Lecture 8

# Overlay Applications and Mechanisms

Universität
Zürich UZH

Communication Systems Group

# 0. Lecture Overview

- **Hot Topics / P2P in the News / Interesting Developments**
- **BitTorrent (Application)**
- **Incentives (Mechanism)**
  - Categorization
  - TFT, transitive TFT
- **Rsync (Mechanism)**
  - Introduction
  - Mechanism, Example
- **Network Coding (Mechanism)**

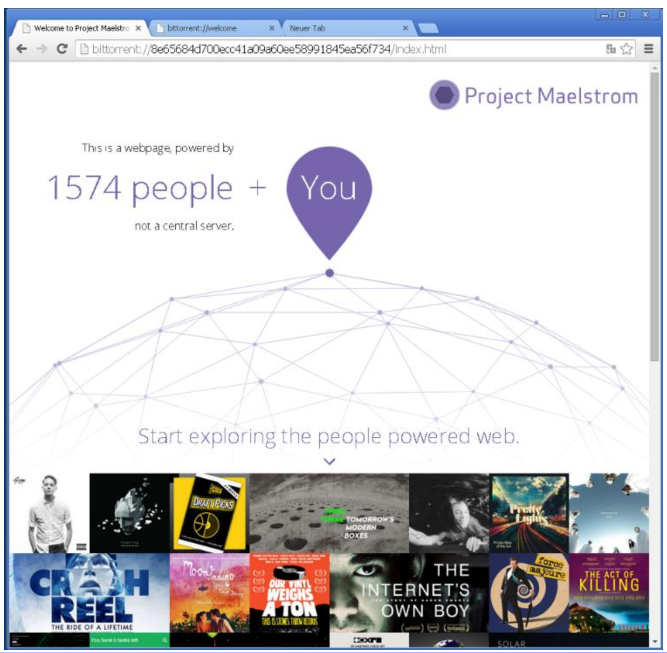
# Hot Topics / P2P in the News / Interesting Developments

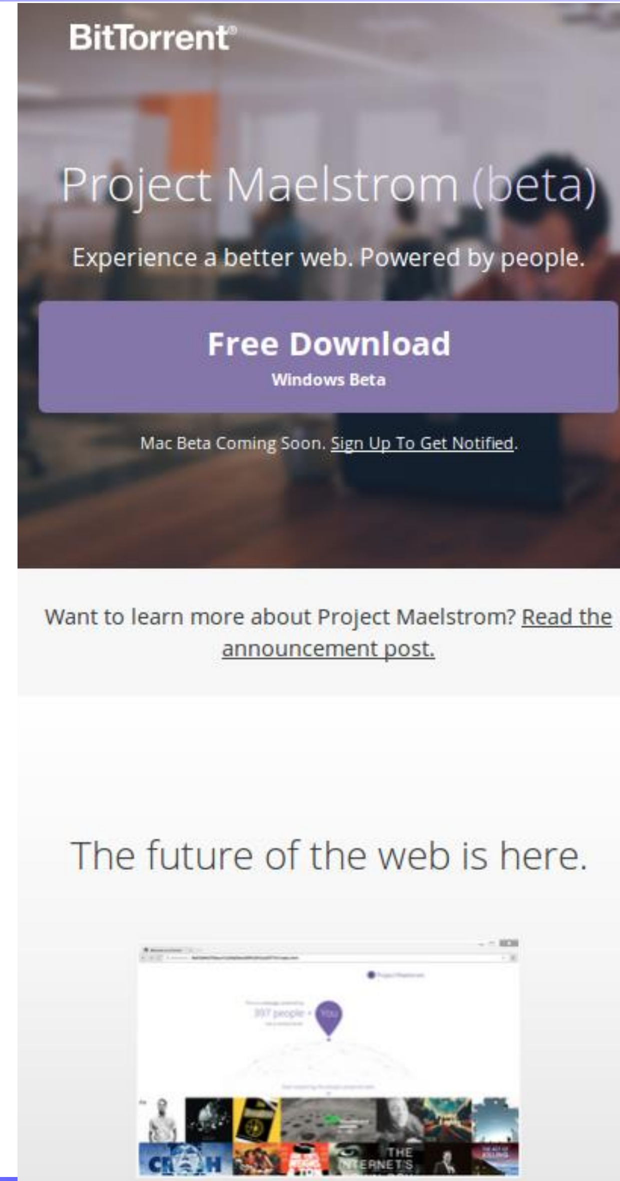
- **(10.03.2015) There is No Now**
  - Problems with simultaneity in distributed systems
  - NTP – over Internet accuracy ~5 - 100ms , LAN ~1ms
  - GPS – Googles Spanner – worst-case clock drift ~1 to 7ms
  - vDHT → not time sensitive
- **(6.2.2015) Eight Fallacies of Distributed Computing**

1. The Network is Reliable
2. Latency is Zero
3. Bandwidth is Infinite
4. The Network is Secure
5. Topology Doesn't Change
6. There is One Administrator
7. Transport Cost is Zero
8. The Network is Homogeneous

# Hot Topics / P2P in the News / Interesting Developments

- (10.4.2015) [Project Maelstrom Enters Beta]
  - Web via BitTorrent, Windows only
  - Run on closed software, run by a single corporation ☹



# 1. BitTorrent

**Introduction, Mesh-based File Sharing, Mechanisms, Scenario, Phases, Evaluation**

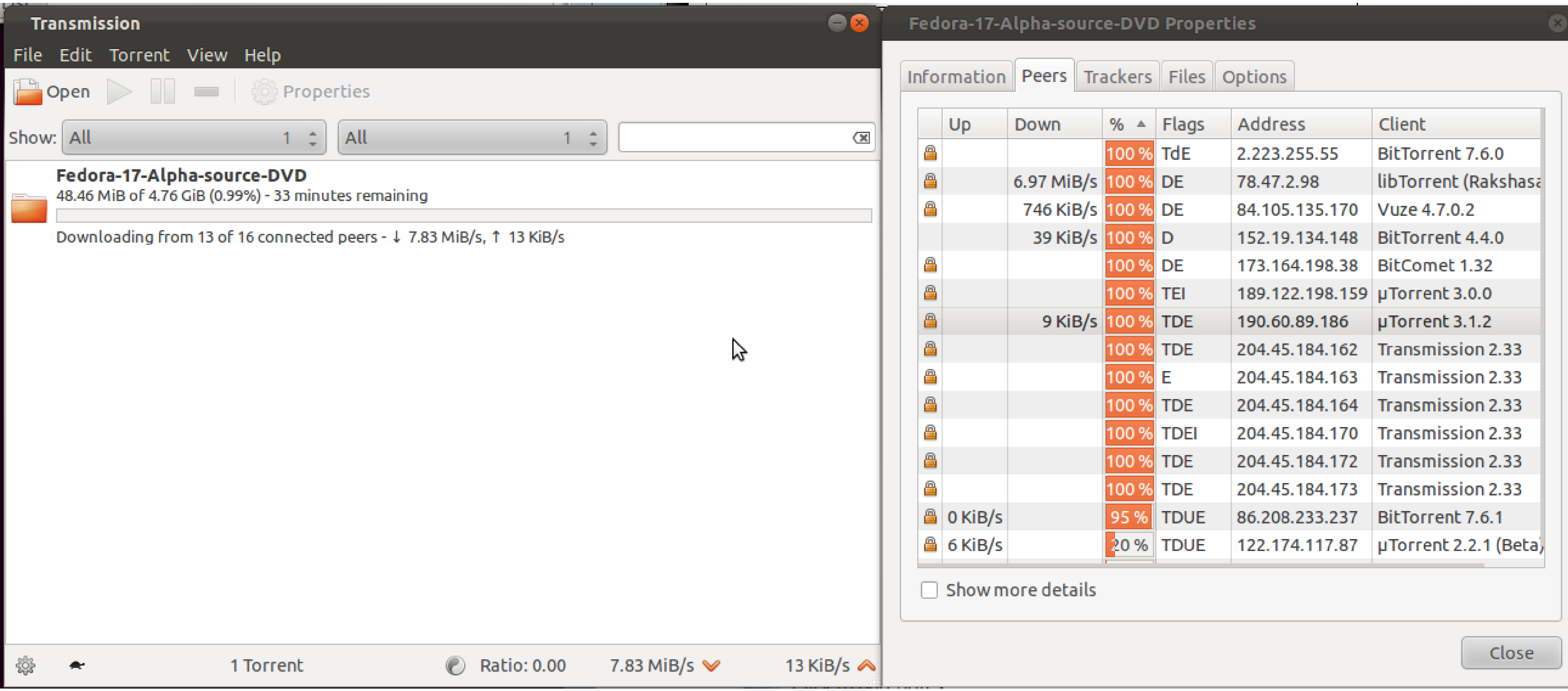
# Introduction

- **BitTorrent**
  - Peer to peer file-sharing system
  - Used to transfer large files
  - Defines a protocol, many clients (Wikipedia):

    ABC, Acquisition, BitComet, BitLet, BitLord, BitThief, BitTornado, BitTorrent 5 / Mainline, BitTorrent 7, Bits on Wheels, BitTyrant, Blog Torrent, Deluge, FlashGet, Free Download Manager, Gnome BitTorrent, Kget, Ktorrent, LimeWire, Meerkat Bittorrent Client, Miro, MLDonkey, MP3 Rocket, µTorrent, OneSwarm, Opera, qBittorent, rTorrent, Shareaza, SymTorrent, Tomato Torrent, Tonido Torrent, Torrent Swapper, TorrentFlux, Transmission, Tribler, Vuze (formerly Azureus), Wyzo, ZipTorrent

