Message Queues @ DIRAC

Wojciech Krzemień (NCBJ)

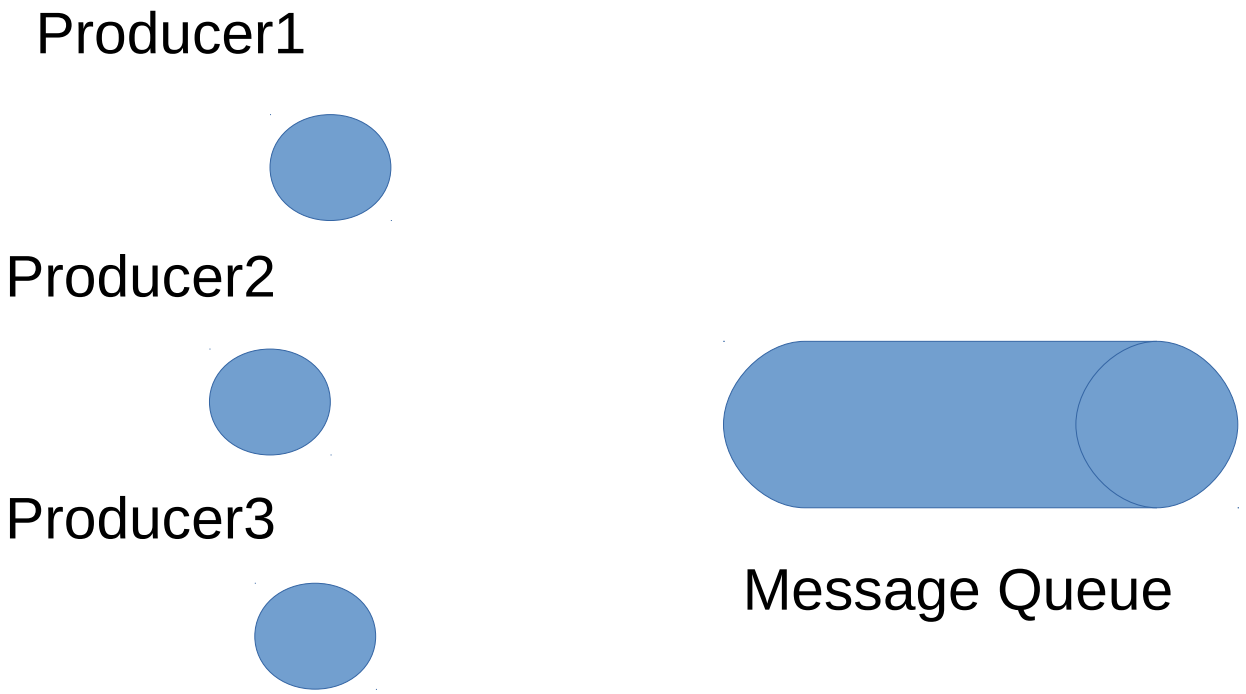
The 8th DIRAC Users Workshop
23.05 2018, Lyon

Outline

- Intro
- MQ in DIRAC
- Use cases:
  - Network monitoring with PerfSONAR
  - DIRAC service logging
  - DIRAC pilot logging
- Summary & Outlook

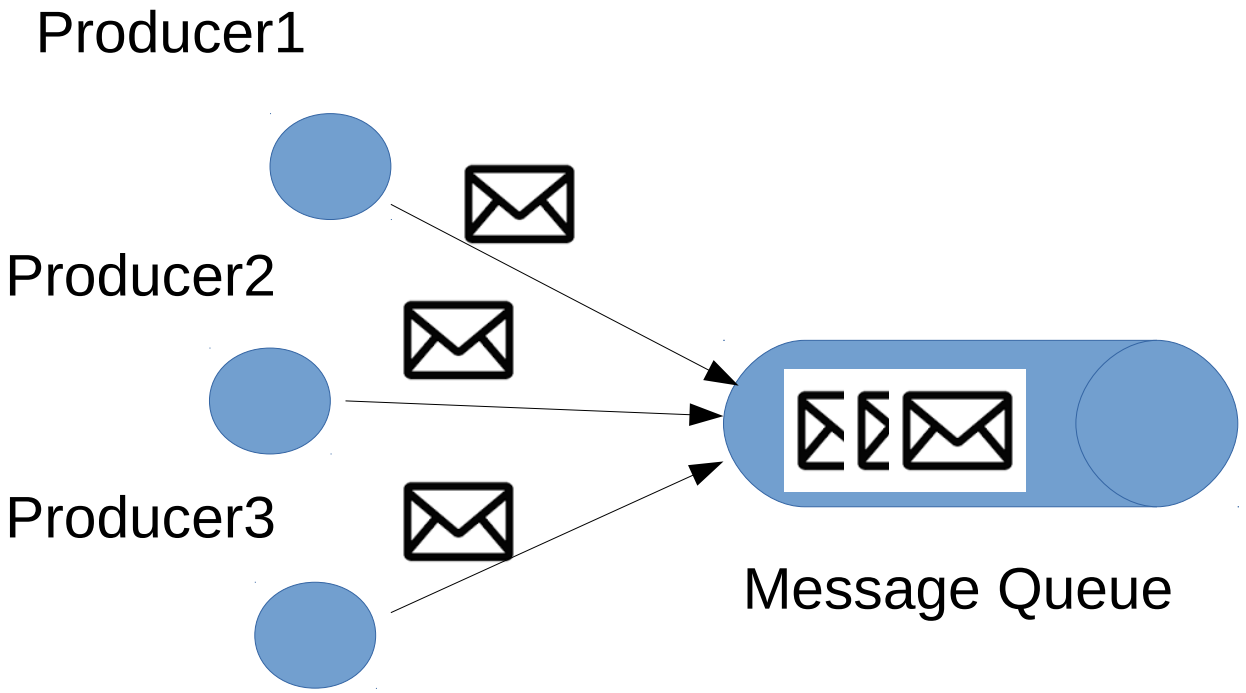
Message Queue architecture

- **Asynchronous** communication scheme
- Components are **decoupled** by the **queue** in which **messages** are stored



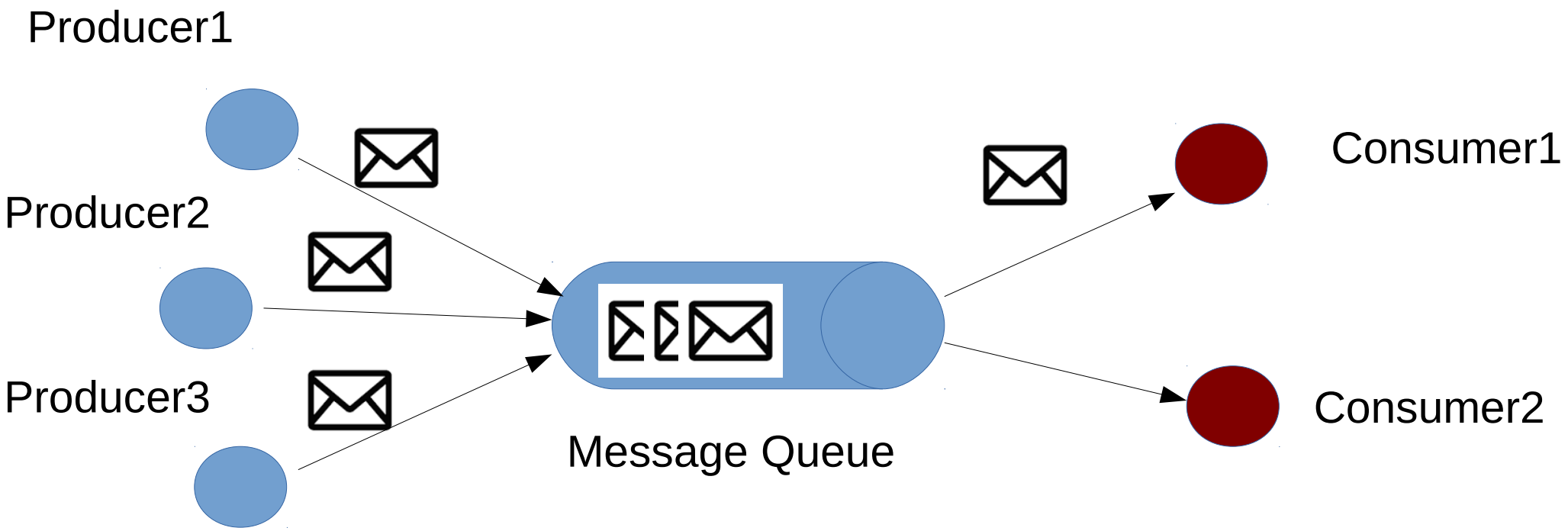
Message Queue architecture

- **Asynchronous** communication scheme
- Components are **decoupled** by the **queue** in which **messages** are stored



