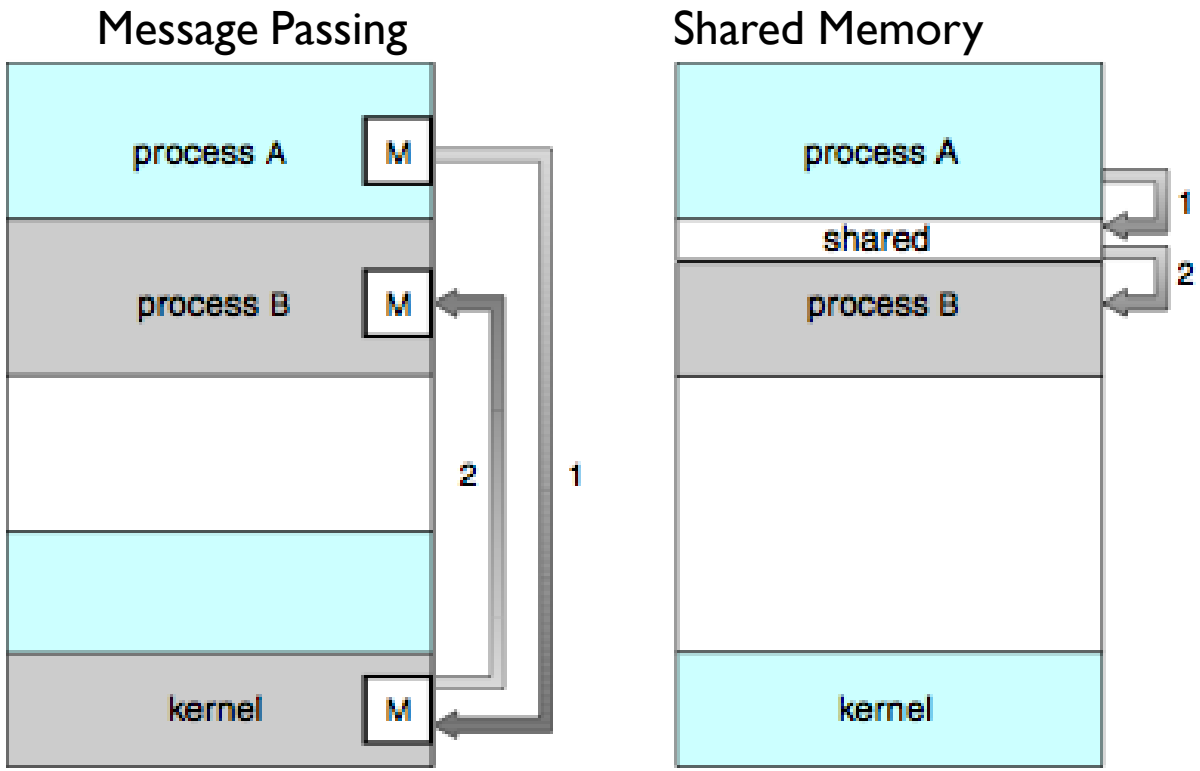
Processes

Back to Process Management

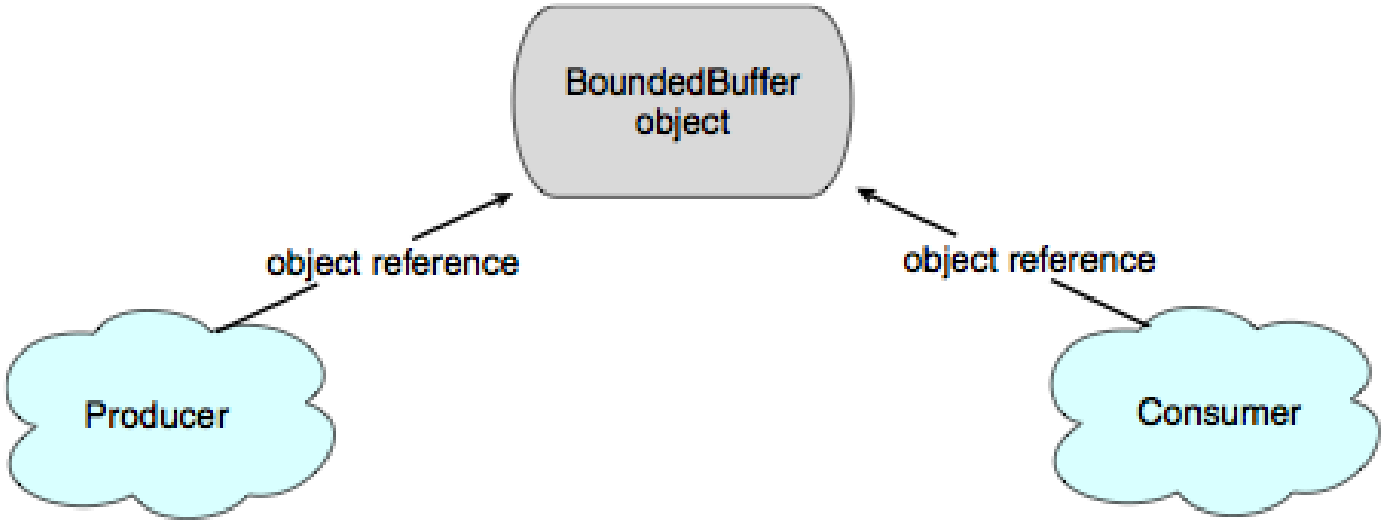
Interprocess Communication (1)