# Introduction

- E.g. Transmission




- uTP, UPNP/NAT-PMP, PEX, DHT, Magnet, ([Flags](#))

# Mesh-based File Sharing – BitTorrent

- **Involved peers / roles:**

- **Tracker**
  - Non-content-sharing node
  - Actively tracks
    - Swarm: all peers (seeders and leeches), status of downloaded data - volume
  - Centralized / Decentralized (HTTP/DHT)
- **Seeders**
  - Have complete copies and is uploading to other peers
- **Leechers**
  - Incomplete copies of the desired content
  - Leeches try to download missing pieces
- **Swarm**
  - Sum of all leechers and seeders, participating in a particular torrent
  - One swarm per torrent

# Mesh-based File Sharing – BitTorrent

- **Involved peers / roles:**

- **.torrent provider**
  - File contains: name, size, date of files, number of pieces and SHA1 hash of all pieces, trackers for the torrent
  - File is split into chunks, typically 256kB (262144)
  - Bencode:

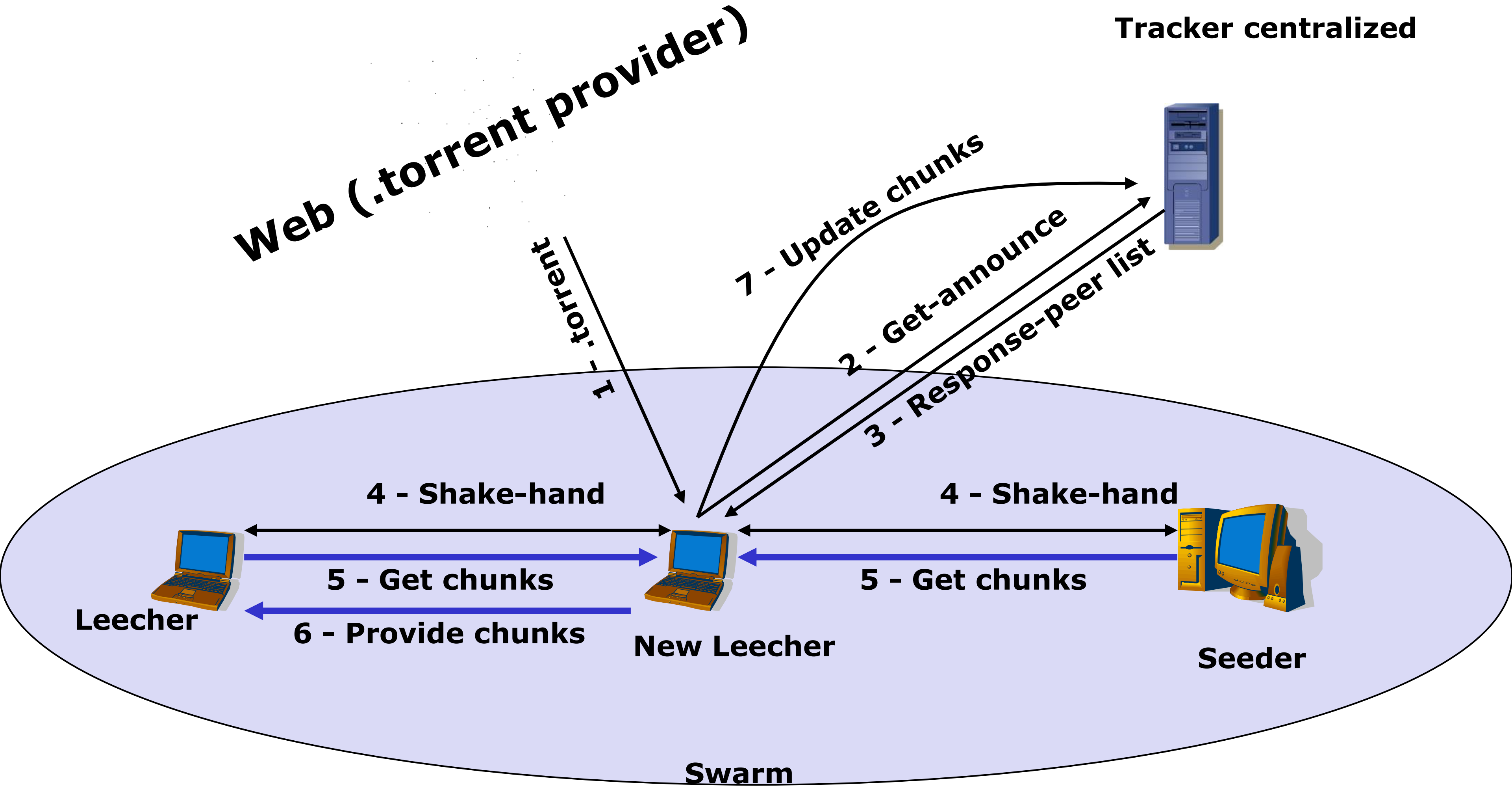
```
i<number>e -> i100e
<length>:<contents> -> 4:test
l<contents>e -> l4:testi100ee
  (two values)
d<contents>e -> d4:testi100ee
  (key test, value 100)
```


```
4 70 3A 2F 2F  d8:announce46:http://
F 6A 65 63 74  torrent.fedoraproject
3 65 31 33 3A  .org:6969/announce13:
3 30 34 31 35  creation datei1330415
5 73 6C 64 36  875e4:infod5:filesld6
1 74 68 6C 33  :lengthi1036e4:pathl3
1 2D 73 6F 75  1:Fedora-17-Alpha-sou
A 6C 65 6E 67  rce-CHECKSUMeed6:leng
0 61 74 68 6C  thi5114945536e4:pathl
8 61 2D 73 6F  30:Fedora-17-Alpha-so
A 6E 61 6D 65  urce-DVD.isoeee4:name
8 61 2D 73 6F  26:Fedora-17-Alpha-so
0 6C 65 6E 67  urce-DVD12:piece leng
5 73 33 39 30  thi262144e6:pieces390
A 37 D1 32 DC  240:.Q..c.....X..7.2.
F B6 29 11 11  ......9...H..[.Lo.)..
B 09 FC CA 89  ....e.( ......j......
7 92 9A 22 B8  ..!.."....(.*.T....".
F 7B 6C 41 F4  ...S.7v.g<...c...{lA.
3 EC C0 2C DC  .(K.]y..g..._...S..,.
E 9A CA BE 62  .c.......#...hM.~...b
D 9C 2B 99 C6  M3..S.[.>i..!1W...+..
1 F2 E5 83 D3  ,.p."....d.....C.....
4 30 D6 B0 31  .....ke.......k..0..1
B 1E 2E 05 4B  2........a.0........K
5 5A D4 C9 58  ./..B.Eg ..!..m.5Z..X
7 C7 66 F8 CF  .C.o....B..'.''v..f..
```

# BitTorrent Phases

- **1. Get torrent from Web**
  - Several websites offer torrents
- **2. Get announce, contact tracker**
  - Tracker maintains a list of active peers sharing the file
- **3. Response peer list**
  - Peer list contains up to 50 peers (seeders and leechers)
- **4. Shake-hand**
  - Peer establishes connection to 20-40 peers from the peer-list
  - Uses just ~4 for exchanging chunks
- **5. Get chunk**
  - Peers request chunks using the (local) rarest first policy
  - Parallel downloads from multiple peers
- **6. Provide chunk**
  - Tit-for-tat strategy: Peers give as long as they receive
  - Slow startup phase, as nothing to upload
- **7. Re-announce at tracker (new contacts, keep alive, progress)**

# BitTorrent: A Scenario




11

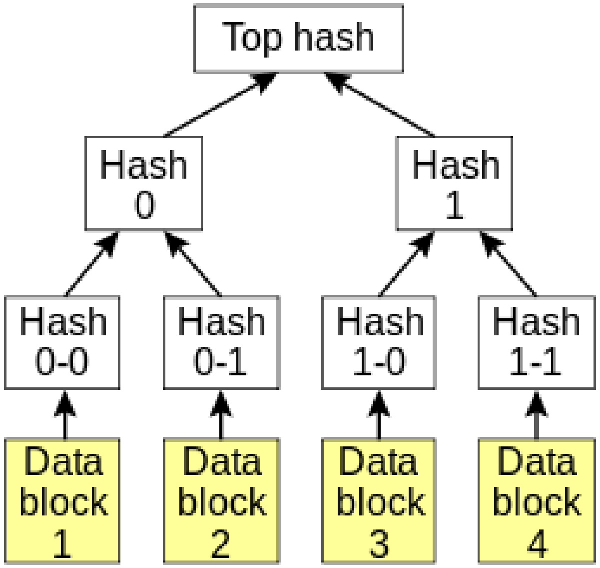
# BitTorrent: Mechanisms

- **Classification**
- Originally: centralized (tracker-based) P2P
  - Decentralized chunk exchange
- But: decentralized tracker-replacements exist
  - DHT-based tracker
    - The Pirate Bay anounced 2012 to use DHT trackers
    - Store tracker information on ~20 peers (Vuze)
  - Gossiping protocol (PEX = Peer Exchange Protocol)
    - Distribute active peers every minute (push-based)
  - B-Tracker
    - Every leecher and seeder becomes a tracker
- Also combined usage possible

