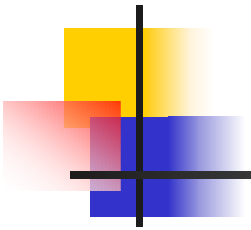
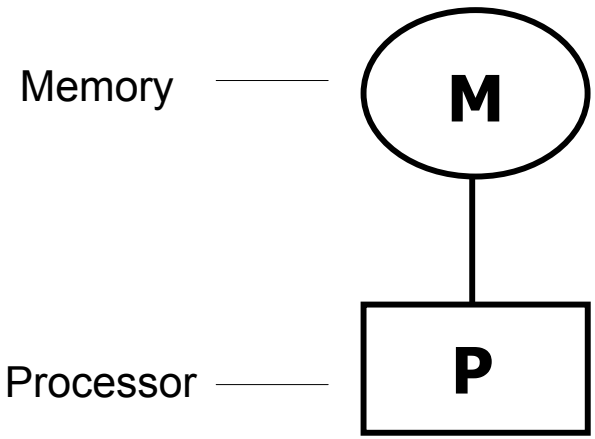
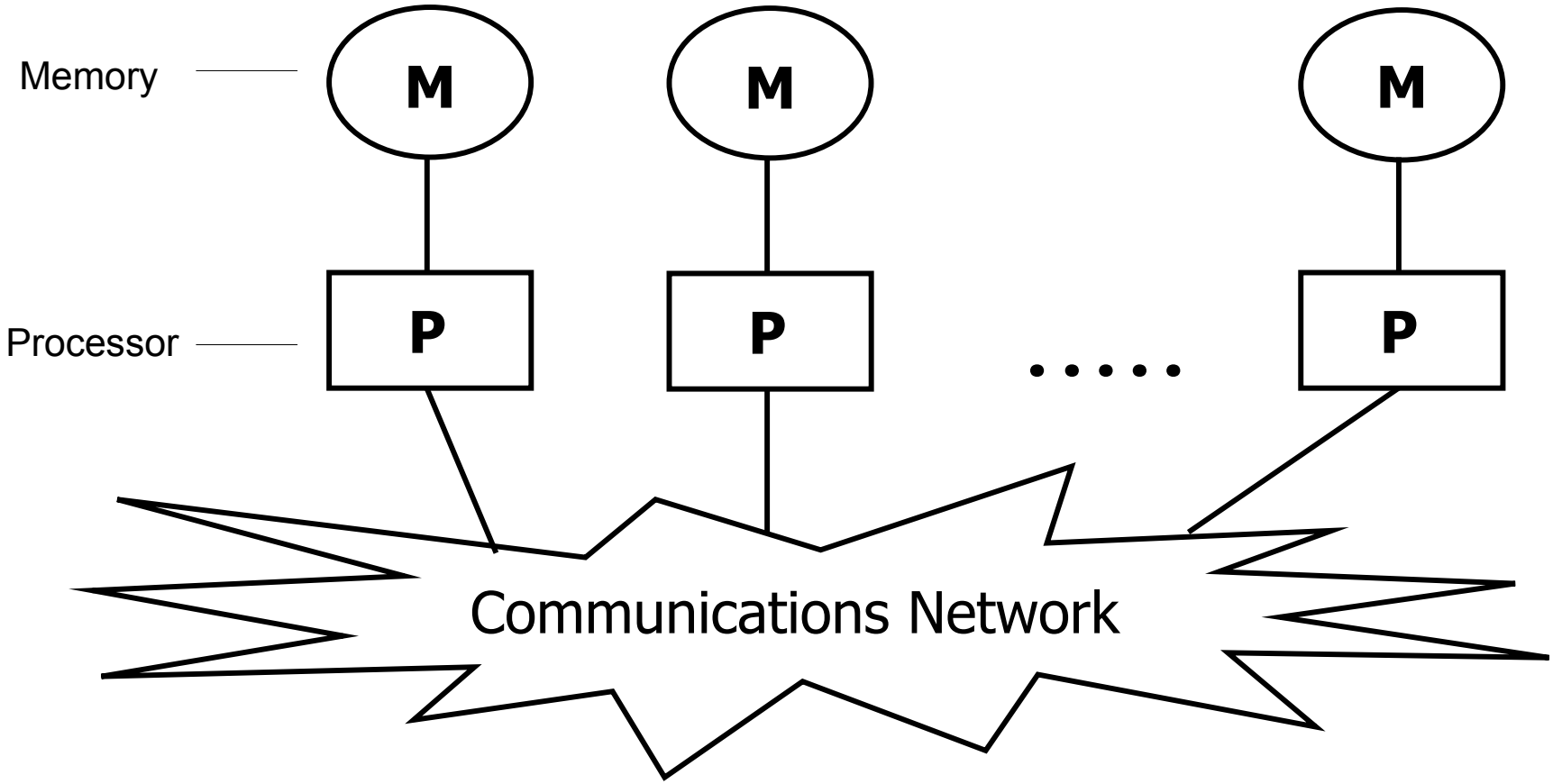


