Evaluation AMQP protocol over unstable and mobile networks

- Jorge Luzuriaga
- Miguel Perez
- Pablo Boronat
- Juan Carlos Cano
- Carlos Calafate
- Pietro Manzoni

IDCS’14
Calabria, Sept. 24th, 2014
Outline

- Goals
- Context & Background
- Testbed
  - Configuration
  - Schema
  - Parameters
- Results
- Conclusions & future work
Goals

- Assess the **performance and viability** of the AMQP messaging system with inexpensive hardware.

- Identify the **effects** of the network changes on message delivery.

- Obtain the **saturation boundary** values for the AMQP protocol with a specific hardware.
In order to communicate some **entities** over a wireless network:
- Handheld systems
- Remote servers
- Cloud resources
- Cloud services

It is necessary to find the best transfer protocol

**Solutions** that fit the requirements of mobile computing
- MOM standard* -> Message-Oriented Middleware
  - AMQP protocol

But AMQP implementations **have not been thoroughly tested** over unstable networks

---

Unstable networks

- Unpredictable environments
  - Channel interferences
  - Links
    - Frequently modified
    - Broken without control
- Nodes frequently join and leave the network
- Require a reasonable level of stability to execute applications

- Examples:
  - Mobile networks
  - Community networks

Image taken from: www.guifi.net
Advanced Message Queuing Protocol

Open internet protocol for business messaging

Defines
- The interaction between the entities (producer/consumer, broker)
- The representation of messages and commands

Facts
- On top of TCP
- Data agnostic
- Less verbose than traditional HTTP (Binary exchange)

Features
- Reliable delivery
- Store-and-forward transactional messaging
- Publish-and-subscribe notifications
RabbitMQ

- Is an open source messaging broker*
- Reference implementation of AMQP protocol
- Supported on multiple operative systems
- Several clients written in different programming languages
- Active community

* [www.rabbitmq.com/specification.html](http://www.rabbitmq.com/specification.html)
Focus on Access Point migration of the producer

Scenario details:
- Server used as both AMQP broker and consumer client
- Netbook used as producer client
- Ubuntu 12.04 O.S. running in computers
- OpenWRT O.S. running in routers
Generating Mobility

- Simulate the node mobility in an indoor scenario

- Queues
- Exchanges
- Channels
Simulate the node mobility in an indoor scenario

- Queues
- Exchanges
- Channels
- Simulate the node mobility in an indoor scenario

- Queues
- Exchanges
- Channels
Simulate the node mobility in an indoor scenario

Generating Mobility

- Queues
- Exchanges
- Channels
Generating Mobility 2

- Using scripts that shuts down/activates the router’s radio and disassociates/associates all clients devices
Synthetic load generator [amqperf]

- Command line tool
- Uses the RabbitMQ library

```
connection | start
tune         open
channel      open
exchange     declare

<table>
<thead>
<tr>
<th>producer</th>
<th>consumer</th>
</tr>
</thead>
<tbody>
<tr>
<td>while (test_duration){</td>
<td>while(true){</td>
</tr>
<tr>
<td>thread{ publish(message) }</td>
<td>read message with a timeout</td>
</tr>
<tr>
<td>sleep Period</td>
<td>if (message!=NULL)</td>
</tr>
<tr>
<td>if (exception)</td>
<td>treat_message</td>
</tr>
<tr>
<td>reOpenConnection()</td>
<td>else</td>
</tr>
<tr>
<td>}</td>
<td>endtest=true</td>
</tr>
<tr>
<td></td>
<td>if(endtest)</td>
</tr>
<tr>
<td></td>
<td>treat_test</td>
</tr>
</tbody>
</table>

channel | close
connection | close
```
Synthetic load generator [amqperf]

- Command line tool
- Uses the RabbitMQ library
- Generates workload for the message queuing system

```java
producer
while (test_duration){
    thread{ publish(message) }
    sleep Period
    if (exception)
        reOpenConnection()
}

consumer
while(true){
    read message with a timeout
    if (message!=NULL)
        treat_message
    else
        endtest=true
    if(endtest)
        treat_test
}
```

