Improving Peer-to-Peer File Distribution: Winner Doesn't Have to Take All

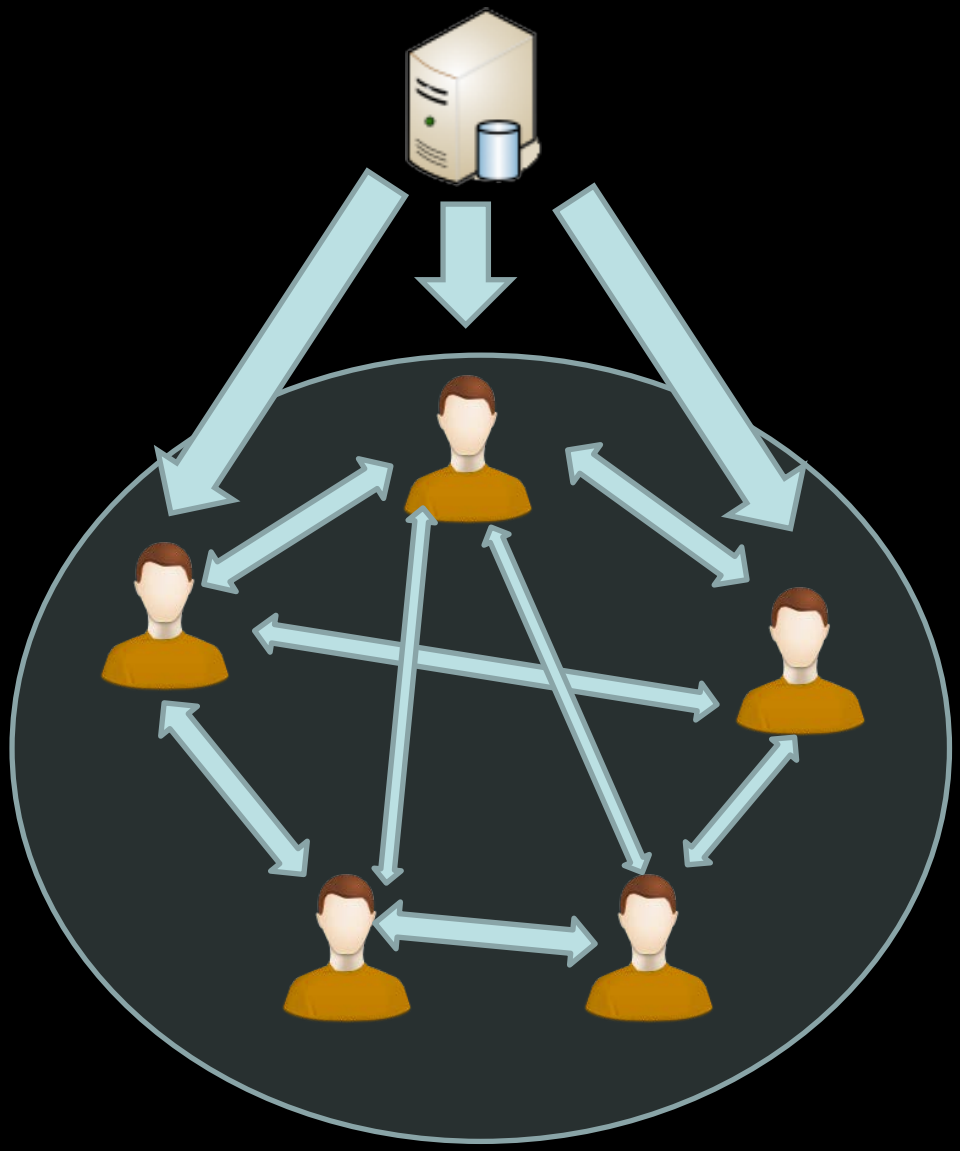
Ben Leong, Youming Wang, Su Wen, Cristina Carbunaru, Yong Meng Teo

National University of Singapore

Christopher Chang, Tracey Ho

California Institute of Technology

P2P File Distribution



File Distribution != P2P File Sharing

A set of clients download the file from server:

- Typically minimize Max. or Average Download Time with some minimal QoS.
- Nodes may leave when done