Message Queue architecture

- **Asynchronous** communication scheme
- Components are **decoupled** by the **queue** in which **messages** are stored



Message Queue architecture

Advantages:

- **Scalability**
- Performance
- Resilience
- **Connect heterogeneous environments**
- Redundancy
- Delivery Guarantee
- ….

Message Queue architecture

Advantages:

- **Scalability**
- Performance
- Resilience
- **Connect heterogeneous environments**
- Redundancy
- Delivery Guarantee
- ….

MQ communication protocols:

- Advanced Message Queueing Protocol (AMPQ)
- Streaming Text-Oriented Messaging Protocol (STOMP)
- others

Message Queue architecture

Advantages:

- **Scalability**
- Performance
- Resilience
- **Connect heterogeneous environments**
- Redundancy
- Delivery Guarantee
- ….

MQ communication protocols:

- Advanced Message Queueing Protocol (AMPQ)
- Streaming Text-Oriented Messaging Protocol (STOMP)
- others

Many open source MQ projects:

ActiveMQ

JBoss Messaging

Kafka

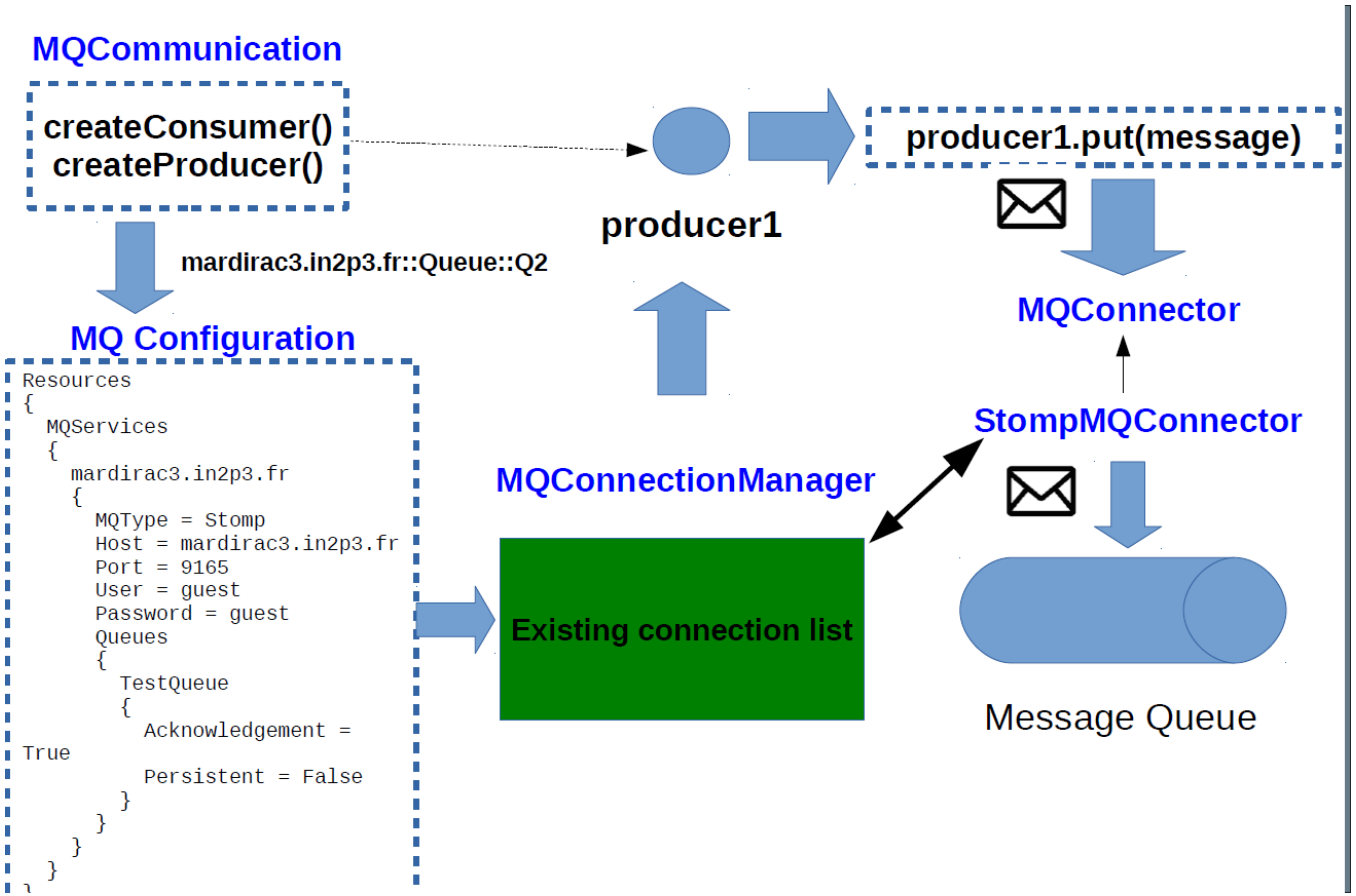
RabbitMQ

JORAM powered by ScalAgent

Qpid

Message Queues in DIRAC

- MQ can be used for sending messages between DIRAC components or to communicate with third-part services
- Generic MQ interface since **DIRAC v6r17**
- **STOMP** protocol handler implementation with **SSL** and **topics** support
- All MQ configuration are loaded from the Configuration Service



Code snippet

Create Producer and send a message to the queue

```
from DIRAC.Resources.MessageQueue.MQCommunication import createProducer

result = createProducer( "mardirac3.in2p3.fr::Queue::TestQueue" )
if result['OK']:
   producer = result['Value']
# Publish a message which is an arbitrary json structure
result = producer.put( message )
```

Code snippet

Create Producer and send a message to the queue

```
from DIRAC.Resources.MessageQueue.MQCommunication import createProducer

result = createProducer( "mardirac3.in2p3.fr::Queue::TestQueue" )
if result['OK']:
   producer = result['Value']
# Publish a message which is an arbitrary json structure
result = producer.put( message )
```

Create Consumer and read a message from the queue

```
from DIRAC.Resources.MessageQueue.MQCommunication import createConsumer

result = createConsumer( "mardirac3.in2p3.fr::Queue::TestQueue" )
if result['OK']:
   consumer = result['Value']
result = consumer.get( message )
if result['OK']:
  message = result['Value']
```

Code snippet

Create Producer and send a message to the queue

```
from DIRAC.Resources.MessageQueue.MQCommunication import createProducer

result = createProducer( "mardirac3.in2p3.fr::Queue::TestQueue" )
if result['OK']:
   producer = result['Value']
# Publish a message which is an arbitrary json structure
result = producer.put( message )
```

