Electronics and
Computer Science

UNIVERSITY OF
Southampton

Message Queues

COMP3211 Advanced Databases

Dr Nicholas Gibbins – nmg@ecs.soton.ac.uk
2014-2015

UNIVERSITY OF Southampton

Transactions Revisited

Complete units of work – ACID

Transactions may be distributed across more than one processor

- Dedicated link between programs
- Operations synchronized
- Distributed two phase commit

Many transactions need not be completed synchronously

- Communication must still be guaranteed
- Work can be completed later

Synchronous versus Asynchronous

HTTP is a *synchronous* protocol

- Request from client to server is followed by reply from server to client in the same TCP connection

SMTP is an *asynchronous* protocol

- Email messages are sent on a store-and-forward basis
- Final message recipient need not be available when message is sent

Messaging and Queuing (MQ)

MQ enables programs to communicate across a network asynchronously

- No private, dedicated link required
- Systems can be heterogeneous
- Message delivery is guaranteed

Messages are placed on queues by one system and taken off by another

- Similar idea to email, but more sophisticated mechanism

MQ can also be used between programs running on the same system

Direct Transaction Processing

Conventional (direct) transaction processing has weaknesses:

- Designed for synchronous processing
- Has difficulties with long-lived transactions and communication errors
- Difficult to balance loads between several servers carrying out the same tasks
- Difficult to prioritise one request over another

Direct TP Problems: Server Failure

If server is down, client does not receive an immediate answer

Cannot distinguish between:

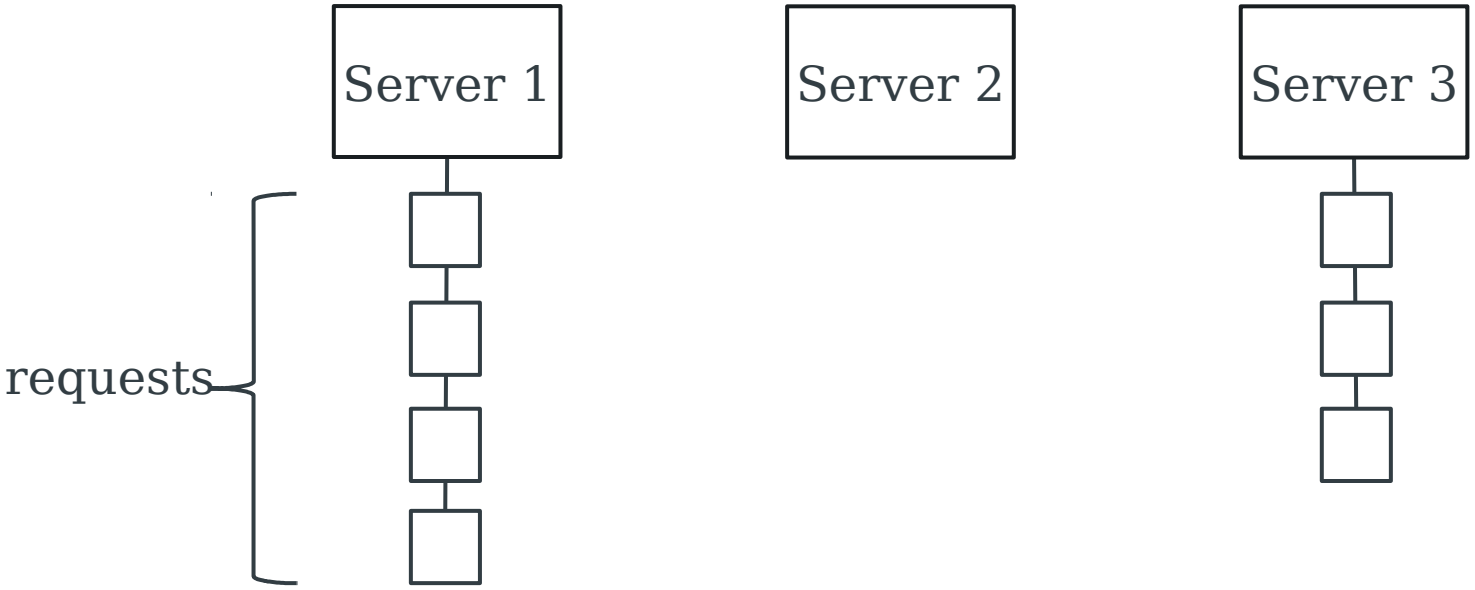
- request not delivered to server
- server failure
- reply not delivered to client