# BitTorrent: Mechanisms

- **Exchange strategy: tit-for-tat**
  - If I give you – you give me
    - Symmetric interest in a swarm
    - Encourages peers to contribute
      (early Gnutella showed that majority does not share anything)
  - Choked: “not sending right now” - connection not being used to transmit
  - Unchoking: best download rates, evaluated every 10sec
  - Optimistic unchoking: rotate every 30sec
  - BitThief!
- **Exchange strategy: rarest-first or random**
  - Good distribution of pieces - in case seeders go offline

- **Magnet links**
  - Magnet is URI scheme, does not point to a centralized tracker
    - No centralized tracker: pointer to DHT
    - General purpose, not only for BT
    - `magnet:?xl=1000&dn=song1.mp3&xt=urn:tree:tiger:2A3B…`
- `tree:tiger` → Hash Tree
    - Tree of hashes (|| → concatenation)
    - `hash 0 = hash( hash 0-0 || hash 0-1 )`
    - `hash 1 = hash( hash 1-0 || hash 1-1 )`
    - `Top hash = hash( hash 0 || hash 1 )`
- Merkle hash / hash tree also seen in Bitcoin blocks (transactions)




http://en.wikipedia.org/wiki/Hash_tree

# BitTorrent: Mechanisms

- Verification
  - Peer A has top hash (root hash)
  - Peer downloads C4 from peer B
    - create hash 8
  - Need hash 10, 13, 3 (uncle hash)
    - Can be from peer B
  - With 8,10,13,3 can create root hash

→ verify this root hash

```
                 7 = root hash
                / \
              /     \
            /         \
          /             \
        3*                11
       / \               / \
      /   \             /   \
     /     \           /     \
    1       5         9        13* = uncle hash
   / \     / \       / \      / \
  0   2   4   6     8   10* 12   14

  C0  C1  C2  C3    C4  C5  C6   E
  =chunk index      ^^           = empty hash
```

The Merkle hash tree of an interval of width W=8

http://datatracker.ietf.org/doc/draft-ietf-ppsp-peer-protocol/ Section 5.2

# BitTorrent: Mechanisms

- **uTP**
  - Micro Transport Protocol - application-layer congestion-control protocol using UDP at the transport-layer
  - Congestion (queue) detected by time rather than packet loss (TCP)
  - TCP friendly
  - Future work: delay-based congestin in TomP2P / UDT (experiments with [uTP in Java](#) / TomP2P)

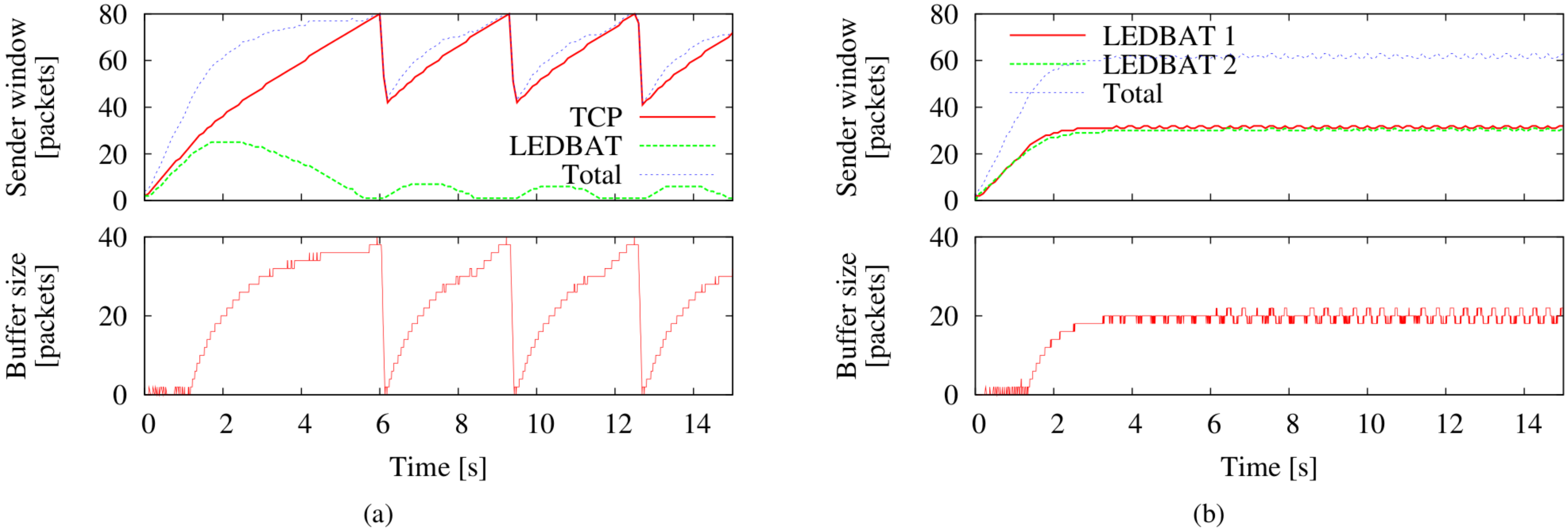



Fig. 2. Temporal evolution of the sender window (top) and of the queue size (bottom) for TCP-LEDBAT (a) and LEDBAT-LEDBAT interaction (b)

http://perso.telecom-paristech.fr/~valenti/papers/ledbat-icccn10.pdf

# BitTorrent: Summary

- **Good bandwidth utilization**
- **Limit free riding → Tit-for-tat**
- **Limit leech attack**
  - Coupling upload & download
- **Many implementations**
  - Various clients available
- **Decentralized tracker replacements exist**
- **Small files → Lead to latency, overhead**
- **Central tracker server needed to bootstrap swarm**
  - Single point of failure
  - Scalability issue
- **Cannot totally avoid malicious attacks at leechers (BitThief)**

# BitTorrent / P2P Future

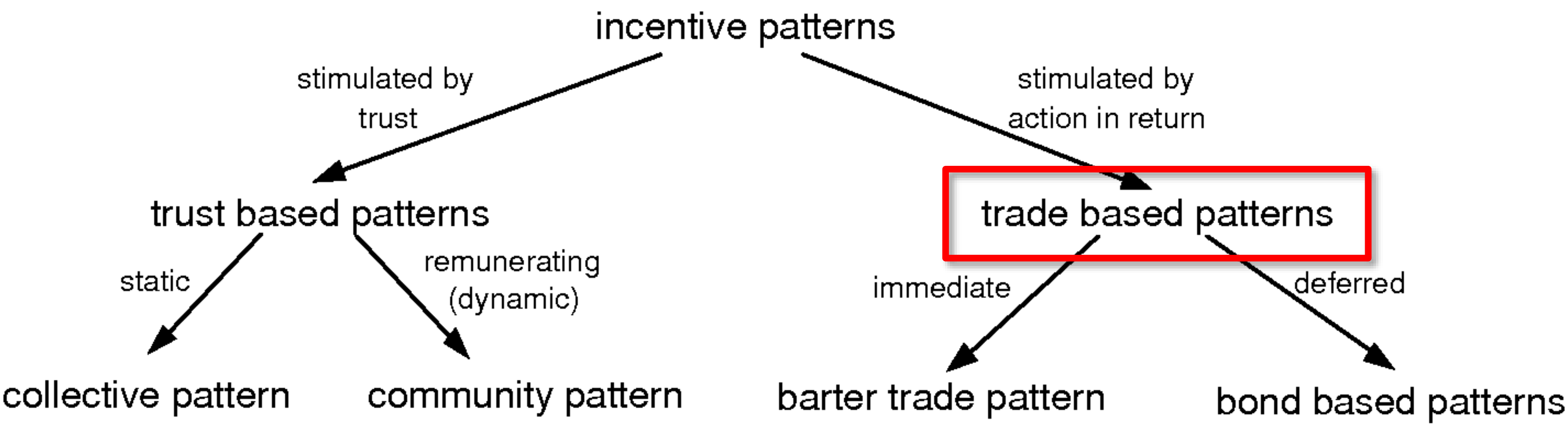
- **One-click hoster seem to be able to handle traffic**
  - Spotify changed from P2P to cloud, can handle traffic
  - What is the need for P2P?
- **Popcorn Time with very good user experience**
- **BitTorret Labs – P2P experiments**
  - BitTorrent Live – shut down in Feb. 2014
  - BitTorrent Sync – distributed Dropbox
  - BitTorrent Chat – secure instant message
  - BitTorrent Maelstrom – distributed web
- **CSG, UZH – P2P CT experiments**
  - Distributed Skype-clone, distributed Dropbox clone, distributed P2P radio, distributed social network, distributed file system

