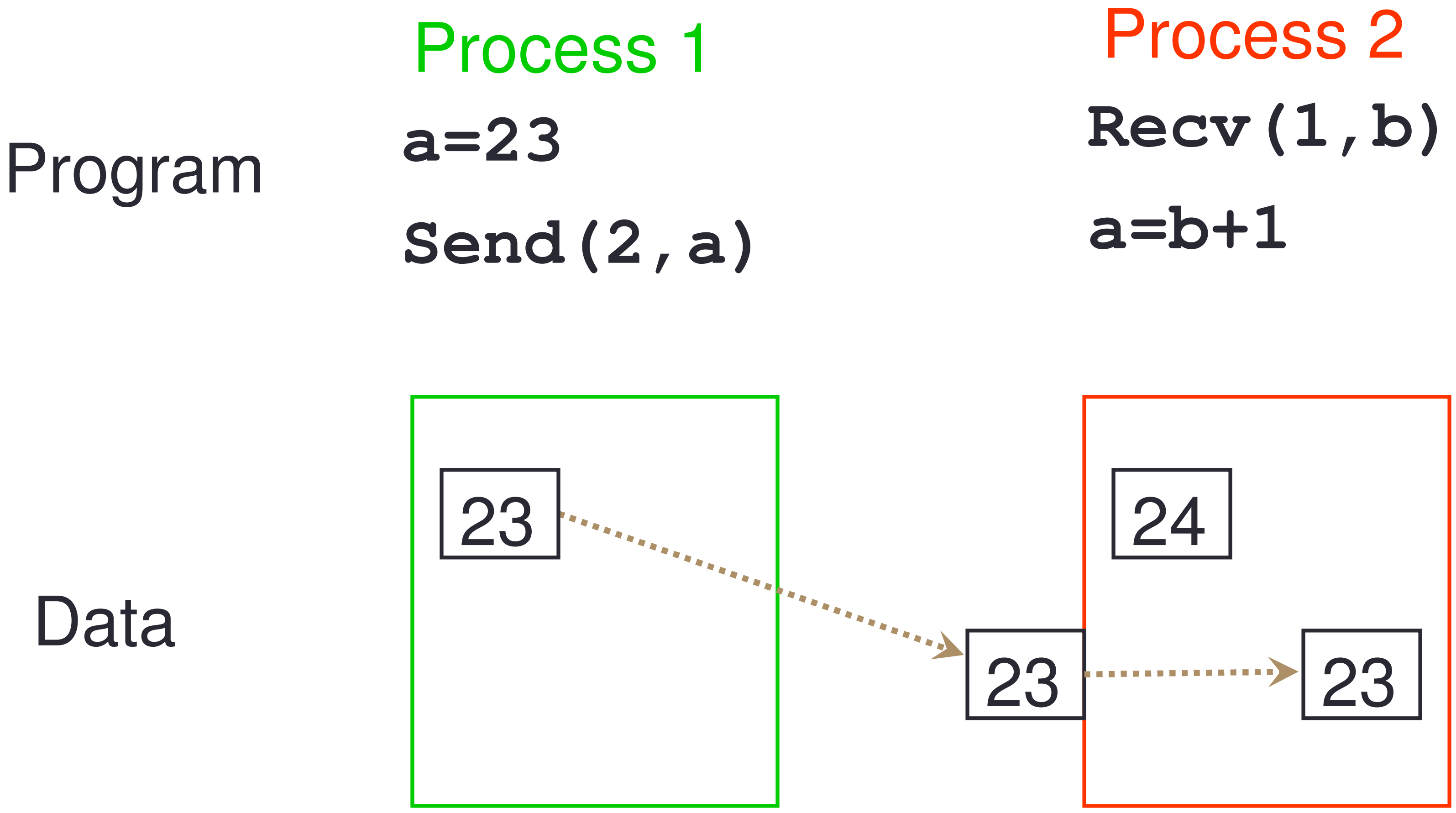
Message Passing Concepts

Message Passing Model

- The message passing model is based on the notion of processes
  - can think of a process as an instance of a running program, together with the program's data
- In the message passing model, parallelism is achieved by having many processes co-operate on the same task
- Each process has access only to its own data
- Processes communicate with each other by sending and receiving messages

Process Communication



SPMD

- Most message passing programs use the Single-Program-Multiple-Data (SPMD) model
- All processes run the same program
- Each process has a separate copy of the data
- To make this useful, each process has a unique identifier
- Processes can follow different control paths through the program, depending on their process ID
- Usually run one process per processor

Messages

- A message transfers a number of data items of a certain type from the memory of one process to the memory of another process

