Overlay networks maximizing throughput

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

Cepage team, LaBRI, Bordeaux, France

IPDPS
April 20, 2010

Outline

1. Introduction
2. Complexity
3. Successive algorithms
4. Simulations
5. Conclusions

Introduction

In this talk: broadcast/streaming operation

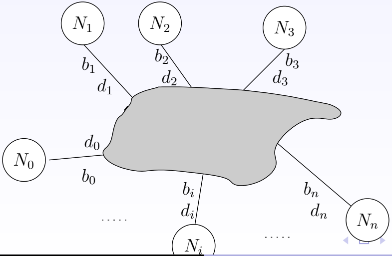
- One source node holds (or generates) a message
- All nodes must receive the complete message
- Steady-state: quantity of data per time unit
- Goal: optimize throughput

Communication model

Explore the Bounded Multi Port model

- P2P setting, Application-Level: no *a priori* communication network
- Simultaneous communications, with a per-node bandwidth bound
- Internet-like: no contention inside the network
- Steady-state approach

- Goal of algorithms: build an (efficient) overlay
- Keep things reasonable: degree constraint

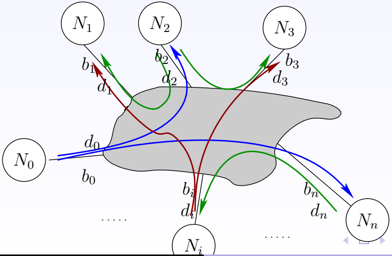


Communication model

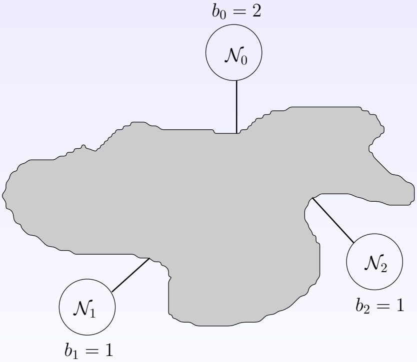
Explore the Bounded Multi Port model

- P2P setting, Application-Level: no *a priori* communication network
- Simultaneous communications, with a per-node bandwidth bound
- Internet-like: no contention inside the network
- Steady-state approach

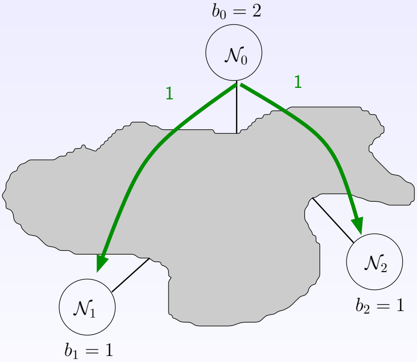
- Goal of algorithms: build an (efficient) overlay
- Keep things reasonable: degree constraint

