UNIVERSITAT ROVIRA i VIRGILI

Lab 4 – Indirect communication and RabbitMQ

Distributed Systems

Indirect communication

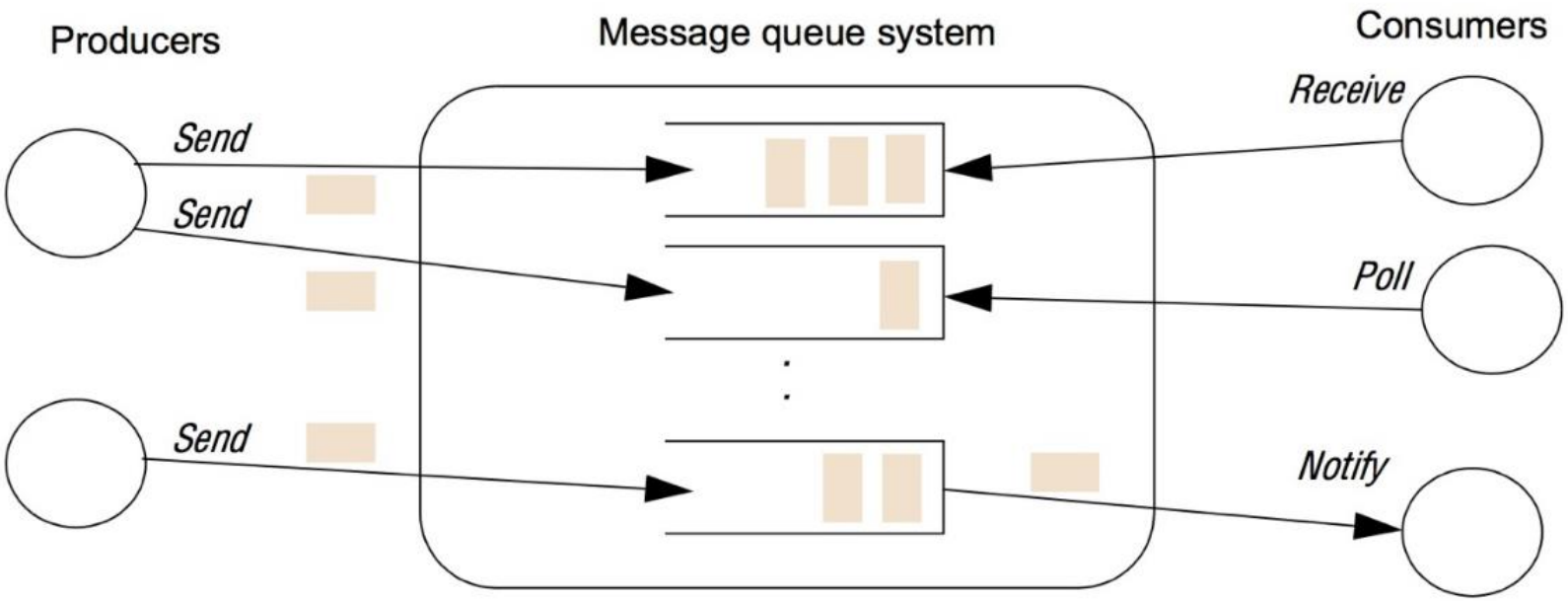
Indirect communication

- Communicating components in a distributed systems through an **intermediary**.
- System components are **loosely coupled**.
  - They are not directly connected.
  - They have little or no knowledge of each other.
- Typical in complex distributed systems.

Message Oriented Middleware

- A **Message Oriented Middleware** (MOM) enables **indirect communication** through **messages** among system components.
- System components are **space** and **time-decoupled**.
- Messages are sent **asynchronously** to **queues,** from/to which system components consume/publish.
- MOMs allow the implementation of communication patterns such as the **publish-subscribe** model.

Publish/subscribe model



Source: Barry Linnert, linnert@inf.fu-berlin.de, Netzprogrammierung WS 2015/16

Why use MOMs?

- For building **scalable** systems (variable senders/receivers).
- To ensure the **order** of arrival of messages.
- For **uncoupled** systems (such as the Cloud).
- For **load balancing**.
- For complex communication patterns (such as *one-to-all*).

