Getting Started with MPI Java

- **Website:**
  http://www.hpjava.org/courses/arl/lectures/mpi.ppt
  http://www.hpjava.org/reports/mpiJava-spec/mpiJava-spec.pdf
- **Creating a machines file:**
  ```
  [mfukuda@cssmpi1h mfukuda]$ vi mpd.hosts
  cssmpi2h
  cssmpi3h
  cssmpi4h
  ```
- **Compile a source program:**
  ```
  [mfukuda@cssmpi1h mfukuda]$ javac MyProg.java
  ```
- **Run the executable file:**
  ```
  [mfukuda@cssmpi1h mfukuda]$ mpirun -n 4 java MyProg args
  ```

Program Using MPI

```
import mpi.*;

class MyProg {
  public static void main( String[] args ) {
    MPI.Init( args );                        // Start MPI computation

    int rank = MPI.COMM_WORLD.Rank( );   // Process ID (from 0 to #processes – 1)
    int size = MPI.COMM_WORLD.Size( );   // # participating processes

    System.out.println( "Hello World! I am " + rank + " of " + size );

    MPI.Finalize();                          // Finish MPI computation
  }
}
```

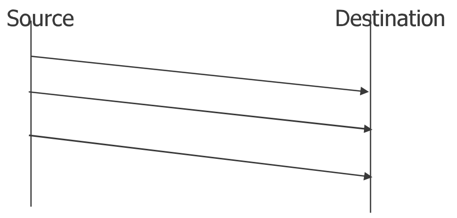
MPI_Send and MPI_Recv

```
void MPI.COMM_WORLD.Send(
                    Object[]        message  /* in */,
                    int             offset   /* in */,
                    int             count    /* in */,
                    MPI.Datatype    datatype /* in */,
                    int             dest     /* in */,
                    int             tag      /* in */)
Status MPI.COMM_WORLD.Recv(
                    Object[]        message  /* in */,
                    int             offset   /* in */,
                    int             count    /* in */,
                    MPI.Datatype    datatype /* in */,
                    int             source   /* in */,
                    int             tag      /* in */)
int Status.Get_count( MPI.Datatype, datatype ) /* #objects received */

MPI.Datatype =    BYTE, CHAR, SHORT, INT, LONG, FLOAT, DOUBLE, OBJECT
```

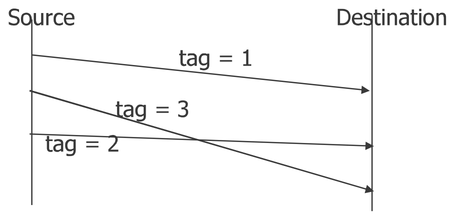
MPI.Send and MPI.Recv

```
import mpi.*;

class myProg {
  public static void main( String[] args ) {
    int tag0 = 0;
    MPI.Init( args );                       // Start MPI computation
    if ( MPI.COMM_WORLD.rank() == 0 ) { // rank 0…sender
      int loop[1]; loop[0] = 3;
      MPI.COMM_WORLD.Send( "Hello World!", 12, MPI.CHAR, 1, tag0 );
      MPI.COMM_WORLD.Send( loop, 1, MPI.INT, 1, tag0 );
    } else {                                // rank 1…receiver
      int loop[1]; char msg[12];
      MPI::COMM_WORLD.Recv( msg, 12, MPI.CHAR, 0, tag0 );
      MPI::COMM_WORLD.Recv( loop, 1, MPI.INT, 0, tag0 );
      for ( int i = 0; i < loop[0]; i++ ) System.out.println( msg );
    }
    MPI.Finalize( );                        // Finish MPI computation
  }
}
```

Message Ordering in MPI



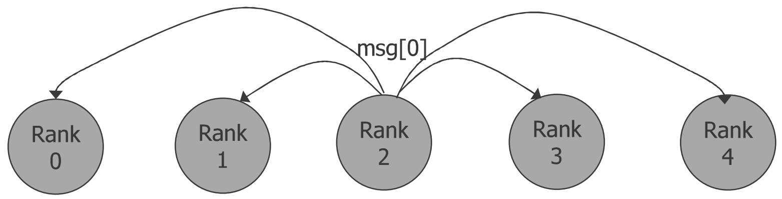
- FIFO Ordering in each data type



- Messages reordered with a tag in each data type

MPI.Bcast

```
void MPI.COMM_WORLD.Bcast(
                          Object[]        message  /* in */,
                          int             offset   /* in */,
                          int             count    /* in */,
                          MPI.Datatype    datatype /* in */,
                          int             root     /* in */)
```

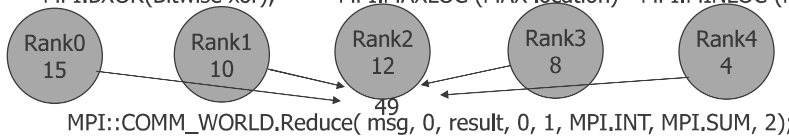


MPI::COMM_WORLD.Bcast(msg, 0, 1, MPI.INT, 2);

MPI_Reduce

```
void MPI.COMM_WORLD.Reduce(
                          Object[]        sendbuf      /* in */,
                          int             sendoffset   /* in */,
                          Object[]        recvbuf      /* out */,
                          int             recvoffset   /* in */,
                          int             count        /* in */,
                          MPI.Datatype    datatype     /* in */,
                          MPI.Op          operator     /* in */,
                          int             root         /* in */ )
```

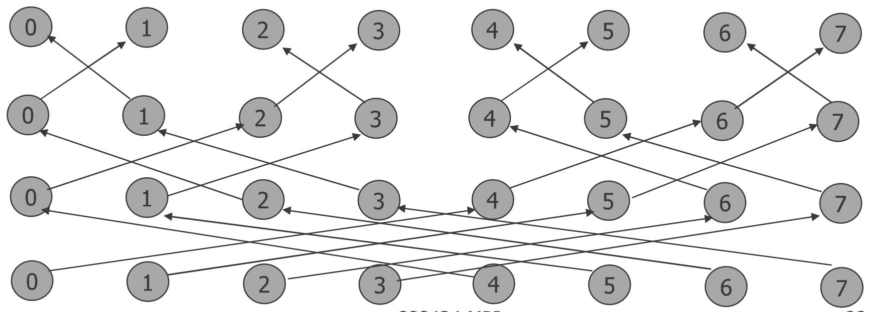
MPI.Op = MPI.MAX (Maximum), MPI.MIN (Minimum), MPI.SUM (Sum),
MPI.PROD (Product), MPI.LAND (Logical and), MPI.BAND (Bitwise and),
MPI.LOR (Logical or), MPI.BOR (Bitwise or), MPI.LXOR (logical xor),
MPI.BXOR(Bitwise xor), MPI.MAXLOC (MAX location) MPI.MINLOC (MIN loc.)



MPI::COMM_WORLD.Reduce(msg, 0, result, 0, 1, MPI.INT, MPI.SUM, 2);

MPI_Allreduce

```
void MPI.COMM_WORLD.Allreduce(
                          Object[]        sendbuf      /* in */,
                          int             sendoffset   /* in */,
                          Object[]        recvbuf      /* out */,
                          int             recvoffset   /* in */,
                          int             count        /* in */,
                          MPI.Datatype    datatype     /* in */,
                          MPI.Op          operator     /* in */)
```


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