Direct TP Problems: Client Failure

Cannot tell if response has been received by client

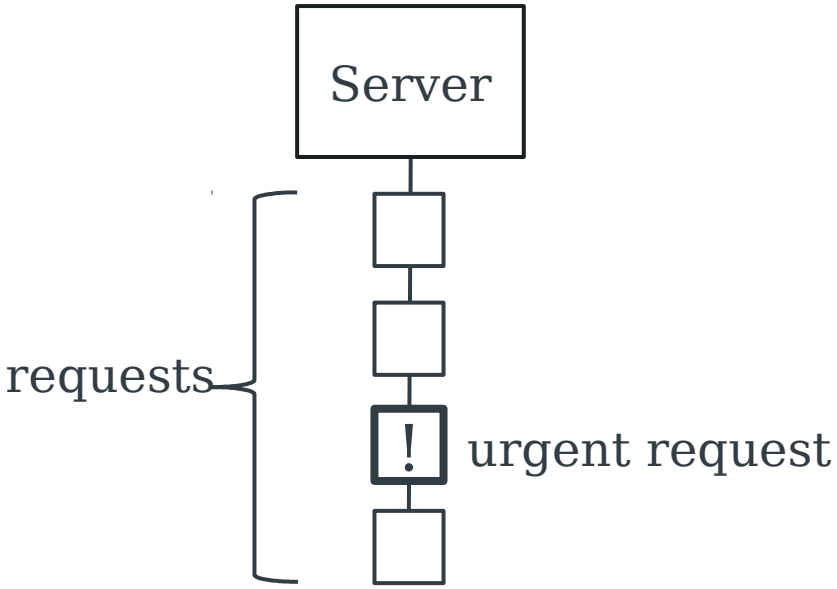
Direct TP Problems: Unbalanced Load

Cannot ensure that servers carrying out a task have an even load



Direct TP Problems: No Prioritisation

Tasks handled on a first-come, first-served basis; cannot process high-priority requests early

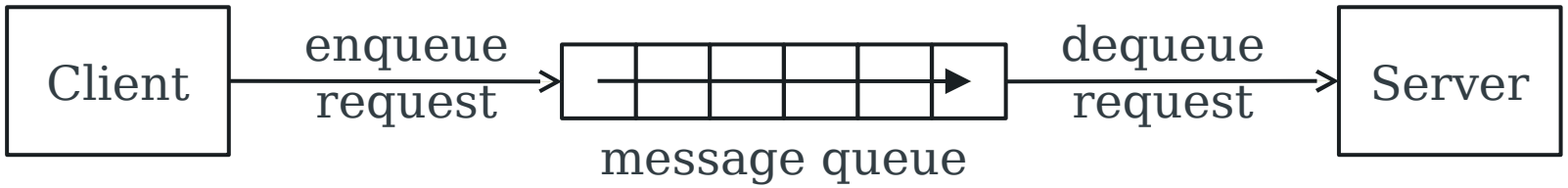


Message Queues

Client adds request to queue (enqueues)

