Distributed Systems
Communication Paradigms

Universidad Icesi

daniel.barragan@correo.icesi.edu.co

Activity

- Using a tree diagram propose a classification of the communication methods that you know, provide examples

Communication Paradigms

ME ME ME ME ME ME ME ME ME ME ME ME ME ME ME ME ME ME ME ME
©joy-of-cartoon-pictures.com

Communication Paradigms

Communication Paradigms

Communication Paradigms

Key properties:

- **Space uncoupling:** sender does not known the identity of the receiver(s) and vice versa. Participants can be replaced, updated, replicated or migrated

- **Time uncoupling:** sender and receiver(s) can have independent lifetimes. The participants do not need to exist at the same time to communicate

Communication Paradigms

	Time coupled	Time uncoupled
Space coupling	?	?
Space uncoupling	?	?

Communication Paradigms

	Time coupled	Time uncoupled
Space coupling	?	?
Space uncoupling	?	**Our study case**

Communication Paradigms

	upled
Space cou	
Space unc	case

Indirect Communication

- Communication between entities in a distributed system through an intermediary with no direct coupling between the sender and the receiver(s) (Asynchronicity)

- Components in a distributed systems that were not designed to interoperate can be made to work together. Communication is based on events/messages and interfaces (Heterogeneity)

Indirect Communication - Types

- Group communication
- Publish-subscribe systems
- Message queues
- Shared memory

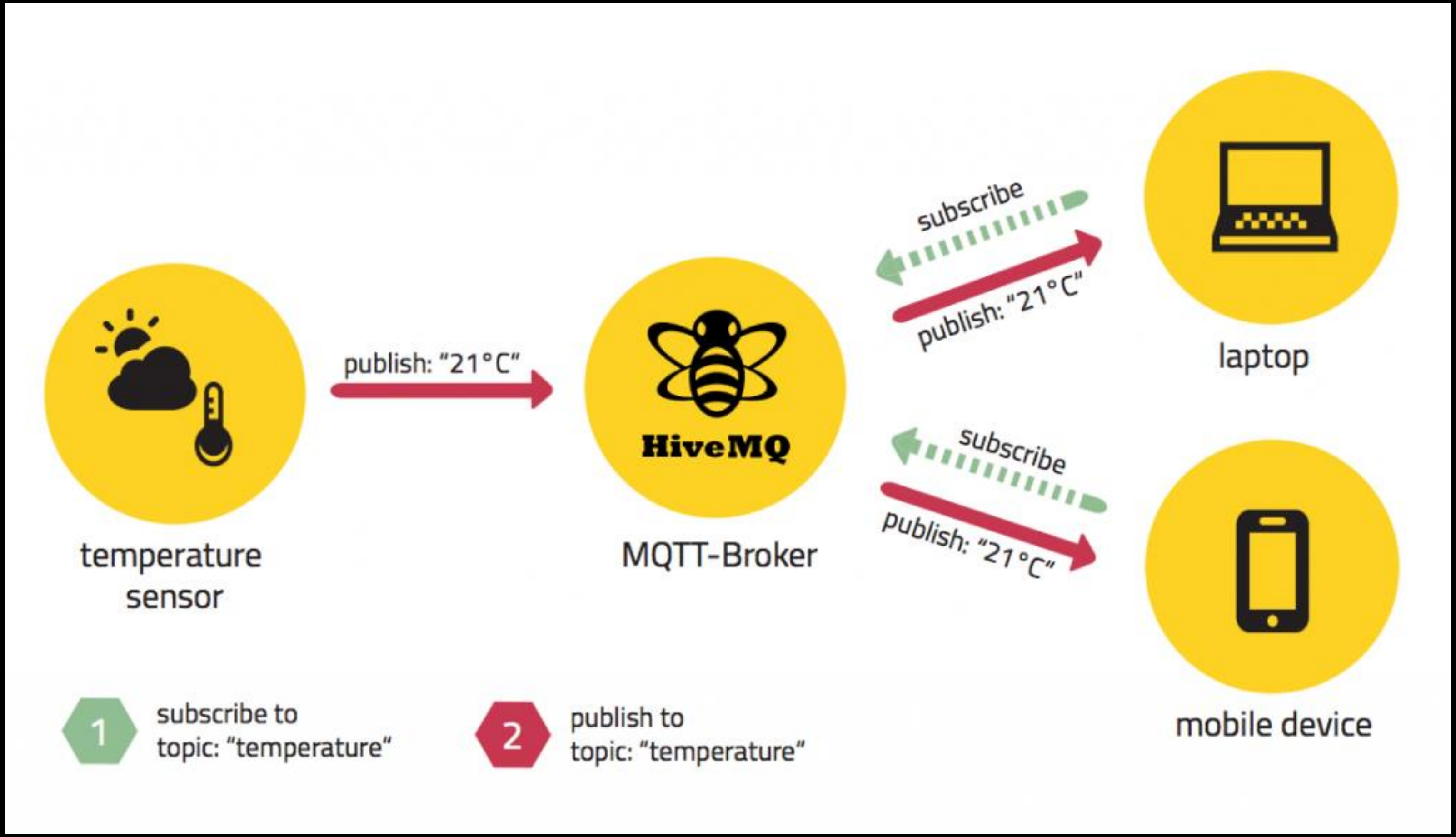
Indirect Communication - Types

- Group communication
- **Publish-subscribe systems**
- **Message queues**
- Shared memory

