UNICAM
Università di Camerino
1336

Università di Camerino

Data Analytics 2020-2021
Apache Qpid

Studente

Alessandro Zallocco

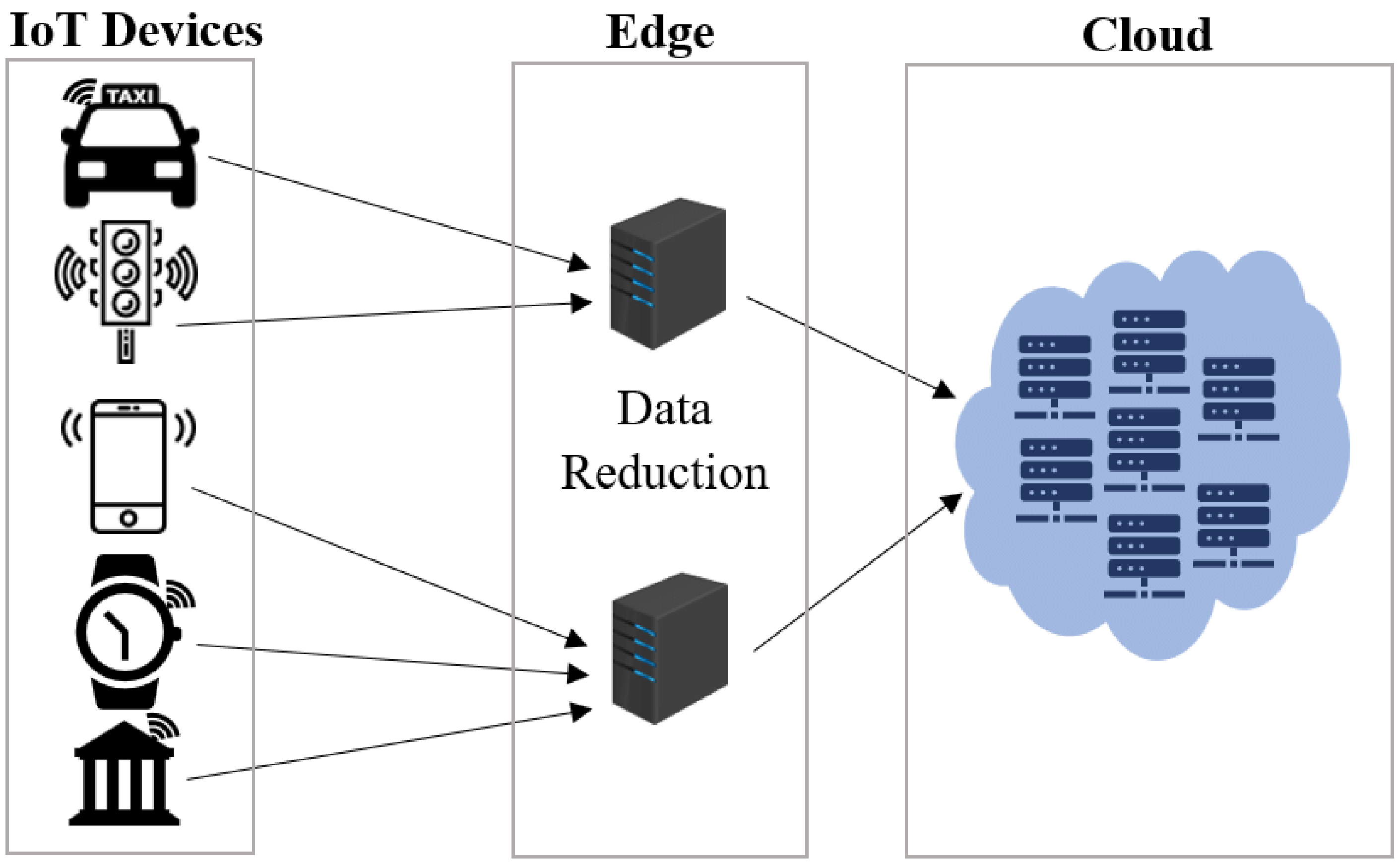
Data

26/07/2021

Goals

- To study Apache Qpid project
- To offer a presentation of its features
- To investigate its possible use in IoT scenarios in relation to edge and cloud computing
- To develop a prototype in order to show the lessons learned from this research

IoT scenarios



Edge and cloud computing

Apache Qpid

The Qpid project offers messaging APIs and message brokers for use in diverse applications as well as core libraries based on AMQP, the first open standard wire protocol for reliably sending and receiving messages.

