Analyzing and Improving a BitTorrent Network's Performance Mechanisms

INFOCOM2006

Jeng-Long Chiang

MNET Lab Meeting

June 14, 2006

Introduction

- This paper presents a **simulation-based** study of BitTorrent with a goal to deconstruct the system and evaluate the impact of its **core mechanisms**, both individually and in combination, on overall system performance in terms of **peer link utilization**, file download time, and **fairness** under a variety of workloads.

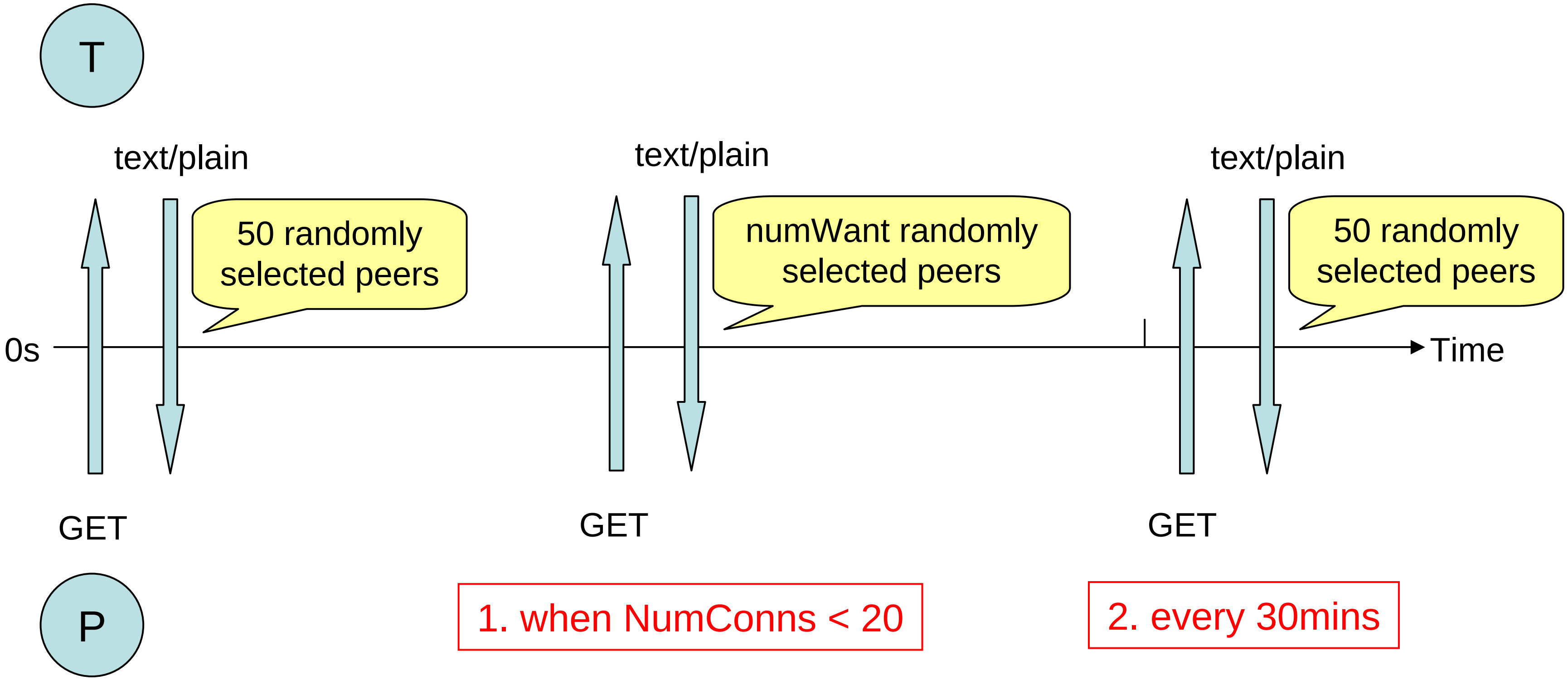
The Unknowns

- Could BitTorrent have achieved even higher bandwidth utilization in this setting?
- Does BitTorrent’s Local Rarest First (LRF) effectively avoid the last block problem?
- How effective is BitTorrent’s tit-for-tat policy in avoiding unfairness?
- If nodes depart as soon as they finish, is the stability or scalability hurt significantly?
- …