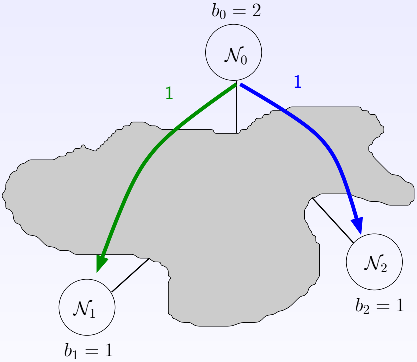


An example



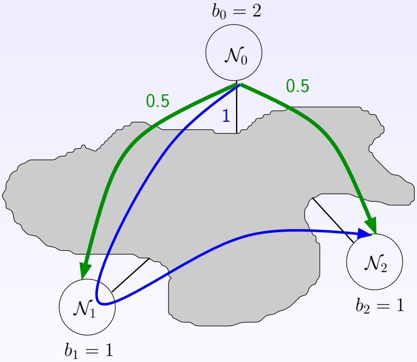
An example

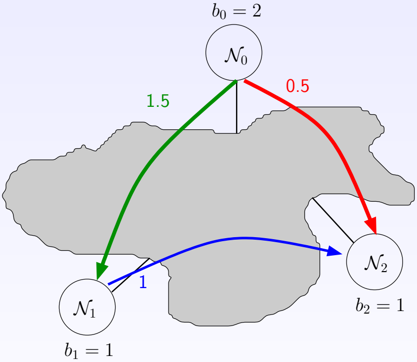




Best tree: $T = 1$

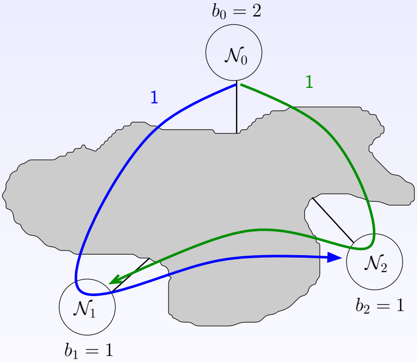
An example

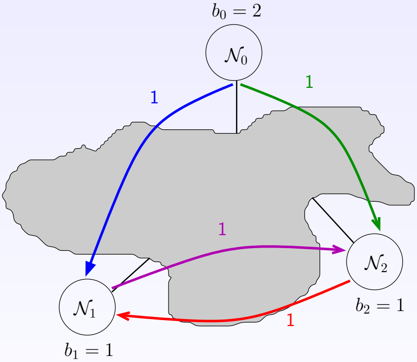




Best DAG: $T = 1.5$

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 (Trees)

- A weighted set of spanning trees (w_k, T_k)
- $\forall j, \quad \sum_k \sum_i \chi_k(\mathcal{N}_j, \mathcal{N}_i) w_k \leq b_j$ (capacity constraint at node j)
- $\forall j, \quad \sum_i \max_k \chi_k(\mathcal{N}_j, \mathcal{N}_i) \leq d_j$ (degree constraint at node j)
- Maximize $T = \sum_k w_k$

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 $T = \min_j \text{mincut}(\mathcal{N}_0, \mathcal{N}_j)$

Outline

1. Introduction
2. Complexity
3. Successive algorithms
4. Simulations
5. Conclusions

NP-Hardness

3-Partition

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

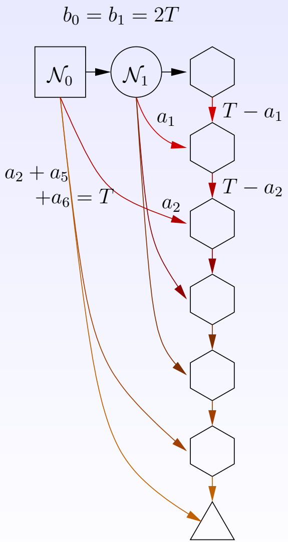
NP-Hardness

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, $b_{4p} = 0$, $d_{4p} = 0$



Outline

1 Introduction

2 Complexity

3 Successive algorithms
- Acyclic Algorithm
- With cycles

4 Simulations

5 Conclusions

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, $T \leq b_0$

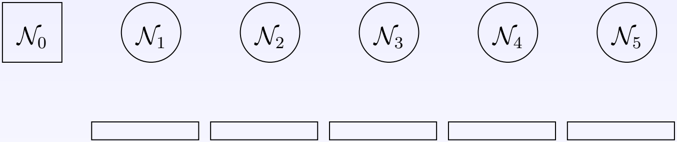
Our algorithms

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

ACYCLIC algorithm

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

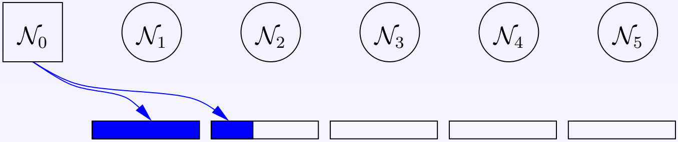
- Order nodes by capacity : $X_1 \geq X_2 \geq \cdots \geq X_n$
- Each node k sends throughput T to as many nodes as possible, in consecutive order



ACYCLIC algorithm

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

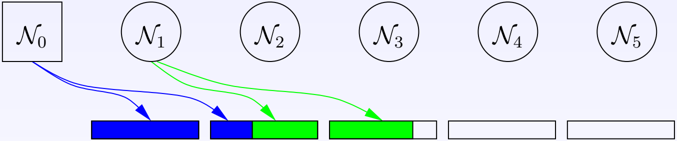
- Order nodes by capacity : $X_1 \geq X_2 \geq \cdots \geq X_n$
- Each node k sends throughput T to as many nodes as possible, in consecutive order



