Introduction to Parallel Processing

Lecture 7 : Performance of MPI Operations

Professor Amanda Bienz

Simple Performance Model

- **Postal model** : $T = \alpha \cdot n + \beta \cdot s$
  - n : number of messages
  - s : number of bytes communicated
  - α : latency, per-message startup cost
  - β : inverse bandwidth, per-byte transport cost

Point-to-Point Communication Costs

- How many messages am I sending?
- What are the total number of bytes?
- For now, calculate cost with $T = \alpha \cdot n + \beta \cdot s$
- Example : Each process sends messages to neighbors (on the left and right). Each message has 10 doubles. If my latency is 1e-6 and inverse bandwidth is 1e-9, how long does it take to communicate?
    - Do I calculate the model for every process?
    - Which do I actually care about?

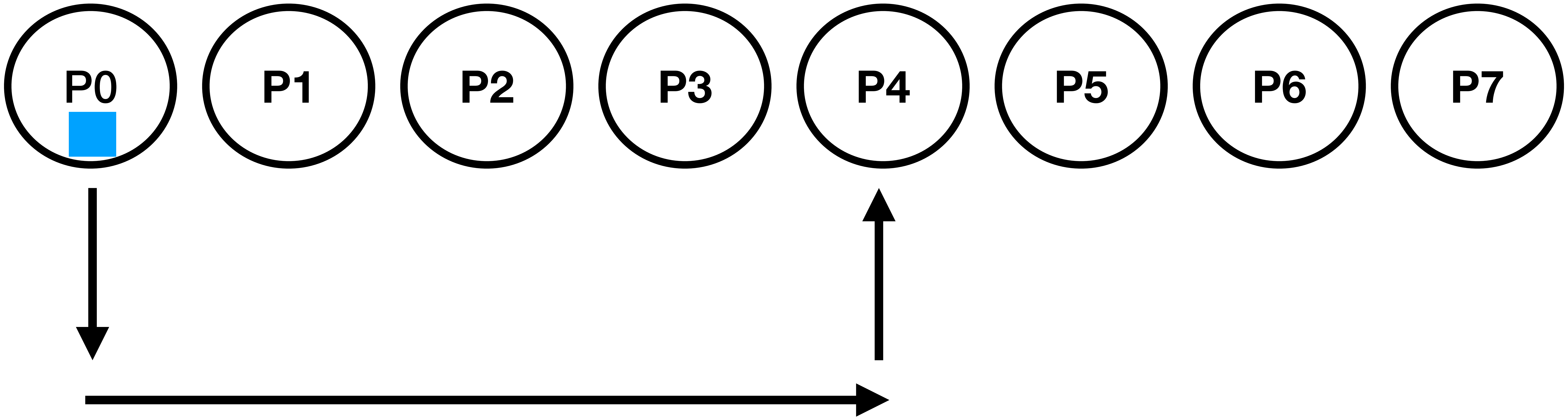
Simple Performance Analysis

- Typically, α is much larger than β
- As computers advance, bandwidth is getting even faster!
- Latency is not advancing at the same rate. At times, it is actually getting slower.
- A large number of small messages is very expensive!

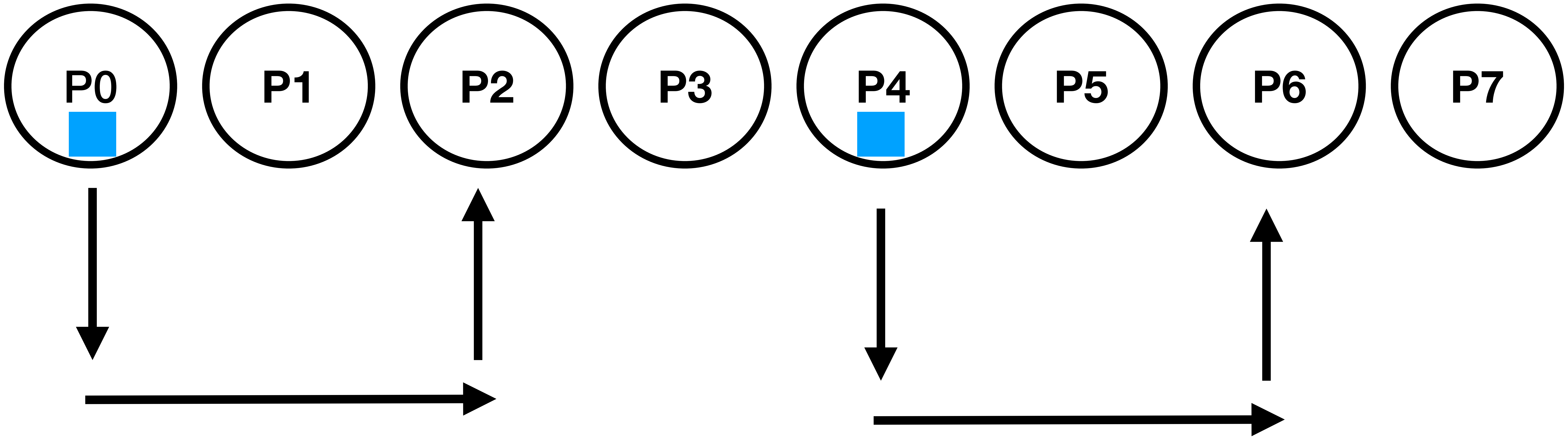
Broadcast

- Have data one on process, want to send all data to every other process
- One option, just send ‘num_procs - 1’ messages at one time
- Is this a good option?

