Bounded Multi Port Model: Motivations, Feasability & Application to Broadcast

Olivier Beaumont, **Lionel Eyraud-Dubois**, Shailesh Kumar Agrawal, Hejer Rejeb

Cepage team, LaBRI, Bordeaux, France

Scheduling in Aussois
June 2010

Outline

1. Introduction
2. TCP Bandwidth Sharing
3. Broadcast with Bounded Degree
4. Conclusions

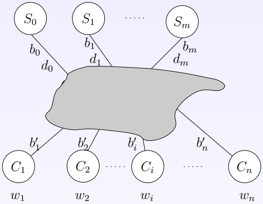
Introduction

Scheduling: what is a good model for communications ?

- Standard communication model: One-Port Model
  - a node is involved in at most one communication at the same time
  - corresponds well to old MPI implementations
- Problem: if the network is strongly heterogeneous, then the bandwidth of the server may be wasted
  - Imagine a server with 1GB/sec bandwidth sending 10MB to a client with 1MB/sec download bandwidth
  - It is not realistic to assume that the server will be busy for 10 secs
- In the context of large scale distributed and strongly heterogeneous platforms, one port model is not the right model

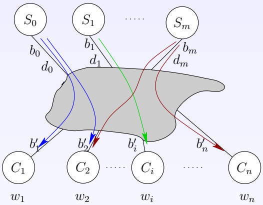
Explore the Bounded Multi Port model

- Simultaneous communications, with a per-node bandwidth bound (both upload and download)
- Internet-like: no contention inside the network



Explore the Bounded Multi Port model

- Simultaneous communications, with a per-node bandwidth bound (both upload and download)
- Internet-like: no contention inside the network



- In this talk, we will see:
  - A model for TCP bandwidth sharing
  - Its influence on several scheduling problems
  - A particular study of the broadcast operation

Outline

1 Introduction

2 TCP Bandwidth Sharing
- Model
- Influence on Scheduling Algorithms

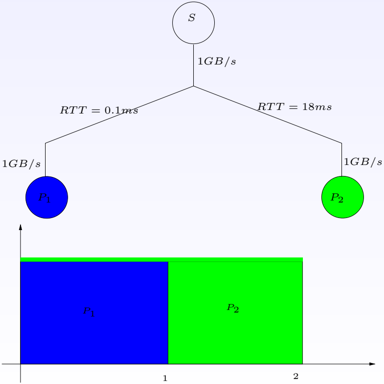
3 Broadcast with Bounded Degree

4 Conclusions

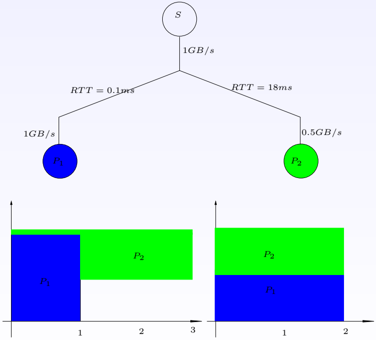
Why is it important ?

- Experiments using French Grid G5K
- The master has N threads, each sending data to N node

Experiment 1: master in Bordeaux, 1 client in Bordeaux (in a different cluster), 1 client in Nancy



Experiment 2: the same, except that the incoming bandwidth of the client in Nancy is twice smaller.



Modeling TCP Bandwidth Sharing

- Increase TCP window sizes until congestion
- TCP window increases quickly for nodes closer to the master
- $\Rightarrow$ closer nodes get higher bandwidth
- Max-Min Fairness algorithm

Model: Casanova and Marchal

Let b_i denote the achievable bandwidth between M and P_i (if alone)
Let λ_i denote the inverse of the RTT between M and P_i

- If $\sum b_i \leq B$, then $all(P_i) = b_i$
- Else
  - While $\exists i,\ b_i \leq \frac{\lambda_i B}{\sum \lambda_k}$: $all(P_i) = b_i$ and update $B \leftarrow B - b_i$
  - $\forall i$ s.t. $b_i > \frac{\lambda_i B}{\sum \lambda_k}$: set $all(P_i) = \frac{\lambda_i B}{\sum \lambda_k}$.

Note that $\sum all(P_i) = \min(B, \sum b_i)$.

