Bine Trees: Enhancing Collective Operations by Optimizing Communication Locality

Daniele De Sensi[1], Saverio Pasqualoni[1], Lorenzo Piarulli[1],
Tommaso Bonato[2], Seydou Ba[3], Matteo Turisini[4], Jens Domke[3], Torsten Hoefler[2]

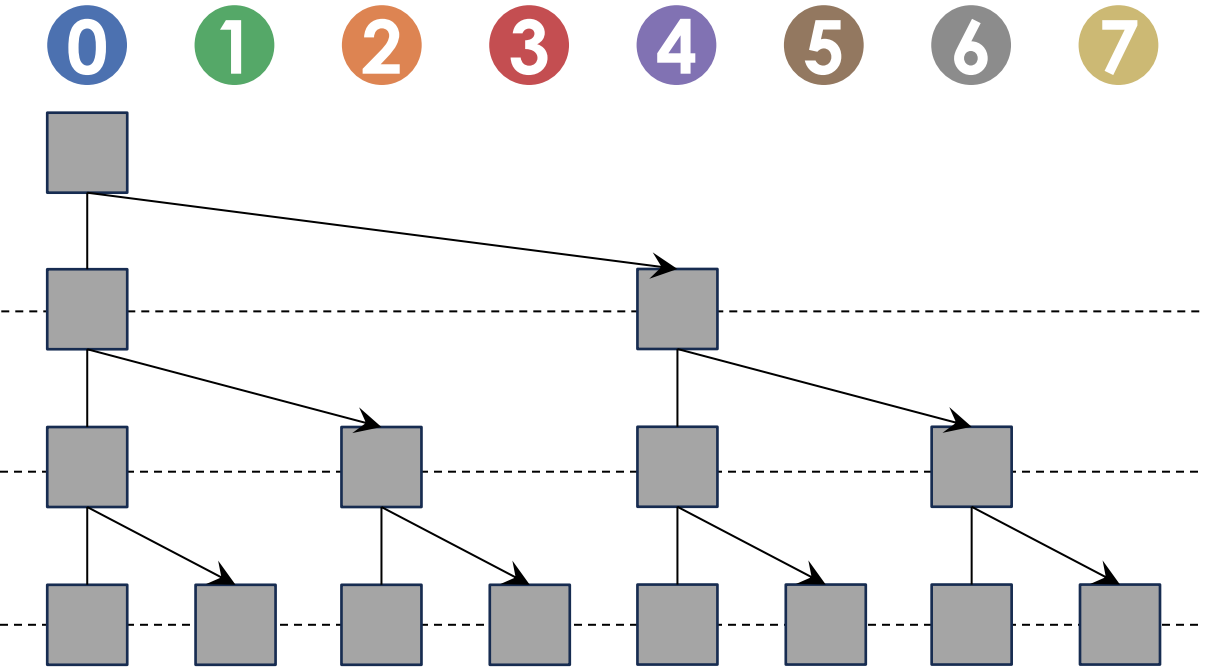
[1] Sapienza University of Rome [2] ETH Zurich [3] RIKEN R-CCS [4] CINECA

SAPIENZA UNIVERSITÀ DI ROMA

The International Conference for High Performance Computing, Networking, Storage, and Analysis
St. Louis, Nov. 16-21