Create Consumer and use a callback function

to handle messages

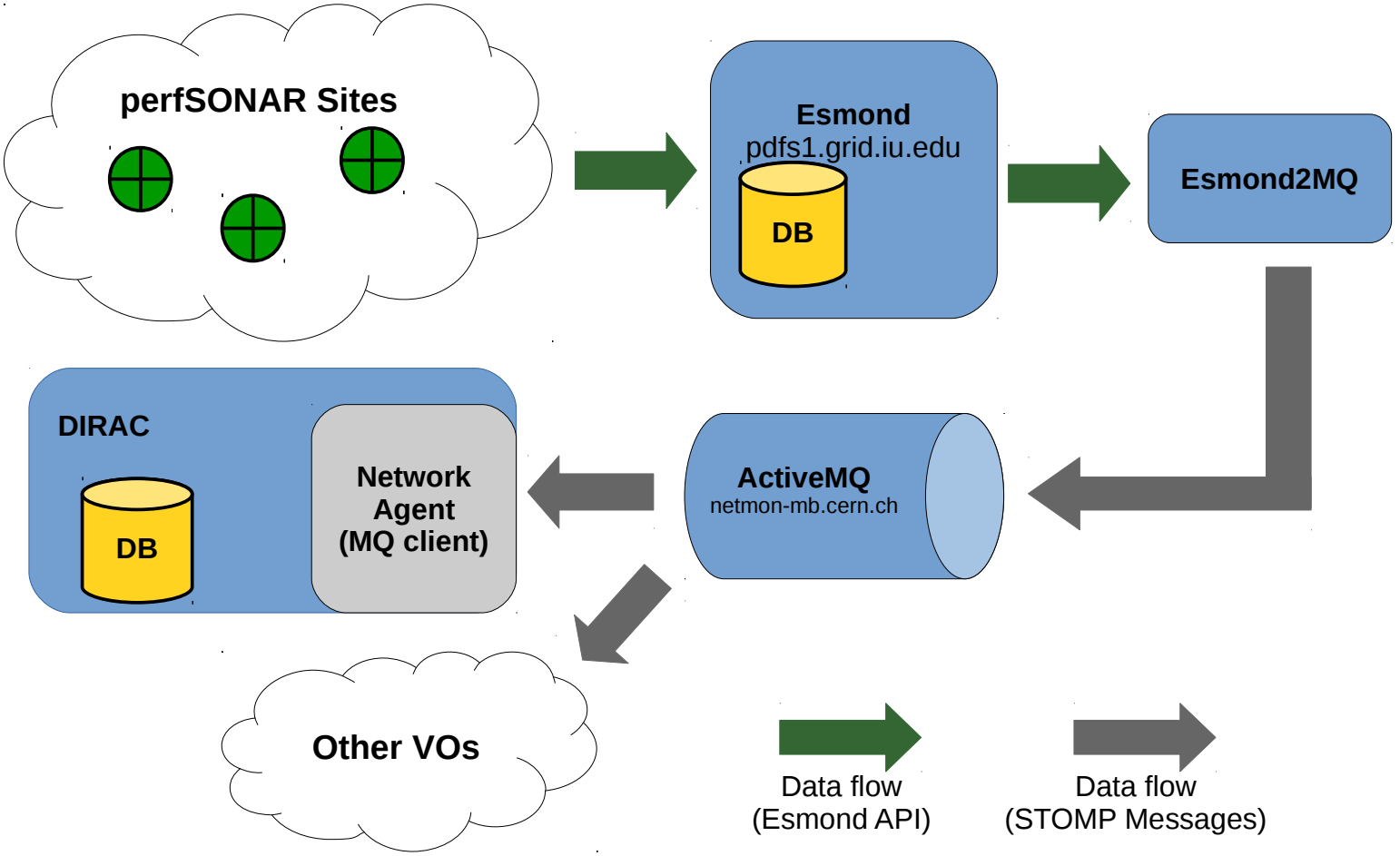
```
from DIRAC.Resources.MessageQueue.MQCommunication import createConsumer

def myCallback( headers, message ):
  <function implementation>

 result = createConsumer( "mardirac3.in2p3.fr::Queue::TestQueue", callback = myCallback )
 if result['OK']:
    consumer = result['Value']
```

Network monitoring with PerfSONAR

- Goals:
  - Monitoring of network activities on the network layer
  - Have precise information whether a problem is network related or have a different cause
  - Optimize data transfers (further plans)
- Solution
  - Cooperation with WLCG Network Throughput Working Group
  - Specialized modules in DIRAC to retrieve and present perfSONAR metrics (packet loss rate, one way delay)
  - Correlation of perfSONAR metrics with LHCb data operations

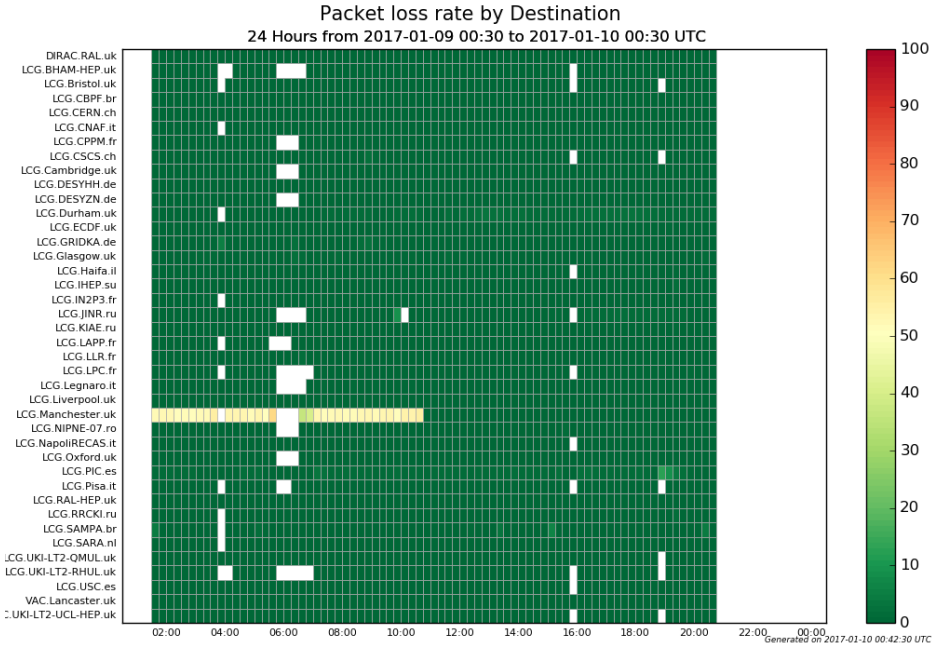
(Work by Henryk Giemza (NBCJ) & Federico Stagni (CERN))

