http://oasis.cs.berkeley.edu/retreats/jun2005/

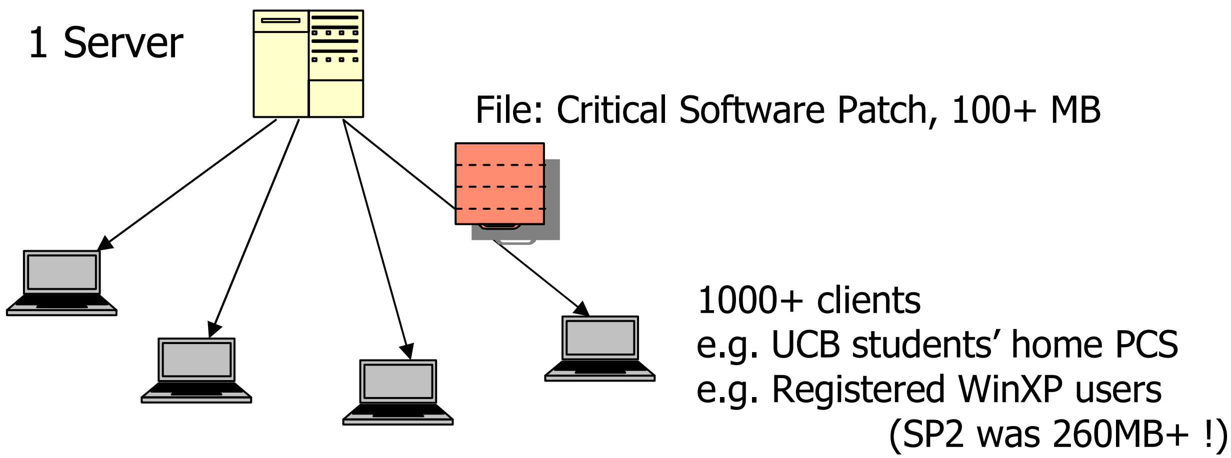
On Cooperative Content Distribution and the Price of Barter

Mukund Seshadri
(mukunds@cs.berkeley.edu)

Prasanna Ganesan
prasannag@cs.stanford.edu

Prof. Randy Katz
randy@cs.berkeley.edu

Motivating Scenario



Objective: minimize time by which *all* clients have received the file.

-- i.e., Completion Time --

Environment: upload-bandwidth constrained

Key: use client upload capacities

Secondary Applications

- New OS releases
  - E.g. Red-hat ISOs (1.5GB+) on the first day of release
- Independent publishing of large files+flash-crowds
  - Handle the “slashdot” effect.
- Commercial (legal) video distribution
  - Download TV shows quickly!

Completion Time less critical for these applications.

Intelligent use of client upload capacities still required.

1 -> Many Distribution: Background

- *d*-ary tree multicast [1,2]
  - Inefficiency: Leaf upload capacities unused
  - Target: client reception rate, in-order delivery
- Parallel trees [3,4]
  - Inefficiency: Client upload capacity growth sub-optimal
  - Target: load-balance, fairness
- BitTorrent [bittorrent.com]
  - Unstructured P2P solution: randomly built overlay graphs
  - Target: per-client download time, incentivizing cooperation

No method is targeted to optimize completion time
Completion Time of these algorithms not well-understood

[1] Chu et al. "A Case for End-System Multicast" SIGMETRICS'00. [2] Jannotti et al."Overcast…" OSDI'00
[3] Karger et al. "Consistent Hashing and Random Trees…" STOC'97. [4] Castro et al."Splitstream…" SOSP'03

Goals and Assumptions

Optimality – define, design, compare.

Two scenarios of client behaviour

- Cooperative
  - Clients freely upload data to each other
- Non-Cooperative
  - Clients need incentive to upload data to other clients

Assumptions:

- Upload-constrained system
- Homogenous nodes
- Static client set
- No node or network failures

Outline of Research

Cooperative Clients

- Analysis of synchronous model, parameterized by no. of clients and blocks.
- Optimal completion time algorithm designed for arbitrary number of clients and blocks
  - Prior work[5,6] achieved this only in special cases or with high complexity
- Simpler randomized variants proposed, evaluated by simulation
- Comparison of completion times of prior work
  - (Simulations for BitTorrent)

[5] Yang et al. "Service Capacity of peer-to-peer Networks" INFOCOM'04.
[6] Bar-Noy et al. "Optimal Multiple Message Broadcasting…" Discrete Applied Math. '00 No.100, Vol 1-2.