# 2. Incentives

**Introduction, Idea, Examples, Demo**

# Incentives

- **Incentive Patterns**

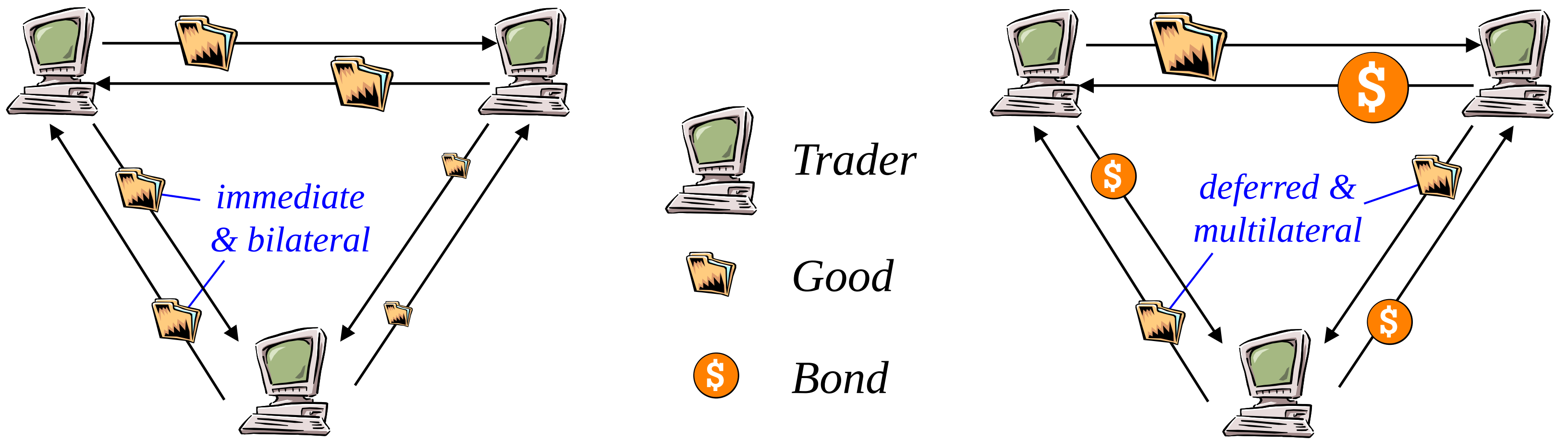



- **Example: BitTorrent**

# Incentives

- **Barter-Trade-based**
  - \- Only immediate & bilateral trading
  - \- Exchanged goods must be of equal value
  - \+ Low transaction costs

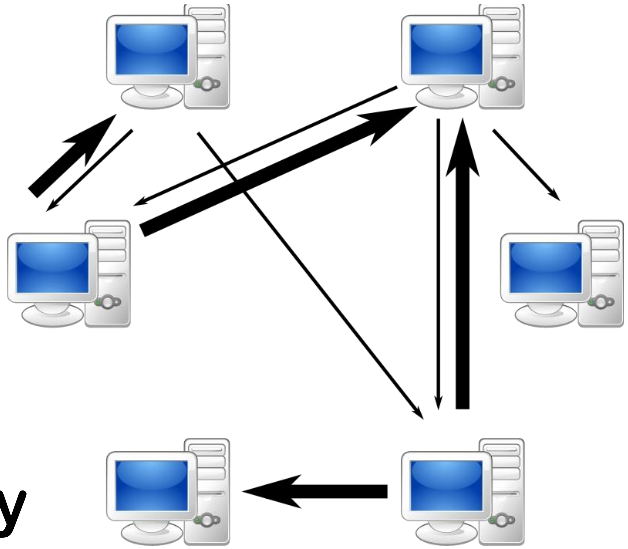
- **Bond-based**
  - \+ Flexibility: deferred & multilateral trading
  - \- Forgery
  - \- Double-spending
  - \- High transaction costs
  - \- User acceptance?



# Incentives

- **Decentralized networks: lack of control**
  - Free-riding, selfish, malicious peers
  - Early Gnutella: 70% peers share nothing

  → **Incentive mechanisms necessary**



- **TFT (~ used in BT) exploits symmetry of interest**
  - Peer S → peer T / peer T → peer S
    - symmetry of interest
    - private history
  - Asymmetry not exploited! → Transitive TFT
    - S → I → T / T → S

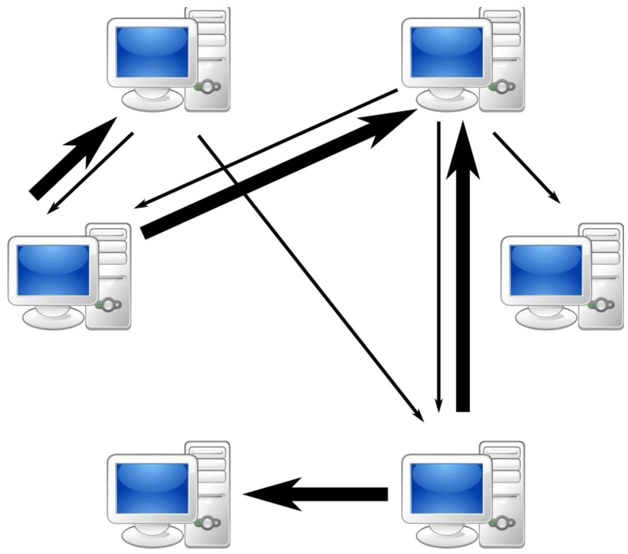
# Incentives

- **CompactPSH: exploits asymmetry of interest**
  - ▶ 1st step: find transitive path
  - ▶ 2nd step: verify information using private history (maxflow)

  \+ Collusion resistant (initial trust! BitThief)

  \+ Exploit more reciprocity

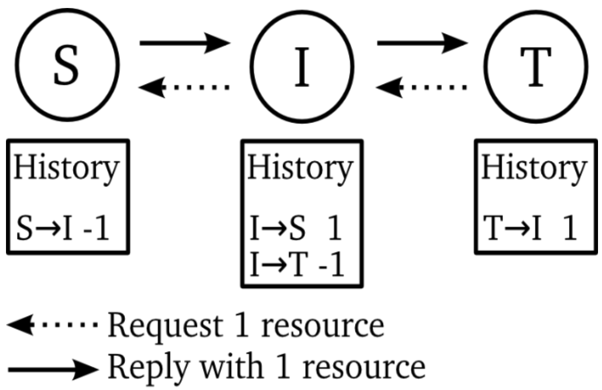
  \- Overhead for finding transitive path

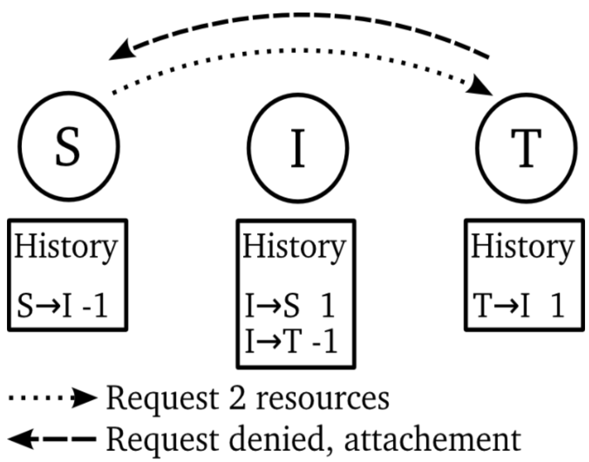


- **Analysis / experiments showed: 1 hop transitive path works best**
  - ▶ Tradeoff: exploitation / path search complexity

# Incentives

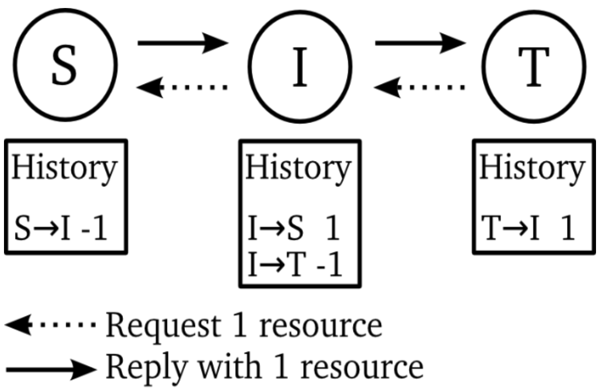
- **Key concept IOU**: S provided to I, I provided to T, initial credit: 1



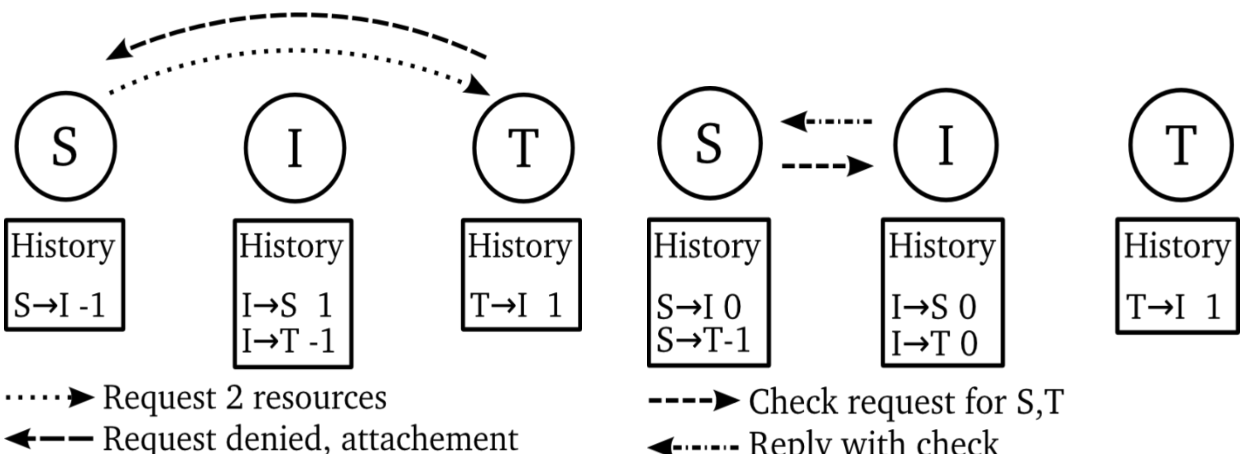



# Incentives

- **Key concept IOU: S provided to I, I provided to T , initial credit: 1**

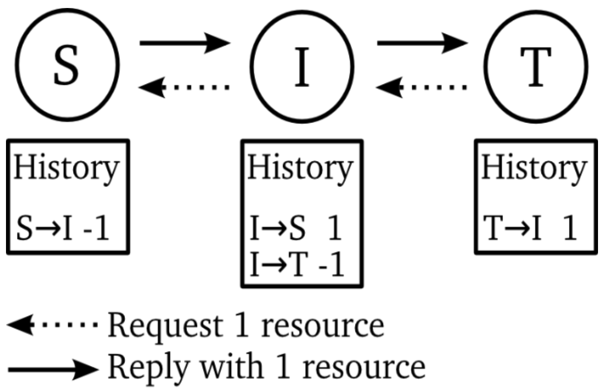



- S can request IOU from I
- I issues a IOU to S:
  - contains: S provided to T
  - I clears history information S and T

