Testing AMQP protocol on unstable and mobile networks

Jorge E. Luzuriaga[1], Miguel Perez[2], Pablo Boronat[2],
Juan Carlos Cano[1], Carlos Calafate[1], and Pietro Manzoni[1]

[1] Department of Computer Engineering
Universitat Politècnica de València, Valencia, SPAIN
jorlu@upv.es,{jucano,calafate,pmanzoni}@disca.upv.es
[2] Universitat Jaume I, Castelló de la Plana, SPAIN
mperez@icc.uji.es,boronat@uji.es

Abstract. AMQP is a middleware protocol extensively used for exchanging messages in distributed applications. It provides an abstraction of the different participating parts and simplifies communication programming details. AMQP provides reliability features and alleviates the coordination of different entities of an application.
However, implementations of this protocol have not been well tested in the context of mobile or unstable networks. This paper is the starting point of an experimental evaluation of AMQP protocol in such kind of scenarios. Our goal is to identify the limits of applicability of this middleware, assessing its the capacity in terms of message losses, latencies or jitter, when wireless devices are interrupted and reconnected. This evaluation is of interest for the upcoming applications in which personal devices and vehicles will collaborate, forming part of large complex systems.

Keywords: client-server systems; performance evaluation; AMQP; advanced message queuing protocol; Mobile communication; Mobile Computing.

1 Introduction

With current expectations around the *Internet of Things* (IoT) there is a need to build and extend what is known as *intelligent spaces*. The idea behind these spaces is to connect computing elements such as sensors and actuators through a distributed network. The computing elements interact cooperatively in order to offer services to users. The network is usually a MANET or the mobile phone system (i.e. 3G, 4G) due to easy deployment. The massive use of smartphones or even On Board Units (OBU) in vehicles are favouring the development of these kinds of systems and applications.

The cooperation of the computing elements makes it easier to identify situations and then to provide data or to react when confronted with a set of stimulus. Intelligent spaces are highly dynamic due to the spontaneity with which elements connect or disconnect to the network. A flexible way to communicate the computing elements are message queuing middlewares such as the *Java Message Service* (JMS) or the emerging *Advanced Message Queuing Protocol* (AMQP).

AMQP is an application layer protocol which takes into account Message-oriented middleware (MOM) standards [1]. AMQP has been used in challenging applications, including Autonomous Computing [2], Cloud computing [3] or in security aspects related to the Internet of Things [4].

AMQP is designed to facilitate the dialogue among the components of a system, by making easy the exchange of messages independently of their underlying platforms. There are libraries available for most popular programming languages, and there are implementations for most of common operating systems. In addition, AMQP cares about security and confidentiality issues without affecting significantly the communication's performance.

In AMQP the messages are self-contained and data content in a message is opaque and immutable. The size of a messages in not limited. It can either support a 4 GByte message or a 4 KByte one. For message delivering, several possibilities are possible, as it can be *point-to-point*, *store-and-forward* or *publish-and-subscribe*. For instance, when a message is sent to an AMQP broker, actually it is sent to a queue, and after it is delivered to all subscribed customers to this queue as a push notification [1]. With AMQP the number of subscribers is unbounded.

This work is a starting point to test the behaviour of AMQP protocol over unstable networks. We call *unstable networks* those in which links can be frequently modified or broken without control. Examples of unstable networks could be mobile networks or wireless networks in urban environments, which suffer channel interferences, as occurs in community networks. Our goal is to determine whether AMQP provides satisfactory service, depending of the applications' load needs, in terms message size and communication rates. We detect the extreme working values at which message losses starts, as well as the effect of network changes on the messages' jitter.

In the present paper we introduce the first results in which we test the effect of a mobile producer which changes from one WiFi access point (AP) to another in the same IP network. We have developed a synthetic load generator which we call *amqperf*. This program sends messages with a sequence number to detect losses or messages delivered in different sending order. The size of messages and the frequency in which they are send by the producer can be modified. In the results we use a simple scenario with just one producer and one consumer.

The rest of the article is organized as follows: Section 2 presents a literature review related to the topic. In Section 3 there is a description of the methodology used in this work, showing how measurements have been done in order to be reproducible. The Section 4 presents the results and, finally, Section 5 provides some conclusions and the next steps to follow in this research line.