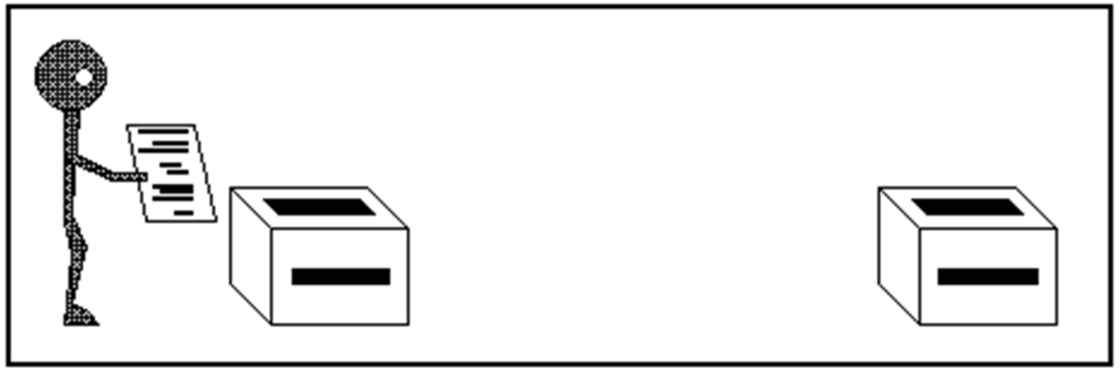
- A message typically contains
  - the ID of the sending processor
  - the ID of the receiving processor
  - the type of the data items
  - the number of data items
  - the data itself
  - a message type identifier

Communication modes

- Sending a message can either be synchronous or asynchronous
- A synchronous send is not completed until the message has started to be received
- An asynchronous send completes as soon as the message has gone
- Receives are usually synchronous - the receiving process must wait until the message arrives

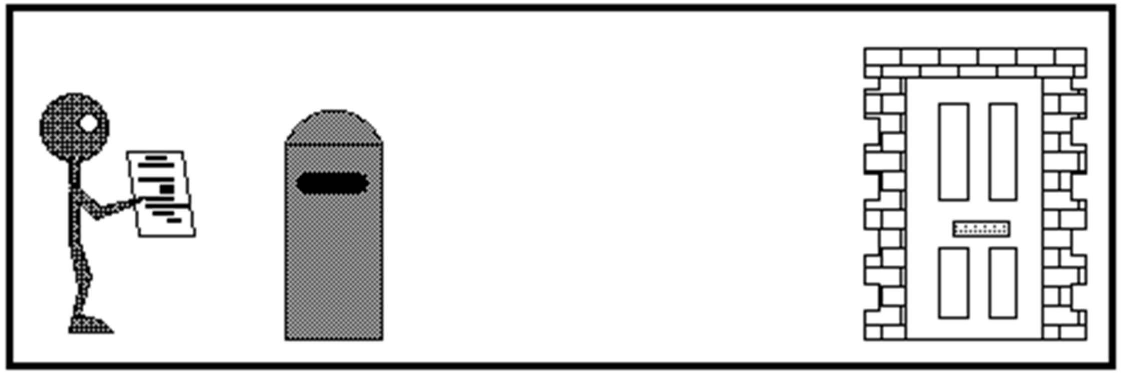
Synchronous send

- Analogy with faxing a letter.
- Know when letter has started to be received.



Asynchronous send

- Analogy with posting a letter.
- Only know when letter has been posted, not when it has been received.



Point-to-Point Communications

- We have considered two processes
  - one sender
  - one receiver

- This is called point-to-point communication
  - simplest form of message passing
  - relies on matching send and receive