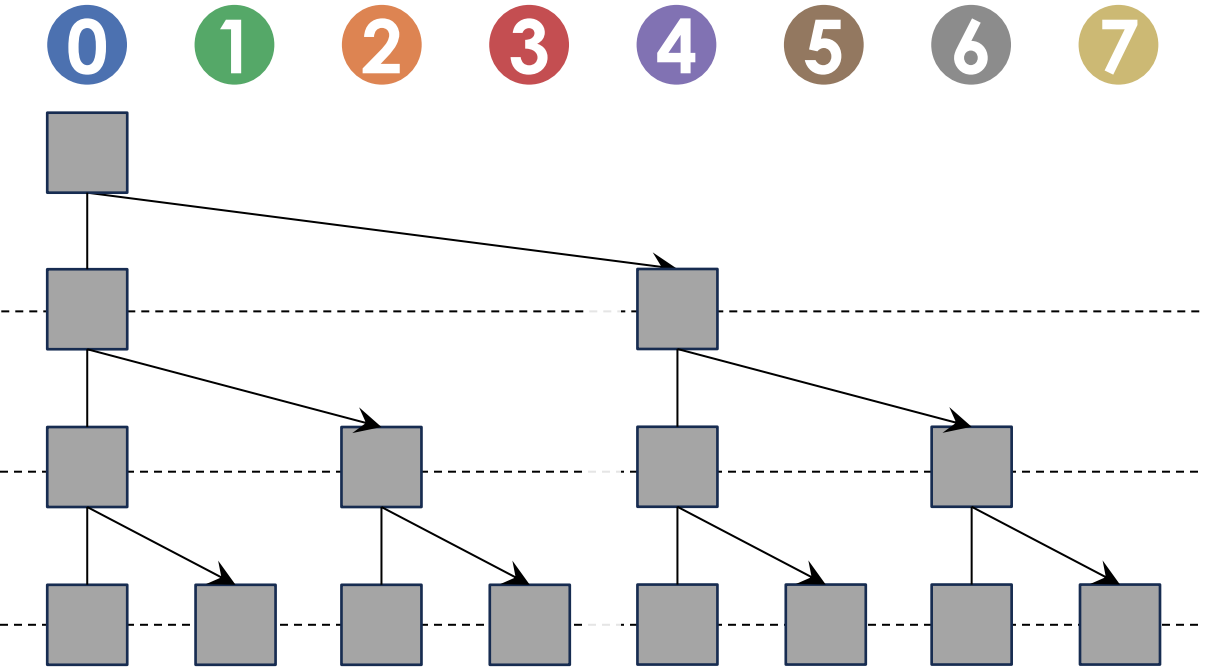
Motivation

Distance-Halving Binomial Tree

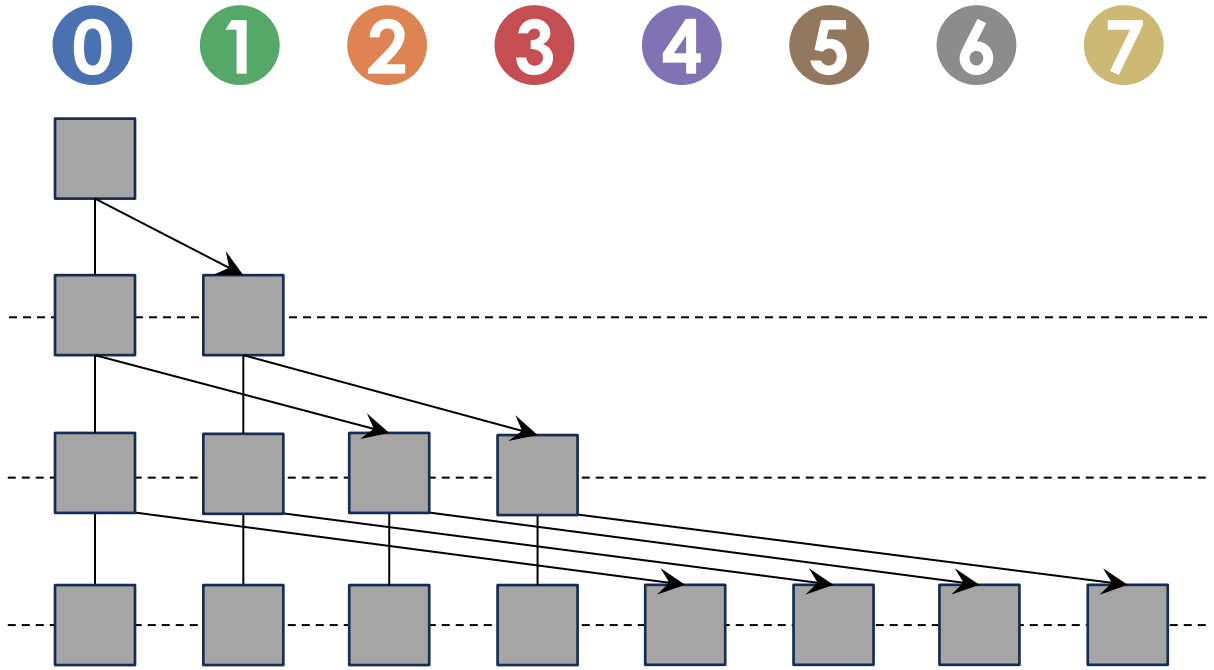


Motivation

Distance-Halving Binomial Tree

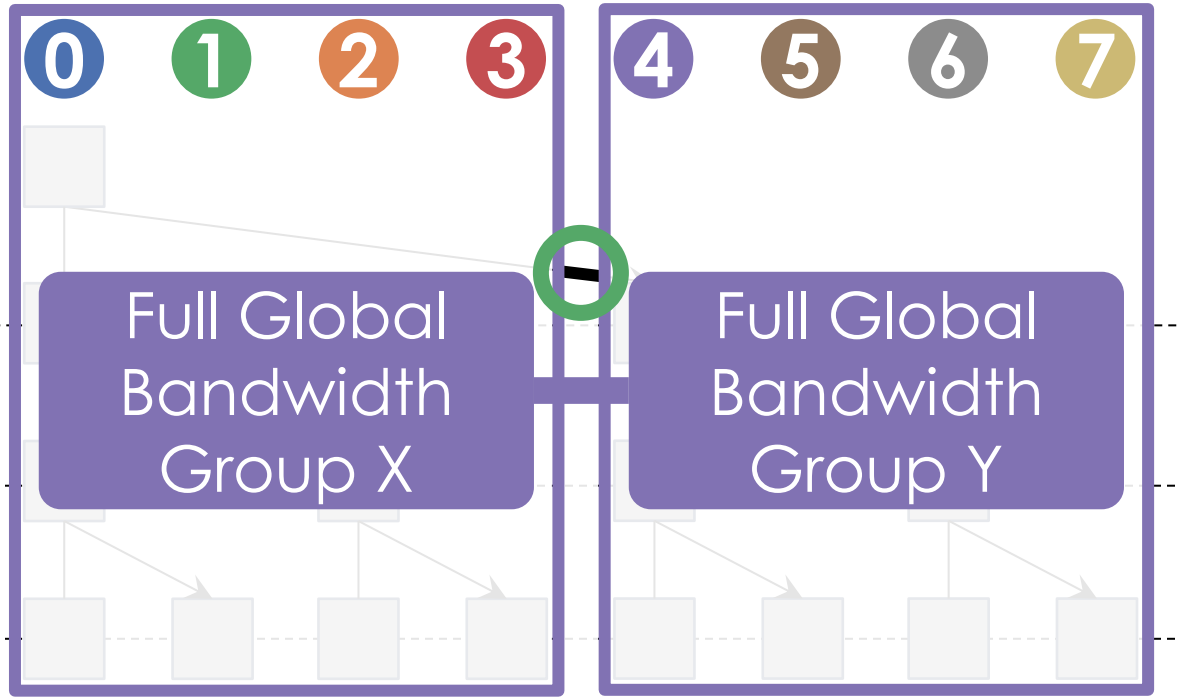


Distance-Doubling Binomial Tree



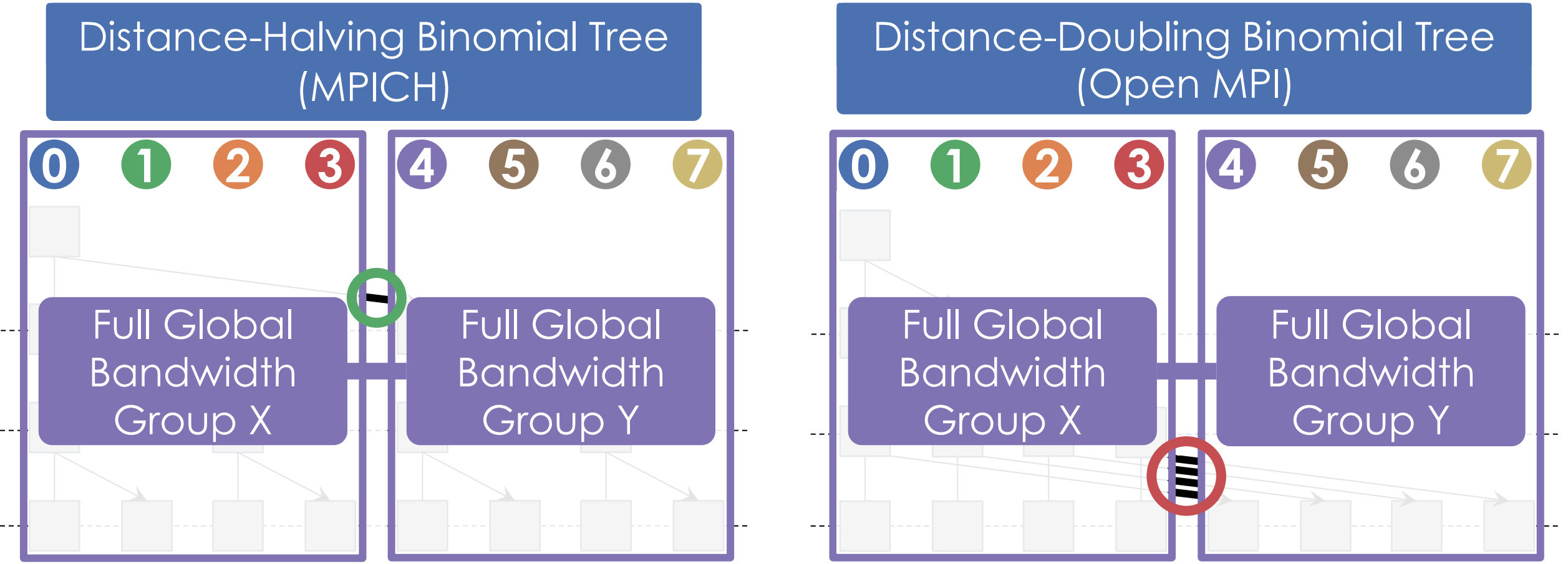
Motivation

Distance-Halving Binomial Tree