BT Overview

- Block v.s. subblock (piece v.s. subpiece)
- Tracker
- Seed v.s. leecher
- Neighbor (peer set)
- Local rarest first (LRF)
- Tit-for-tat (TFT)
- Choke v.s. optimistic unchoke

Client <-> Tracker



T

text/plain

50 randomly selected peers

text/plain

numWant randomly selected peers

text/plain

50 randomly selected peers

0s

Time

GET

GET

GET

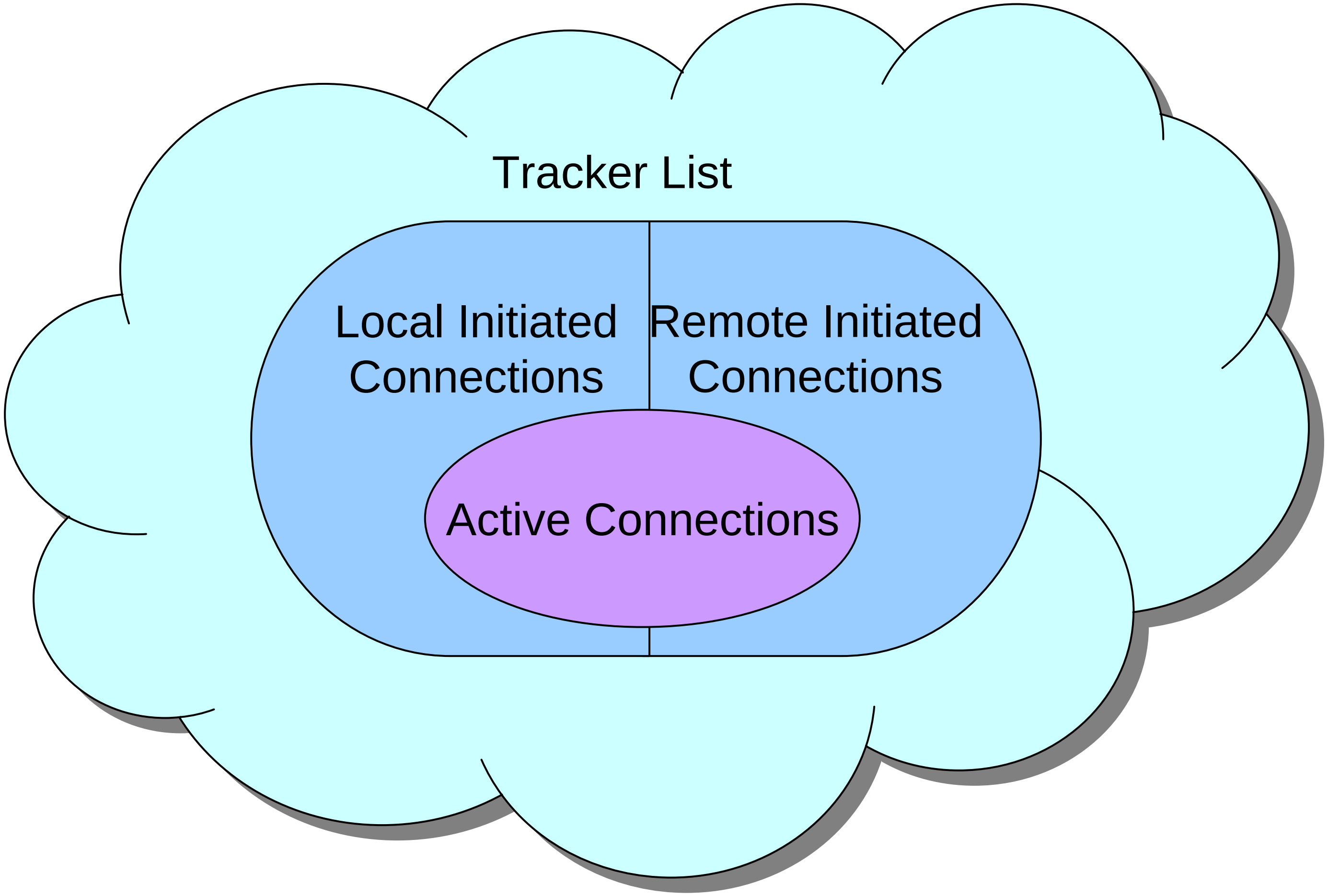
P

1. when NumConns < 20

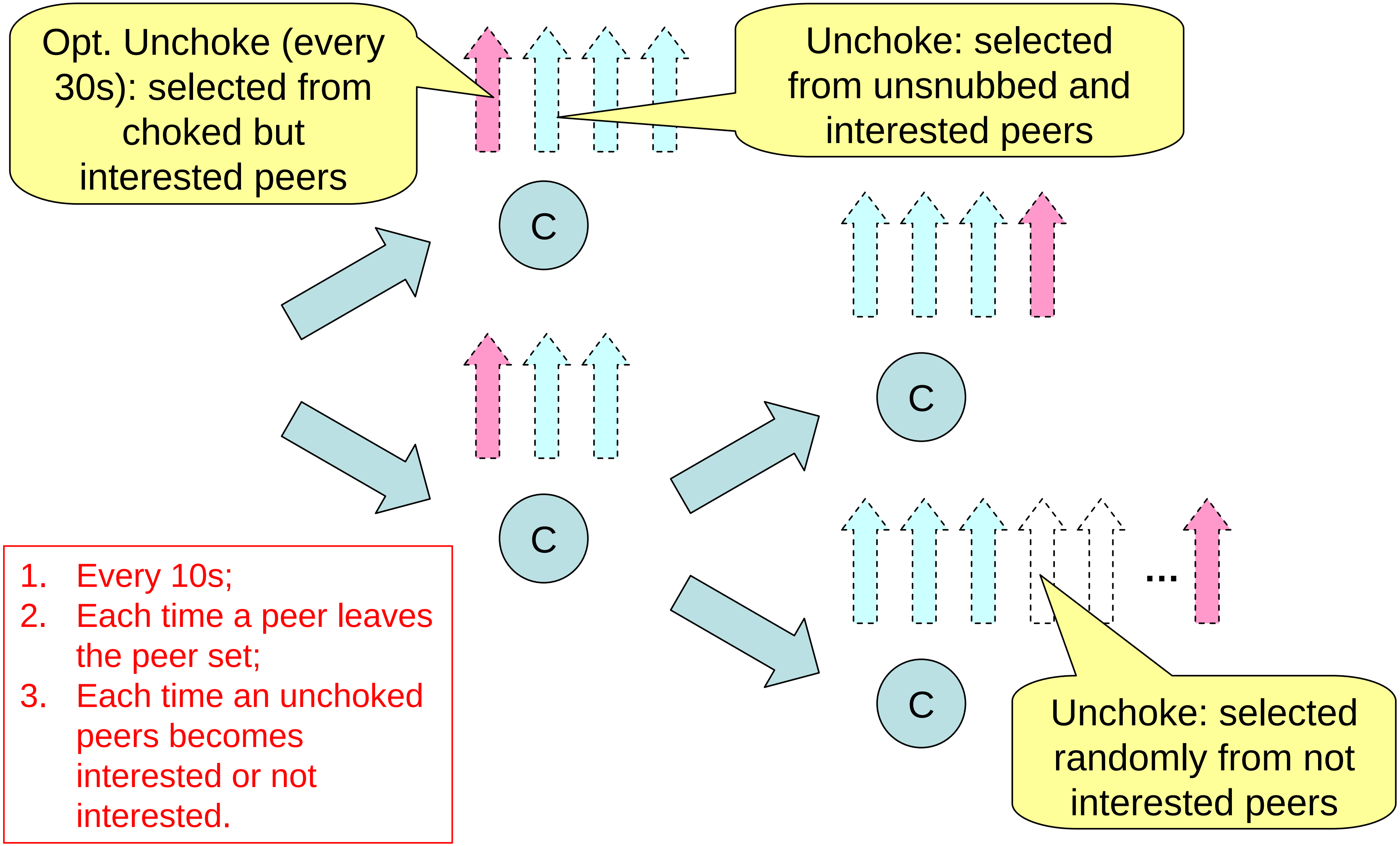
2. every 30mins

InitNumConns=40
maxNumConns=80

Levels of Peer Sets



Choke Algorithm (Leecher)



