A Concurrent Lambda Calculus for MPI

Keith Allen, Dr. Andrew Hirsch

ktallen@buffalo.edu

Upstate PL

August 28, 2025

University at Buffalo
The State University of New York

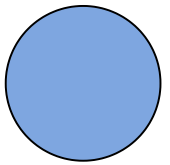
What is MPI?

- The Message-Passing Interface (MPI) is a popular standard in High-performance Computing
- Often implemented in C, C++, or Fortran
- When you need 10,000 nodes to help compute a simulation of a rocket launch you turn to MPI

The problem: MPI implementations vary and libraries are unverified.

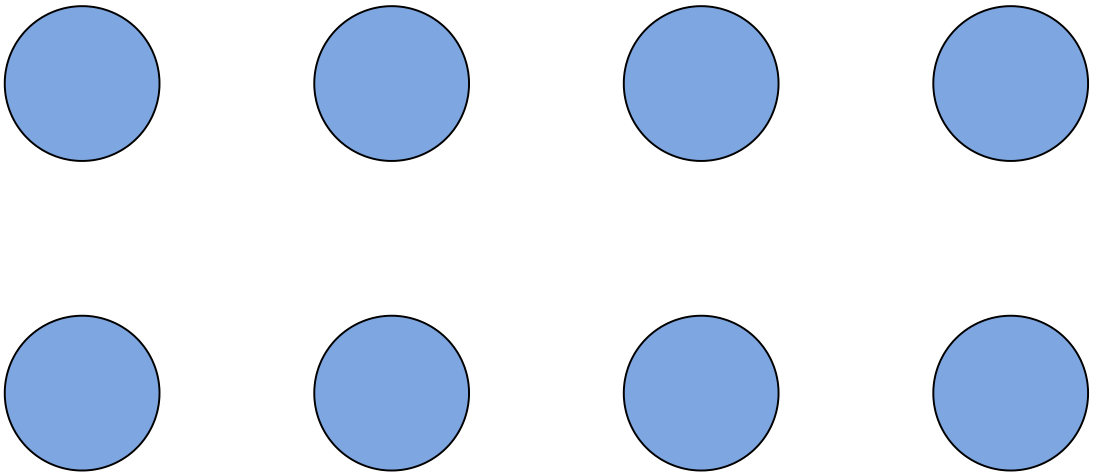
Concurrent Lambda Calculus (CLC)

- Lambda Calculus + concurrency + communication
- Functional language with CCS style concurrency
- Set number of named processes that communicate with each other by name



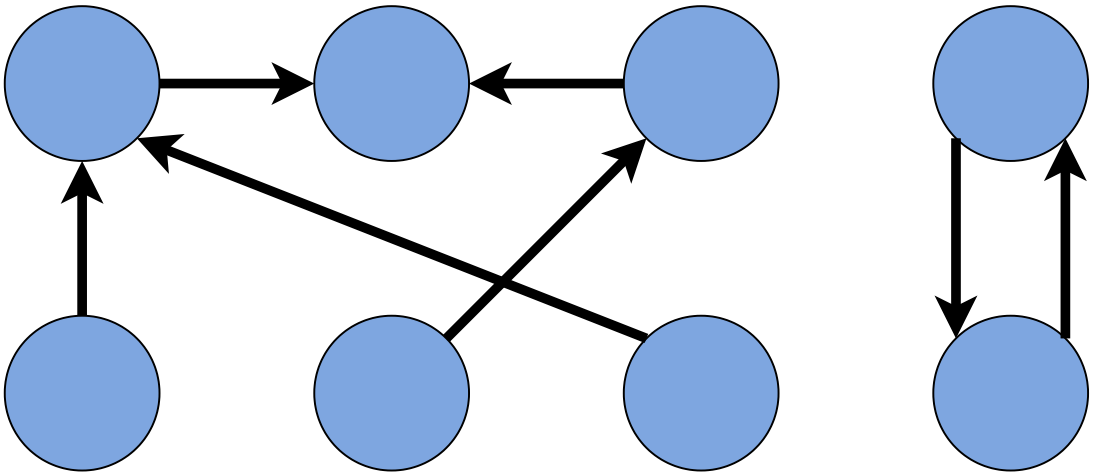
Concurrent Lambda Calculus (CLC)