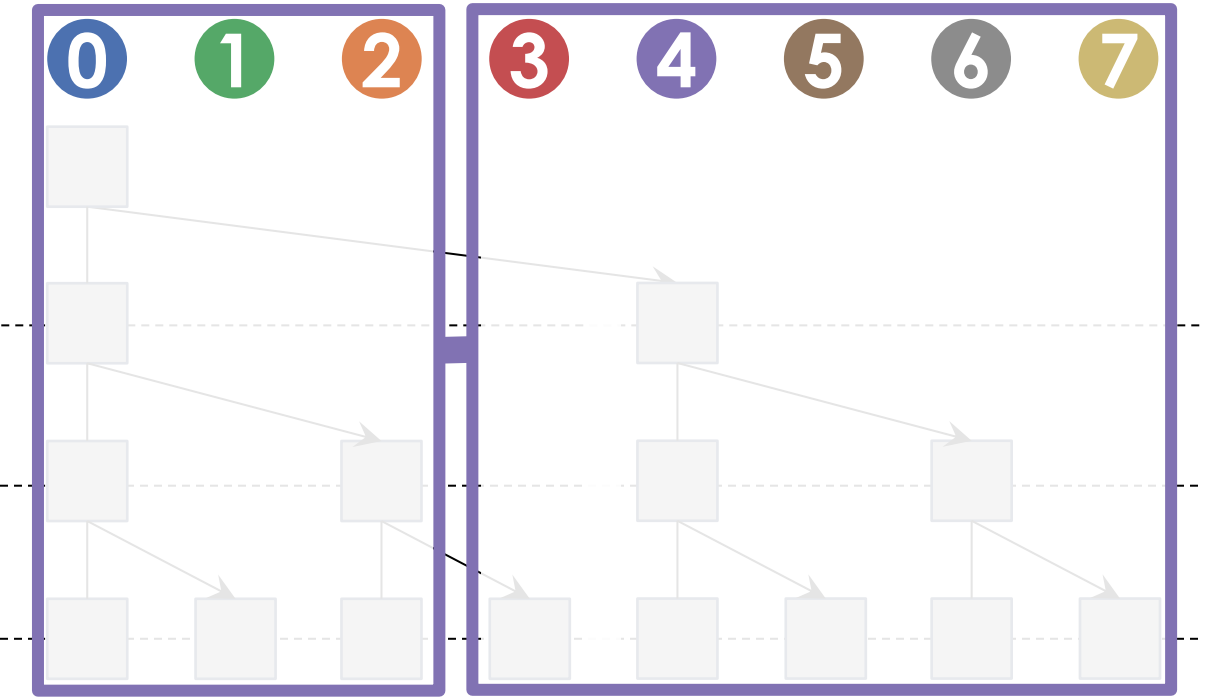
Distance-Doubling Binomial Tree

Motivation

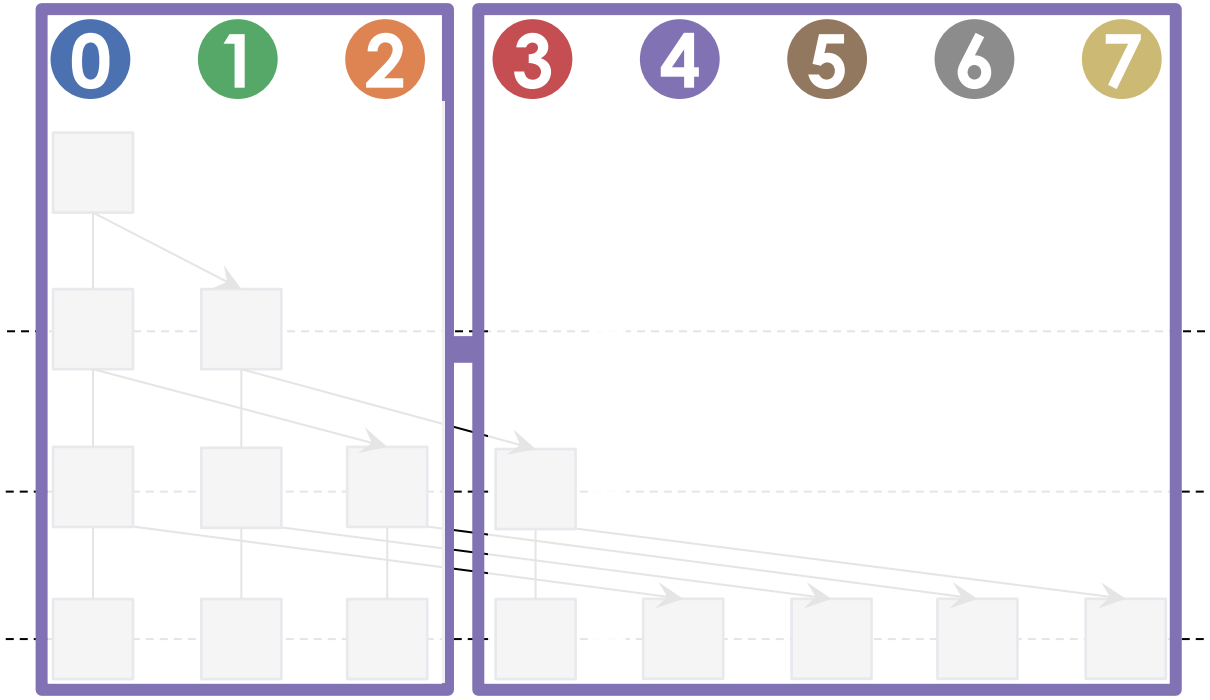

Distance-Halving Binomial Tree (MPICH)
0
1
2
3
4
5
6
7
Full Global Bandwidth Group X
Full Global Bandwidth Group Y
Distance-Doubling Binomial Tree (Open MPI)
0
1
2
3
4
5
6
7
Full Global Bandwidth Group X
Full Global Bandwidth Group Y

