Overlay networks maximizing throughput

Olivier Beaumont, Lionel Eyraud-Dubois, Shailesh Kumar Agrawal

Cepage team, LaBRI, Bordeaux, France

RO Working Group
May 24, 2011
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
- Keep things reasonable: degree constraint
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
- Keep things reasonable: degree constraint
An example

$N_0 \quad b_0 = 2$

$N_1 \quad b_1 = 1$

$N_2 \quad b_2 = 1$

$N_0 \quad b_0 = 2$

$N_1 \quad b_1 = 1$

$N_2 \quad b_2 = 1$
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 $N_0$ is the master node that holds the data

A solution (Trees)
- A weighted set of spanning trees $(w_k, T_k)$ rooted at $N_0$
- $\chi_k(N_j, N_i) = 1$ iff there is an edge from $N_j$ to $N_i$ in tree $T_k$
- $\forall j, \sum_k \sum_i \chi_k(N_j, N_i) w_k \leq b_j$ (capacity constraint at node $j$)
- $\forall j, \sum_i \max_k \chi_k(N_j, N_i) \leq d_j$ (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 \( N_0 \) is the master node that holds the data

A solution (Flows)

- Flow \( f^i_j \) from node \( N_j \) to \( N_i \)
- \( \forall j, \sum_i f^i_j \leq b_j \) (capacity constraint at node \( j \))
- \( \forall j, \left| \left\{ i, f^i_j > 0 \right\} \right| \leq d_j \) (degree constraint at node \( j \))
- Maximize \( T = \min_j \text{mincut}(N_0, N_j) \)
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, $b_{4p} = 0$, $d_{4p} = 0$

$$b_0 = b_1 = 2T$$
$$N_0 \rightarrow N_1$$

$$a_2 + a_5 + a_6 = T$$

$$T - a_1$$

$$T - a_2$$
Outline

1 Introduction

2 Successive algorithms
   - Acyclic Algorithm
   - With cycles

3 Simulations

4 Conclusions
Upper bound

If $S$ has throughput $T$

- Node $N_i$ 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

\[ \begin{align*}
  \mathcal{N}_0 & \quad \mathcal{N}_1 & \quad \mathcal{N}_2 & \quad \mathcal{N}_3 & \quad \mathcal{N}_4 & \quad \mathcal{N}_5 \\
\end{align*} \]
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

Overlay networks maximizing throughput
Acyclic Algorithm

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

- 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.

![Diagram of nodes and connections](image-url)
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 $d_k + 1$
Successive algorithms with cycles

**General case:** \( \sum_{i=0}^{n} X_i \geq nT \)

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

\[ N_0 \quad N_1 \quad N_2 \quad N_3 \quad N_4 \quad N_5 \]
Successive algorithms

With cycles

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 \( \sum_{i=0}^{k_0} X_i < (k_0 + 1)T \)
Successive algorithms with cycles

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
- 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 \( \mathcal{N}_{k-1} \)
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

- 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 $\mathcal{N}_{k+2}$. 
A running example

\[ b_0 = \frac{5}{4} \]
\[ \forall i > 0, b_i = \frac{7}{8} \]
A running example

\[ b_0 = \frac{5}{4} \]

\[ \forall i > 0, b_i = \frac{7}{8} \]
A running example

\[ \begin{align*}
    b_0 &= \frac{5}{4} \\
    \forall i > 0, b_i &= \frac{7}{8}
\end{align*} \]
A running example

\[ b_0 = \frac{5}{4} \]
\[ \forall i > 0, b_i = \frac{7}{8} \]
A running example

\[ \forall i > 0, b_i = \frac{7}{8} \]

\[ b_0 = \frac{5}{4} \]
A running example

\[
\begin{align*}
 b_0 &= \frac{5}{4} \\
 \forall i > 0, b_i &= \frac{7}{8}
\end{align*}
\]
A running example

\[ N_0 \]

\[ N_1 \rightarrow N_2 \rightarrow N_4 \rightarrow N_5 \rightarrow N_6 \]

\[ b_0 = \frac{5}{4} \]

\[ \forall i > 0, b_i = \frac{7}{4} \]
A running example

\[ b_0 = \frac{5}{4} \]

\[ \forall i > 0, b_i = \frac{7}{8} \]
A running example

$$\forall i > 0, b_i = \frac{7}{8}$$

$$b_0 = \frac{5}{4}$$
Successive algorithms
With cycles

A running example

∀i > 0, b_i = \frac{5}{8}

b_0 = \frac{5}{4}
A running example

\[ b_0 = \frac{5}{4}, \quad \forall i > 0, b_i = \frac{7}{4} \]
Outline

1. Introduction
2. Successive algorithms
3. Simulations
4. 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 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

Throughput ratio vs. Number of nodes for different structures and depths.

- DAG d=3
- DAG d=5
- DAG d=10
- Tree d=3
- Tree d=5
- Tree d=10
Results: Cyclic vs Unconstrained

Cycle ratio against Optimal

Throughput ratio vs Output degree for different values of N (N=10, N=100, N=1000).
Outline

1. Introduction
2. Successive algorithms
3. Simulations
4. 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 \textsc{Acyclic}?
- Study the \textit{robustness} of our algorithms
- Design \textit{on-line} and/or \textit{distributed} versions