

Pipelining Broadcasts on Heterogeneous Platforms

Olivier Beaumont
Aranud Legrand, Loris Marchal, Yves Robert (ENS Lyon)

LaBRI – Équipe Scalaplix

February 8, 2006

Introduction

- Complex applications on grids require collective communication schemes:
 - one-to-all Broadcast, Multicast, Scatter
 - all-to-one Gather, Reduce
 - all-to-all Gossip, All-to-All
- Numerous studies concentrate on a single communication scheme, mainly about one single broadcast \Rightarrow find the best broadcast tree.
- Pipelining communications:
 - ▶ data parallelism involves a large amount of data
 - ▶ not a single communication, but a series of same communication schemes (e.g. a series of broadcasts from the same source)
 - ▶ maximize throughput of the steady-state operation
 - ▶ use several broadcast trees

Pipelining Broadcasts

- Minimize the time to broadcast a unit size message at steady state
- \implies *optimal pattern* \rightsquigarrow *periodic* schedule + init. + clean-up
- \implies *asymptotically optimal* schedule for makespan minimization
- n messages from P_0 to all other P_i 's
- Let $T_{opt}(n)$ denote the optimal time for broadcasting the n messages, possibly usually n (distinct) broadcast trees.
- Asymptotic optimality:
$$\lim_{n \rightarrow +\infty} \frac{T_{alg}(n)}{T_{opt}(n)} = 1$$

Throughput maximization vs Single broadcast makespan minimization

For large size messages, communications need to be pipelined (either using a single broadcast tree or several broadcast trees)

For large size messages throughput maximization

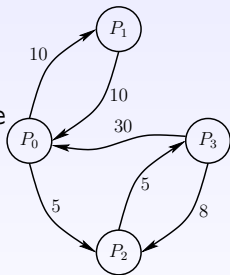
- 😊 provides better results (use of several broadcast trees, thus better use of the platform capabilities)
- 😊 is more tractable (as we will see, polynomial algorithms can be derived, at least for broadcasts)
- 😞 large messages \implies long time \implies stability of the platform???
- platform dynamism must be taken into account

Outline

- 1 Platform Modeling
- 2 Pipelining broadcasts: general framework
- 3 Efficient algorithm: bidirectional one-port model
- 4 Decentralized Solutions (Awerbuch Leighton Algorithm)
- 5 Network Coding
- 6 Conclusion

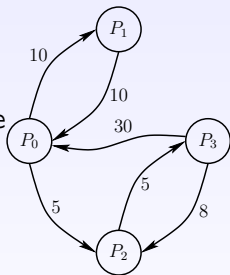
Modeling the platform:bidirectional one port model

- $G = (P, E, c)$
- Let P_1, P_2, \dots, P_n be the n processors
- $(P_j, P_k) \in E$ denotes a communication link between P_i and P_j
- $c(P_j, P_k)$ denotes the time to transfer one unit-size message from P_j to P_k
- one-port for incoming communications
- one-port for outgoing communications
- at a given time step P_i receives a message from at most one neighbor and sends a message to at most one neighbor.



Modeling the platform: unidirectional one port model

- $G = (P, E, c)$
- Let P_1, P_2, \dots, P_n be the n processors
- $(P_j, P_k) \in E$ denotes a communication link between P_i and P_j
- $c(P_j, P_k)$ denotes the time to transfer one unit-size message from P_j to P_k
- one-port for **both** incoming and outgoing communications
- at a given time step P_i can either receive a message from at most one neighbor or send a message to at most one neighbor.



Related Work (1): models for heterogeneous networks

Model by Cappello, Fraigniaud, Maus and Rosenberg:

- $s_u + \sigma + \lambda + r_v$ time to send the message from u to v .
- u is busy during $s_u + \sigma$ time units (does not depend on v)
- v is busy during $r_v + \sigma$ time units (does not depend on u)
- heterogeneous processors but homogeneous fully connected network

Results

- 1 Single Broadcast (+personal processing): NP-Complete, the approximation ratio depends on $\frac{r_{\max}}{r_{\min}}$.
- 2 Single Multicast: NP-Complete, unknown approximation ratio

Related Work (2): models for heterogeneous networks

Model by Bar-Noy, Guha, Naor and Schieber:

- $\lambda_{u,v}$ time to send the message from u to v .
- u is busy during s_u time units (does not depend on v)
- v is busy during r_v time units (does not depend on u)
- s_u , r_v and $\lambda_{u,v}$ are affine in the size of the message