Motivation: Why not Hierarchical Approaches

Distance-Halving Binomial Tree

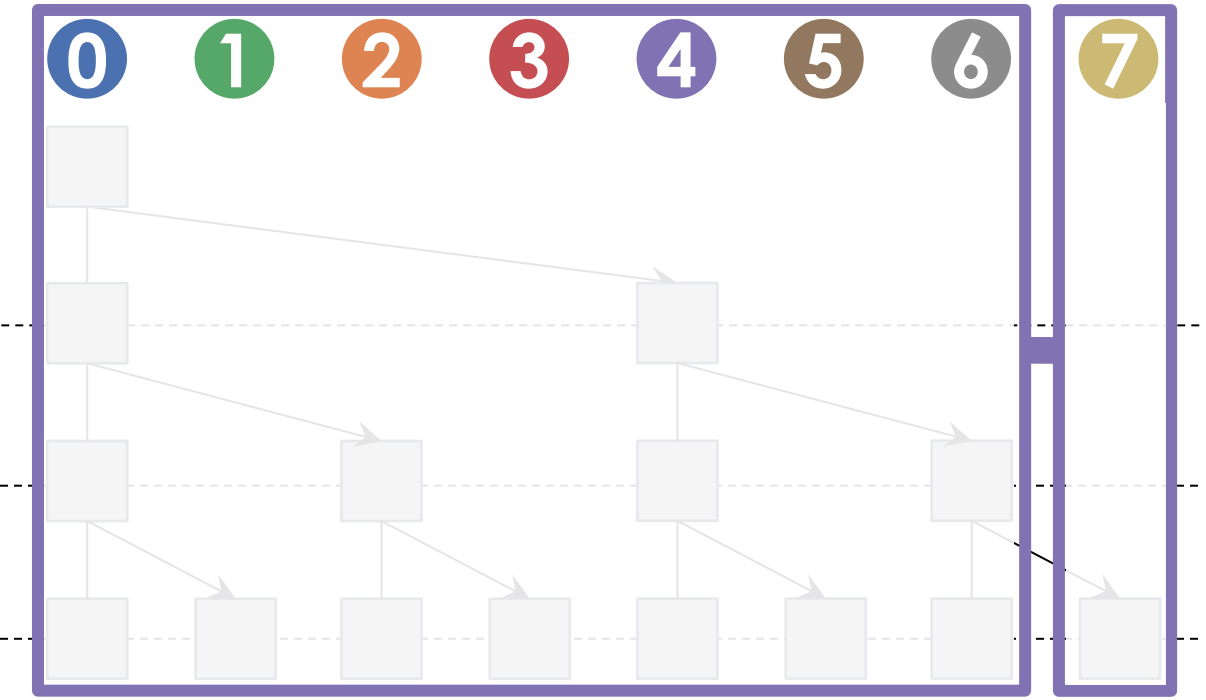


Distance-Doubling Binomial Tree

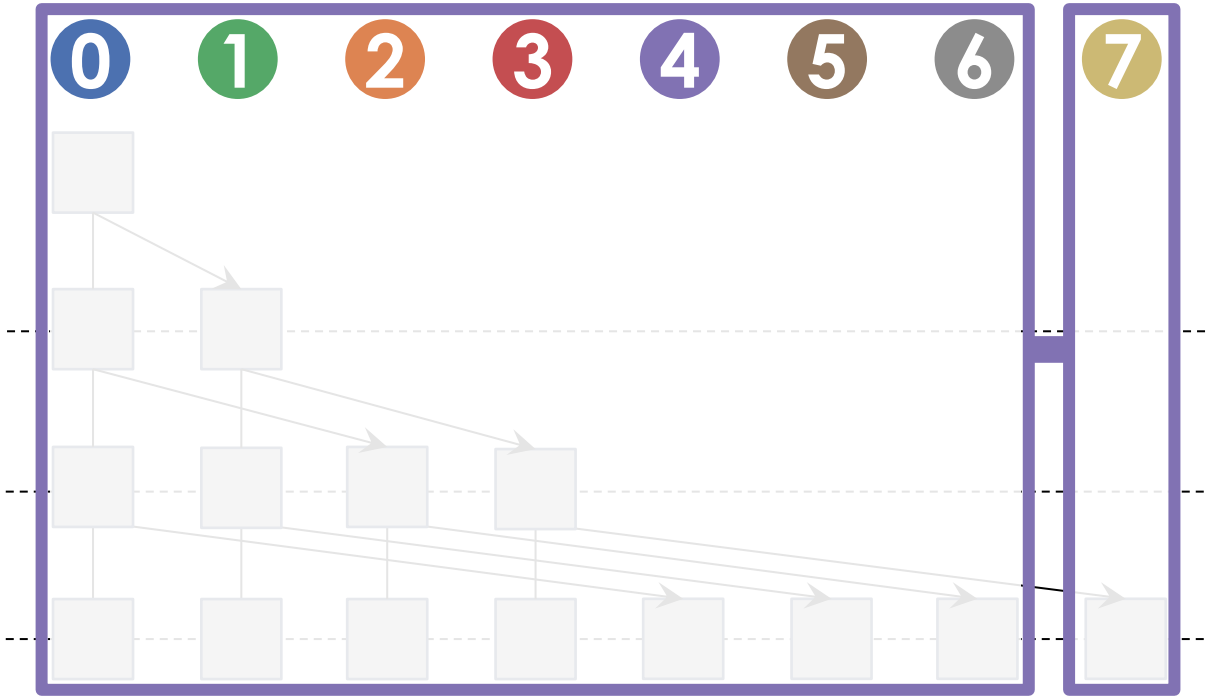


Motivation: Why not Hierarchical Approaches

Distance-Halving Binomial Tree

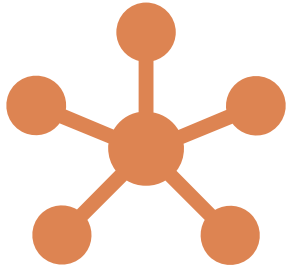


Distance-Doubling Binomial Tree

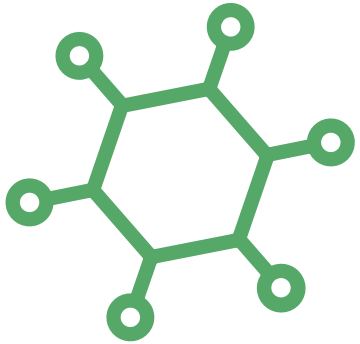


Goal

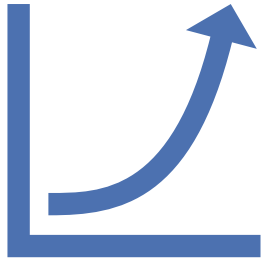
Reduce rank distance to
reduce usage of **oversubscribed links**



Do not make assumptions
on topology or node allocation



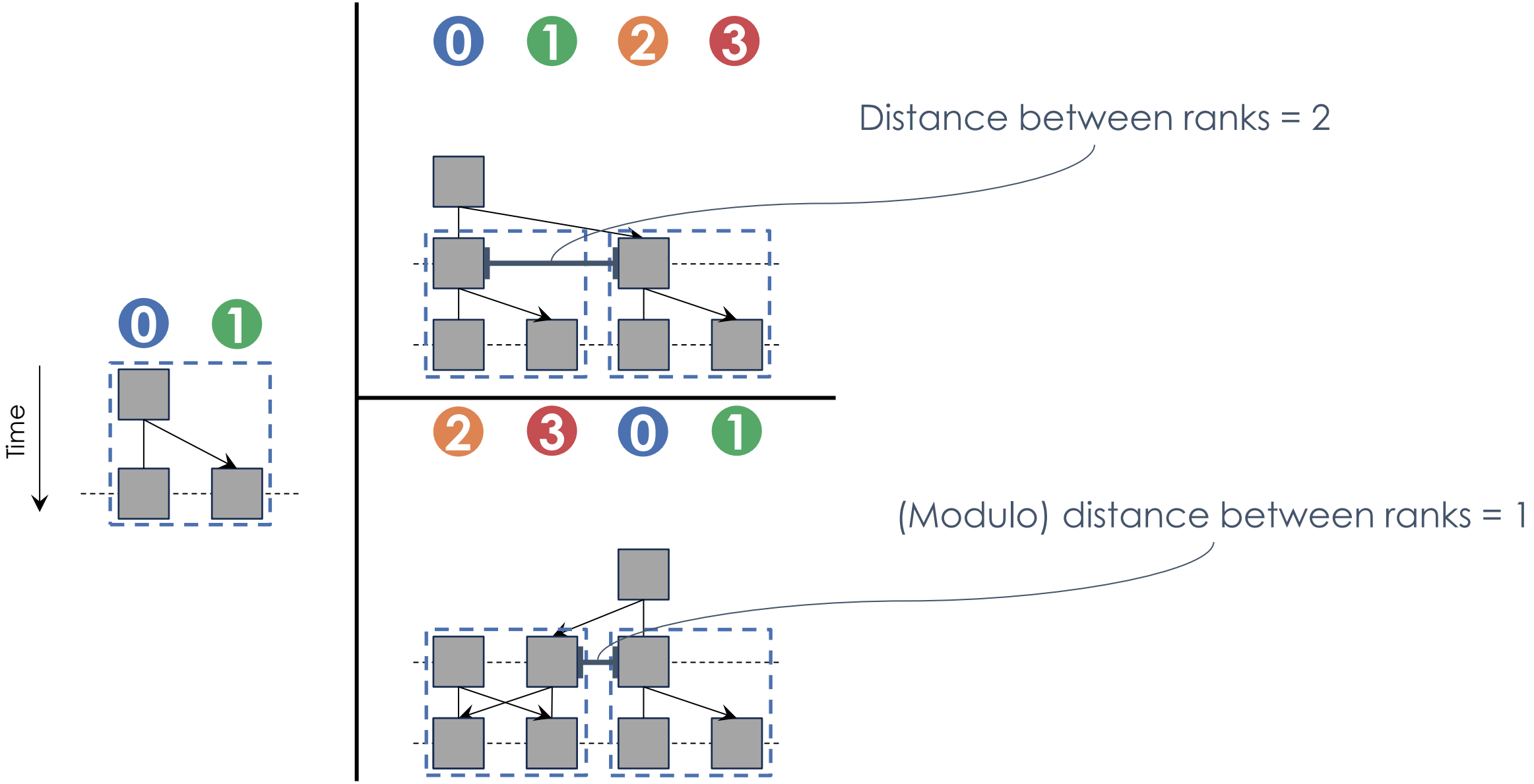
33% traffic reduction on
oversubscribed links



