Complexity results and heuristics for pipelined multicast operations on heterogeneous platforms

Olivier Beaumont, Arnaud Legrand, Loris Marchal, Yves Robert

Laboratoire de l'Informatique du Parallélisme
École Normale Supérieure de Lyon, France

ICPP - August 2004

Outline

Outline

Outline

Outline

Outline

Outline

Introduction

- Complex applications on grid environment require collective communications, in particular broadcast
- multicast = broadcast to a strict subset of targets in the platform nodes
- Numerous studies of multicast:
  - Steiner trees (minimize the cost of a single multicast tree, NP hard problem)
  - for a wide variety of particular architectures and technologies (wormhole-routed, wireless, ad-hoc, optical networks)

Introduction

- Complex applications on grid environment require collective communications, in particular broadcast
- multicast = broadcast to a strict subset of targets in the platform nodes
- Numerous studies of multicast:
  - Steiner trees (minimize the cost of a single multicast tree, NP hard problem)
  - for a wide variety of particular architectures and technologies (wormhole-routed, wireless, ad-hoc, optical networks)

Introduction

- Complex applications on grid environment require collective communications, in particular broadcast
- multicast = broadcast to a strict subset of targets in the platform nodes
- Numerous studies of multicast:
  - Steiner trees (minimize the cost of a single multicast tree, NP hard problem)
  - for a wide variety of particular architectures and technologies (wormhole-routed, wireless, ad-hoc, optical networks)

Introduction

- Complex applications on grid environment require collective communications, in particular broadcast
- multicast = broadcast to a strict subset of targets in the platform nodes
- Numerous studies of multicast:
  - Steiner trees (minimize the cost of a single multicast tree, NP hard problem)
  - for a wide variety of particular architectures and technologies (wormhole-routed, wireless, ad-hoc, optical networks)

Introduction

- Complex applications on grid environment require collective communications, in particular broadcast
- multicast = broadcast to a strict subset of targets in the platform nodes
- Numerous studies of multicast:
  - Steiner trees (minimize the cost of a single multicast tree, NP hard problem)
  - for a wide variety of particular architectures and technologies (wormhole-routed, wireless, ad-hoc, optical networks)

Introduction

- Pipelining communications:
  - data parallelism involves a large amount of data
  - not a single communication, but series of same communication schemes (e.g. series of multicasts from same source)
  - maximize throughput of steady-state operation
- with the same framework as in previous work for other collective communications:

$$\text{scatter, reduce, broadcast} \Rightarrow \begin{cases} \text{optimal throughput,} \\ \text{asymptotically optimal algorithms} \end{cases}$$

- surprisingly, multicast turned out to be more challenging

Introduction

- Pipelining communications:
  - data parallelism involves a large amount of data
  - not a single communication, but series of same communication schemes (e.g. series of multicasts from same source)
  - maximize throughput of steady-state operation
- with the same framework as in previous work for other collective communications:

$$\text{scatter, reduce, broadcast} \Rightarrow \begin{cases} \text{optimal throughput,} \\ \text{asymptotically optimal algorithms} \end{cases}$$

- surprisingly, multicast turned out to be more challenging

Introduction

- Pipelining communications:
  - data parallelism involves a large amount of data
  - not a single communication, but series of same communication schemes (e.g. series of multicasts from same source)
  - maximize throughput of steady-state operation
- with the same framework as in previous work for other collective communications:

$$\text{scatter, reduce, broadcast} \Rightarrow \begin{cases} \text{optimal throughput,} \\ \text{asymptotically optimal algorithms} \end{cases}$$

- surprisingly, multicast turned out to be more challenging

Introduction

- Pipelining communications:
  - data parallelism involves a large amount of data
  - not a single communication, but series of same communication schemes (e.g. series of multicasts from same source)
  - maximize throughput of steady-state operation
- with the same framework as in previous work for other collective communications:

$$\text{scatter, reduce, broadcast} \Rightarrow \begin{cases} \text{optimal throughput,} \\ \text{asymptotically optimal algorithms} \end{cases}$$

- surprisingly, multicast turned out to be more challenging

Introduction

- Pipelining communications:
  - data parallelism involves a large amount of data
  - not a single communication, but series of same communication schemes (e.g. series of multicasts from same source)
  - maximize throughput of steady-state operation
- with the same framework as in previous work for other collective communications:

$$\text{scatter, reduce, broadcast} \Rightarrow \begin{cases} \text{optimal throughput,} \\ \text{asymptotically optimal algorithms} \end{cases}$$

- surprisingly, multicast turned out to be more challenging

Introduction

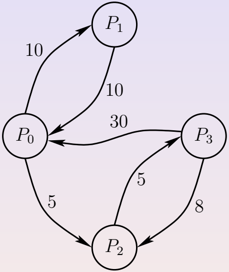
- Pipelining communications:
  - data parallelism involves a large amount of data
  - not a single communication, but series of same communication schemes (e.g. series of multicasts from same source)
  - maximize throughput of steady-state operation
- with the same framework as in previous work for other collective communications:

$$\text{scatter, reduce, broadcast} \Rightarrow \begin{cases} \text{optimal throughput,} \\ \text{asymptotically optimal algorithms} \end{cases}$$

