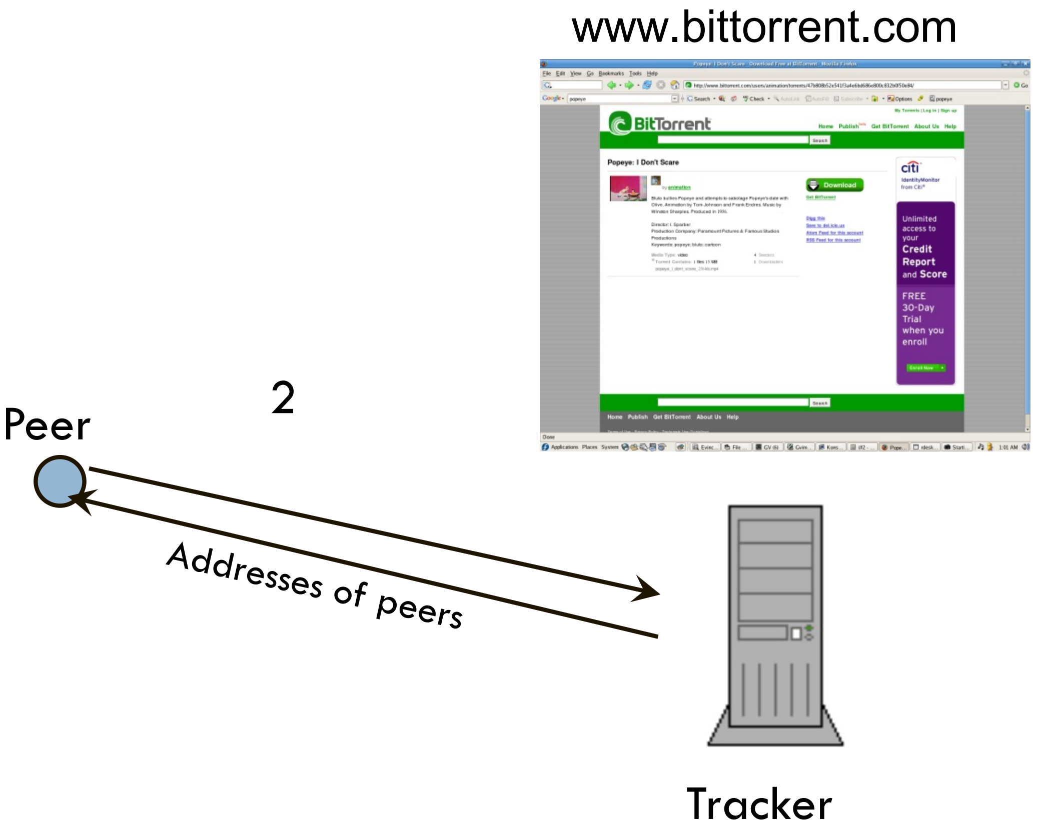
Director: I. Sparber
Production Company: Paramount Pictures & Famous Studios Productions
Keywords: popeye; bluto; cartoon

Media Type: video
Torrent Contains: 1 files 13 MB
popeye_i_dont_scare_256kb.mp4

4 Seeders
1 Downloaders

- □ File popeye.mp4.torrent hosted at a (well-known) webserver
- □ The .torrent has the address of the **TRACKER** for the file
- □ The tracker, which runs on a webserver as well, is a **SERVER** that keeps track of all peers downloading the file. It refresh the track list continuously (approx 30s)

How a node enters a swarm for file “popeye.mp4”



Popeye: I Don't Scare - Download Free at BitTorrent - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://www.bittorrent.com/users/animation/torrents/47b808b52e541f3a4e6bd686e800c832b0f50e84/

BitTorrent

My Torrents | Log in | Sign up

Home Publish Get BitTorrent About Us Help

Search

Popeye: I Don't Scare

by animation

Bluto bullies Popeye and attempts to sabotage Popeye's date with Olive. Animation by Tom Johnson and Frank Endres. Music by Winston Sharples. Produced in 1936.

