

Bachelor of Information Technology

Middleware Architecture
IT6505



Teacher's Notes

Lesson 08



Viraj Brian Wijesuriya

University of Colombo School of Computing

No. 35, Reid Avenue , Colombo 7, Sri Lanka.

Message Broker Pattern (1)

- Message broker is an intermediary program module that translates a message from the formal messaging protocol of the sender to the formal messaging protocol of the receiver.
- Message brokers are elements in telecommunication networks where software applications communicate by exchanging formally-defined messages.
- Message brokers are a building block of Message oriented middleware.

Message Broker Pattern (2)

- **Pattern** is a general repeatable solution to a commonly occurring problem in software **design**.
- A message broker is an architectural pattern for message validation, transformation and routing.
- It mediates communication amongst applications, minimizing the mutual awareness that applications should have of each other in order to be able to exchange messages, effectively implementing decoupling.
- The purpose of a broker is to take incoming messages from applications and perform some action on them.

Actions taken by Message Broker

- Route messages to one or more of many destinations
- Transform messages to an alternative representation
- Perform message aggregation, decomposing messages into multiple messages and sending them to their destination, then recomposing the responses into one message to return to the user
- Invoke Web services to retrieve data
- Respond to events or errors

Message Broker Functionality

- A message broker may be used to manage a workload queue or message queue for multiple receivers, providing reliable storage, guaranteed message delivery and perhaps transaction management.
- Some message broker software:
 - Apache ActiveMQ
 - Apache Kafka
 - HornetQ (Red Hat)
 - RabbitMQ (Mozilla Public License, written in Erlang)
 - WSO2 Message Broker
- Many message-oriented middleware systems require a message transfer agent (message broker).

Question 1 (1)

- “Some messaging functions do not require an intermediary message broker.” Do you agree with this statement?
- Yes
- For example, end-point objects can take the roles of publisher and subscriber.
- Let see elaborate more on this.

Question 1 (2)

- Publish–subscribe is a messaging pattern where senders of messages, called **publishers**, do not program the messages to be sent directly to specific receivers, called **subscribers**, but instead characterize published messages into classes without knowledge of which subscribers, if any, there may be. Similarly, subscribers express interest in one or more classes and only receive messages that are of interest, without knowledge of which publishers, if any, there are.
- Publish–subscribe is a sibling of the message queue paradigm, and is typically one part of a larger message-oriented middleware system.
- Most messaging systems support both the pub/sub and message queue models in their API, e.g. Java Message Service (JMS).

Message Queueing

- We have been talking about middleware which has been about program-program.
- Message queueing is about **program-to-message queue**.
- **Message queue**: a very fast mailbox and you can put a message in the box without the recipient(s) being active.
- To put a message in the queue program does **Put**.
- To take a message out of the queue program does **Get**.

Duty of the Middleware

- Middleware does the transfer of messages from an to queue.
- It ensures, whatever happens to the network, the message arrives eventually.
- Only one copy of the message is placed in the destination queue.

Characteristics of a Queue

- Queues have names.
- Independent of the programme.
- Stored in the Queue during network failure.
- Queues can be backed upon disks and retrieved when system is up again.
- Queue can be a resource manager & corporate with the transaction Manager to manage transactions & queues.