File Distribution Examples

1. Facebook/Twitter needs to update/synchronize the data on thousands of servers.

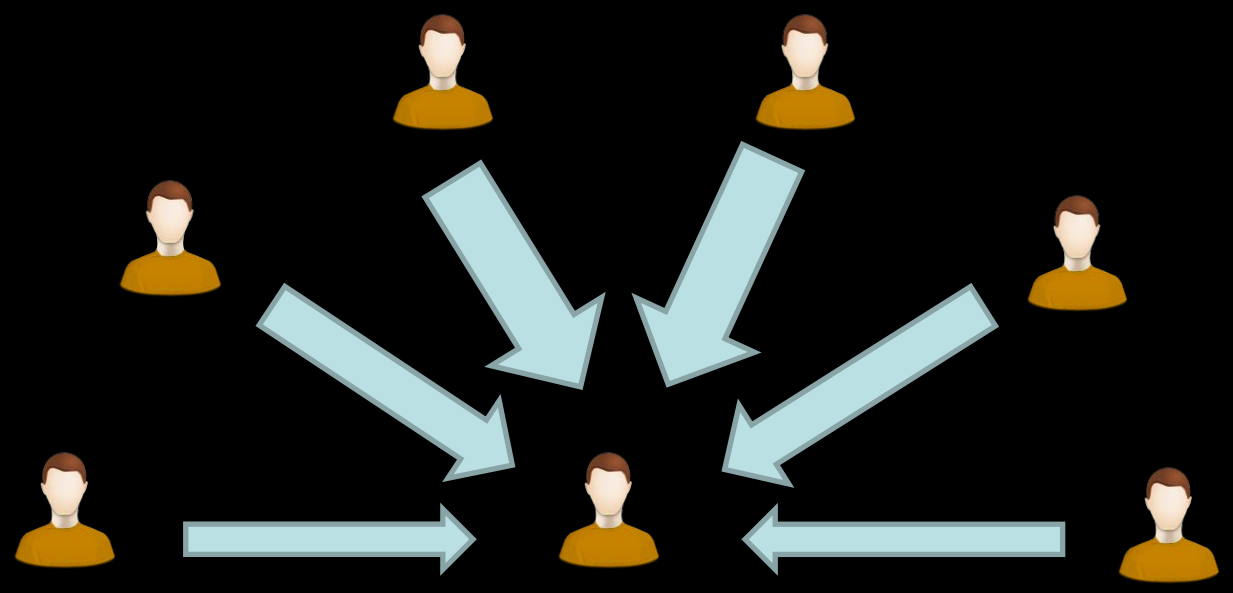
   Use BitTorrent

2. Microsoft issues server pack update to thousands of clients.

   Use Servers