Director: I. Sparber
Production Company: Paramount Pictures & Famous Studios Productions
Keywords: popeye; bluto; cartoon

Media Type: video
Torrent Contains: 1 files 13 MB
popeye_i_dont_scare_256kb.mp4

4 Seeders
1 Downloaders

Download
Get BitTorrent

Digg this
Save to del.icio.us
Atom Feed for this account
RSS Feed for this account

citi IdentityMonitor from Citi®

Unlimited access to your Credit Report and Score

FREE 30-Day Trial when you enroll

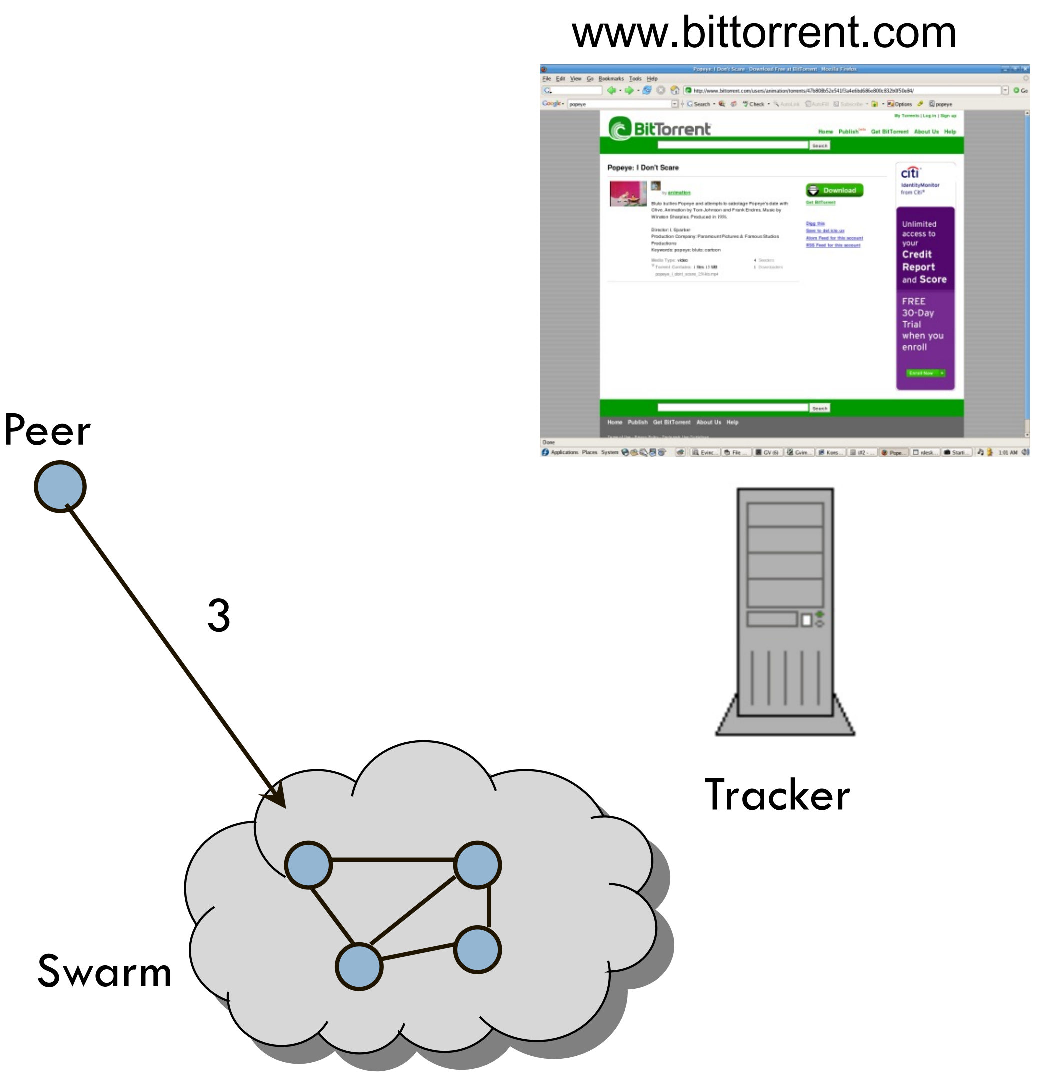
Enroll Now

Home Publish Get BitTorrent About Us Help

Done

- □ File popeye.mp4.torrent hosted at a (well-known) webserver
- □ The .torrent has the address of the TRACKER for the file
- □ The tracker, which runs on a webserver as well, keeps track of all peers downloading the file. It refresh the track peer list continuously (approx 30s)

