Corrected Trees for Reliable Group Communication

Martin Küttler¹  Maksym Planeta¹  Jan Bierbaum¹  Carsten Weinhold¹  Hermann Härtig¹
Amnon Barak²  Torsten Hoefler³

¹TU Dresden  ²The Hebrew University of Jerusalem  ³ETH Zürich

Principles and Practice of Parallel Programming, Washington D.C.
2019-02-19
Motivation
Motivation

Maksym Planeta: “Corrected Trees”
Motivation

\[ P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow \ldots \rightarrow P_n \]

Communication
Motivation
Motivation
Motivation

Scalability bottleneck

Communication

\[ P_1 \quad P_2 \quad P_3 \quad \ldots \quad P_n \]
Race to Exascale

200 PFLOPS
2 million cores
Race to Exascale

200 PFLOPS
2 million cores

1000 PFLOPS
Race to Exascale

1000 PFLOPS
Race to Exascale

“...quantitative change alters quality”
Friedrich Engels

1000 PFLOPS
Race to Exascale

“...quantitative change alters quality”

Friedrich Engels

1000 PFLOPS

Scalability ↔ Fault Tolerance
Broadcast

- Collective operation
- $N$ processes
Broadcast

- Collective operation
- $N$ processes
- Designated root
Broadcast

- Collective operation
- $N$ processes
- Designated root
- Point-to-point messages
Broadcast

- Collective operation
- $N$ processes
- Designated root
- Point-to-point messages
- Messages color processes
Broadcast

- Collective operation
- \( N \) processes
- Designated root
- Point-to-point messages
- Messages color processes

Diagram:

0 → 1 → 2 → 3 → 4 → 5 → 6 → 7
Broadcast

- Collective operation
- $N$ processes
- Designated root
- Point-to-point messages
- Messages color processes
- Linear broadcast
Broadcast

- Collective operation
- \( N \) processes
- Designated root
- Point-to-point messages
- Messages color processes
- Linear broadcast
- Bad scalability
Outline

- Scalable broadcast
- Fault tolerant broadcast
Outline

- Scalable broadcast
- Fault tolerant broadcast
- Our contribution: Corrected Trees
  1. Construction for several variants
  2. Theoretical analysis
  3. Study with network simulation
  4. Proof-of-concept implementation
Tree-based Broadcast

- Binomial tree
  - Start from the root
Tree-based Broadcast

- Binomial tree
  - Start from the root
  - Copy the tree
Tree-based Broadcast

- Binomial tree
  - Start from the root
  - Copy the tree
Tree-based Broadcast

- Binomial tree
  - Start from the root
  - Copy the tree
  - Repeat
Tree-based Broadcast

- Binomial tree
  - Start from the root
  - Copy the tree
  - Repeat
Tree-based Broadcast

- Binomial tree
  - Start from the root
  - Copy the tree
  - Repeat
Tree-based Broadcast

- Binomial tree
  - Start from the root
  - Copy the tree
  - Repeat
  - Assign processes in Depth First order
Tree-based Broadcast

- Binomial tree
  - Start from the root
  - Copy the tree
  - Repeat
  - Assign processes in Depth First order

\[ O(\log(N)) \]
Tree-based Broadcast

- Binomial tree
  - Start from the root
  - Copy the tree
  - Repeat
  - Assign processes in Depth First order
Tree-based Broadcast

- Binomial tree
  - Start from the root
  - Copy the tree
  - Repeat
  - Assign processes in Depth First order

\[ O \left( \log(N) \right) \]
Tree-based Broadcast

- Binomial tree
  - Start from the root
  - Copy the tree
  - Repeat
  - Assign processes in Depth First order

\[ O(\log N) \]
Tree-based Broadcast

- Binomial tree
  - Start from the root
  - Copy the tree
  - Repeat
  - Assign processes in Depth First order

- Broadcast
  - Root has the message
Tree-based Broadcast

- Binomial tree
  - Start from the root
  - Copy the tree
  - Repeat
  - Assign processes in Depth First order
- Broadcast
  - Root has the message
Tree-based Broadcast

- Binomial tree
  - Start from the root
  - Copy the tree
  - Repeat
  - Assign processes in Depth First order