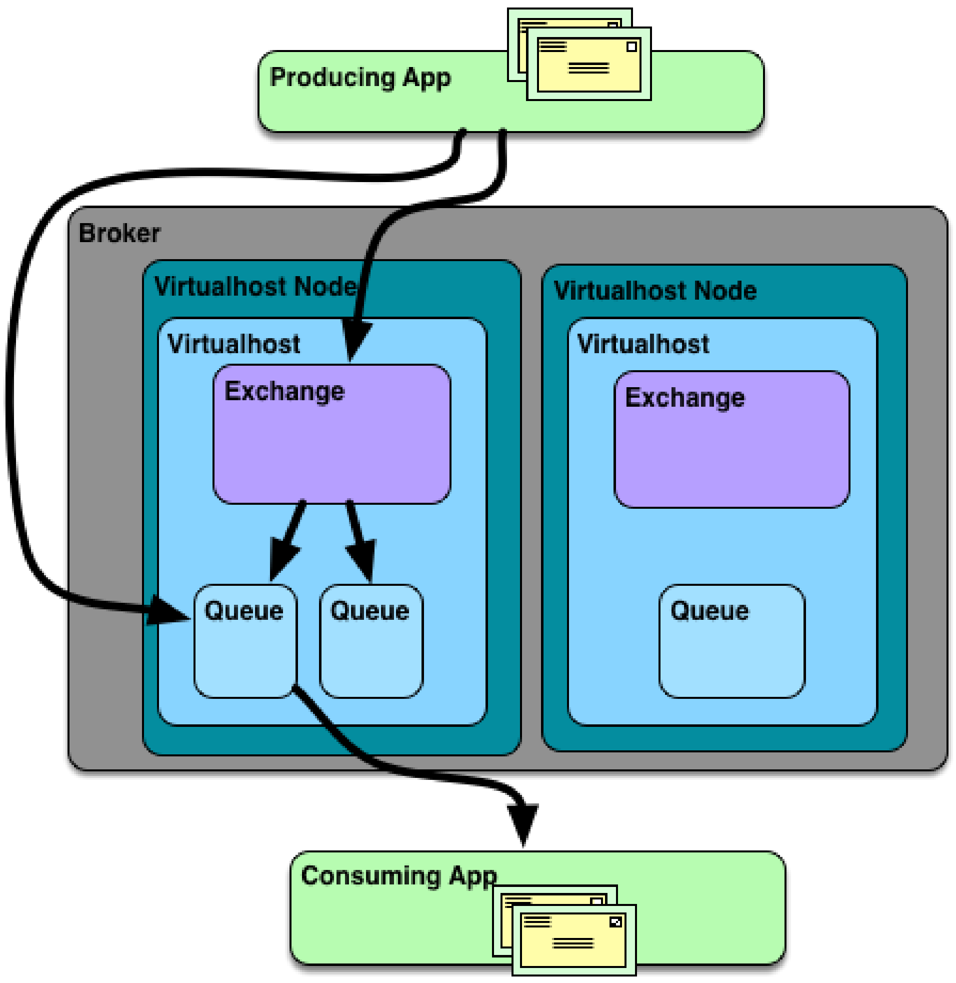
http://qpid.apache.org/index.html

Components

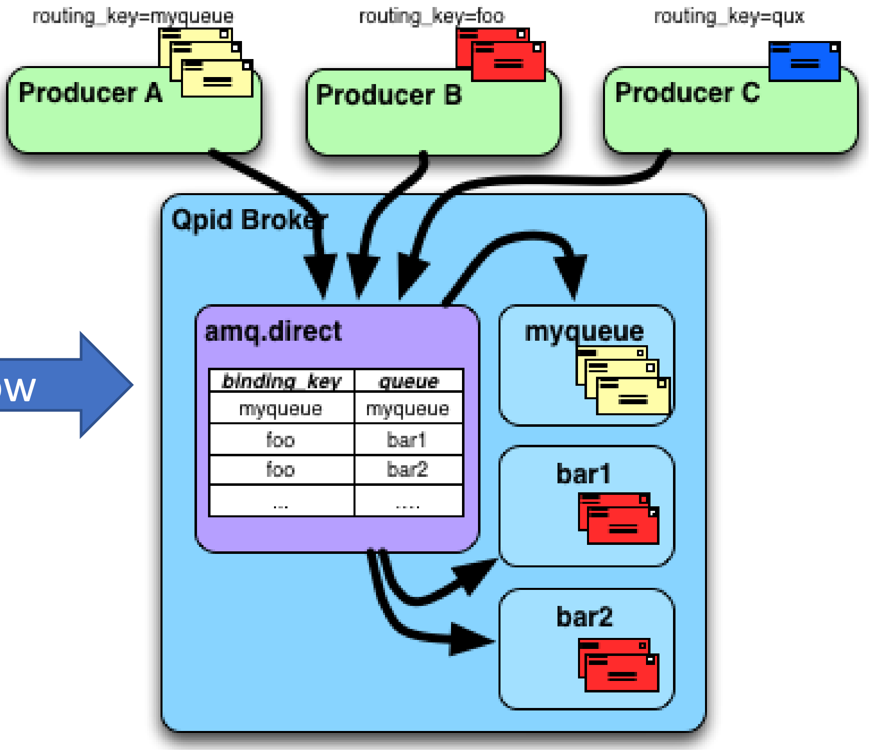
- **Messaging APIs**
  - Qpid Proton is a toolkit allowing any application to speak AMQP (used by other Qpid components to implement AMQP 1.0 protocol support)
  - Qpid JMS is an AMQP-fluent Java Message Service implementation
  - Qpid Messaging API is a connection-oriented messaging API that supports many languages
- **Messaging servers**
  - Broker-J is a pure-Java AMQP message broker
  - C++ broker is a native-code AMQP message broker
  - Dispatch router is an AMQP router for scalable messaging interconnect