How a node enters a swarm for file "popeye.mp4"



Popeye: I Don't Scare - Download Free at BitTorrent - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://www.bittorrent.com/users/animation/torrents/47b808b52e541f3a4e6bd686e800c832b0f50e84/

BitTorrent

My Torrents | Log in | Sign up

Home Publish Get BitTorrent About Us Help

Search

Popeye: I Don't Scare

by animation

Bluto bullies Popeye and attempts to sabotage Popeye's date with Olive. Animation by Tom Johnson and Frank Endres. Music by Winston Sharples. Produced in 1956.

Director: I. Sparber
Production Company: Paramount Pictures & Famous Studios Productions
Keywords: popeye; bluto; cartoon

Media Type: video
Torrent Contains: 1 files 15 MB
popeye_i_dont_scare_256kb.mp4

4 Seeders
1 Downloaders

Download
Get BitTorrent

Digg this
Save to del.icio.us
Atom Feed for this account
RSS Feed for this account

citi IdentityMonitor from Citi®

Unlimited access to your Credit Report and Score

FREE 30-Day Trial when you enroll

Enroll Now

Search

Home Publish Get BitTorrent About Us Help

Done

Applications Places System

1:01 AM

- □ File popeye.mp4.torrent hosted at a (well-known) webserver
- □ The .torrent has address of tracker for file
- □ The tracker, which runs on a webserver as well, keeps track of all peers downloading the file. It refresh the track list continuously (approx 30s)