Publish-Subscribe

- Provides a point-to-multipoint service using a broker as an intermediary.
- Publishers publish events to an event service and subscribers express interest in particular events through suscriptions
- The task of the publish-subscribe system is to match subscriptions against published events and ensure the correct delivery of event notifications

Publish-Subscribe - Common Example



Publish-Subscribe - Properties

- Support different subscription models: Channel-based, Topic-based, Content based, Type based
- Support different types of event routing: Flooding, Filtering, Rendezvous, Gossip
- Support distributed implementations

Message Queues

- Provides a point-to-point service using a queue as an intermediary.
- Producer processes send messages and consumer processes receives messages from a specific queue
- The consumer can be on a totally different server than the producer, or they can be located on the same server

Message Queues - Reliable communication

Conditions:

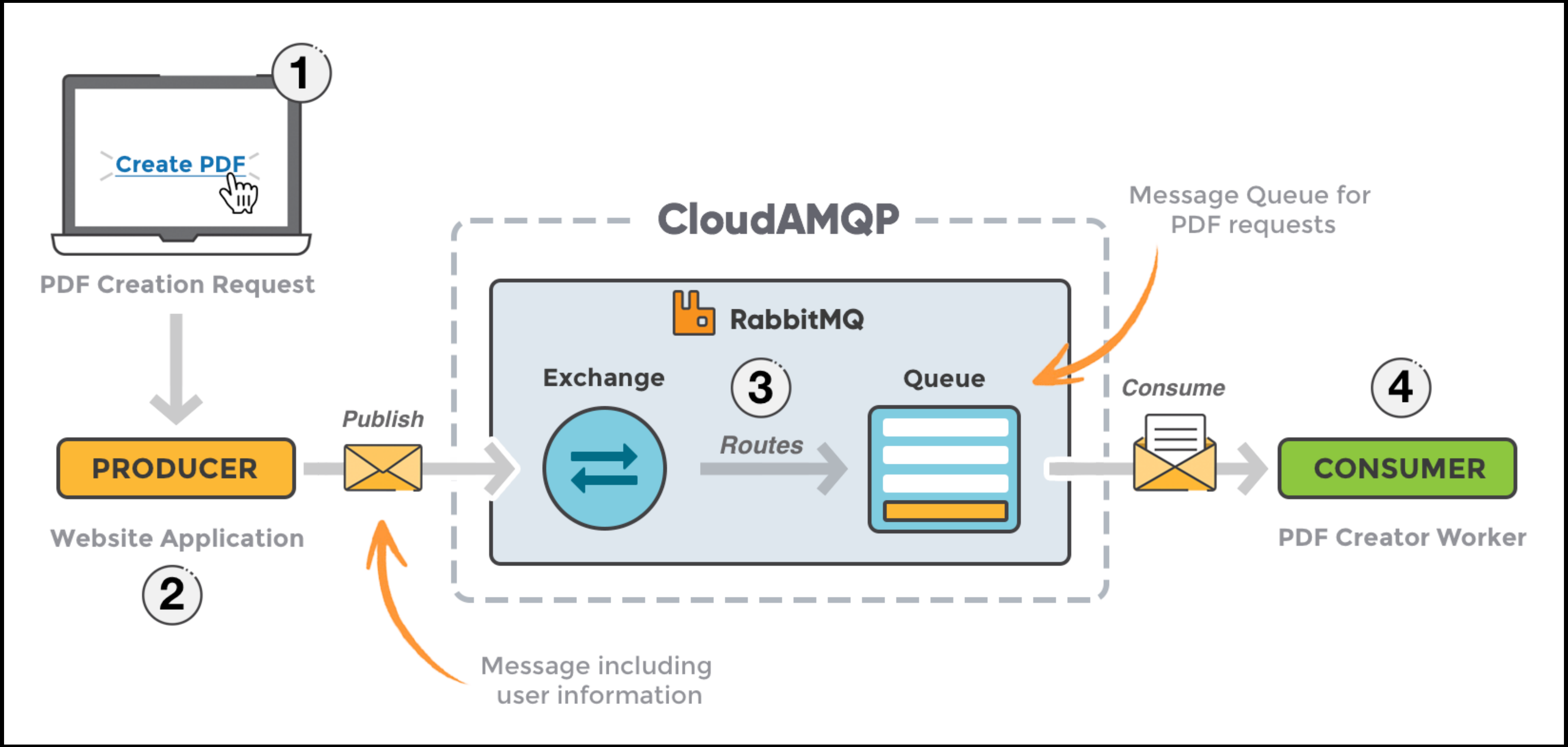
- Stores messages until they are consumed (validity)
- No messages are delivered twice (integrity)

