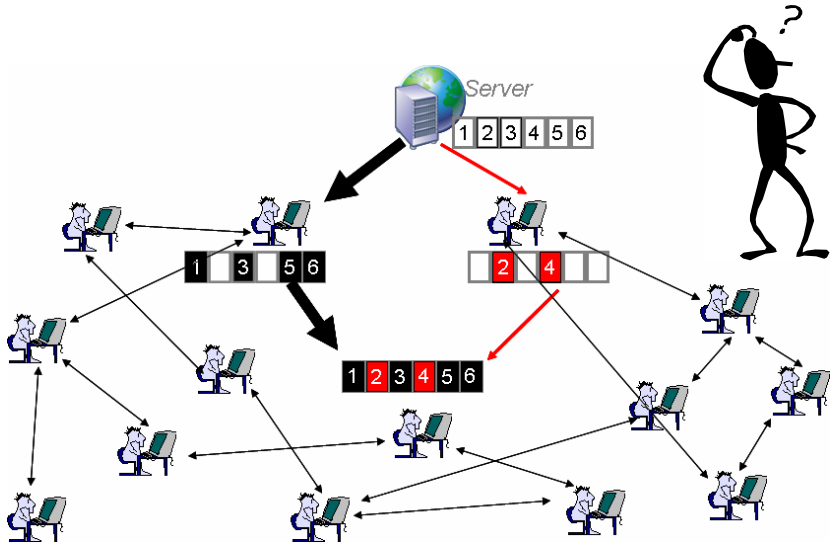
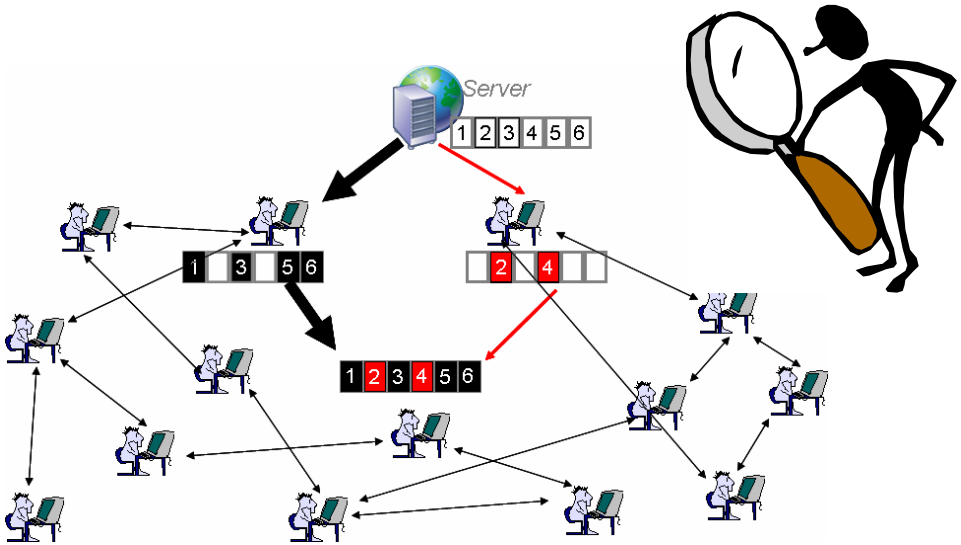
Anatomy of a P2P Content Distribution System with Network Coding