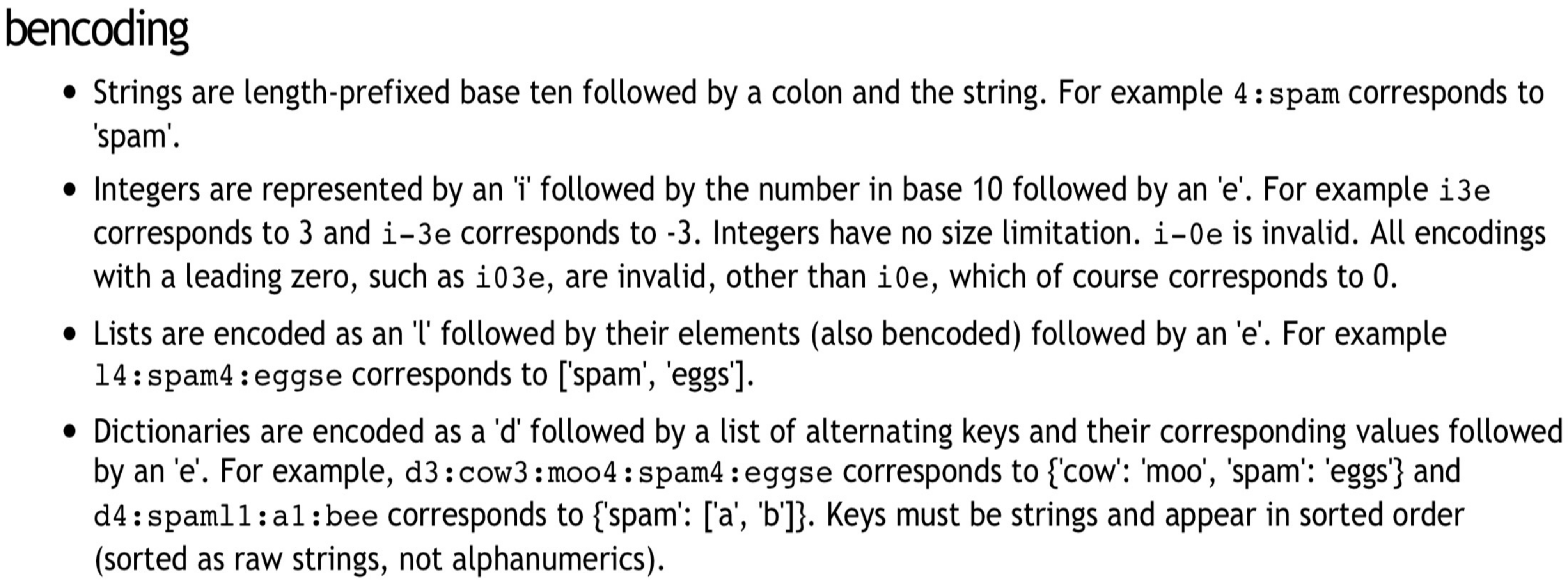
Contents of .torrent file

- HTTP/URL/UDP of tracker
- Piece length – Usually 256 KB up to 2 MB
- SHA-1 hashes of each piece in file
  - For reliability/data integrity
- Files names and file hierarchy – allows download of multiple files, e.g. a directory containing files

Inside a .torrent file

- The file is encoded using an original *B-encoding*