Outline of Research (Part 2)

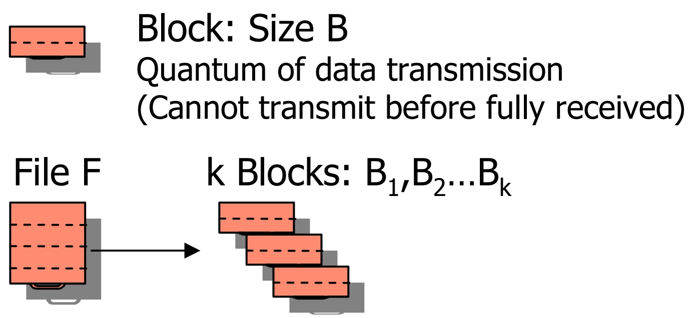
Non-Cooperative Clients

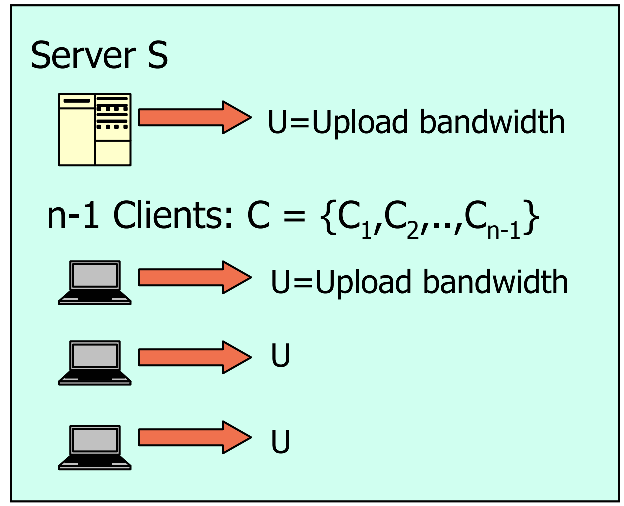
- Requirement: fast, simple, decentralized algorithms.
- Developed several models of distribution based on ***barter***.
  - Based on "credit limits"
- Analyzed completion time in several special cases
  - And found the optimal algorithm for these cases.
- Evaluated randomized variants, by simulation
  - Investigated impact of several parameters
    - Low overlay graph degree and low completion time can be achieved, using Rarest-first block-selection policy

Scenario: Cooperative Clients

(in detail…)

Cooperative Distribution - Model



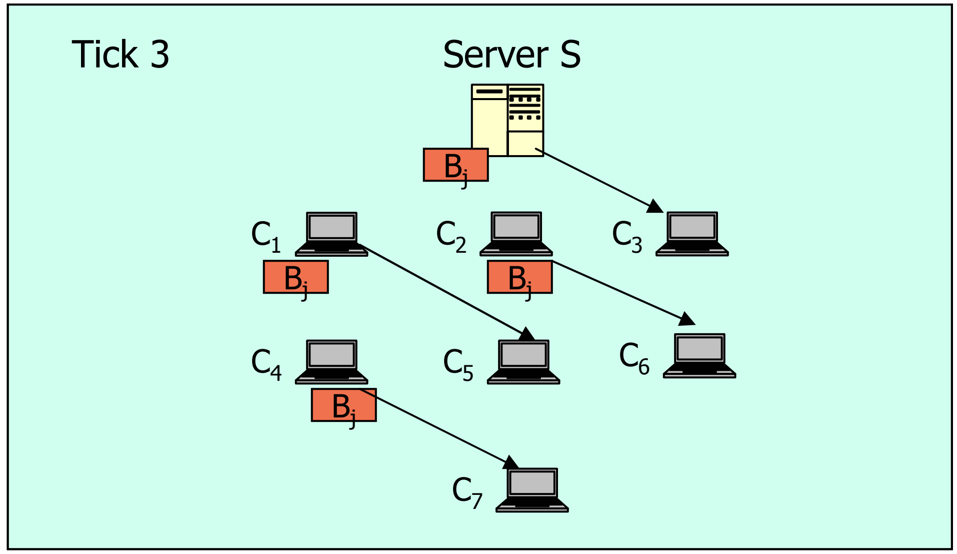


- T(k,n) = time taken for ***all*** clients to receive ***all*** blocks.
  - Time unit: ***Tick*** =B/U.