Christos Gkantsidis, John Miller, and Pablo Rodriguez

Microsoft Research, Cambridge

Anatomy of a P2P Content Distribution System with Network Coding

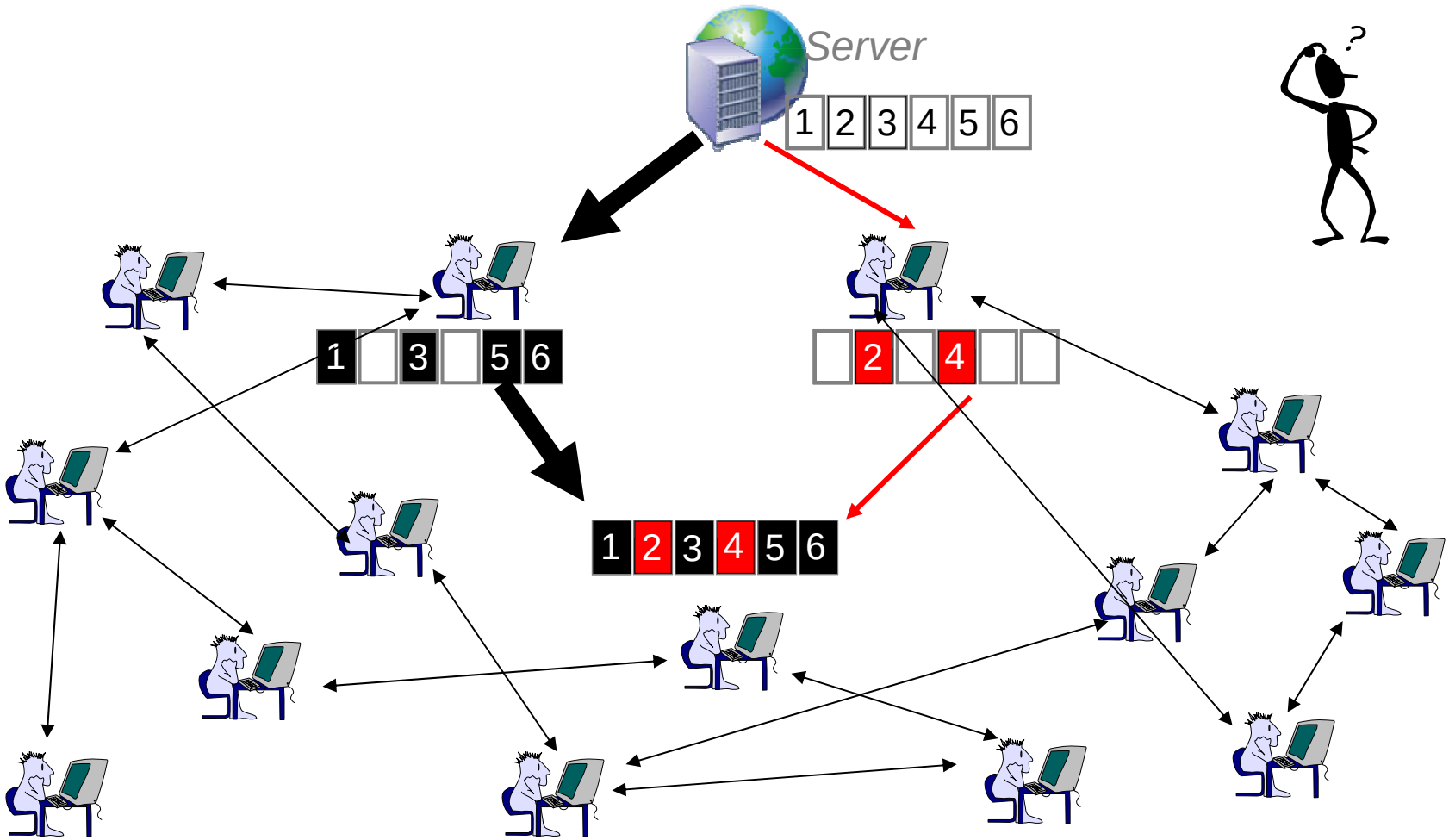


Christos Gkantsidis, John Miller, and Pablo Rodriguez

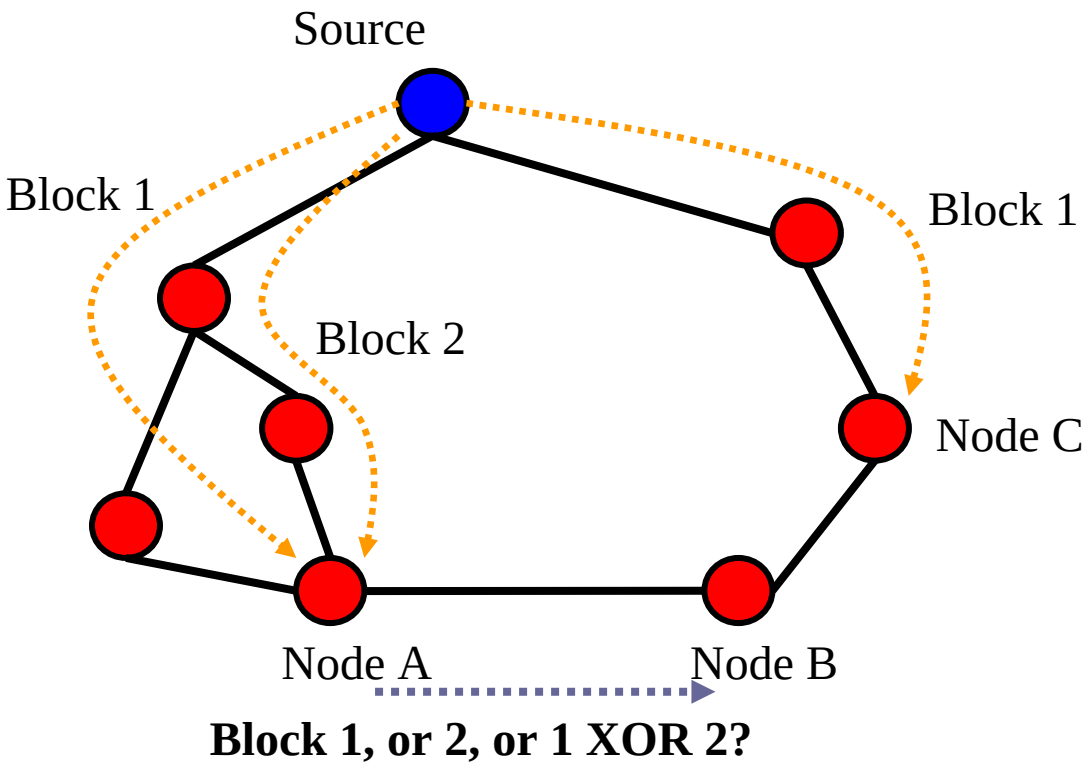
Microsoft Research, Cambridge

Peer-to-Peer File Swarming

What's the best path for each data block?



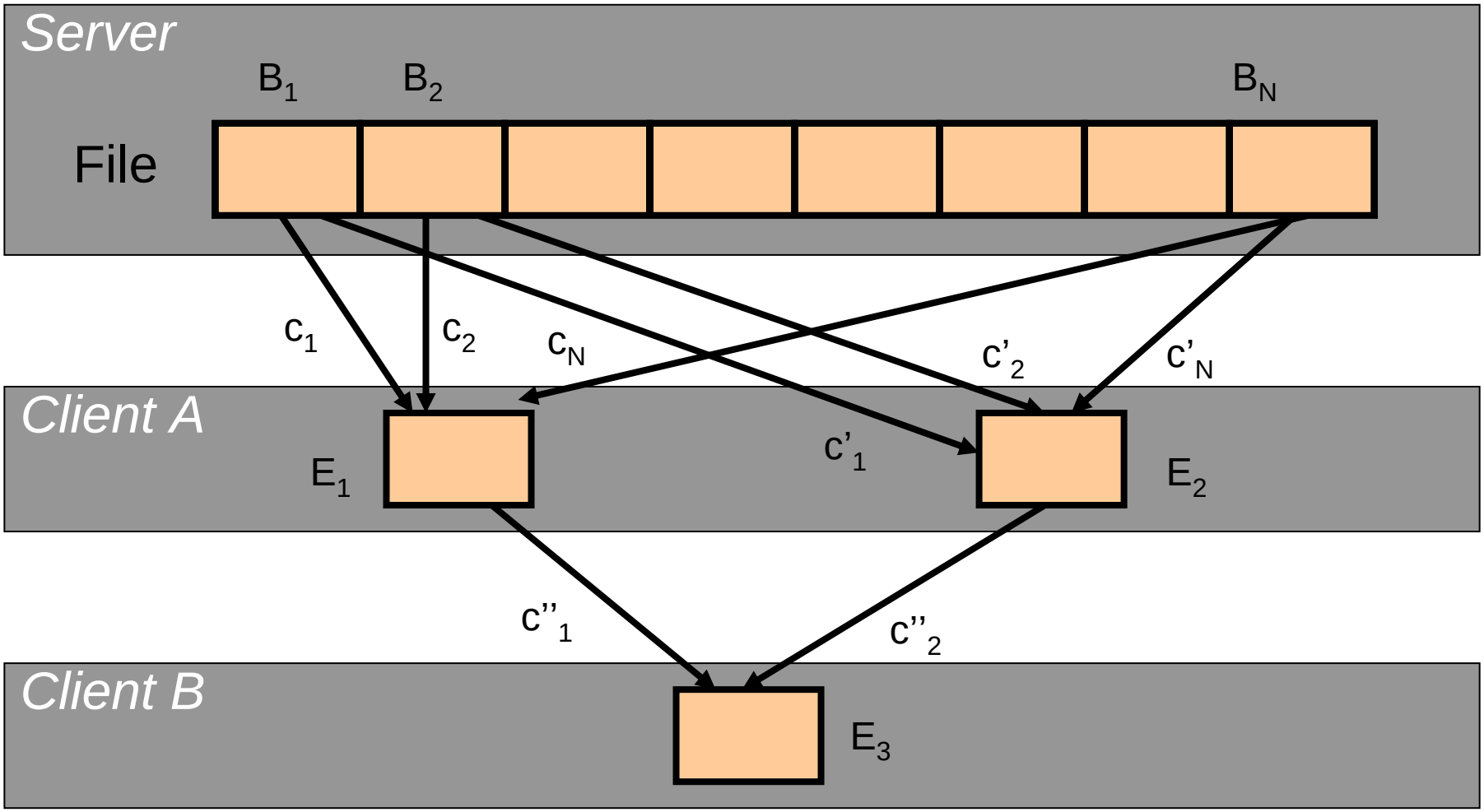
The Problem of Efficient Scheduling of Information