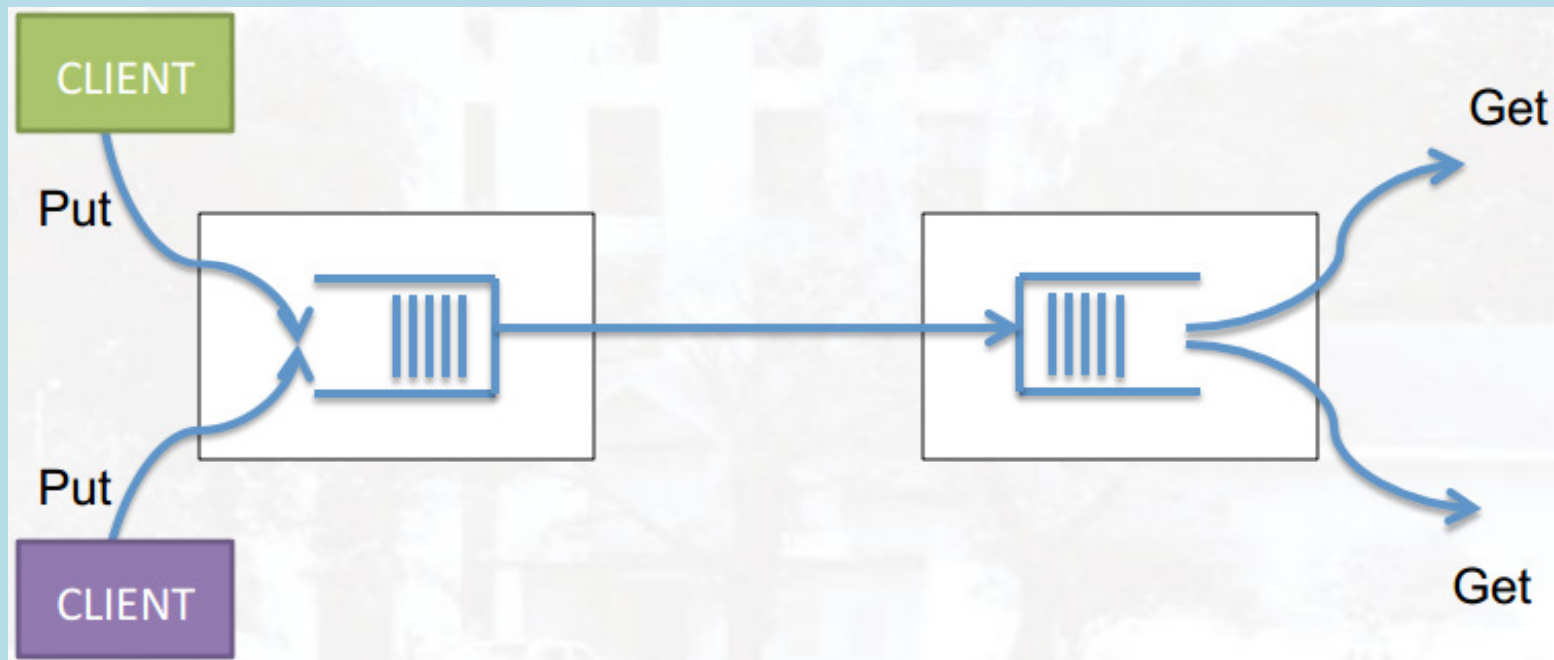
Message Queues and IDL (1)

- A disadvantage of message queueing is that there is no IDL.
- This means there is no **Marshalling**.
- It is up to you to ensure that the sender and the receiver knows the message layout.

Message Queues and IDL (2)

- MQ lacks an IDL hence there are special programs to reformat messages for different platforms.
- A peer---to---peer middleware rather than a Client/Server .
- Light versions are aimed at Mobile devices, cause when devices come online are free they can process the queues.