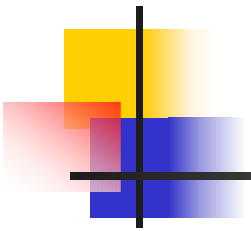
Writing Message-Passing Parallel Programs with MPI

Sequential Programming Paradigm



Message-Passing Programming Paradigm





MPI - Basics

- Header File

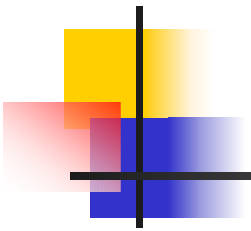
```
#include “mpi.h “
```

- Initializing MPI

```
int MPI_Init(int *argc, char ***argv)
```

- Exiting MPI

```
int MPI_Finalize()
```



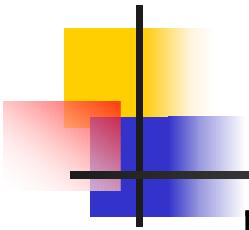
MPI - Functions

- Identifying different processes

```
MPI_Comm_rank(MPI_Comm comm, int *rank)
```

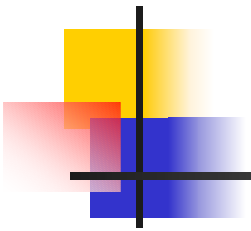
- Counting the processes

```
MPI_Comm_size(MPI_Comm comm, int *size)
```



MPI Basic Datatypes - C

MPI Datatype	C Datatype
MPI_CHAR	signed char
MPI_SHORT	signed short int
MPI_INT	signed int
MPI_LONG	signed long int
MPI_UNSIGNED_CHAR	unsigned char
MPI_ UNSIGNED_SHORT	unsigned short int
MPI_ UNSIGNED	unsigned int
MPI_ UNSIGNED_LONG	unsigned long int
MPI_FLOAT	float
MPI_DOUBLE	double
MPI_LONG_DOUBLE	long double
MPI_BYTE	
MPI_PACKED	



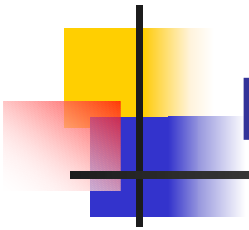
Messages

- Sending a message

```
int MPI_Send(void *mes, int count,
    MPI_Datatype datatype, int dest,
    int tag, MPI_Comm comm)
```

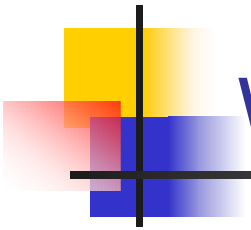
- Receiving a message

```
int MPI_Recv(void *mes, int count,
     MPI_Datatype datatype, int source,
     int tag, MPI_Comm comm,
     MPI_Status status)
```



For a communication to succeed:

- ✓ Sender must specify a valid destination rank.
- ✓ Receiver must specify a valid source rank.
- ✓ The communicator must be the same.
- ✓ Tags must match.
- ✓ Message types must match.
- ✓ Receiver’s buffer must be large enough.



Wildcarding

- Receiver can wildcard.
- To receive from any source `MPI_ANY_SOURCE`
- To receive with any tag `MPI_ANY_TAG`
- Actual source and tag are returned in the receiver's status parameter:

```
status.MPI_SOURCE
status.MPI_TAG
```