BitTorrent

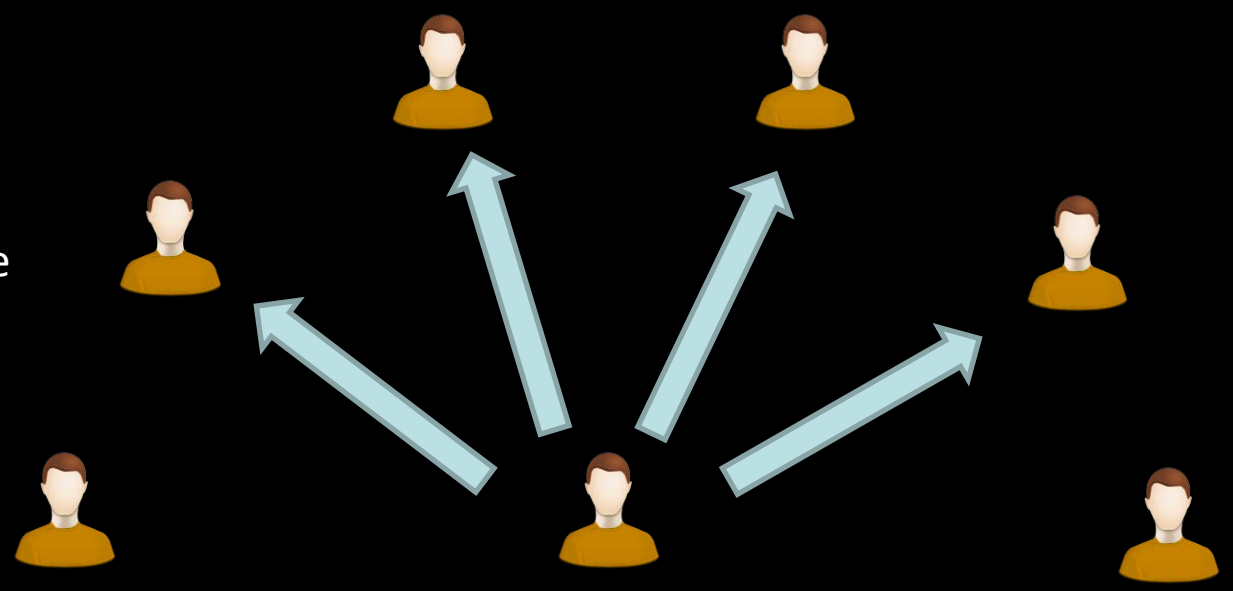
1. Data



Choke/Unchoke Mechanism

BitTorrent

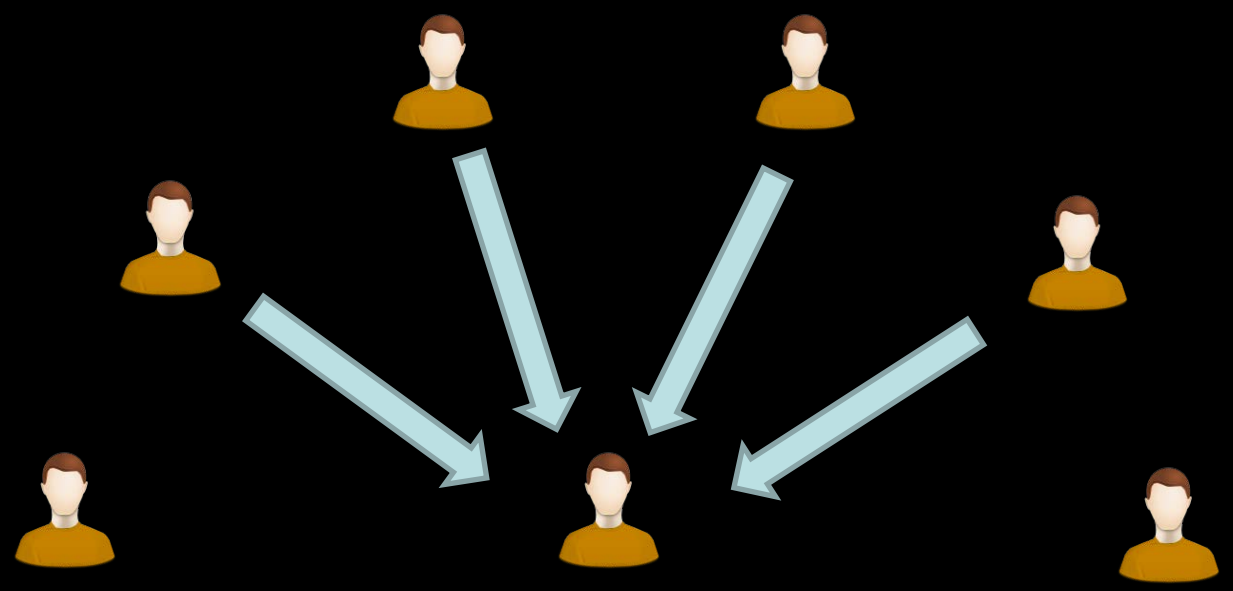
2. Unchoke



Choke/Unchoke Mechanism

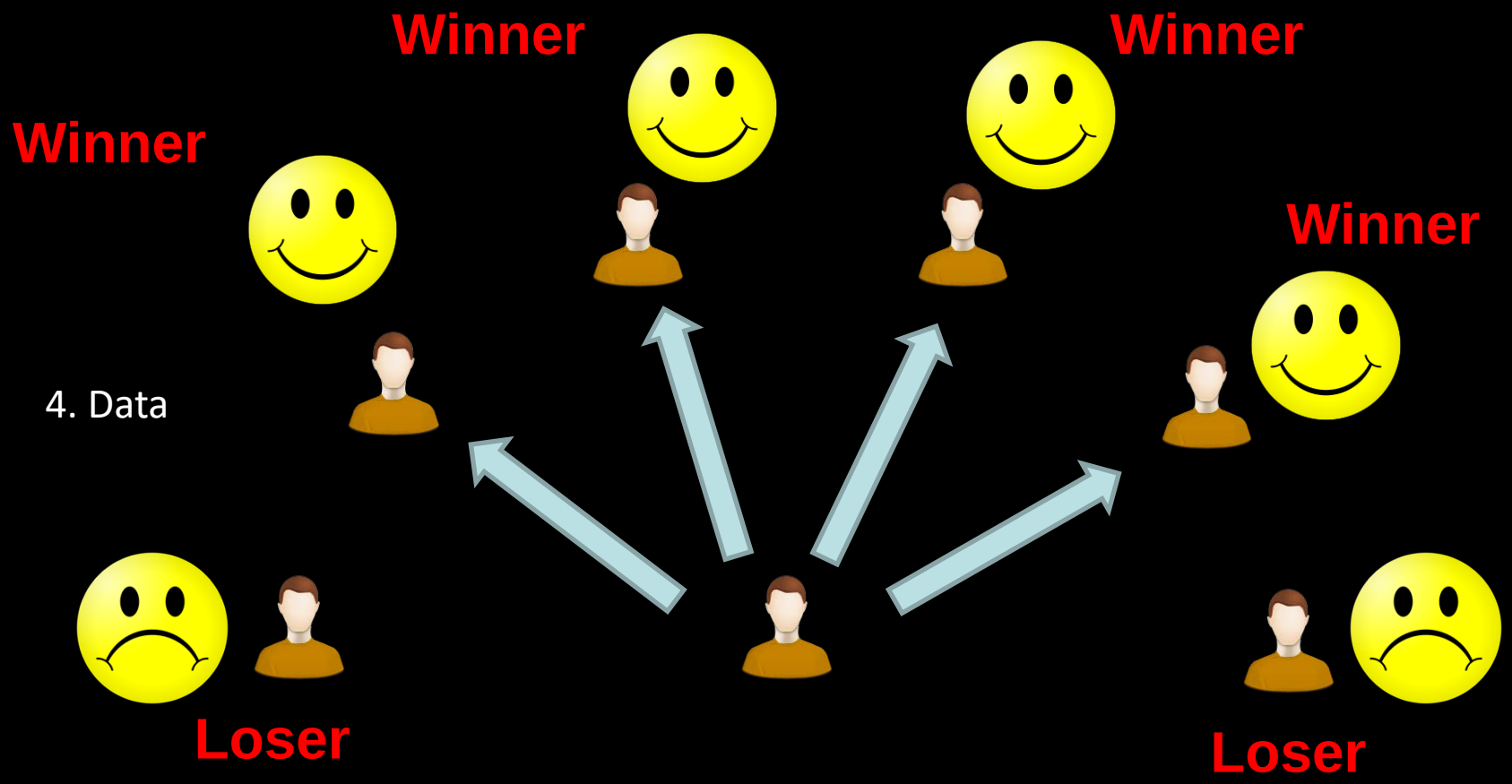
BitTorrent

3. Request