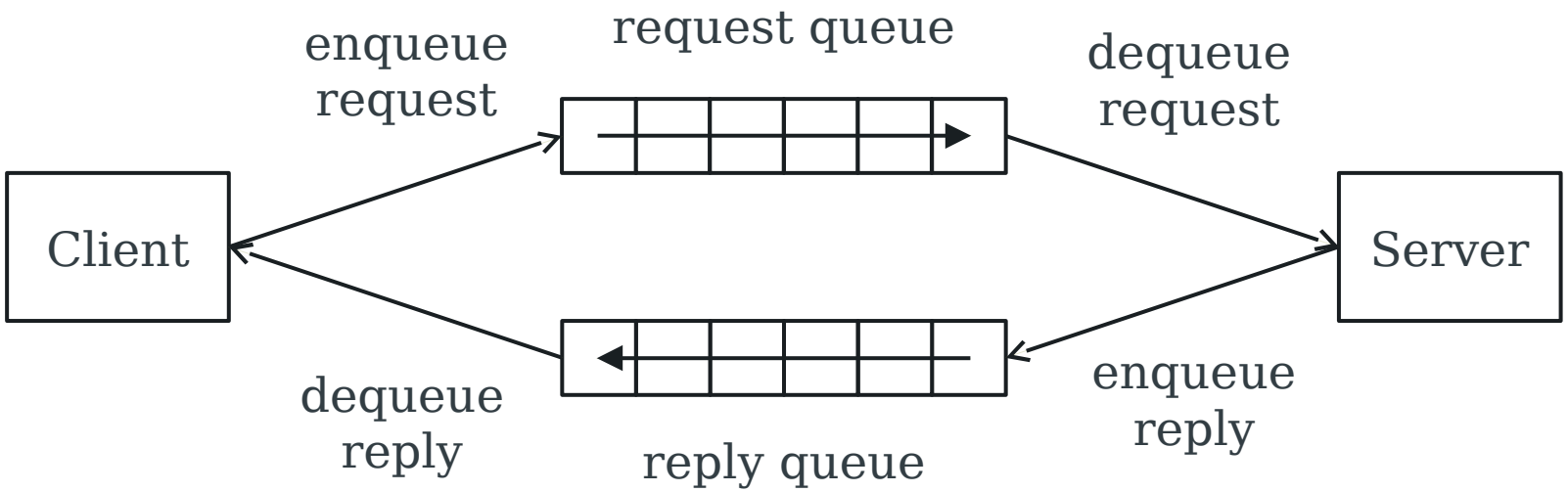
Server removes request from queue (dequeues)

If server is unavailable (busy, down, disconnected), the request is stored in the queue until the server is able to process it



Bidirectional Queues

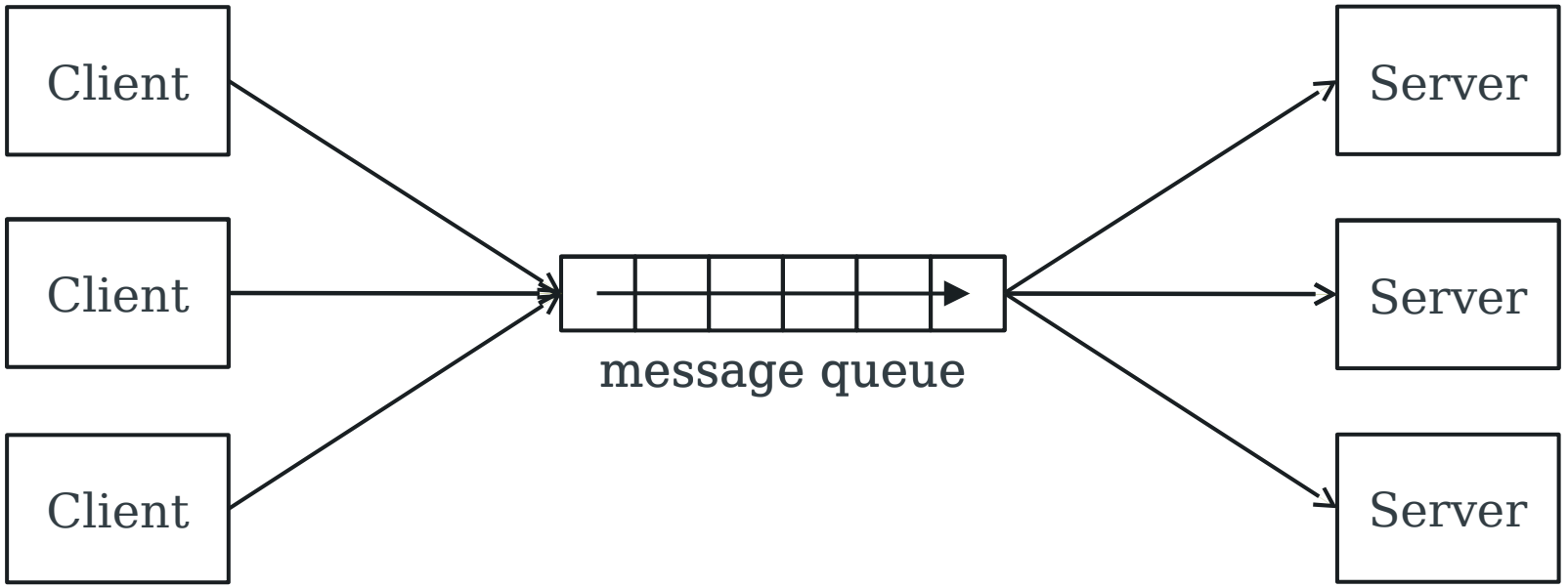
Bidirectional communication between client and server requires a separate queue for communications in each direction