- Broadcast
  - Root has the message
Tree-based Broadcast

- Binomial tree
  - Start from the root
  - Copy the tree
  - Repeat
  - Assign processes in Depth First order

- Broadcast
  - Root has the message
  - Send in parallel
Tree-based Broadcast

- **Binomial tree**
  - Start from the root
  - Copy the tree
  - Repeat
  - Assign processes in Depth First order

- **Broadcast**
  - Root has the message
  - Send in parallel
Tree-based Broadcast

- Binomial tree
  - Start from the root
  - Copy the tree
  - Repeat
  - Assign processes in Depth First order

- Broadcast
  - Root has the message
  - Send in parallel
  - Double every time
Tree-based Broadcast

- Binomial tree
  - Start from the root
  - Copy the tree
  - Repeat
  - Assign processes in Depth First order
- Broadcast
  - Root has the message
  - Send in parallel
  - Double every time
Tree-based Broadcast

- Binomial tree
  - Start from the root
  - Copy the tree
  - Repeat
  - Assign processes in Depth First order

- Broadcast
  - Root has the message
  - Send in parallel
  - Double every time
  - $O(\log(n))$ depth
Broadcast with a Failed Process

- On-leaf failure
- Message is silently lost
- Partial coloring
- Healthy processes remain uncolored

Maksym Planeta: “Corrected Trees”
Broadcast with a Failed Process

- Non-leaf failure
Broadcast with a Failed Process

- Non-leaf failure
- Message is silently lost
Broadcast with a Failed Process

- Non-leaf failure
- Message is silently lost
Broadcast with a Failed Process

- Non-leaf failure
- Message is silently lost
- Partial coloring

Diagram:

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7

Trees remain uncolored.
Broadcast with a Failed Process

- Non-leaf failure
- Message is silently lost
- Partial coloring
- Healthy processes remain uncolored
Failures

- Every live process receives a message (like linear broadcast)
- Low latency (like tree broadcast)
- Fail-stop model
- Reliable messages
- The root is alive
- Multiple failures: details in the paper
Failures

- Every live process receives a message (like linear broadcast)
- Low latency (like tree broadcast)
- Fail-stop model
- Reliable messages
- The root is alive
- Multiple failures: details in the paper
- Future MPI standards will handle failures (example: ULFM)
Acknowledged Broadcast\textsuperscript{1} 

- Broadcast along the tree

\textsuperscript{1}Buntinas, “Scalable Distributed Consensus to Support MPI Fault Tolerance.”
Acknowledged Broadcast

- Broadcast along the tree

---

1Buntinas, “Scalable Distributed Consensus to Support MPI Fault Tolerance.”
Acknowledged Broadcast

- Broadcast along the tree
- Waiting for ACKs

---

1 Buntinas, “Scalable Distributed Consensus to Support MPI Fault Tolerance.”
Acknowledged Broadcast

- Broadcast along the tree
- Waiting for ACKs

---

1 Buntinas, “Scalable Distributed Consensus to Support MPI Fault Tolerance.”
Acknowledged Broadcast\textsuperscript{1}

- Broadcast along the tree
- Waiting for ACKs

\textsuperscript{1}Buntinas, “Scalable Distributed Consensus to Support MPI Fault Tolerance.”
Acknowledged Broadcast

- Broadcast along the tree
- Waiting for ACKs
- Ready when no ACK pending

---

1 Buntinas, “Scalable Distributed Consensus to Support MPI Fault Tolerance.”
**Acknowledged Broadcast**

- Broadcast along the tree
- Waiting for ACKs
- Ready when no ACK pending
- Children send ACKs

---

1 Buntinas, “Scalable Distributed Consensus to Support MPI Fault Tolerance.”
Acknowledged Broadcast

- Broadcast along the tree
- Waiting for ACKs
- Ready when no ACK pending
- Children send ACKs

\[1\text{ Buntinas, “Scalable Distributed Consensus to Support MPI Fault Tolerance.”}\]
Acknowledged Broadcast

- Broadcast along the tree
- Waiting for ACKs
- Ready when no ACK pending
- Children send ACKs