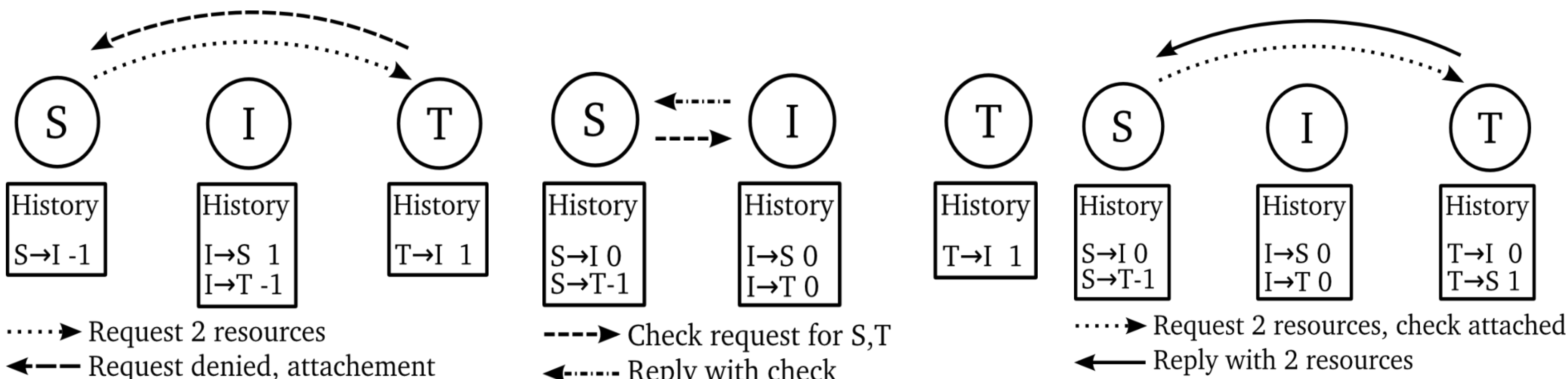


# Incentives

- **Key concept IOU: S provided to I, I provided to T , initial credit: 1**

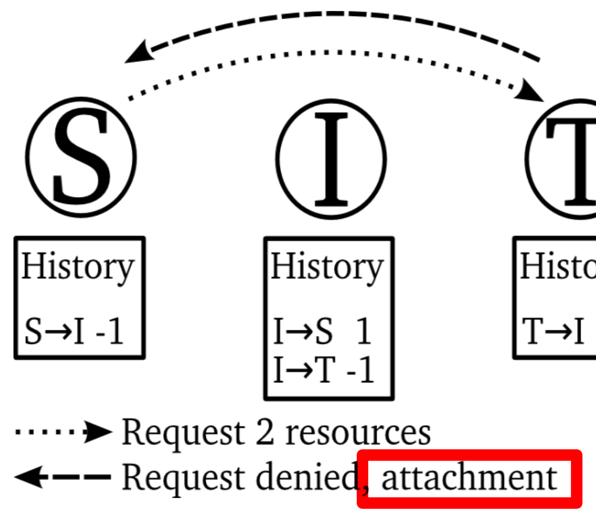


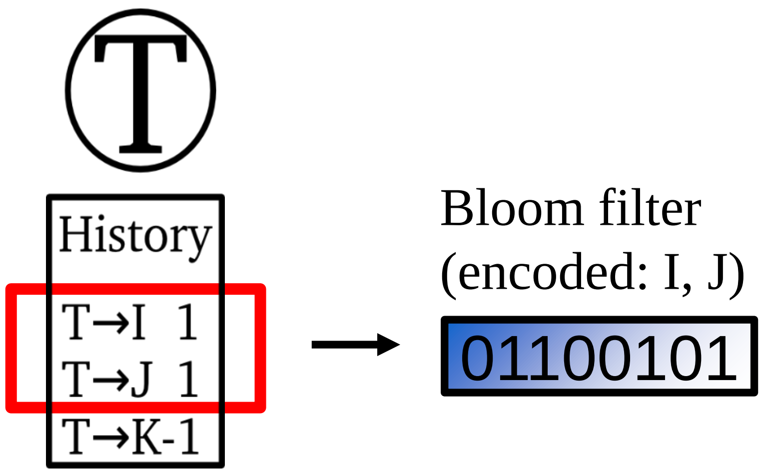

- S can request IOU from I
- I issues a IOU to S:
  - contains: S provided to T
  - I clears history information S and T
- S attaches IOU and requests from T



# Incentives / Bloom Filter

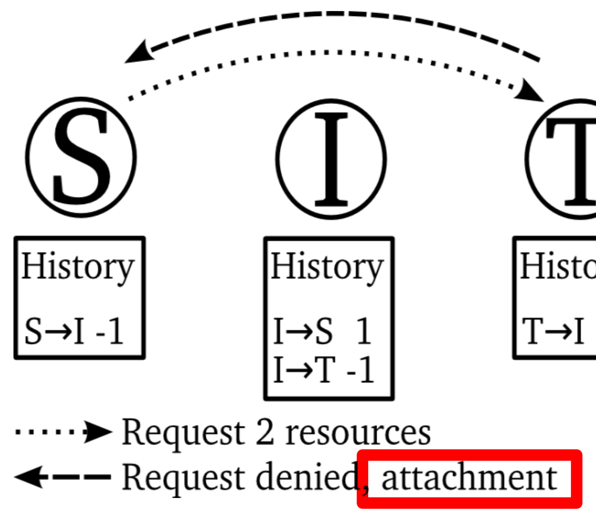
- **Reducing overhead for finding transitive path**
  - Bloom Filters!
- **T: generate Bloom filter with all peers having a positive balance → peer I, peer J**

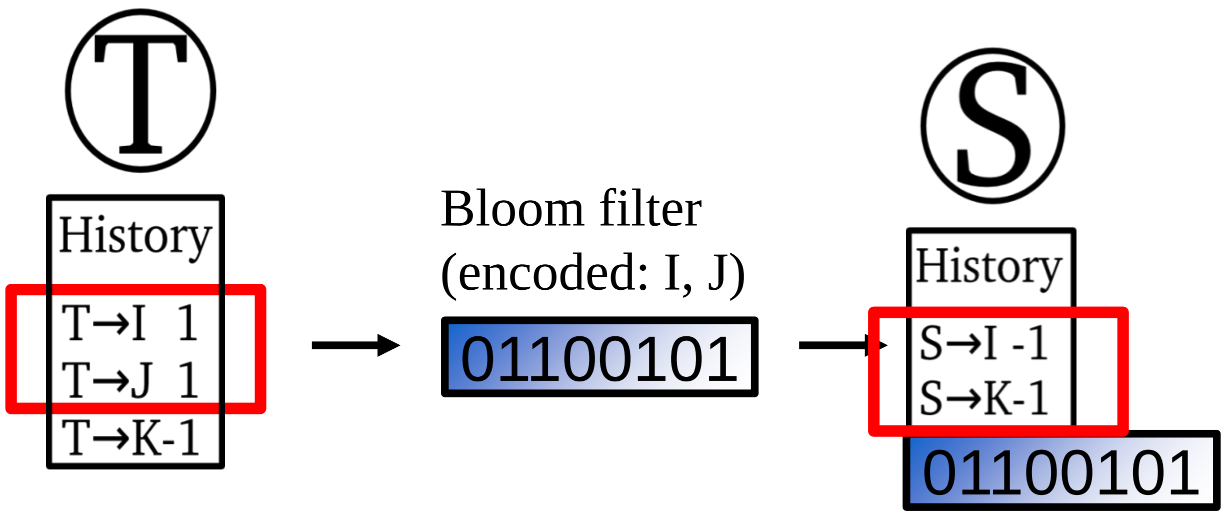


# Incentives / Bloom Filter

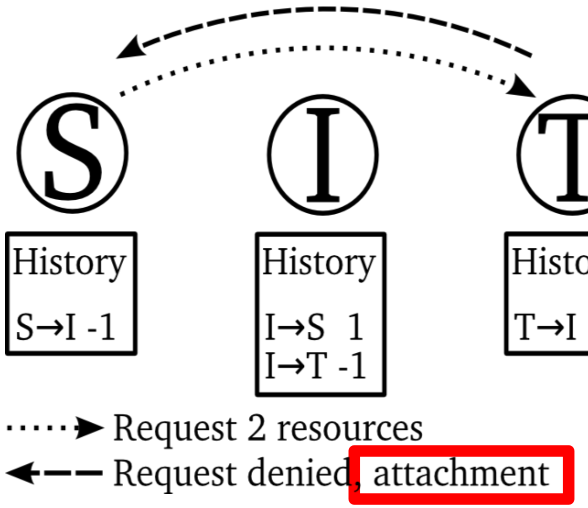
- **T: generate Bloom filter with all peers having a positive balance → peer I, peer J**
- **S: tests all peers with negative balance with Bloom filter**