Opt. Unchoke (every 30s): selected from choked but interested peers

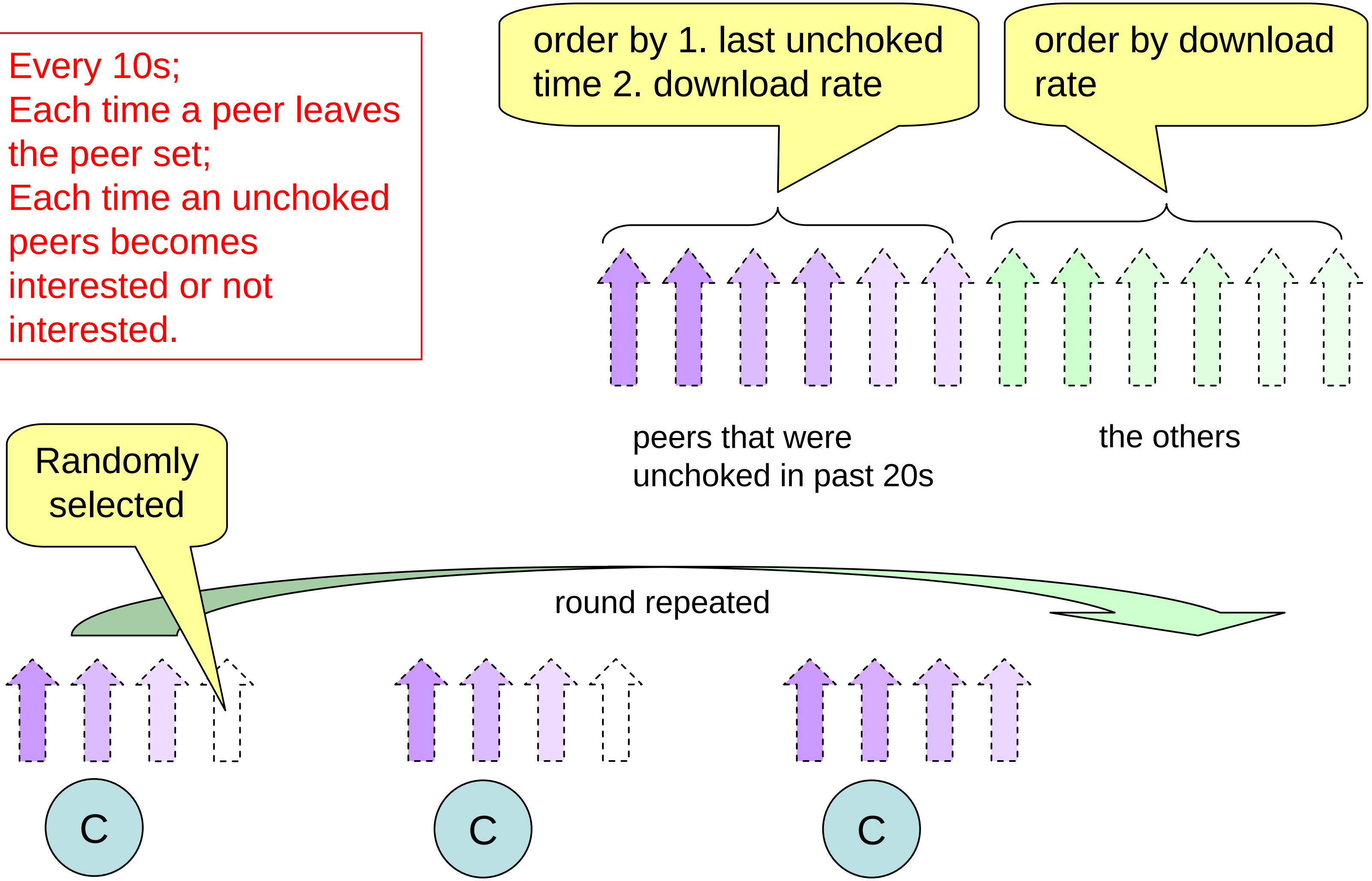
Unchoke: selected from unsnubbed and interested peers

Unchoke: selected randomly from not interested peers

1. Every 10s;
2. Each time a peer leaves the peer set;
3. Each time an unchoked peers becomes interested or not interested.

Choke Algorithm (Seed)

1. Every 10s;
2. Each time a peer leaves the peer set;
3. Each time an unchoked peers becomes interested or not interested.