2 Related Work

There are several works in which the AMQP protocol is evaluated. In [5] the performance of AMQP is assessed using Infiniband and Gigabit Ethernet networks with Qpid as AMQP middleware. Five simple synthetic benchmarks modeled after the OSU Micro-benchmarks for MPI were used. They exercise the number of Publishers, the number of Consumers, and the Exchange type. Each benchmark measures performance for data capacity (the amount of raw data in MegaBytes per second), message rate (the number of discrete messages transmitted), and speed (average time one message takes to travel from the publisher to the consumer).

In [6] it is shown a way to evaluate the performance of AMQP by using an adapted version of the well-known *SPECjms2007* and *jms2009-PS* benchmarks. This would allow to compare AMQP with other messaging systems such as *JMS* (Java Message Service), in terms of performance, stability and scalability.

In [7] a performance comparison between AMQP and RESTful *web services* is presented. Three different tests are performed, which consist of several client applications sending messages during 30 minutes to the broker or the web server respectively; once the messages arrive to the server they are stored in a database. Then, the average number of messages per second that have been sent is compared to the total number of messages stored in the database. They conclude that, when the AMQP protocol is used to exchange messages, a larger number of messages per second is supported.

A study about MQTT, a "light weight" publish-subscribe based messaging protocol, is presented in [8]. The correlation between the end-to-end latency and loss of system messages is studied. Three different QoS levels with different sizes of payload (from 1 to 16 Kbytes) are tested on a real world scenario with both wired and wireless clients using 3G. They prove that there is a strong correlation between these two variables.

However, few studies are focused on the effectiveness of AMQP over unstable networks.

3 Methodology of the experiments

In the set of experiments we present in this paper we use a *producer* which, at a given frequency, sends messages of a prefixed size to an AMQP broker. The AMQP broker automatically creates a queue to the exchange of *fanout* type. Finally, a *consumer*, connected to the same broker is always ready to get messages. The producer is connected to a WiFi access point to reach the broker. The broker and the consumer are executed in the same computer. This scenario can be seen in Figure 1 and a picture can be seen in Figure 2.

The consumer records the sending (timestamped by the producer in each message) and reception time and the sequence number in a log file. There is not a strict synchronization between the producer and consumer clocks. When there are changes in the producer link, the regularity in the reception of messages is

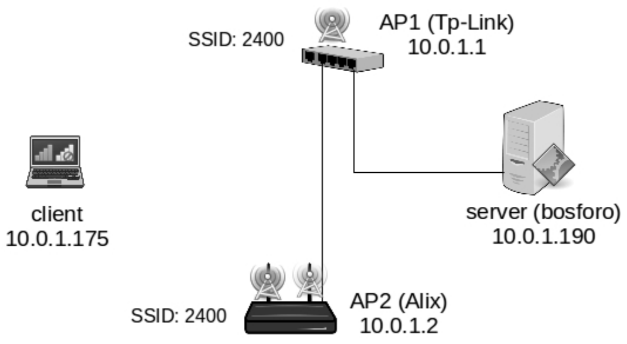


Fig. 1: Schema of the network.

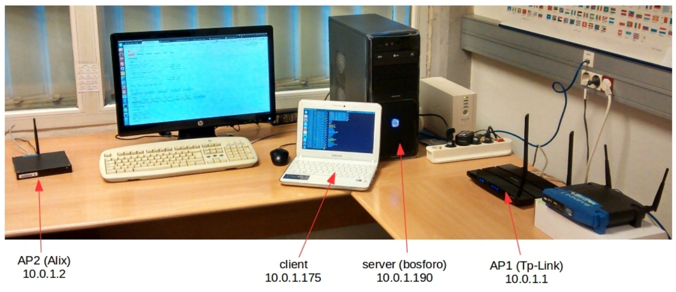
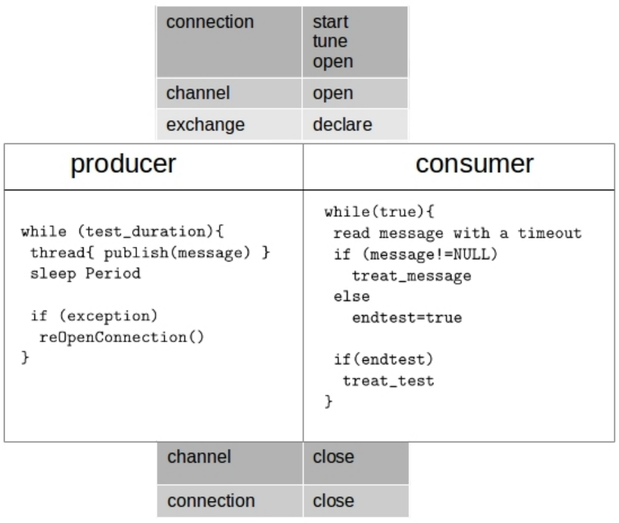