+60% performance on Leonardo
+80% on LUMI
+2x on MareNostrum5
+5x on Fugaku

Bine Trees Construction

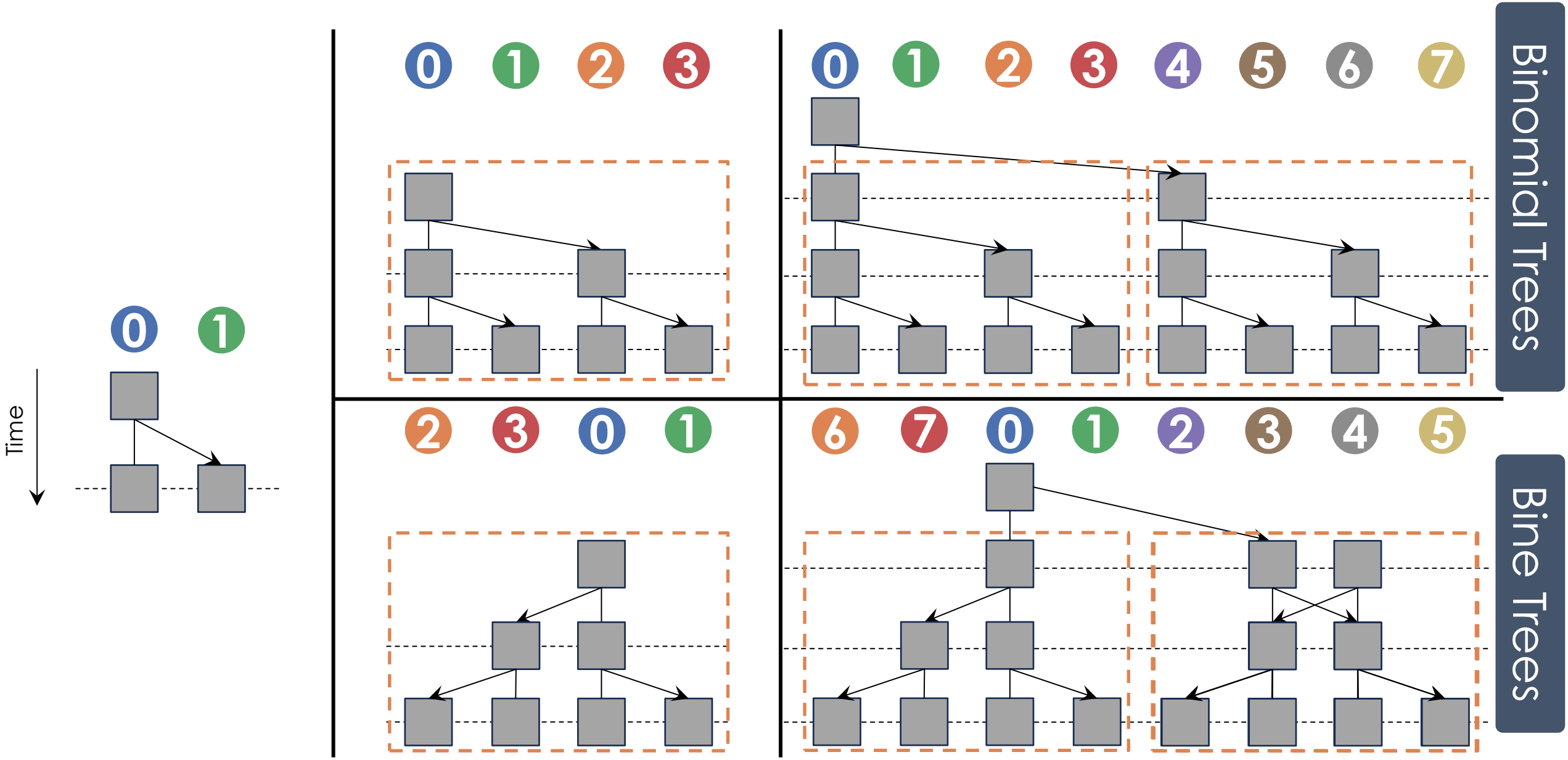
Bine Tree Construction



Binomial Trees

Bine Trees

Bine Tree Construction



negabinary value (e.g., Fig. 3, F). Ranks greater than m (to the left of rank 0) are represented using the negabinary representation of $r - p$. For instance, rank $r = 6$ in an 8-node tree is represented as $6 - 8 = -2 = 010_{-2}$ (Fig. 3, G).

We define the functions $rank2nb(r, p)$ and $nb2rank(r, p)$ to convert a rank identifier to its negabinary representation, and vice versa, for a collective on p ranks. For example, $rank2nb(2, 8) = 110_{-2}$ and $rank2nb(6, 8) = 010_{-2}$. Both functions can be efficiently implemented using bit masking and an addition or subtraction. When the context is clear, we will omit the p parameter and the $_{-2}$ base.

2.3.2 ***Determining the Communication Partner.*** In the broadcast, each rank needs to determine when to join the tree, or in other words, at which step it will receive data from its parent. We denote by u the number of consecutive least significant bits that are equal to each other, starting from the least significant bit. For instance, for a 16-node *Bine* tree, $u = 3$ for 1000, and $u = 2$ for 1011. In a distance-halving *Bine* tree broadcast, rank r receives data at step $i = s - u$ (with $i \in [0, s - 1]$). As illustrated in Fig. 4 A, rank 8 receives the data from its parent at step $i = 1$, since $rank2nb(8) = 1000$, and thus $u = 3$, giving $i = s - u = 4 - 3 = 1$.

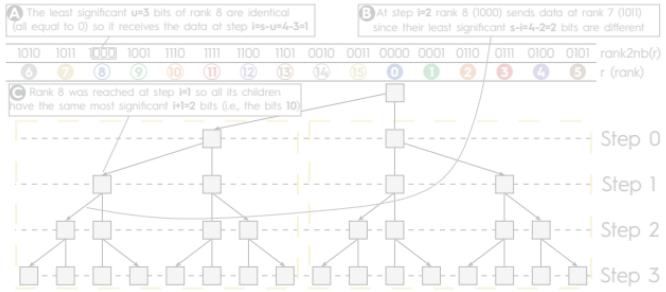


Figure 4: A 16-node (order 4) distance-halving *Bine* tree.

Once a rank receives the data, in each subsequent step i it forwards the data to rank:

$$q = nb2rank(rank2nb(r) \oplus \overbrace{111\ldots1}^{s-i \text{ bits}}) \quad (1)$$

```
int MPI_Bcast(void *buf, int count, MPI_Datatype dt, int root,
              MPI_Comm comm){
  int p, r;
  MPI_Comm_size(comm, &p);
  MPI_Comm_rank(comm, &r);
  int recvd = (root == rank);
  int vrank = mod(r - root, p);
  int vrank_nb = rank2nb(vrank, p); // negabinary repr. of vrank
  int mask = 0x1 << (int) log2(p) - 1; // mask with log2(p) LSBs set to 1
  while(mask > 0){
    int mask_lsbs = (mask << 1) - 1; // mask with s-i LSBs set to 1
    int q = nb2rank(vrank_nb ^ mask_lsbs, p); // Eq. 1
    q = mod(q + root, size); // Get the rank id
    if(recvd){ // I send only if I alredy received the data
      MPI_Send(buf, count, dt, q, 0, comm);
    }else{
      int lsbs = vrank_nb & mask_lsbs; // Get the s-i LSBs
      if(!lsbs || lsbs == mask_lsbs){ // check if s-i LSBs all equal
        MPI_Recv(buf, count, dt, q, 0, comm, MPI_STATUS_IGNORE);
        recvd = 1;
      }
    }
    mask >>= 1;
  }
  return MPI_SUCCESS;
}
```

Listing 1: *Bine* tree distance-halving broadcast code.

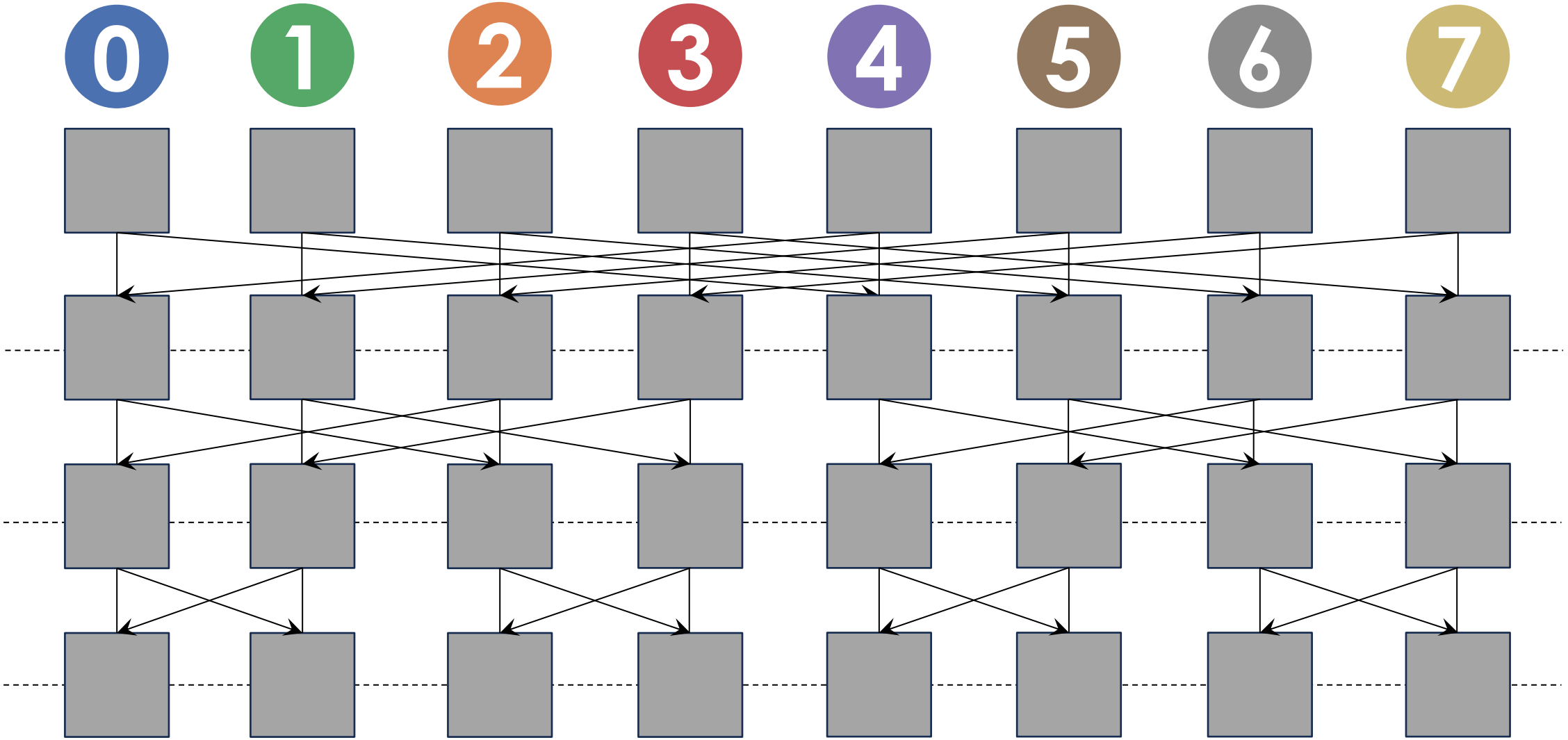
2.3.3 ***Determining Ranks in a Subtree.*** As we will discuss in Sec. 4, collectives like scatter (Sec. 4.2) require identifying all ranks in the subtree rooted at a given rank. After receiving data at step i, a rank applies XOR on its least significant bits to determine its next destination. This process recurses, and at each step, the $i + 1$ most significant bits remain unchanged. As a result, all descendants of a rank share the same $i + 1$ leading bits in their negabinary representation. For instance, in a 16-node *Bine* tree, rank 8 is reached at step $i = 1$, so all the ranks in the subtree rooted in 8 share its two most significant bits (i.e., 10, as shown in Fig. 4, C).

