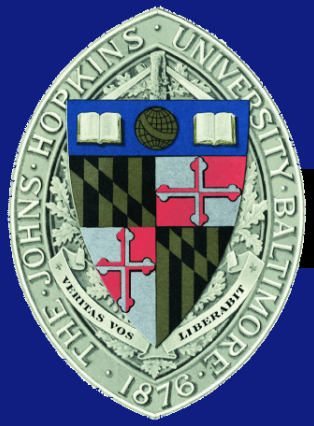
Lecture 6.2
MPI Messaging II

EN 600.320/420

Instructor: Randal Burns

15 February 2017



Department of Computer Science, *Johns Hopkins University*

Tags: Out of Order Delivery

```
MPI_Send (buff, count, datatype,
        dest, tag, comm )
MPI_Recv (buff, count, datatype,
        source, tag, comm )
```

- Tag is an application defined concept used to determine delivery order
  - Specify a tag, get the message you desire, regardless of delivery order
  - There are wildcards to receive all messages (Reg. exps?)
- Communicator specifies a subset of nodes running a parallel application
  - Has a rank and size
  - Default MPI_COMM_WORLD

How about asynchronous I/O?

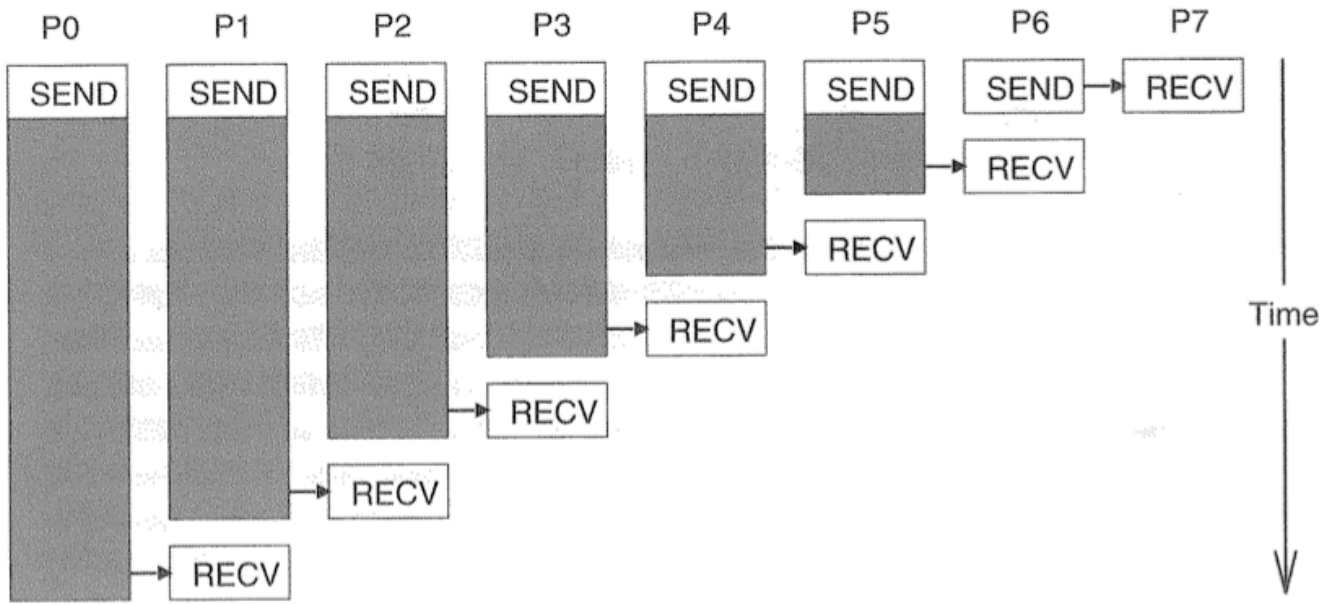
- MPI has support for non-blocking I/O
  - Send/recv request (returns as soon as resources allocated)
  - `MPI_Isend( … )`
  - Do some useful work
  - `MPI_Wait( &request, &status ) //finalize`

- `MPI_Wait:` await the completion of operation
- `MPI_Test:` check the completion of operation and return immediately

- Program must leave buffer intact until completion!
  - Tie up memory in application space
  - Source of errors

(Aside) MPI_SendRecv

- For pairwise exchange, MPI_SendRecv
  - Always non-blocking
  - Useless: doesn't implement pairwise communication
- Don't use it's lazy and inefficient



Asychronous I/O Useful?