Results

- 1 Single Broadcast: NP-Hard to provide a 3 approximation
- 2 Single Multicast: NP Hard to provide a $\log k$ approximation, where k is the size of the set of targets

Related Work (3): models for heterogeneous networks

Model by Bo Hong and V. Prasanna:

- $\lambda_{u,v}$ to send the message from the front end of u to the front end of v
- s_u to transfer the message between u and its front-end
- r_v to transfer the message between the front end of v and v
- s_u , r_v and $\lambda_{u,v}$ are linear in the size of the message

Results

- 1 no results for pipelined collective communications (complexity results for previous model hold true for single broadcast)
- 2 some problems (see later) can be formulated as flow problems

Comparison of the different models

- All models are more or less 1-port $(s_u, r_v, \sigma, c_{u,v})$
- Some models encompass some processor heterogeneity (s_u, r_v) .
- Some models encompass some link heterogeneity $(\lambda_{u,v}, c_{u,v})$.

Which one is the more realistic?

strongly depends on

- network architecture
- program implementation (synchronous sends...)

Outline

- 1 Platform Modeling
- 2 Pipelining broadcasts: general framework**
- 3 Efficient algorithm: bidirectional one-port model
- 4 Decentralized Solutions (Awerbuch Leighton Algorithm)
- 5 Network Coding
- 6 Conclusion

Main theorem for solving pipelined problems

In general

- 1 a set of weighted allocation schemes: broadcast \leftrightarrow trees
- 2 a way to organize communications: matchings (unidirectional \leftrightarrow platform graph, bidirectional \leftrightarrow bipartite graph)

Theorem.

From a set of weighted trees $(\alpha_1, T_1) \dots (\alpha_T, T_T)$ and a set of weighted matchings $(x_1, \chi_1) \dots (x_X, \chi_X)$ such that

- $\forall (V_j, V_k) \in E, \left(\sum_{(j,k) \ni T_t} \alpha_t \right) c_{j,k} = \sum_{(j,k) \ni \chi_x} x_x$
- and $\sum_{\chi_x} x_x = 1,$

it is possible to build a periodic schedule achieving throughput $\sum_t \alpha_t$.
Time and size are polynomial in G, T and X .

Consequence

The solution of the following linear program

$$\begin{cases} \text{Maximize } \sum \alpha_t \\ \forall (V_j, V_k) \in E, & \left(\sum_{(j,k) \ni T_t} \alpha_t \right) c_{j,k} = \sum_{(j,k) \ni \chi_x} x_x \\ \sum_{\chi_x} x_x = 1 \\ \alpha_t \geq 0, \quad x_x \geq 0 \end{cases}$$

provides a periodic schedule of optimal throughput.

This LP can be solved in polynomial time

- using Ellipsoid Algorithm
- either under bidirectional or unidirectional models

Outline

- 1 Platform Modeling
- 2 Pipelining broadcasts: general framework
- 3 Efficient algorithm: bidirectional one-port model**
 - Set of matchings
 - Set of trees
- 4 Decentralized Solutions (Awerbuch Leighton Algorithm)
- 5 Network Coding
- 6 Conclusion

Main theorem for solving pipelined problems

Under bidirectional model

- 1 a set of weighted allocation schemes: broadcast \leftrightarrow trees
- 2 a way to organize communications: matchings \leftrightarrow bipartite graph

Theorem.

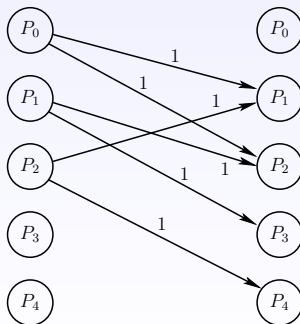
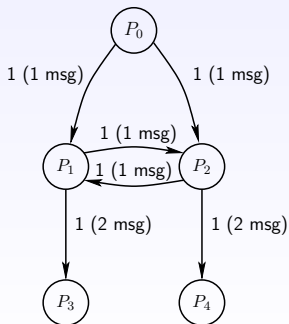
From a set of weighted trees $(\alpha_1, T_1) \dots (\alpha_T, T_T)$ such that

- $\forall V_j \in E, \left(\sum_{(j,k) \ni T_t} \alpha_t \right) c_{j,k} \leq 1,$
- $\forall V_j \in E, \left(\sum_{(k,j) \ni T_t} \alpha_t \right) c_{k,j} \leq 1,$

it is possible to build a periodic schedule achieving throughput $\sum_t \alpha_t$.
Time and size are polynomial in G, T and X .