- Lambda Calculus + concurrency + communication
- Functional language with CCS style concurrency
- Set number of named processes that communicate with each other by name



Concurrent Lambda Calculus (CLC)

- Lambda Calculus + concurrency + communication
- Functional language with CCS style concurrency
- Set number of named processes that communicate with each other by name



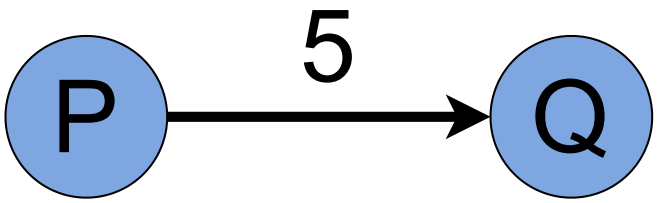
Concurrent Lambda Calculus (CLC)

- Lambda Calculus + concurrency + communication
- Functional language with CCS style concurrency
- Set number of named processes that communicate with each other by name

Let's look at an example of sending and receiving

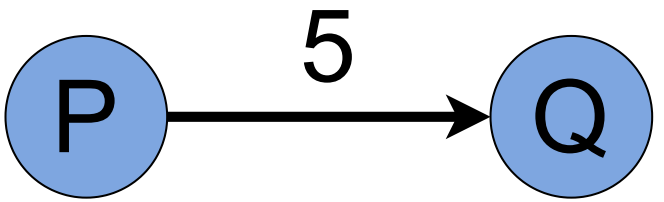
CLC Send and Receive

I, process P, want to send the value 5 to Q



CLC Send and Receive

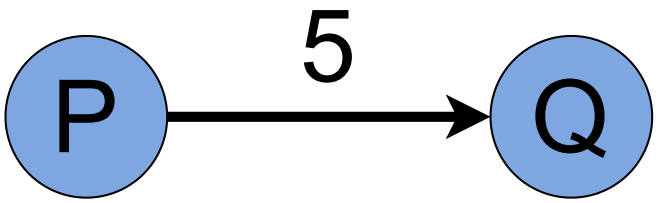
I, process P, want to send the value 5 to Q



Process P	Process Q	Can step

CLC Send and Receive

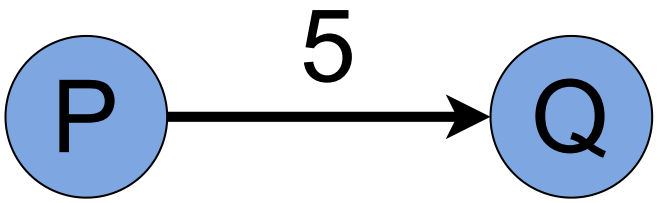
I, process P, want to send the value 5 to Q



Process P	Process Q	Can step
send 5 to q	**recv x from p in x**	✓

CLC Send and Receive

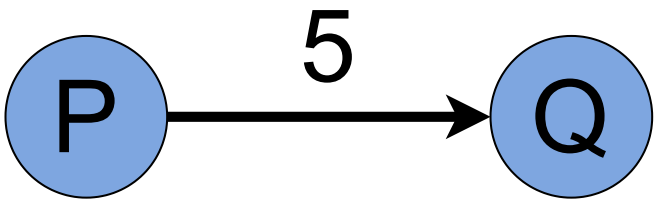
I, process P, want to send the value 5 to Q



Process P	Process Q	Can step
send 5 to q	**recv x from p in x**	✓
send 5 to q	**send 5 to p**	X