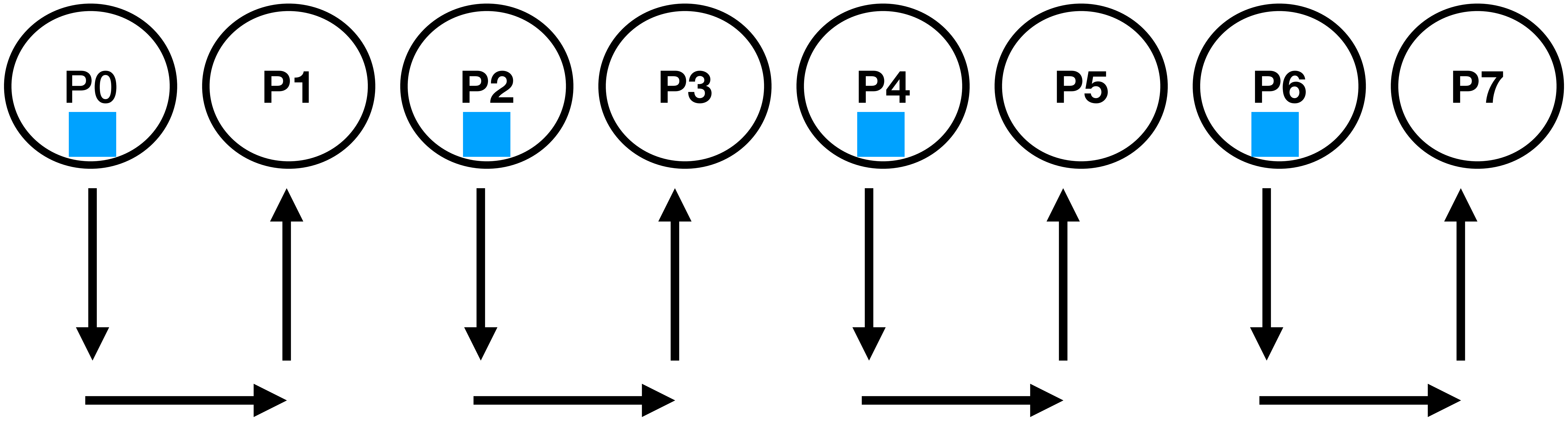
Binomial Tree Algorithm



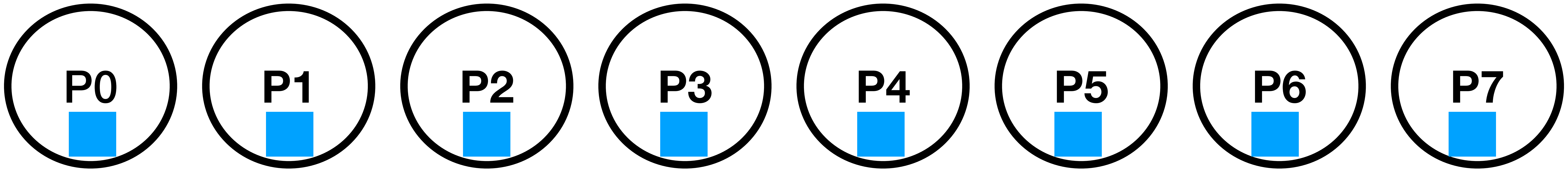
Binomial Tree Algorithm



Binomial Tree Algorithm

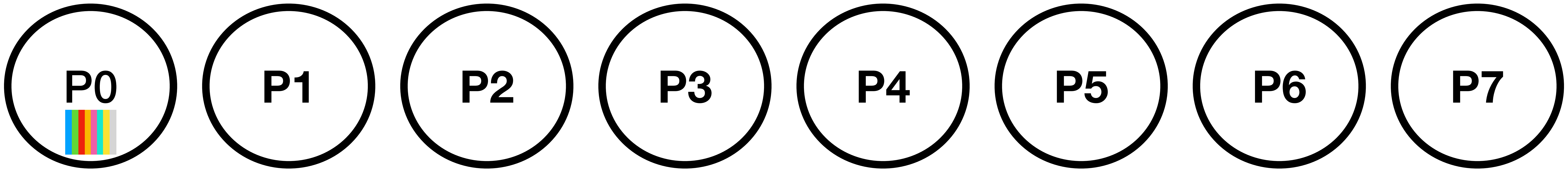


Binomial Tree Algorithm

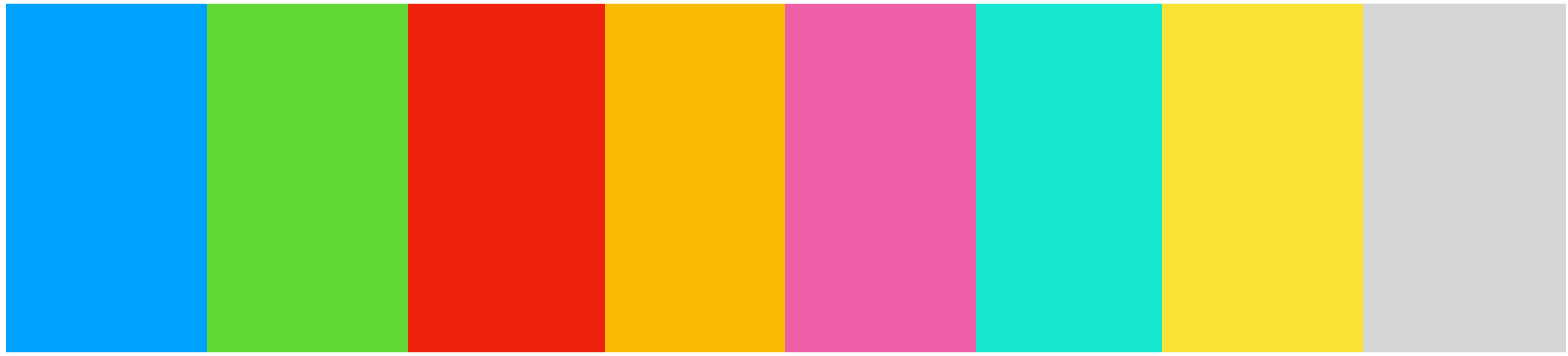


- log(num_procs) steps
- At each step, all n values are communicated
  - i.e. full blue box communicated at each of the log(num_procs) steps
- $T = \log_2(p) \cdot (\alpha + \beta n)$

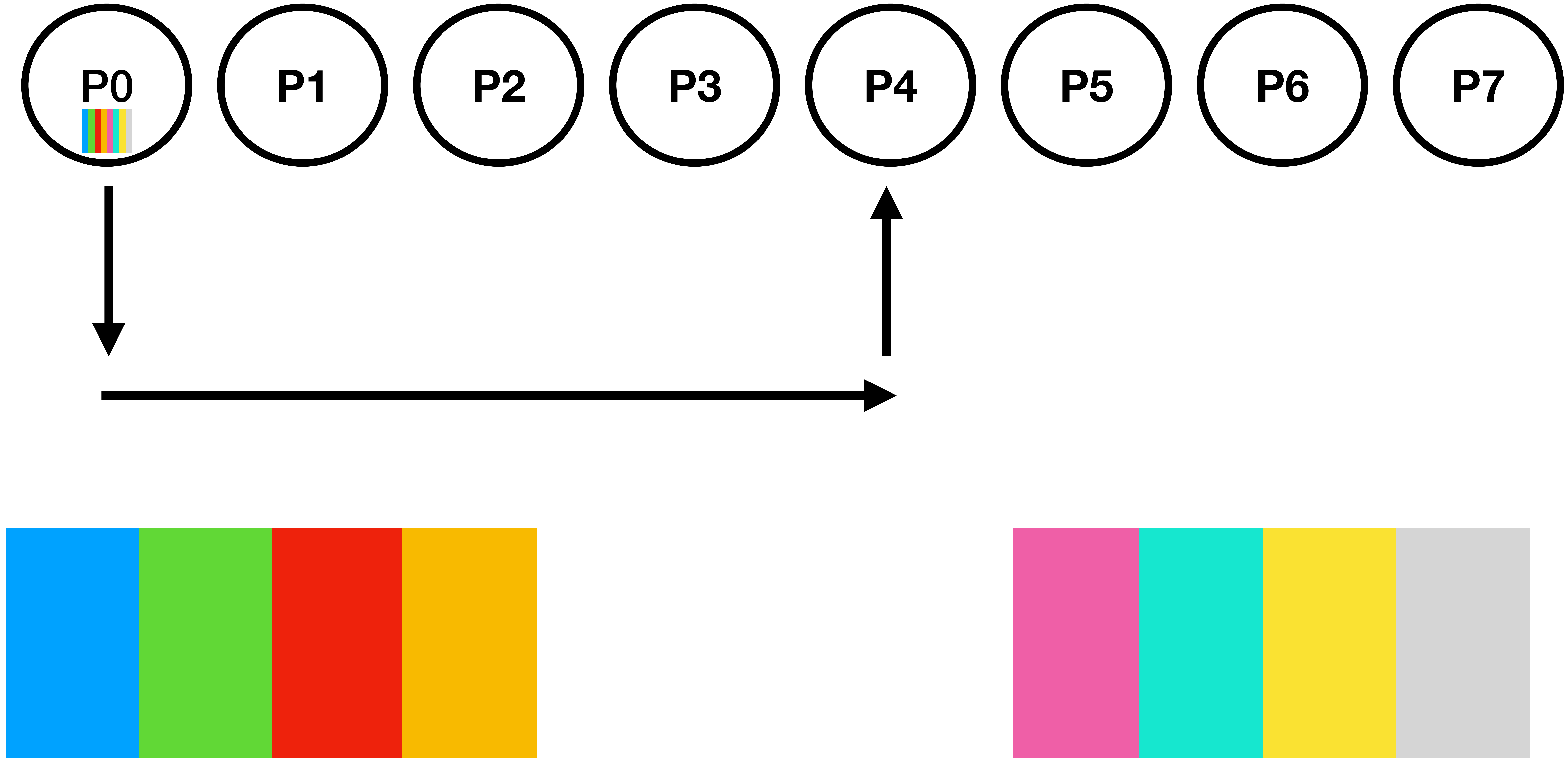
Scatter + Allgather



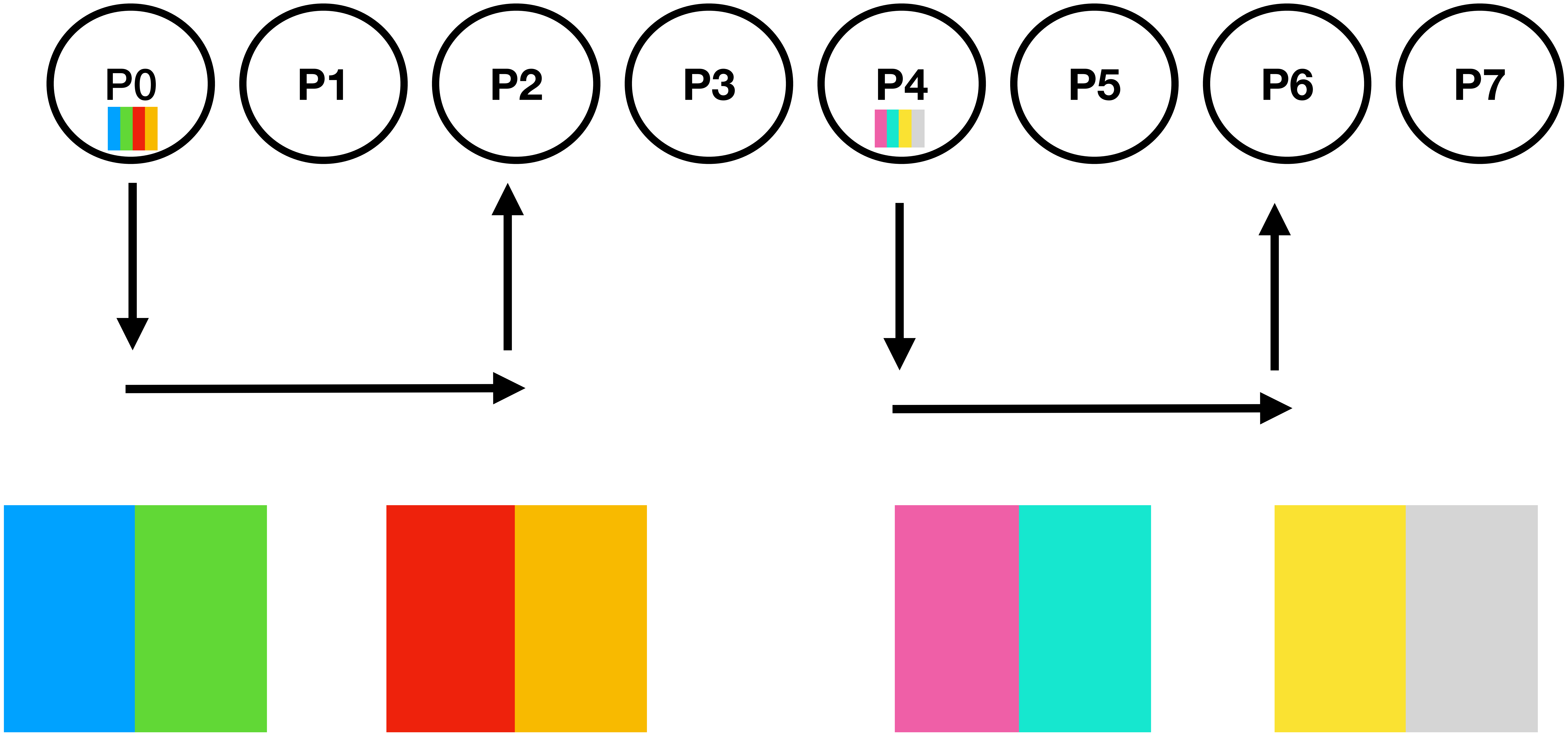
- For larger broadcasts, it is actually cheaper to first scatter the data, and then perform an allgather!