RabbitMQ

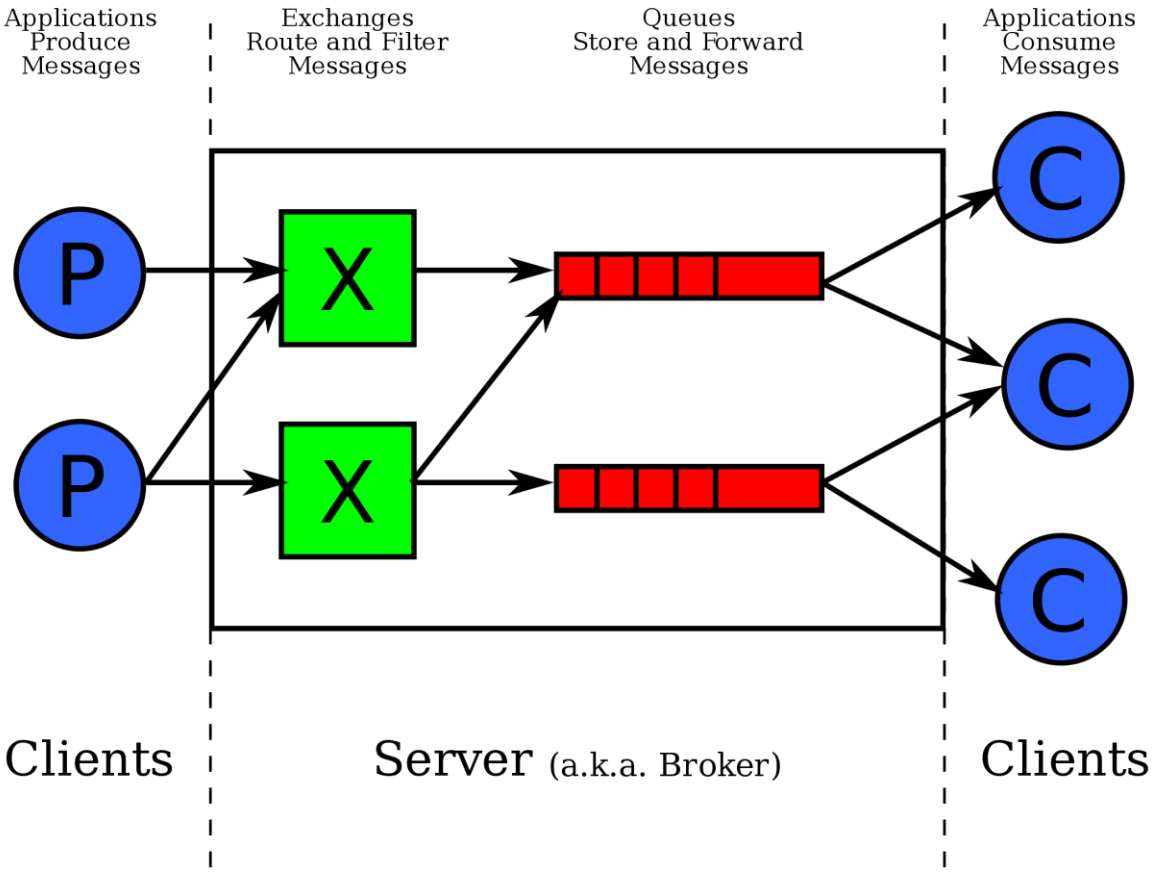
RabbitMQ

- RabbitMQ is an **open-source MOM**.
- Enables **indirect communication** via the **Advanced Message Queuing Protocol** (AMQP).
- Based on the **publish/subscribe** model.
- RabbitMQ provides **fault tolerance** mechanisms:
  - Allows persistence of enqueued messages to disk.

AMQP

- **Message-oriented application-level** indirect communication protocol.
- AMQP provides the following **basic components**:
  - **Exchanges**: responsible for delivering messages to queues. Types: **direct**, **fanout**, **topic,** etc.
  - **Messages**: they feature a **header** and a **body**.
    - **Header**: **routing key** (used by the exchange to determine destination), **delivery mode** (persistent/non-persistent), **priority**, **expiration**.
    - **Body**: content of the message (bytes).
  - **Queues**: receive messages. They are **bind** to one or more exchanges. Messages from queues can be consumed in a **pull** or **push**-based fashion. Queues are identified by a queue name.

AMQP



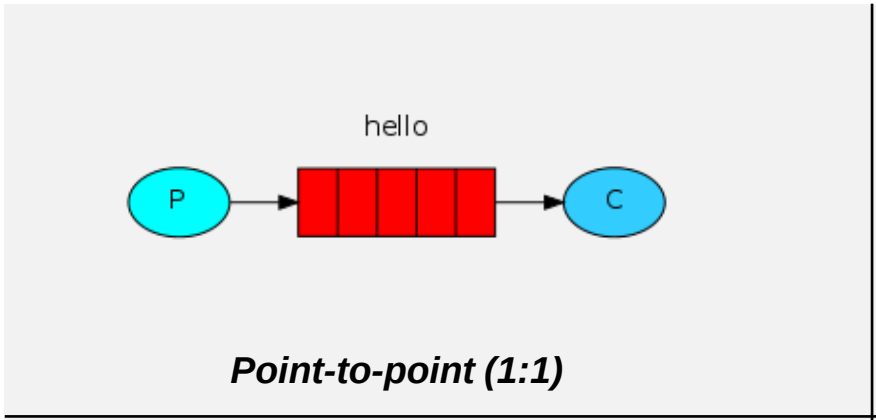
Source: https://es.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol#/media/Archivo:The-amqp-model-for-wikipedia.svg