CLC Send and Receive

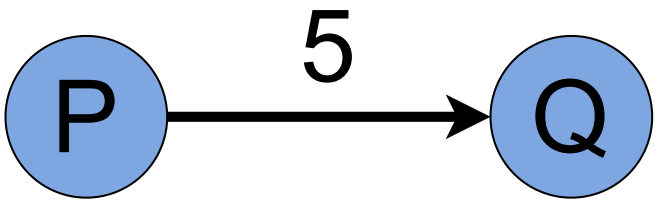
I, process P, want to send the value 5 to Q



Process P	Process Q	Can step
send 5 to q	**recv x from p in x**	✓
send 5 to q	**send 5 to p**	X
send 5 to q	**recv x from r in x**	X

CLC Send and Receive

I, process P, want to send the value 5 to Q



Process P	Process Q	Can step

When sends and receives don't match the **program becomes stuck**

Goal + Agenda

We need a Concurrent Lambda Calculus that captures the key message passing capability of MPI that we can use to verify common algorithms

Goal + Agenda

~~We need~~ **a Concurrent Lambda Calculus that captures the key message passing capability of MPI** ~~that we can use to verify common algorithms~~

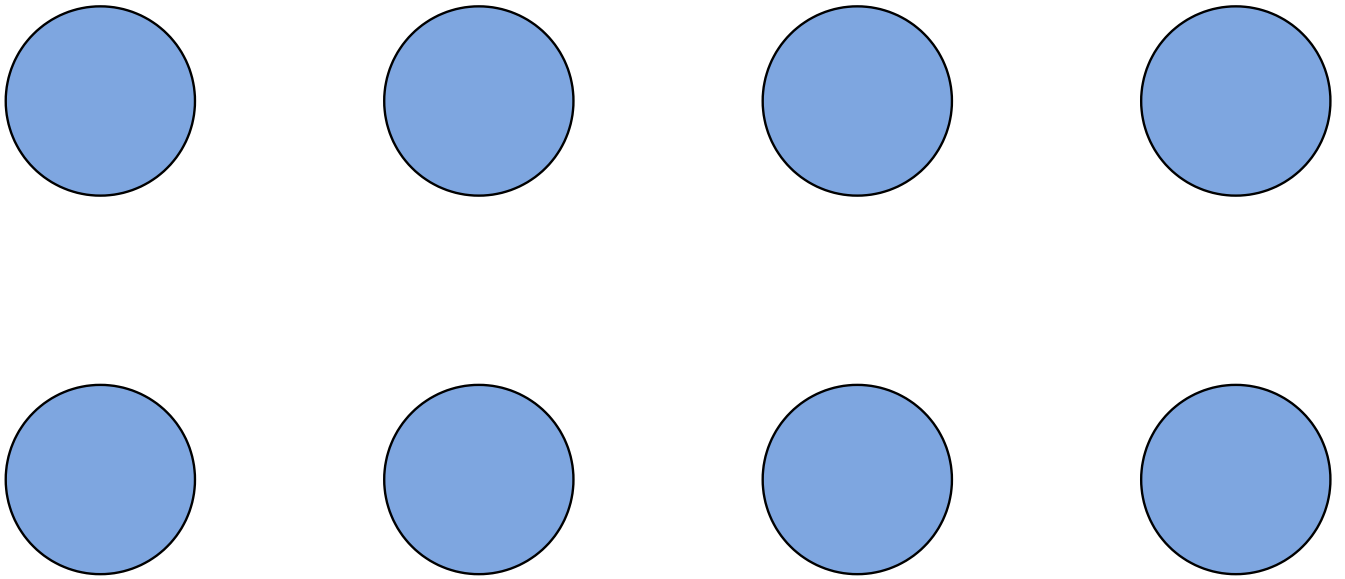
Agenda

- What are the key features of MPI
- Collective communication example

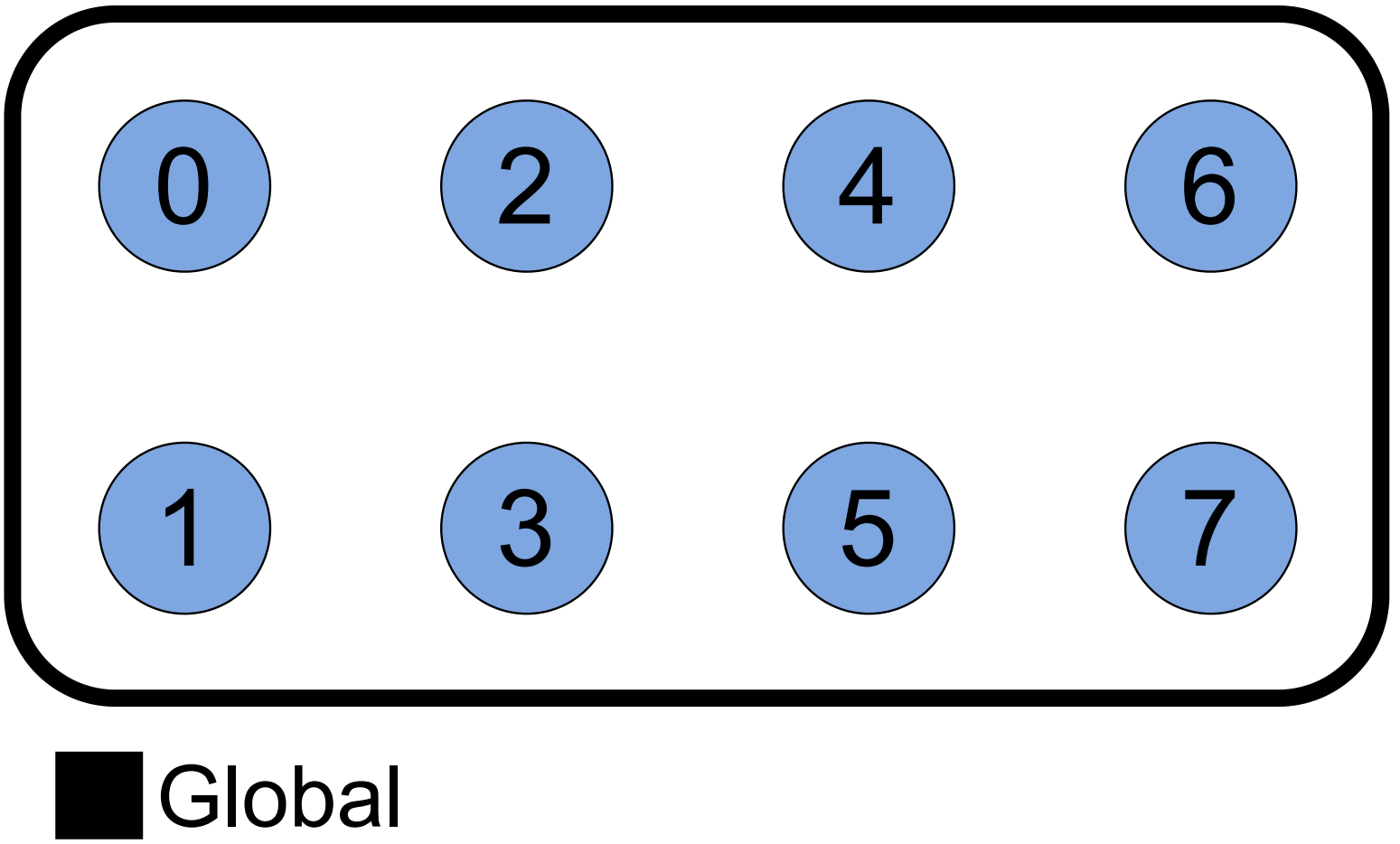
Groups

- Group - Logical collection of processes
- Rank - Each process in a group (size n) has a **rank** 0 to (n-1)
- Root - Each group has **one root**, rank 0, that has a unique role in collective communication