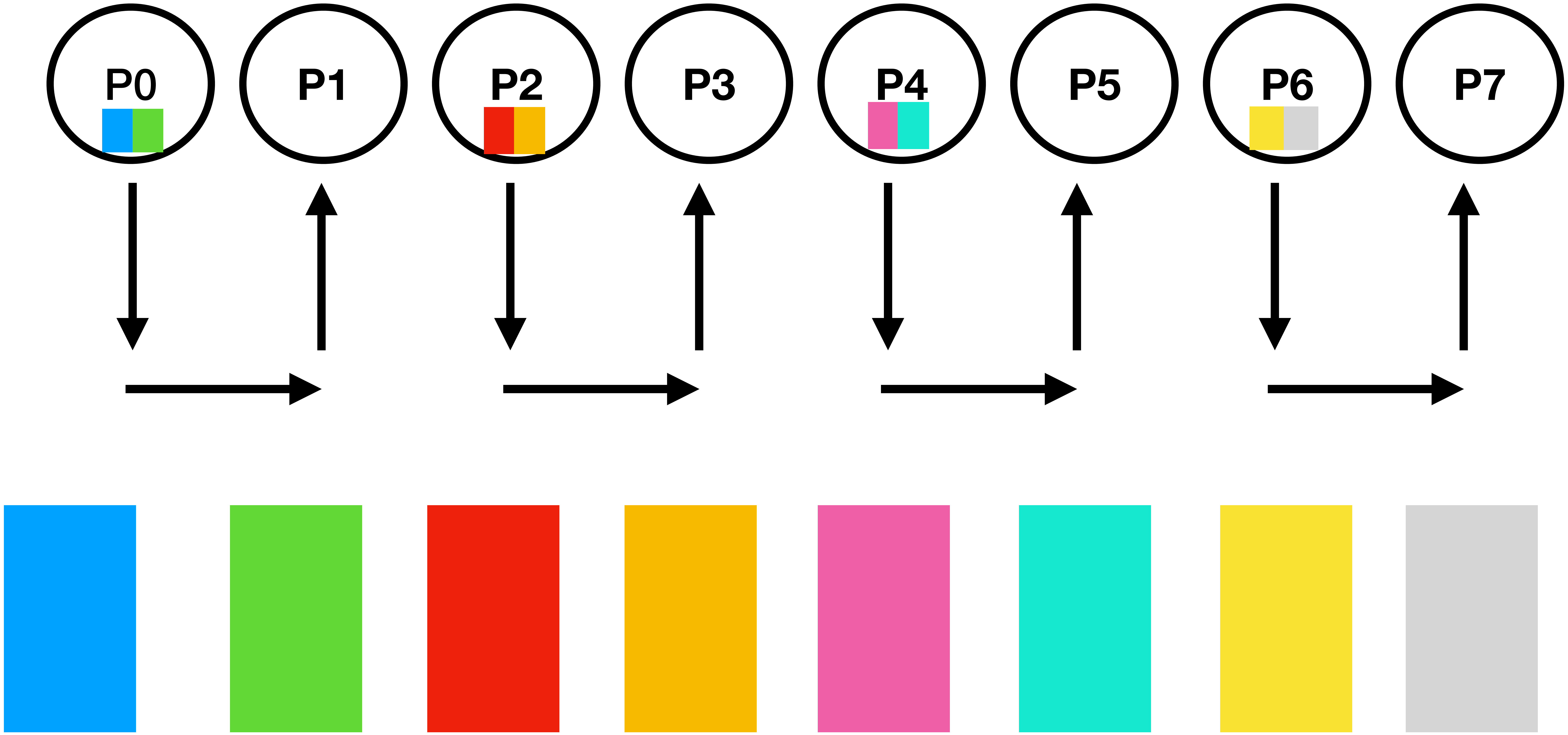
Scatter + Allgather



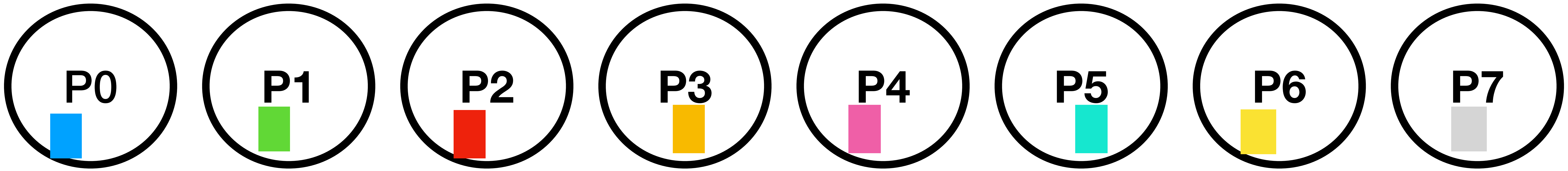
Scatter + Allgather



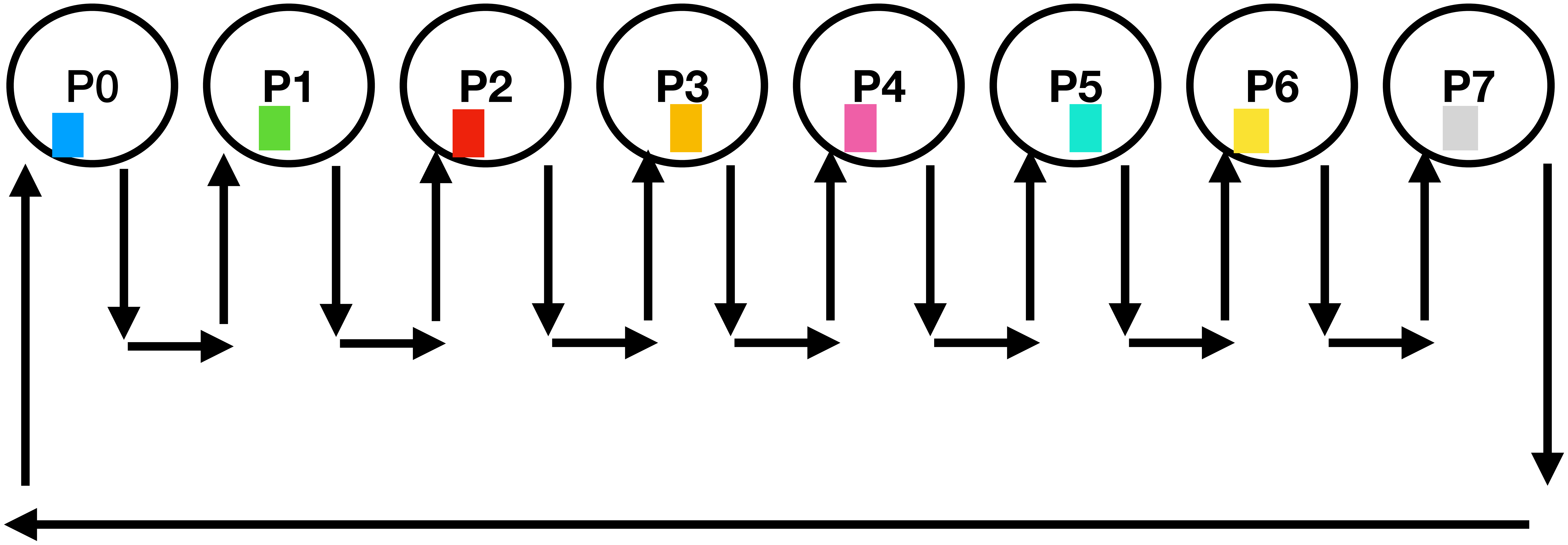
Scatter + Allgather



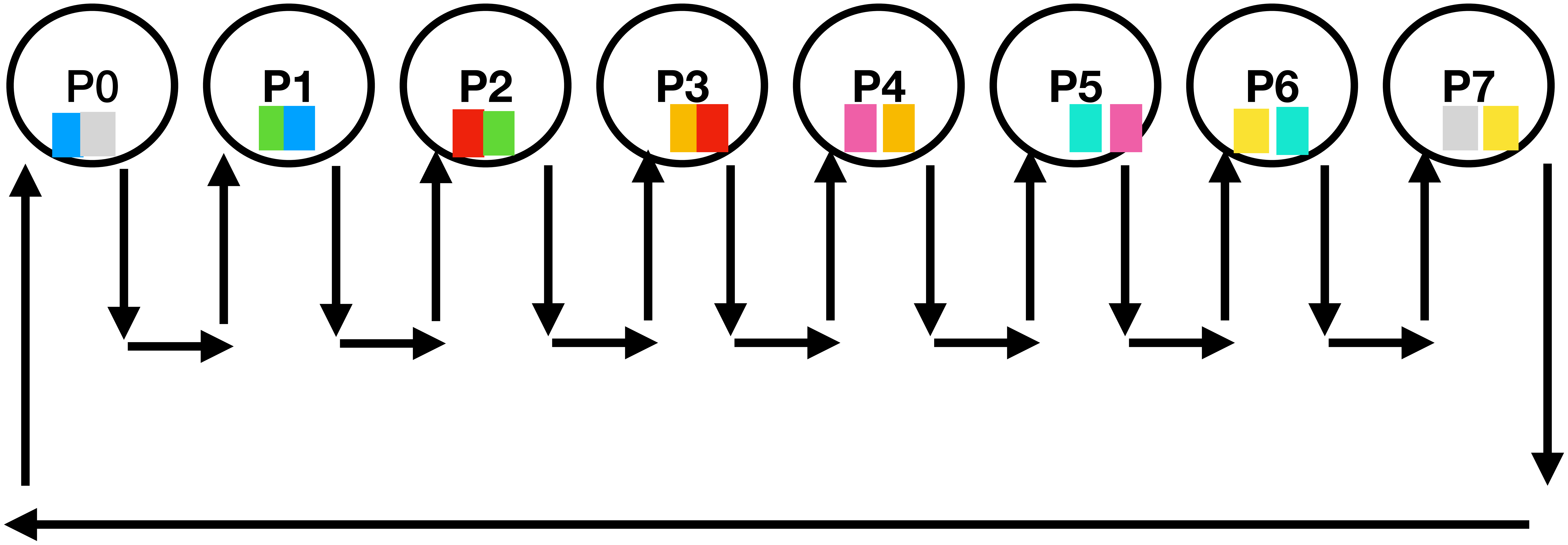
Scatter + Allgather



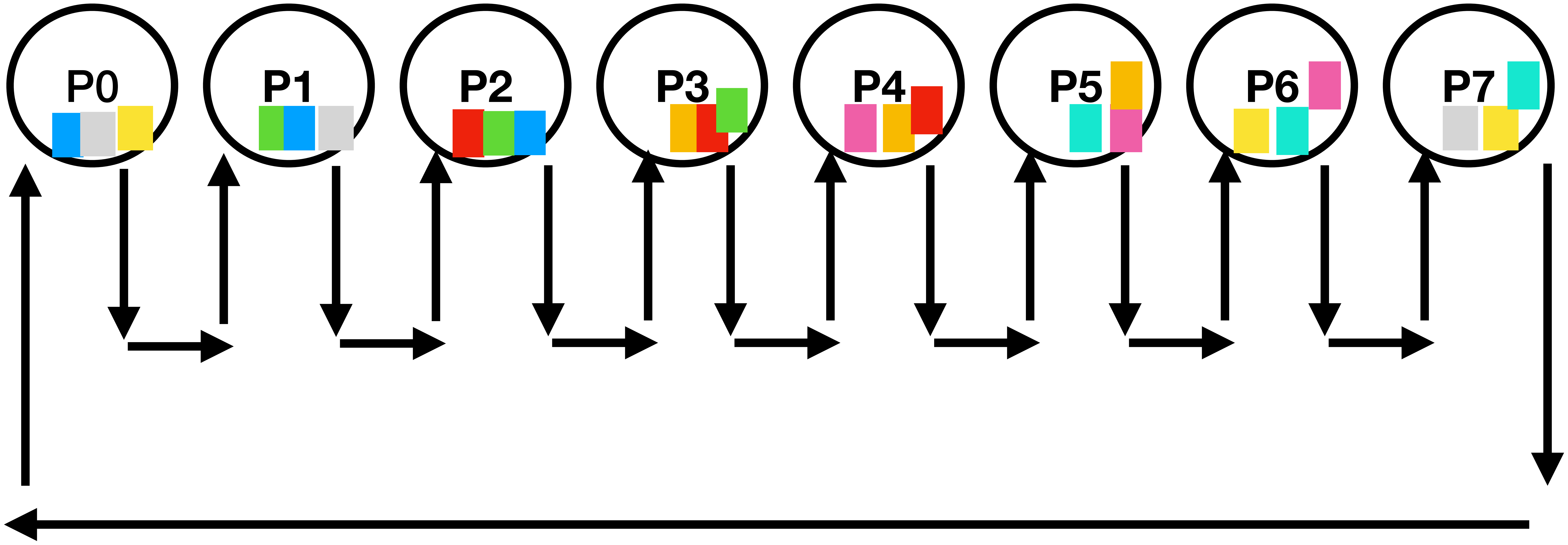
Scatter + Allgather



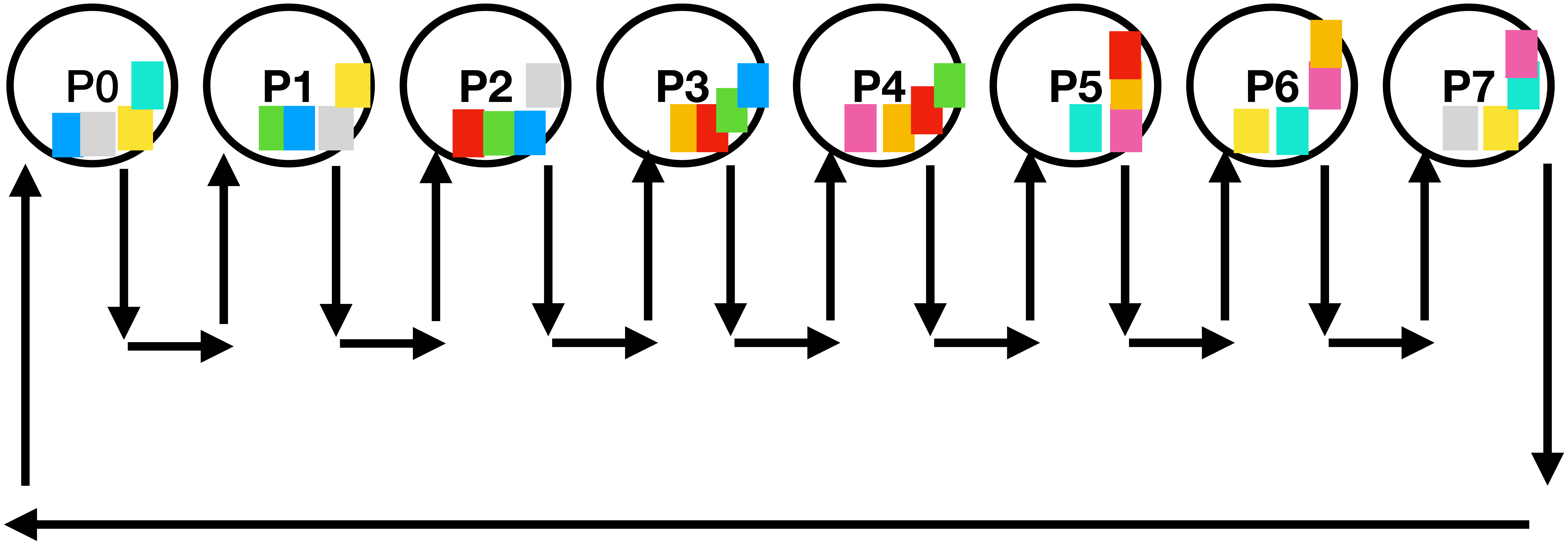