Message Queues - Reliable communication

Conditions:

- Stores messages until they are consumed (validity)
- No messages are delivered twice (integrity)

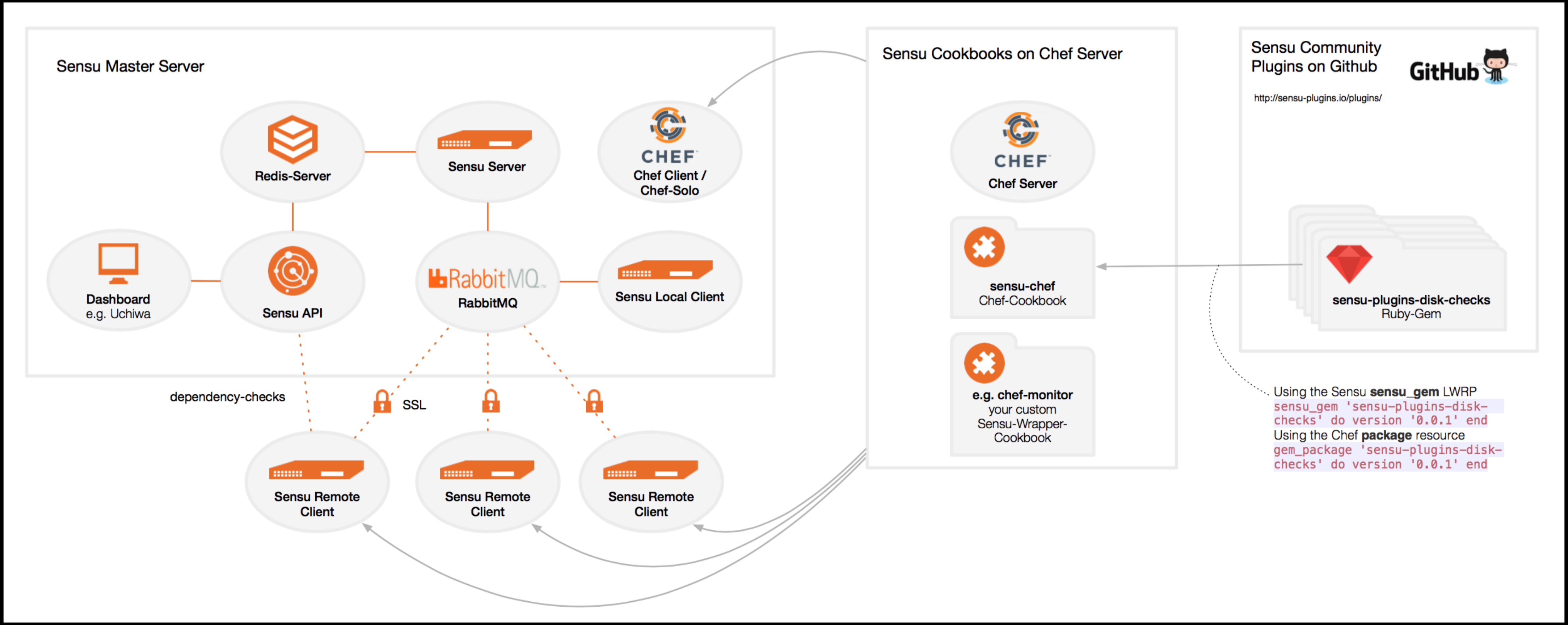
Message Queues – Common Example



Message Queues - Properties

- Supports different reception types: (Non) Blocking, Notify
- Supports metadata base delivery
- Support transactions (Rollouts, Rollbacks)
- Support for security (SSL)
- Support distributed implementations
- Payload is serialized in a standard format, usually JSON

Indirect Communication
Use Case in Distributed Systems



Using the Sensu **sensu_gem** LWRP

```
sensu_gem 'sensu-plugins-disk-checks' do version '0.0.1' end
```

Using the Chef **package** resource

```
gem_package 'sensu-plugins-disk-checks' do version '0.0.1' end
```

Activities

- Deploy a rabbitmq server and watch the status of the queue in the management console.

  https://www.rabbitmq.com/tutorials/tutorial-one-python.html

- Perform a load test against a rabbitmq queue

  https://github.com/rabbitmq/rabbitmq-perf-test

- Deploy a distributed sensu monitoring infrastructure using at least 5 nodes of the classroom

Questions?

daniel.barragan@correo.icesi.edu.co