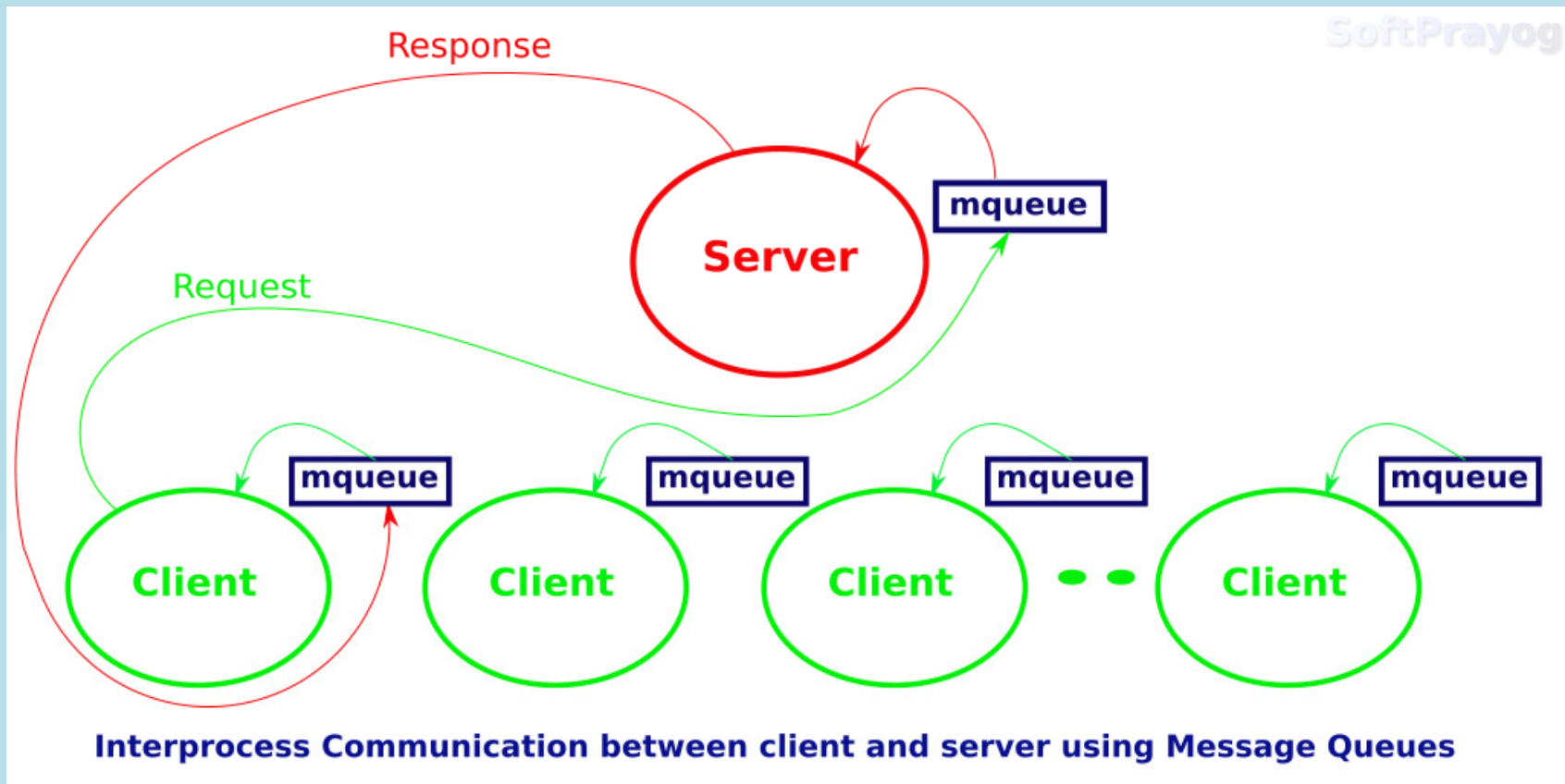
How Message Queues Work



Question 2 (1)

- You hear people talk about clients and servers with message queues. What are they talking about?
- A message queue server physically stores the queue
- The client does Puts and Gets
- Use a protocol to transfer the messages to the server
- Server does the real Puts and Gets