---

1 Buntinas, “Scalable Distributed Consensus to Support MPI Fault Tolerance.”
Acknowledged Broadcast

- Broadcast along the tree
- Waiting for ACKs
- Ready when no ACK pending
- Children send ACKs
- The root waits for all ACKs

---

1 Buntinas, “Scalable Distributed Consensus to Support MPI Fault Tolerance.”
Acknowledge Broadcast with Failures

- Run as usual
Acknowledged Broadcast with Failures

- Run as usual

Diagram:

```
  0
 /  \
1  5 7
|    |
2  4  6
|    |
3
```
Acknowledged Broadcast with Failures

- Run as usual

Maksym Planeta: “Corrected Trees”
Acknowledged Broadcast with Failures

- Run as usual
- Message is lost
Acknowledged Broadcast with Failures

- Run as usual
- Message is lost
Acknowledged Broadcast with Failures

- Run as usual
- Message is lost
- Failure detection

Maksym Planeta: “Corrected Trees”
Acknowledged Broadcast with Failures

- Run as usual
- Message is lost
- Failure detection
- Notify the root

Maksym Planeta: “Corrected Trees”
Acknowledged Broadcast with Failures

- Run as usual
- Message is lost
- Failure detection
- Notify the root
- Restructure the tree
Acknowledged Broadcast with Failures

- Run as usual
- Message is lost
- Failure detection
- Notify the root
- Restructure the tree
- Double the latency

Maksym Planeta: “Corrected Trees”
Acknowledged Broadcast with Failures

- Run as usual
- Message is lost
- Failure detection
- Notify the root
- Restructure the tree
- Double the latency
- Unnecessary waiting

Maksym Planeta: “Corrected Trees”
Simulated Latency: Acknowledged Broadcast

Latency, steps

Processes

Better
Simulated Latency: Acknowledged Broadcast

![Graph showing latency in steps against processes](image)

- Better

Maksym Planeta: “Corrected Trees”
Simulated Latency: Acknowledged Broadcast

Latency, steps

Processes

Binomial (ack.)

Binomial

Maksym Planeta: “Corrected Trees”
Gossip-based Broadcast\(^2\)

- No tree structure

\(\text{\textsuperscript{2}}\)Birman et al., “Bimodal Multicast.”
Gossip-based Broadcast

- No tree structure
- Send to random process (Gossip)

---

2Birman et al., “Bimodal Multicast.”
Gossip-based Broadcast\textsuperscript{2}

- No tree structure
- Send to random process (Gossip)

\textsuperscript{2}Birman et al., “Bimodal Multicast.”
Gossip-based Broadcast$^2$

- No tree structure
- Send to random process (Gossip)
- Multiple rounds

$^2$Birman et al., “Bimodal Multicast.”
Gossip-based Broadcast$^2$

- No tree structure
- Send to random process (Gossip)
- Multiple rounds

---

$^2$Birman et al., “Bimodal Multicast.”
Gossip-based Broadcast

- No tree structure
- Send to random process (Gossip)
- Multiple rounds

---

2 Birman et al., “Bimodal Multicast.”
Gossip-based Broadcast

- No tree structure
- Send to random process (Gossip)
- Multiple rounds

\[^2\text{Birman et al., “Bimodal Multicast.”}\]
Gossip-based Broadcast\textsuperscript{2}

- No tree structure
- Send to random process (Gossip)
- Multiple rounds

\textsuperscript{2}Birman et al., “Bimodal Multicast.”
Gossip-based Broadcast\textsuperscript{2}

- No tree structure
- Send to random process (Gossip)
- Multiple rounds
- Full reachability with high probability

\textsuperscript{2}Birman et al., “Bimodal Multicast.”
Gossip-based Broadcast$^2$

- No tree structure
- Send to random process (Gossip)
- Multiple rounds
- Full reachability with high probability
- Inherently fault tolerant
- No need for failure detector

$^2$Birman et al., “Bimodal Multicast.”
Corrected Gossip-based Broadcast

- Keep processes in a ring

---

Corrected Gossip-based Broadcast\(^3\)