Scatter + Allgather



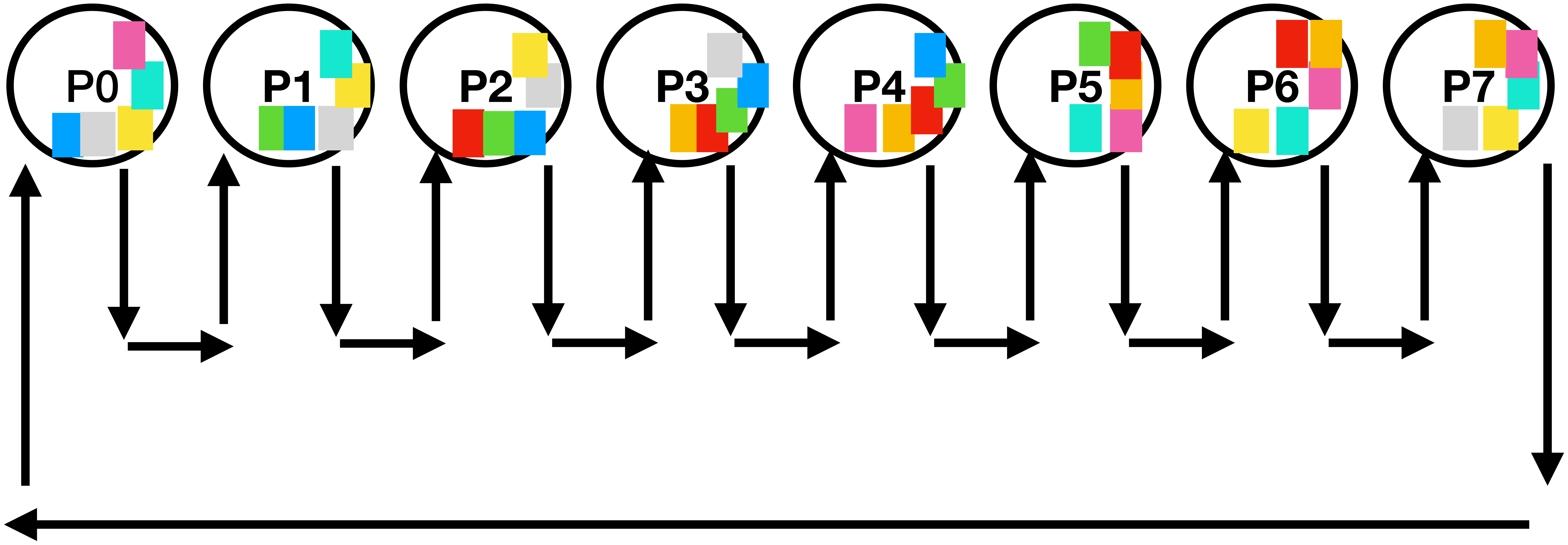
Scatter + Allgather



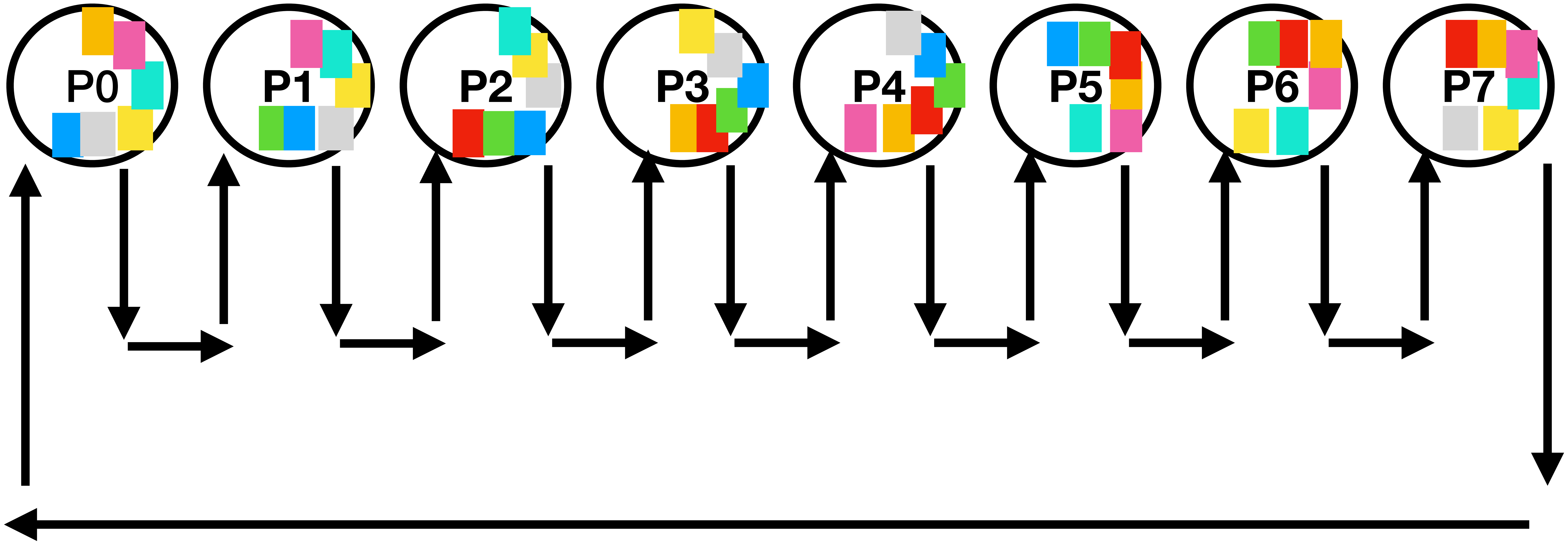
Scatter + Allgather



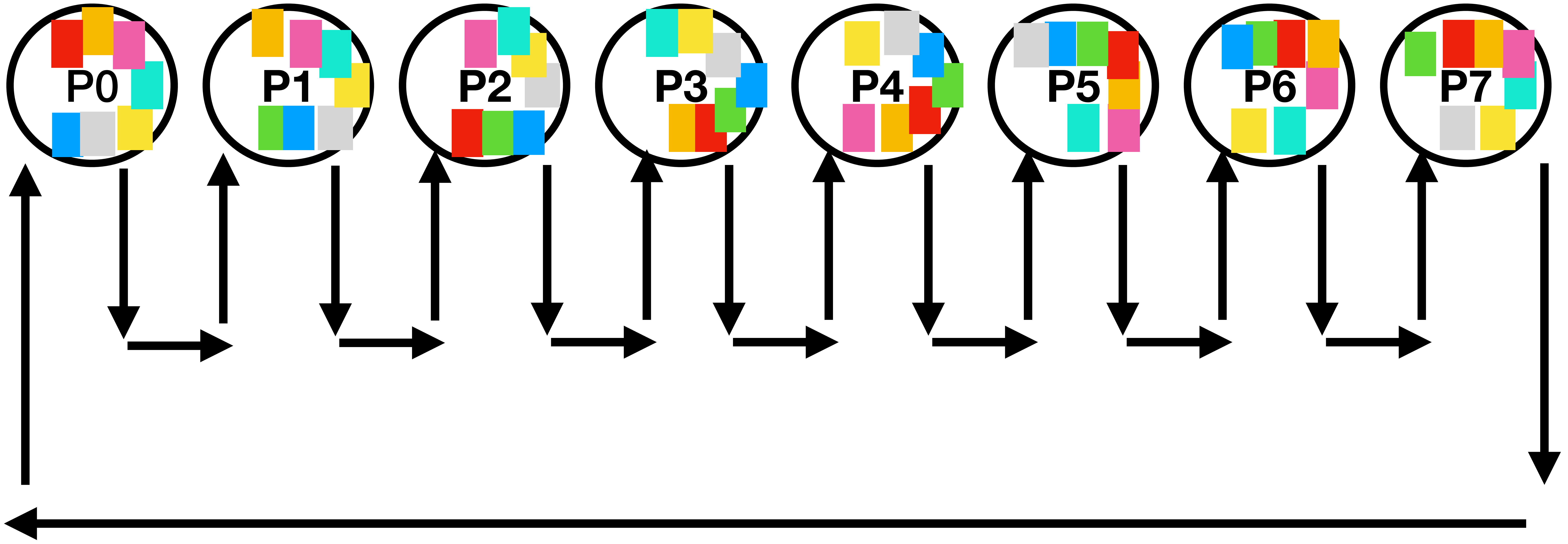
Scatter + Allgather



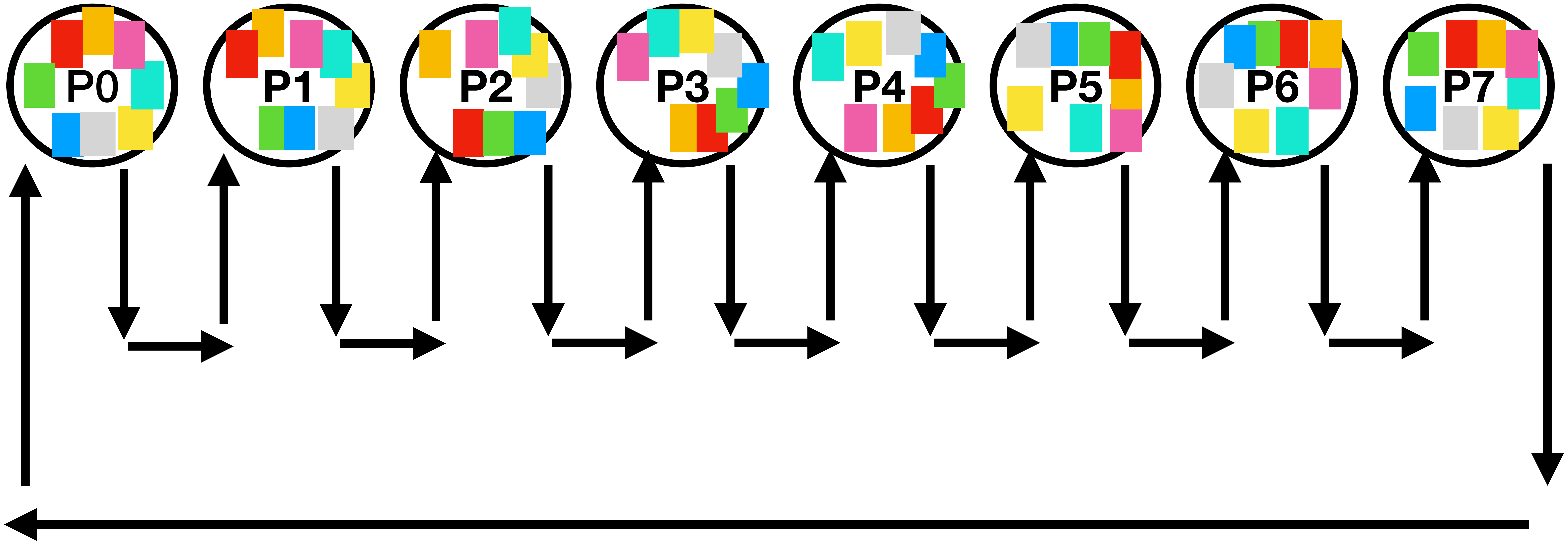
Scatter + Allgather



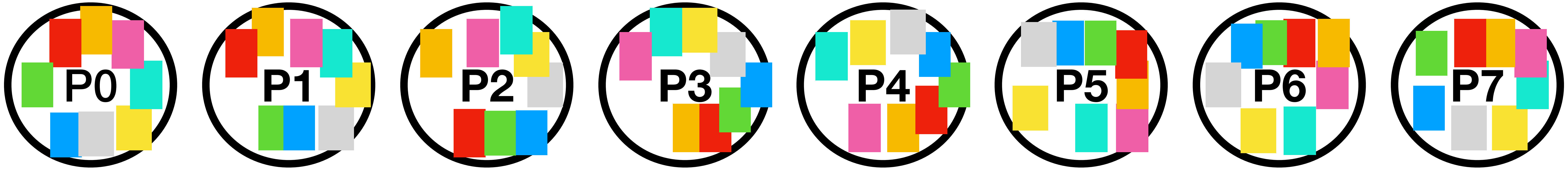
Scatter + Allgather



Scatter + Allgather