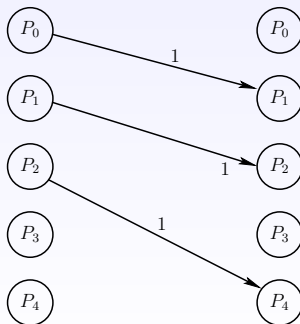
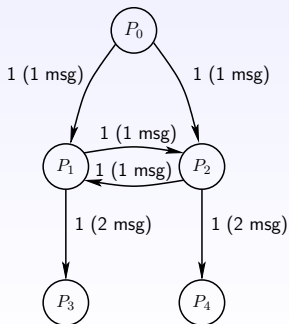
Set of matchings

- 1 Set of communications to execute within period T
- 2 One-port equations \rightarrow local constraints
- 3 Pairwise-disjoint communications to be scheduled simultaneously \Rightarrow extract a collection of matchings



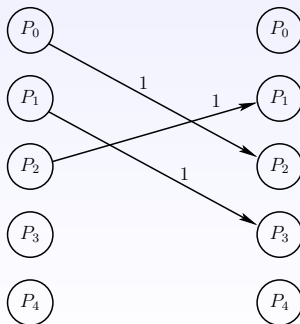
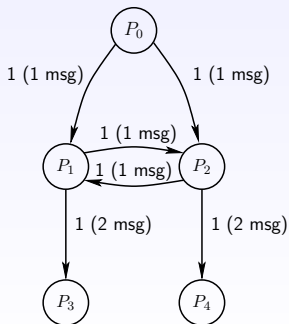
Set of matchings

- 1 Set of communications to execute within period T
- 2 One-port equations \rightarrow local constraints
- 3 Pairwise-disjoint communications to be scheduled simultaneously \Rightarrow extract a collection of matchings



Set of matchings

- 1 Set of communications to execute within period T
- 2 One-port equations \rightarrow local constraints
- 3 Pairwise-disjoint communications to be scheduled simultaneously \Rightarrow extract a collection of matchings



Set of matchings(2)

Solution

- Peel off bipartite communication graph
- **Idea:** Use Schrijver's weighted version of König's edge-coloring algorithm
 - ▶ extract a matching and subtract maximum weight from participating edges
 - ▶ zero out at least one edge for each matching
 - ▶ strongly polynomial
- Given the set of weighted trees and the set of matchings \Rightarrow we can build up the schedule

Optimal throughput: Linear Program (1)

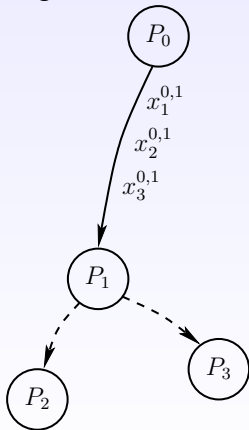
Best throughput: minimal time to broadcast a unit size message that can be arbitrarily split.

Step 1: Number of messages, no time, no congestion

$x_i^{j,k}$ denotes the fraction of the message from P_0 to P_i that uses edge (P_j, P_k)

The conditions are

- $\forall i, \sum_k x_i^{0,k} = 1$
- $\forall i, \sum_j x_i^{j,i} = 1$
- $\forall j \neq 0, i, \sum_k x_i^{j,k} = \sum_k x_i^{k,j}$



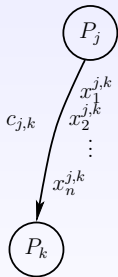
Optimal throughput: Linear Program (2)

Step 2: number of messages \Rightarrow time and congestion
 $t_{j,k}$ denotes the time to transfer all the messages
between P_j and P_k

- $t_{j,k} \leq \sum x_i^{j,k} c_{j,k}$????
- too pessimistic since $x_{i_1}^{j,k}$ and $x_{i_2}^{k,j}$ may be the same message
- not good for a lower bound

or

- $\forall i, t_{j,k} \leq x_i^{j,k} c_{j,k}$????
- too optimistic since it supposes that all the messages are sub-messages of the largest one
- OK for a lower bound, may not be feasible



Optimal throughput: Linear Program (3)

Step 3: one port constraints
one-port model, during one time unit

- at most one sending operation:
$$\sum_{(P_j, P_k) \in E} t_{j,k} \leq t_j^{out}$$
- at most one receiving operation:
$$\sum_{(P_k, P_j) \in E} t_{k,j} \leq t_j^{in}$$

and at last,

- $\forall j, \quad t_j^{out} \leq t^{broadcast}$
- $\forall j, \quad t_j^{in} \leq t^{broadcast}$