To find: the lowest possible value of T(k,n);
and the algorithm that achieves this value.

Lower bound

e.g. 1 block, 7 nodes:
"Binomial Tree" is optimal



Observations:
- K blocks take at least k ticks to leave server.
- Last block takes another $\lceil \log_2 n \rceil$ -1

Lower bound for T(k,n): $k + \lceil \log_2 n \rceil - 1$ (ticks)

Well known solutions

Completion times T(k,n) for…

- Multicast tree of degree d: ~ $d(k+\lceil \log_d n \rceil -2)$
- Splitstream with d parallel trees: ~ $k+d \log_d n$
- Linear pipeline: k+n-1
- Server serves each client: kn

All of the above are sub-optimal:
Compare with: $k+\lceil \log_2 n \rceil -1$ (ticks)

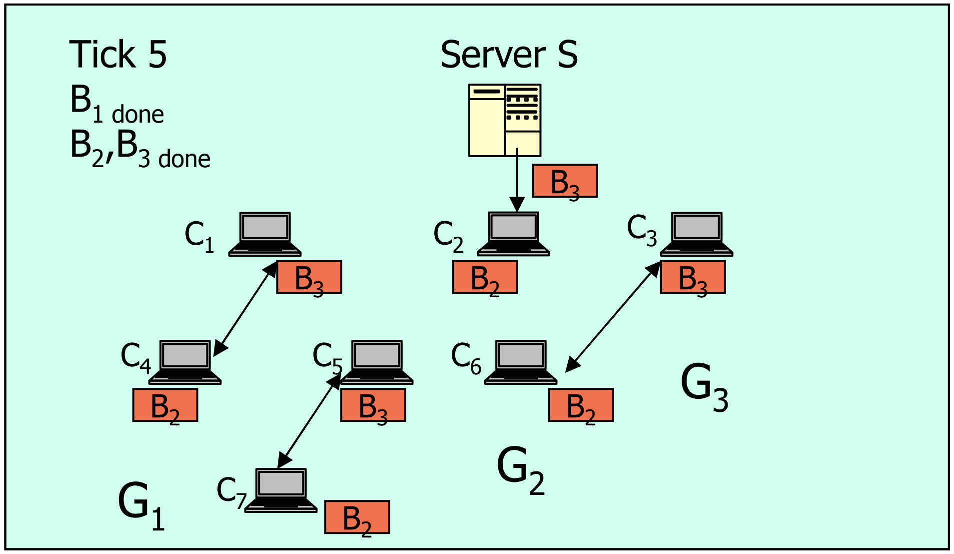
Towards an Optimal Algorithm

Challenge

- Disseminate each block as fast as possbile: binomial tree.
- For k blocks? – need to schedule across blocks.
  - Ensure maximal *growth* and *utilization* of client upload capacity

Binomial Pipeline ($n=2^L$) [5]

- Opening phase of L ticks
  - nodes in L groups: G_i has 2^{L-i} nodes.
- Middle phase: in $(L+j)^{th}$ tick
  - no. of clients without B_j equals no. of clients with B_j minus-1. So match and swap!
  - Server transmits to unmatched client.
- End: server keeps sending B_k



[5] Yang et al. "Service Capacity of peer-to-peer Networks" INFOCOM'04. : discusses a version of this algorithm for n=power-of-2

Optimal Algorithm

- The binomial pipeline is optimal
  - A new block leaves the server every tick (1st k ticks)
  - Every block doubles in presence every tick
- Matching scheme left unspecified