Choke/Unchoke Mechanism

BitTorrent



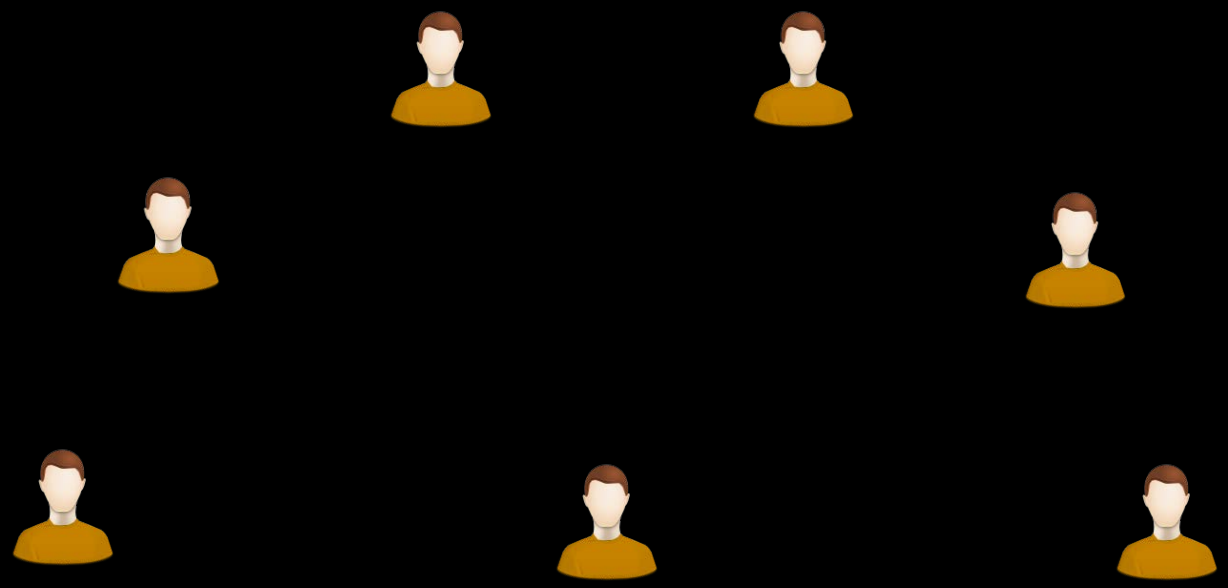
Choke/Unchoke Mechanism

Winner(s) Takes All

Auction!

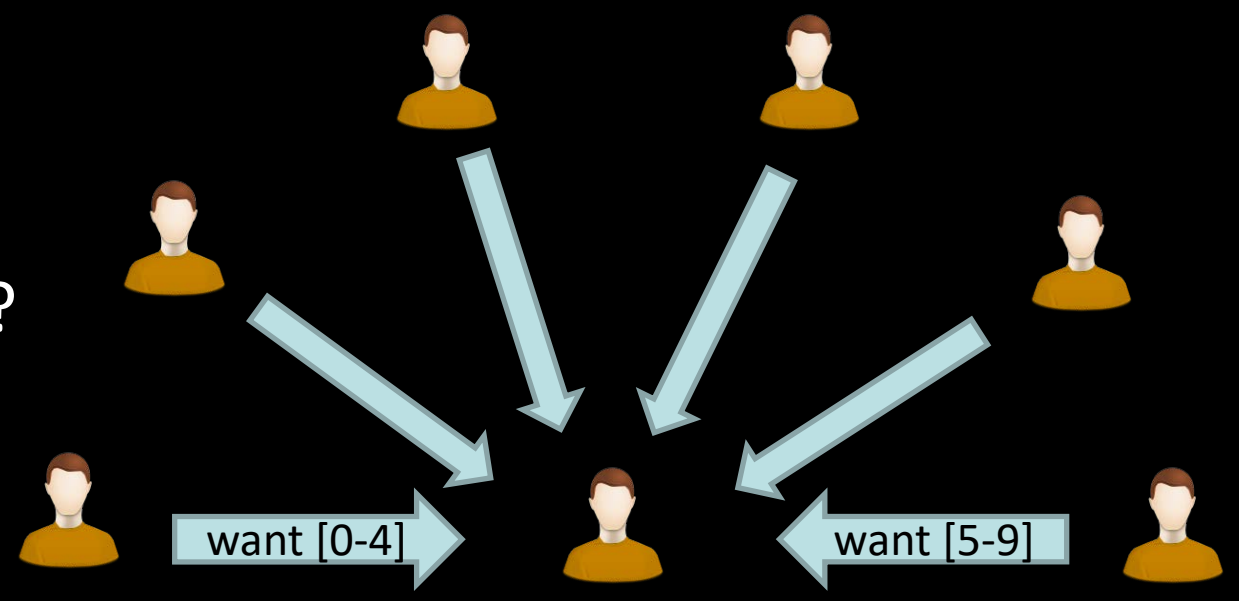
[Levin et al., SIGCOMM'08]

Is there a another way?



Is there a another way?