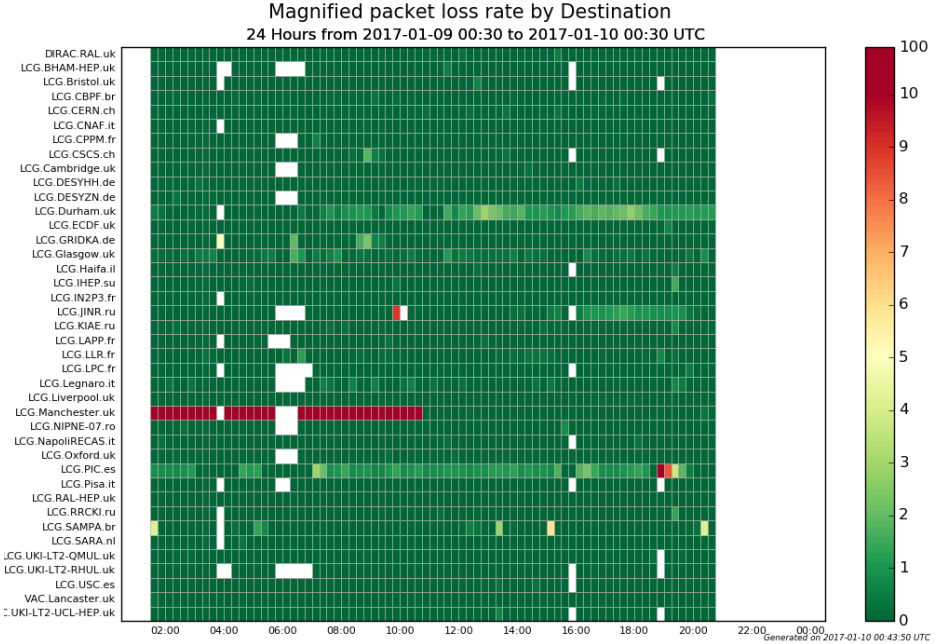
PerfSONAR-DIRAC bridge

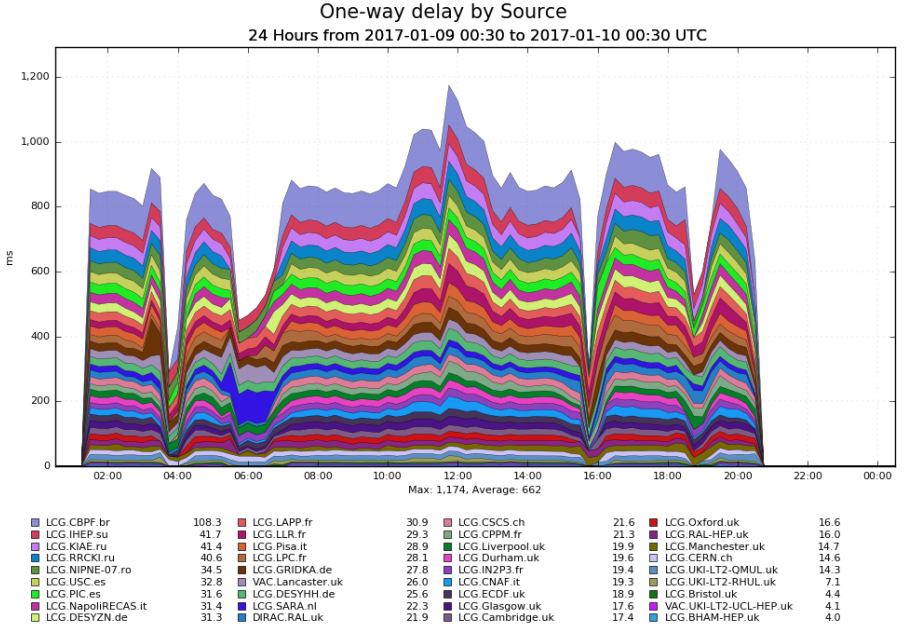


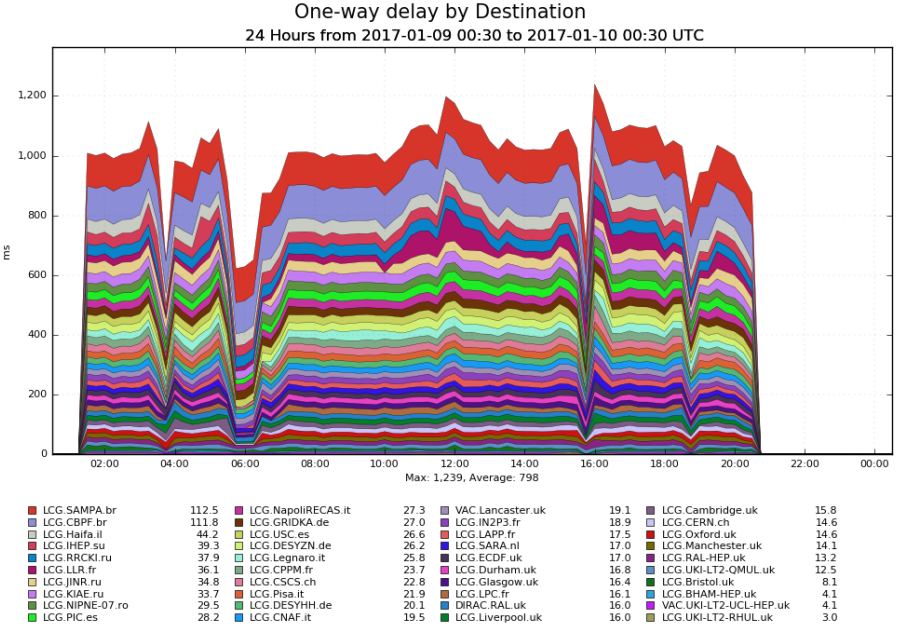
(work by Henryk Giemza (NBCJ) & Federico Stagni (CERN))

Metric visualization in DIRAC





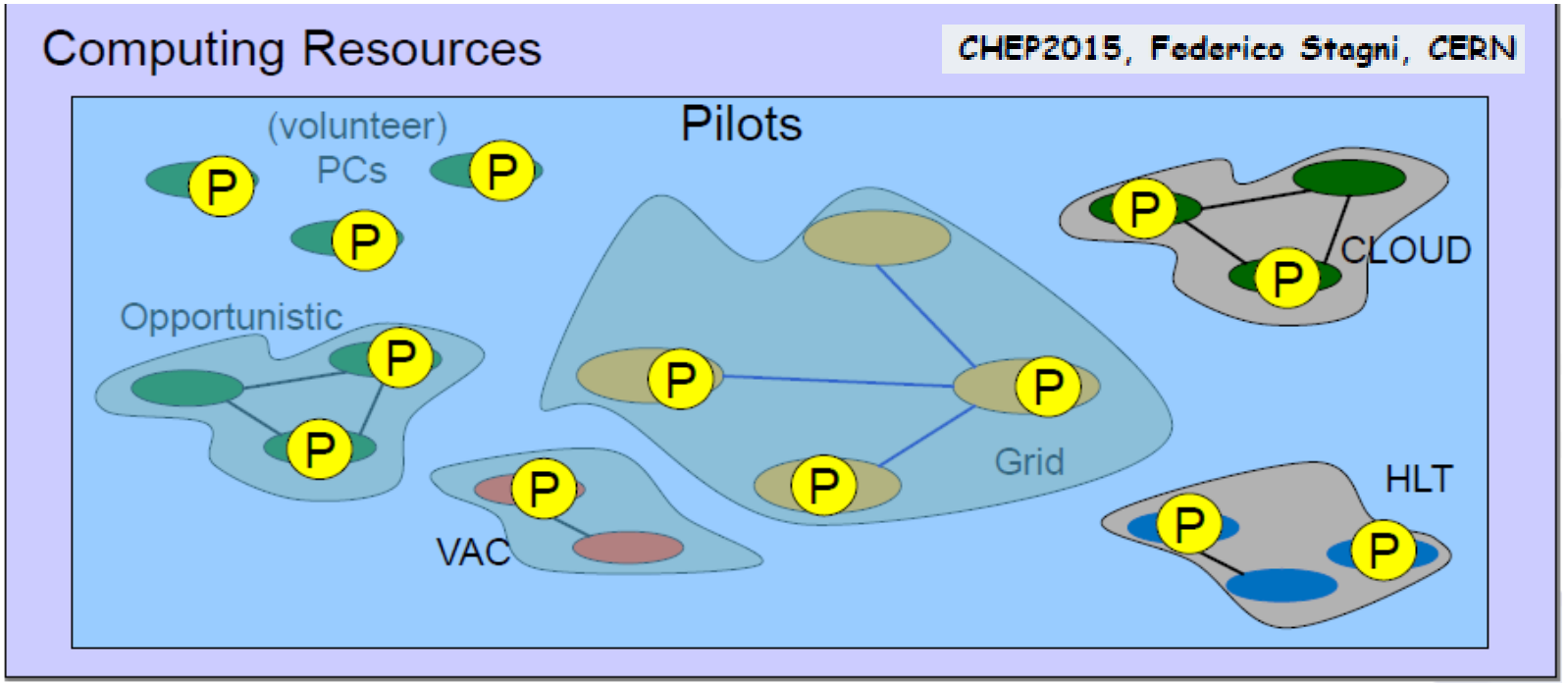




DIRAC service logging

- Goals:
  - All DIRAC service logs should be centralized
  - Server options include Logstash, Elasticsearch, CERN IT infrastructure, etc
- MQ seems nicely suitable for this work:
  - Agnostic to the endpoint type
  - Several endpoints possible
  - MQ acts as a buffer
- This extension should be relatively easy due to the work on DIRAC logger (work by A. Boyer & Ch. Haen)