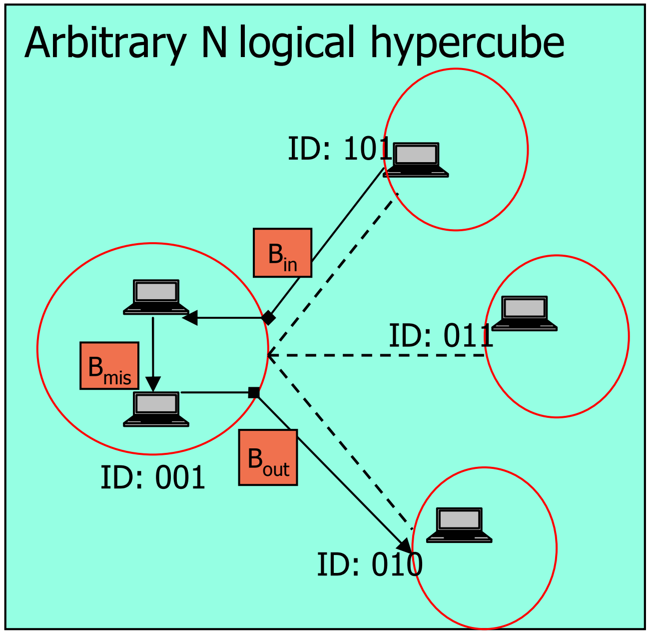
Our solution: Hypercube Algorithm

- Hypercube overlay graph of clients and server
  - Each client has an L-bit ID, and S has 0 ID.
- $N_i(C_m)$ is C_m's neighbour on the hypercube
  - the node whose ID differs from C_m in the (i+1)st most significant bit.
- Nodes transmit to clients in round-robin order
  - At time t, C_m uploads a block it has, to $N_{(t \bmod L)}(C_m)$.
- The highest-indexed block is always transmitted
  - S uploads B_t to $N_{t \bmod L}(S)$, or B_k if t>k.
- This finishes in $k+\lceil \log_2 n \rceil -1$ ticks.

Arbitrary *n*

- Use a hypercube of ***logical*** nodes
  - Logical node can have 1 or 2 physical nodes
  - Dimension of hypercube = L = Floor($\log_2 n$)

- At most one block mismatch *within* a logical node
- This finishes in $k + \lceil \log n \rceil - 1$ ticks



Our optimal algorithm design is complete

Towards Easier Implementation

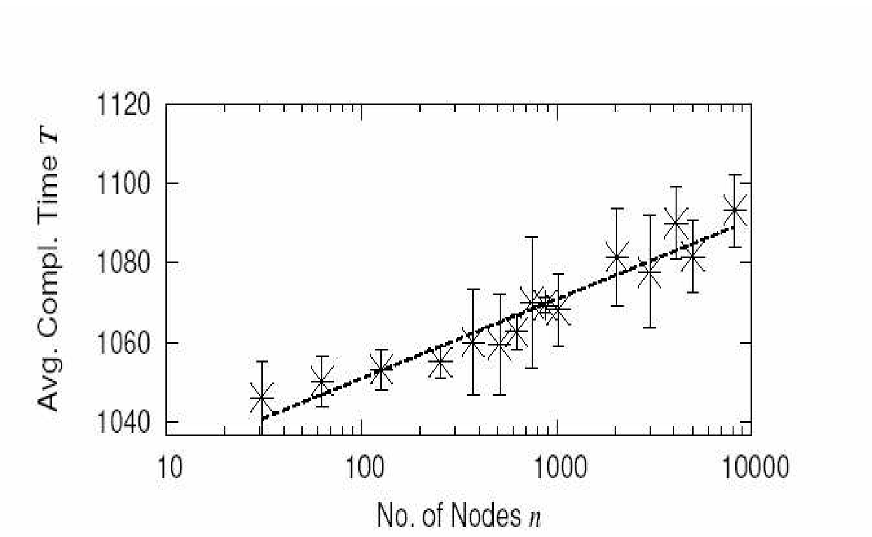
- Hypercube algorithm requires rigid communication pattern
- Key operation: optimal mapping of nodes that need a block to nodes that have that block,
  - to ensure maximal utilization of client upload capacity
- Can we do this mapping *randomly?*

  - Nodes form an overlay graph of given type (can be random) and degree.
  - Each node X finds a random neighbour Y that requires a block B that X has.
  - X uses a handshake with Y to ensure download capacity and resolve redundant block transmissions.
  - X sends block B to Y
  - Y notifies all its neighbours that it now has B.
  - Repeat…

- What is the impact on completion time?
  - We estimate this via simulations

Randomized Algorithm Simulations

- Synchronous simulations
  - Metric: completion time T (k,n)
  - Constant B; T in ticks(=B/U).
  - Overall range: k~10-10000, n~10-10000