An Upper Bound on Performance Degradation

- We consider a set of simultaneous communications between M and the P_is.
- Each communication has a release date r_i and starts immediately.

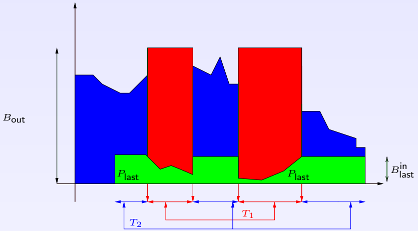
Lemma

If $\sum all(P_i) \leq B$, *then* $\forall i, \quad all(P_i) > 0 \Rightarrow all(P_i) = b_i$.

Theorem

The makespan obtained when relying on TCP Bandwidth Sharing mechanism can be at most twice the optimal makespan

An Upper Bound on Performance Degradation

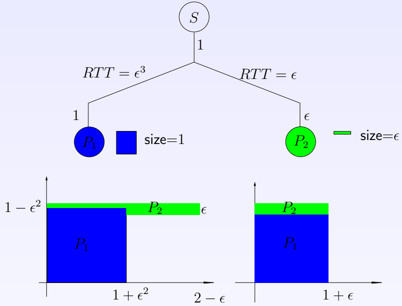


Proof.

- Same as Graham's proof... Consider P_{LAST}, whose last communication ends at T
- Partition $T - r_{\text{LAST}}$ into
  - T_1 instants when all bandwidth B is used, and T_2 the rest
- Then, if T_{OPT} denotes the optimal makespan
  - $T_1 \leq T_{\text{OPT}}$ (nothing is wasted during T_1)
  - $T_2 + r_{\text{LAST}} \leq T_{\text{OPT}}$ (P_{LAST} communicates at maximal rate during T_2)
- Therefore, $T = r_{\text{LAST}} + T_1 + T_2 \leq 2T_{\text{OPT}}$.

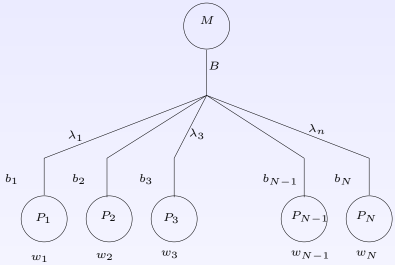
The above bound is tight

Let us consider the following platform



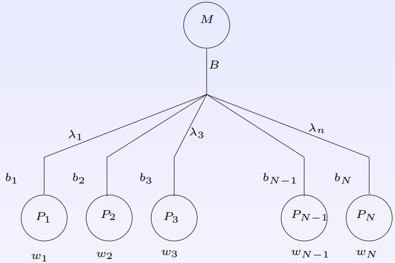
- If we rely on TCP bandwidth mechanism, then P_1 gets too much bandwidth: $1-\epsilon^2$ instead of $1-\epsilon$
- and it takes almost 1 time unit to finish the transfer with P_2
- If we enforce the bandwidth with P_1 to be at most $1-\epsilon$, both transfers end up in time $1+\epsilon$
- The ratio between both solutions is $2-3\epsilon$.

Steady State Scheduling of Independent Tasks



- b_i: number of tasks that can be sent to P_i in one time unit
- w_i: number of tasks that can be processed by P_i in one time unit
- Goal: Maximize the number of tasks that can be processed in steady state by the platform

Scheduling of Independent Tasks: Optimal Solution



- Let n_i denote the number of tasks processed by P_i
- Clearly, $n_i \leq b_i$, $n_i \leq w_i$ and $n_i \leq B$.
- Let us denote by $c_i = \min(b_i, w_i)$ and $C = \sum c_i$

Optimal Solution

- If $C \leq B$, then set $\forall i, \quad n_i = c_i$
- Else set $\forall i, \quad n_i = c_i \frac{B}{C}$.

Scheduling of Independent Tasks: Optimal Algorithm

Implementation 1

In order to avoid starvation, each slave node starts with two tasks in its local buffer. Each time P_i starts processing a new task, it asks for another task and the master node initiates the communication immediately with bandwidth rate n_i.

Proof.

- The bandwidth requested at master node is never larger than B since $\sum n_i \leq B$
- It takes $1/n_i$ time units to P_i to receive a task, and it takes at most $1/n_i$ time units to process it.
- Thus, the processing rate at P_i is exactly n_i.