- surprisingly, multicast turned out to be more challenging

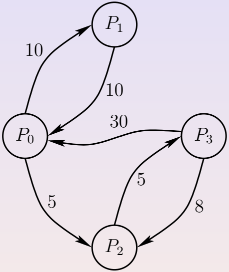
Platform Model

- $G = (V, E, c)$
- $P_1, P_2, \ldots, P_n$: processors
- P_{source} : processor initiating the multicast
- $\mathcal{P}_{\text{target}}$: set of target processors
- $(j,k) \in E$: communication link between P_i and P_j
- $c(j,k)$: time needed to transfer one unit message from P_j to P_k
- one-port model for incoming communications
- one-port model for outgoing communications



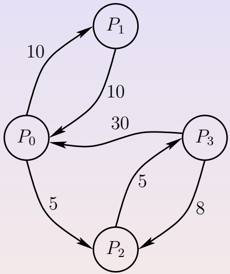
Platform Model

- $G = (V, E, c)$
- $P_1, P_2, \ldots, P_n$: processors
- P_{source} : processor initiating the multicast
- $\mathcal{P}_{\text{target}}$: set of target processors
- $(j, k) \in E$: communication link between P_i and P_j
- $c(j, k)$: time needed to transfer one unit message from P_j to P_k
- one-port model for incoming communications
- one-port model for outgoing communications



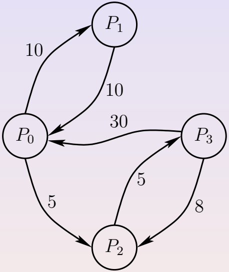
Platform Model

- $G = (V, E, c)$
- $P_1, P_2, \ldots, P_n$: processors
- $P_{\textsf{source}}$: processor initiating the multicast
- $\mathcal{P}_{\textsf{target}}$: set of target processors
- $(j, k) \in E$: communication link between P_i and P_j
- $c(j, k)$: time needed to transfer one unit message from P_j to P_k
- one-port model for incoming communications
- one-port model for outgoing communications



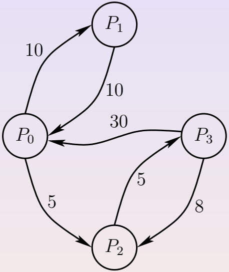
Platform Model

- $G = (V, E, c)$
- $P_1, P_2, \ldots, P_n$: processors
- P_{source} : processor initiating the multicast
- $\mathcal{P}_{\text{target}}$: set of target processors
- $(j, k) \in E$: communication link between P_i and P_j
- $c(j, k)$: time needed to transfer one unit message from P_j to P_k
- one-port model for incoming communications
- one-port model for outgoing communications



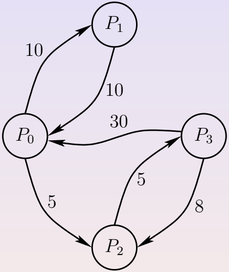
Platform Model

- $G = (V, E, c)$
- $P_1, P_2, \ldots, P_n$: processors
- P_{source} : processor initiating the multicast
- $\mathcal{P}_{\text{target}}$: set of target processors
- $(j, k) \in E$: communication link between P_i and P_j
- $c(j, k)$: time needed to transfer one unit message from P_j to P_k
- one-port model for incoming communications
- one-port model for outgoing communications



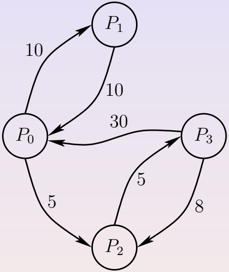
Platform Model

- $G = (V, E, c)$
- $P_1, P_2, \ldots, P_n$: processors
- P_{source} : processor initiating the multicast
- $\mathcal{P}_{\text{target}}$: set of target processors
- $(j, k) \in E$: communication link between P_i and P_j
- $c(j, k)$: time needed to transfer one unit message from P_j to P_k
- one-port model for incoming communications
- one-port model for outgoing communications



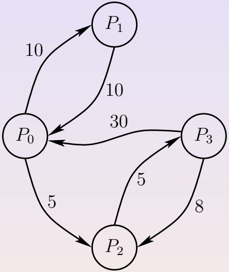
Platform Model

- $G = (V, E, c)$
- $P_1, P_2, \ldots, P_n$: processors
- P_{source} : processor initiating the multicast
- $\mathcal{P}_{\text{target}}$: set of target processors
- $(j, k) \in E$: communication link between P_i and P_j
- $c(j, k)$: time needed to transfer one unit message from P_j to P_k
- one-port model for incoming communications
- one-port model for outgoing communications



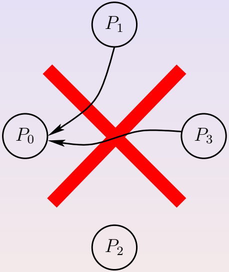
Platform Model

- $G = (V, E, c)$
- $P_1, P_2, \ldots, P_n$: processors
- P_{source} : processor initiating the multicast
- $\mathcal{P}_{\text{target}}$: set of target processors
- $(j, k) \in E$: communication link between P_i and P_j
- $c(j, k)$: time needed to transfer one unit message from P_j to P_k
- one-port model for incoming communications
- one-port model for outgoing communications



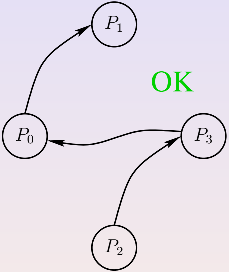
Platform Model

- $G = (V, E, c)$
- $P_1, P_2, \ldots, P_n$: processors
- P_{source} : processor initiating the multicast
- $\mathcal{P}_{\text{target}}$: set of target processors
- $(j, k) \in E$: communication link between P_i and P_j
- $c(j, k)$: time needed to transfer one unit message from P_j to P_k
- one-port model for incoming communications
- one-port model for outgoing communications