- Keep processes in a ring
- Gossip phase

\(^3\)Hoefler et al., “Corrected Gossip Algorithms for Fast Reliable Broadcast on Unreliable Systems.”
Corrected Gossip-based Broadcast\textsuperscript{3}

- Keep processes in a ring
- Gossip phase

\textsuperscript{3}Hoefler et al., “Corrected Gossip Algorithms for Fast Reliable Broadcast on Unreliable Systems.”
Corrected Gossip-based Broadcast\(^3\)

- Keep processes in a ring
- Gossip phase

\(^3\)Hoefler et al., “Corrected Gossip Algorithms for Fast Reliable Broadcast on Unreliable Systems.”
Corrected Gossip-based Broadcast³

- Keep processes in a ring
- Gossip phase

Corrected Gossip-based Broadcast

- Keep processes in a ring
- Gossip phase
- Correction phase
  - Send along the ring

---

Corrected Gossip-based Broadcast

- Keep processes in a ring
- Gossip phase
- Correction phase
  - Send along the ring

---

Corrected Gossip-based Broadcast

- Keep processes in a ring
- Gossip phase
- Correction phase
  - Send along the ring
- Better coloring guarantee

---

Corrected Gossip-based Broadcast\(^3\)

- Keep processes in a ring
- Gossip phase
- Correction phase
  - Send along the ring
- Better coloring guarantee
- Inherently fault tolerant

\(^3\)Hoefler et al., “Corrected Gossip Algorithms for Fast Reliable Broadcast on Unreliable Systems.”
Corrected Gossip-based Broadcast

- Keep processes in a ring
- Gossip phase
- Correction phase
  - Send along the ring
- Better coloring guarantee
- Inherently fault tolerant
- Different correction types

---

Simulated Latency: Corrected Gossip

![Graph showing latency steps for different processes.]

- Binomial (ack.)
- Binomial (corr.)
- Gossip (corr.)
Simulated Latency: Corrected Gossip

![Graph showing latency steps for different processes with lines for Binomial (ack.), Optimal (ack.), Binomial (corr.), and Optimal (corr.).]
Correcting a Broadcast Tree

Maksym Planeta: “Corrected Trees”
Correcting a Broadcast Tree

- Tree phase

Maksym Planeta: “Corrected Trees”
Correcting a Broadcast Tree

- Tree phase
Correcting a Broadcast Tree

- Tree phase
  - Binomial tree
Correcting a Broadcast Tree

- Tree phase
  - Binomial tree
Correcting a Broadcast Tree

- Tree phase
  - Binomial tree
- Correction phase
Correcting a Broadcast Tree

- Tree phase
  - Binomial tree
- Correction phase
Correcting a Broadcast Tree

- Tree phase
  - Binomial tree
- Correction phase
- Guaranteed coloring
Correcting a Broadcast Tree with a Failure

- On-Leaf Failure
- Partial Coloring
- Correction is in Chosen, IVE Processes Stay UnColored

Maksym Planeta: “Corrected Trees”
Correcting a Broadcast Tree with a Failure

- Non-leaf failure
Correcting a Broadcast Tree with a Failure

- Non-leaf failure
Correcting a Broadcast Tree with a Failure

- Non-leaf failure
Correcting a Broadcast Tree with a Failure

- Non-leaf failure
- Partial coloring
Correcting a Broadcast Tree with a Failure

- Non-leaf failure
- Partial coloring
- Correction is inefficient
Correcting a Broadcast Tree with a Failure

- Non-leaf failure
- Partial coloring
- Correction is inefficient
Correcting a Broadcast Tree with a Failure

- Non-leaf failure
- Partial coloring
- Correction is inefficient
- Live processes stay uncolored
Corrected Tree-based Broadcast

1. Tree phase

Maksym Planeta: “Corrected Trees”
Corrected Tree-based Broadcast

1. Tree phase
Corrected Tree-based Broadcast

1. Tree phase
   - Interleave parents and children

Maksym Planeta: “Corrected Trees”
1. Tree phase
   - Interleave parents and children
Corrected Tree-based Broadcast

1. Tree phase
   - Interleave parents and children
Corrected Tree-based Broadcast

1. Tree phase
   - Interleave parents and children
   - Recursively repeat
1. Tree phase
   - Interleave parents and children
   - Recursively repeat
Corrected Tree-based Broadcast