□

Scheduling of Independent Tasks: Upper Bound

Implementation 1

Each time P_i starts processing a new task, it asks for another task and the master node initiates the communication immediately with bandwidth rate n_i.

Implementation 2

Each time P_i starts processing a new task, it asks for another task and the master node initiates the communication immediately.

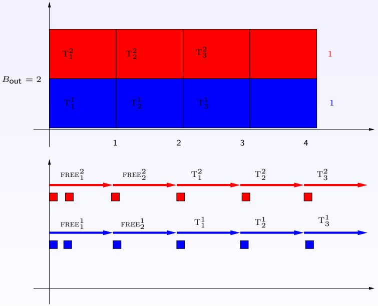
Theorem

The waste W experienced by Implementation 2 per unit time is bounded by $W \leq \frac{1}{4}B$, and hence its throughput verifies $T_2 \geq \frac{3}{4}T_1$

The bound is tight

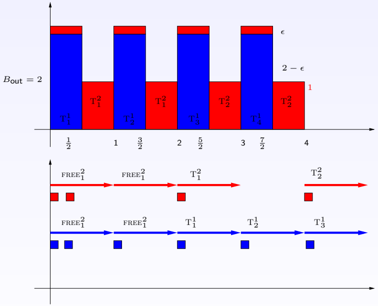
- 2 slave processors
  - P_1: $w_1 = 1,\ b_1 = 2,\ \mathrm{RTT} = \epsilon^2$
  - P_2: $w_2 = 1,\ b_2 = 1,\ \mathrm{RTT} = \epsilon$
- The ratio between both implementations is $\frac{4}{3}$

Using Implementation 1



2 tasks every 2 time units

Using Implementation 2



3 tasks every 4 time units

Summary of first part

- Multiport model is more realistic than 1-port model
- TCP Bandwidth sharing mechanism
  - is complicated
  - and strongly depends on RTT values.
- On the other hand,
  - we usually know what bandwidth should be allocated
  - and many mechanisms exist to limit the bandwidth of a connexion
- So, use them!

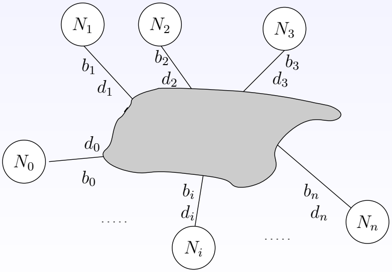
Outline

1. Introduction
2. TCP Bandwidth Sharing
3. Broadcast with Bounded Degree
    - Problem and Complexity
    - Algorithms
    - Evaluation
4. Conclusions

Introduction

From now on: broadcast/streaming operation

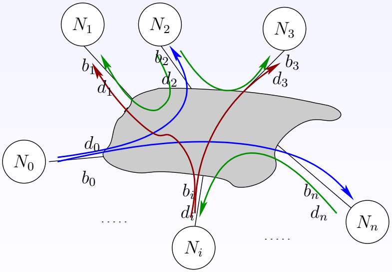
- One source node holds (or generates) a message
- All nodes must receive the complete message
- Steady-state: quantity of data per time unit
- Goal: optimize throughput
- Keep things reasonable: degree constraint



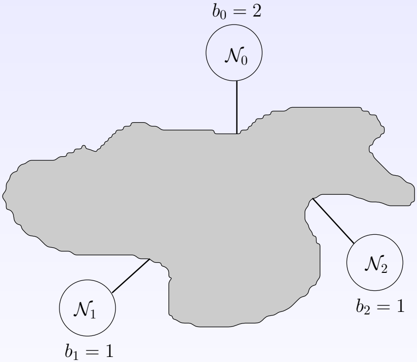
Introduction

From now on: broadcast/streaming operation

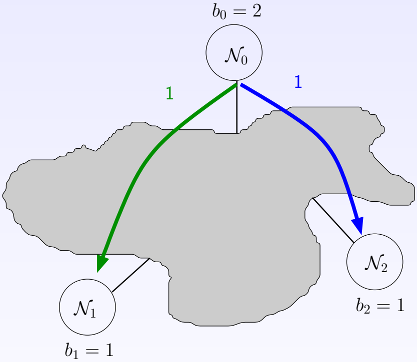
- One source node holds (or generates) a message
- All nodes must receive the complete message
- Steady-state: quantity of data per time unit
- Goal: optimize throughput
- Keep things reasonable: degree constraint