Platform Model

- $G = (V, E, c)$
- $P_1, P_2, \ldots, P_n$: processors
- P_{source} : processor initiating the multicast
- $\mathcal{P}_{\text{target}}$: set of target processors
- $(j, k) \in E$: communication link between P_i and P_j
- $c(j, k)$: time needed to transfer one unit message from P_j to P_k
- one-port model for incoming communications
- one-port model for outgoing communications



Outline

Framework for pipelined Broadcast and Scatter

1. express optimization problem as set of linear constraints (variables = fraction of time a processor spends sending to another during one time-unit)
2. solve linear program (in rational numbers)
3. use solution to build periodic schedule reaching best throughput

two preliminaries operations:

- *Scatter:* P_{source} sends different messages to each target in $\mathcal{P}_{\text{target}}$
- *Broadcast:* P_{source} sends the same messages to every processor

Framework for pipelined Broadcast and Scatter

1. express optimization problem as set of linear constraints (variables = fraction of time a processor spends sending to another during one time-unit)
2. solve linear program (in rational numbers)
3. use solution to build periodic schedule reaching best throughput

two preliminaries operations:

- *Scatter*: P_{source} sends different messages to each target in $\mathcal{P}_{\text{target}}$
- *Broadcast*: P_{source} sends the same messages to every processor

Framework for pipelined Broadcast and Scatter

1. express optimization problem as set of linear constraints (variables = fraction of time a processor spends sending to another during one time-unit)
2. solve linear program (in rational numbers)
3. use solution to build periodic schedule reaching best throughput

two preliminaries operations:

- *Scatter*: P_{source} sends different messages to each target in $\mathcal{P}_{\text{target}}$
- *Broadcast*: P_{source} sends the same messages to every processor

Framework for pipelined Broadcast and Scatter

1. express optimization problem as set of linear constraints (variables = fraction of time a processor spends sending to another during one time-unit)
2. solve linear program (in rational numbers)
3. use solution to build periodic schedule reaching best throughput

two preliminaries operations:

- *Scatter*: P_{source} sends different messages to each target in $\mathcal{P}_{\text{target}}$
- *Broadcast*: P_{source} sends the same messages to every processor

Framework for pipelined Broadcast and Scatter

1. express optimization problem as set of linear constraints (variables = fraction of time a processor spends sending to another during one time-unit)
2. solve linear program (in rational numbers)
3. use solution to build periodic schedule reaching best throughput

two preliminaries operations:

- *Scatter*: P_{source} sends different messages to each target in $\mathcal{P}_{\text{target}}$
- *Broadcast*: P_{source} sends the same messages to every processor

Framework for pipelined Broadcast and Scatter

1. express optimization problem as set of linear constraints (variables = fraction of time a processor spends sending to another during one time-unit)
2. solve linear program (in rational numbers)
3. use solution to build periodic schedule reaching best throughput

two preliminaries operations:

- *Scatter*: P_{source} sends different messages to each target in $\mathcal{P}_{\text{target}}$
- *Broadcast*: P_{source} sends the same messages to every processor

Linear constraints for broadcast and scatter

- variables: average quantities during one time-unit
- express constraints on these quantities:
  - bounded capacities on links:

$$(\text{data sent through } i \to j) \times c(i,j) \leqslant 1$$

  - one-port assumptions:

$$\text{time spent by } P_i \text{ sending (or receiving) data} \leqslant 1$$

  - conservation laws:

Linear constraints for broadcast and scatter

- variables: average quantities during one time-unit
- express constraints on these quantities:
  - bounded capacities on links:

$$(\text{data sent through } i \to j) \times c(i,j) \leqslant 1$$

  - one-port assumptions:

$$\text{time spent by } P_i \text{ sending (or receiving) data} \leqslant 1$$

  - conservation laws:

Linear constraints for broadcast and scatter

- variables: average quantities during one time-unit
- express constraints on these quantities:
  - bounded capacities on links:

$$(\text{data sent through } i \rightarrow j) \times c(i,j) \leqslant 1$$

  - one-port assumptions:

$$\text{time spent by } P_i \text{ sending (or receiving) data} \leqslant 1$$

  - conservation laws:

Linear constraints for broadcast and scatter

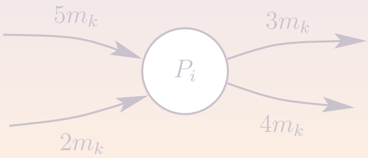
- variables: average quantities during one time-unit
- express constraints on these quantities:
  - bounded capacities on links:

$$(\text{data sent through } i \to j) \times c(i,j) \leqslant 1$$

  - one-port assumptions:

$$\text{time spent by } P_i \text{ sending (or receiving) data} \leqslant 1$$

  - conservation laws:



Linear constraints for broadcast and scatter

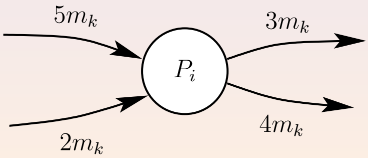
- variables: average quantities during one time-unit
- express constraints on these quantities:
  - bounded capacities on links:

    $$(\text{data sent through } i \rightarrow j) \times c(i,j) \leqslant 1$$

  - one-port assumptions:

    $$\text{time spent by } P_i \text{ sending (or receiving) data} \leqslant 1$$

  - conservation laws:



Linear Program for Broadcast and Scatter

previous linear programs gives, for *Scatter* and *Broadcast*:

- optimal throughput,
- number of messages of each type on each link,
- occupation time of each edge

Using classical graph techniques, we orchestrate communications and derive a periodic schedule.