- Independent versus cooperating processes
  - Information sharing, Computation speedup, Modularity, Convenience



Interprocess Communication (2)

- Shared-memory systems
  - Producer – consumer problem
    - Unbounded buffer
    - Bounded buffer



Interprocess Communication (3)

```
public interface Buffer <E> {
        // producers call this method
        public void insert(E item);

        // consumers call this method
        public E remove();
}
```

```
public class BufferImpl<E> implements Buffer<E>  {
        private static final int BUFFER_SIZE = 5;
        private E[] elements;
        private int in, out, count;

        public BufferImpl() {
                count = 0; in = 0; out = 0;
                elements = (E[]) new Object[BUFFER_SIZE];
        }

        // producers call this method
        public void insert(E item) {
        }

        // consumers call this method
        public E remove() {

        }
}
```

Interprocess Communication (4)

```
public void insert(E item) {
        while (count == BUFFER_SIZE)
                ; // do nothing -- no free space

        // add an element to the buffer
        elements[in] = item;
        in = (in + 1) % BUFFER_SIZE;
        ++count;
}


  public E remove() {
        E item;

        while (count == 0)
                ; // do nothing - nothing to consume

        // remove an item from the buffer
        item = elements[out];
        out = (out + 1) % BUFFER_SIZE;
        --count;

        return item;
  }
```

Interprocess Communication (5)

- Message passing
  - No shared memory
  - send(message)
  - receive(message)
  - Fixed or variable size messages
    - Complexity: system level implementation or programming task
  - Establish communication link
    - Various physical implementations (shared memory, hardware bus, network)
    - Logical implementation
      - Direct or indirect communication
      - Synchronous or asynchronous communication
      - Automatic or explicit buffering

Interprocess Communication (6)

- Naming
  - Direct communication - - hard coding of names
    - send(P, message)
    - receive(Q, message)
    - Link properties
      - Between a pair of processes that need to know each others' identity
      - Between exactly two processes
      - A single link between each pair of processes
    - Symmetry versus asymmetry in addressing
      - receive(id, message)

Interprocess Communication (7)

- Indirect communication
  - Mailboxes or ports
  - send (A, message)
  - receive(A, message)
  - Link properties
    - Between a pair of processes sharing a mailbox
    - Link maybe shared by more than two processes
    - A pair of processes may share any number of mailboxes
  - Who receives a message?
    - Associate a link to only a pair of processes
    - Allow at most one receive
    - Allow arbitrarily or algorithmically which process receives the message, possibly identifying the recipient to the sender
  - Mailbox owner: system or process
    - Create, Send and Receive messages, Delete, Change owner

Interprocess Communication (8)

- Synchronisation
  - Blocking or non-blocking send and receive
    - Synchronous or asynchronous
  - Rendezvous: blocking send and receive
    - Trivial solution to consumer/producer problem
- Buffering
  - Zero capacity – no messages waiting
  - Bounded capacity – at most n messages waiting
  - Unbounded capacity – any number of messages waiting
  - Latter two automatic buffering

Interprocess Communication (9)

```
public interface Channel<E> {
        public void send(E item);
        public E receive();
}
```

```
public class MessageQueue<E> implements Channel<E> {
        private Vector<E> queue;

        public MessageQueue() {
                queue = new Vector<E>();
        }

        public void send(E item) {
                queue.addElement(item);
        }

        public E receive() {
                if (queue.size() == 0) return null;
                else return queue.remove(0);
        }
}
```