Pilot Logging motivation



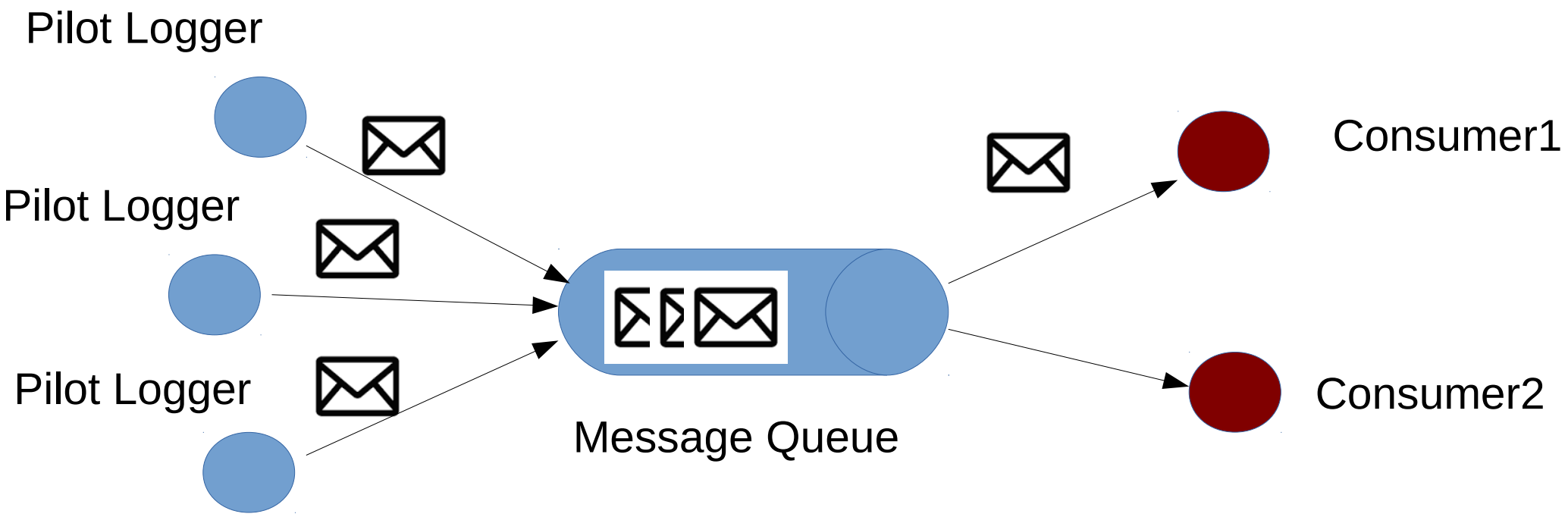
- We want info **before** DIRAC is installed
- Some logs are available - „WLCG” (from CREAM and ARCs CEs only),
- No automatized, general (and scalable) log system exists

"I've booted up" …
"I found the DIRAC pilot ok" ...
"I'm about to shutdown"...

"I installed DIRAC via SetupProject/dirac-install"...
"This machine has power of 11 HS06"....
"This machine is SLC6/CC7"...
"I matched a job" or
"I failed to match a job"... and so on.