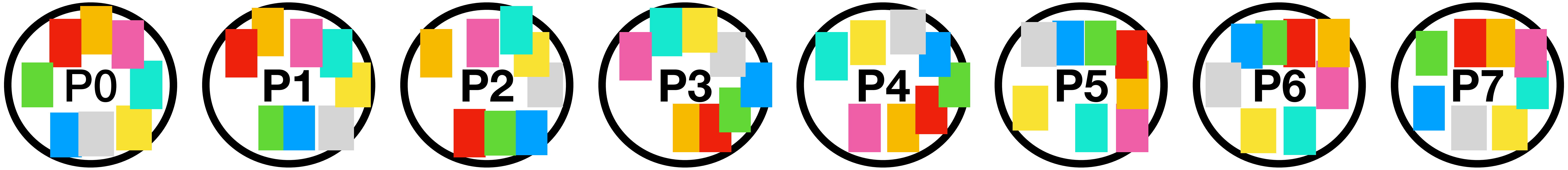


Scatter + Allgather



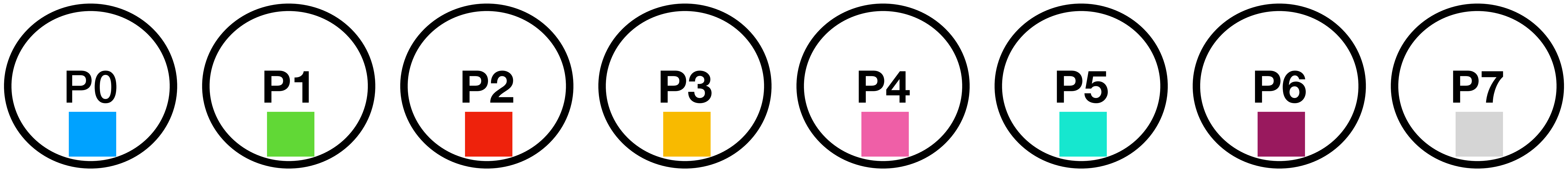
- Scatter has log(num_procs) steps
- Allgather ring algorithm has num_procs - 1 steps
- Scatter sends a total of n values
- Allgather ring algorithm sends n/p values at each step
- $T = (\log_2(p) + p - 1) \cdot \alpha + 2\frac{p-1}{p}n\beta$

Why is Scatter+Allgather ever used?