Linear Program for Broadcast and Scatter

previous linear programs gives, for *Scatter* and *Broadcast*:

- optimal throughput,
- number of messages of each type on each link,
- occupation time of each edge

Using classical graph techniques, we orchestrate communications and derive a periodic schedule.

Linear Program for Broadcast and Scatter

previous linear programs gives, for *Scatter* and *Broadcast*:

- optimal throughput,
- number of messages of each type on each link,
- occupation time of each edge

Using classical graph techniques, we orchestrate communications and derive a periodic schedule.

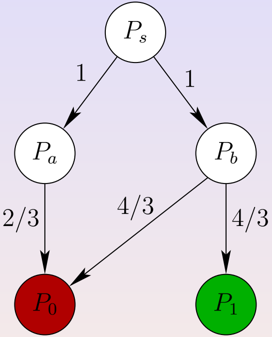
Linear Program for Broadcast and Scatter

previous linear programs gives, for *Scatter* and *Broadcast*:

- optimal throughput,
- number of messages of each type on each link,
- occupation time of each edge

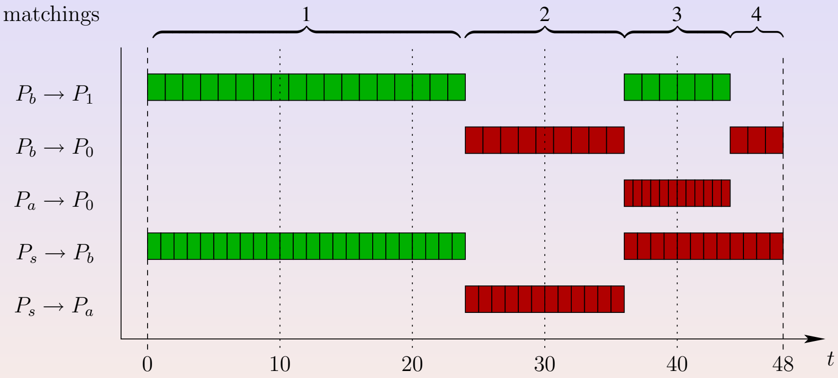
Using classical graph techniques, we orchestrate communications and derive a periodic schedule.

Toy example



Topology

Toy example



Periodic schedule obtained from the solution of the linear program

Outline

Complexity result

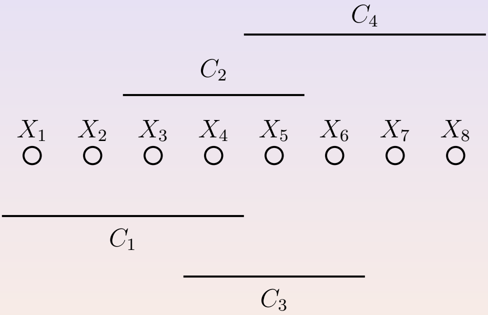
Theorem.

Computing the best throughput for a multicast operation on a given platform is NP-hard

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?

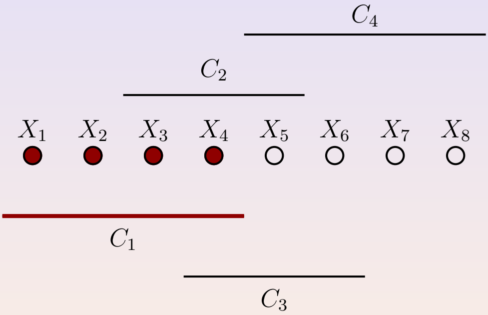
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



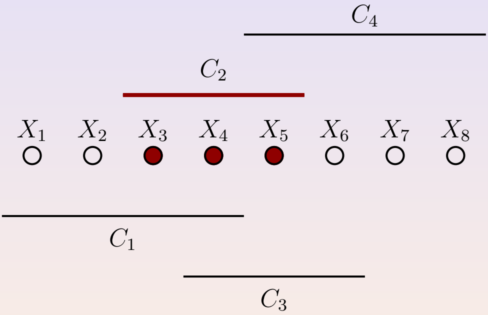
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



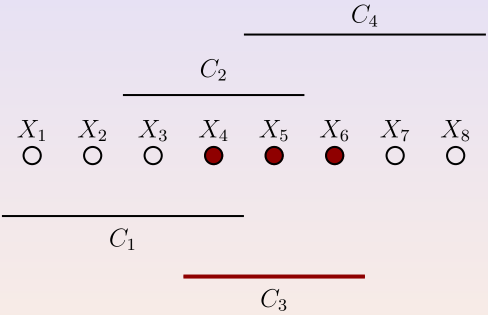
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



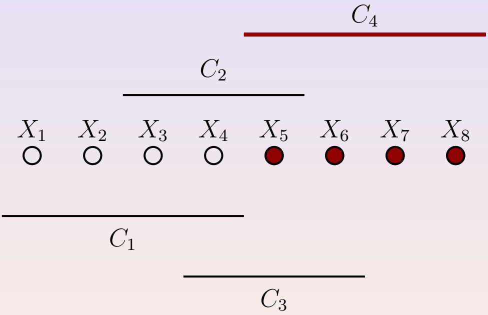
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