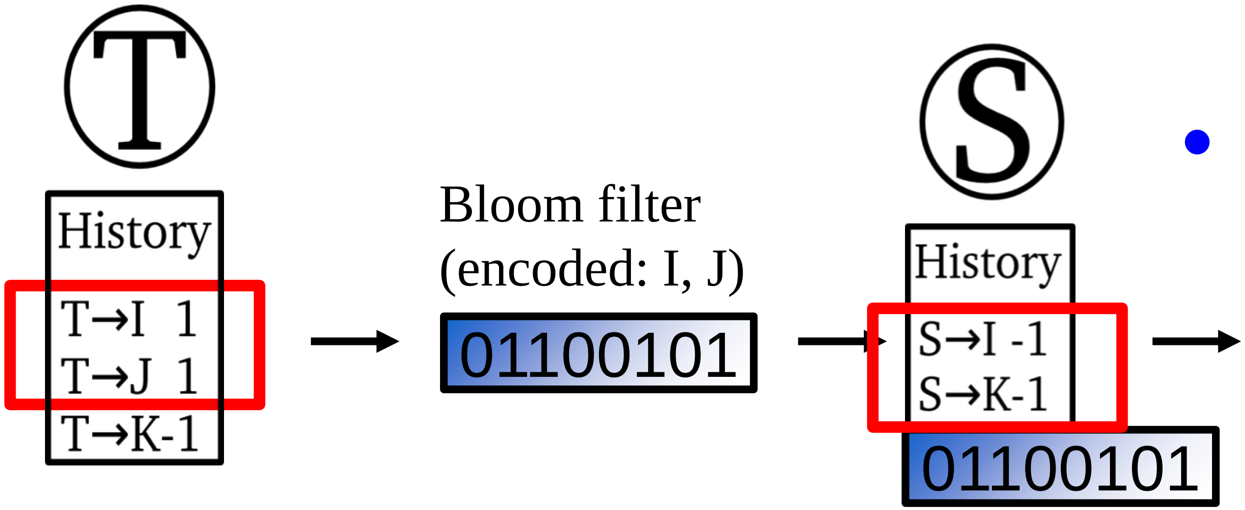


# Incentives / Bloom Filter

- T: generate Bloom filter with all peers having a positive balance → peer I, peer J
- S: tests all peers with negative balance with Bloom filter




- Resulting in peer I
  - I as intermediate
  - Request IOU from peer I

# CompactPSH

- **Application scenarios**
  - System with asymmetric interest → CDN
- **Main conclusions**
  - One-hop transitive TFT works best
  - Path search expensive, outdated information


- T. Bocek, W. Kun, F. V. Hecht, D. Hausheer and B. Stiller: PSH: A Private and Shared History-based Incentive Mechanism. 2nd International Conference on Autonomous Infrastructure, Management and Security Resilient Networks and Services (AIMS 2008), Bremen, Germany, July 2008
- M. Piatek, T. Isdal, A. Krishnamurthy and T. Anderson: One hop Reputations for Peer to Peer File Sharing Workloads. 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI'08), San Francisco, California, pp 1-14, Berkeley, CA, USA, April 2008
- M. Meulpolder, J. Pouwelse, D. Epema and H. Sips: BarterCast: Fully Distributed Sharing-Ratio Enforcement in BitTorrent. Technical Report, Delft University of Technology, August 2008

# 4. Rsync

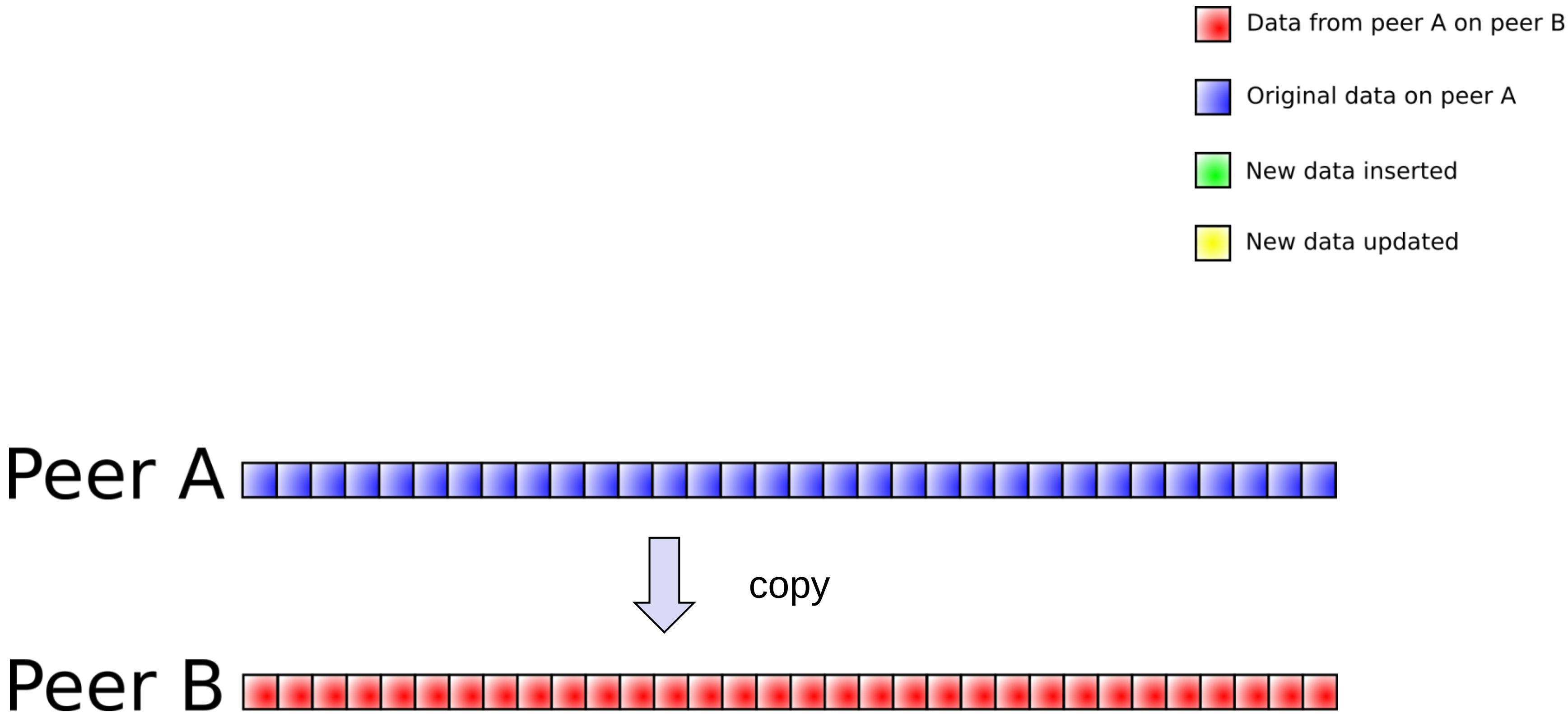
## Introduction, Example, and Discussion

# Rsync - Introduction

- **Rsync used to synchronize data over network**
  - Minimizing data transfer (delta)
- **Command line client (standard utility)**
  - E.g. `rsync -aP --link-dest=$HOME/Backups/current /path/to/important_files $HOME/Backups/back-$date`
  - Unchanged files are hard linked (`--link-dest`) → Can be used for incremental backups
- **Main idea**
  - Receiver compute two checksums (strong, weak) → sent to sender
  - Sender computes with weak checksum and checks for known blocks
  - Sender verifies with strong checksum → sends difference to receiver
- **Example with two peers:**