ACYCLIC algorithm

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

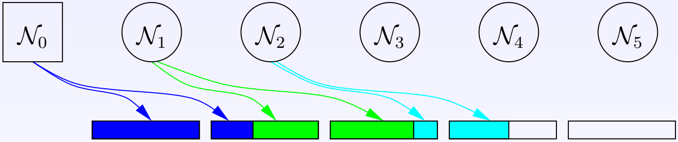
- Order nodes by capacity : $X_1 \geq X_2 \geq \cdots \geq X_n$
- Each node k sends throughput T to as many nodes as possible, in consecutive order



ACYCLIC algorithm

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

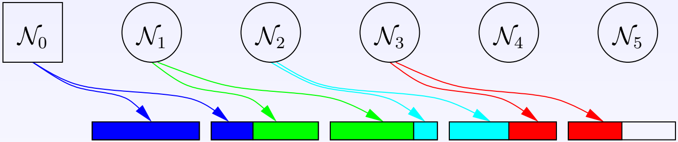
- Order nodes by capacity : $X_1 \geq X_2 \geq \cdots \geq X_n$
- Each node k sends throughput T to as many nodes as possible, in consecutive order



ACYCLIC algorithm

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

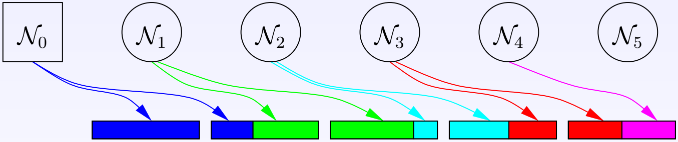
- Order nodes by capacity : $X_1 \geq X_2 \geq \cdots \geq X_n$
- Each node k sends throughput T to as many nodes as possible, in consecutive order



ACYCLIC algorithm

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

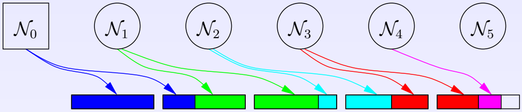
- Order nodes by capacity : $X_1 \geq X_2 \geq \cdots \geq X_n$
- Each node k sends throughput T to as many nodes as possible, in consecutive order



Provides a valid solution

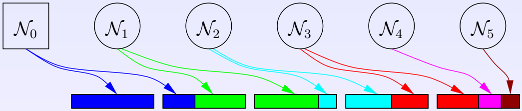
- $b_0 \geq T$
- Sort by $X_i \implies \forall k, \sum_{i=0}^{k} X_i \geq (k+1)T$
- Since $X_k \leq Td_k$, the outdegree of $\mathcal{N}_k$ is at most $d_k + 1$

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



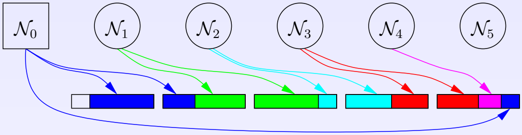
- Acyclic algorithm until k_0 such that $\sum_{i=0}^{k_0} X_i < (k_0 + 1)T$

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



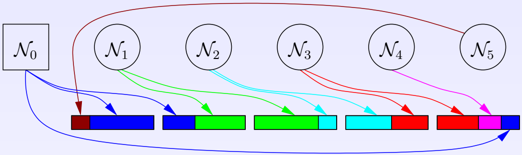
- Acyclic algorithm until k_0 such that $\sum_{i=0}^{k_0} X_i < (k_0 + 1)T$

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



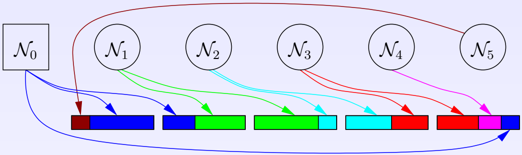
- Acyclic algorithm until k_0 such that $\sum_{i=0}^{k_0} X_i < (k_0 + 1)T$

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



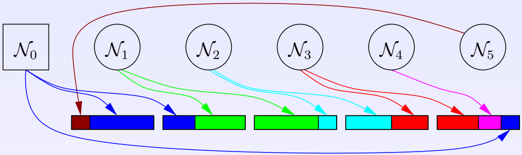
- Acyclic algorithm until k_0 such that $\sum_{i=0}^{k_0} X_i < (k_0+1)T$