Fig. 2: A picture of the scenario.

affected. The inter-message times is modified, bursts of messages can be delivered to the consumer and even the sending order can be changed.

Message losses are produced when the hand-off time is too important with respect the load parameters of the test. Current implementations of AMQP use TCP connections to the broker in order to enhance reliability. If the producer connection is interrupted, the producer's AMQP client has to stock the messages until it can send it to the broker. If the sending buffer is full, messages will be lost. But this is not the only reason for losing sequence numbers. In the producer part, we create a thread to produce and send each message. Thus, in some extreme tests, the amount of threads exceeded the operating system limit. We have not tuned this parameter given that the workload in these circumstances is far from reasonable; for instance, bigger than high definition video streaming.

A testing application, which we call *amqperf* has been developed to generate a workload for the message queuing system in the part of the producer. Amqperf uses the RabbitMQ library [9], which is an AMQP implementation. An schema of amqperf can be seen in Figure 3.



```
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
}
```

Fig. 3: Schema of amqperf.

In the experiments, we perform tests of 20 seconds because we are interested in the AP migration of the producer. We checked whether there were message losses or if messages arrive out of order. The n^{th} message jitter of inter-arrival times is computed with the following equation:

$J_n = t'_n - t'_{n-1} - T$, where t'_n is the arrival time of message n to the consumer and T is the (fixed) period between messages produced by the producer. T is one of the variables fixed for each experiment. Note that with this formula we are not concerned by a possible asynchrony between the producer and the consumer, which are executed in different computers. A simplification of the times involved in the experiments can be seen in Figure 4.

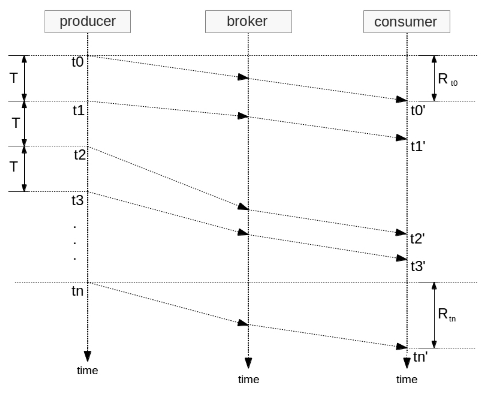


Fig. 4: Times involved in the experiments.

The values we used in the test were decided considering the bandwidth needed by high definition video streaming, which is about 5 Mbps. We obtain this value by transmitting 12500 byte messages every 0.02 seconds or 625000 byte messages each second. In any case, we have made some tests to detect the point at which messages start being lost in both cases: if there is a migration of access point is produced or without interruption in the WiFi network. These values are detailed in the following section.

The AMQP broker was created on a server with an AMD 8-core processor and 16Gb of RAM memory. The client had an Atom N450 processor and 1Gb of RAM memory. Both of them were running Ubuntu 12.04 operating system. For the wireless network we have used the OpenWRT operative system with Attitude Adjustment version on a Alix PC-Enginees (alix2d2) and a Tplink (TL-WDR3600) routers. And the test were run on a dedicated LAN with no other traffic.