http://qpid.apache.org/index.html

Broker-J

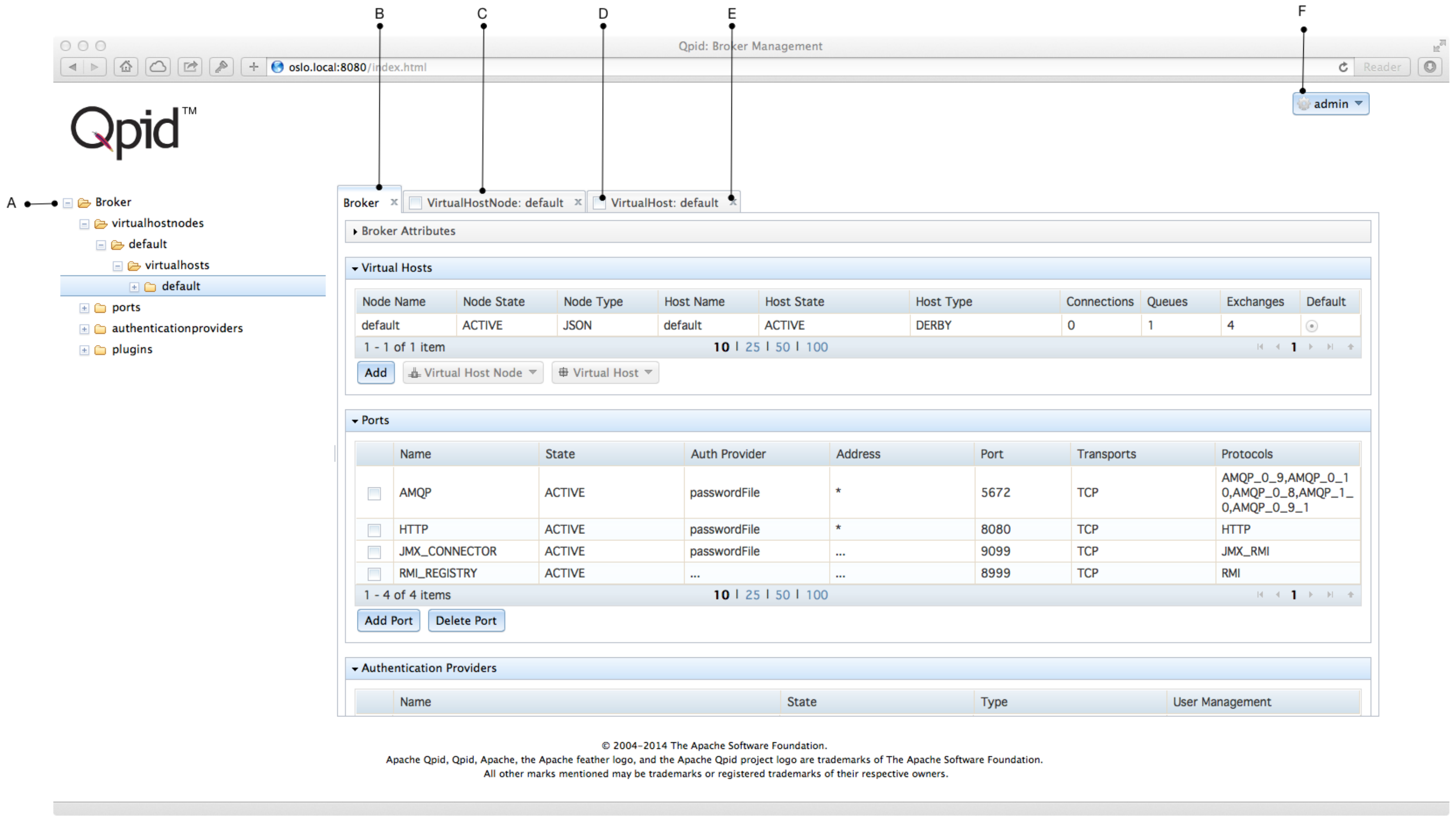


Message flow



binding_key	*queue*
myqueue	myqueue
foo	bar1
foo	bar2
...	....

Web Management Console



- Broker
  - virtualhostnodes
    - default
      - virtualhosts
        - default
  - ports
  - authenticationproviders
  - plugins

Broker × | VirtualHostNode: default × | VirtualHost: default ×

▸ Broker Attributes

▾ Virtual Hosts

Node Name	Node State	Node Type	Host Name	Host State	Host Type	Connections	Queues	Exchanges	Default
default	ACTIVE	JSON	default	ACTIVE	DERBY	0	1	4	⊙

1 - 1 of 1 item 10 | 25 | 50 | 100 1

Add Virtual Host Node Virtual Host

▾ Ports

	Name	State	Auth Provider	Address	Port	Transports	Protocols
☐	AMQP	ACTIVE	passwordFile	*	5672	TCP	AMQP_0_9,AMQP_0_10,AMQP_0_8,AMQP_1_0,AMQP_0_9_1
☐	HTTP	ACTIVE	passwordFile	*	8080	TCP	HTTP
☐	JMX_CONNECTOR	ACTIVE	passwordFile	...	9099	TCP	JMX_RMI
☐	RMI_REGISTRY	ACTIVE	...	...	8999	TCP	RMI