2.4 Advantages over Binomial Trees

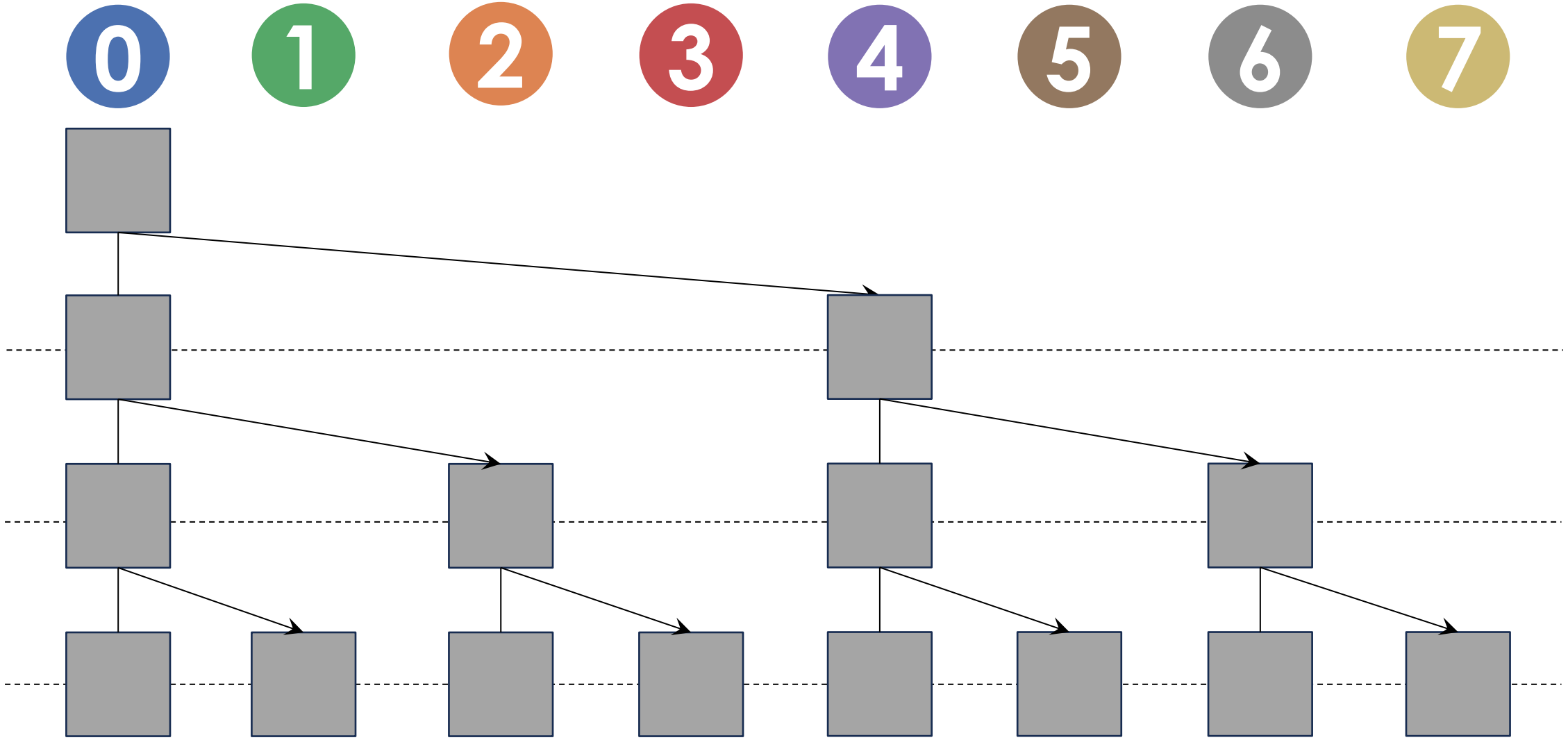
We know quantify the maximum reduction in global links traffic both theoretically (Sec. 2.4.1) and empirically (Sec. 2.4.2).

2.4.1 ***Theoretical Bound Computation.*** To explain why we expect *Bine* trees to reduce the number of bytes on global links compared to standard binomial trees, we start by considering the dis-

