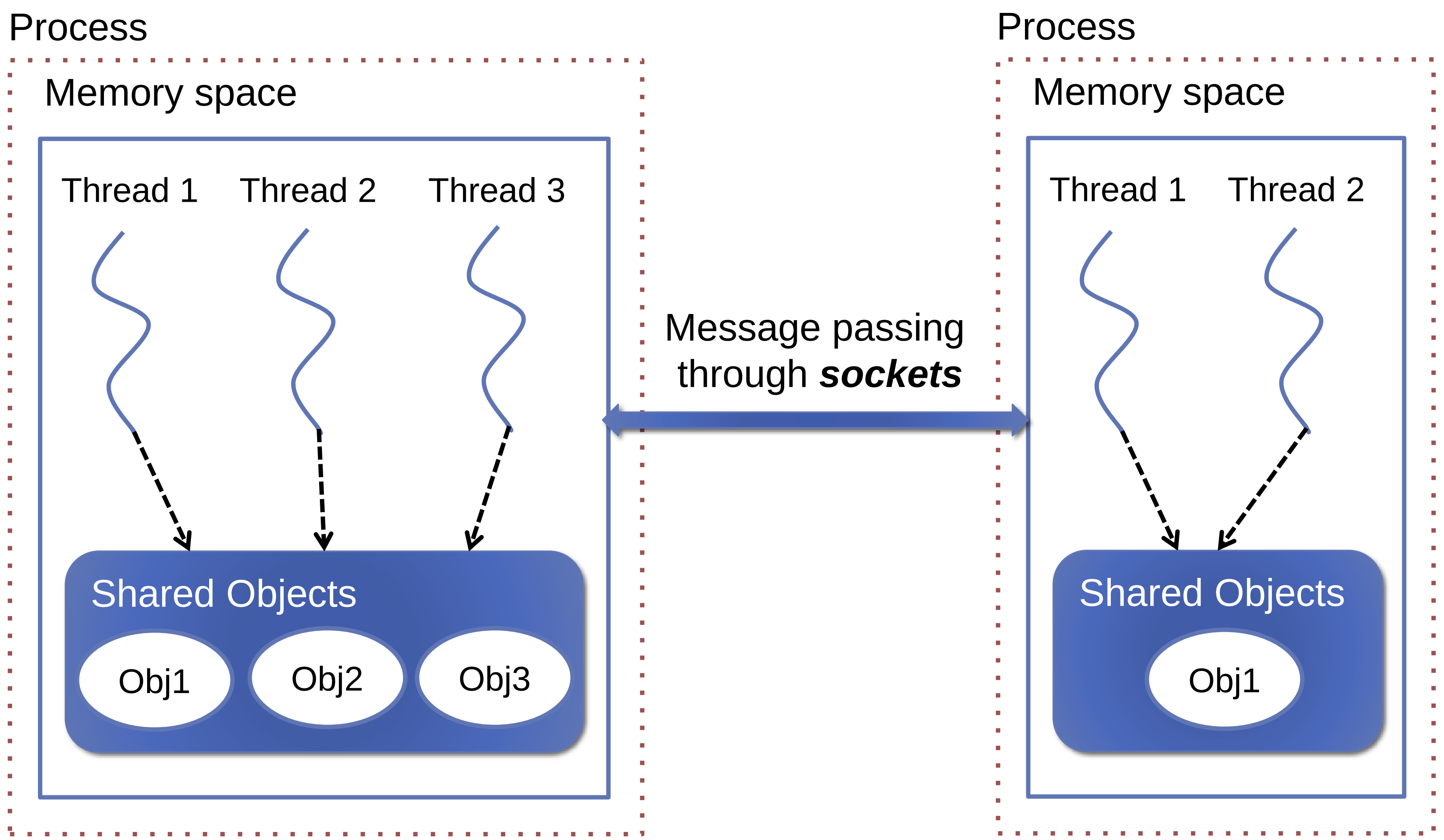
DISTRIBUTED SYSTEMS

COMP90015 2018 SM2

Agenda

- **Process vs. Thread**
- Sockets + Threads Demo
- Project Questions

Process vs. Thread

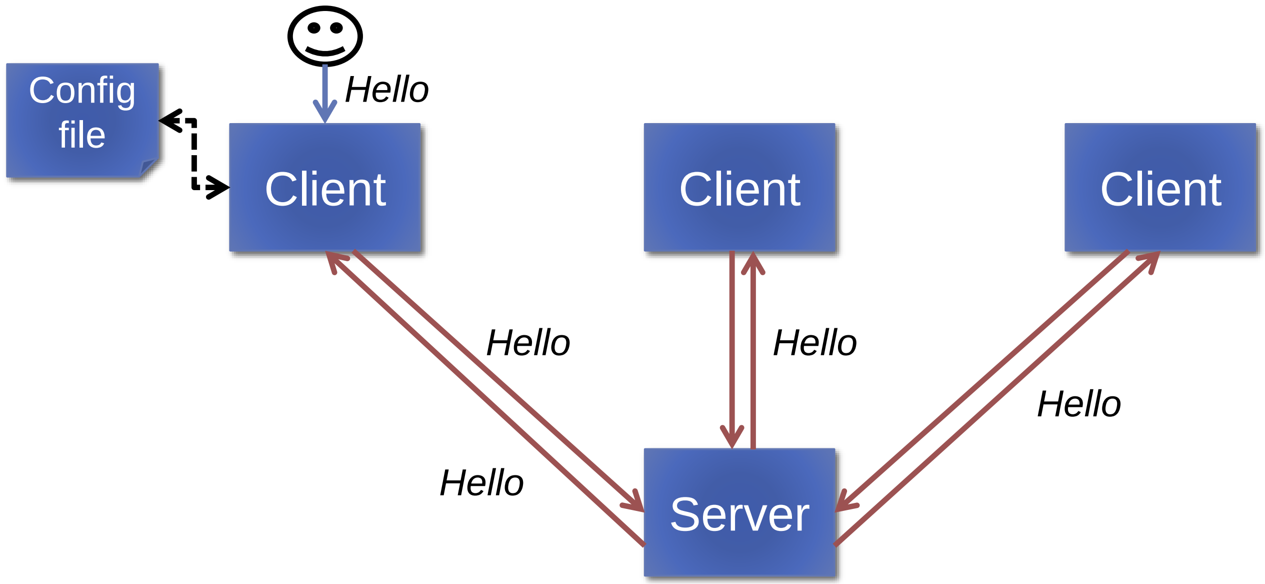


Agenda

- Process vs. Thread
- **Sockets + Threads Demo**
- Project Questions

Sockets + Threads Demo

- Message broadcaster
  - Clients read server IP/port from config file
  - Clients establish a TCP connection with the server
  - Users type a message using the console, the client sends it to the server, the server broadcasts that message to all clients currently connected (including the sending client) and the clients display the message to the user



Message Broadcaster

- Client design
  - Single-threaded? Multi-threaded?
    - One thread to ‘listen’ for user input
    - One thread to ‘listen’ for incoming messages from the server
  - When does it establish the TCP connection with the server?
    - After loading the server port/IP from the file
  - When does it send a message to the server?
    - When the user inputs a line of text in the console
  - What does it do with messages received from the server?
    - It prints them to the console
- Server design
  - Single-threaded? Multi-threaded?
  - Server state?

Server Design

- Example 1
  - One main thread listening for incoming connection requests
  - One thread per client connection
  - Each thread is responsible for listening for incoming client messages
  - One coordinating object (singleton) responsible for maintaining server state and processing client messages one at a time (synchronized)

Server Design

- Example 2
  - One main thread listening for incoming connection requests
  - One thread per client connection
  - Each thread is responsible for listening for incoming client messages AND processing them
  - A single (singleton) object holding the server state

Server Design

- Example 3
  - One main thread listening for incoming connection requests
  - Two threads per client connection:
    1. Listens for incoming client messages and places them in a queue
    2. Monitors the queue, takes messages from it when available and process them
  - A singleton object holds the server state