1 - 4 of 4 items 10 | 25 | 50 | 100 1

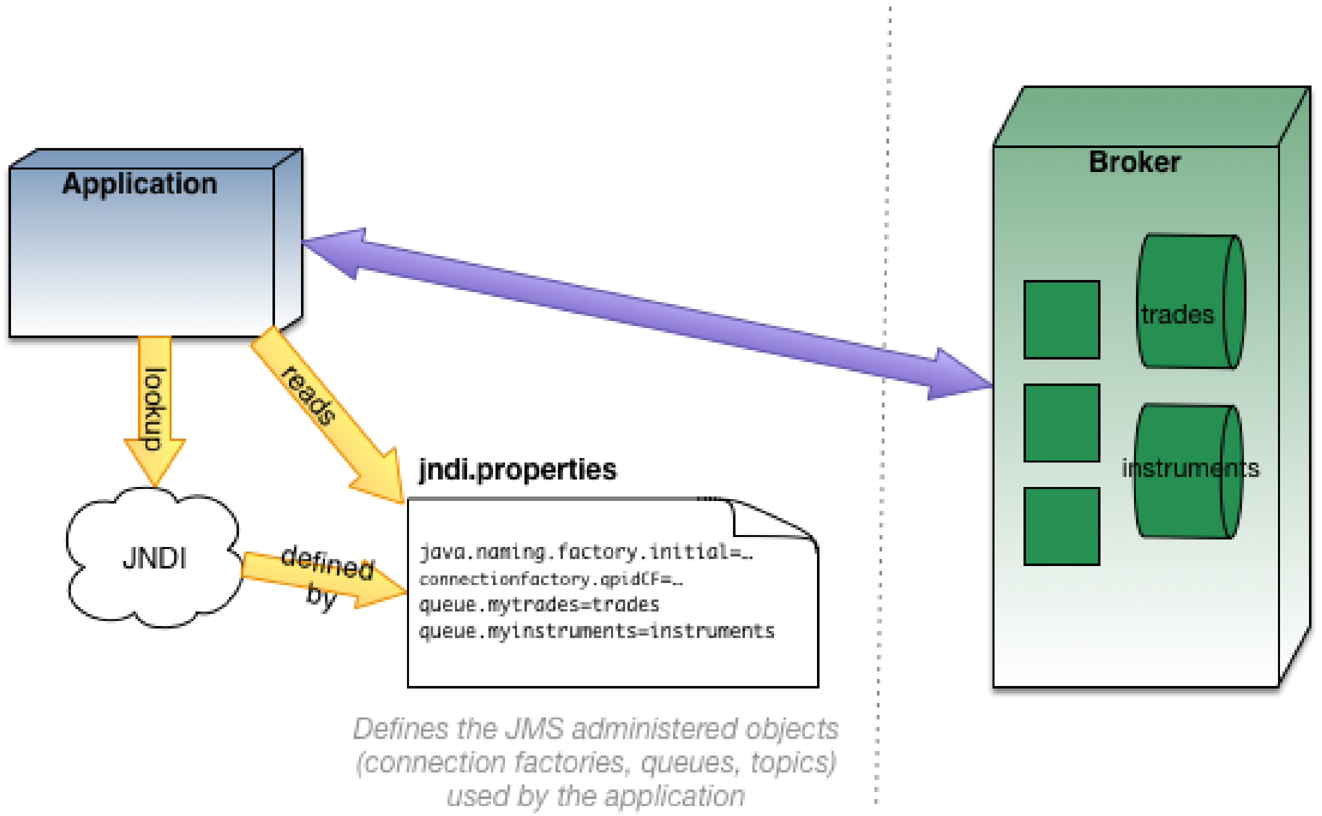
Add Port Delete Port

▾ Authentication Providers

Name	State	Type	User Management

© 2004-2014 The Apache Software Foundation.
Apache Qpid, Qpid, Apache, the Apache feather logo, and the Apache Qpid project logo are trademarks of The Apache Software Foundation.
All other marks mentioned may be trademarks or registered trademarks of their respective owners.

Prototype using Apache Qpid JMS



```
java.naming.factory.initial=…
connectionfactory.qpidCF=…
queue.mytrades=trades
queue.myinstruments=instruments
```

Defines the JMS administered objects (connection factories, queues, topics) used by the application

```
# Set the InitialContextFactory class to use

java.naming.factory.initial =
org.apache.qpid.jms.jndi.JmsInitialContextFactory


# Define the required ConnectionFactory instances

# connectionfactory.<JNDI-lookup-name> = <URI>

connectionfactory.myFactoryLookup =
amqp://localhost:5672


# queue.<JNDI-lookup-name> = <queue-name>

queue.myQueueLookup = queue
```

Initial configuration

https://github.com/alesszall/DA21

```
Context context = new InitialContext();

ConnectionFactory factory = (ConnectionFactory) context.lookup("myFactoryLookup");
Destination queue = (Destination) context.lookup("myQueueLookup");

Connection connection = factory.createConnection("guest","guest");
connection.start();

Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
connection.close();
```

Producer & Consumer

```
MessageConsumer messageConsumer = session.createConsumer(queue);
TextMessage requestMessage = (TextMessage) messageConsumer.receive();

MessageProducer messageProducer = session.createProducer(queue);
TextMessage requestMessage = session.createTextMessage(“data”);

messageProducer.send(requestMessage, DeliveryMode.NON_PERSISTENT,
        Message.DEFAULT_PRIORITY, Message.DEFAULT_TIME_TO_LIVE);
```

Apache Spark Streaming connector for AMQP

```
Function<Message, Option<String>> converter = new JavaAMQPBodyFunction<>();

JavaReceiverInputDStream<String>  receiveStream =
        AMQPUtils.createStream(jssc,
                "127.0.0.1",
                5672,
                Option.apply("guest"),
                Option.apply("guest"),
                "queue", converter, StorageLevel.MEMORY_ONLY());
```

Conclusion

- Apache Qpid components presentation
- Broker-J review and installation
- Qpid JMS API analysis
- Prototype implementation based on Qpid JMS and Broker-J (edge computing)
- Apache Spark Streaming connector for AMQP project integration (cloud computing)

Thanks For Your Attention