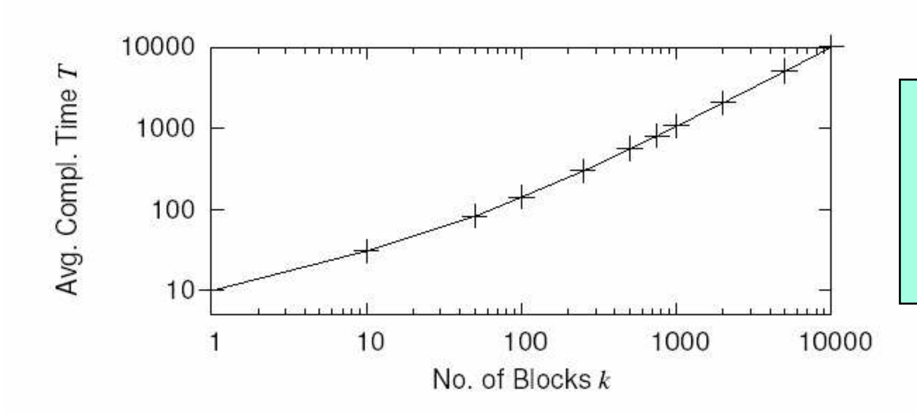


T vs. n, with fixed k=1000 (note log-scale X-axis)

T exhibits a linear dependence on $\log_2 n$

Results

T vs. n, with fixed n=1024 (note log-scale on *both* axes)



T exhibits a linear dependence on k

Over the entire range of k=10-10000 and n=10-10000:

Least squares estimate of T(k,n)~ 1.01k+4.4log n+3.2

Randomized algorithm likely to be close to optimal in normal operating ranges ($k>>\log_2 n$)

BitTorrent comparison

- Asynchronous simulator modeling client/client messages in BitTorrent spec.
  - Assumed k blocks and n nodes (all arriving near time 0)
  - Metric: completion time T (of *all* nodes)
  - Varied k and n from 10-2000
- Least-squares estimate of $T(k,n) \sim 2.2k + 47\log_2 n - 173$.
  - With default parameters
- This can be improved to $1.3k + 9.8\log_2 n - 9$
  - By tuning parameters: decreasing frequency of peer prioritization decisions, and number of simultaneous uploads.

BitTorrent can be 2.2x worse than optimal (in completion time).
That factor can fall to 1.3x, by changing certain features
(at the risk of weakening the tit-for-tat scheme)

For this talk and related materials, go to:

http://www.cs.berkeley.edu/retreat/retreat.html

Future Work

- Investigate and adapt algorithms to:
  - Heterogeneity
    - Hypercube optimization algorithms [10]
  - Streaming delivery
    - Note: the Hypercube algorithm has a log n bound on required buffer size for in-order delivery.
    - Randomized algo: experiment with block selection schemes
  - Dynamicity
  - Cyclic barter
    - The hypercube satisifies cyclic barter, optimally.
  - Overcome communication failures (current work)
- Implement algorithms and evaluate on PlanetLab.

---

[10] Ganesan et al. "Apocrypha: Making P2P Overlays Network-aware" Stanford U. Tech. Rpt.

Scenario: Non-cooperative Clients

(summarized...)

Backup Slide

Background: Non-cooperative clients

- Clients need incentive to upload data to other clients.
- Cash-like mechanisms: e.g. Turner04, Paypal
  - Complex, some centralization required
- Barter-based mechanisms: simpler, no centralization
  - e.g. Chun03, Cox03 (storage and bandwidth)
  - BitTorrent: loosely defined bandwidth tit-for-tat
    - Ill-defined client relationships

- Goal: design/evaluate fast decentralized barter-based content distribution schemes.
  - Requirement: well-defined client relationships/invariants
    - We do *not* focus on incentive analysis[7]
  - Requirement: low graph degrees