Optimal throughput: Linear Program (4)

MINIMIZE $t^{\text{broadcast}}$,

SUBJECT TO

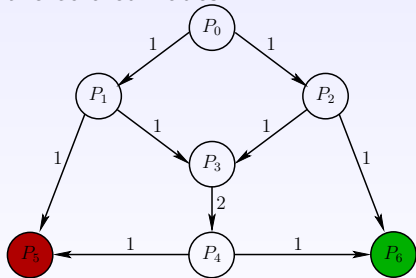
$$\left\{ \begin{array}{l} \forall i, \\ \forall i, \\ \forall i, \forall j \neq 0, i, \\ \forall i, j, k \\ \forall j, \\ \forall j, \\ \forall j, \\ \forall j, \end{array} \right. \begin{array}{l} \sum x_i^{0,k} = 1 \\ \sum x_i^{j,i} = 1 \\ \sum x_i^{j,k} = \sum x_i^{k,j} \\ t_{j,k} \leq x_i^{j,k} c_{j,k} \\ \sum_{(P_j, P_k) \in E} t_{j,k} \leq t_j^{\text{out}} \\ \sum_{(P_k, P_j) \in E} t_{k,j} \leq t_j^{\text{in}} \\ t_j^{\text{out}} \leq t^{\text{broadcast}} \\ t_j^{\text{in}} \leq t^{\text{broadcast}} \end{array}$$

Caveats

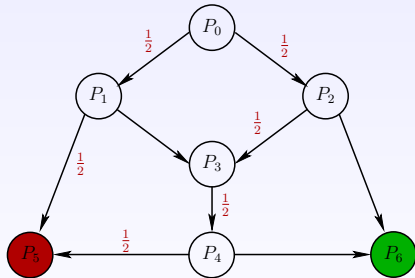
- The linear program provides a lower bound for the broadcasting time of a unit-size divisible message
- It is not obvious that this lower bound is feasible since we considered that all the messages using the same communication link are sub-messages of the largest one.

Caveats: Multicast Example (1)

Consider the following platform, where the multicast set consists in the colored nodes:

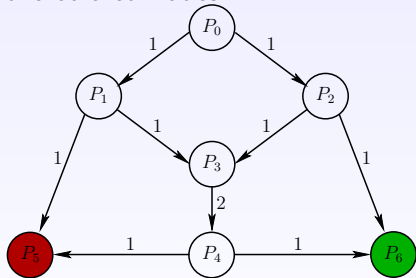


The linear program provides the following solution with throughput 1:

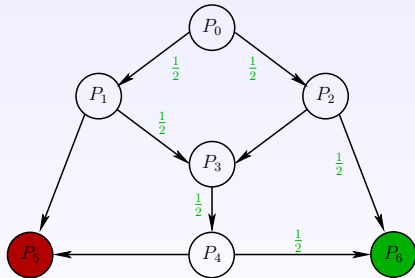


Caveats: Multicast Example (1)

Consider the following platform, where the multicast set consists in the colored nodes:

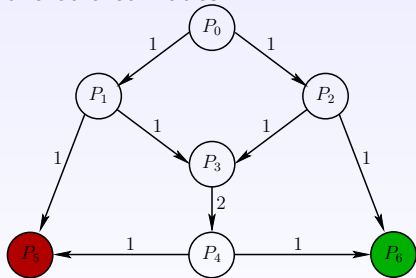


The linear program provides the following solution with throughput 1:

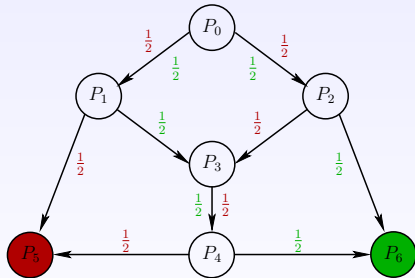


Caveats: Multicast Example (1)

Consider the following platform, where the multicast set consists in the colored nodes:

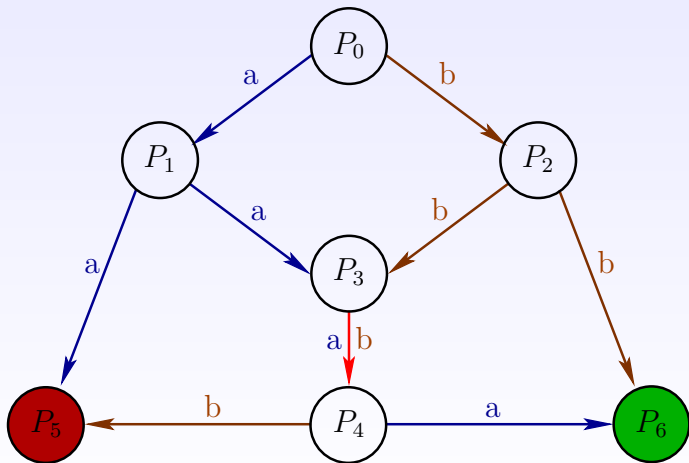


The linear program provides the following solution with throughput 1:



Caveats: Multicast Example (2)

Nevertheless, the obtained throughput is not feasible:



Set of trees

- $c(P_0, P_i)$ minimum weight to remove to disconnect = 1
- $c(P_0) = \min c(P_0, P_i) = 1$
- $n_{j,k} = \max_i \left\{ x_i^{j,k} \right\}$ is the fraction of messages through (P_j, P_k) .

Theorem (Weighted version of Edmond's branching Theorem).

Given a directed weighted $G = (V, E, n)$, $P_0 \in P$ the source we can find P_0 -rooted trees T_1, \dots, T_T and weights $\alpha_1, \dots, \alpha_T$ with $\sum \alpha_i \delta(T_i) \leq n$ with

$$\sum \alpha_i = c(P_0) = 1,$$

in strongly polynomial time, and $T \leq |E| + |V|^3$.

Set of trees

Theorem (Weighted version of Edmond's branching Theorem).

Given a directed weighted $G = (V, E, n)$, $P_0 \in P$ the source we can find P_0 -rooted trees T_1, \dots, T_T and weights $\alpha_1, \dots, \alpha_T$ with $\sum \alpha_i \delta(T_i) \leq n$ with

$$\sum \alpha_i = c(P_0) = 1,$$

in strongly polynomial time, and $T \leq |E| + |V|^3$.

This theorem provides:

- the set of trees, their weights
- and the number of trees is “low”: $\leq |E| + |V|^3$.
- thus we can derive an asymptotically optimal schedule

Outline

- 1 Platform Modeling
- 2 Pipelining broadcasts: general framework
- 3 Efficient algorithm: bidirectional one-port model
- 4 Decentralized Solutions (Awerbuch Leighton Algorithm)**
- 5 Network Coding
- 6 Conclusion

Steady state scheduling: good news and bad news

- 😊 Steady state scheduling: throughput maximization is much easier than makespan minimization and still realistic
- 😊 One-port model: first step towards designing realistic scheduling heuristics (other realistic models have been proposed in this context)
- 😊 Steady-state circumvents complexity of scheduling problems ... while deriving efficient (often asymptotically optimal) scheduling algorithms

Steady state scheduling: good news and bad news

- 😊 Steady state scheduling: throughput maximization is much easier than makespan minimization and still realistic
- 😊 One-port model: first step towards designing realistic scheduling heuristics (other realistic models have been proposed in this context)
- 😊 Steady-state circumvents complexity of scheduling problems ... while deriving efficient (often asymptotically optimal) scheduling algorithms

- 😞 Memory constraints, latency, period size may be large...
- 😞 Need to acquire a good knowledge of the platform graph (ENV, Alnem, NWS...)
- 😞 Taking into account changes in resource performances is still difficult: build super-steps and recompute optimal solution at the end of each super-step...

Dynamic platforms

On large scale distributed systems:

- resource performances may change over time (resource sharing, node may appear and disappear)
- impossible to maintain a coherent snapshot of the platform at a given node and recompute optimal solution
- using fully greedy dynamic scheduling algorithms is known to lead to bad results
- inject some static knowledge into dynamic schedulers

Taking dynamic performances into account

Need for decentralized and robust scheduling algorithms based on static knowledge

What do robust and dynamic mean?

Need for metrics in order to analyze algorithms

Robust

- If $\text{THROUGHPUT}(t)$ denotes the optimal throughput for platform at time t and $\text{TIME}(N)$ denotes the time to process N tasks using proposed scheduling algorithm
- The objective is

$$(N) / \int_{t=0}^{\text{TIME}(N)} \text{THROUGHPUT}(t) dt \xrightarrow{N \rightarrow +\infty} 1$$

Decentralized

at any time step, a node makes its decisions according to

- its state (local memory)
- the states of its immediate neighbors

Fluid relaxation (cont'd!)