1. Tree phase
   - Interleave parents and children
   - Recursively repeat
Corrected Tree-based Broadcast

1. Tree phase
   - Interleave parents and children
   - Recursively repeat
1. Tree phase
   - Interleave parents and children
   - Recursively repeat
Corrected Tree-based Broadcast

1. Tree phase
   - Interleave parents and children
   - Recursively repeat
Corrected Tree-based Broadcast

1. Tree phase
   - Interleave parents and children
   - Recursively repeat
Corrected Tree-based Broadcast

1. Tree phase
   - Interleave parents and children
   - Recursively repeat

2. Correction phase

Maksym Planeta: “Corrected Trees”
Corrected Tree-based Broadcast with Failure

Maksym Planeta: “Corrected Trees”
Corrected Tree-based Broadcast with Failure

- Non-leaf failure

Maksym Planeta: “Corrected Trees”
Corrected Tree-based Broadcast with Failure

- Non-leaf failure
- Continue coloring
Corrected Tree-based Broadcast with Failure

- Non-leaf failure
- Continue coloring
Corrected Tree-based Broadcast with Failure

- Non-leaf failure
- Continue coloring
- Small holes
Corrected Tree-based Broadcast with Failure

- Non-leaf failure
- Continue coloring
- Small holes
- Correction phase
Simulated Latency: Corrected Trees

![Graph showing latency for different algorithms with respect to processes."

- Binomial (ack.)
- Optimal (ack.)
- Gossip (corr.)

Latency, steps vs. Processes

Maksym Planeta: “Corrected Trees”
Simulated Latency: Corrected Trees

![Graph showing latency steps for different processes]

- Binomial (ack.)
- Optimal (ack.)
- Binomial (corr.)
- Gossip (corr.)
- Optimal (corr.)
Evaluation

- OpenMPI-based
- $512 \times 72$-core nodes $= 36,864$ processes
- Piz Daint (Cray XC 40)
MPI-based Implementation

![Graph showing latency in µs vs processes]
MPI-based Implementation

Latency, $\mu$s

Processes

1152 2304 4608 9216 18432 36864

Binomial (Cray)
MPI-based Implementation

Better

Latency, μs

Processes

1152 2304 4608 9216 18432 36864

Binomial (corr.)

Binomial (Cray)

Maksym Planeta: “Corrected Trees”
MPI-based Implementation

Better

Latency, $\mu$s

Processes

Binomial (corr.)
Binomial (Cray, no shared memory)
Binomial (Cray)
MPI-based Implementation

![Graph showing latency versus number of processes for different communication methods.]

- **Gossip (corr.)**
- **Binomial (corr.)**
- **Binomial (Cray, no shared memory)**
- **Binomial (Cray)**

**Processes**: 1152, 2304, 4608, 9216, 18432, 36864

**Latency**, μs: 0, 20, 40, 60, 80, 100

Maksym Planeta: “Corrected Trees”
Conclusion

- Corrected Trees
  - Interleaved Trees
  - Correction Phase
Conclusion

- Corrected Trees
  - Interleaved Trees
  - Correction Phase
- Properties
  - Reliable broadcast
  - Faster than acknowledged tree
  - Less messages than Corrected Gossip
Conclusion

- Corrected Trees
  - Interleaved Trees
  - Correction Phase
- Properties
  - Reliable broadcast
  - Faster than acknowledged tree
  - Less messages than Corrected Gossip
- Closed-form expression for binomial
- Simple construction for other practical tree types
- Future work: Other collectives
Correction Types\(^4\)

1. Opportunistic correction
   - Fixed rounds
   - Probabilistic reachability

\(^4\)Hoefler et al., “Corrected Gossip Algorithms for Fast Reliable Broadcast on Unreliable Systems.”
Correction Types

1. Opportunistic correction
   - Fixed rounds
   - Probabilistic reachability
2. Checked correction
   - Variable rounds
   - Guaranteed reachability