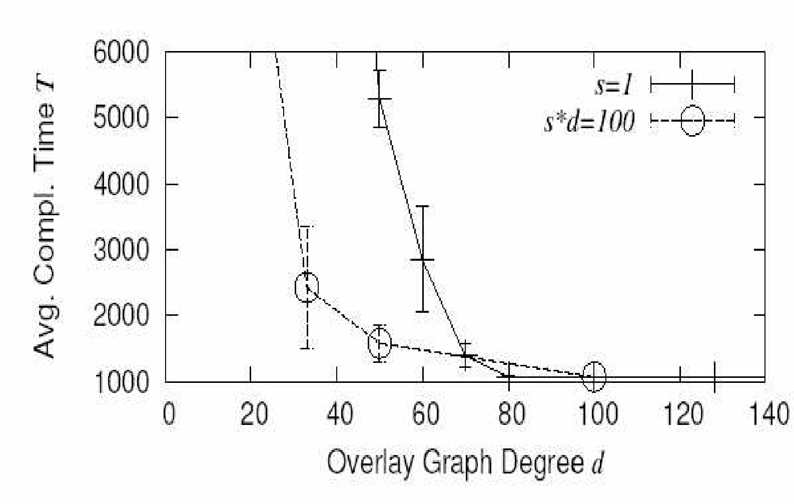
Backup Slide

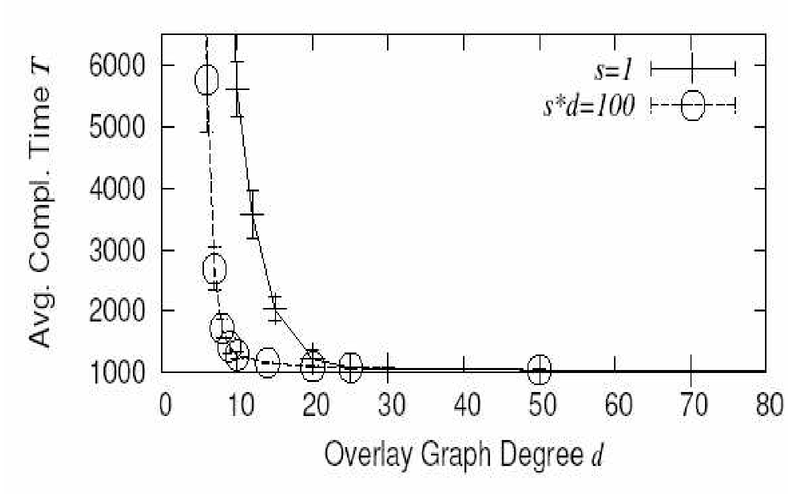
Barter Models

- Strict Barter: lower bound~k+n/2.
  - If download capacity>=2*U, we have an algorithm with T(k,n)=k+n-1.
  - High start-up cost => high completion times
- Relaxed Barter
  - X uploads to Y only if the ***net*** no. of blocks from X to Y is <= S.
  - But: Y can get S*(degree) free blocks
  - So S has to impose a degree limit (issuing tokens to allow peering)
- Special case analyses of Relaxed barter indicate much lower completion times than strict barter
  - S=2,n=power-of-2: Hypercube algorithm can be used.
  - S=1: T(k,n) upper-bounded by k+n-2.
- Simulations for general cases.

Backup Slide

Barter Results





Random Block Selection: Low completion time only at high degrees.

Rarest-first block selection policy is necessary to maintain low degree.

Backup Slide

Cooperative Clients: Properties of the Hypercube Algorithm

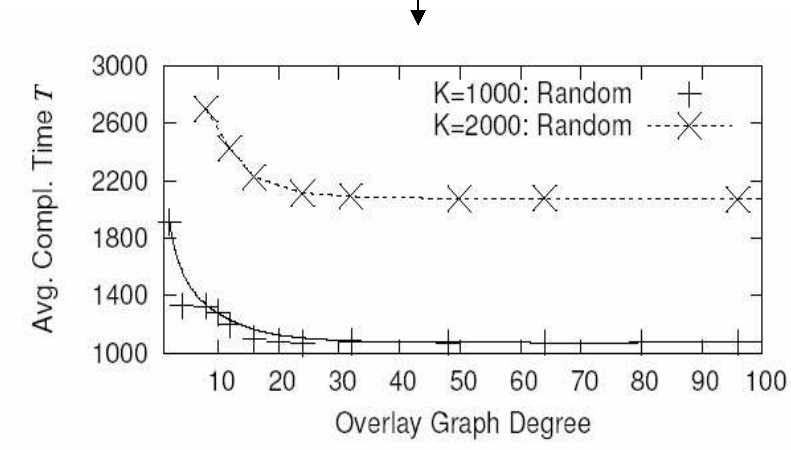
- Low overlay graph degree: Ceil($\log_2 n$)
  - Low overhead of message exchange.
  - Prior algorithms[6] more complex, no degree bound.
- All client-client transfers are *exchanges.*
- Bounded completion time delay per block: Ceil($\log_2 n$)
- All nodes finish together (within 1 tick).
- Satisfies "triangular" barter with credit-limit S=2