4 Results

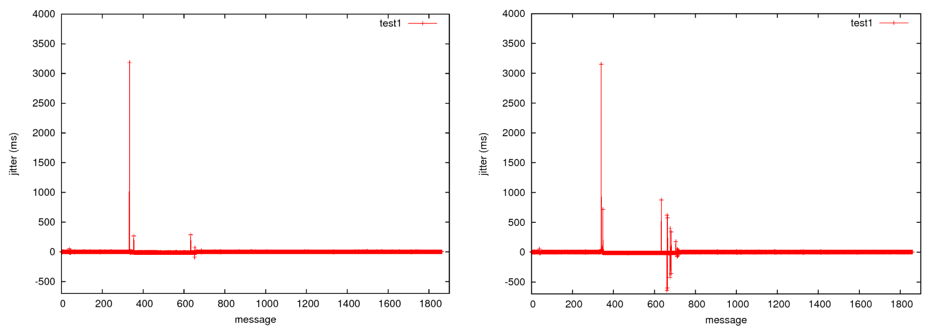
In this section we present the results of the first set of experiments in which a wireless producer migrates from access point but remaining in the same IP network. In these tests, the TCP connection between the producer and the AMQP broker is maintained. Each combination of message production frequency and message size is repeated 100 times and we analyse the distribution function of the maximum jitter for each one of the tests.

4.1 Behaviour during access point transition

For the experiments, we have used a completely dedicated network without external traffic. If there is not any interruption in the wireless link of the producer, a very limited jitter is observed for reasonable workloads, for instance, less than 5 Mbps.

In order to see a typical behaviour with an AP migration, we provide Figures 5 and 6. These figures shows the jitter for each message received by the consumer. In these figures, the positive peak corresponds to the hand-off, and the negative values are due to the reception of a burst of messages which the producer have retained during this communication's interruption.

In 5, the small oscillations after the handoff peak are produced because the messages in the interruption burst are delivered in the inverted order (like in a LIFO queue).



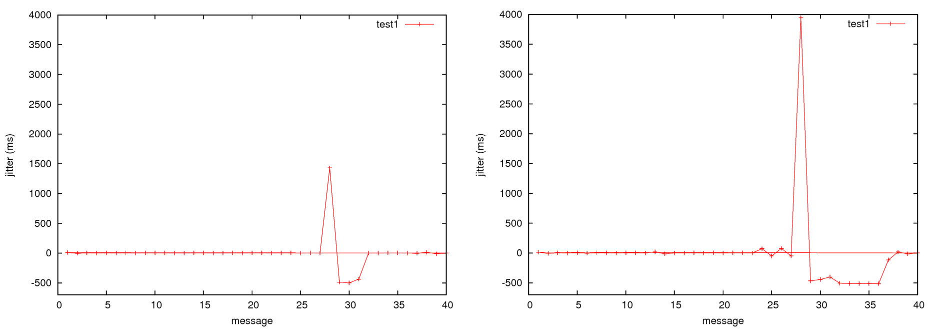
(a) 512 Bytes (b) 6 KBytes

Fig. 5: Behaviour of jitter with migration of access point. Both tests producing a message each 10 ms and with messages sizes: a) 512 Bytes and b) 6 KBytes.

As expected, the number of messages with negative jitter can be approximated by the peak positive jitter divided by the message producing period. For instance, in figure 6b, it is $4000/500 \approx 8$ messages.

4.2 Workload boundary

Without performing an exhaustive delimitation of the workloads producing message losses, we have made some tests to provide hints about the applicability of AMQP. Note that these "extreme" workloads can be dependent on the platform used, and even the configuration of these platforms. These experiments have



(a) 512 Bytes (b) 6 KBytes

Fig. 6: Behaviour of jitter with migration of access point. Both tests producing a message each 500 ms and with messages sizes:a) 512 Bytes and b) 6 KBytes.

been conducted without access point migration, in order to know the capacity of the system.

Figure 7 shows an approximation of the capacity of the system in terms of message size and production frequency. For loads above the red line, the system is saturated and not all produced messages arrive to consumers. The limits are around to 20 Mbps. This bandwidth is close to that we obtain with the *iperf* tool using the TCP test.

4.3 Jitter analysis

We analyse the jitter of the messages arriving to the consumer when the producer makes an AP migration using the maximum jitter's distribution function after repeating 100 times the same scenario (the same combination of messages production frequency and size). We know that the maximum jitter in our tests is due to access point migration given that the network has not external traffic and that the workloads used in these tests do not saturate the system (no message losses are observed).