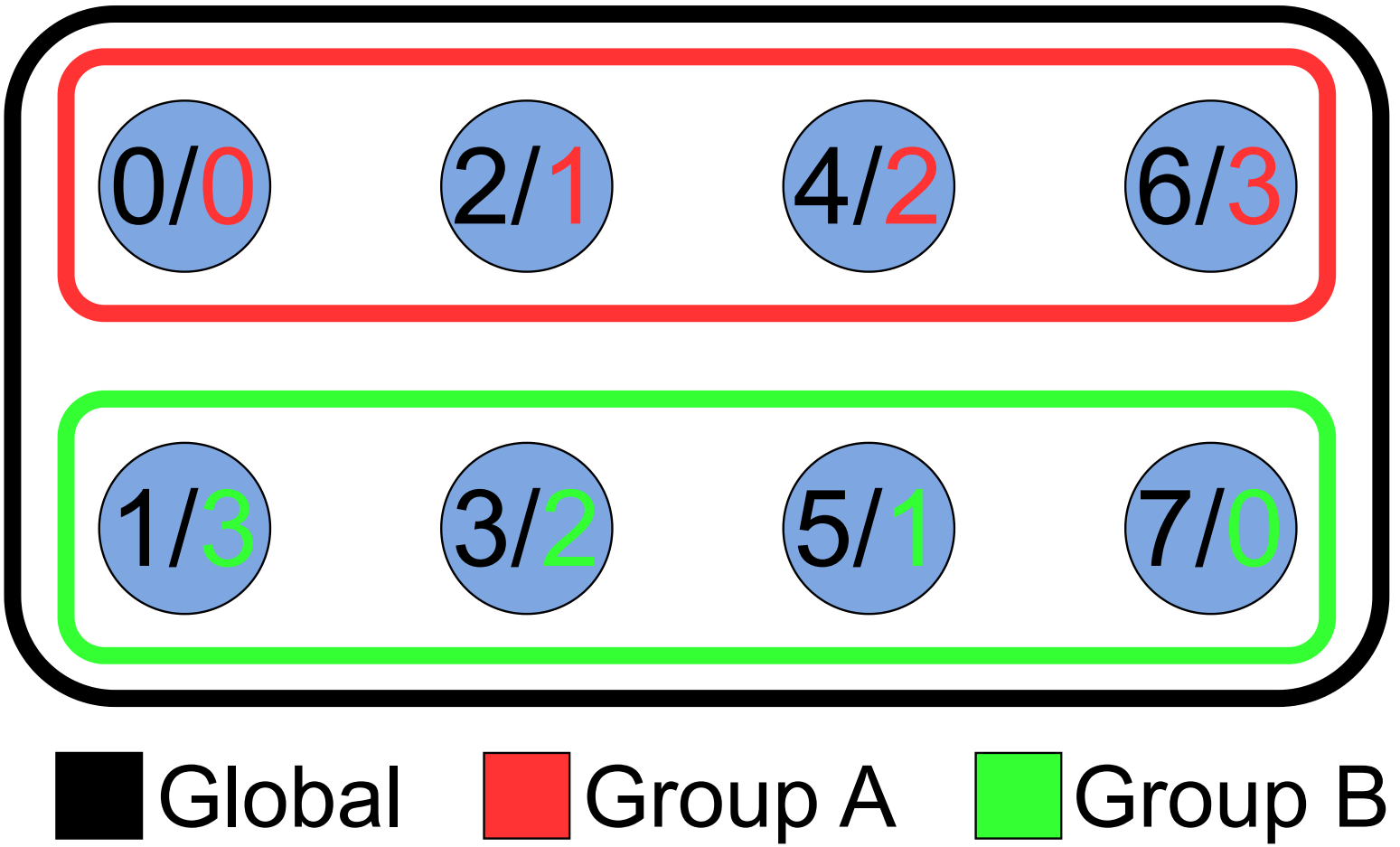
Groups



Groups



Groups



Expression Categories

- Simply Typed Lambda Calculus

Expression Categories

- Simply Typed Lambda Calculus
- Point to point communication

Expression Categories

- Simply Typed Lambda Calculus
- Point to point communication
- Collective communication