Backup Slide

Cooperative Clients: Properties of the Randomized Algorithm

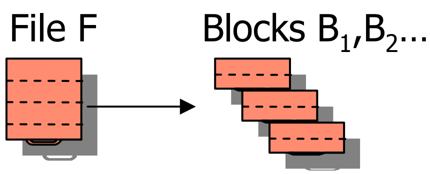
(Cooperative Clients)

- All nodes finish in the last 10% of time.
- Log n – hypercube overlay: random algo has nearly same results.
- Random regular graphs – lower degree {O(log n)} required for near-optimality
  - Degree impact (n=1000) shown below

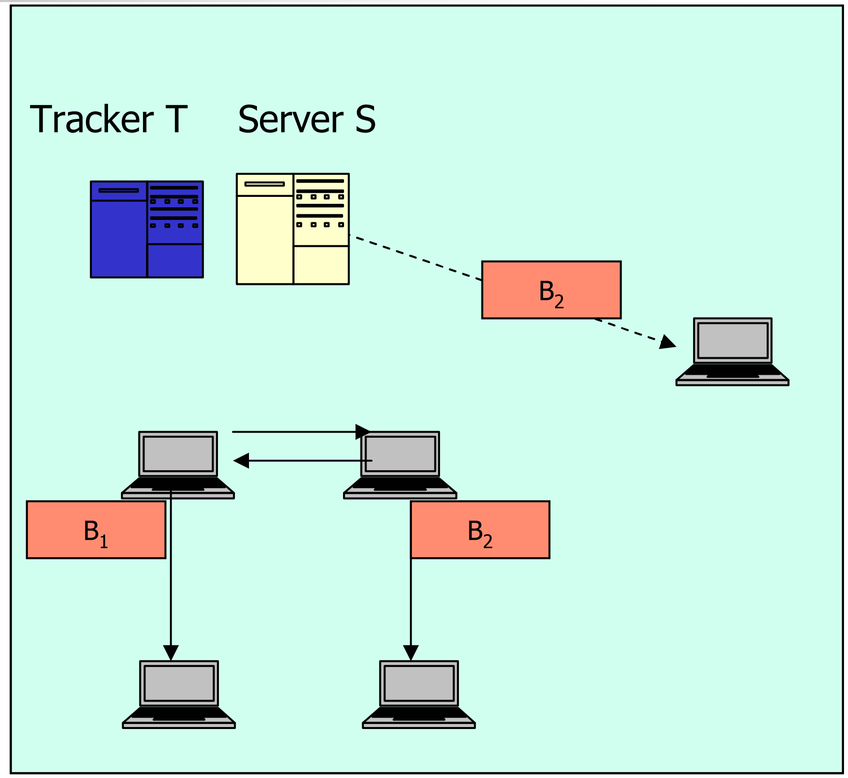


Backup Slide

BitTorrent [.com] - Background



- Tracker (can be at S): enables client rendezvous
- Clients in random overlay graph
- Utilizes clients' upload capacity
- Sub-optimal capacity growth
- Tit-for-tat: prioritize transmissions on incoming bandwidth periodically
  - "choke"/"unchoke"



Completion Time has not been researched

Backup Slide

Summary of Contributions

- Proposed (and analyzed) an optimal algorithm to distribute bulk content with the least *completion* time: the Hypercube algorithm
- For greater ease of deployment, we proposed a randomized variant (and evaluated by simulations)
  - Both the above are faster,simpler, and more general than related prior work [Bittorrent, Qiu04, Xang04, BarNoy00, Splitstream]
- Adapted the above algorithms to non-cooperative scenarios by developing fast barter-based schemes
  - Evaluated the impact of overlay graphs and block-selection policies on completion time