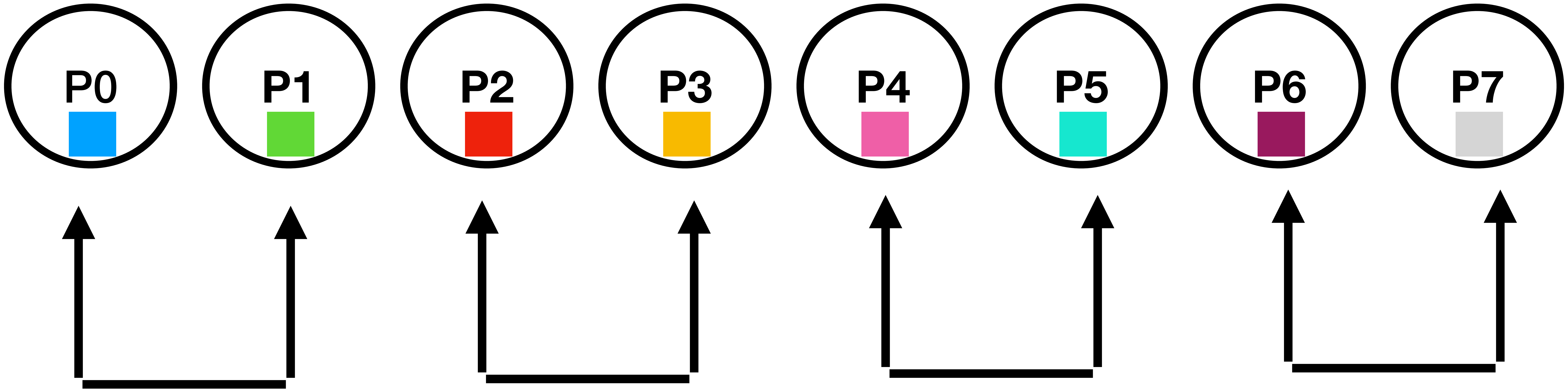
- Binomial Tree : $T = \log_2(p) \cdot (\alpha + \beta n)$
- Scatter + Allgather : $T = (\log_2(p) + p - 1) \cdot \alpha + 2\frac{p-1}{p}n\beta$

Allgather

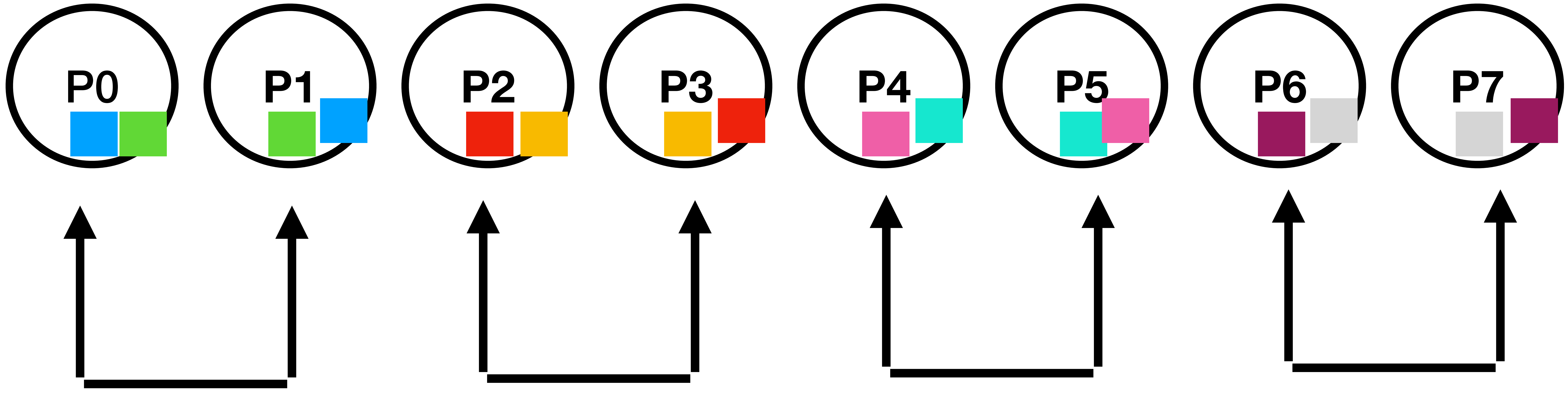


- We’ve talked about the ring algorithm
- Is there a more optimal way to perform all gather (for smaller messages)