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



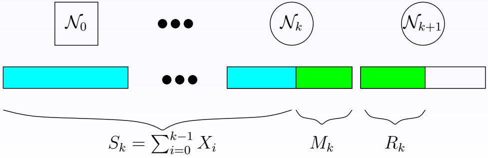
- Acyclic algorithm until k_0 such that $\sum_{i=0}^{k_0} X_i < (k_0 + 1)T$

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



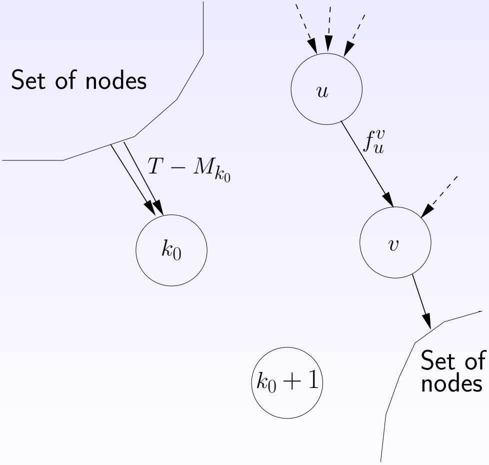
- Acyclic algorithm until k_0 such that $\sum_{i=0}^{k_0} X_i < (k_0 + 1)T$
- Recursively build partial solutions in which
  - All nodes up to $\mathcal{N}_k$ are served
  - Only node $\mathcal{N}_k$ has remaining bandwidth

Notations:



- $M_k = kT - \sum_{i=0}^{k-1} X_i$
- $M_k + R_k = X_k$
- $M_{k+1} = T - R_k$

Recursion: initial case

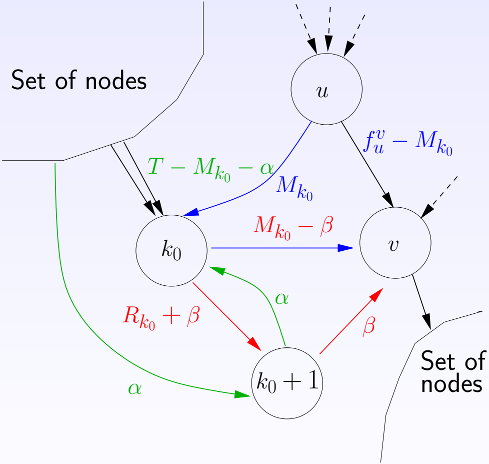


Remind

- $M_k + R_k = X_k$
- $R_k + M_{k+1} = T$

- $f_u^v \geq M_{k_0}$

Recursion: initial case

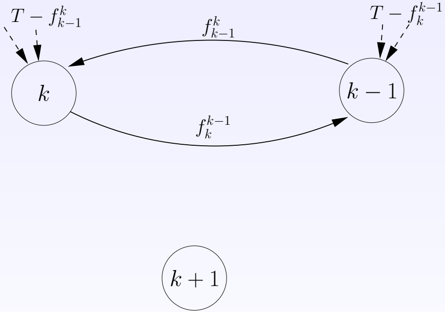


Remind

- $M_k + R_k = X_k$
- $R_k + M_{k+1} = T$

- $f_u^v \geq M_{k_0}$
- $\beta = \frac{M_{k_0+1}}{T} M_{k_0}$
- $\alpha = \frac{M_{k_0+1}}{T}(T - M_{k_0})$
- $\alpha + \beta = M_{k_0+1}$
- $f_{k_0}^{k_0+1} + f_{k_0+1}^{k_0} = T$