  - Announce URL/HTTP/UDP of the tracker
  - Some optional fields
    - Creation date, comment, created by
- Info key
  - Length on the content in bytes
  - md5 hash of the content
  - Pieces SHA-1 hash are enough
  - File Name
  - Piece length (256kB, 512kB, 1024kB, etc.)
  - Concatenation of all pieces SHA-1 hash

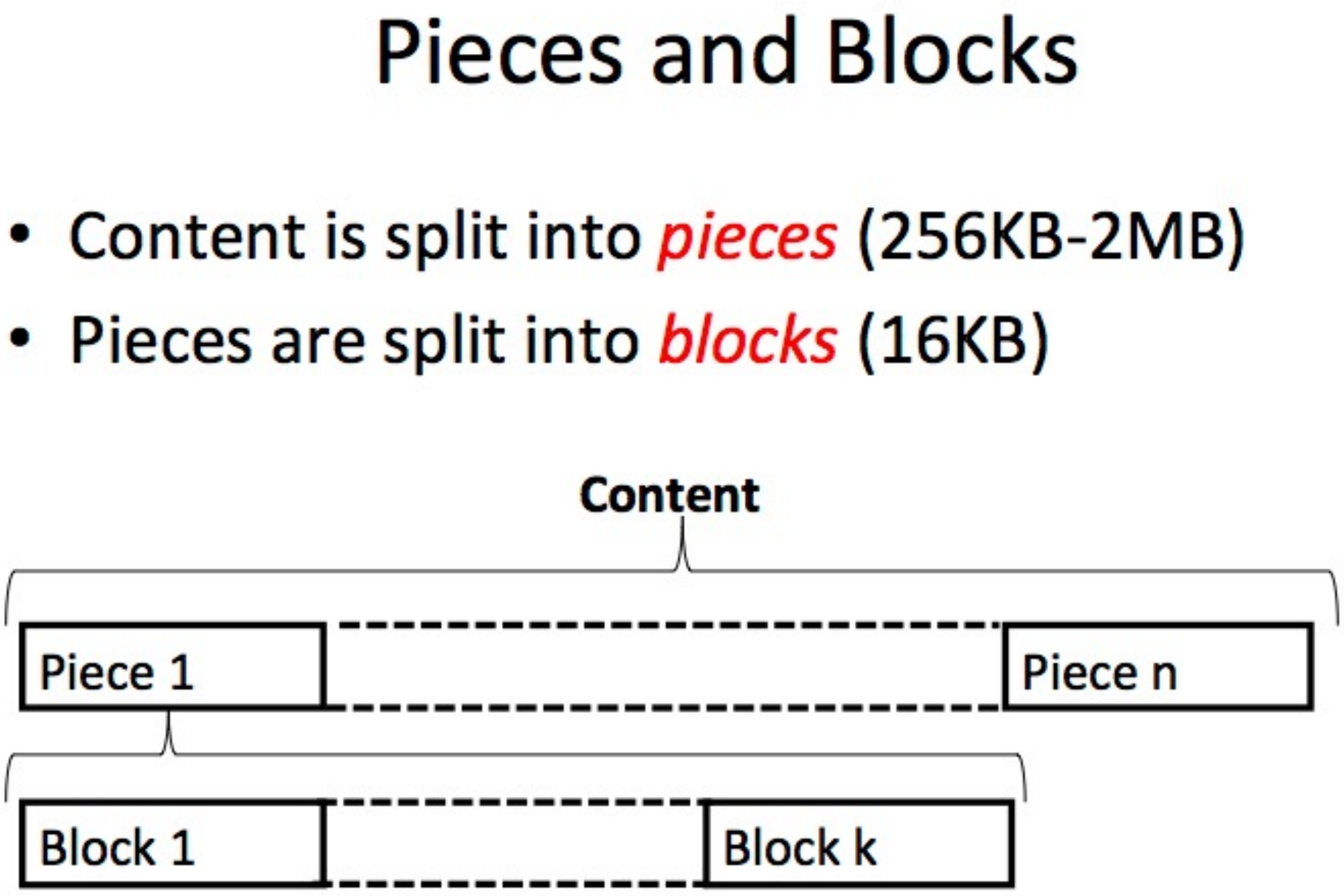
BENCODING (in a torrent file)