An example

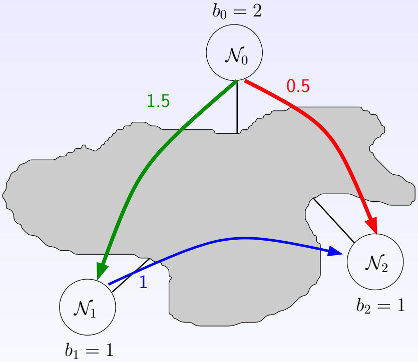


An example



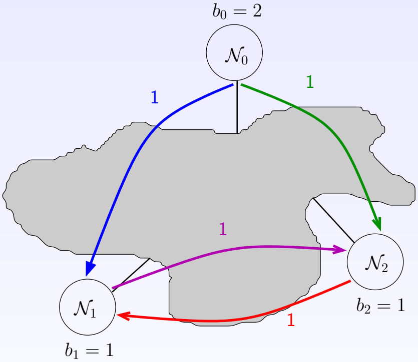
Best tree: $T = 1$

An example



Best DAG: $T = 1.5$

An example



Optimal: $T = 2$

Precise model

An instance

- n nodes, with output bandwidth b_i and maximal out-degree d_i
- node $\mathcal{N}_0$ is the master node that holds the data

A solution (Flows)

- Flow f_j^i from node $\mathcal{N}_j$ to $\mathcal{N}_i$
- $\forall j, \quad \left|\left\{i,\ f_j^i > 0\right\}\right| \leq d_j$ degree constraint at $\mathcal{N}_j$
- $\forall j, \quad \sum_i f_j^i \leq b_j$ capacity constraint at $\mathcal{N}_j$
- Maximize $T = \min_j \text{mincut}(\mathcal{N}_0, \mathcal{N}_j)$

Complexity

3-Partition

- $3p$ integers a_i such that $\sum_i a_i = pT$
- Partition into p sets S_l such that $\sum_{i \in S_l} a_i = T$

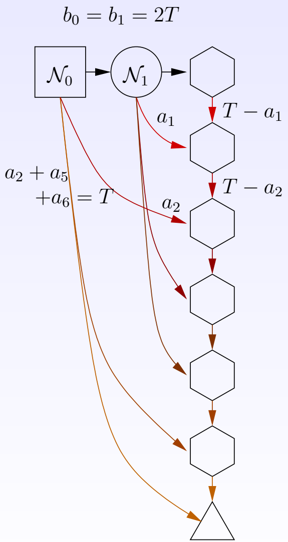
Complexity

3-Partition

- $3p$ integers a_i such that $\sum_i a_i = pT$
- Partition into p sets S_l such that $\sum_{i \in S_l} a_i = T$

Reduction

- p "server" nodes, $b_j = 2T$ and $d_j = 4$
- $3p$ "client" nodes, $b_{j+p} = T - a_j$ and $d_{j+p} = 1$
- 1 "terminal" node, $b_{4p} = 0$, $d_{4p} = 0$



Upper bound

If $\mathcal{S}$ has throughput T

- Node $\mathcal{N}_i$ uses at most $X_i = \min(b_i, Td_i)$
- Total received rate: nT
- Thus $\sum_{i=0}^{n} \min(b_i, Td_i) \geq nT$
- Of course, $T \leq b_0$

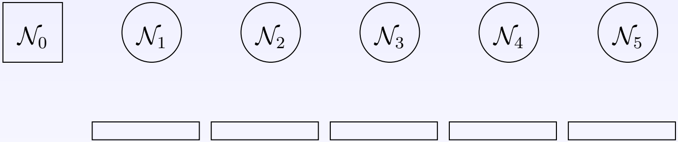
Our algorithms

- Inputs: an instance, and a goal throughput T
- Output: a solution with resource augmentation (additional connections allowed)