Expression Categories

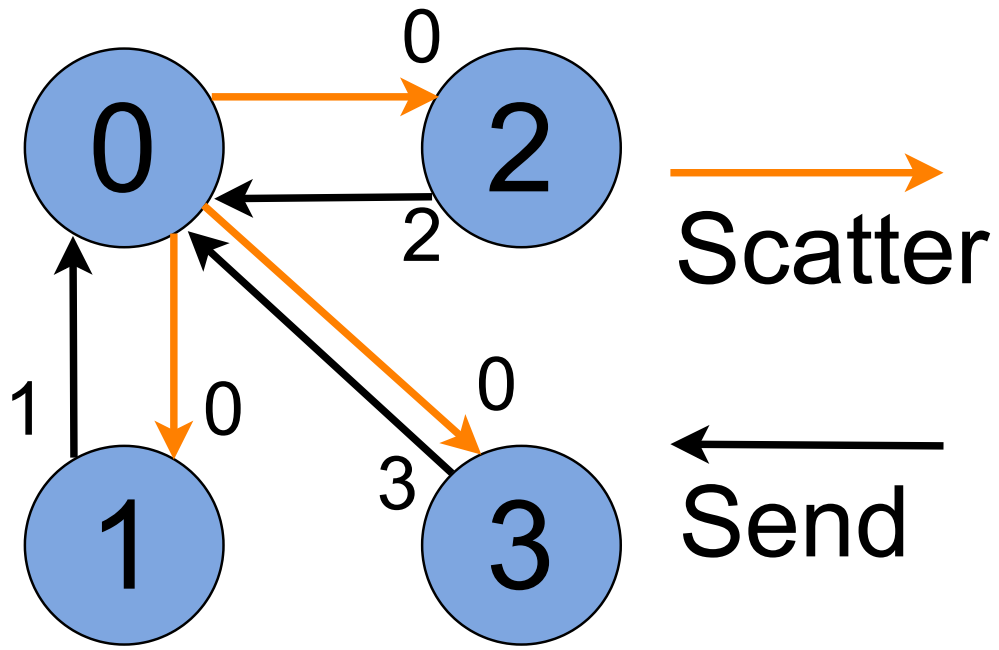
- Simply Typed Lambda Calculus
- Point to point communication
- Collective communication
- Group management

Expression Categories

- Simply Typed Lambda Calculus
- Point to point communication
- Collective communication
- Group management
- Arrays

Collective Communication Example

We will scatter an array of 0's and ask each process to add their rank before returning the new value.



Process 0

Process P_0:

```
[redacted]
  scatter [0;0;0;0] as x
    in (x + (rank global))
    to global)
  [redacted]
```