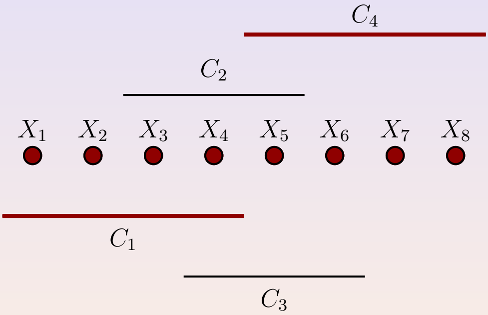
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



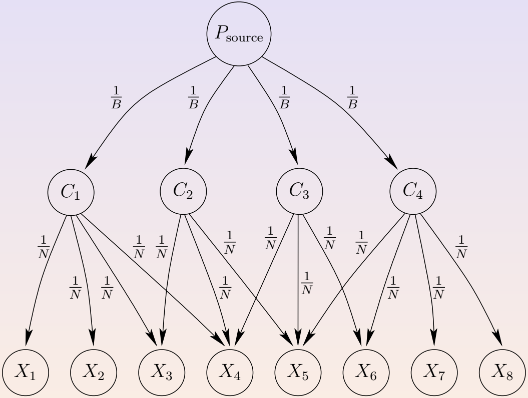
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



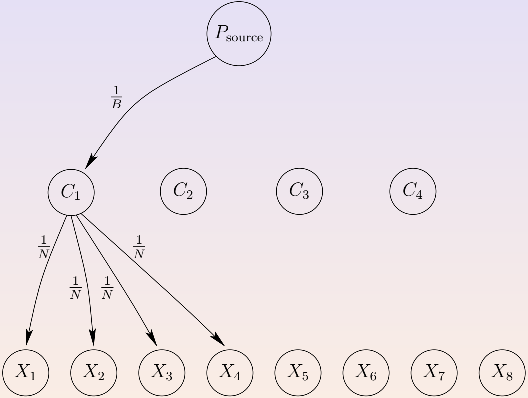
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



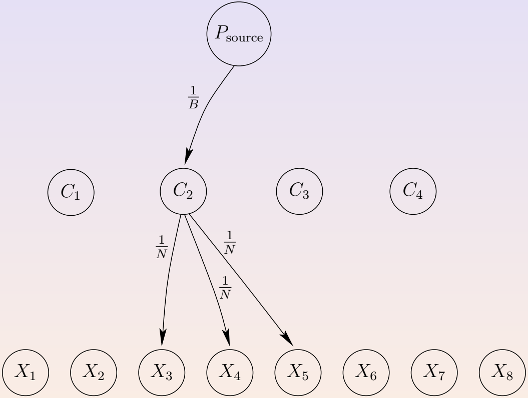
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



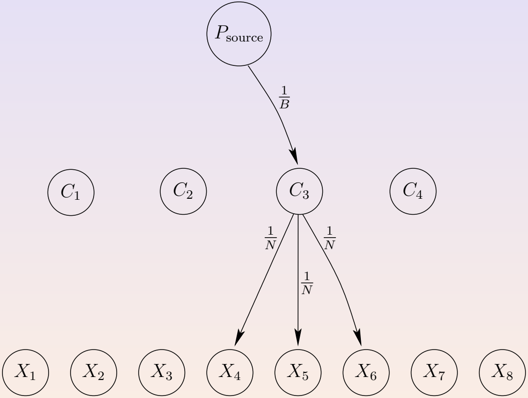
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



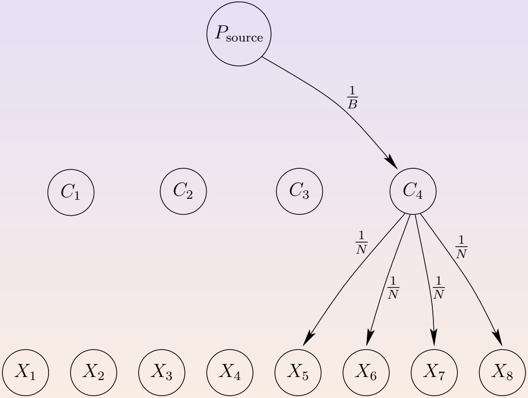
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



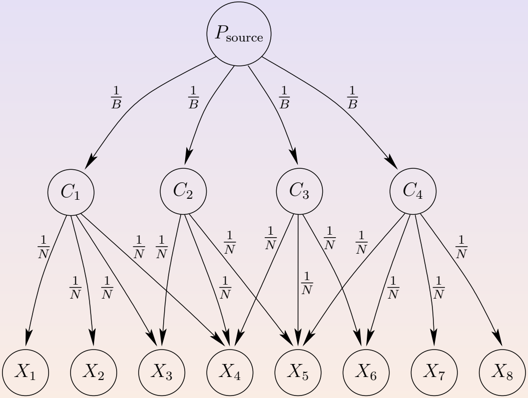
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



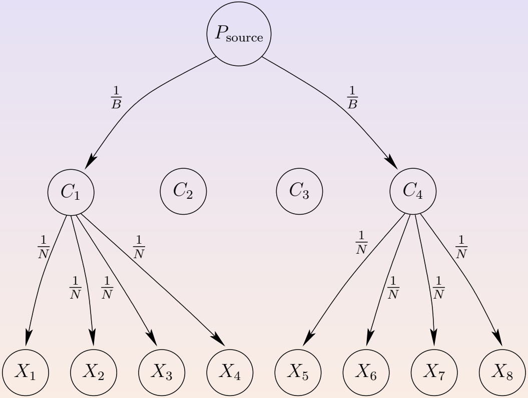
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



