UNIVERSITY of WISCONSIN-MADISON
Computer Sciences Department

CS 537 A. Arpaci-Dusseau
Intro to Operating Systems Spring 2000

Message Systems: Interprocess Communication (IPC)

Questions answered in these notes:

- Why use message passing?
- How does a Remote Procedure Call (RPC) work?
- What are the possibilities for naming?
- What are the relationships between synchronization and buffering?

Reading

- Return back to Chapter 4, Section 6

Motivation

Previous: Assumed processes communicate with shared data

- Use synchronization primitives to ensure no conflicts

Advantages of Messages

- Remove shared data to reduce errors
  - Be explicit when sharing
- Improves modularity
  - Processes can be written by different programmers
- Might have processes that do not trust each other
  - OS vs. user
- Environment where no shared memory exists
  - Distributed memory systems with MPI and PVM

Old Problem: How to keep processes from interfering w/ each other

New Problem: How to get proceesses to cooperate

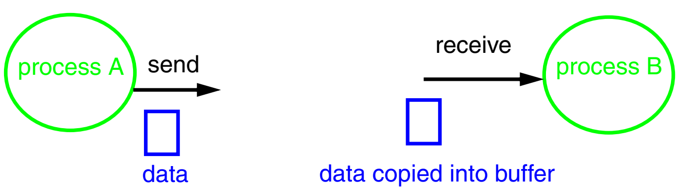
Overview of Message Passing

Message: Piece of information passed between processes

Mailbox: Place where messages are stored between time they are sent and time are received

Operations

- Send(destination, data)
- Receive(source, buffer)



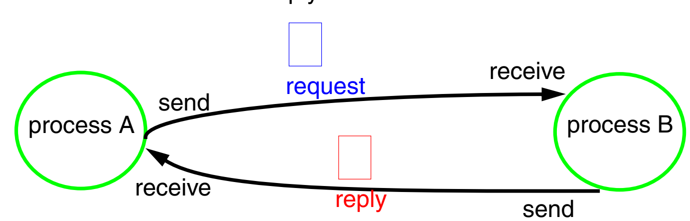
Directionaltiy

One-way: One process always sends, other always receives

- Example: ??

Two-way: Receiver sends back a reply

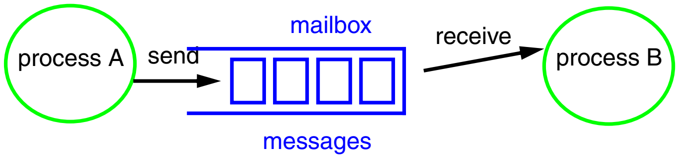
- Example: Remote procedure calls in client/server applications
- Send name of procedure and parameters in request
- Server unpackages message to get command; processes
- Server returns results in reply



Naming

How are destination and source specified?

- Specify process directly vs. indirectly with mailbox (or port)



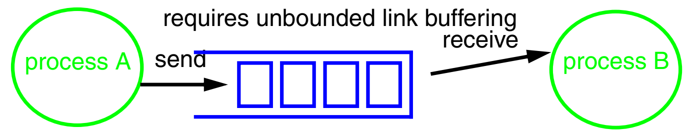
- Specify one destination vs. set of destinations (multicast to all)
- Specify one source vs. any in set of sources
- Specify additional tags to restrict matches

Synchronization

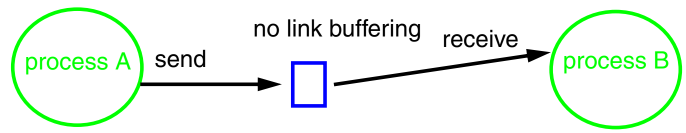
Does process wait for operation to complete?

- Send() blocks the sender vs. sender continues
- Receiver() always blocks the receiver

Common: Send is non-blocking, receive is blocking



Rendezvous: Sender and receiver wait for other to reach statement



Other Issues

How are message boundaries specified?

- Messages are fixed sized vs. variable sized
- Enforced message boundaries vs. flexible
  - Example: UNIX sends stream of bytes

Reliability

- Guarantee messsage not garbled
  - Use Error Checking Codes
- Guarantee message not lost
  - Resend if not Acknowledged (ACK) before timeout