In Figure 8 it can be seen the distribution function of the maximum jitter using a period of 10 ms for message production and two message sizes: a) 512 Bytes and b) 1024 Bytes. In both cases, the jitter is concentrated around 3 s.

In Table 1 it is shown the maximum, mean and standard deviation of the maximum jitter in 100 tests for different combinations of message size (from 0.5 KByte to 6 KByte) and message production periods (10, 100, 500 and 1000 ms).

To see the jitter evolution depending on each of the two parameters we use in the workload, we present the mean values in Table 1 in two ways: as a function of the message size (Figure 9a) or as function of the message production frequency (Figure 9b).

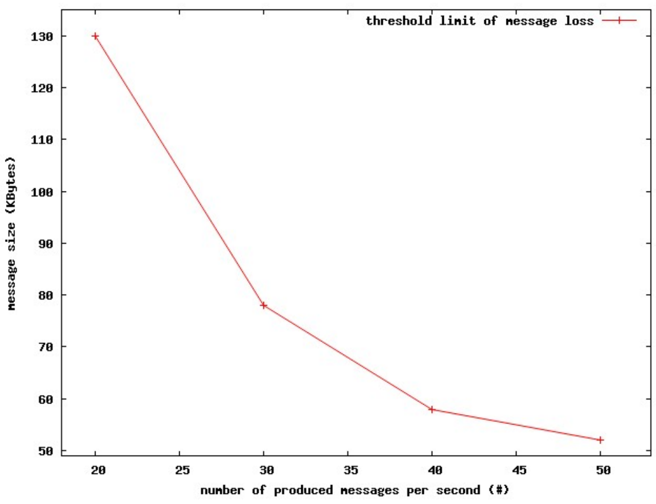
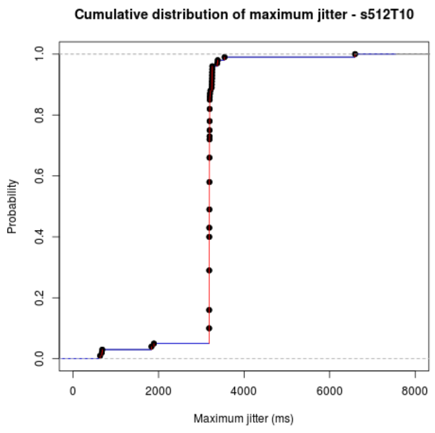
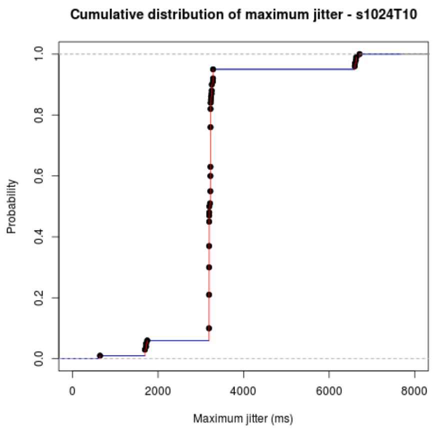


Fig. 7: Threshold limit of message losses for different message production frequency and message size.



(a) 512 Bytes

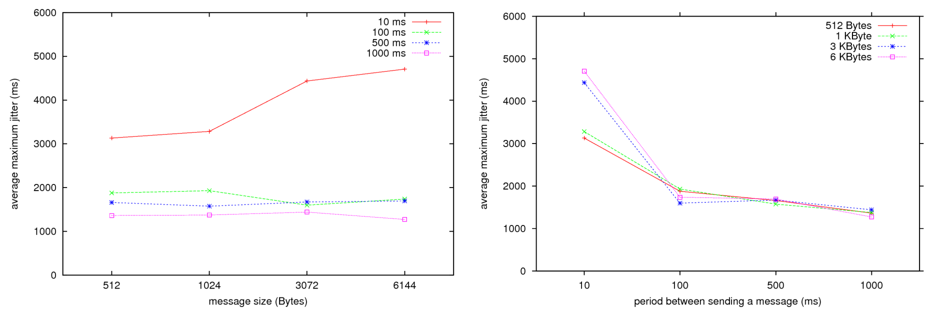


(b) 1024 Bytes

Fig. 8: Distribution function of the maximum jitter using a period of 10 ms for the production of messages and two message sizes: a) 512 Bytes and b) 1024 Bytes.