---

Correction Types$^4$

1. Opportunistic correction
   - Fixed rounds
   - Probabilistic reachability

2. Checked correction
   - Variable rounds
   - Guaranteed reachability

3. Failure-proof correction
   - Extended checked correction
   - Tolerates online failures

Correction Types

1. Opportunistic correction
   - Fixed rounds
   - Probabilistic reachability
2. Checked correction
   - Variable rounds
   - Guaranteed reachability
3. Failure-proof correction
   - Extended checked correction
   - Tolerates online failures

---

Correction Types

1. Opportunistic correction
   - Fixed rounds
   - Probabilistic reachability
2. Checked correction
   - Variable rounds
   - Guaranteed reachability
3. Failure-proof correction
   - Extended checked correction
   - Tolerates online failures

Limitations:
- Only broadcast for now
- Small messages

---

\(^4\) Hoefler et al., “Corrected Gossip Algorithms for Fast Reliable Broadcast on Unreliable Systems.”
Different Corrected Trees

Figure: Broadcast median latency (system X)
Latency Model

- LogP-like model
  - Send overhead

P_0 → P_1 → P_2

Time

Maksym Planeta: “Corrected Trees”
Latency Model

- LogP-like model
  - Send overhead
  - p2p-Latency

P_0 \rightarrow P_1 \rightarrow P_2

Time

L
Latency Model

- LogP-like model
  - Send overhead
  - p2p-Latency
  - Receive overhead
Latency Model

- LogP-like model
  - Send overhead
  - p2p-Latency
  - Receive overhead
  - Parallel send and receive
Latency Model

- LogP-like model
  - Send overhead
  - p2p-Latency
  - Receive overhead
  - Parallel send and receive
Latency Model

- LogP-like model
  - Send overhead
  - p2p-Latency
  - Receive overhead
  - Parallel send and receive
Latency Model

- LogP-like model
  - Send overhead
  - p2p-Latency
  - Receive overhead
  - Parallel send and receive

- Fault model
  - Failed processes
Latency Model

- LogP-like model
  - Send overhead
  - p2p-Latency
  - Receive overhead
  - Parallel send and receive

- Fault model
  - Failed processes
Latency Model

- LogP-like model
  - Send overhead
  - p2p-Latency
  - Receive overhead
  - Parallel send and receive

- Fault model
  - Failed processes
  - Failures not detected
Simulation: Message Count

![Simulation chart showing message count for different correction types and distance d. The chart compares Binomial, 4-ary, Lamé, Optimal, and Gossip methods.](chart.png)

**Message per process**
- Binomial
- 4-ary
- Lamé
- Optimal
- Gossip

**Correction Type (distance d)**
- Opport. (1)
- Opport. (2)
- Opport. (4)
- Checked

**Acknowledged**

Maksym Planeta: “Corrected Trees”
Broadcast Timeline
Broadcast Timeline

Maksym Planeta: “Corrected Trees”
Broadcast Timeline
Broadcast Timeline

Maksym Planeta: “Corrected Trees”
Broadcast Timeline
Broadcast Timeline

Maksym Planeta: “Corrected Trees”
Broadcast Timeline

Maksym Planeta: “Corrected Trees”
Simulation: Latency Sturdiness

Figure: Average quiescence latency grows with fault rate
Simulation: Message Count Sturdiness

Figure: Average number of messages goes down with higher fault rate
Component Failures in HPC Systems\textsuperscript{5}

<table>
<thead>
<tr>
<th>Component</th>
<th># of Nodes Affected</th>
<th>MTBF</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFS, core switch</td>
<td>1,408</td>
<td>65.10 days</td>
</tr>
<tr>
<td>Rack</td>
<td>32</td>
<td>86.90 days</td>
</tr>
<tr>
<td>Edge switch</td>
<td>16</td>
<td>17.37 days</td>
</tr>
<tr>
<td>PSU</td>
<td>4</td>
<td>28.94 days</td>
</tr>
<tr>
<td>Compute node</td>
<td>1</td>
<td>15.8 hours</td>
</tr>
</tbody>
</table>

Data gathered between 2010-11-01 and 2012-04-06 on TSUBAME 2.0

\textsuperscript{5}Sato et al., “Design and Modeling of a Non-blocking Checkpointing System.”