ACYCLIC algorithm

If $\sum_{i=0}^{n-1} \min(b_i, Td_i) \geq nT$

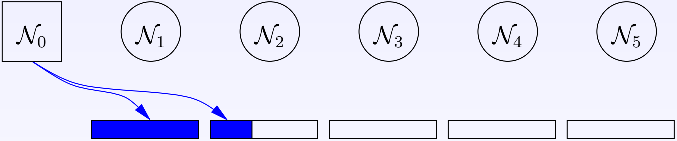
- Order nodes by capacity : $X_1 \geq X_2 \geq \cdots \geq X_n$
- Each node k sends throughput T to as many nodes as possible, in consecutive order



ACYCLIC algorithm

If $\sum_{i=0}^{n-1} \min(b_i, Td_i) \geq nT$

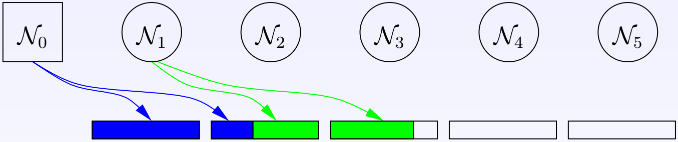
- Order nodes by capacity : $X_1 \geq X_2 \geq \cdots \geq X_n$
- Each node k sends throughput T to as many nodes as possible, in consecutive order



ACYCLIC algorithm

If $\sum_{i=0}^{n-1} \min(b_i, T d_i) \geq nT$

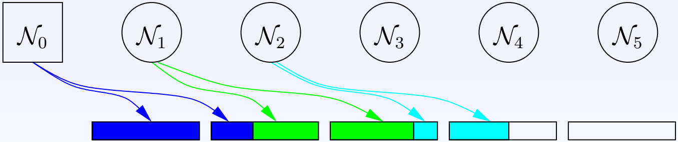
- Order nodes by capacity : $X_1 \geq X_2 \geq \cdots \geq X_n$
- Each node k sends throughput T to as many nodes as possible, in consecutive order



ACYCLIC algorithm

If $\sum_{i=0}^{n-1} \min(b_i, Td_i) \geq nT$

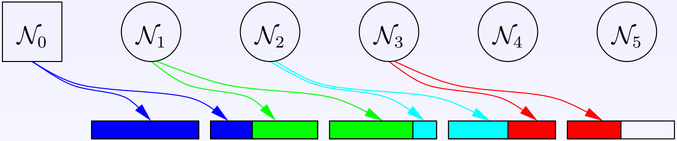
- Order nodes by capacity : $X_1 \geq X_2 \geq \cdots \geq X_n$
- Each node k sends throughput T to as many nodes as possible, in consecutive order



ACYCLIC algorithm

If $\sum_{i=0}^{n-1} \min(b_i, Td_i) \geq nT$

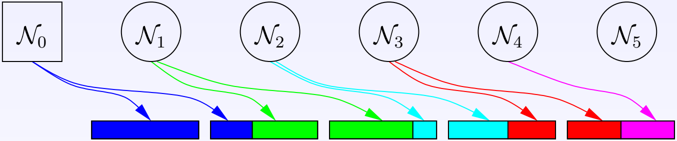
- Order nodes by capacity : $X_1 \geq X_2 \geq \cdots \geq X_n$
- Each node k sends throughput T to as many nodes as possible, in consecutive order



ACYCLIC algorithm

If $\sum_{i=0}^{n-1} \min(b_i, Td_i) \geq nT$

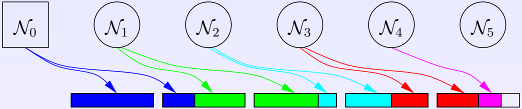
- Order nodes by capacity : $X_1 \geq X_2 \geq \cdots \geq X_n$
- Each node k sends throughput T to as many nodes as possible, in consecutive order



Provides a valid solution

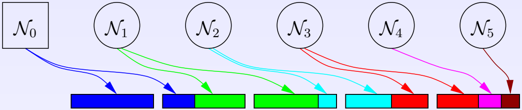
- $b_0 \geq T$
- Sort by $X_i \implies \forall k, \sum_{i=0}^{k} X_i \geq (k+1)T$
- Since $X_k \leq Td_k$, the outdegree of $\mathcal{N}_k$ is at most $d_k + 1$

General case: $\sum_{i=0}^{n} X_i \geq nT$



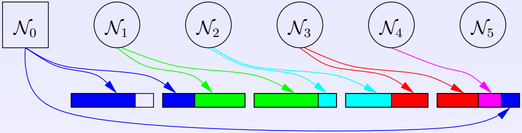
- Start with ACYCLIC, until k_0 such that $\sum_{i=0}^{k_0} X_i < (k_0+1)T$