1. Trade?

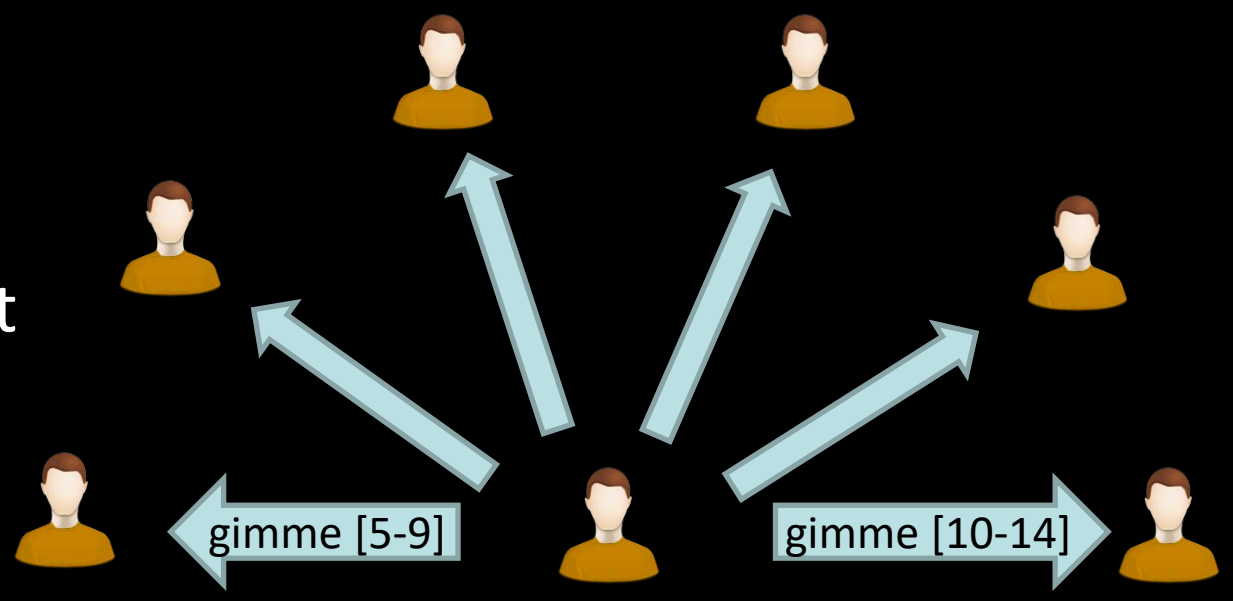


has [5-9]

has [0-4], not [5-9]

Is there a another way?

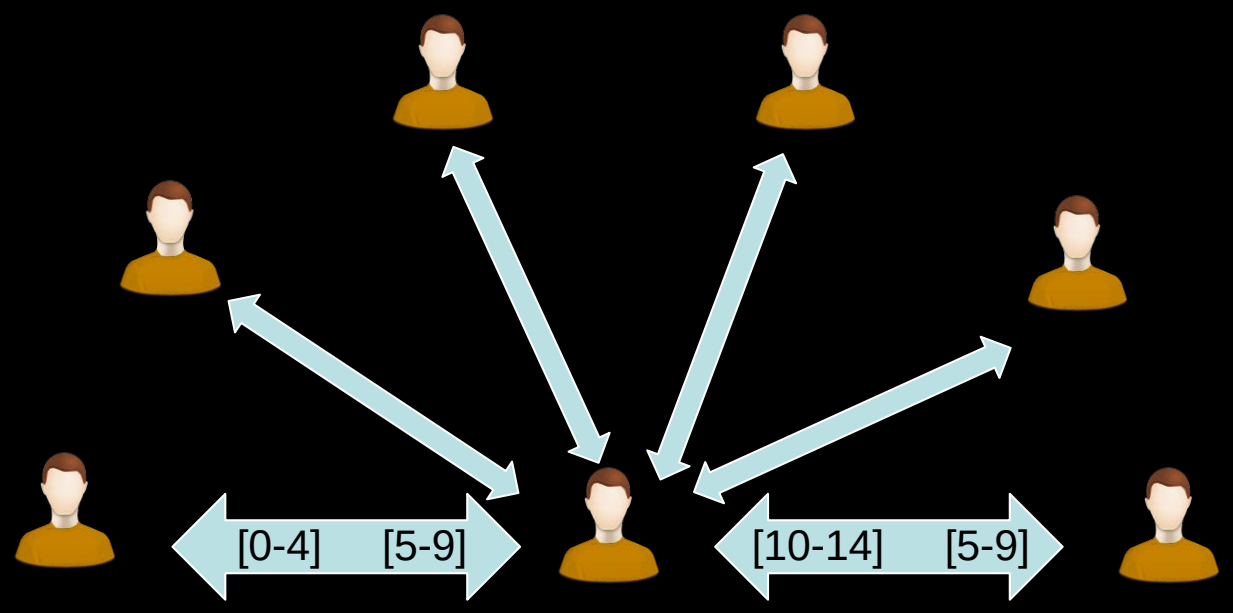
2. Accept



has [0-4], not [5-9]

Is there a another way?