Application Structure

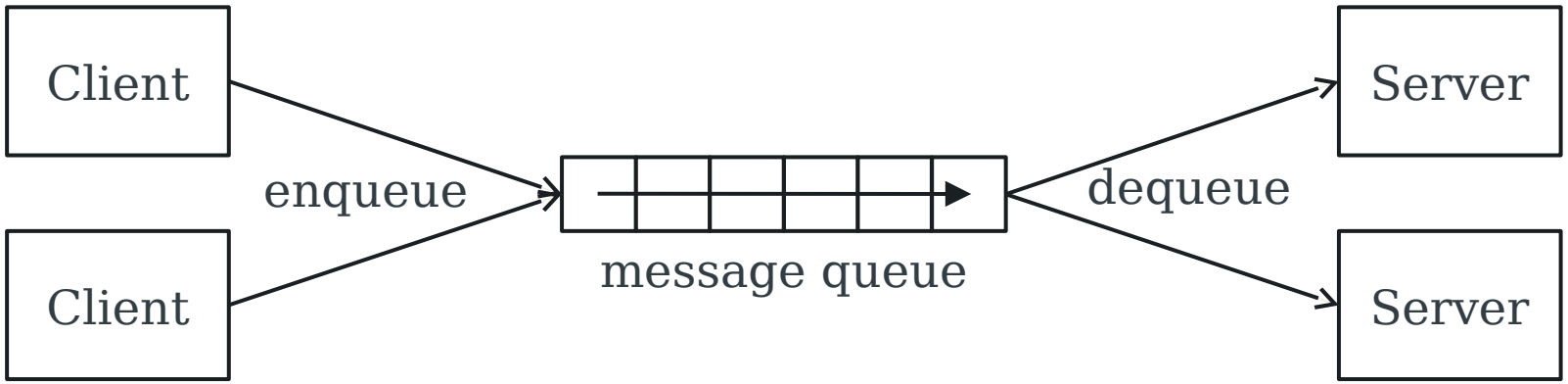
Communication may be one-to-many

Communication may be many-to-one

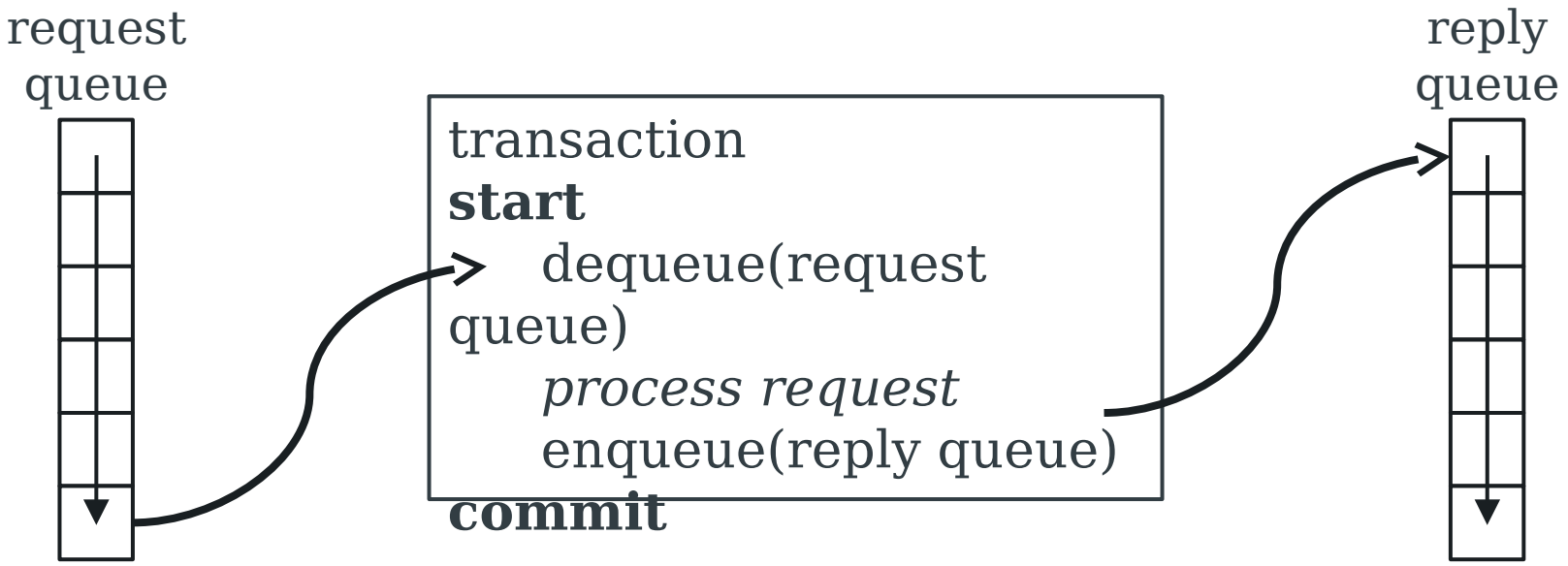


Load Balancing

When a server finishes processing a request, it takes the next from the queue



Queued Transaction Processing



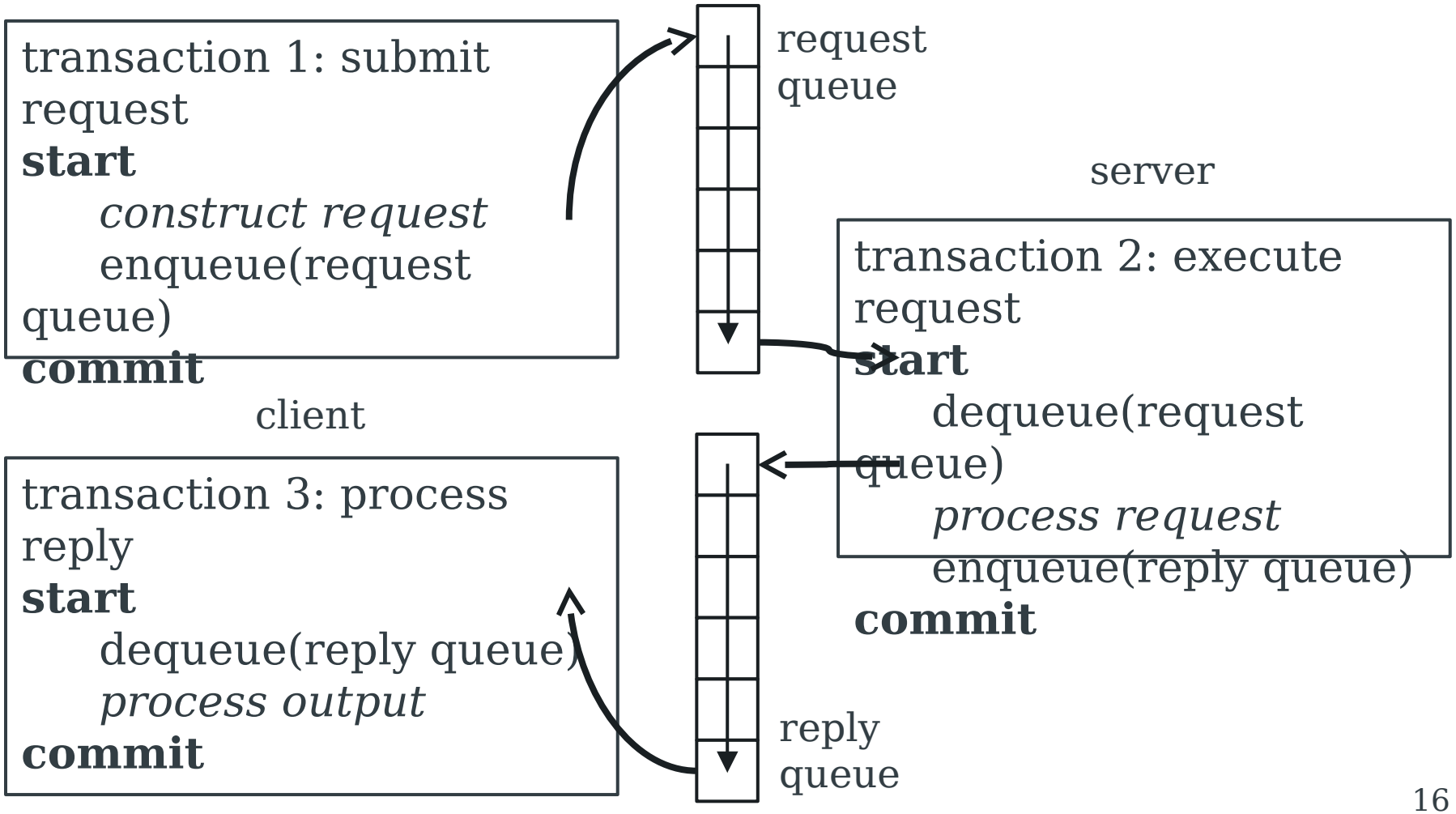
Queued Transaction Processing

If transaction aborts:

- request returned to input queue
- changes made by transaction are rolled back
- if necessary, reply removed from output queue

Repeated aborts (due to a poisoned message) may be prevented with a maximum limit on retries

Queued Transaction Processing



Message Ordering

Description of message queues so far does not consider how messages are ordered in a queue

- First-come, first-served
- Highest-priority-first

Aborted transactions may lead to out-of-order processing:

- T1 dequeues M1
- T2 dequeues M2
- T2 commits
- T1 aborts, returns M1 to queue

Benefits

- Simple APIs hide communications complexity
- Fewer constraints on inter-program operation
  - Programs communicate indirectly – asynchronous
  - Client and server do not need to be running at the same time
- Fewer network sessions needed, and programs are less vulnerable to network failures
- Business change easier to handle
- Assured message delivery
- Asynchronous transaction processing

Implementations and Standards

Message Queuing is not a new idea!

- Used for, example, within IBM's Information Management System (IMS) 30 years ago

Both proprietary and open source APIs and platforms

- IBM Websphere MQ, Microsoft Message Queuing, Oracle Advanced Queuing
- Apache ActiveMQ, Rabbit MQ

Some products inter-operate via common standards

- AMQP, MQTT

Example 1: Client-Server Applications

Insurance agents throughout the country request insurance quotations using an online system.