Pilot Logger & MQ

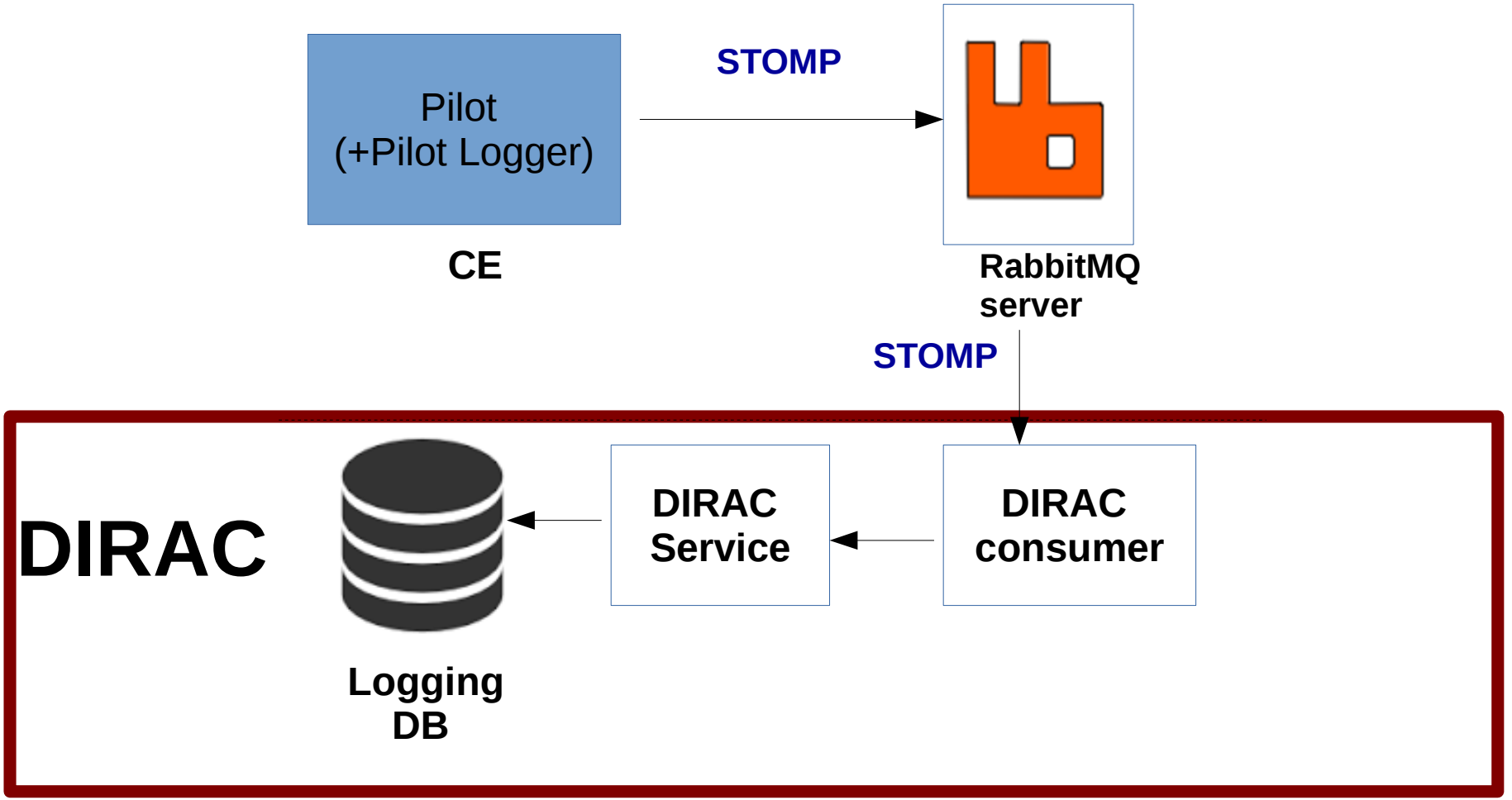
Producers



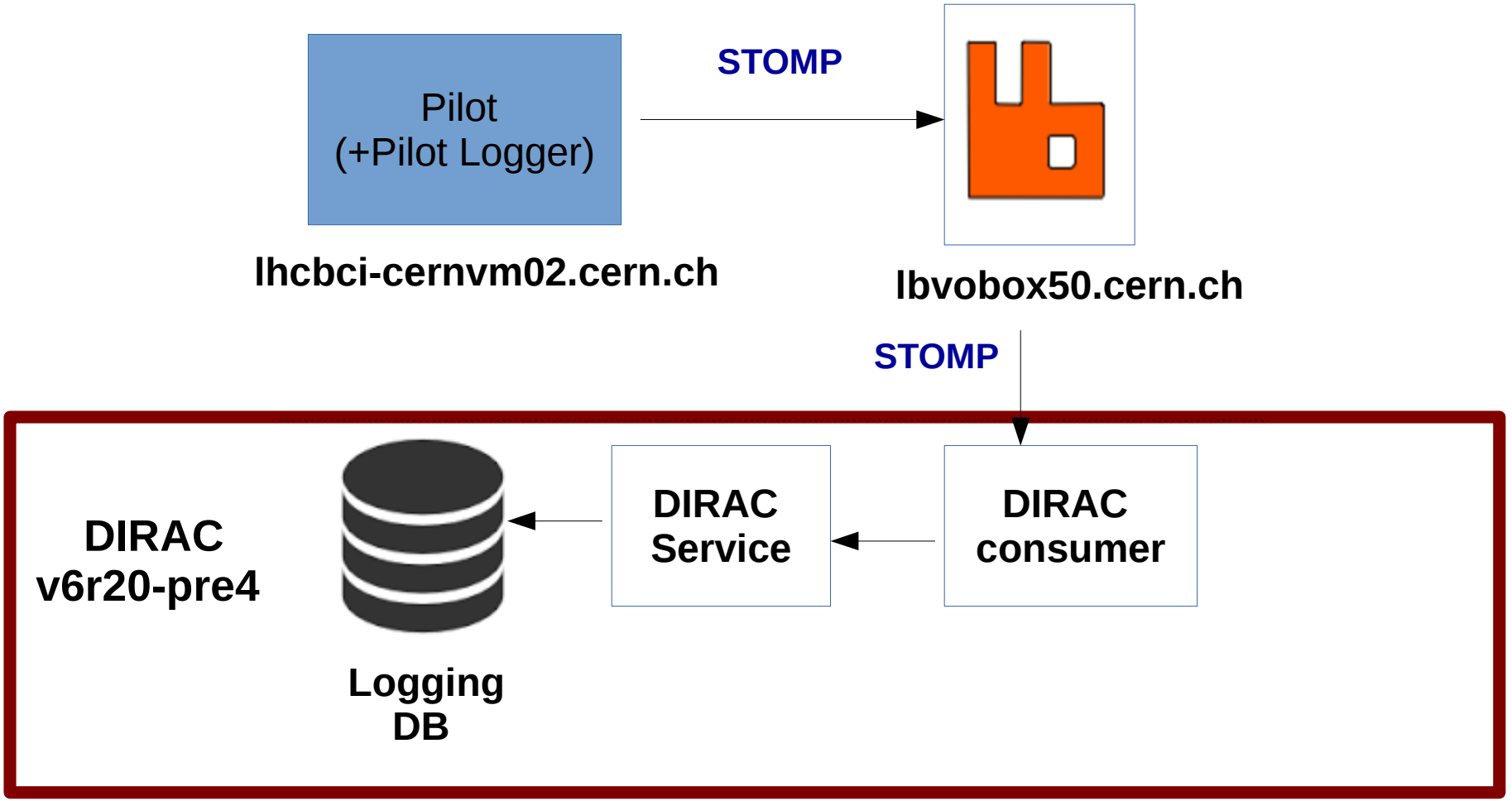
Idea: send logs to some dedicated MQ server

- Pilot Logger transparently added to Pilot repository
- Can be activated using the option