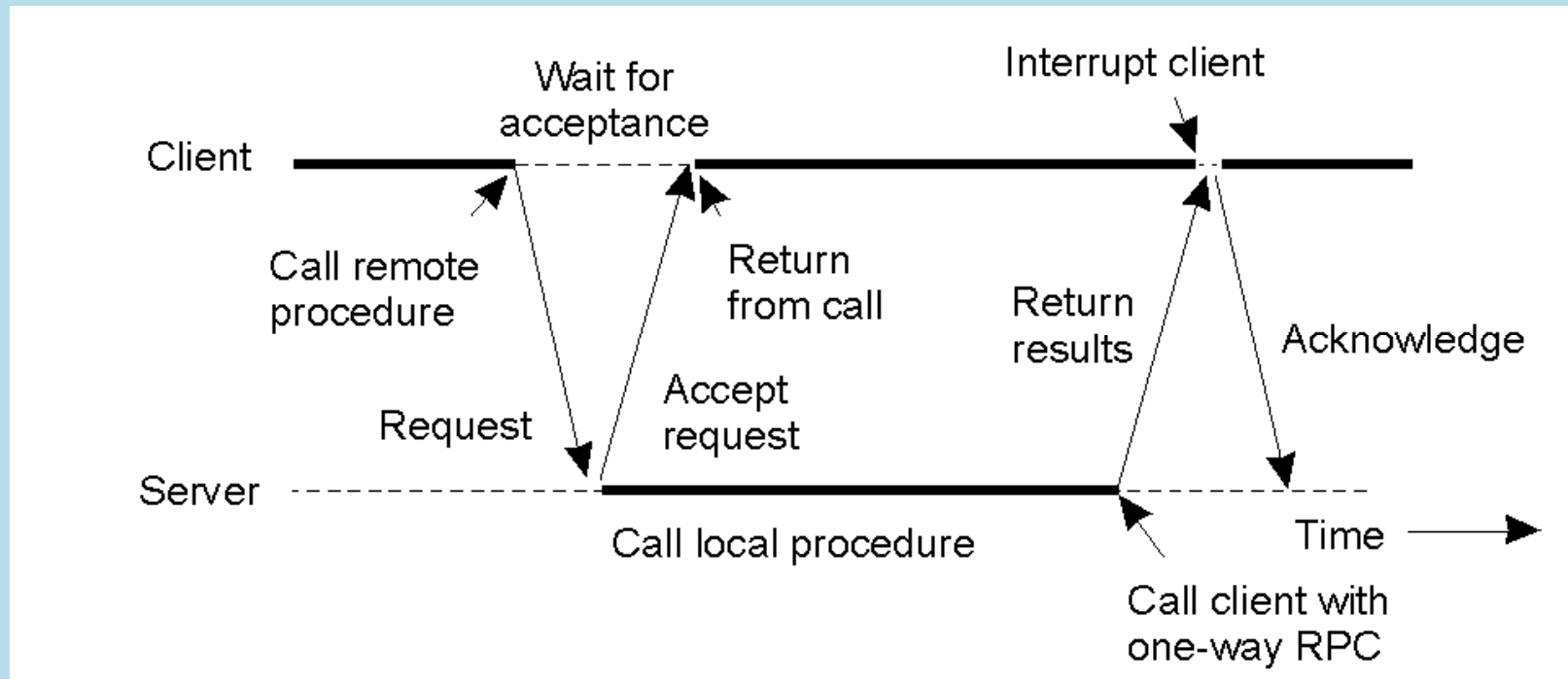
Question 2 (2)



Message Queueing vs RPC (1)

- Message queue is a very fast mailbox since you can put a message in the box without the recipient being active.
- In contrast, RPC which is more like a telephone conversation.
- Client-Server message queueing services use RPC-like protocols to transfer the messages to the server.
- Both Message Queueing and RPC supports Asynchronicity and Synchronicity.

Message Queueing vs RPC (2)



Question 3

- “Implementing Message Queues rely a lot on having proper knowledge of Queueing Theory.” What is Queueing Theory?
- Queueing theory is the mathematical study of waiting lines, or queues.
- In queueing theory a model is constructed so that queue lengths and waiting time can be predicted.
- Queueing theory is generally considered a branch of operations research because the results are often used when making business decisions about the resources needed to provide a service.

Lesson Summary

- Message Broker Pattern as a Building Block of Message-Oriented-Middleware.
- Discussed details of Message Queueing.
- Message Queueing vs RPC.
- Clients and servers with message queues.