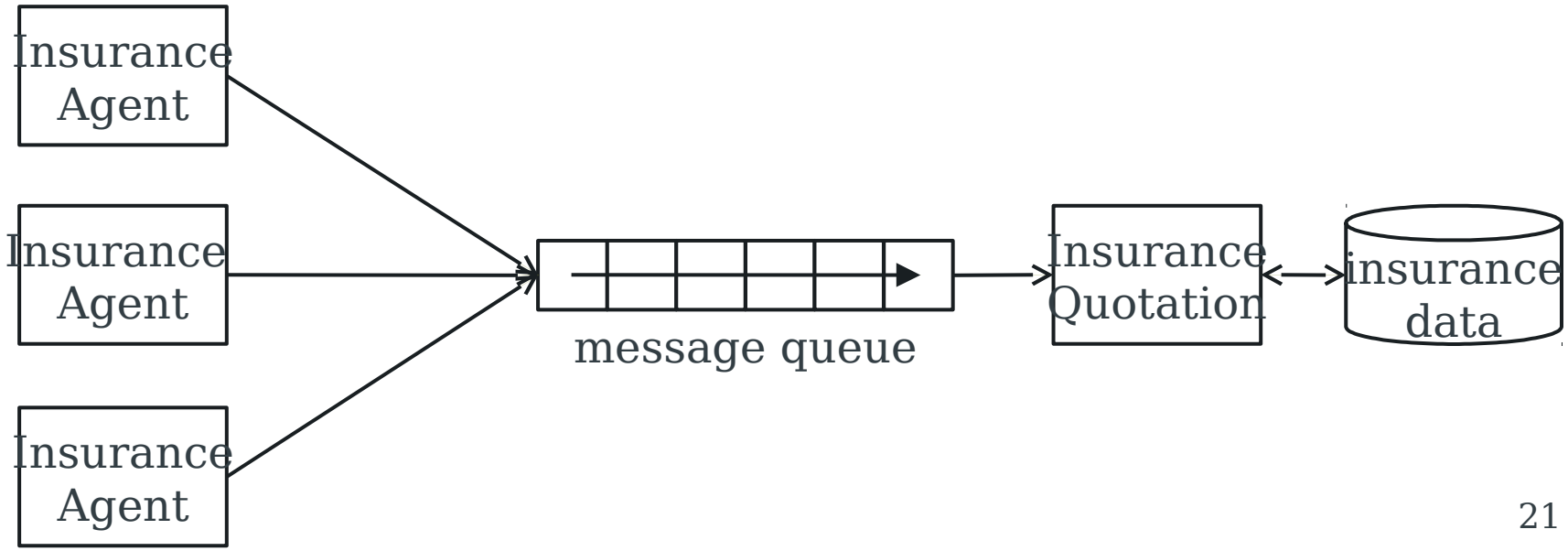
This system is implemented as a traditional client-server application, with client programs (the insurance agents) sending requests for quotations to a central server program.

The server does some calculations using data from a central insurance database, then sends a quotation to the requesting agent.

Example 1: Client-Server Applications

Client programs put request messages on a single queue, from which the server program takes them.

Responses may be sent back to clients via extra message queues (one per client)



Example 2: Output-only Devices

A device providing a service is output-only, and does not send messages back to the requester. Only one-way message flows are required.