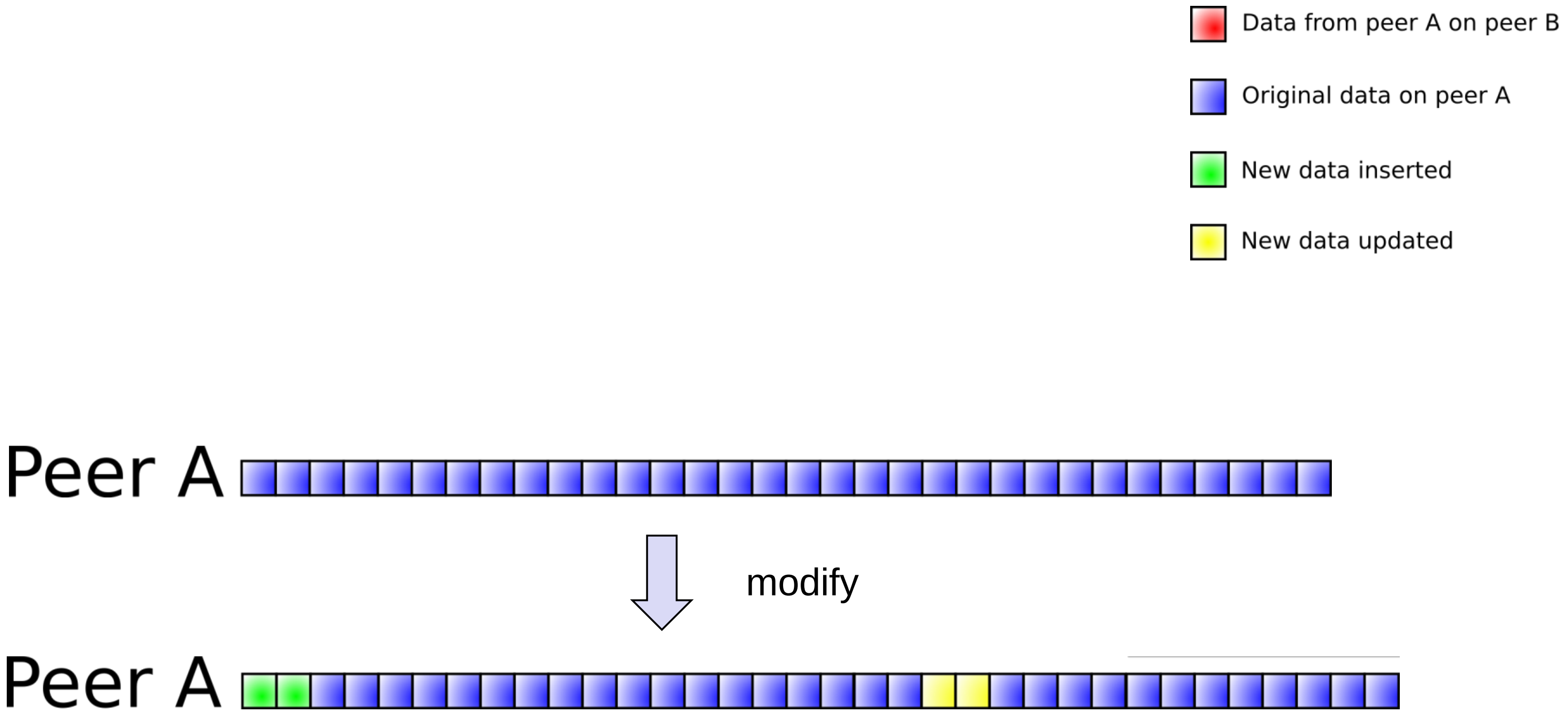
# Rsync - Example

- **Peer B does not have the data → peer A copies it to peer B, no need for rsync**