Benefits:

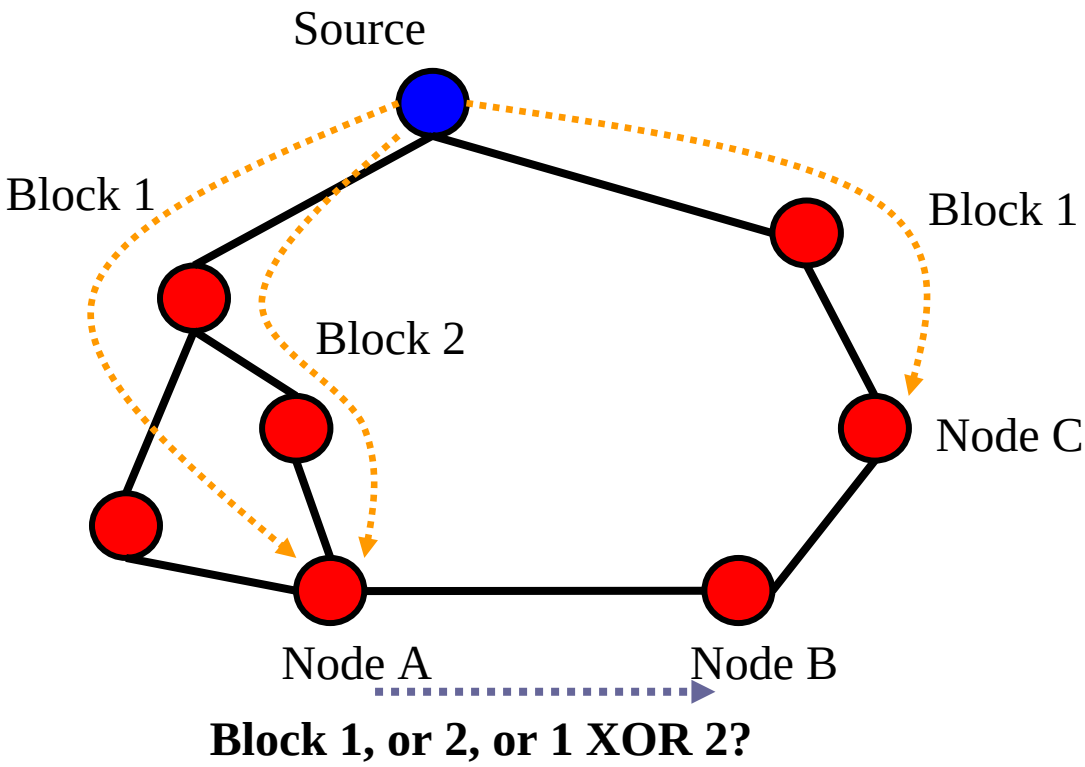
- **Performance improvements**: analytically and experimentally
- **Simplified system design**:
  - No need to identify rarest block in a distributed system
  - Complexity at the edges and *not* at the network
  - Solves reconciliation problem
  - Smaller neighbour set

Encoding in the Network



Coefficient vector: ($c''_1 c_1+c''_2c'_1$, $c''_1 c_2+c''_2c'_2$, …)

The Problem of Efficient Scheduling of Information



Benefits:

- **Performance improvements**: analytically and experimentally
- **Simplified system design**:
  - No need to identify rarest block in a distributed system
  - Complexity at the edges and *not* at the network
  - Solves reconciliation problem
  - Smaller neighbour set

But:

- **Is it practical?**
  a) Encoding / decoding costs,
  b) Content security.
- **Does network coding result in better performance?**

Summary of findings

- Network coding:
  - Good delivery performance
  - Easy to implement, i.e. small & bug free
  - Low encoding/decoding overheads
  - Efficient security mechanism
- General peer-to-peer distribution
  - Quantified benefits for content providers
  - Effect of unreachable users (NATs, firewalls)