order by 1. last unchoked time 2. download rate

order by download rate

peers that were unchoked in past 20s

the others

Randomly selected

round repeated

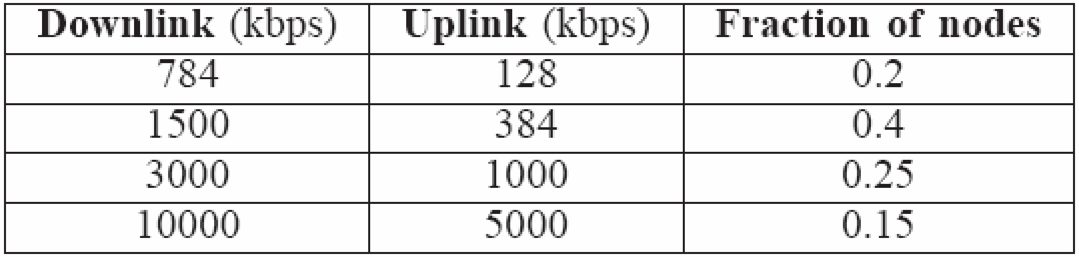
Performance Metrics

- The system is said to be optimal if it has optimal utilization as well as complete fairness.

- Link utilization
  - The ratio of the actual flow to the maximum possible
  - (also for link download time)
- Fairness
  - The number of blocks uploaded divided by the number of blocks in the file
  - (also for seed’s load)

A Real Workload

- 200MB file with a block size 256KB



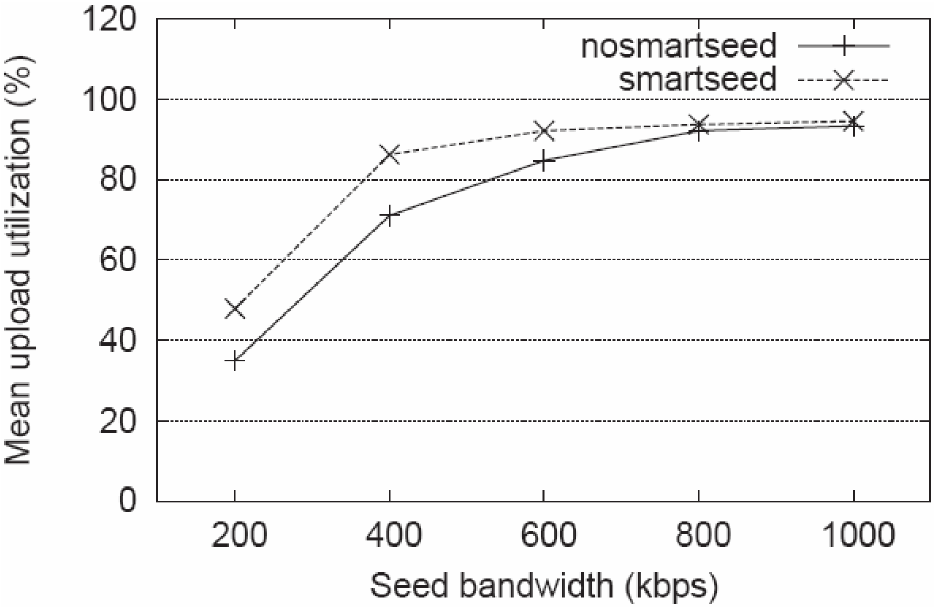
Downlink (kbps)	Uplink (kbps)	Fraction of nodes
784	128	0.2
1500	384	0.4
3000	1000	0.25
10000	5000	0.15

Seed’s uplink: 6000 Kbps

- The simulation result of the second day of the flash crowd (10000 arrivals, 300 simultaneities)
  - Uplink utilization: 91% -> high link utilization is achieved
  - Seed’s load: 127 -> seed’s bandwidth is precious
  - Unfairness: worst client load=6.26 -> unfairness

Smart Seed

- The seed does not choke a leecher unless it serves a complete block.
- The seed always serves the block that it has served the least.