Producer

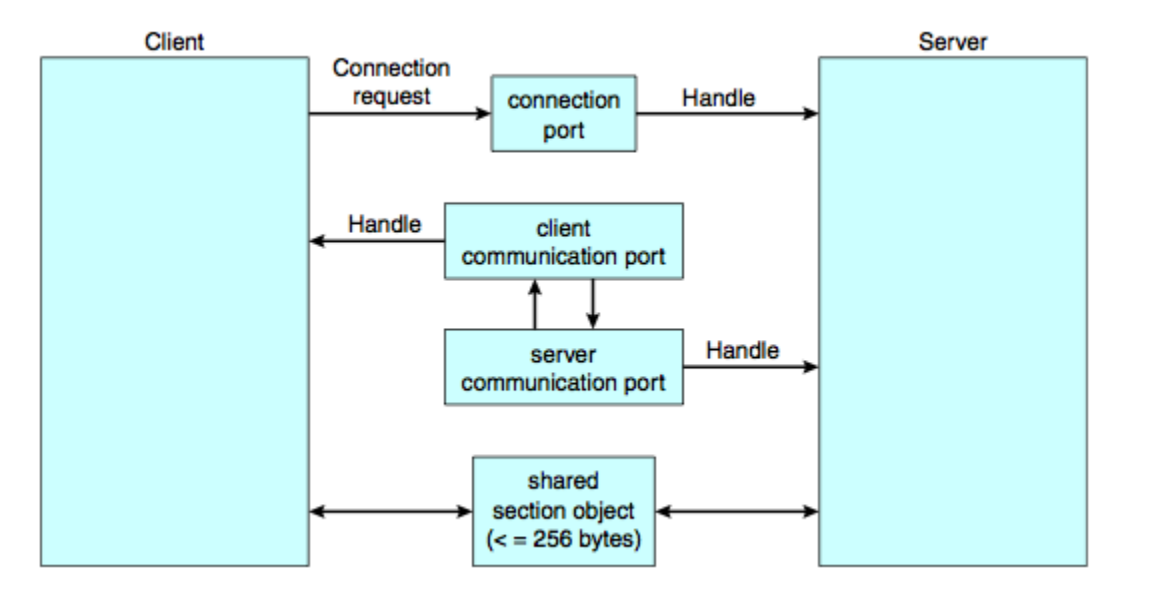
```
Channel<Date> mailBox =
        new MessageQueue<Date>();
        mailBox.send(new Date());
```

Consumer

```
Date rightNow =  mailBox.receive();
        System.out.println(rightNow);
```

INTERPROCESS COMMUNICATION (9)

Local procedure call subsystem

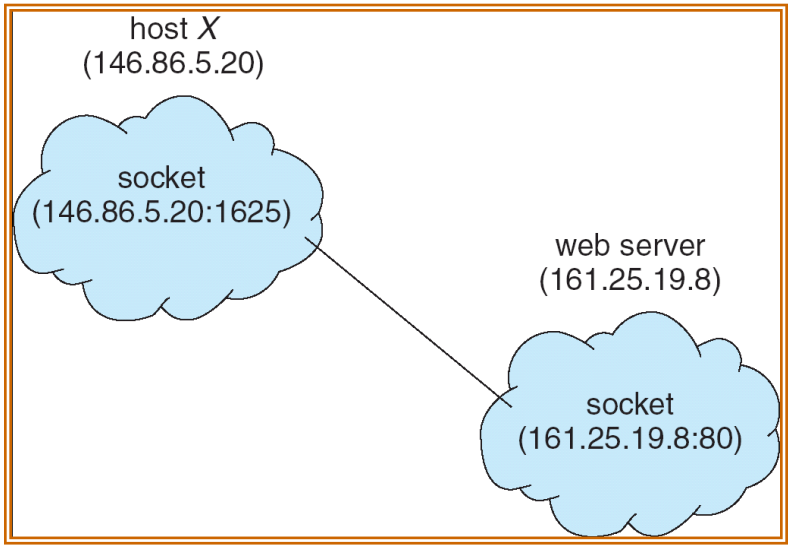


Client/Server Communication

- Sockets
- Remote procedure calls (RPC)
- Remote method invocation (RMI)

Sockets (1)

- A socket is defined as an endpoint for communication
  - Concatenation of IP address and port (>1024), e.g. 161.25.19.8:1625
- Communication between a pair of sockets
- Client/Server architecture
  - All connections must be unique



Sockets (2)

- `java.net`
- Connection-oriented (TCP) sockets – `Socket`
- Connectionless (UDP) sockets – `DatagramSocket`
- `MulticastSocket`
  - subclass of `DatagramSocket`

```
public class DateServer
{
  public static void main(String[] args) {
    try {
      ServerSocket sock = new ServerSocket(6013);

      // now listen for connections
      while (true) {
        Socket client = sock.accept();

        PrintWriter pout = new
         PrintWriter(client.getOutputStream(), true);

        // write the Date to the socket
        pout.println(new java.util.Date().toString());

        // close the socket and resume
        // listening for connections
        client.close();
      }
    }
    catch (IOException ioe) {
      System.err.println(ioe);
    }
  }
}
```

Sockets (3)

```
public class DateClient
{
   public static void main(String[] args) {
      try {
         //make connection to server socket
         Socket sock = new Socket("127.0.0.1",6013);

         InputStream in = sock.getInputStream();
         BufferedReader bin = new
            BufferedReader(new InputStreamReader(in));

         // read the date from the socket
         String line;
         while ( (line = bin.readLine()) != null)
            System.out.println(line);

         // close the socket connection
         sock.close();
      }
      catch (IOException ioe) {
         System.err.println(ioe);
      }
   }
}
```

Loopback

For contemplation

- What are the benefits and detriments of each of the following? Consider both the systems and the programmers' levels.
  - Symmetric and asymmetric communication
  - Automatic and explicit buffering
  - Fixed-sized and variable-sized messages