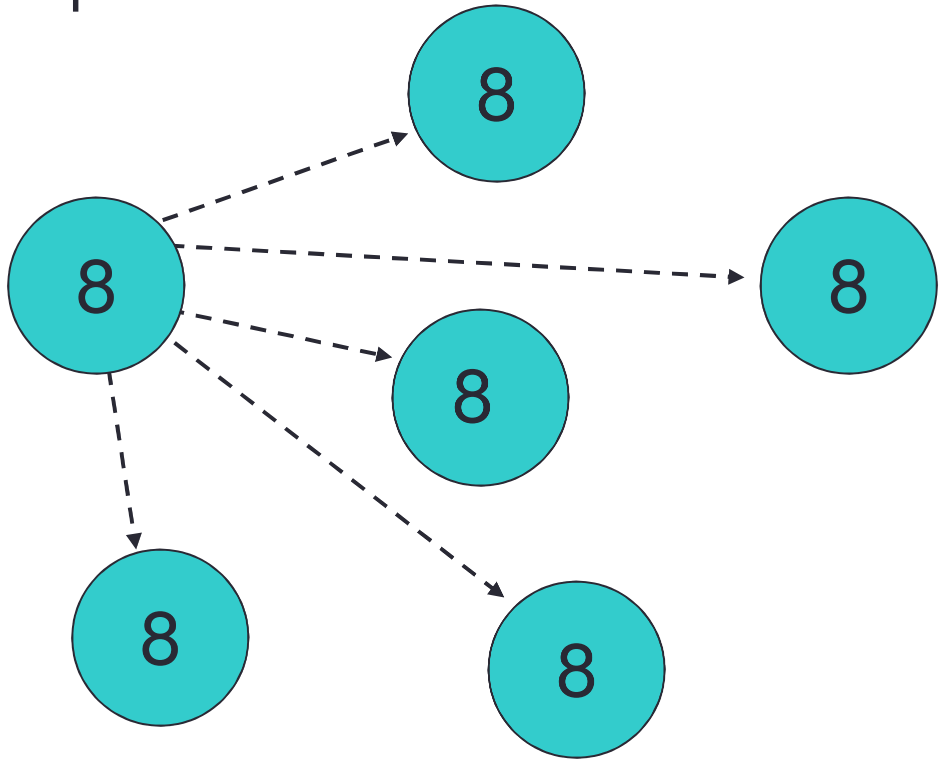
- Close analogy to sending personal emails

Collective Communications

- A simple message communicates between two processes
- There are many instances where communication between groups of processes is required
- Can be built from simple messages, but often implemented separately, for efficiency