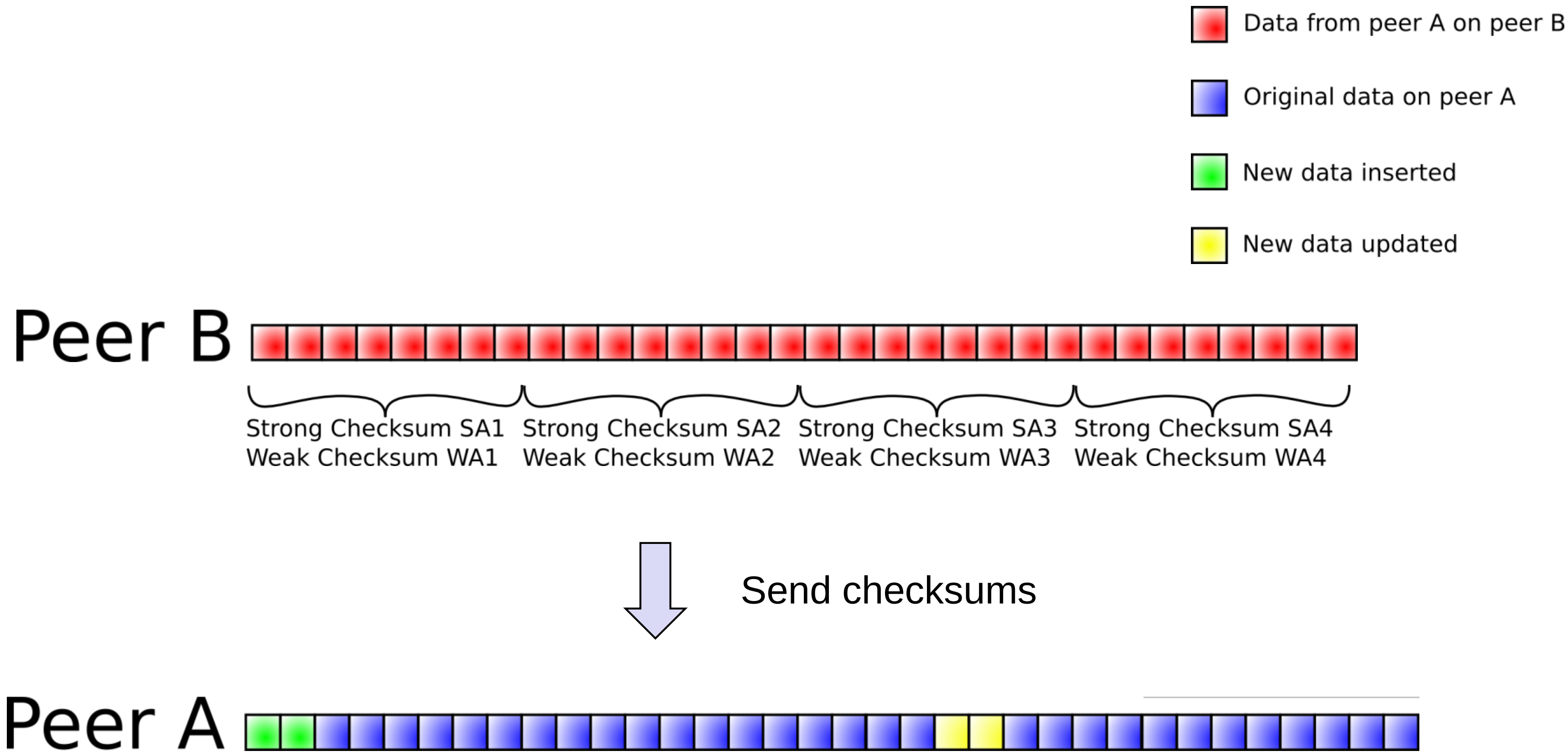
# Rsync - Example

- **Peer A modifies data (insert, update)**
  - Wants to synchronize with peer B

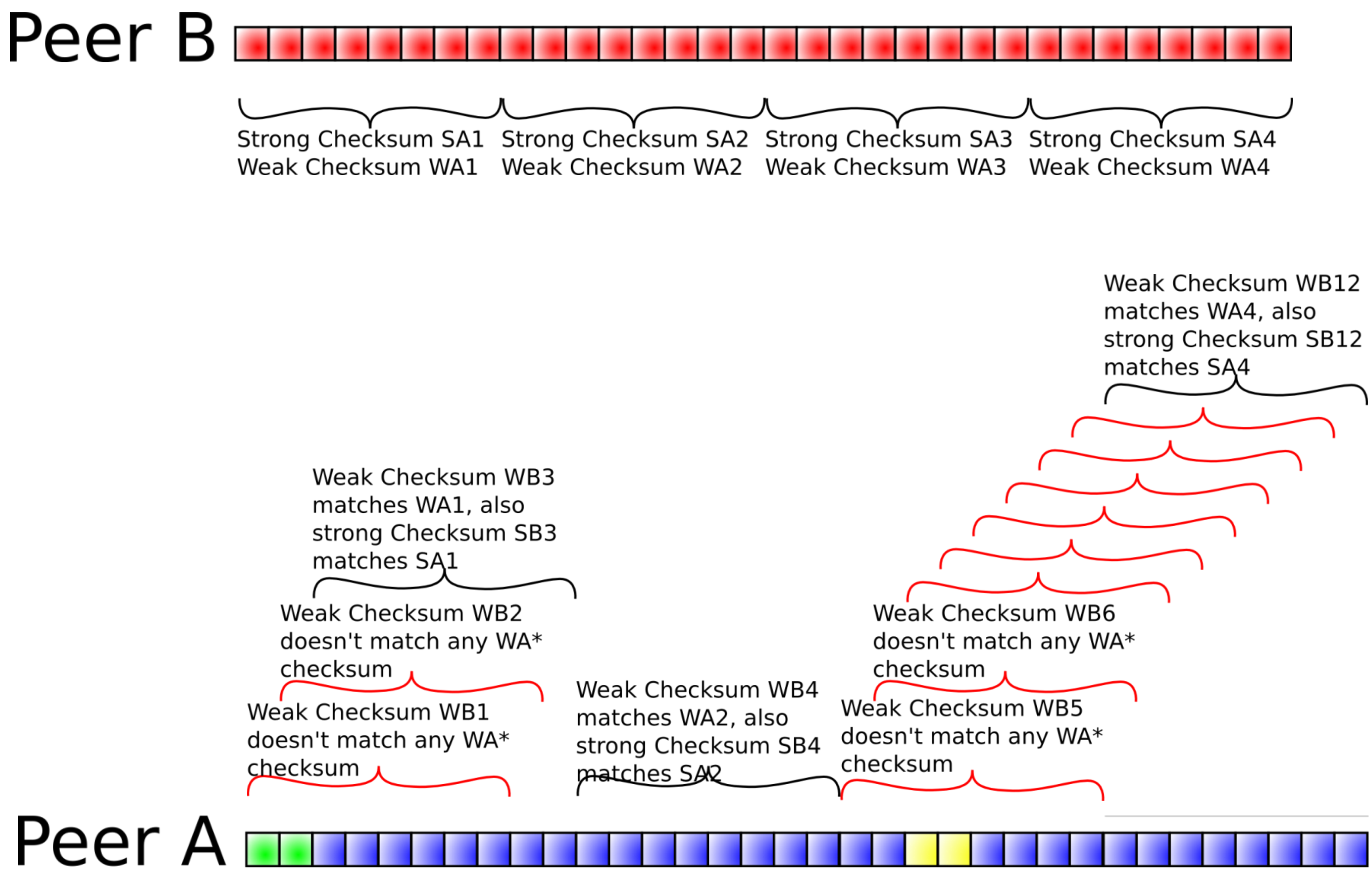


# Rsync - Example

- **Peer A modifies data (insert, update)**
  - Wants to synchronize with peer B

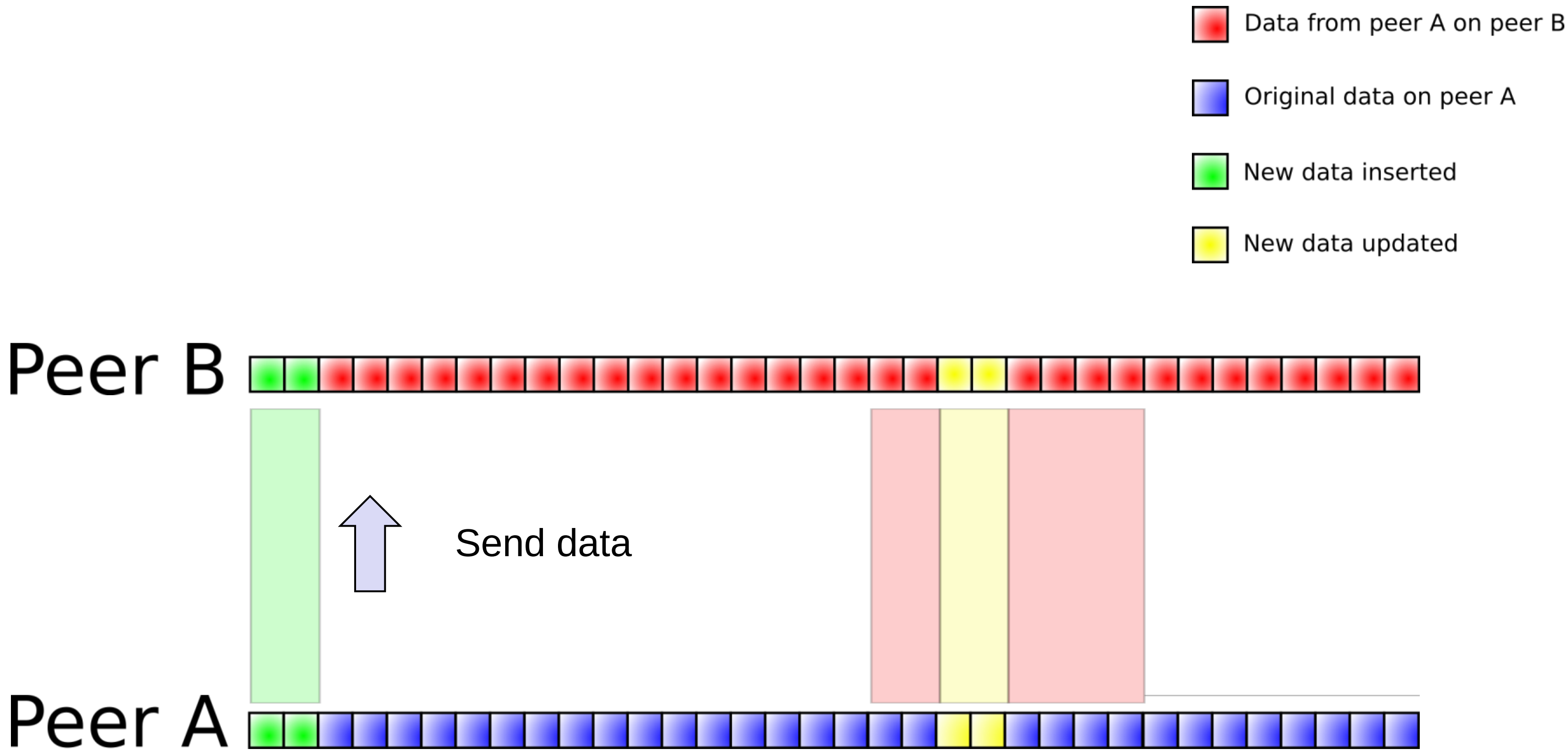


# Rsync - Example



# Rsync - Example

- **Peer A sends 2 + 8 blocks to peer B**
  - Peer A and peer B have same data



# Rsync - Mechanism / Discussion

- **If data does not exist → copy**
  - Use-case: portion of data stays the same
  - Replication
- **Problem if data is compressed**
  - Compressed data changes (adaptive compression)
  - Restart adaption / reset compression algorithm
- **Two checksums for performance (MD5 and Adler-32)**
  - Collisions possible, but unlikely $2^{-160}$
- **Rsync in TomP2P (demo)**
  - If you use CoW, don't use Rsync!
  - net.tomp2p.examples.ExampleRsync (new)

# Network Coding

**Motivation**

**Theory**

**Example**

# Network Coding

- **P2P problem area: distribute large data packets to many clients**
  - Solution 1: exchange piece/chunk bitmap
    - Request contains the pieces we are looking for, recipient replies with piece or that pieces are not here
    - Peer may also collect piece/chunk bitmaps first
  - Solution 2: use network coding
    - Simple request, when there is nothing new, don’t request.

- **Random network coding: send linear combination of all received chunks**

→ no need to exchange piece/chunk information

# Network Coding

- M1, …, Mn → **original packets /** g1, …, gn → **random coefficients**
- **Encoding:** $X = \sum_{i=1}^{n} g_i M^i$
  - Encoding vector g, information vector X
  - Send X,g
- **Re-encoding,** h1, …, hm → **random coefficients :**

$$X' = \sum_{j=1}^{m} h_j X^j$$

  - Encoding vector h → g′ $g'_i = \sum_{j=1}^{m} h_j g_i^j$
  - Information vector X′
  - Send X′,g′
- **Random network coding example:**

# Network Coding

- Peer 1 chooses coefficients (encoding vector), calculates linear combination, sends two to Peer 2 and one to Peer 3

Coefficient (randomly chosen or try not to be linear dependant)

| Peer 1 | | |
|---|---|---|
| | M1=17 | 1,3,2 |
| | M2=8 | 2,1,3 |
| | M3=9 | 2,3,1 |

$X1 = 1 * 17 + 3 * 8 + 2 * 9 =$ 59 (1,3,2)

$X2 = 2 * 17 + 1 * 8 + 3 * 9 =$ 69 (2,1,3)

$X3 = 2 * 17 + 3 * 8 + 1 * 9 =$ 67 (2,3,1)

# Network Coding

- Peer 2 also has random coefficients, gets two linear comb. From Peer 1, creates new linear combination (adapt coeff.), send to Peer 3

(Coefficient)

Peer 2

5,4

1,1

59 (1,3,2)

69 (2,1,3)

$$X1' = 59 * 5 + 69 * 4 = \boxed{531}$$
$$X2' = 59 * 1 + 69 * 1 = \boxed{138}$$

$$\text{Coefficient } 1' = (5*1+4*1, 5*3+4*3, 5*2+4*2) = \boxed{(9,27,18)}$$
$$\text{Coefficient } 1'' = (1*2+1*2, 1*1+1*1, 1*3+1*3) = \boxed{(4,2,6)}$$

# Network Coding

- Peer 3 gets all 3 linear combinations with coding vector, does Gauss elimination (linear system with 3 equations and 3 unknowns) [online]

Peer 3

67 (2,3,1)

531 (9,27,18)

138 (4,2,6)

$$
\begin{aligned}
9M1 + 27M2 + 18M3 &= 531 \\
4M1 + 2M2 + 6M3 &= 138 \\
2M1 + 3M2 + 1M3 &= 67 \\
\hline
0M1 - 10M2 - 2M3 &= -98 \\
0M1 - 3M2 - 3M3 &= -51 \\
\hline
0M1 + 0M2 + 1M3 &= 9
\end{aligned}
$$

M1=17

M2=8

M3=9

# Network Coding

- Peer 3 gets other linear combinations with coding vector, does Gauss elimination (linear system with 3 equations and 3 unknowns) [online]

$$X1 = \boxed{1} * \boxed{17} + \boxed{3} * \boxed{8} + \boxed{4} * \boxed{9} = \boxed{77\ (1,3,4)}$$

Peer 3

$\boxed{531\ (9,27,18)}$

$\boxed{138\ (4,2,6)}$

$$
\begin{array}{l}
9M1 + 27M2 + 18M3 = 531 \\
4M1 + 2M2 + 6M3 = 138 \\
1M1 + 3M2 + 4M3 = 77 \\
\hline
0M1 - 10M2 - 2M3 = -98 \\
0M1 - 0M2 - 2M3 = 18 \\
\hline
0M1 + 0M2 + 1M3 = 9
\end{array}
$$

$\boxed{M1=17}$

$\boxed{M2=8}$

$\boxed{M3=9}$

# Network Coding

- **Send packets as long as linear independant**
  - No exchange of bitmap
- **Random coefficients: low probability of collision**
- **Overhead of transmitting the encoding vectors is small, size of block is Kbytes**
  - CPU overhead $O(n^3)$, (n = number of blocks)

- **Experiments in P2P networks: improve robustness and throughput**