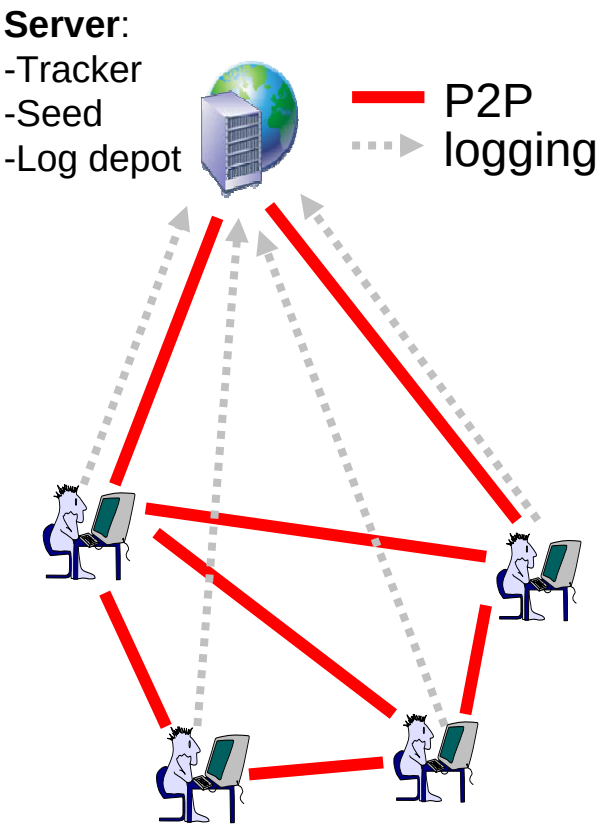
Outline

- Performance of P2P content distribution
  - Experimental setup
  - Delivery performance
  - Impact of unreachable nodes
- Network coding implementation
  - Encoding/decoding overhead
  - Protection against content pollution
- Summary

Outline

- **Performance of P2P content distribution**
  - **Experimental setup**
  - **Delivery performance**
  - **Impact of unreachable nodes**
- Network coding implementation
  - Encoding/decoding overhead
  - Protection against content pollution
- Summary

Experimental Setup



Server capacity: 2.5Mbps

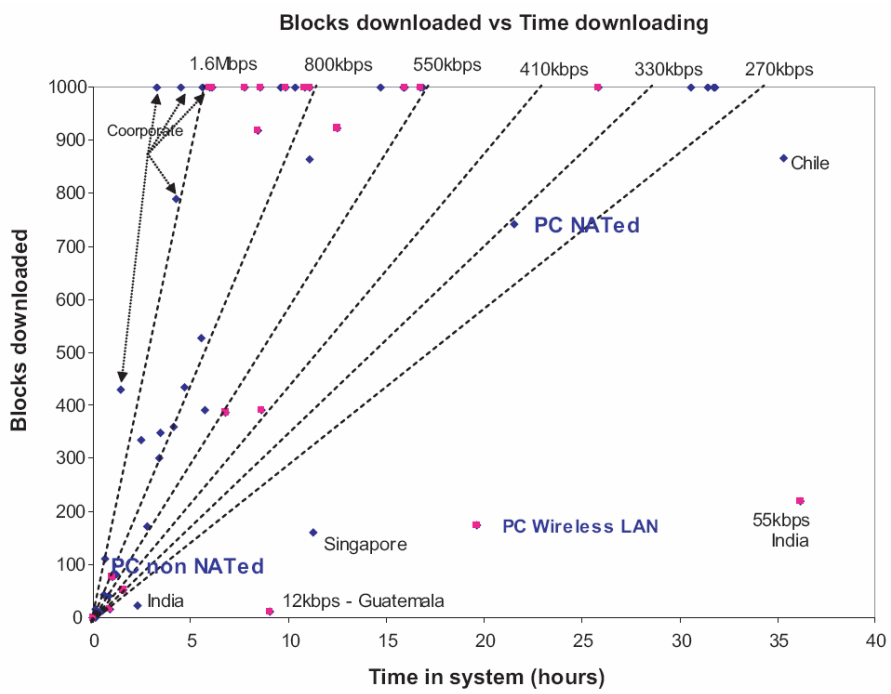
Peer-to-Peer network (BitTorrent-like):

- Single seed server.
- Tracker: facilitates peer discovery.
- Random peer discovery.
- Block exchanges using network coding.
- C# prototype

***Extensive* Logging** (complete view):

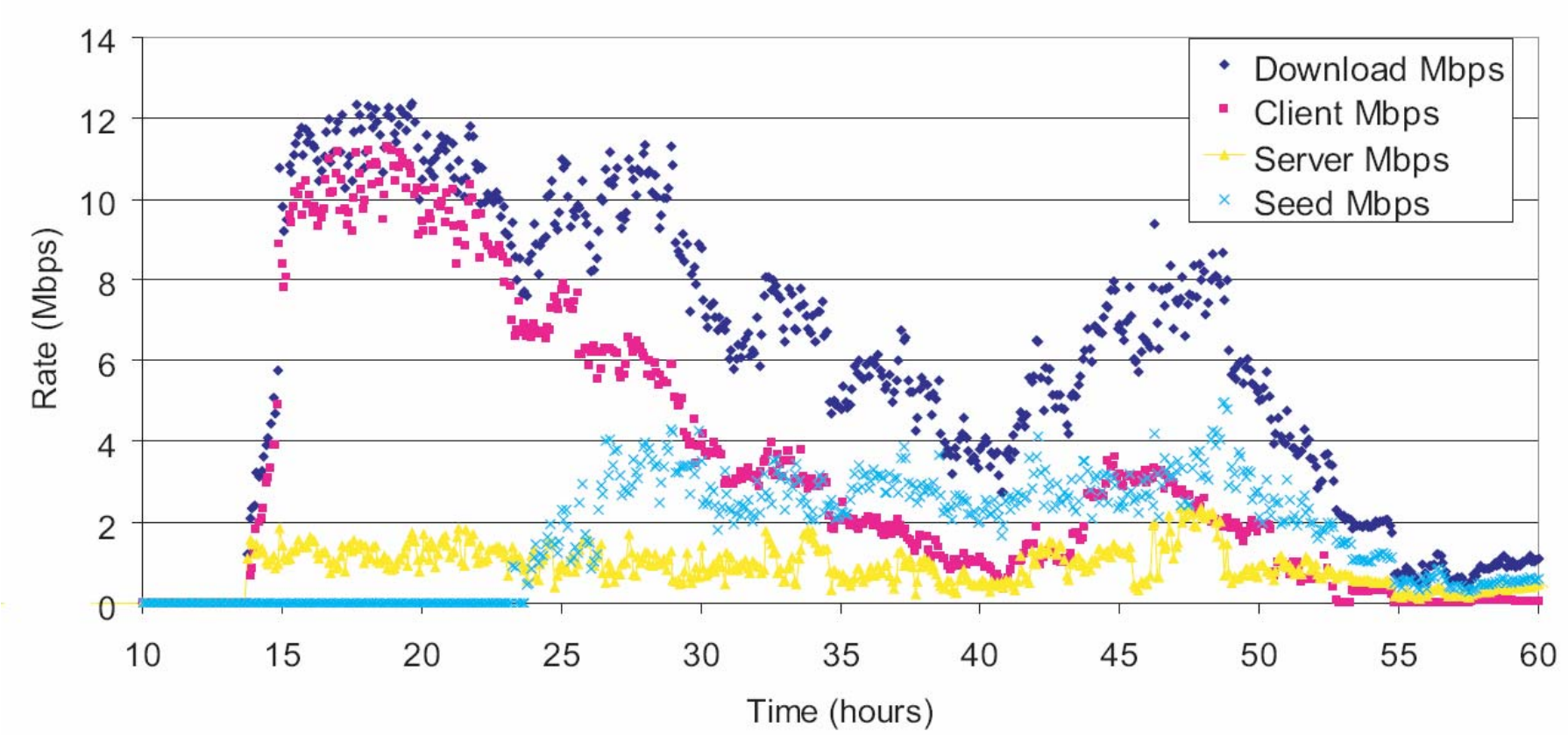
- Connection events:
  - Attempted and succeeded.
  - Reasons for failure (e.g. unreachability).
  - Periodic connectivity testing (e.g. find NATs).
- Block transfers
- Upload & download rates