bencoding

- Strings are length-prefixed base ten followed by a colon and the string. For example `4:spam` corresponds to 'spam'.
- Integers are represented by an 'i' followed by the number in base 10 followed by an 'e'. For example `i3e` corresponds to 3 and `i-3e` corresponds to -3. Integers have no size limitation. `i-0e` is invalid. All encodings with a leading zero, such as `i03e`, are invalid, other than `i0e`, which of course corresponds to 0.
- Lists are encoded as an 'l' followed by their elements (also bencoded) followed by an 'e'. For example `l4:spam4:eggse` corresponds to ['spam', 'eggs'].
- Dictionaries are encoded as a 'd' followed by a list of alternating keys and their corresponding values followed by an 'e'. For example, `d3:cow3:moo4:spam4:eggse` corresponds to {'cow': 'moo', 'spam': 'eggs'} and `d4:spaml1:a1:bee` corresponds to {'spam': ['a', 'b']}. Keys must be strings and appear in sorted order (sorted as raw strings, not alphanumerics).

Pieces and Blocks



Pieces and Blocks

- Content is split into *pieces* (256KB-2MB)
- Pieces are split into *blocks* (16KB)

Terminology

- Seed: peer with the entire file
  - Original Seed: the first seed
- Leech: peer that's downloading the file
  - Fairer term might have been "downloader"
- Block: Further subdivision of a piece
  - The "unit for requests" is a block (16KB)
  - A peer uploads only after assembling complete piece

Peer-peer transactions: Choosing pieces to request 1

- Rarest-first: Look at all pieces at all peers, and request piece that’s owned by fewest peers
  - Increases diversity in the pieces downloaded
    - avoids case where a node and each of its peers have exactly the same pieces; increases throughput
  - Increases likelihood all pieces still available even if original seed leaves before any one node has downloaded entire file

Choosing pieces to request 2

- Random First Piece:
  - When peer starts to download, request random piece.
    - So as to assemble first complete piece quickly
    - Then participate in uploads
  - When first complete piece assembled, switch to rarest-first

Choosing pieces to request 3

- End-game mode:
  - When requests sent for all blocks, (re)send requests to all peers
  - To speed up completion of download
  - ``Cancel`` request for downloaded blocks

Tit-for-tat as incentive to upload

- Want to encourage all peers to contribute
- Peer *A* said to choke peer *B* if it (*A*) decides not to upload to *B*
- Each peer unchokes at most 4 *interested* peers at any time
  - The three with the largest upload rates to *A*
    - Where the tit-for-tat comes in
  - Another randomly chosen (Optimistic Unchoke)
    - To periodically look for better choices

Tit-for-tat

- □ Best deterministic strategy for the ``Prisoner’s Dilemma``
- □ *1*- Unless provoked, the agent will always cooperate
- □ 2 - If provoked, the agent will retaliate
- □ 3 - The agent is quick to forgive
- □ The agent must have a good chance of competing against the opponent more than once

Anti-snubbing

- A peer is said to be snubbed if each of its peers chokes it
- To handle this, snubbed peer stops uploading to its peers
- Optimistic unchoking done more often
  - Hope is that will discover a new peer that will upload to us