- Throughput maximization
 - ▶ concentrate on steady state
 - ▶ define activity variables
 - ▶ then, rebuild allocations and schedule

- Dynamic platforms:
 - ▶ put messages in different queues
 - ▶ define potential functions associated to those queues
 - ▶ let messages move "by themselves" from high to low potentials
 - ▶ areas where messages are retrieved quickly will become low potential areas
 - ▶ areas where messages reach destinations slowly will become high potential areas

Example: broadcast

For the sake of simplicity, we will assume that

- that $\rho_{\min} = \min \text{THROUGHPUT}(t)$ is known
- and we will prove that

$$\left(\frac{N}{\text{TIME}(N)} \right) \geq \rho_{\min}.$$

Example: broadcast

For the sake of simplicity, we will assume that

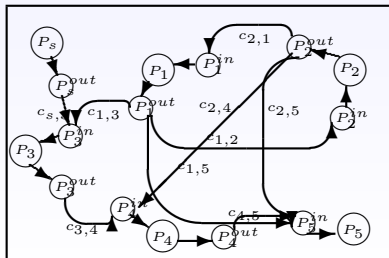
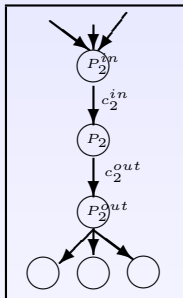
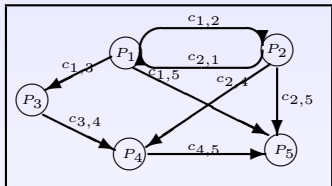
- that $\rho_{\min} = \min \text{THROUGHPUT}(t)$ is known
- and we will prove that

$$\left(\frac{N}{\text{TIME}(N)} \right) \geq \rho_{\min}.$$

- We will consider a slightly different communication model (see next slide)
- In fact, with more care, we can prove

$$N \geq \sum_{i=0}^{\text{TIME}(N)} \int_{t=i}^{i+1} \text{THROUGHPUT}(t) dt$$

System Model



Credits

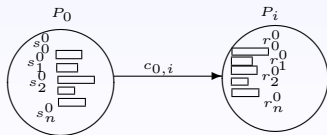
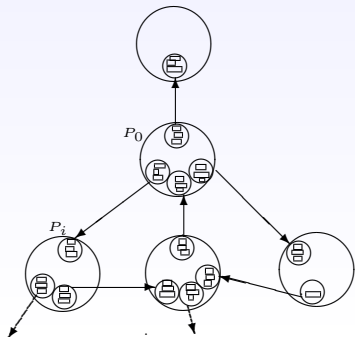
based on an algorithm for multi-commodity flows (Awerbuch Leighton)

- if we denote (as previously) by $x_i^{j,k}$ the fraction of the message for P_i that is shipped along P_j, P_k
- the $x_i^{*,*}$ define a flow between P_s and P_i
- the overall problem is not a multi-commodity flow problem since (as we have seen) flow do not sum up but rather max on the edges
- in fact, this is slightly more complicated since flows sum up on incoming and outgoing edges and max on regular edges...
- Awerbuch-Leighton algorithm must be adapted to this condition.

Queues (1)

Queues at intermediate nodes

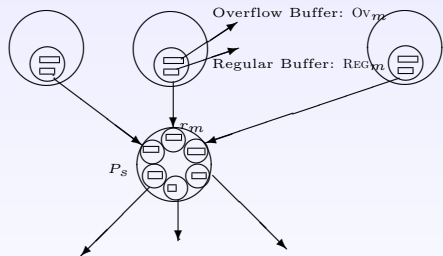
- each node P_0 stores non-shipped messages in $NBH \times N$ queues, where NBH denotes the neighbors of P_0 and N the overall number of nodes.
- each node P_i has a queue for incoming messages for each destination (from its neighbor)



Queues (2)

Queues at source node

- the source node is split into $n + 1$ parts.
- the upper source nodes (1 per commodity) hold a regular buffer and an overflow buffer
- the overflow buffer holds tasks that do not fit in the regular buffer
- the lower source node works like any other node



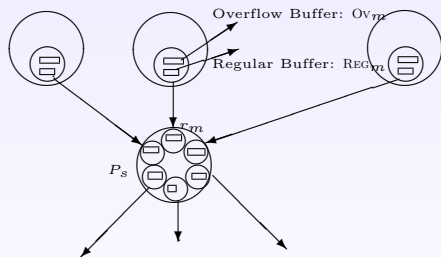
Queues and potential functions

- Each queue (regular or overflow) is associated with an increasing (with the size of the queue) potential function
- The potential of an edge is the sum of queue potentials at the tail and head of the edge.
- Nodes try to minimize their potential, given resource constraints (bandwidth).
- Thus, tasks go from high potential to low potential.