- Forces for:
  - Overlap communication with computation
- Forces against:
  - Ties up buffers
  - Complex code
  - Little overlap available for time-step synchronous programs
- Use as a last resort
  - Remember the runtime is trying to do this for you

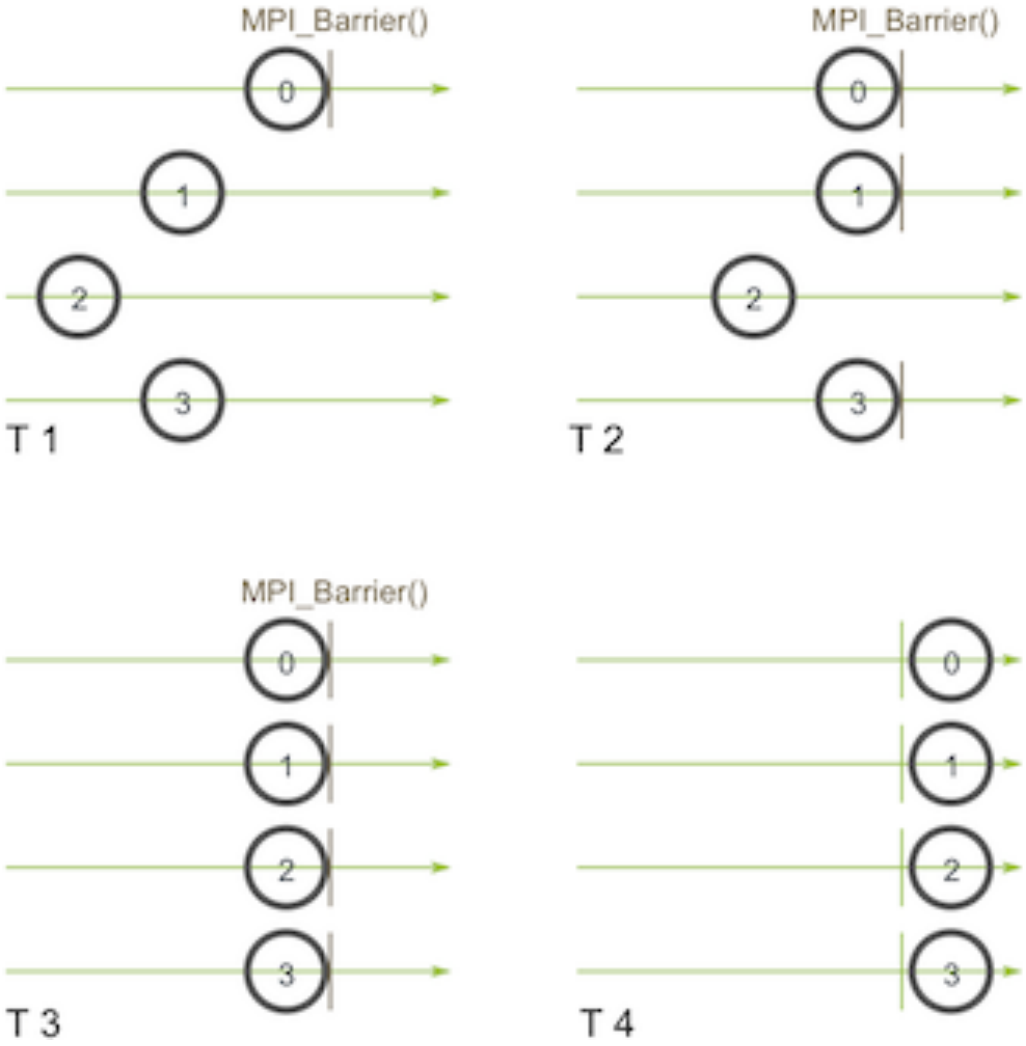
Synchronization

- Implicit synchronization (blocking send/receives)
  - Most common model
  - Allows for fine-grained dependency resolution

- Explicit synchronization (barriers)
  - `MPI_Barrier ( MPI_COMM_WORLD )`
  - All processes must enter barrier before any continue
  - Coarse-grained stops all
  - Common when interacting with shared resources, e.g. parallel file systems or shared-memory (when available)

Barrier Illustrated

http://www.mpitutorial.com/mpi-broadcast-and-collective-communication/



Barriers vs. Send/Receive

- Barriers are useful when awaiting a global condition:
  - Data ready
  - Previous pipeline complete
  - Library call finished
  - Checkpoint written
- But, not a good replacement for pairwise sends and receives
  - They allow nodes to complete whenever their local synchronization constraints are met
  - Barriers are global and create global stalls