<table>
<thead>
<tr>
<th>connection</th>
<th>start tune open</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel</td>
<td>open</td>
</tr>
<tr>
<td>exchange</td>
<td>declare</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>channel</th>
<th>close</th>
</tr>
</thead>
<tbody>
<tr>
<td>connection</td>
<td>close</td>
</tr>
</tbody>
</table>

Synthetic load generator [amqperf]

- Command line tool
- Uses the RabbitMQ library
- Generates workload for the message queuing system
- Consumes messages and generates log files

```
while (test_duration) {
    thread { publish(message) }
    sleep Period
    if (exception)
        reOpenConnection()
}
```

```
while (true) {
    read message with a timeout
    if (message != NULL)
        treat_message
    else
        endtest = true
    if (endtest)
        treat_test
}
```

```
channel  close
connection close
```
Focus on AP migration of the producer

Each parameter combination were tested 100 times

A typical user application normally sends a few messages per second

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>size (bytes)</td>
<td>512, 1024, 3072, 6144</td>
</tr>
<tr>
<td>period (ms)</td>
<td>10, 100, 500, 1000</td>
</tr>
<tr>
<td>heartbeat (s)</td>
<td>default=580</td>
</tr>
<tr>
<td>duration (s)</td>
<td>20</td>
</tr>
<tr>
<td>transition instant (s)</td>
<td>1 (T=10), 5 (rest)</td>
</tr>
</tbody>
</table>
- Asynchrony between the different components of a distributed system
- Equation of message jitter inter-arrival times

\[ J_n = t'_n - t'_{n-1} - T \]

- \( t'_n \): Arrival time of message \( n \) (consumer)
- \( T \): Inter-message arrival (producer)
Behaviour during access point transition

- Maximum jitter is due to access point migration
- No message losses are observed

Period = 10 ms
 Behaviour during access point transition

- Maximum jitter is due to access point migration
- No message losses are observed

$\textit{hand-off} \approx 3.3 \text{ seconds}$

Bursts of messages
Behaviour during access point transition 2

- Maximum jitter is due to access point migration
- No message losses are observed

Period = 500 ms
Behaviour during access point transition 2

- Maximum jitter is due to access point migration
- No message losses are observed
- The number of messages with negative jitter can be approximated by the jitter divided by the period (4000/8 ≈ 8 messages)

Bursts of messages
Jitter is concentrated around 3.3 seconds

Cumulative Probability Distribution Function

- 512 Bytes
- 1 KBytes

Period = 10 ms
- Jitter is concentrated around 3.3 seconds

**Cumulative Probability Distribution Function**

- Cumulative distribution of maximum jitter - s512T10
- Cumulative distribution of maximum jitter - s1024T10

- 512 Bytes
- 1 KBytes

Period = 10 ms
- As a function a message size
- The sending period is more relevant than the size of messages
- As a function of the message production frequency
- Fine-grain precision is achieved by sending more messages per second (low periods)
- Combination of parameters have a non-linear influence

![Graph showing Jitter's Evolution 2](image-url)
Workload Boundary

- Packet loss results in non-delivery of messages
- Extreme working values at which message losses start
Packet loss results in non-delivery of messages

Extreme working values at which message losses starts

Bandwidth need by high definition video streaming $\approx 5$ Mbps
Conclusions & Future Work

- The testbed was conducted with inexpensive hardware to observe how the jitter is affected using AMQP exchanges on network changes.
- During the handover, the mean jitter values are between 3 and 4.7 seconds with peak values up to 7 seconds.
- AMQP messaging system is robust and guarantees message delivery without losses.
- Message losses appear in the producer side when the load generation is higher than its system buffer capacity.

future work

- Evaluate this protocol together with other solutions on more complex scenarios where the active TCP connection has to be reset.
Evaluation AMQP protocol over unstable and mobile networks

- Jorge Luzuriaga [ jorlu@upv.es ]
- Miguel Perez
- Pablo Boronat
- Juan Carlos Cano
- Carlos Calafate
- Pietro Manzoni

Grazie!