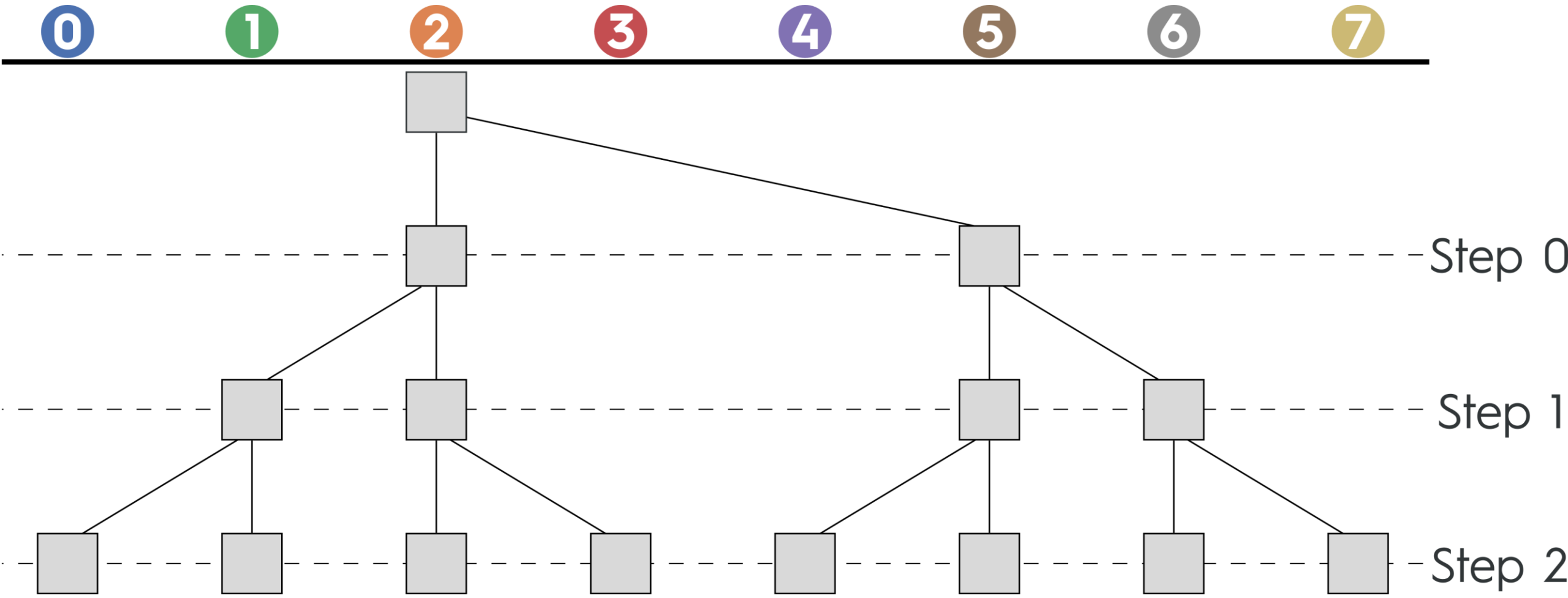
Butterflies



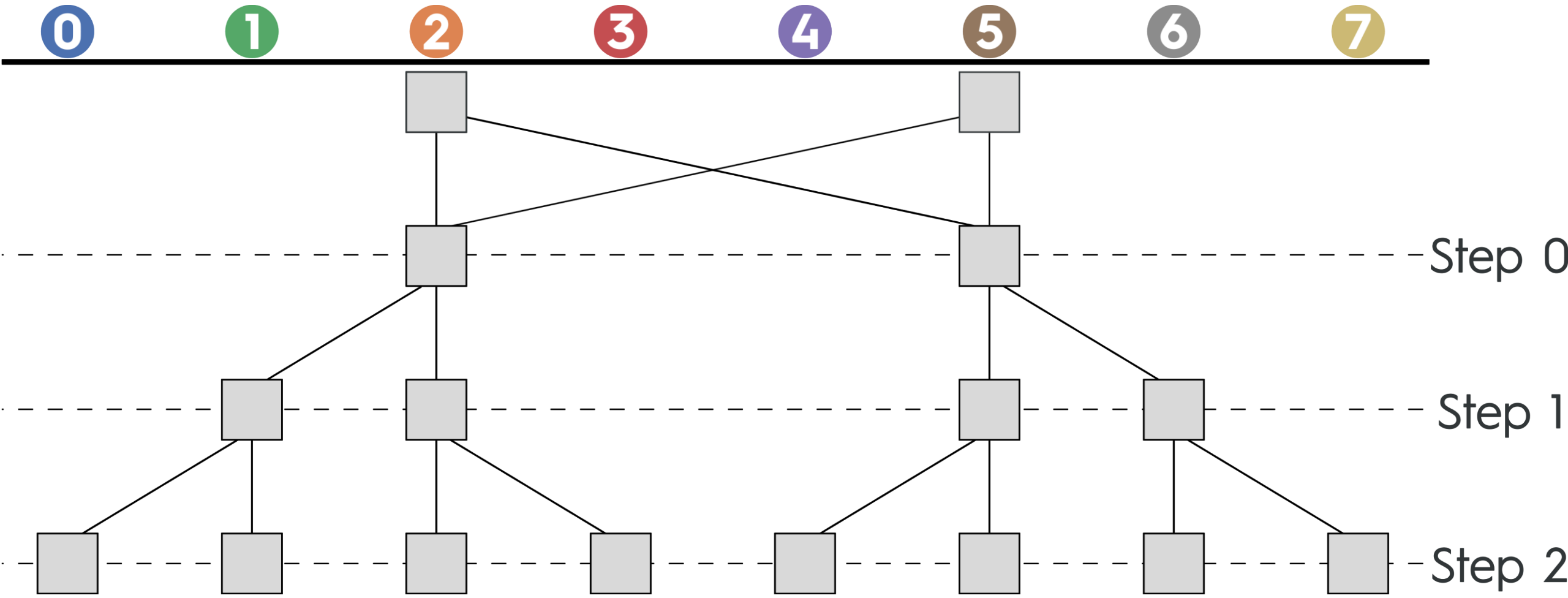
Butterflies



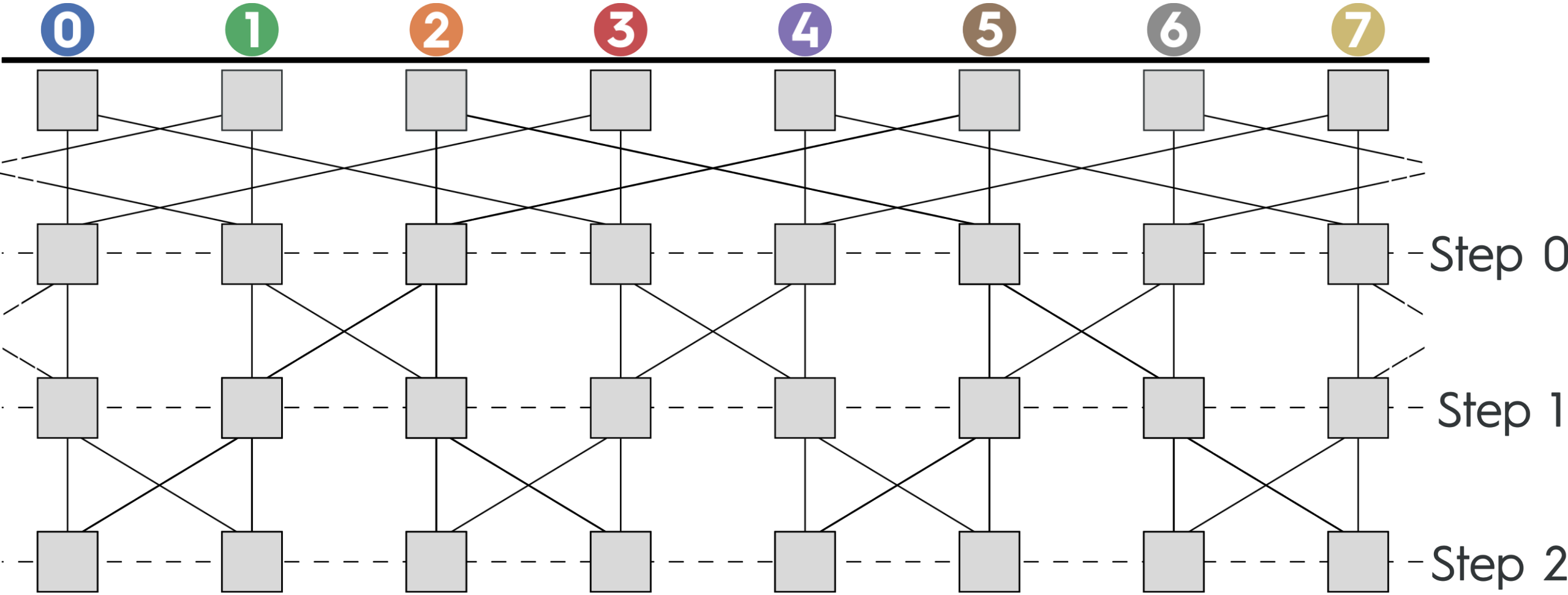
Bine Butterflies



Bine Butterflies



Bine Butterflies



Experimental Evaluation

Experimental Evaluation

System (Top500 Rank)	Topology	MPI Library
LUMI (#8)	Dragonfly	Cray MPICH v8.1.29
Leonardo (#9)	Dragonfly+	Open MPI v4.1.6
MareNostrum 5 (#11)	2:1 Oversubscribed Fat Tree	Open MPI v4.1.5
Fugaku (#6)	6D Torus	Fujitsu MPI v4.0.1

Bine vs Binomial

Bine vs. Binomial Trees/Butterflies: Leonardo

From 16 to 2,048 nodes, from 32 B to 512 MiB

Bine vs. Binomial Trees/Butterflies: Leonardo

From 16 to 2,048 nodes, from 32 B to 512 MiB

Coll.	% Win	Avg/Max Perf. Gain	% Loss	Avg/Max Perf. Drop	Avg/Max Traffic Red.
Allreduce	67%	11%/46%	20%	10%/13%	19%/26%
Allgather	91%	9%/54%	0%	0%/0%	19%/26%
Red.-Scat.	71%	4%/12%	14%	6%/6%	17%/23%
Alltoall	79%	25%/70%	0%	0%/0%	15%/15%
Bcast	94%	41%/148%	0%	0%/0%	89%/92%
Reduce	44%	43%/72%	33%	7%/9%	13%/19%
Gather	93%	23%/71%	7%	12%/12%	12%/20%
Scatter	93%	22%/63%	0%	0%/0%	12%/20%

Bine vs. Binomial Trees/Butterflies: Leonardo

From 16 to 2,048 nodes, from 32 B to 512 MiB

Coll.	% Win	Avg/Max Perf. Gain	% Loss	Avg/Max Perf. Drop	Avg/Max Traffic Red.
Allreduce	67%	11%/46%	20%	10%/13%	19%/26%
Allgather	91%	9%/54%	0%	0%/0%	19%/26%
Red.-Scat.	71%	4%/12%	14%	6%/6%	17%/23%
Alltoall	79%	25%/70%	0%	0%/0%	15%/15%
Bcast	94%	41%/148%	0%	0%/0%	89%/92%
Reduce	44%	43%/72%	33%	7%/9%	13%/19%
Gather	93%	23%/71%	7%	12%/12%	12%/20%
Scatter	93%	22%/63%	0%	0%/0%	12%/20%

Bine vs. Binomial Trees/Butterflies: Leonardo

From 16 to 2,048 nodes, from 32 B to 512 MiB

Coll.	% Win	Avg/Max Perf. Gain	% Loss	Avg/Max Perf. Drop	Avg/Max Traffic Red.
Allreduce	67%	11%/46%	20%	10%/13%	19%/26%
Allgather	91%	9%/54%	0%	0%/0%	19%/26%
Red.-Scat.	71%	4%/12%	14%	6%/6%	17%/23%
Alltoall	79%	25%/70%	0%	0%/0%	15%/15%
Bcast	94%	41%/148%	0%	0%/0%	89%/92%
Reduce	44%	43%/72%	33%	7%/9%	13%/19%
Gather	93%	23%/71%	7%	12%/12%	12%/20%
Scatter	93%	22%/63%	0%	0%/0%	12%/20%