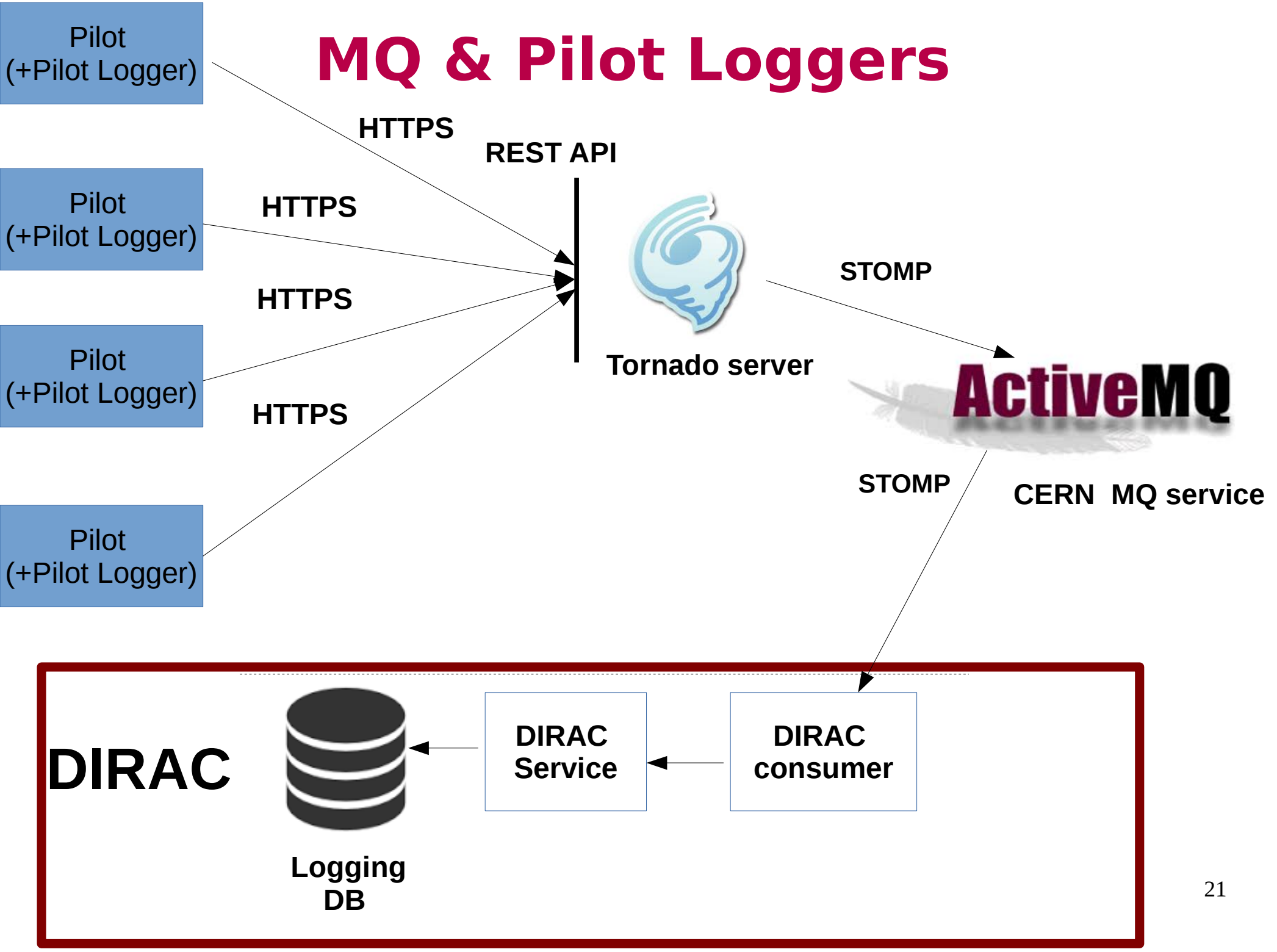
Pilot Logger & MQ



Pilot Logger & MQ



MQ & Pilot Loggers

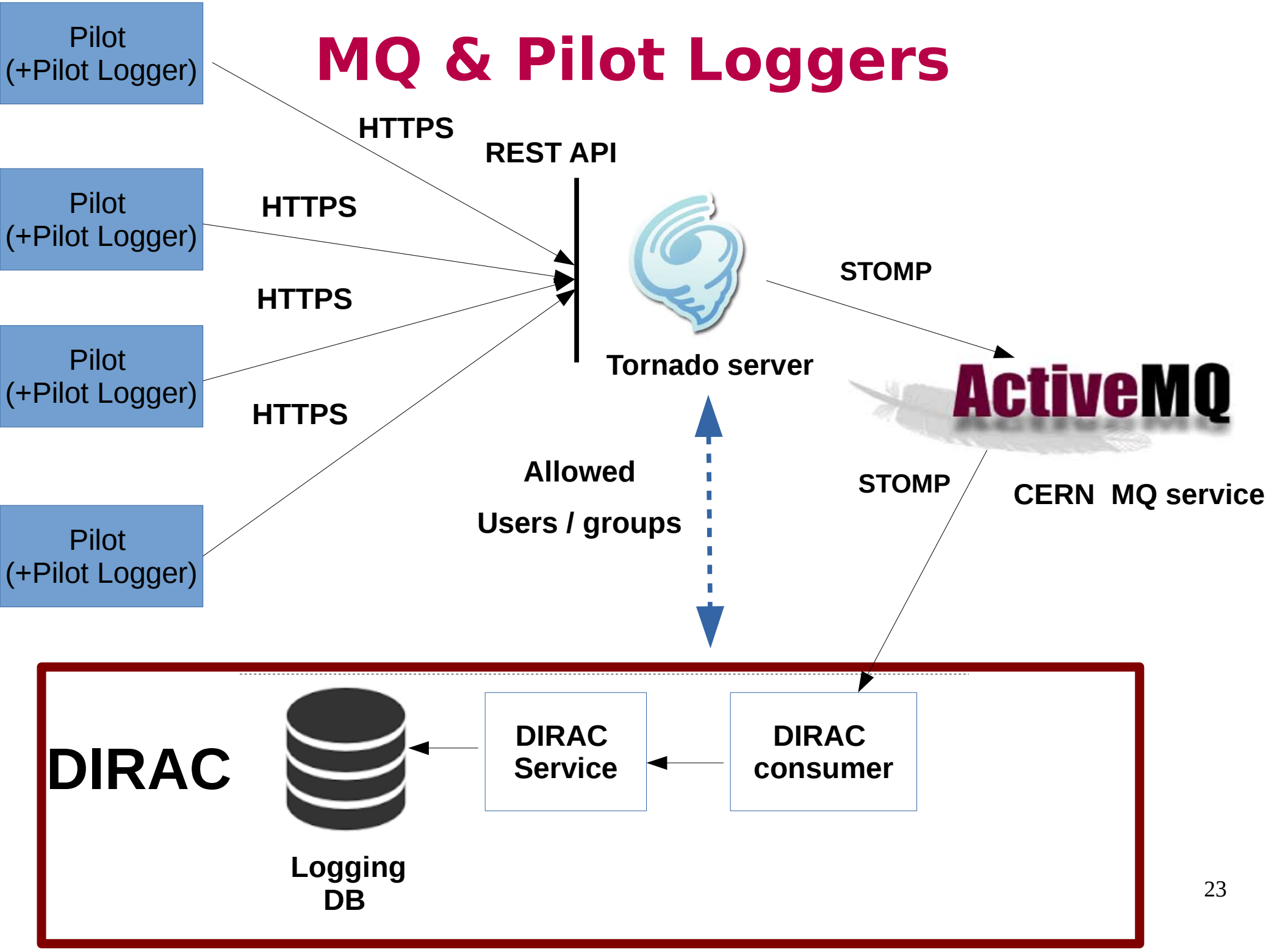


Tornado

- Python-based web framework
- Non-blocking, asynchronous transmission
- very good scalability performance (e.g. *Facebook*)
- Handles SSL certificates and RFC proxy certificates
- Already used by DIRAC extensions:
  - WebAppDIRAC
  - RestDIRAC

http://www.tornadoweb.org

MQ & Pilot Loggers



Summary

- Message Queueing as established communication scheme for scalable, distributed computing
- General MQ interface included since **DIRAC v6r17**
- Tests performed with RabbitMQ server with SSL support
- Ongoing works to incorporate the ActiveMQ CERN system (REST interface with Tornado server)

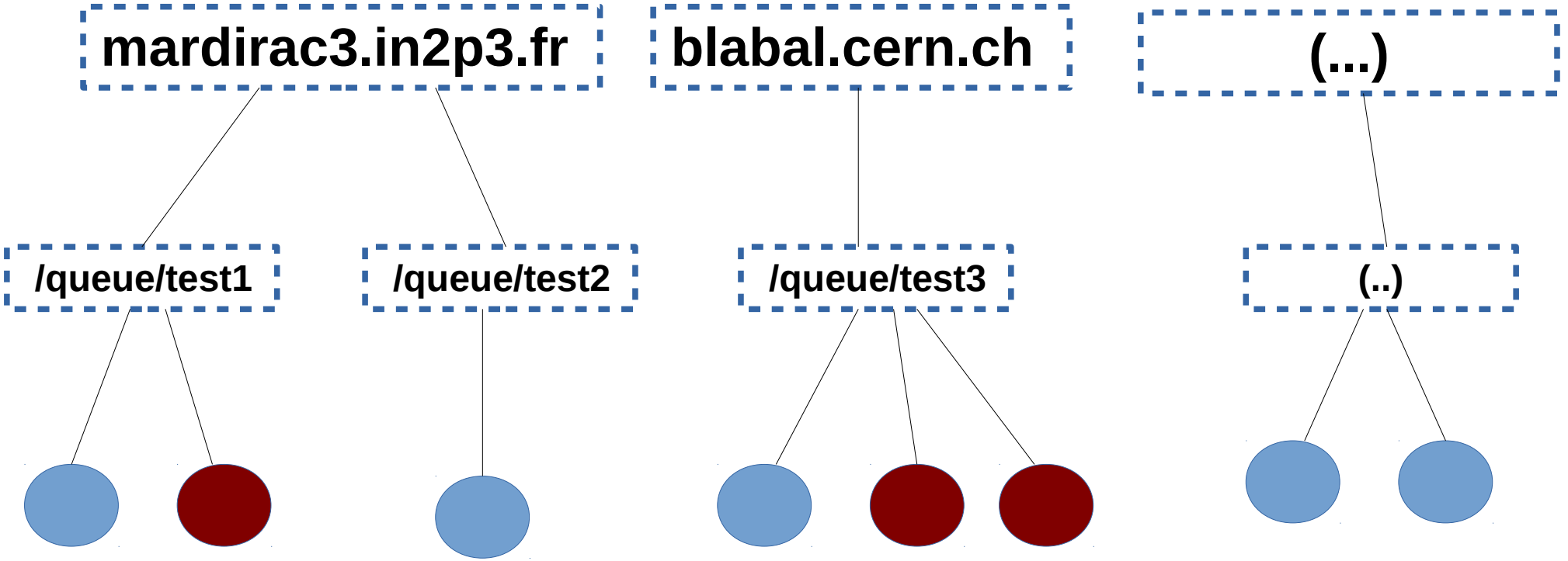
Use cases:

- MQ as a compontent in the **perfSONAR-DIRAC** bridge (**in production**)
- MQ as a failover mecanism for Elasticsearch used by MonitoringReporter (**in production**)
- MQ as a part of the **Pilot Logging** architecture (**work in progress**)
- MQ for service loggins (**future plans**)

Thank you

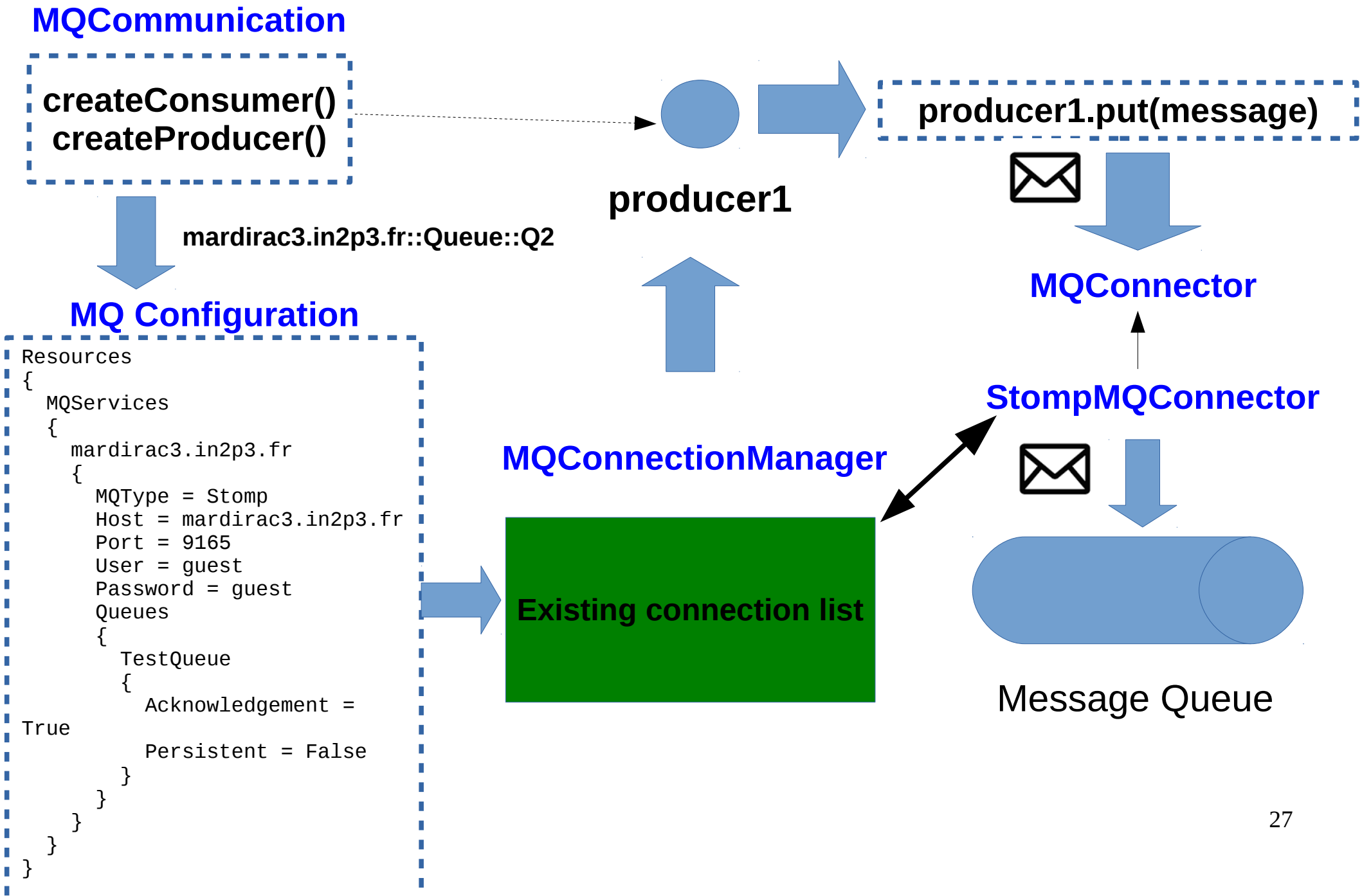
A bit of details II

MQConnectionManager



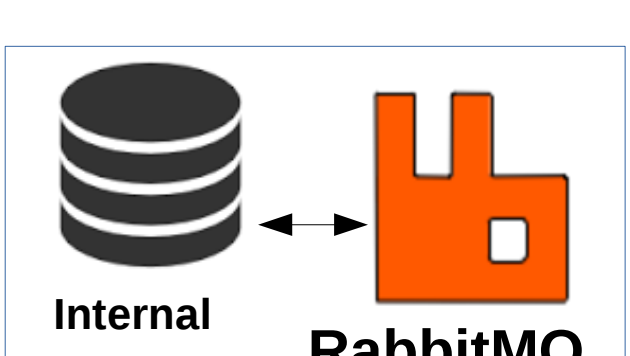
- Connections to MQ servers can be reused
- **MQConnectionManager** internally manages connections
- Thread-safety is assured

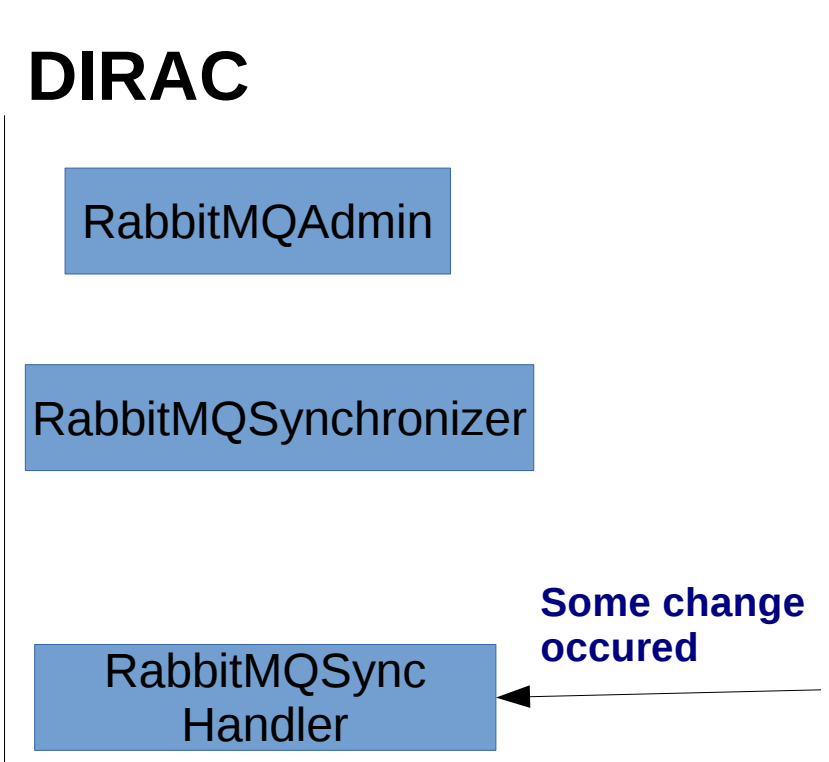
A bit of details



Sync between DIRAC CS and RabbitMQ

- RabbitMQ has internal User DB, with user loggins and authentication rights
- It should be synchronized with DIRAC Configuration Service (CS)
- RabbitMQAdmin module:
  - AddUser()
  - SetUserPermission()
  - DeleteUser()
  - ...





Sync between DIRAC CS and RabbitMQ

- RabbitMQ has internal User DB, with user loggins and authentication rights,
- It should be synchronized with DIRAC Configuration Service (CS)
- RabbitMQAdmin module:
  - AddUser()
  - SetUserPermission()
  - DeleteUser()
  - ...