Outline

Straightforward heuristics

Optimal solutions (obtained by linear programming) on Scatter and Broadcast problem give two heuristics:

- *scatter:*
  - forget that messages sent to different targets are the same
  - scatter has a guarantee factor of $|\mathcal{P}_{\text{target}}|$:

$$\text{throughput(scatter)} \geqslant \frac{\text{optimal throughput}}{\text{number of targets}}$$

- *broadcast:*
  consider each node is a target, broadcast messages on the whole platform

Straightforward heuristics

Optimal solutions (obtained by linear programming) on Scatter and Broadcast problem give two heuristics:

- *scatter:*
  - forget that messages sent to different targets are the same
  - scatter has a guarantee factor of $|\mathcal{P}_{\text{target}}|$:

$$\text{throughput}(\text{scatter}) \geqslant \frac{\text{optimal throughput}}{\text{number of targets}}$$

- *broadcast:*
  consider each node is a target, broadcast messages on the whole platform

Refined heuristics

- *Reduced Broadcast*:
  1. compute the solution of the broadcast
  2. choose the node P_{min} which forwards the minimum of messages to the targets
  3. delete this vertex from the platform graph and start again until the throughput is not improved
- *Augmented Multicast*:
  1. compute the solution of the scatter
  2. choose the node P_{max} which forwards the maximum of messages to the targets
  3. add this node to $\mathcal{P}_{\text{target}}$ if it improves the throughput of a broadcast on the set of nodes $\{P_{\text{source}}\} \cup \mathcal{P}_{\text{target}}$

Refined heuristics

- *Reduced Broadcast*:
  1. compute the solution of the broadcast
  2. choose the node P_{min} which forwards the minimum of messages to the targets
  3. delete this vertex from the platform graph and start again until the throughput is not improved
- *Augmented Multicast*:
  1. compute the solution of the scatter
  2. choose the node P_{max} which forwards the maximum of messages to the targets
  3. add this node to $\mathcal{P}_{\text{target}}$ if it improves the throughput of a broadcast on the set of nodes $\{P_{\text{source}}\} \cup \mathcal{P}_{\text{target}}$

Refined heuristics

- *Reduced Broadcast*:
  1. compute the solution of the broadcast
  2. choose the node P_{min} which forwards the minimum of messages to the targets
  3. delete this vertex from the platform graph and start again until the throughput is not improved
- *Augmented Multicast*:
  1. compute the solution of the scatter
  2. choose the node P_{max} which forwards the maximum of messages to the targets
  3. add this node to $\mathcal{P}_{\text{target}}$ if it improves the throughput of a broadcast on the set of nodes $\{P_{\text{source}}\} \cup \mathcal{P}_{\text{target}}$

Refined heuristics

- *Reduced Broadcast*:
  1. compute the solution of the broadcast
  2. choose the node P_{min} which forwards the minimum of messages to the targets
  3. delete this vertex from the platform graph and start again until the throughput is not improved
- *Augmented Multicast*:
  1. compute the solution of the scatter
  2. choose the node P_{max} which forwards the maximum of messages to the targets
  3. add this node to $\mathcal{P}_{\text{target}}$ if it improves the throughput of a broadcast on the set of nodes $\{P_{\text{source}}\} \cup \mathcal{P}_{\text{target}}$

Refined heuristics

- *Reduced Broadcast*:
  1. compute the solution of the broadcast
  2. choose the node P_{min} which forwards the minimum of messages to the targets
  3. delete this vertex from the platform graph and start again until the throughput is not improved
- *Augmented Multicast*:
  1. compute the solution of the scatter
  2. choose the node P_{max} which forwards the maximum of messages to the targets
  3. add this node to $\mathcal{P}_{\text{target}}$ if it improves the throughput of a broadcast on the set of nodes $\{P_{\text{source}}\} \cup \mathcal{P}_{\text{target}}$

Refined heuristics

- *Reduced Broadcast*:
  1. compute the solution of the broadcast
  2. choose the node P_{min} which forwards the minimum of messages to the targets
  3. delete this vertex from the platform graph and start again until the throughput is not improved
- *Augmented Multicast*:
  1. compute the solution of the scatter
  2. choose the node P_{max} which forwards the maximum of messages to the targets
  3. add this node to $\mathcal{P}_{\text{target}}$ if it improves the throughput of a broadcast on the set of nodes $\{P_{\text{source}}\} \cup \mathcal{P}_{\text{target}}$

Refined heuristics

- *Reduced Broadcast*:
  1. compute the solution of the broadcast
  2. choose the node P_{min} which forwards the minimum of messages to the targets
  3. delete this vertex from the platform graph and start again until the throughput is not improved
- *Augmented Multicast*:
  1. compute the solution of the scatter
  2. choose the node P_{max} which forwards the maximum of messages to the targets
  3. add this node to $\mathcal{P}_{\text{target}}$ if it improves the throughput of a broadcast on the set of nodes $\{P_{\text{source}}\} \cup \mathcal{P}_{\text{target}}$