Data collected



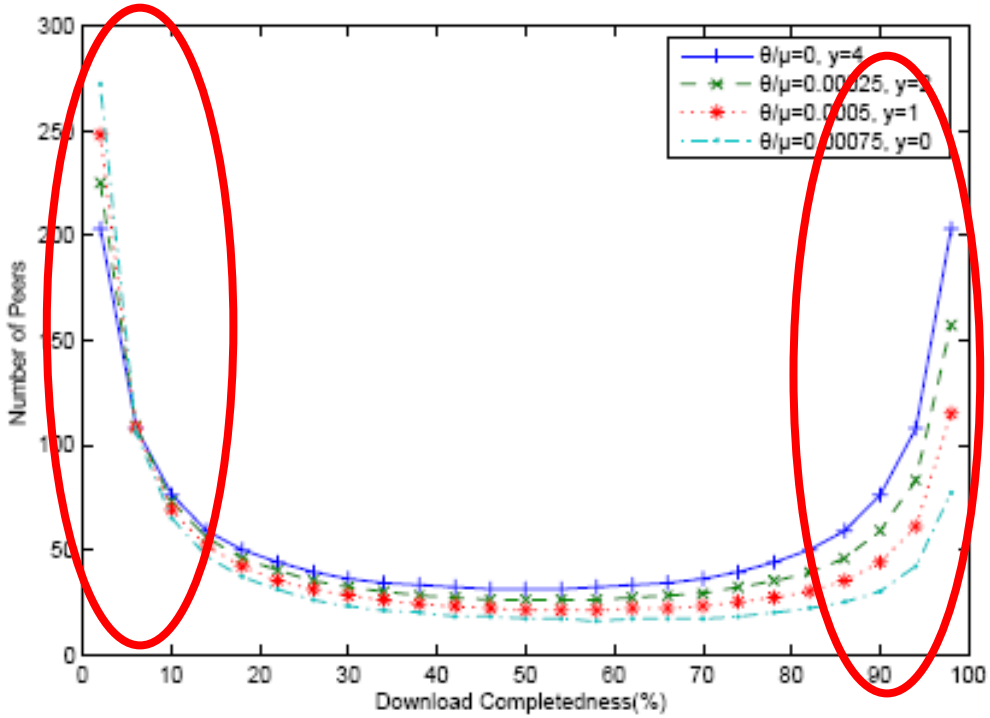
- Four trials
- Great variety of users:
  - Access capacities
  - Geographical
- File size: ~ 3.5GB
- Duration: 80-180hr
- Total # users: ~350
- Avg. Download time: 9-16hr

Bandwidth Contribution



- Easily withstands flash crowds
- Server contribution is fixed, Client contribution scales
- >10 fold savings in content provider's bandwidth using peer-to-peer.

