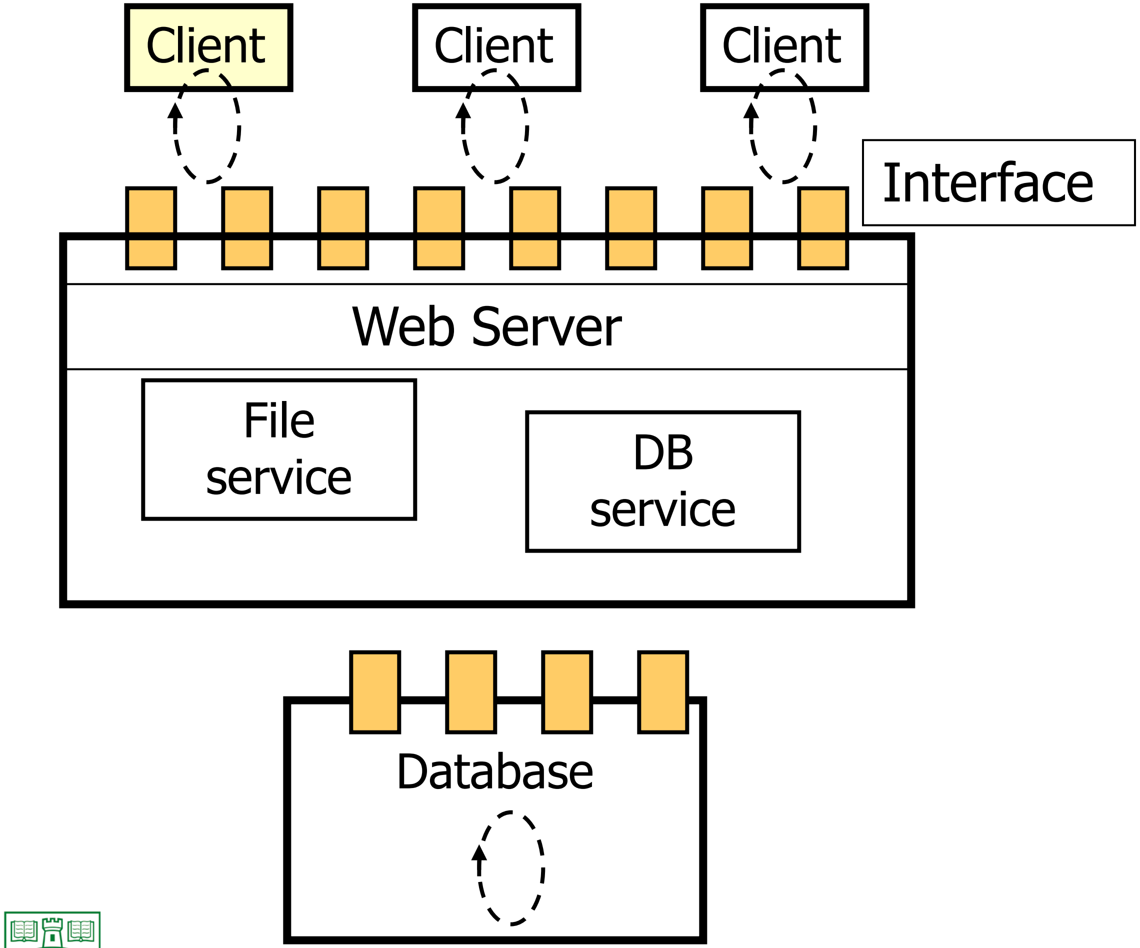
UNIVERSITY of STIRLING

www.cs.stir.ac.uk

Concurrent and Distributed Systems

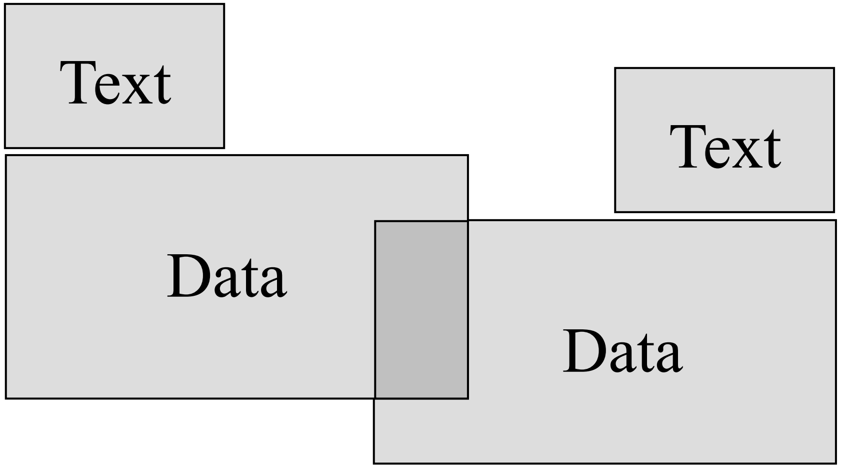
Running in Parallel - Concurrency

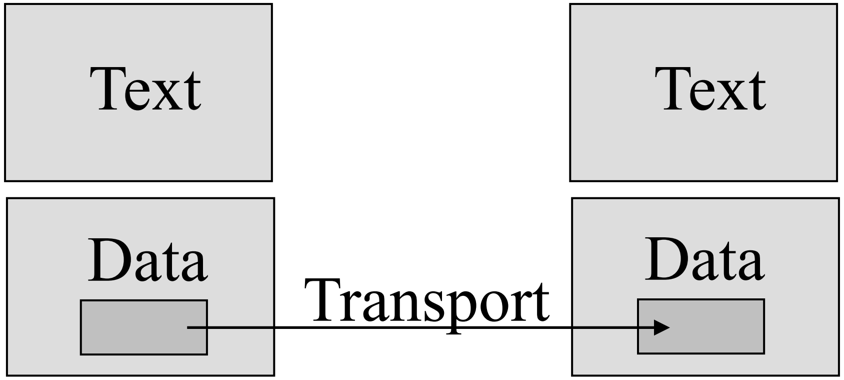
An Example


Client
Client
Client
Interface
Web Server
File service
DB service
Database

Forms of Process Interactions

- Co-operation (shared memory)
- Communication (message passing)





Implementing IPC

- Shared Memory
  - Processes/threads involved share a common buffer pool
  - Buffer can be explicitly implemented by programmer
- Inter-Process Communication without shared memory
  - IPC has at least two operations
    - Send (message)
    - Receive(message)
  - Messages can be either fixed or variable size
  - A link between the involved processes must exist

Synchronisation: send(), receive()

- Calls to send() and receive() may be blocking or non-blocking (synchronous and asynchronous)
- Blocking send
  - Sending process is blocked until the message has been received by the receiving process or mailbox
- Non-blocking send
  - Sending process resumes operation immediately after sending the message

- Blocking receive
  - The receiving process blocks until a message has been received
- Non-blocking receive
  - The receiver retrieves a valid message or a NULL message

Queue Capacity

- Messages exchanged always reside in a temporary queue
- Zero capacity
  - Maximum length 0 → no messages can ‘wait’ in the queue
  - Sender must block until the receiver gets the message
  - Also called a message passing system without buffering
- Bounded capacity
  - Finite length n → the queue can hold at most n messages
  - Queue not full: message is stored in the queue (either a copy or a ref); sender can continue execution without waiting
  - Queue full: sender blocks until space is available
- Unbounded capacity
  - Potentially infinite length
  - Sender never blocks

Example – Message Queue

```
import java.util.*;

public class MessageQueue
{
  public MessageQueue() {
    queue = new Vector();
  }

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

```
public Object receive() {
   Object item;
   if (queue.size() == 0)
     return null;
   else {
     item = queue.firstElement();
     queue.removeElementAt(0);
     return item;
   }
 }
 private Vector queue;
}
```

Example – Message Queue

- Message Queue for Producer Consumer Example from lecture 3
- Buffer is unbounded and provided by Vector class
- send() and receive() are non-blocking
- Consumer needs to evaluate the result from receive()! – message may be NULL
- Race condition on buffer full/empty checks!