Bine vs. Binomial Trees/Butterflies: Leonardo

From 16 to 2,048 nodes, from 32 B to 512 MiB

Coll.	% Win	Avg/Max Perf. Gain	% Loss	Avg/Max Perf. Drop	Avg/Max Traffic Red.
Allreduce	67%	11%/46%	20%	10%/13%	19%/26%
Allgather	91%	9%/54%	0%	0%/0%	19%/26%
Red.-Scat.	71%	4%/12%	14%	6%/6%	17%/23%
Alltoall	79%	25%/70%	0%	0%/0%	15%/15%
Bcast	94%	41%/148%	0%	0%/0%	89%/92%
Reduce	44%	43%/72%	33%	7%/9%	13%/19%
Gather	93%	23%/71%	7%	12%/12%	12%/20%
Scatter	93%	22%/63%	0%	0%/0%	12%/20%

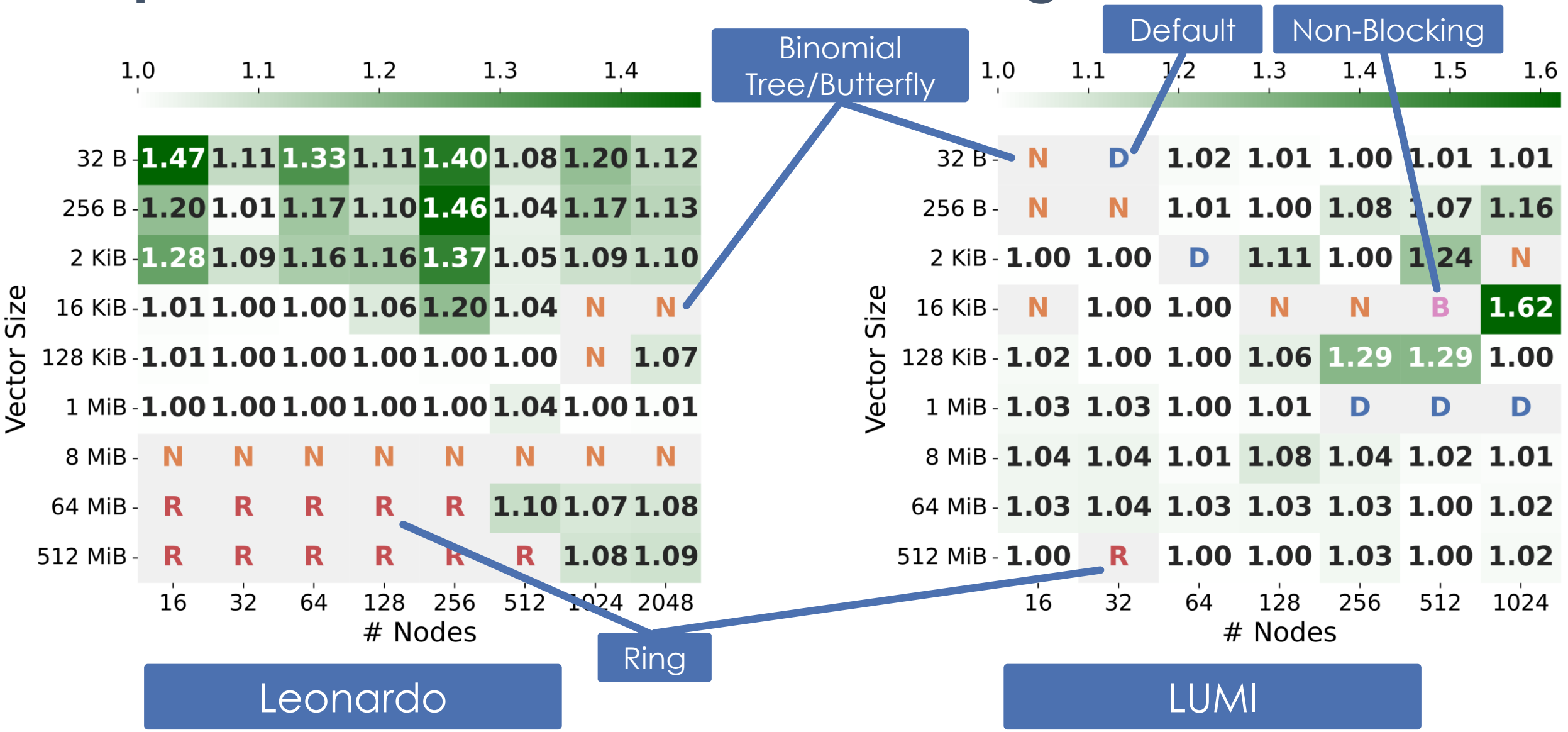
Bine vs. Binomial Trees/Butterflies: Leonardo

From 16 to 2,048 nodes, from 32 B to 512 MiB

Coll.	% Win	Avg/Max Perf. Gain	% Loss	Avg/Max Perf. Drop	Avg/Max Traffic Red.
Allreduce	67%	11%/46%	20%	10%/13%	19%/26%
Allgather	91%	9%/54%	0%	0%/0%	19%/26%
Red.-Scat.	71%	4%/12%	14%	6%/6%	17%/23%
Alltoall	79%	25%/70%	0%	0%/0%	15%/15%
Bcast	94%	41%/148%	0%	0%/0%	89%/92%
Reduce	44%	43%/72%	33%	7%/9%	13%/19%
Gather	93%	23%/71%	7%	12%/12%	12%/20%
Scatter	93%	22%/63%	0%	0%/0%	12%/20%

Bine vs all (including Binomial)

Comparison with State-of-the-art Algorithms



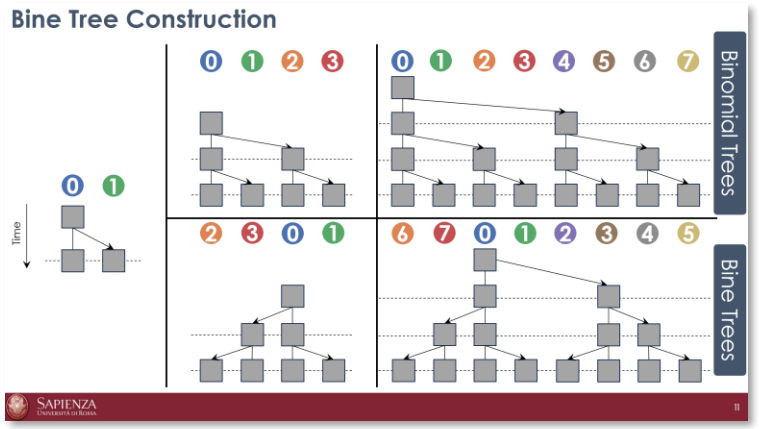
Conclusions

Additional Content

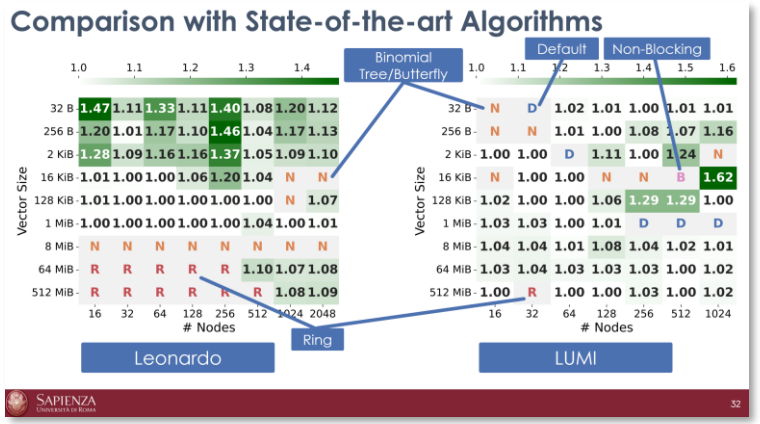
Correctness Proofs

Optimizations for Torus Networks

Hierarchical Multi-GPU Design and Evaluation



Bine trees have **shorter modulo distance** than standard binomial trees



+60% performance on Leonardo
+80% on LUMI
+2x on MareNostrum5
+5x on Fugaku

Paper

Source Code

Leonardo/LUMI/MN5

Fugaku

SAPIENZA UNIVERSITÀ DI ROMA
desensi@di.uniroma1.it
https://danieledesensi.github.io