System's progress in current File Swarming systems

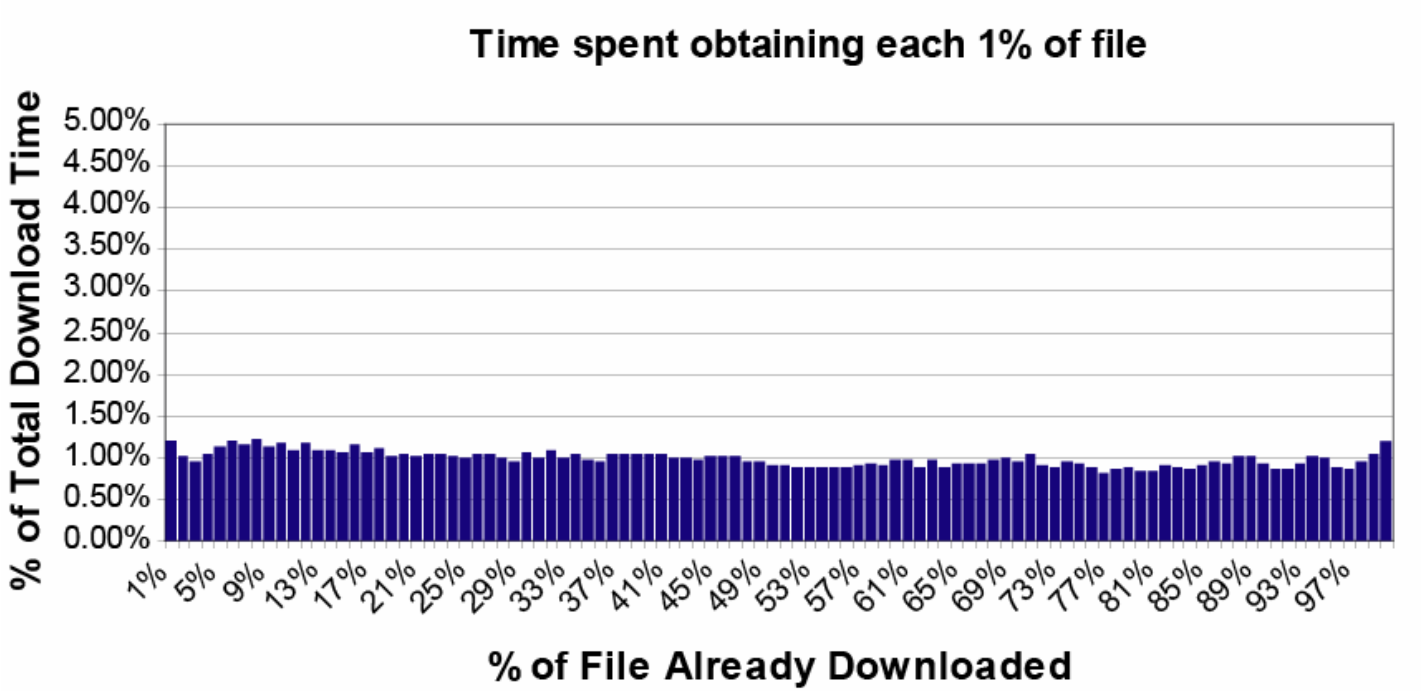


(From Tian et al., Infocom'06)

A lot of time spent at the beginning and finish of download:

- **Beginning of download: finding good blocks to exchange**
- **End of download: discovering the last missing blocks**

System's progress

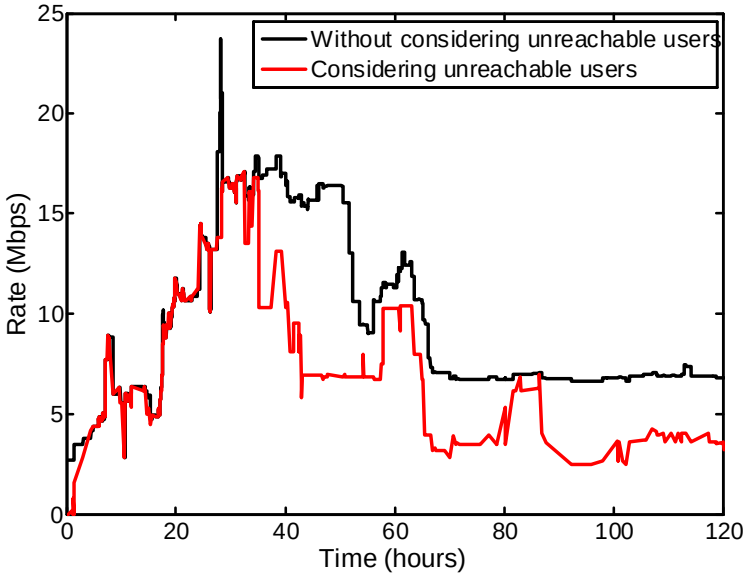


- Smooth download progress
  - No start-up delay
  - No last-block problem

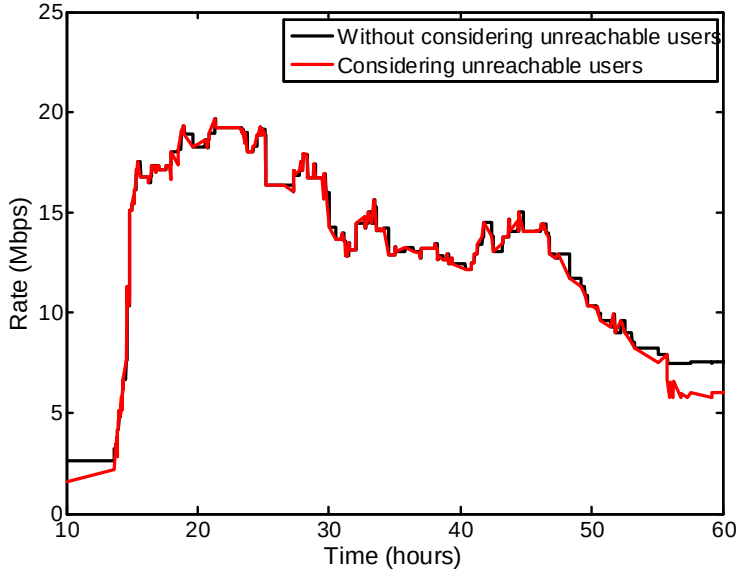
Effect of NATs/Firewalls

- Majority of users are behind NAT's/Firewalls!
- However, system can easily withstand large % of NATed users
  - connections can be initiated both ways
  - few fast nodes act as relays for others