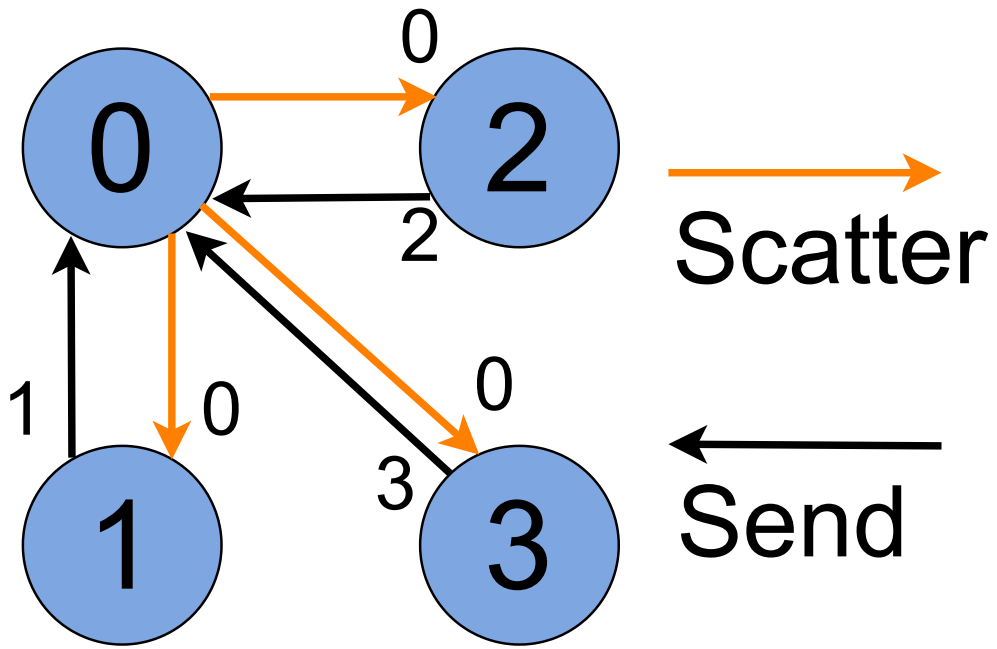
Process 0

Process P_0:

```
gather x and y from (
  scatter [0;0;0;0] as x
    in (x + (rank global))
    to global)
  and global
```

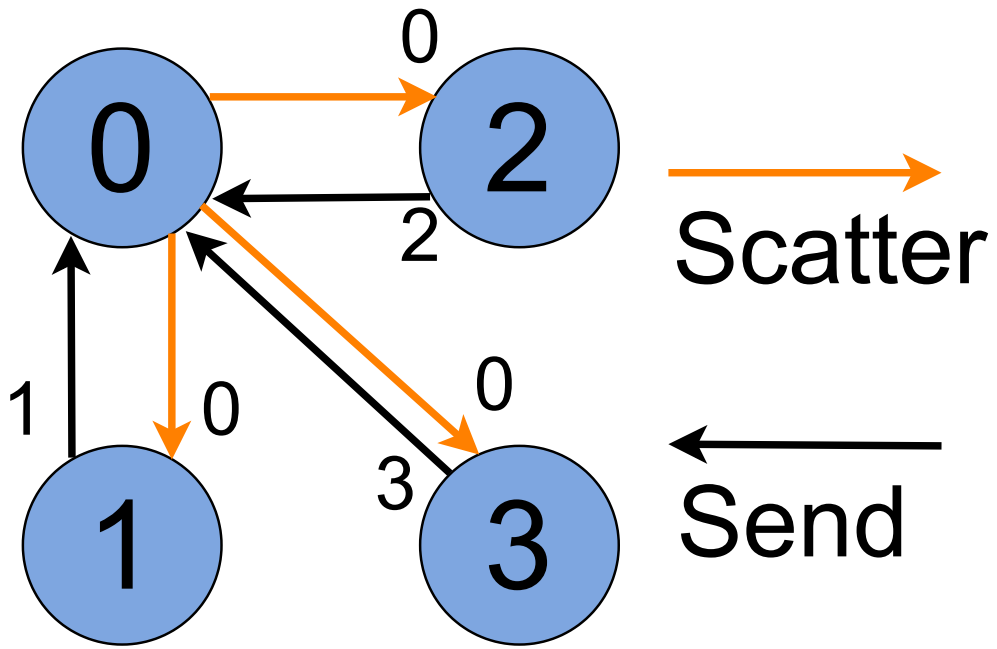
Collective Communication Example

We will scatter an array of 0's and ask each process to add their rank before returning the new value.



Collective Communication Example

I will receive a value from the root, add my rank before returning the result to the root.



Process 1

Process P_1:

```
[redacted]
  (recv x
    in (x + (rank global))
    from 0 of global)
  [redacted]
```

Process 1

Process P_1:

```
send
  (recv x
    in (x + (rank global))
    from 0 of global)
  to 0 of global
```

Next Steps

Adding MPI features

- More ways to create groups
- Allow matching of blocking and non blocking sends
- Envelopes

Writing and verifying common algorithms

A Concurrent Lambda Calculus for MPI

Keith Allen, Dr. Andrew Hirsch

ktallen@buffalo.edu

Upstate PL

August 28, 2025

University at Buffalo
The State University of New York