3. Data

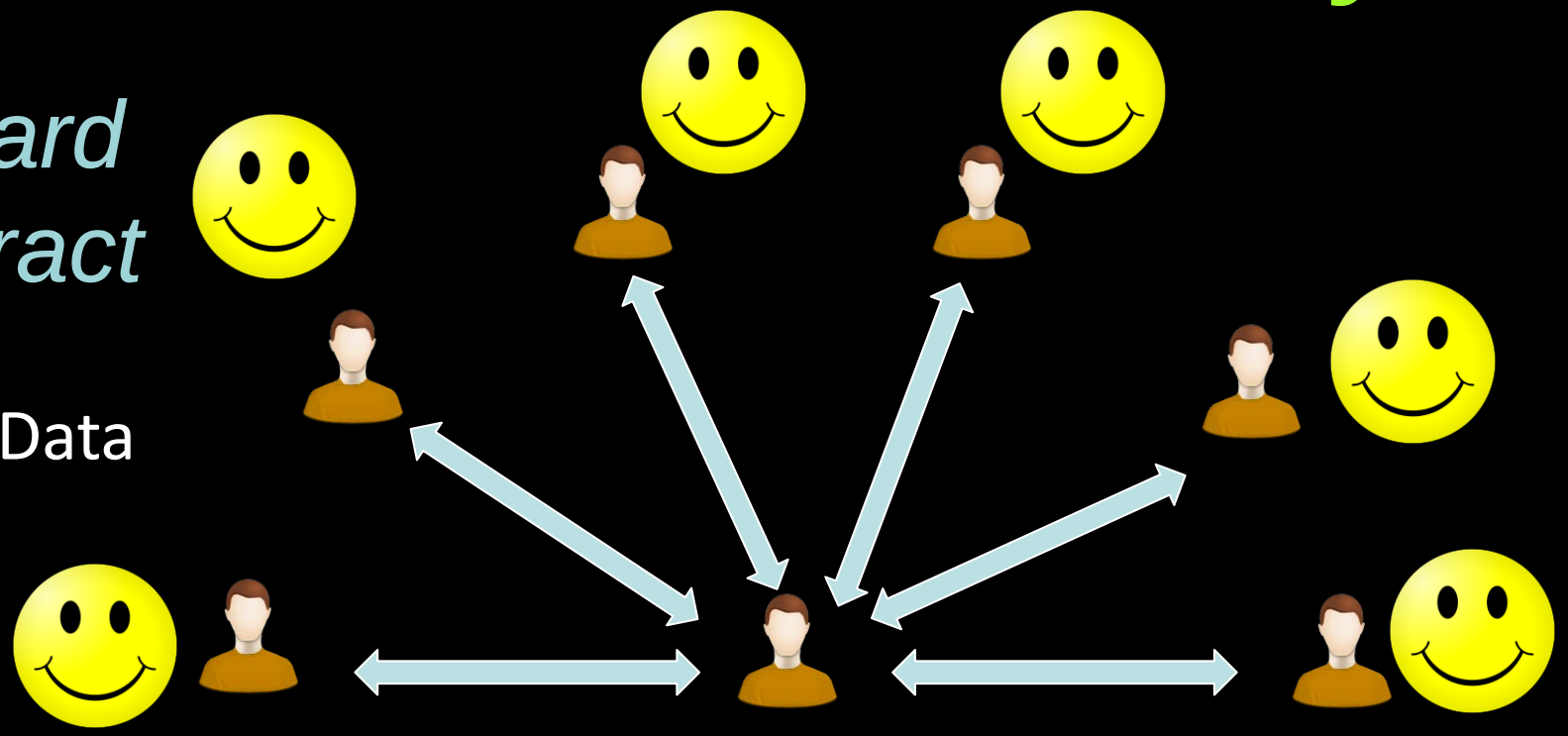


Block-for-block

Is there a another way?

Forward Contract

3. Data



Promise

Tit-For-Tat Transport Protocol (TFTTP)

Evaluation Result on EC2

Algorithm	Download Time (s)	Throughput (kB/s)
BitTorrent	2062	53
TFTTP	1571	70

100MB file, Server 300 kB/s.
24 Clients (8 nodes in each group):
50 kB/s, 100 kB/s, and 150 kB/s

Details in paper

Why?