>85% NATed users



< 65-70% NATed users



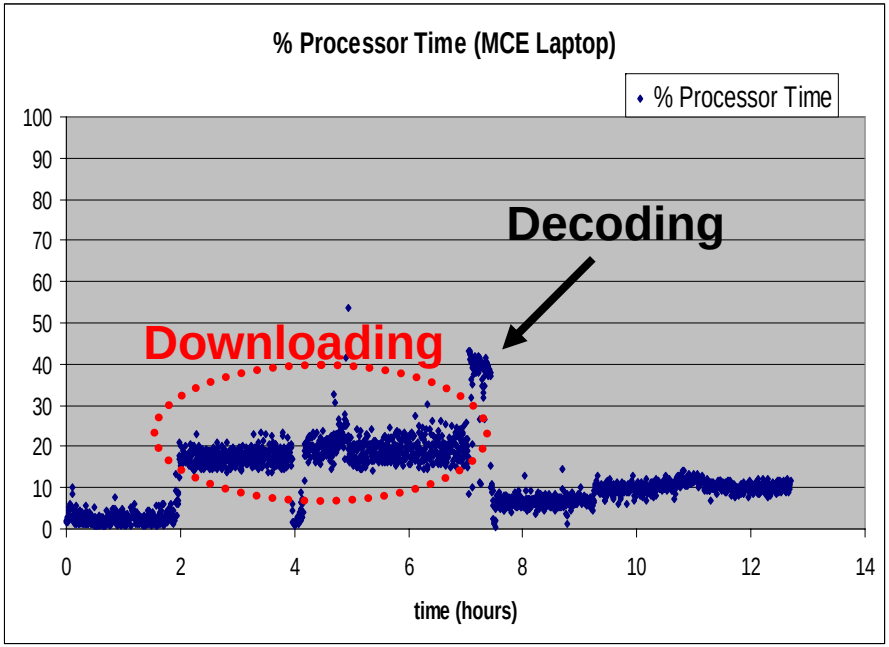
Outline

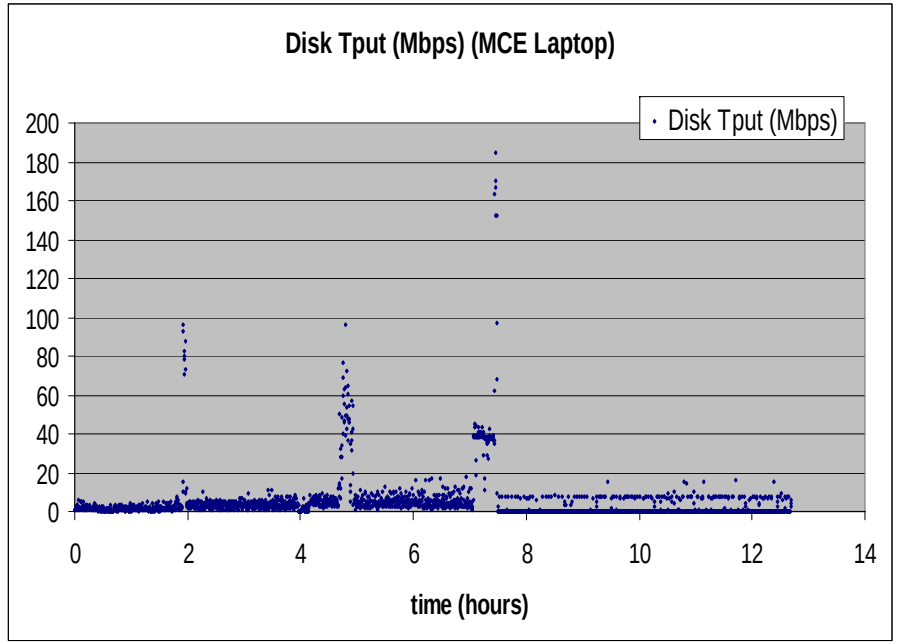
- Performance of P2P content distribution
  - Experimental setup
  - Delivery performance
  - Impact of unreachable nodes
- **Network coding implementation**
  - **Encoding/decoding overhead**
  - Protection against content pollution
- Summary

CG5

CPU and Disk Overhead

Resource Consumption (CPU, disk)

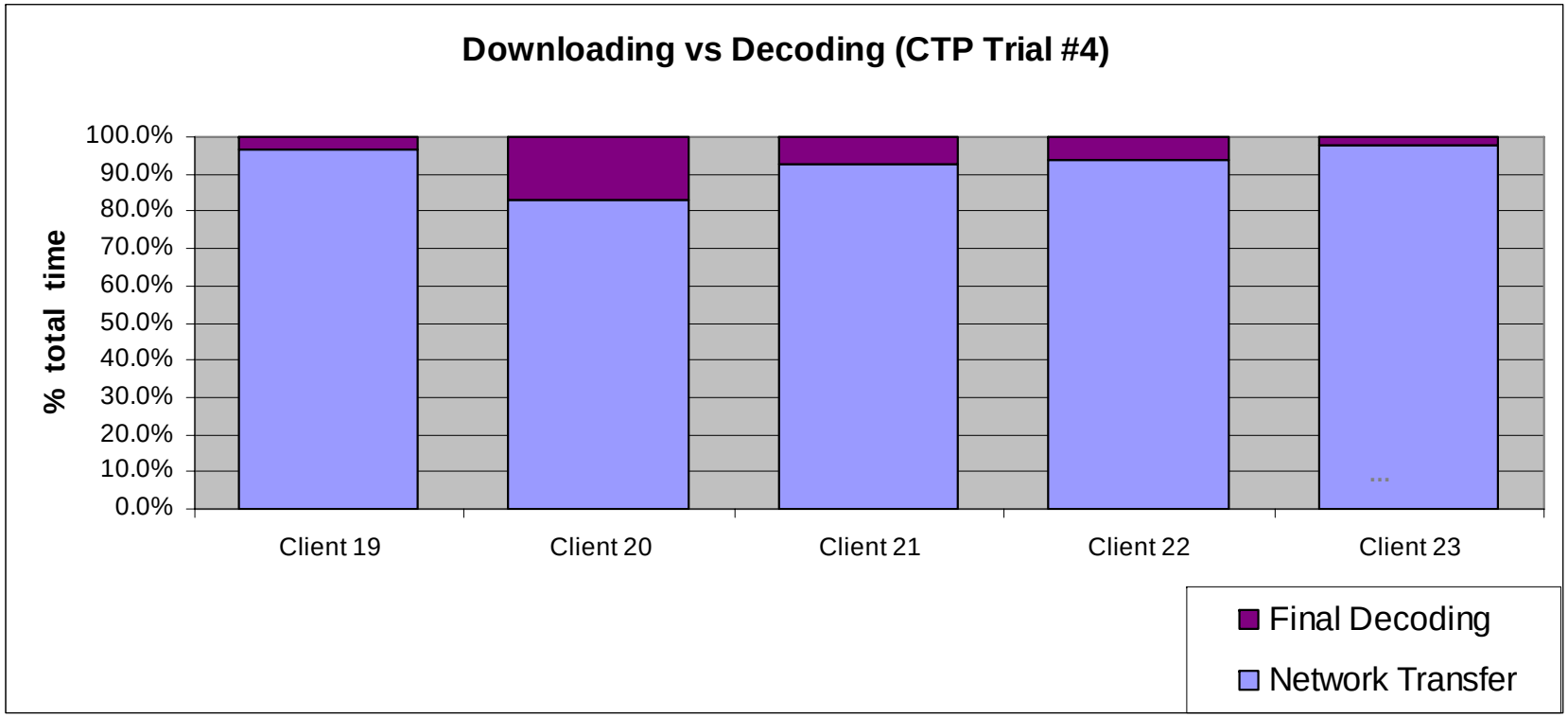




- Trace for 2 Ghz, 512 MB RAM Laptop running managed prototype
- CPU load averaged 20% during download, 40% during decode
- Disk activity quite low overall