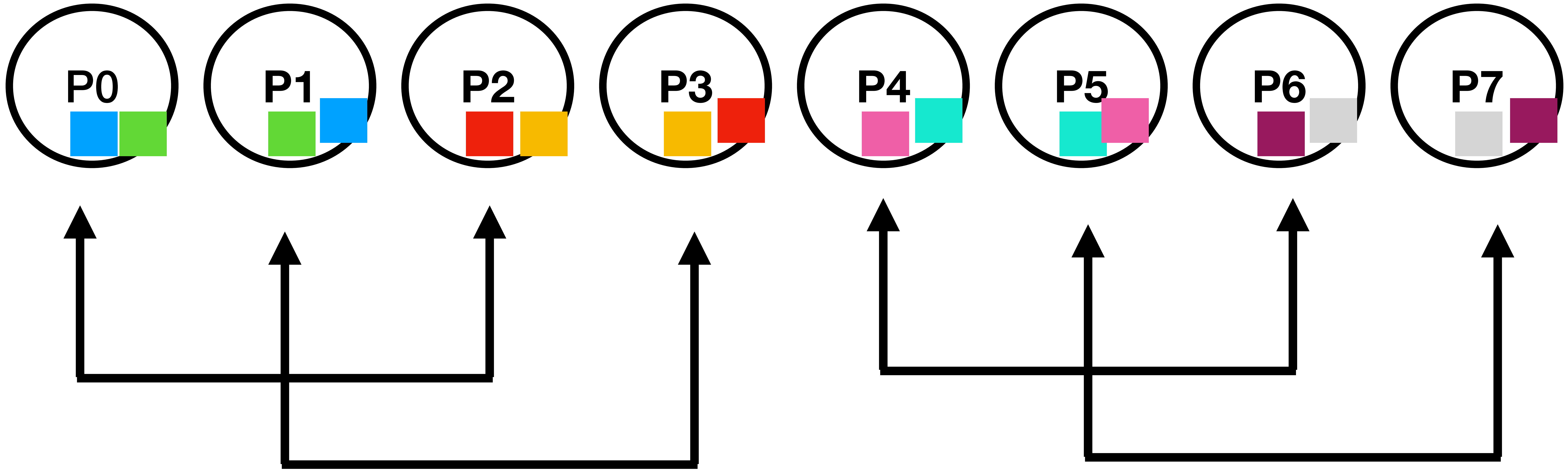
Recursive-Doubling



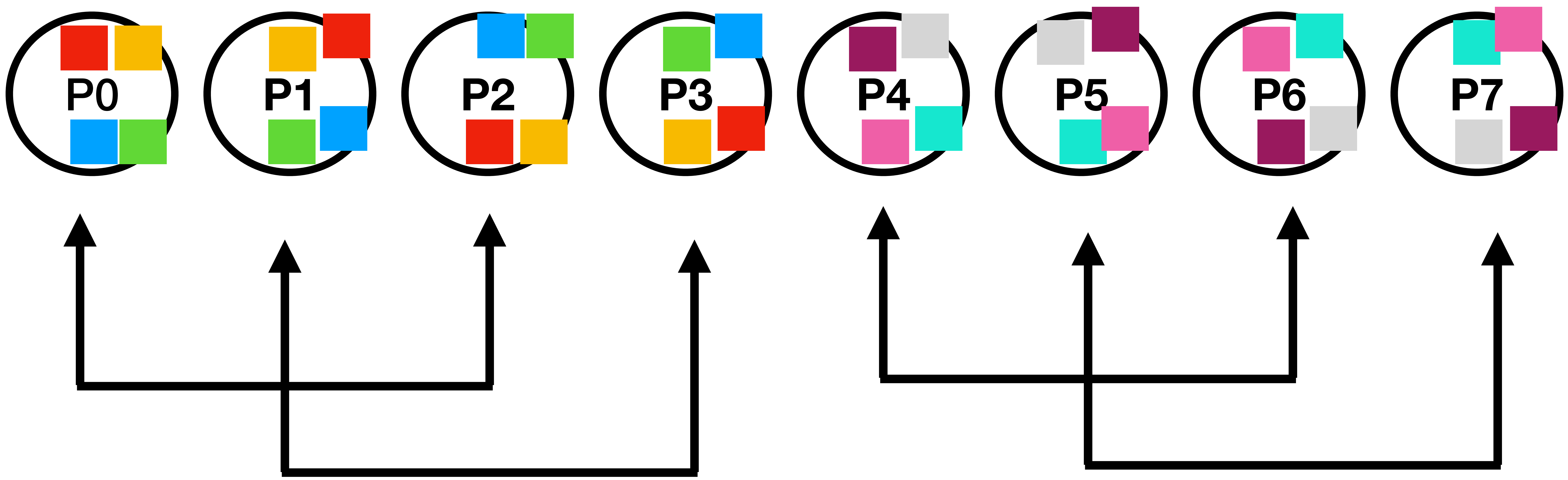
Recursive-Doubling



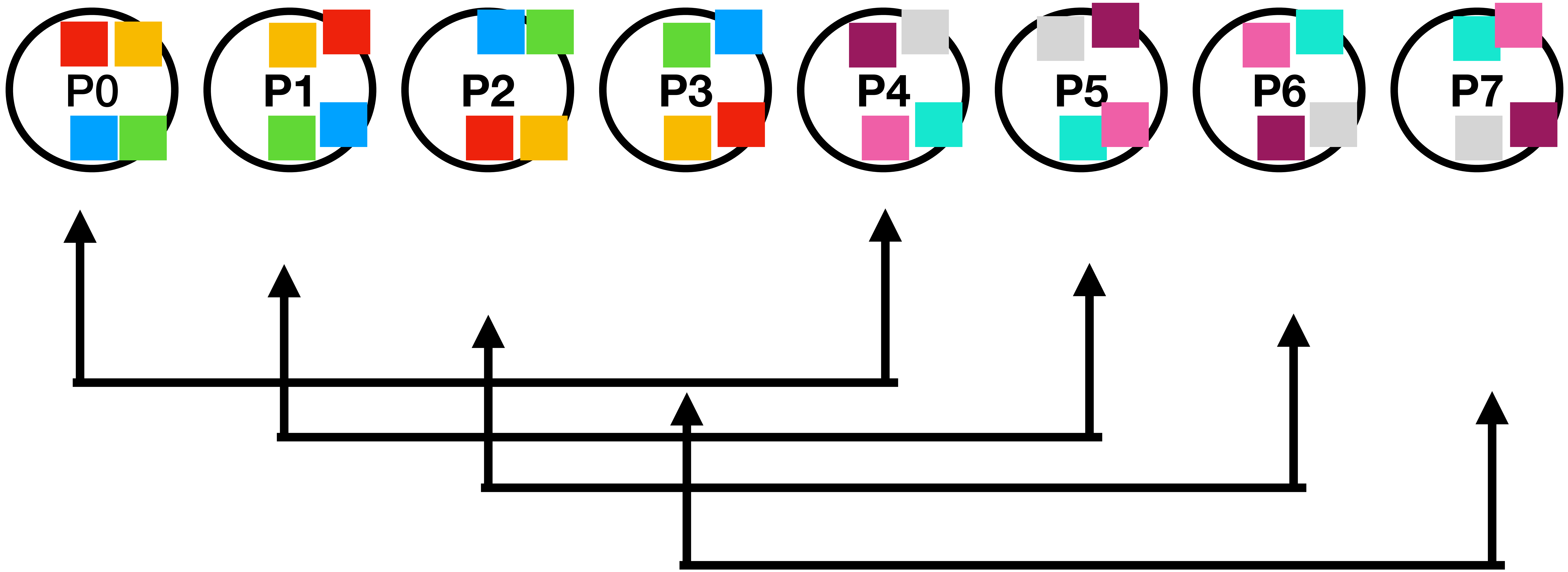
Recursive-Doubling



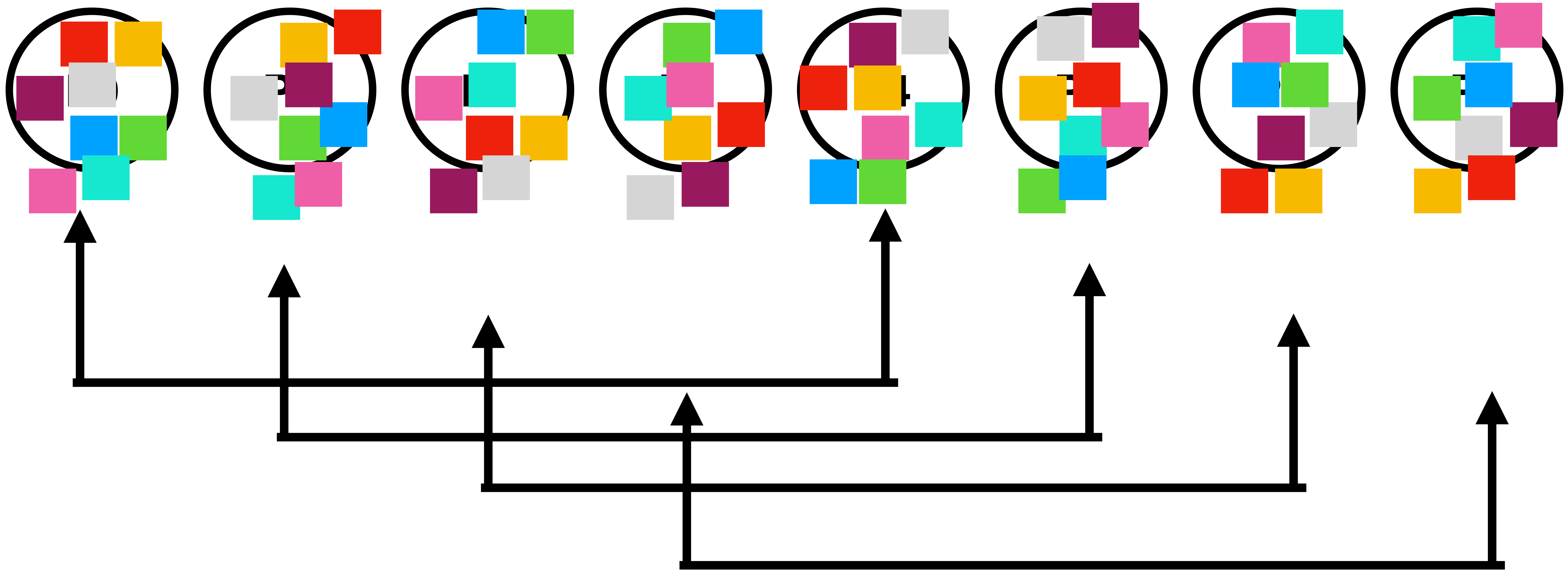
Recursive-Doubling



Recursive-Doubling



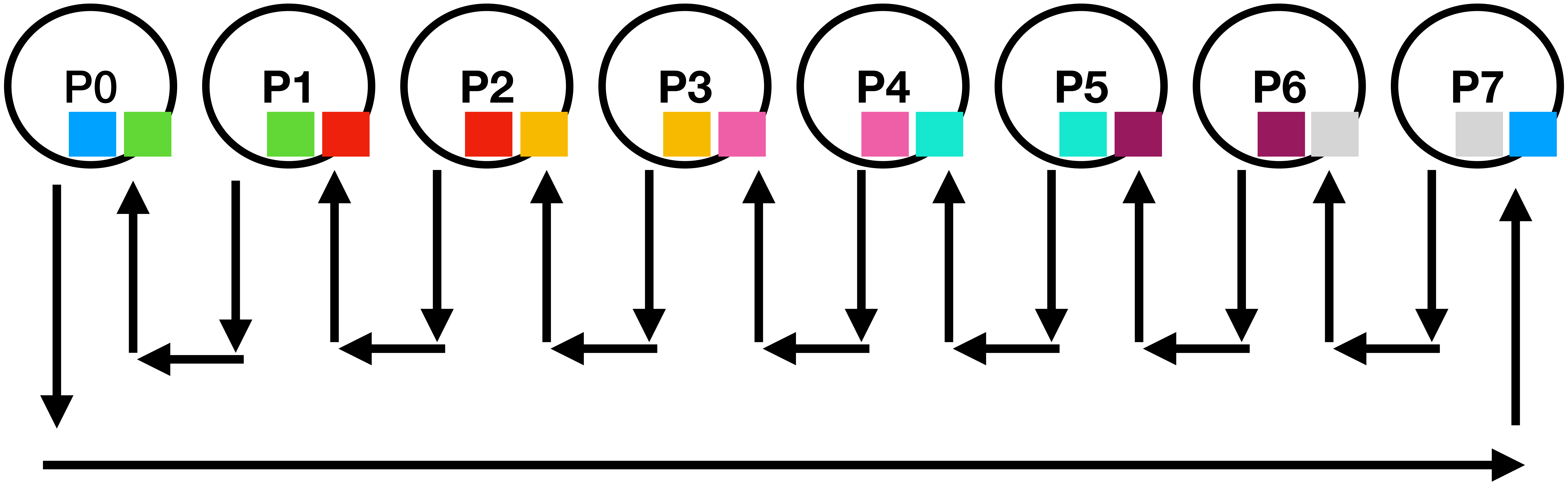
Recursive-Doubling



Recursive Doubling

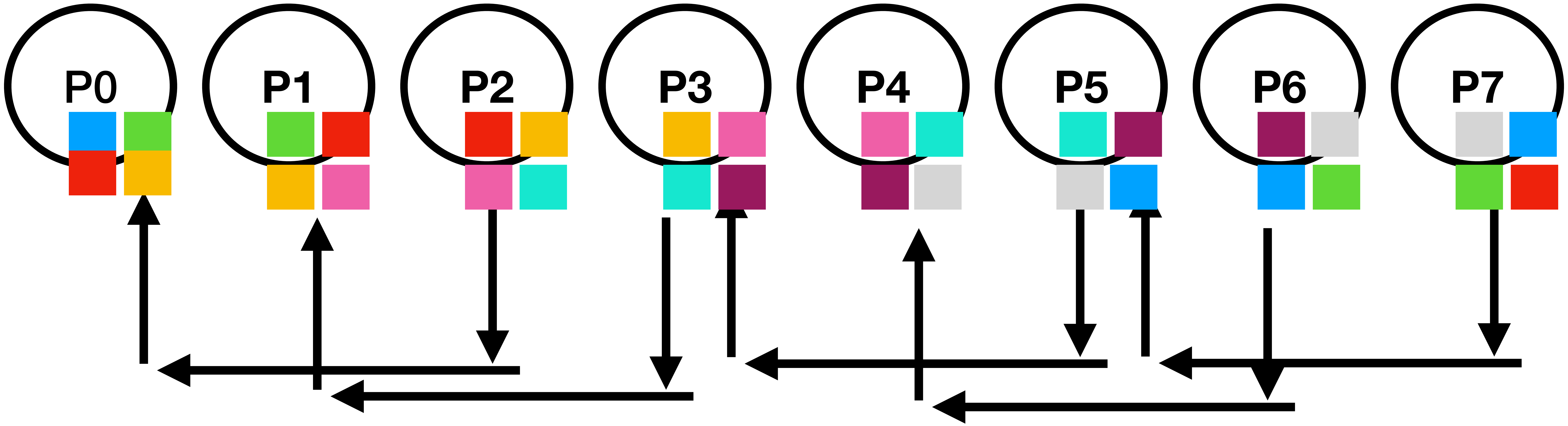
- log(p) steps
- Size of values:
  - Step 1: n/p
  - Step 2: 2n/p
  - Last step: 2^(log(p)-1)*n / p
- $T = \log_2(p) \cdot \alpha + \frac{p-1}{p} n \cdot \beta$

The Bruck Algorithm

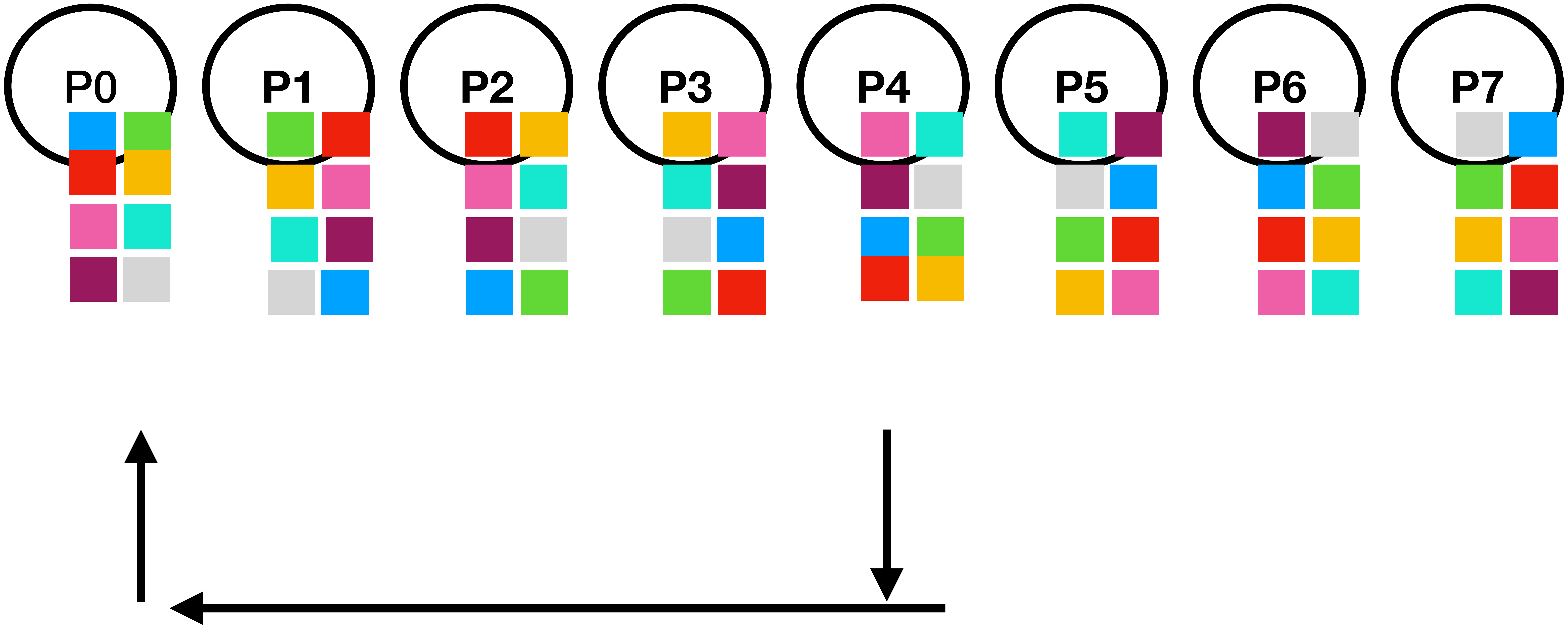


- At each step k, process i sends data to process (i - 2^k)

The Bruck Algorithm



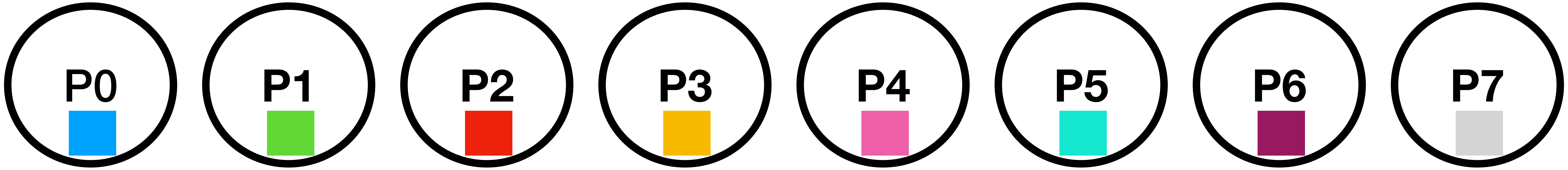
The Bruck Algorithm



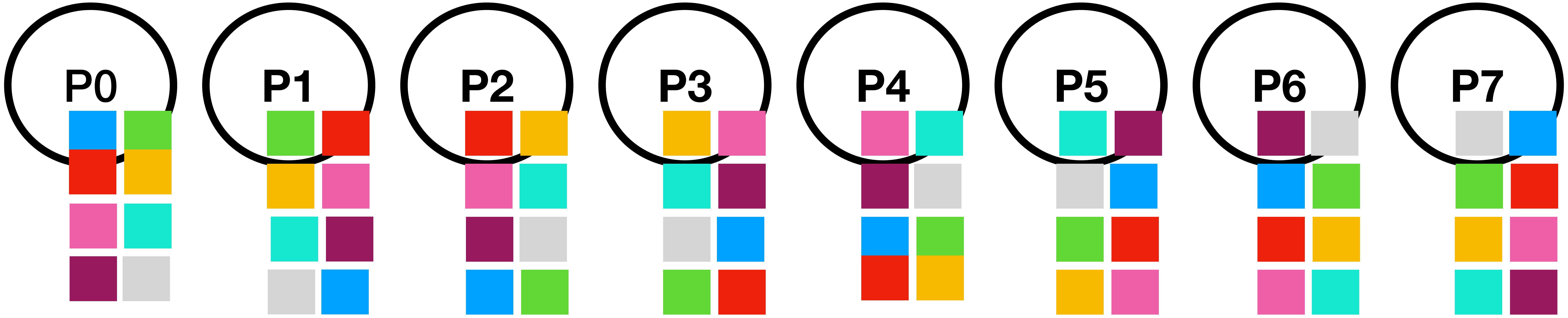
The Bruck Algorithm

- Everything seems exactly the same as recursive doubling
- Same number of steps, same number of values sent
- $T = \log_2(p) \cdot \alpha + \frac{p-1}{p} n \cdot \beta$
- What is the difference?

The Bruck Algorithm



- The Bruck algorithm preserves order of the elements



Allgather Performance

- Recursive Doubling or Bruck : $T = \log_2(p) \cdot \alpha + \frac{p-1}{p} n \cdot \beta$
- Ring : $T = (p-1) \cdot \alpha + \frac{p-1}{p} n\beta$
- Why would you ever use the ring Allgather?
- **We will update our performance models later this semester!**

Other MPI Collectives

- These algorithms are used for a lot of MPI collectives
- **MPI Barrier**: dissemination (what the Bruck algorithm is based off of)
  - At each step k, process i sends to (I+2^k)
- **MPI Allreduce**: recursive doubling for small messages, reduce-scatter for larger messages (similar to the scatter-allgather)
- **MPI Alltoall:** Bruck's algorithm for very small messages. Pairwise exchange for larger messages (in step k, send to rank+k and recv from rank - k)

More Information

- We will go over this more for Wednesday's class
- However, if you are interested in learning more, this is a great paper:
- Optimization of collective communication operations in MPICH: https://www.mcs.anl.gov/~thakur/papers/ijhpca-coll.pdf