message size (Bytes)	period (ms)	max	mean	sta. dev.
512	10	6594	3132	587
	100	7361	1879	950
	500	7043	1660	1006
	1000	3367	1363	1010
1024	10	6711	3285	876
	100	4104	1931	818
	500	6346	1576	809
	1000	3404	1375	875
3072	10	6552	4437	1714
	100	3105	1599	587
	500	4045	1672	823
	1000	3701	1441	934
6144	10	6579	4708	1854
	100	6402	1739	786
	500	6426	1696	949
	1000	3915	1272	949

Table 1: Statistical values of maximum jitter distribution for messages sent with different size and period.



(a) (b)

Fig. 9: Evolution of maximum jitter as function of (a) message's size, or (b) message production period.

In Figure 9 it can be seen that, concerning the jitter, the sending period is more relevant than the size of messages. This is clearly shown in Figure 9(a) for the line corresponding to 10 ms.

Also the combination of both parameters seems to have an important and non linear influence, as it can be seen by the proximity of lines corresponding to message sizes 3 and 6 KBytes and the difference between the 1 and 3 KBytes in Figure 9 (b).

5 Conclusions and future work

In this paper we presented our first results concerning how jitter is affected when using AMQP in unstable networks. AMQP is a middleware protocol which facilitates the development of applications based on producer-consumer or publish-subscribe models and make them platform independent. We have checked a simple workload model of one producer and one consumer in the presence of access point migration in an extended wireless network (i.e. several access points conforming a same Service Set). We have observed that the messaging system is robust and guarantees message delivery without losses. The occurrences of message losses are found when the load is higher than the system buffer capacity in the producer side; but the transfer rate requirement for what is considered a heavy traffic load, such as high quality video streaming across wireless networks, is below the covered area under the curve of the threshold limit of message loss.

We can conclude that, in a simple and controlled scenario with roaming between two access points, we observe jitters between 3 and 4.7 seconds, with peaks of 7 seconds appearing only for high transmission rates (e.g., 100 messages per second) which is a considerable rate for monitoring systems running on a general purpose network. Also, using off-the-self inexpensive hardware, we have tested extreme workloads from which message losses are detected.

As a follow-up of this work, we are planning more complex scenarios in which a roaming producer switches between different IP networks and not only between access point, thereby causing the TCP connection to be reset. Also, a deeper analysis about the relation between jitter, message size and message production rate is needed in order to provide a good characterization which will help developers to decide whether protocols such as AMQP fit their requirements.

References

1. John O'Hara. Toward a Commodity Enterprise Middleware. In *Communications Magazine*. ACM, 2007.
2. S. Gusmeroli, S. Piccione, and D. Rotondi. IoT@Work automation middleware system design and architecture. *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 2012.
3. Foundation OpenStack. AMQP and Nova, 2014.
4. Corporation IMatix. Security and Robustness, 2014.

5. Hari Subramoni, Gregory Marsh, Sundeep Narravula, Ping Lai, and Dhabaleswar K. Panda. Design and evaluation of benchmarks for financial applications using advanced message queuing protocol (AMQP) over infiniband. *2008 Workshop on High Performance Computational Finance, WHPCF 2008*, 2008.
6. Stefan Appel, Kai Sachs, and Alejandro Buchmann. Towards benchmarking of amqp. In *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*, pages 99–100. ACM, 2010.
7. J.L. Fernandes, I.C. Lopes, J.J.P.C. Rodrigues, and S. Ullah. Performance evaluation of RESTful web services and AMQP protocol. In *IEEE ICUFN*, pages 810–815, 2013.
8. Shinho Lee, Hyeonwoo Kim, Dong Kweon Hong, and Hongtaek Ju. Correlation analysis of MQTT loss and delay according to QoS level. *International Conference on Information Networking*, pages 714–717, 2013.
9. Inc. Pivotal Software. Messaging that just works. 2014.