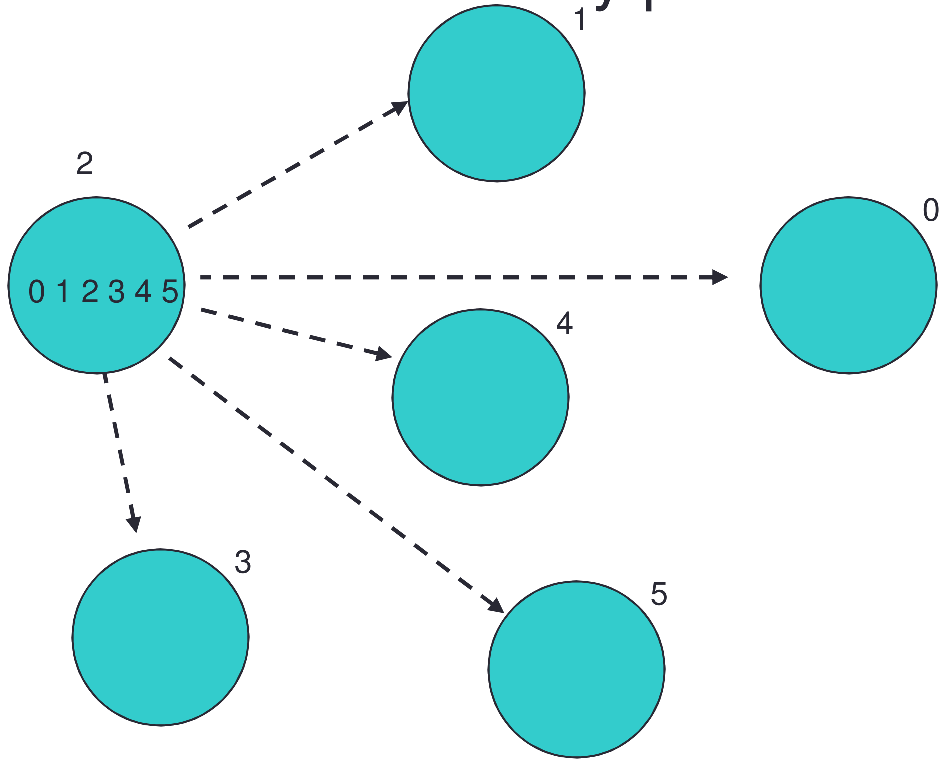
Broadcast

- From one process to all others



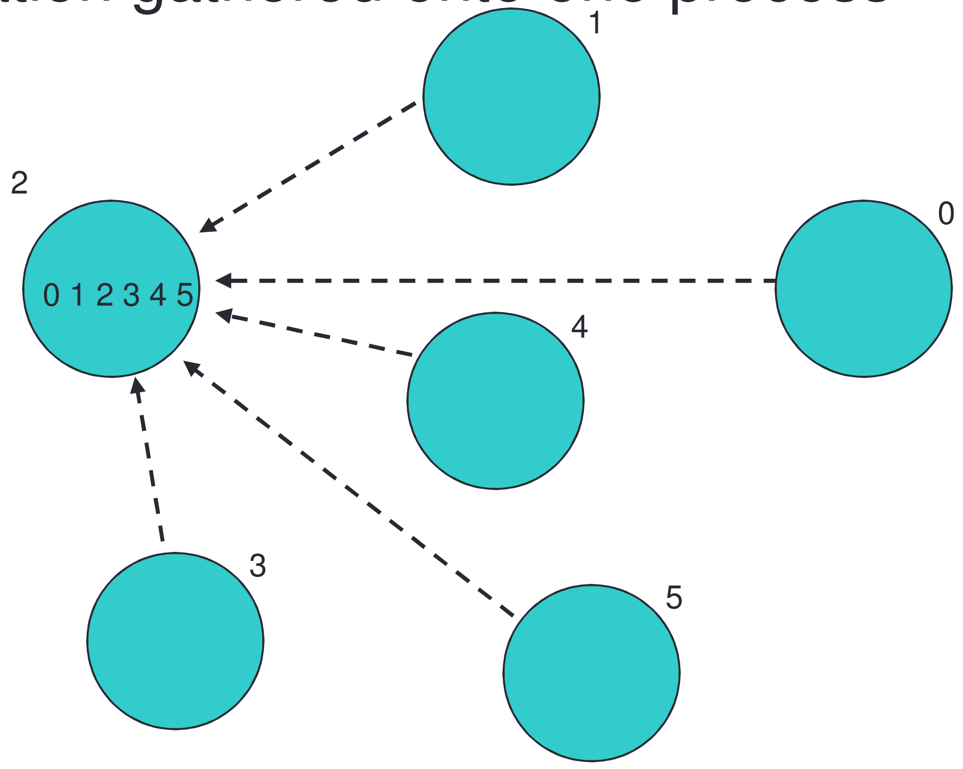
Scatter

- Information scattered to many processes



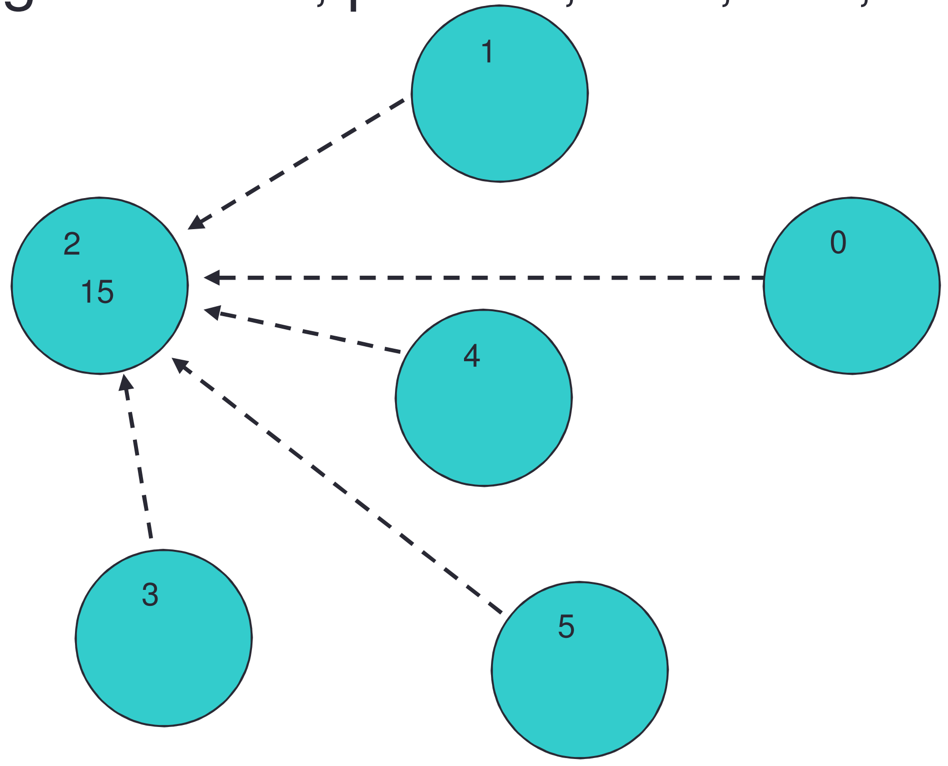
Gather

- Information gathered onto one process



Reduction

- Form a global sum, product, max, min, etc.



Issues

- Sends and receives must match
  - danger of deadlock
- Possible to write very complicated programs
  - most scientific codes have a simple structure
  - often results in simple communications patterns
- Use collective communications where possible
  - may be implemented in efficient ways

Summary

- Messages are the *only* form of communication
  - all communication is therefore explicit
- Most systems use the SPMD model
  - all processes run exactly the same code
  - each has a unique ID
  - processes can take different branches in the same codes
- Basic form is point-to-point
  - collective communications implement more complicated patterns that often occur in many codes