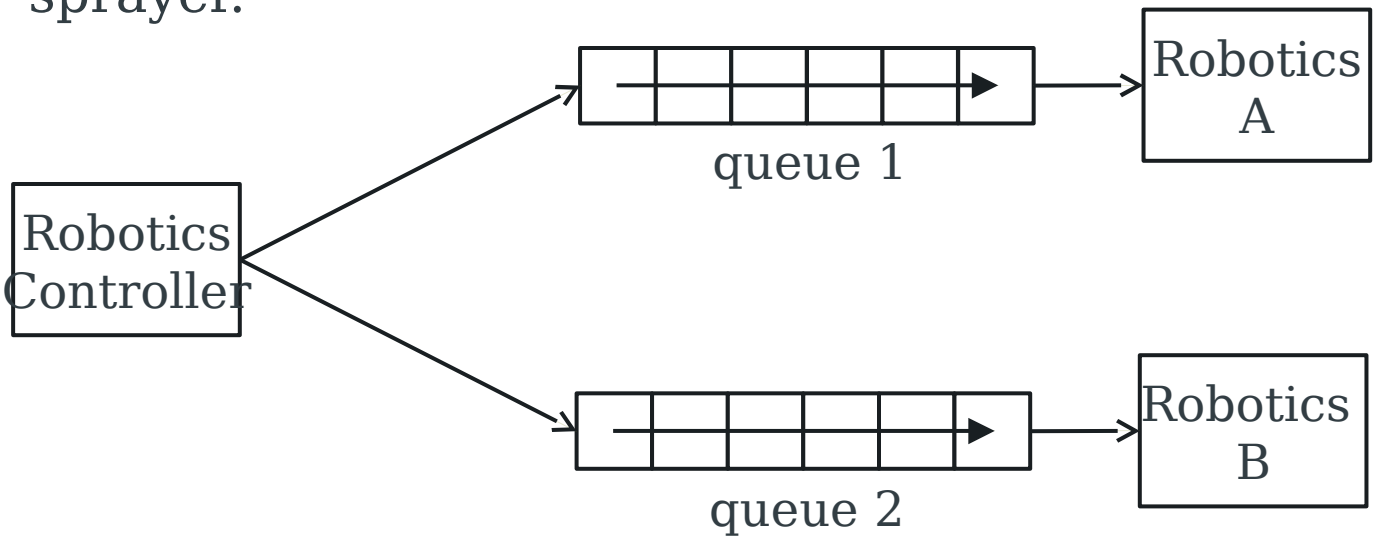
Output-only devices:

- Printing devices
- Displays:- Stock exchange, Flight arrivals and departures
- Factory floor robotics

Example 2: Output-only Devices

Robotics Controller is in control of an automated manufacturing process. It puts messages on:

- Queue 1 for the Robotics A program, which directs some welding machinery
- Queue 2 for the Robotics B program, which controls a paint sprayer.



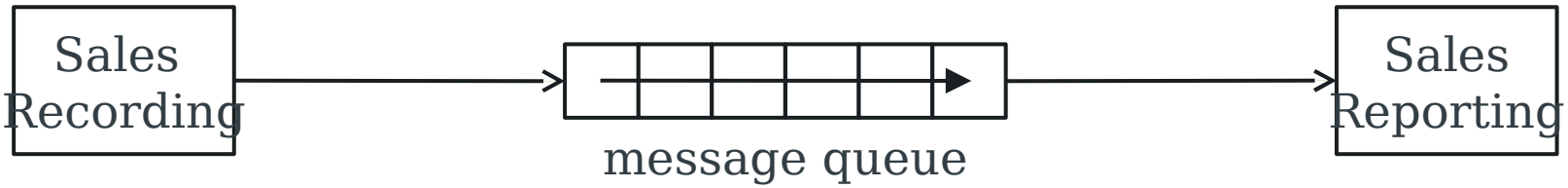
Example 3: The Batch Window

A department store writes its sales figures to a file throughout the day's trading. Overnight, a report of the day's sales is produced using this file of data as input. The report must be on the Sales Manager's desk before the next day's trading begins.

- The amount of time available for producing the report is limited to the "window" of time between the end of business on one day and the start of business on the next
- The start and finish times for the activity are fixed

Example 3: The Batch Window

Instead of operating in sequence and communicating via a file, the two programs could run independently of each other and communicate using a message queue



Summary

Message queuing allows an alternative way to distribute applications

Many applications involve databases, so the technologies intersect

Conceptually and practically, MQ is often a more straightforward approach than direct connections between systems, and synchronous transaction processing