Potential functions

Potential functions at source node

- The potential associated to the overflow buffer of size OV_m is $\sigma(OV_m) = OV_m \alpha \exp(\alpha Q)$.
- The potential associated to the regular buffer of size REG_m is $\Phi(REG_m) = \exp(\alpha REG_m)$.
- where α is a constant and Q is the maximal size of the regular buffer (both depending on the network and the expected throughput).



Potential functions

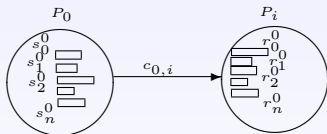
Potential functions at regular nodes

- The potential associated to a regular buffer of size s is

$$\Phi(s) = \exp(\alpha s).$$

- The potential associated to the edge (P_0, P_i) is

$$\Phi(P_0, P_i) = \sum_1^n \exp(\alpha s_k^0) + \exp(\alpha r_k^0).$$



Overall Algorithm

Time is divided in rounds, each round consists in 4 steps

- **Phase 1:** At each upper source node, add $(1 - \varepsilon)\rho_{\min}$ messages to the overflow queue. Then move as many messages as possible from the overflow queue to the regular queue (given maximum height constraint)
- **Phase 2:** For each edge (P_i, P_j) , push messages across the edge so as to minimize the overall potential of the edge (P_i, P_j) without violating capacity constraint.
- **Phase 3:** At P_i , empty all the queues corresponding to messages for P_i
- **Phase 4:** At each node P_i , for each destination P_j , re-balance the queues so that all queues corresponding to P_j at P_i have same size.

How to minimize potential for (P_0, P_i)

- The potential associated to edge (P_0, P_i) is
$$\Phi(P_0, P_i) = \sum_1^n \exp(\alpha s_k^0) + \exp(\alpha r_k^0).$$

- Satisfying capacity constraint:

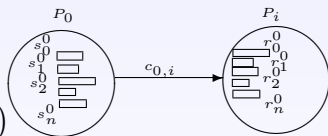
Minimize

$$\sum_k (\exp(\alpha(s_k^0 - f)) + \exp(\alpha(r_k^0 + f)))$$





so that


$$f \geq 0 \text{ (directed edge)}$$



$$f c_{0,i} \leq 1 \text{ (capacity constraint)}$$



Potential analysis during one round

- **Phase 1:** At each upper source node, add $(1 - \varepsilon)\rho_{\min}$ messages to the overflow queue. Then move as many messages as possible from the overflow queue to the regular queue (given maximum height constraint) 
- **Phase 2:** For each edge (P_i, P_j) , push messages across the edge so as to minimize the overall potential of the edge (P_i, P_j) without violating capacity constraint. 
- **Phase 3:** At P_i , empty all the queues corresponding to messages for P_i 
- **Phase 4:** At each node P_i , for each destination P_j , re-balance the queues so that all queues corresponding to P_j at P_i have same size. 

Potential  during phase 1 can be evaluated easily

Potential  during phases 2-4 strongly depend on local queue sizes. 

Potential analysis during one round

Sketch of the proof

Analyzing directly potential decrease during phase 2 is difficult, but

- we "know" that there exists a solution with throughput ρ_{\min}
- since potential minimization is optimal (given resource constraint) during Phase 2
- \implies the potential decrease during Phase 2 is at least the potential decrease that would be induced by the solution with throughput ρ_{\min}
- the potential decrease that would be induced by the solution with throughput ρ_{\min} can be determined easily

\implies we get a lower bound for potential decrease during Phase 2 (and neglect potential decreases during Phases 3-4)

Sketch of the proof (end!)

- Using above technique, we can prove that the overall potential remains bounded
- \implies the overall number of non-shipped messages in the network remains bounded
- Since we inject $(1 - \varepsilon)\rho_{\min}$ tasks at each round, this means that almost all tasks have been processed

\implies the overall throughput is optimal
(almost, due to ε , that can be chosen arbitrarily small)

Conclusion on Awerbuch Leighton algorithm

- 😊 fully decentralized
- 😊 performance guarantee
- 😊 general framework
- 😞 ρ_{\min} ???
- 😞 lost flow?
- 😞 message granularity???