Refined heuristics

- *Reduced Broadcast*:
  1. compute the solution of the broadcast
  2. choose the node P_{min} which forwards the minimum of messages to the targets
  3. delete this vertex from the platform graph and start again until the throughput is not improved
- *Augmented Multicast*:
  1. compute the solution of the scatter
  2. choose the node P_{max} which forwards the maximum of messages to the targets
  3. add this node to $\mathcal{P}_{\text{target}}$ if it improves the throughput of a broadcast on the set of nodes $\{P_{\text{source}}\} \cup \mathcal{P}_{\text{target}}$

Refined heuristics

- *Multisource Multicast*
  1. start from the solution of a scatter
  2. compute the node which forwards the maximum of messages
  3. add this node as *secondary source*:
     it receives all the messages from the previous sources
     it sends part of the messages to the target nodes

Refined heuristics

- *Multisource Multicast*
  1. start from the solution of a scatter
  2. compute the node which forwards the maximum of messages
  3. add this node as *secondary source*:
     it receives all the messages from the previous sources
     it sends part of the messages to the target nodes

Refined heuristics

- *Multisource Multicast*
  1. start from the solution of a scatter
  2. compute the node which forwards the maximum of messages
  3. add this node as *secondary source*:
     it receives all the messages from the previous sources
     it sends part of the messages to the target nodes

Tree based heuristic

- problem: find a low-cost multicast tree
- cost: sum of the weights of the edges in the tree
- *Minimum Steiner Tree*: NP-complete
- some heuristics exist, among other the *Minimum Cost Path Heuristic*:
  - grow a tree until it spans all the target nodes
  - at each step, find the target which could be added with minimum cost to the current tree

Tree based heuristic

- problem: find a low-cost multicast tree
- cost: sum of the weights of the edges in the tree
- *Minimum Steiner Tree*: NP-complete
- some heuristics exist, among other the *Minimum Cost Path Heuristic*:
  - grow a tree until it spans all the target nodes
  - at each step, find the target which could be added with minimum cost to the current tree

Tree based heuristic

- problem: find a low-cost multicast tree
- cost: sum of the weights of the edges in the tree
- *Minimum Steiner Tree*: NP-complete
- some heuristics exist, among other the *Minimum Cost Path Heuristic*:
  - grow a tree until it spans all the target nodes
  - at each step, find the target which could be added with minimum cost to the current tree

Tree based heuristic

- problem: find a low-cost multicast tree
- cost: sum of the weights of the edges in the tree
- *Minimum Steiner Tree*: NP-complete
- some heuristics exist, among other the *Minimum Cost Path Heuristic*:
  - grow a tree until it spans all the target nodes
  - at each step, find the target which could be added with minimum cost to the current tree

Tree based heuristic

- problem: find a low-cost multicast tree
- cost: sum of the weights of the edges in the tree
- *Minimum Steiner Tree*: NP-complete
- some heuristics exist, among other the *Minimum Cost Path Heuristic*:
  - grow a tree until it spans all the target nodes
  - at each step, find the target which could be added with minimum cost to the current tree

Tree based heuristic

- problem: find a low-cost multicast tree
- cost: sum of the weights of the edges in the tree
- *Minimum Steiner Tree*: NP-complete
- some heuristics exist, among other the *Minimum Cost Path Heuristic*:
  - grow a tree until it spans all the target nodes
  - at each step, find the target which could be added with minimum cost to the current tree

Simulation results

- we perform experiments on platforms generated by Tiers
- two types of platforms:
  - one "big": 65 nodes
  - one "small": 30 nodes
- results: comparison of the throughput of our heuristics over two bounds:
  - lower bound (scatter operation)
  - theoretical upper bound

Simulation results

- we perform experiments on platforms generated by Tiers
- two types of platforms:
  - one “big”: 65 nodes
  - one “small”: 30 nodes
- results: comparison of the throughput of our heuristics over two bounds:
  - lower bound (scatter operation)
  - theoretical upper bound

Simulation results

- we perform experiments on platforms generated by Tiers
- two types of platforms:
  - one "big": 65 nodes
  - one "small": 30 nodes
- results: comparison of the throughput of our heuristics over two bounds:
  - lower bound (scatter operation)
  - theoretical upper bound

Simulation results

- we perform experiments on platforms generated by Tiers
- two types of platforms:
  - one "big": 65 nodes
  - one "small": 30 nodes
- results: comparison of the throughput of our heuristics over two bounds:
  - lower bound (scatter operation)
  - theoretical upper bound

Simulation results

- we perform experiments on platforms generated by Tiers
- two types of platforms:
  - one "big": 65 nodes
  - one "small": 30 nodes
- results: comparison of the throughput of our heuristics over two bounds:
  - lower bound (scatter operation)
  - theoretical upper bound

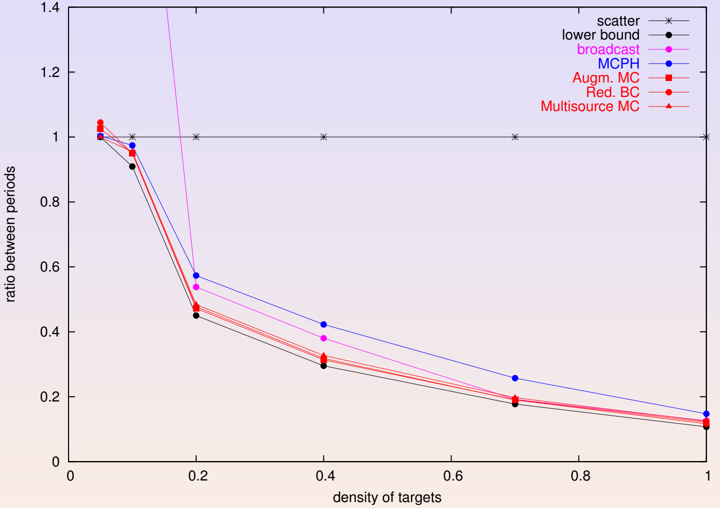
Simulation results

- we perform experiments on platforms generated by Tiers
- two types of platforms:
  - one “big”: 65 nodes
  - one “small”: 30 nodes
- results: comparison of the throughput of our heuristics over two bounds:
  - lower bound (scatter operation)
  - theoretical upper bound