RabbitMQ - Serialization

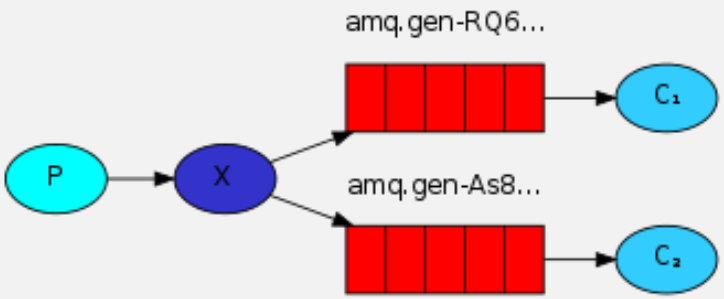
- RabbitMQ and the AMQP do not provide a **serialization** method.
- It is the developers' task to **serialize/deserialize message bodies**.
- The message body in the AMQP must be represented in **binary** format.
- Multiple serialization methods (Python): **Pickle** for any object**,** string **encode**/bytes **decode** for strings, **JSON** + encode/decode for dictionaries, etc.

RabbitMQ

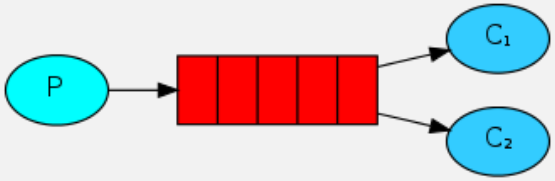
- RabbitMQ allows implementing various **routing patterns**:



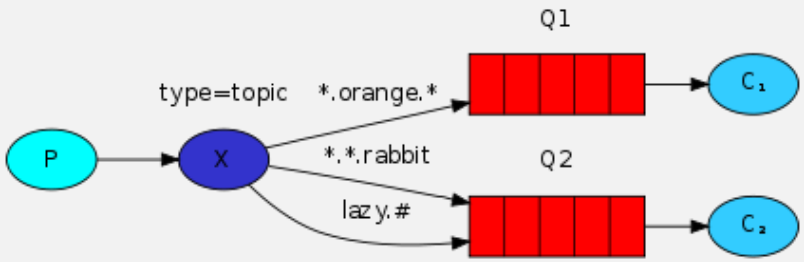
Point-to-point (1:1)



Publish/subscribe (1:N)



Load-balancing (1:1)



Topics (1:N)

Source: https://www.rabbitmq.com/getstarted.html

Exchange types

Exchange Type	Routing Key Needed?	Routing Rule	Use Case
Direct	✅ Yes	Exact match	Logs by severity (error, warning)
Fanout	❌ No	Broadcast to all	Notifications, live updates
Topic	✅ Yes	Wildcard match (*, #)	Logs by category, filtering

Using RabbitMQ

What do we need to use RabbitMQ?

1. To have RabbitMQ installed in our computer.
2. To have RabbitMQ up and running and listening from a certain port (standard port: 5672).
3. To have a RabbitMQ client for our programming language installed (for Python, we will use **`Pika`**).
4. To follow RabbitMQ's beginner tutorials!

Lab 4 assignment

- Download and install RabbitMQ.
- **Python 3.8+** will be used. Additional modules:
  - **Pika** 1.3.1 (Python AMQP client).
  - `pip3 install pika`
- Implement the “Hello World” and “Work queue” examples.
- Test multiple **serialization methods**: Pickle, string encode/bytes decode, JSON + encode/decode.

Running RabbitMQ

```
docker pull rabbitmq:management
docker run -d --name rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:management
```

http://localhost:15672

Guest
Guest

Lab 4 assignment

- **Resources:**
  - Gentle introduction to RabbitMQ and AMQP.
  - RabbitMQ installation.
  - RabbitMQ tutorials for **various communication patterns**.

Which patterns will you use in task 1?

Task 1 - InsultService

Use a Rabbit queue two create the following services:

- InsultProducer: A producer that sends new insults to the queue every five seconds.
- InsultConsumer: A consumer that reads new insults from the queue and them to a REDIS list (INSULTS) only if they are new.

Use Rabbit exchange to create the following services:

- InsultBroadcaster: A publisher that sends insults from INSULT list to a pubsubchannel.
- InsultReceiver: A subscriber that subscribes to events from the broadcaster. Try to launch several InsultReceivers to check that it works.

Task 2 - TextFilter with a Work Queue

TextProducer: A producer that sends a text without insults to a Work Queue every 5 seconds

AngryProducer: A producer that sends a text with insults to a Work Queue every 3 seconds

Insultfilter: A consumer that retrieves a text from the Work Queue, replaces insults from INSULT list in the text for "CENSORED" and store the clean text in a RESULTs list.

Launch several InsultFilters to check that the work is distributed among workers.

UNIVERSITAT
ROVIRA i VIRGILI