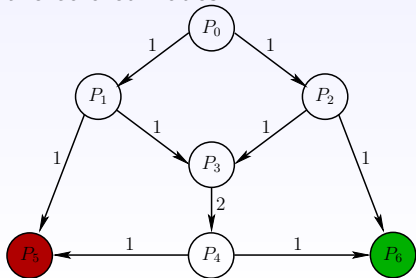
Outline

- 1 Platform Modeling
- 2 Pipelining broadcasts: general framework
- 3 Efficient algorithm: bidirectional one-port model
- 4 Decentralized Solutions (Awerbuch Leighton Algorithm)
- 5 Network Coding**
- 6 Conclusion

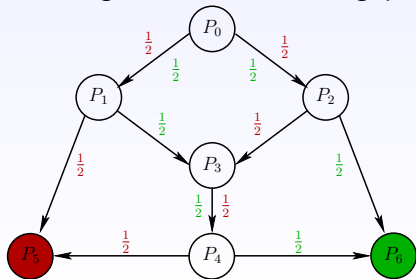
Introduction

- if ☹️'s find answers, AL algorithm answers the size of the message to be sent on each edge...
- but not what to send!
- Network coding may be the answer (first designed to circumvent multicast complexity)

Consider the following platform, where the multicast set consists in the colored nodes:

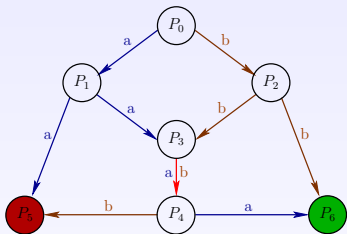


The linear program provides the following solution with throughput 1:



Network Coding

Nevertheless, the obtained throughput is not feasible:



- Unless we change the rule of the game and send $\text{xor}(a, b)$ between P_3 and P_4 !
- Nodes P_5 and P_6 will be responsible for decoding the message

Network Coding – Centralized Version

- initial message is split into blocks B_1, \dots, B_N
- blocks are seen as elements of the field \mathbb{F}_{2^m}
- each node receives linear combinations of blocks $\sum \alpha_i B_i \in \mathbb{F}_{2^m}$.
- each node builds linear combinations of received messages $\in \mathbb{F}_{2^m}$, then transfer them.
- if destination nodes receive N linear combinations, and resulting matrix transfer is non-singular, then it can build the initial message!

Centralized Solution for ensuring non-singularity

- Let $\beta_{i,j}^k$'s denote the coefficients of linear combinations on the different edges
- The product of the determinant of all transfer matrices is a multi-variate polynomial in the $\beta_{i,j}^k$'s.
- If the field where $\beta_{i,j}^k$'s are chosen is "sufficiently large", there exist a choice of the $\beta_{i,j}^k$'s so that the product of determinants is not zero.
- In this case, if destination nodes receive N linear combinations, and resulting matrix transfer is non-singular, then it can build the initial message!

Decentralized Solution for ensuring non-singularity with high probability

- Let $\beta_{i,j}^k$'s denote the coefficients of linear combinations on the different edges
- The product of the determinant of all transfer matrices is a multi-variate polynomial in the $\beta_{i,j}^k$'s.
- If the field where $\beta_{i,j}^k$'s are chosen is "sufficiently large", for any choice of the $\beta_{i,j}^k$'s, with high probability, the product of determinants is not zero.
- In this case, if destination nodes receive N linear combinations, and resulting matrix transfer is non-singular, then it can build the initial message!
- Remark: $\beta_{i,j}^k$ coefficients must be sent together with the messages...

Conclusion on Network Coding

- 😊 fully decentralized
- 😊 Together with Awerbuch Leighton algorithm
 - ▶ Awerbuch Leighton algorithm tells how much should be sent
 - ▶ Network Coding says what to send
- 😞 's mentioned before for Awerbuch Leighton algorithm
- 😞 what if the matrix is singular?
- 😞 the bound for "sufficiently large" field are very bad?
- 😞 how to take decoding cost into account?
- 😞 messages may arrive in any order (and even not at all...) and still need to be stored before decoding. What about memory constraints?

Outline

- 1 Platform Modeling
- 2 Pipelining broadcasts: general framework
- 3 Efficient algorithm: bidirectional one-port model
- 4 Decentralized Solutions (Awerbuch Leighton Algorithm)
- 5 Network Coding
- 6 Conclusion**

Conclusion

😊 Still plenty of work to do before practical implementation

Who wants to join?