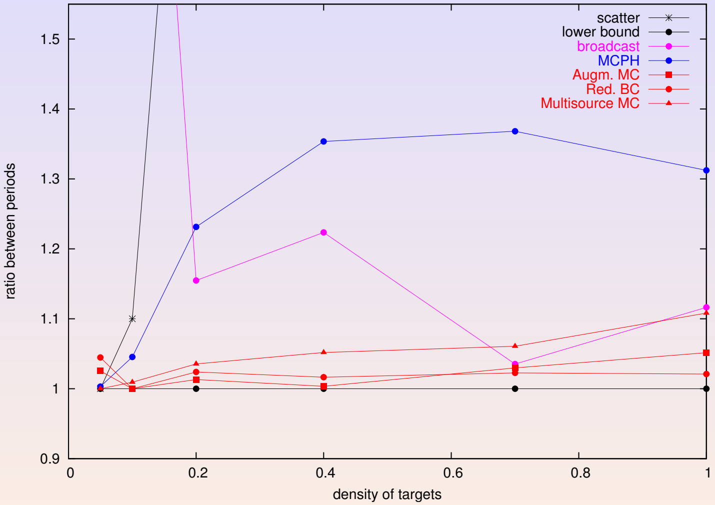
Simulation results

- we perform experiments on platforms generated by Tiers
- two types of platforms:
  - one “big”: 65 nodes
  - one “small”: 30 nodes
- results: comparison of the throughput of our heuristics over two bounds:
  - lower bound (scatter operation)
  - theoretical upper bound

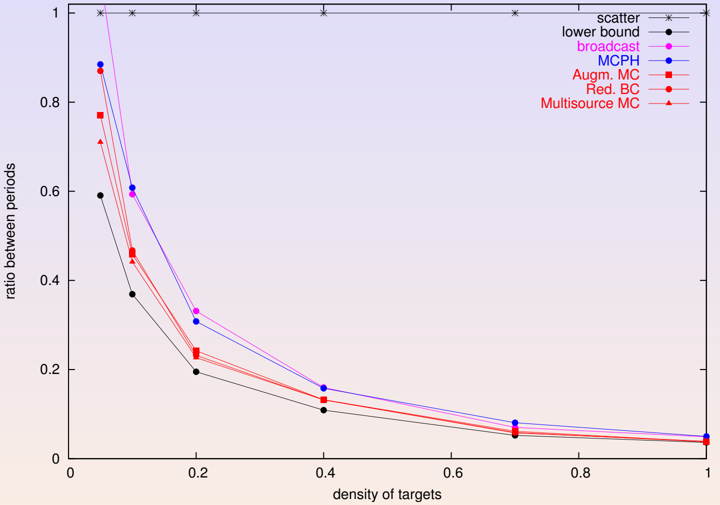
Small platform - comparison over scatter



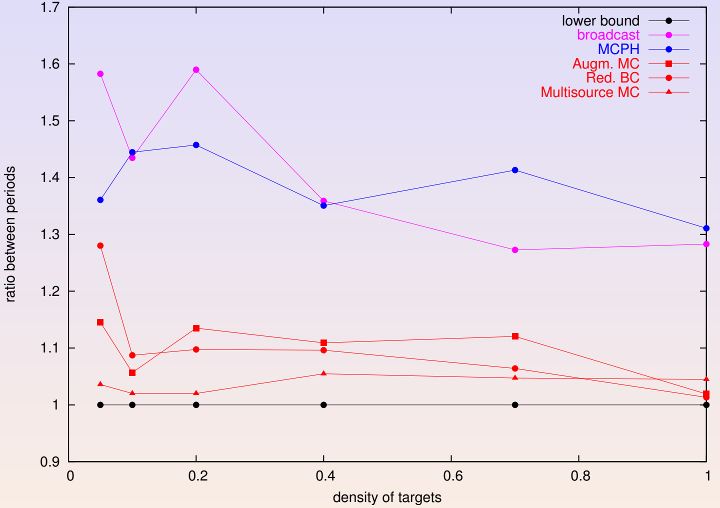
Small platform - comparison over the lower bound



Big platform - comparison over scatter



Big platform - comparison over the lower bound



Outline

Conclusion

- contributions:
  - framework to study pipelined collective communications and optimize throughput
  - Pipelined Multicast is NP-Complete
  - NP-Completeness can be extended to Parallel Prefix computation
  - design of several heuristics based on linear programming or on classical minimum cost tree algorithms
  - comparison by simulations: LP based heuristics are close to the theoretical bound
- limitations:
  - communication model: send-OR-receive more difficult
  - acquiring reliable informations on the platform
  - dynamic version
- next step: tests on bigger platforms, ideally integrate this into a real application