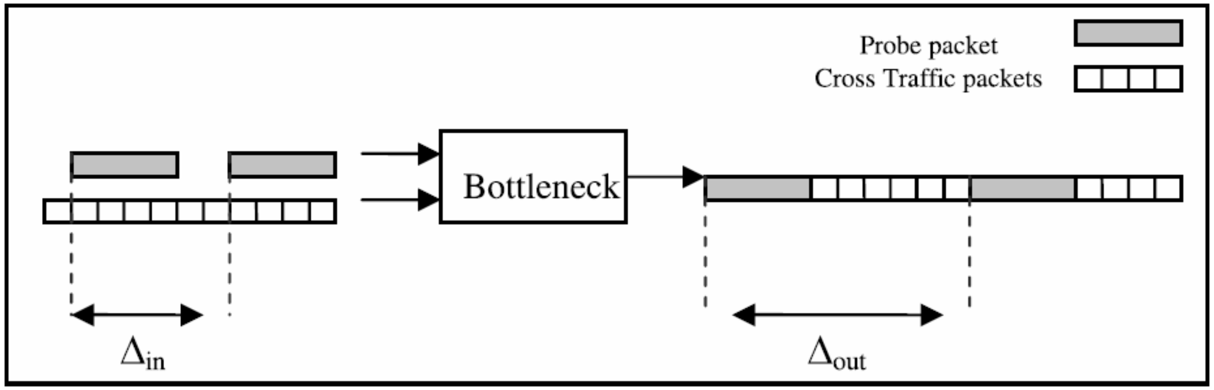


Unfairness

- BT's **optimistic unchoking** significantly increases the chance that a high bandwidth node unchokes and transfers data to nodes with poorer connectivity.
  - It leads to decrease in uplink utilization due to download bottleneck on the target side.
  - It results in the high bandwidth node serving a larger volume of data than it receives in return.

- Replacing optimistic unchoking
  - Quick bandwidth estimation
  - Pairwise block-level TFT
  - Bandwidth-matching tracker

Quick Bandwidth Estimation [18]



$$A = C \times \left(1 - \frac{\Delta_{out} - \Delta_{in}}{\Delta_{in}}\right)$$

A: Available bandwidth
C: the capacity of the bottleneck

Pairwise Block-Level TFT

- Enforcing fairness directly in terms of blocks transferred rather than depending on rate-based TFT.
- A peer x allows to upload a block to y iff

$$Uxy \leq Dxy + \Delta$$

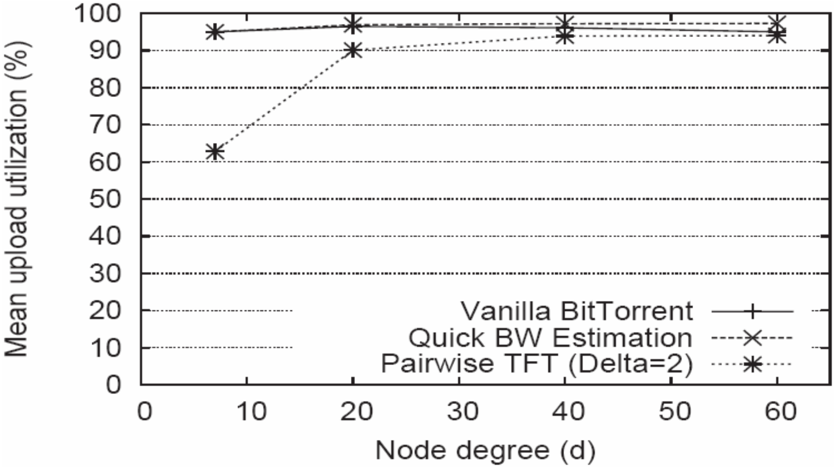
Uxy: the amount that x has uploaded to y
Dxy: the amount that x has downloaded from y
Δ: the unfairness threshold

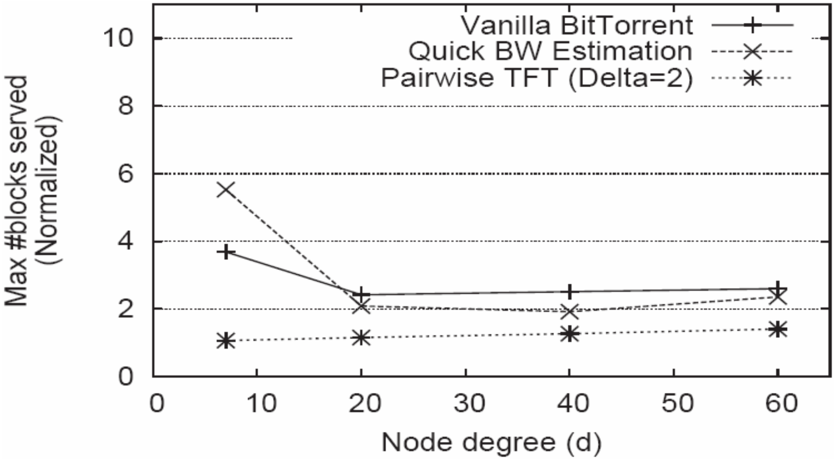
Bandwidth-Matching Tracker

- The tracker returns to a new node a set of candidate neighbors with similar bandwidth to the new node.
- A hybrid policy is employed to avoid groups of nodes being disconnected from the rest of the network.
  - 50% bandwidth-matched
  - 50% random