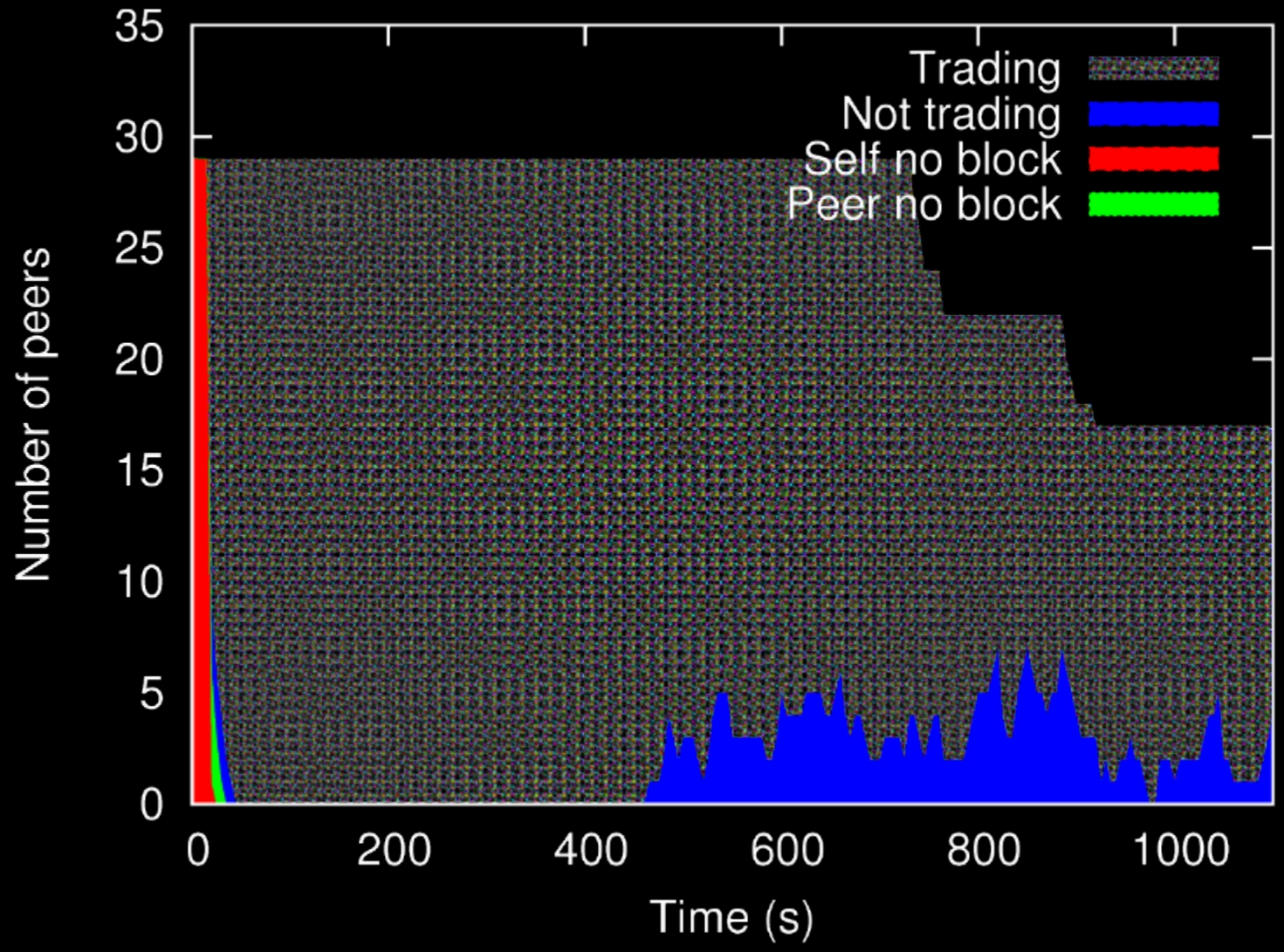
Some intuitions

Two Observations

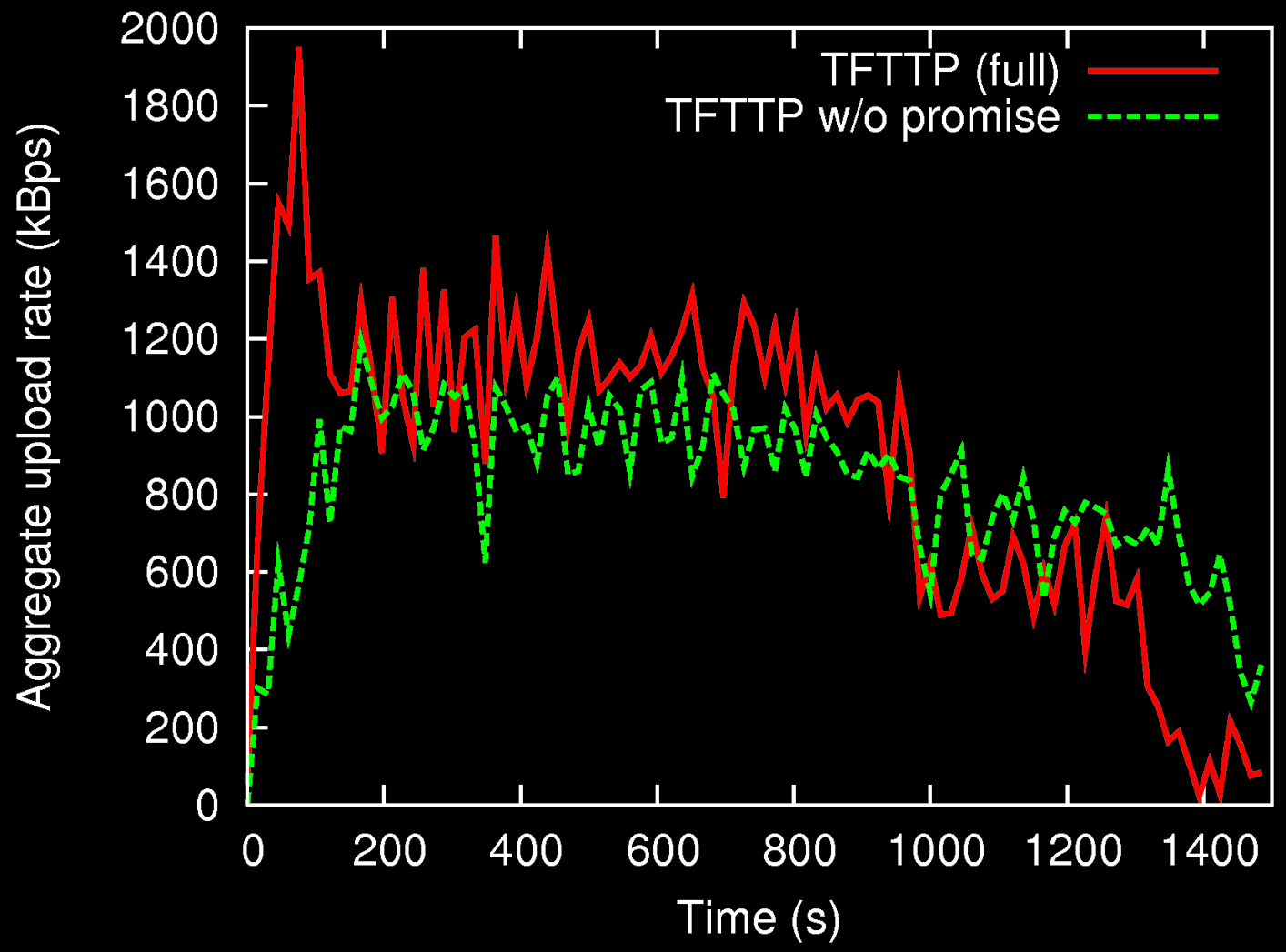
1. Availability – find blocks to download
2. Pipelining – fully utilize the upload bandwidth

1. Nodes can trade blocks they don’t already possess, but will soon

Availability



Availability



Significant Clustering in BitTorrent

[Legout et all, SIGMETRICS’07]