Recursion: general case

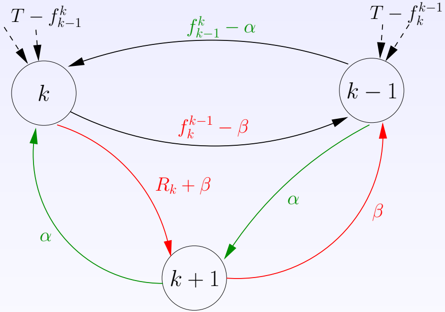


Remind

- $M_k + R_k = X_k$
- $R_k + M_{k+1} = T$

- $f_k^{k-1} + f_{k-1}^{k} = T$

Recursion: general case



Remind

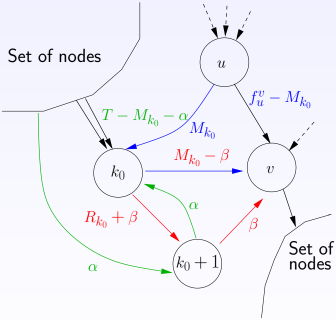
- $M_k + R_k = X_k$
- $R_k + M_{k+1} = T$

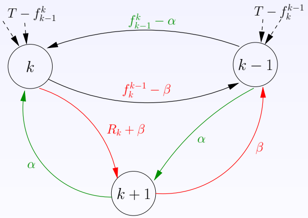
- $f_k^{k-1} + f_{k-1}^{k} = T$
- $\beta = \frac{M_{k+1}}{T} f_k^{k-1}$
- $\alpha = \frac{M_{k+1}}{T} f_{k-1}^{k}$
- $\alpha + \beta = M_{k+1}$
- $f_k^{k+1} + f_{k+1}^{k} = T$

Recursion: result

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

- Acyclic solution: $o_i \leq d_i + 1$
- Degree of $\mathcal{N}_u$ and of one node in the set is increased by 1
- Step k: o_k and o_{k-1} are increased by 1





Outline

1 Introduction

2 Complexity

3 Successive algorithms

4 Simulations

5 Conclusions

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 $\sum_i \min(d_i, \left\lfloor \frac{b_i}{T} \right\rfloor) \geq n$.

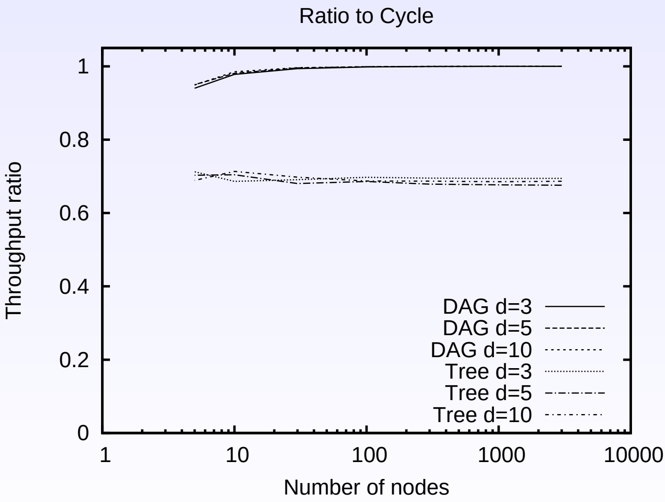
Best Acyclic

Computed by the $\textsc{Acyclic}$ algorithm

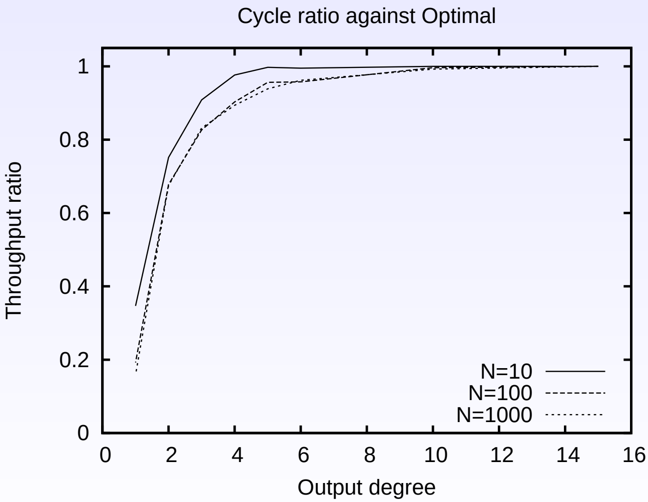
Cyclic

Throughput when adding cycles

Results: comparisons to Cyclic



Results: Cyclic vs Unconstrained



Outline

1 Introduction

2 Complexity

3 Successive algorithms

4 Simulations

5 Conclusions

Summary

- Theoretical study of the 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