General case: $\sum_{i=0}^{n} X_i \geq nT$



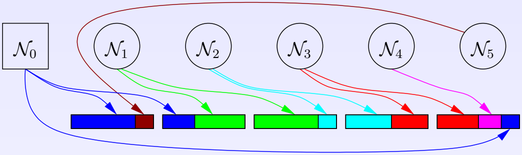
- Start with ACYCLIC, until k_0 such that $\sum_{i=0}^{k_0} X_i < (k_0 + 1)T$

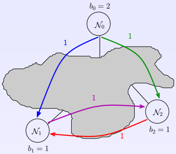
General case: $\sum_{i=0}^{n} X_i \geq nT$



- Start with ACYCLIC, until k_0 such that $\sum_{i=0}^{k_0} X_i < (k_0 + 1)T$

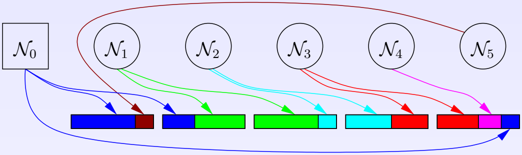
General case: $\sum_{i=0}^{n} X_i \geq nT$

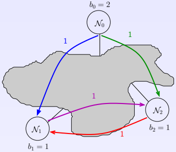




- Start with Acyclic, until k_0 such that $\sum_{i=0}^{k_0} X_i < (k_0+1)T$

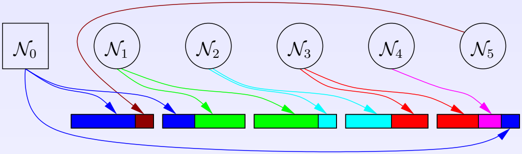
General case: $\sum_{i=0}^{n} X_i \geq nT$

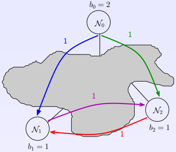




- Start with Acyclic, until k_0 such that $\sum_{i=0}^{k_0} X_i < (k_0 + 1)T$

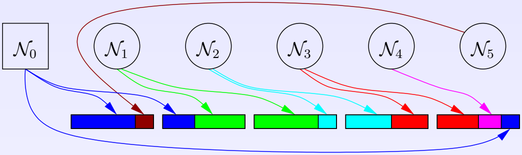
General case: $\sum_{i=0}^{n} X_i \geq nT$

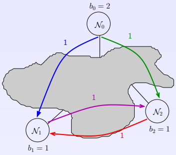




- Start with $\textsc{Acyclic}$, until k_0 such that $\sum_{i=0}^{k_0} X_i < (k_0+1)T$
- Succesively build partial solutions in which
  - All nodes up to $\mathcal{N}_k$ are served
  - Only node $\mathcal{N}_k$ has remaining bandwidth
- Use the source and $\mathcal{N}_{k_0-1}$ to serve $\mathcal{N}_{k_0}$ and $\mathcal{N}_{k_0+1}$
- Then for all k, $\mathcal{N}_{k+1}$ is served by $\mathcal{N}_k$ and $\mathcal{N}_{k-1}$

General case: $\sum_{i=0}^{n} X_i \geq nT$





- Start with ACYCLIC, until k_0 such that $\sum_{i=0}^{k_0} X_i < (k_0 + 1)T$
- Succesively build partial solutions in which
  - All nodes up to $\mathcal{N}_k$ are served
  - Only node $\mathcal{N}_k$ has remaining bandwidth
- Use the source and $\mathcal{N}_{k_0-1}$ to serve $\mathcal{N}_{k_0}$ and $\mathcal{N}_{k_0+1}$
- Then for all k, $\mathcal{N}_{k+1}$ is served by $\mathcal{N}_k$ and $\mathcal{N}_{k-1}$

Final outdegree of $\mathcal{N}_i$: $o_i \leq \max(d_i + 2, 4)$

- Acyclic solution: $o_i \leq d_i + 1$
- Degree of the source and of $\mathcal{N}_{k_0-1}$ is increased by 1
- $\mathcal{N}_k$ has edges to $\mathcal{N}_{k-2}$, $\mathcal{N}_{k-1}$, $\mathcal{N}_{k+1}$ and $\mathcal{N}_{k+2}$.