Not so for TFTTP

Details in paper

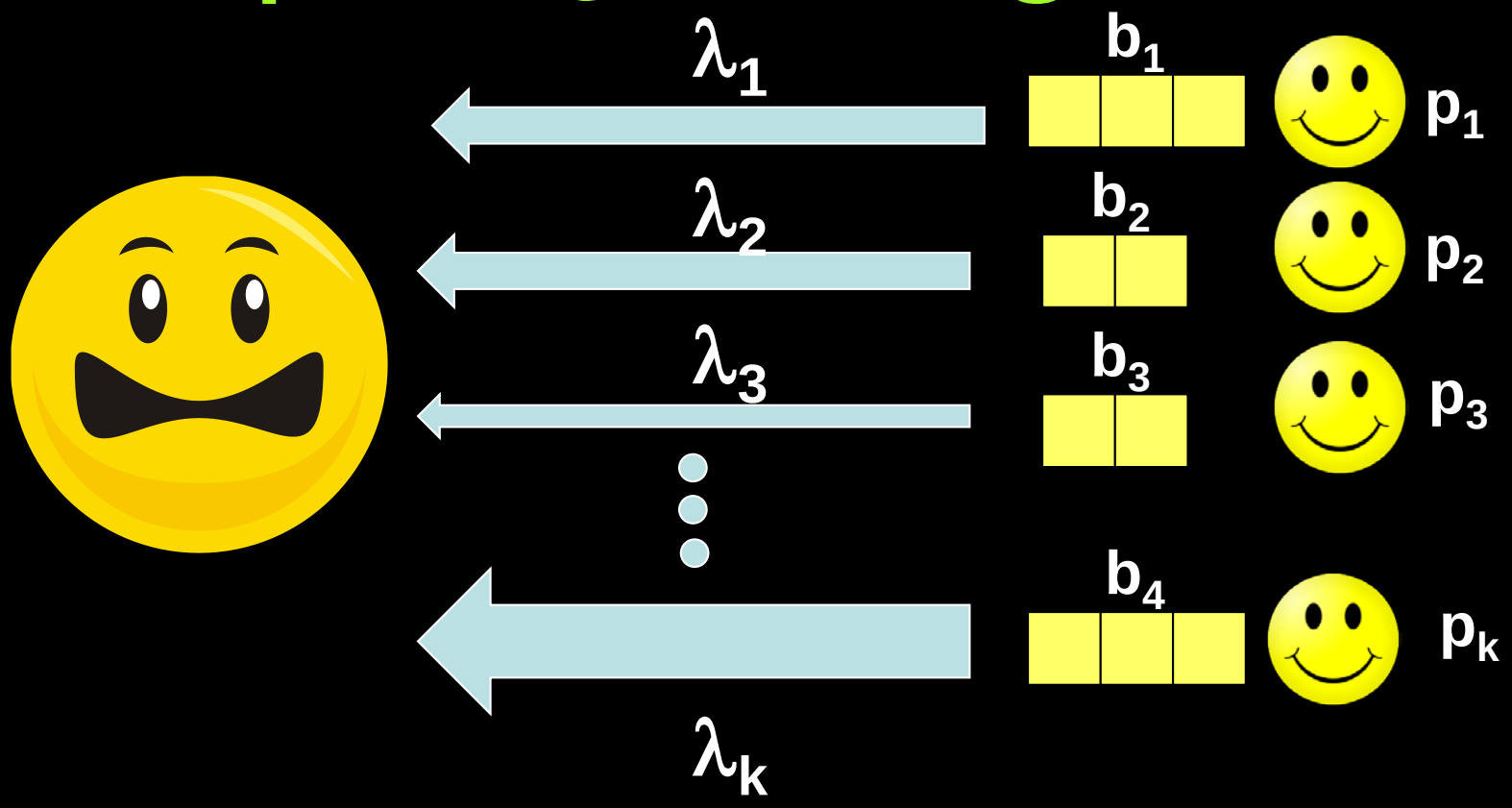
2. Not inherently bad for fast peers to trade with slower peers

Intuition

Upload bandwidth is the <u>key</u> limiting factor

$\Rightarrow$ Nodes should maximize use of upload bandwidth

Simple Queuing Model



$$\frac{b_1}{\lambda_1} = \frac{b_2}{\lambda_2} = \frac{b_3}{\lambda_3} = \ldots = \frac{b_k}{\lambda_k}$$

Promises are Self-Clocking

Promises are Self-Clocking

Promises are Self-Clocking

Promises are Self-Clocking

Promises are Self-Clocking

New trades proposed only when an existing trade is completed.

Promises are Self-Clocking

Promises allow “multiplexing” of data transfers over time

Promises removes uncertainly associated with choke/unchoke – allows better pipelining

Future Work

- Open system
- Altruism
- Smarter server
- Multiple geographically distributed servers


For file distribution
.....

Maybe … BitTorrent isn’t quite the right answer ☺

QUESTIONS

How often do nodes lose?

From	Bid to slow nodes	Bid to medium nodes	Bid to fast nodes	Bid to all nodes
Slow	20.75% (281/1355)	61.64% (586/950)	77.22% (456/590)	45.68% (1323/2896)
Medium	29.08% (197/677)	26.59% (319/1200)	33.46% (373/1113)	29.71% (888/2990)
Fast	27.09% (128/472)	19.72% (232/1177)	14.28% (178/1246)	18.58% (538/2895)

100 MB file for 3 groups of peers (64KB/s, 128KB/s, 196KB/s) before the first node is done

But it's not too bad…

From	To Slow	To Medium	To Fast	To All
Slow	138.42	79.21	56.01	273.64
Medium	88.23	217.83	240.71	546.76
Fast	73.25	273.7	412.42	759.37
Server	17.24	36.82	70.36	124.42
All	317.14	607.56	779.49	1704.19

Data transferred in MB until first node is done.