Why BitTorrent took off 1

- Better performance through “pull-based” transfer
  - Slow nodes don’t bog down other nodes
- Allows uploading from hosts that have downloaded parts of a file
  - In common with other end-host based multicast schemes

Why BitTorrent took off 2

- Practical Reasons (perhaps more important!)
  - Working implementation in Python by Bram Cohen with simple well-defined interfaces for plugging in new content
  - Many recent competitors got sued / shut down
    - Napster, Kazaa, EMule, …
  - Doesn't do "search" per se. Users use well-known, trusted sources to locate content
    - Avoids the "*pollution problem*", where garbage is passed off as authentic content

Pros and cons of BitTorrent 1

- Pros
  - Proficient in utilizing partially downloaded files
  - Discourages "freeloading"/ "free riders"
    - By rewarding fastest uploaders
  - Encourages diversity through "rarest-first"
    - Extends lifetime of swarm
- Works well for very popular contents

Pros and cons of BitTorrent 2

- Cons
  - Assumes all interested peers <u>active</u> at same time; performance deteriorates if swarm “cools off”

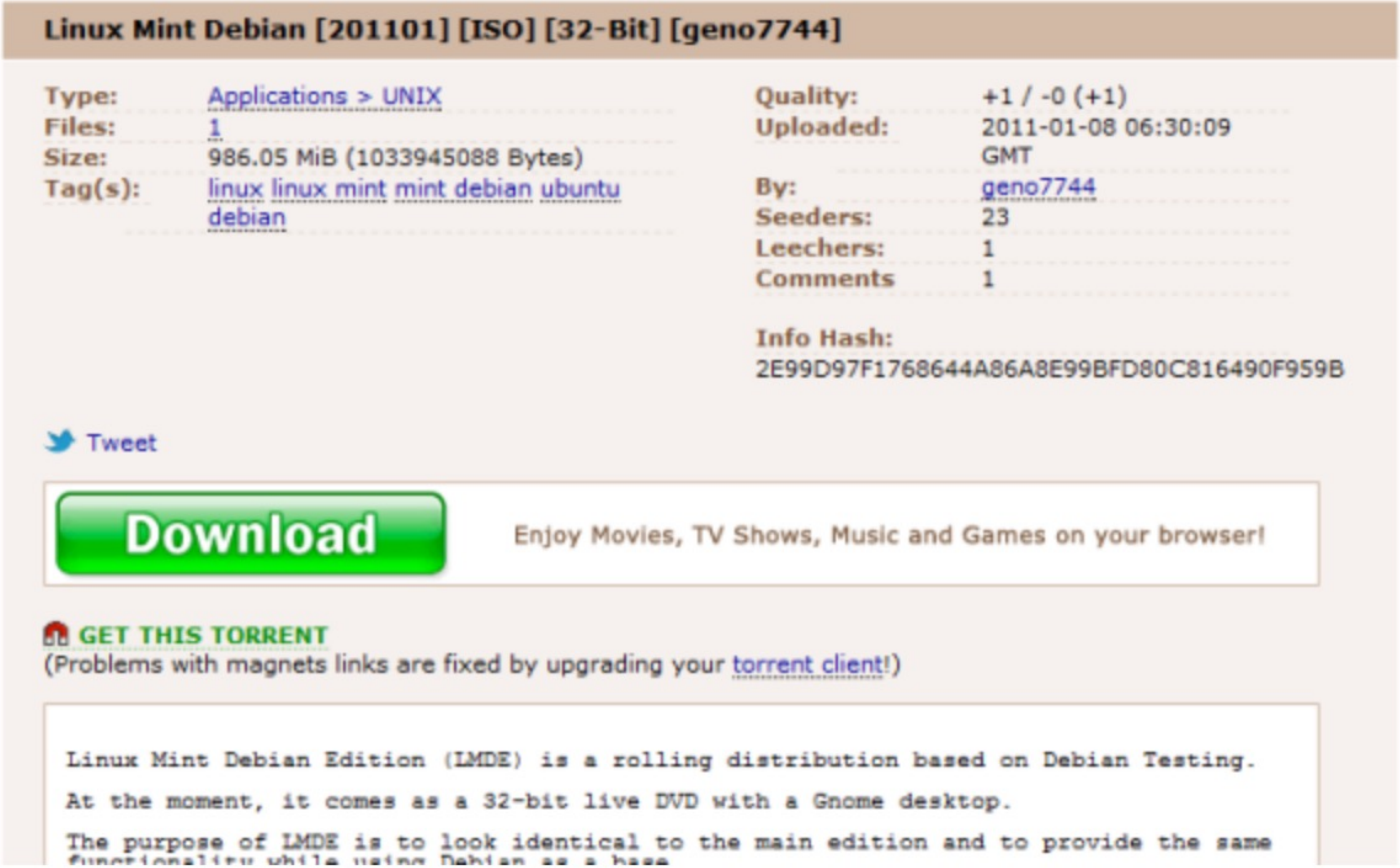
Pros and cons of BitTorrent 3

- Dependence on centralized tracker: pro/cons ?
  - ☹ Single point of failure: new nodes can’t enter swarm if tracker goes down
  - Lack of a search feature
    - ☺ Prevents pollution attacks
    - ☹ Users need to resort to out-of-band search: well known torrent-hosting sites / plain classic web-search

“Trackerless” BitTorrent

- To be more precise, “BitTorrent without a centralized-tracker”
- E.g.: Vuze
- Uses a Distributed Hash Table (Kademlia DHT) as a “distributed tracker”
- Tracker run by a normal end-host (not a web-server anymore)
  - The original seeder could itself be the tracker
  - or have a node in the DHT randomly picked to act as the tracker

Magnet links



Linux Mint Debian [201101] [ISO] [32-Bit] [geno7744]

Type:	Applications > UNIX
Files:	1
Size:	986.05 MiB (1033945088 Bytes)
Tag(s):	linux linux mint mint debian ubuntu debian

Quality:	+1 / -0 (+1)
Uploaded:	2011-01-08 06:30:09 GMT
By:	geno7744
Seeders:	23
Leechers:	1
Comments	1

Info Hash:
2E99D97F1768644A86A8E99BFD80C816490F959B

GET THIS TORRENT
(Problems with magnets links are fixed by upgrading your torrent client!)

```
Linux Mint Debian Edition (LMDE) is a rolling distribution based on Debian Testing.
At the moment, it comes as a 32-bit live DVD with a Gnome desktop.
The purpose of LMDE is to look identical to the main edition and to provide the same
functionality while using Debian as a base
```