Comparison of different solutions

Unconstrained solution

Best achievable throughput without degree constraints: $\frac{\sum_i b_i}{n}$

Best Tree

In a tree of throughput T, flow through all edges must be T. Counting the edges yield $\sum_i \min(d_i, \left\lfloor \frac{b_i}{T} \right\rfloor) \geq n$.

Best Acyclic

Computed by the $\textsc{Acyclic}$ algorithm

Cyclic

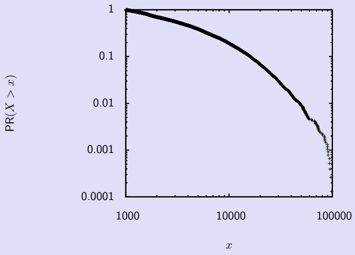
Throughput when adding cycles

Experimental setting

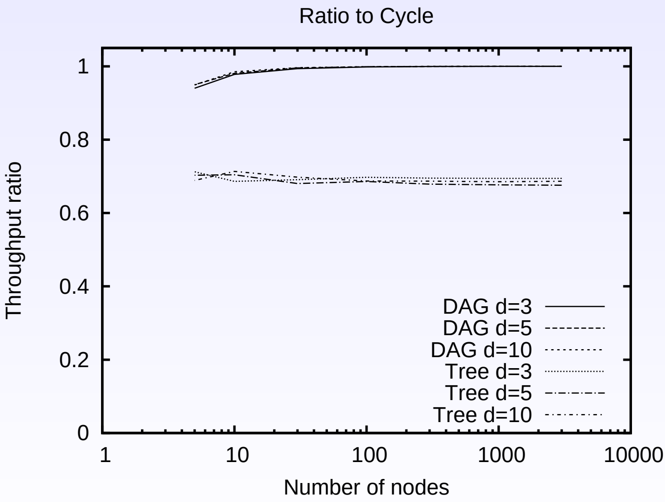
Random instance generation

- Outgoing bandwidths generated from the data of XtremLab project
- Nodes degrees are homogeneous

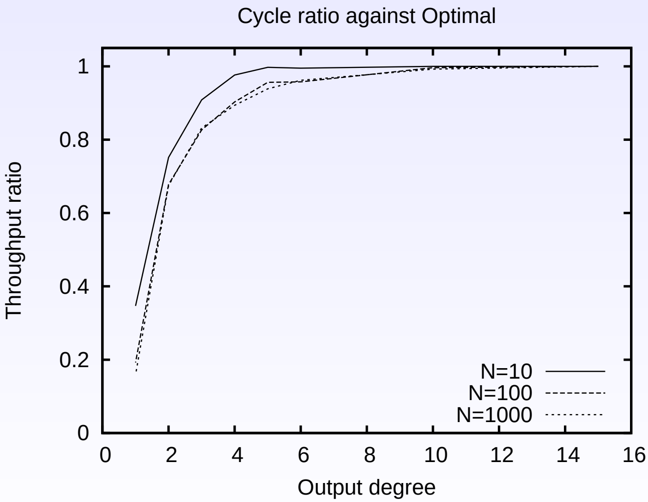
Complementary CDF of the data used



Results: comparisons to Cyclic



Results: Cyclic vs Unconstrained



Outline

1. Introduction
2. TCP Bandwidth Sharing
3. Broadcast with Bounded Degree
4. Conclusions

Summary of first part

- Multiport model is more realistic than 1-port model
- TCP Bandwidth sharing mechanism
  - is complicated
  - and strongly depends on RTT values.
- On the other hand,
  - we usually know what bandwidth should be allocated
  - and many mechanisms exist to limit the bandwidth of a connexion
- So, use them!

Summary of second part

- Theoretical study of the broadcast problem:
  - optimal resource augmentation algorithm
- In practice:
  - a low degree is enough to reach a high throughput
  - an acyclic solution is very reasonable
  - once the overlay is computed, there exist distributed algorithms to perform the broadcast

Going further

- Worst-case approximation ratio of ACYCLIC ?
- Study the robustness of our algorithms
- Design on-line and/or distributed versions