Encoding / Decoding Effort

Coding overhead not significant



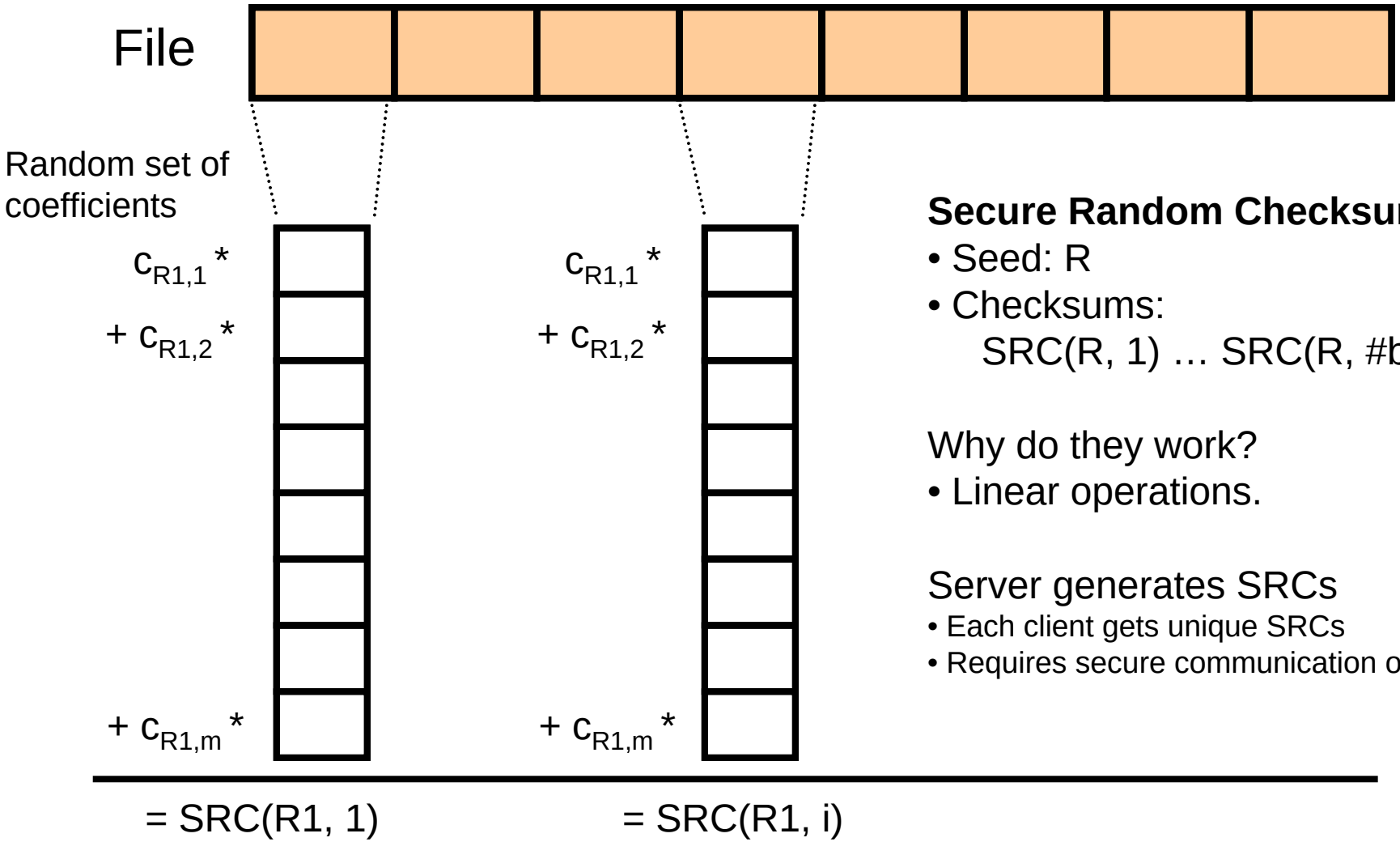
- Each bar represents trial participant who downloaded entire file
- Decoding on average took 6% of the download time
  - Progressive decoding eliminates extra time

Outline

- Performance of P2P content distribution
  - Experimental setup
  - Delivery performance
  - Impact of unreachable nodes
- Network coding implementation
  - Encoding/decoding overhead
  - **Protection against content pollution**
- Summary

Secure Random Checksums

- Peers produce “new” encoded blocks; server cannot sign.
- One bad block, corrupts all subsequent encodings



Secure Random Checksum:
- Seed: R
- Checksums:
  SRC(R, 1) … SRC(R, #blocks)

Why do they work?
- Linear operations.

Server generates SRCs
- Each client gets unique SRCs
- Requires secure communication of SRCs

Performance of SRCs

- Rate of checking encoded blocks:
  ~3.3 blocks/sec (*) or 100Mbps
  (*) For 3.7GB file divided into 1000 blocks
- Rate of producing SRCs: ~20 Mbps
  - Not so critical, since it is done once per client.
  - Pre-computing may be used to increase performance.

- In comparison, rate of checking with homomorphic hash functions:
  - Naïve: ~0.12 Mbps
  - Batching: ~32.7 Mbps

Summary

- Network coding is feasible
  - Easy to implement
  - Good encoding/decoding performance
  - Efficient security using SRCs
- Network coding performs well
- Significant content provider’s savings
- System can withstand high levels of unreachability

Further questions

- Matching algorithms
- Monitoring of P2P performance
  - Commercial applications: BBC iMP, Universal, …
- NAT traversal techniques and their true value
- Ultimate CDNs: P2P + Caching
- Growing imbalance of uplink/downlink ratio.
- Near VoD using unstructured P2P