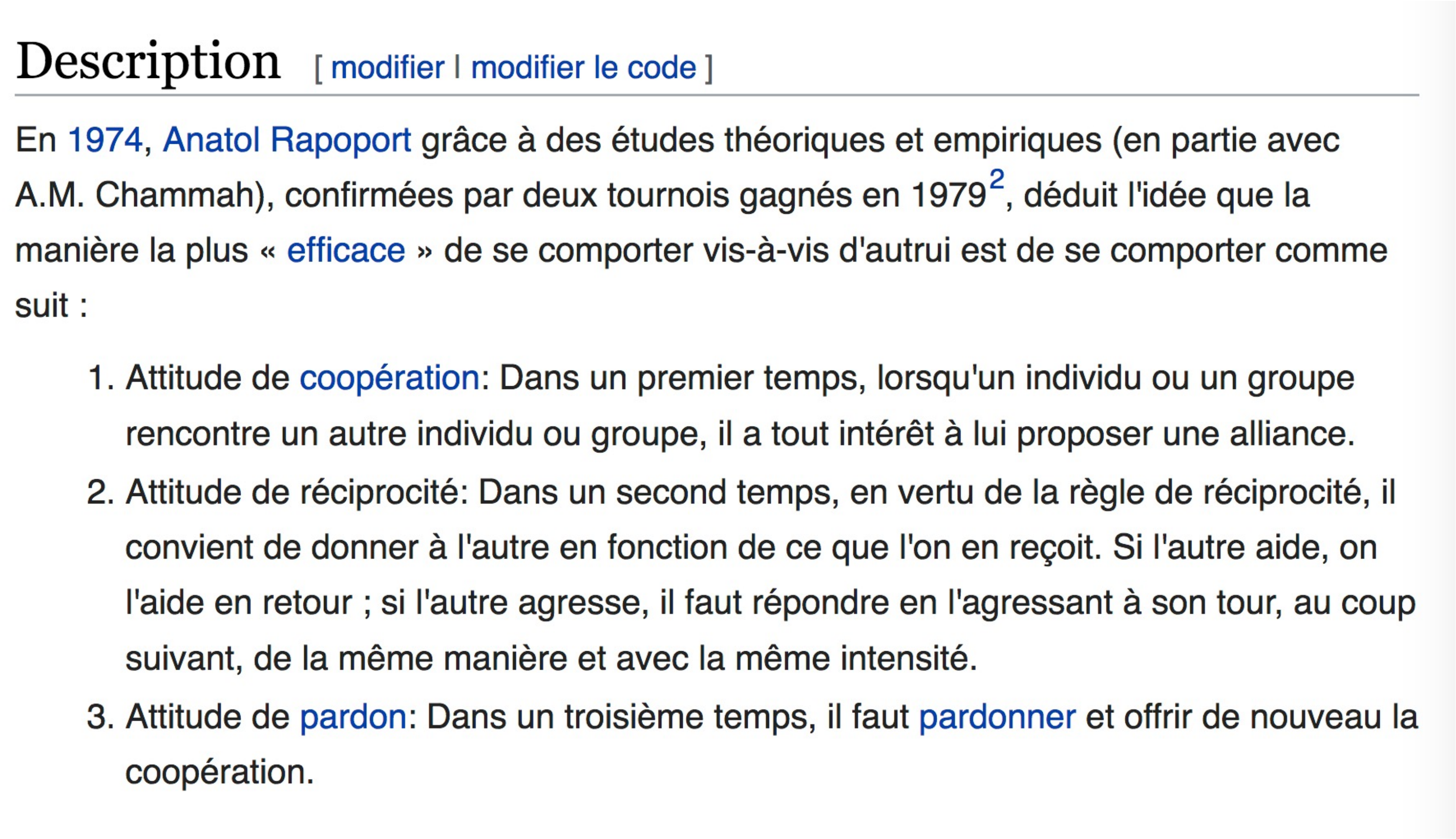
Why is (studying) BitTorrent important?

- BitTorrent consumes significant amount of internet traffic today
  - In 2004, BitTorrent accounted for 30% of all internet traffic (Total P2P was 60%), according to CacheLogic
  - Slightly lower share in 2005 (possibly because of legal action), but still significant
  - BT always used for legal software (linux iso) distribution too
  - Recently: legal media downloads (Fox, BBC), streaming

Implementation (FR)

- Les implémentations les plus connues souhaitent échanger des pieces de taille jusqu'à 2Mb ;
- Elle préfèrent utiliser plutôt protocoles plus ``élastiques`` comme UDP ou HTTP : grâce à cela, elle fractionnent ultérieurement la pièce en sous-pieces, ces derniers appelés BLOCKS de taille fixe de 16 KB.
- Il y a un consensus quasi complet dans la communauté des clients bittorent sur la sous-fragmentation d'une PIECE de taille VARIABLE entre 256Kb et 2Gb en BLOCKS de taille FIXE de 16kb.

Tit-for-tat



Description [modifier | modifier le code]

En 1974, Anatol Rapoport grâce à des études théoriques et empiriques (en partie avec A.M. Chammah), confirmées par deux tournois gagnés en 1979[2], déduit l'idée que la manière la plus « efficace » de se comporter vis-à-vis d'autrui est de se comporter comme suit :

1. Attitude de coopération: Dans un premier temps, lorsqu'un individu ou un groupe rencontre un autre individu ou groupe, il a tout intérêt à lui proposer une alliance.
2. Attitude de réciprocité: Dans un second temps, en vertu de la règle de réciprocité, il convient de donner à l'autre en fonction de ce que l'on en reçoit. Si l'autre aide, on l'aide en retour ; si l'autre agresse, il faut répondre en l'agressant à son tour, au coup suivant, de la même manière et avec la même intensité.
3. Attitude de pardon: Dans un troisième temps, il faut pardonner et offrir de nouveau la coopération.