Performance Evaluation

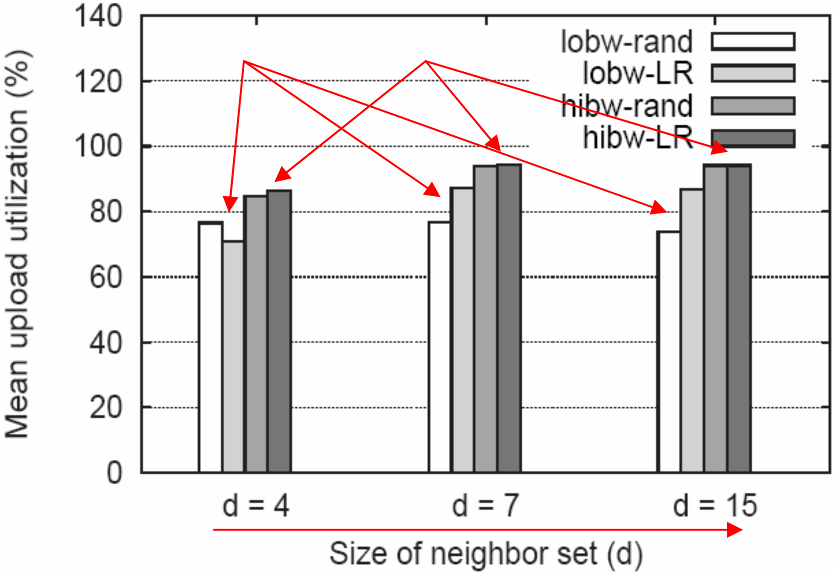
- A flash crowd of 1000 node within 10s
- Node BW: 6000/3000, 1500/400, 784/128 Kbps
- Smart seed: 800-6000 Kbps (?)

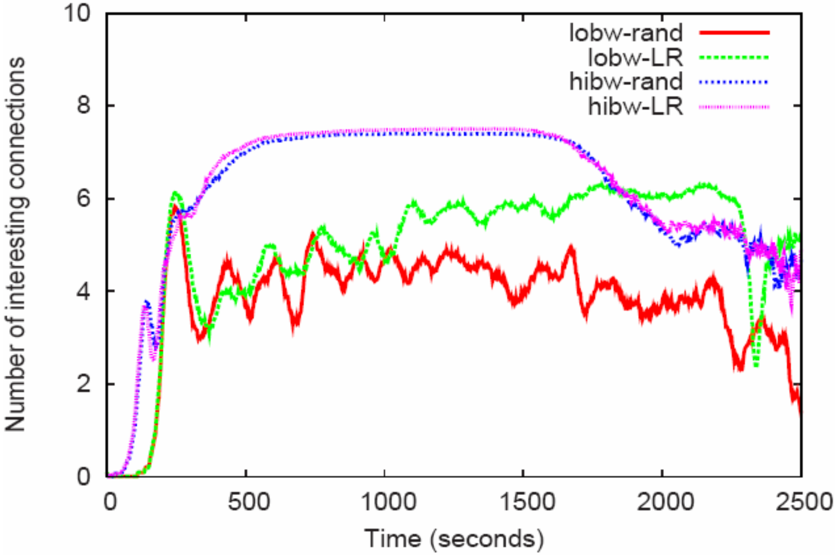




More Issues

- Block choosing policy
  - Random vs. LRF
  - Seed bandwidth: low vs. high (400 vs. 6000 Kbps)
  - Node degree (d): 4, <u>7</u>, 15





Main Findings

- BT is remarkably robust and scalable at ensuring **high uplink bandwidth utilization**.
- BT **scales well** as the number of nodes increases, keeping the load on the original server bounded (127/10000).
- The LRF policy **performs better** than the random policy.
- The bandwidth of the origin server (**seed**) is a precious resource.

Future Issues

- BT’s rate-based TFT do not prevent **unfairness** in terms of the data served by nodes.
- BT is **not effective** at allowing nodes who have most of a file to rapidly find the few blocks that they are missing.
- **Network coding** may be the final solution for the last block problem.

Discussions

- Fairness?
  - BT is about resource sharing, not trading.
  - It might be critical for users that pay for connection time or uploaded bits (e.g. 3G/GPRS)
  - Rate-based vs. pairwise-block-based
    - Integration
    - Adaptive Δ