

Introduction to AMQP Messaging with RabbitMQ

Dmitriy Samovskiy, CohesiveFT



Why Messaging?

- Get data from point A to point B
- Decouple publishers and consumers
- Queueing for later delivery
- Asynchronous
- Load balancing and scalability

RabbitMQ

- RabbitMQ is an AMQP messaging broker
- Developed and maintained by Rabbit Technologies Ltd, www.rabbitmq.com
- Joint venture between Cohesive Flexible Technologies (www.cohesiveft.com) and LShift (www.lshift.net)
- Core development team in London, UK
- Rabbit is a part of AMQP Working Group

AMQP

- Advanced Message Queueing Protocol
- <http://www.amqp.org>
- Broadly applicable for enterprise
- Totally open
- Platform agnostic
- Interoperable
- Standard port is 5672/tcp
- List of brokers:
 - <http://jira.amqp.org/confluence/display/AMQP/AMQP+Products>

AMQP Working Group

Cisco Systems

Credit Suisse

Deutsche Börse Systems

Envoy Technologies

Goldman Sachs

iMatix

IONA

JPMorgan Chase

Novell

Rabbit Technologies

Red Hat

TWIST

WSO2

29West

Before AMQP...

- Start sending blobs from A to B with direct tcp
- Add queuing semantics (discard vs queue)
- Add serialization of metadata (int vs table)
- Add network abstraction (tcp vs multicast)
- Add auth and ACL
- Add virtual connections (channels)
- Add high availability
- Result would be very similar to AMQP !!!

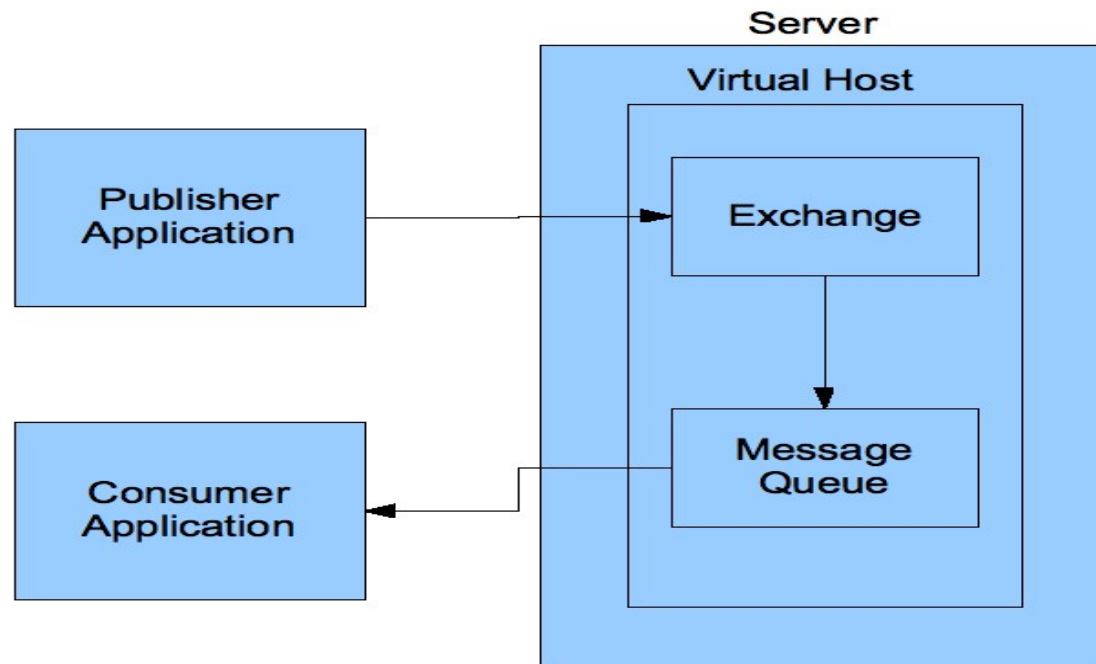
AMQP Protocol

- Network wire-level protocol
 - Defines how clients and brokers talk
 - Data serialization (framing), heartbeat
 - Hidden inside client libraries
- AMQP Model
 - Defines routing and storing of messages
 - Defines rules how these are wired together
 - Exported API

Network Wire-Level Protocol

- Information is organized into “frames”
- Independent threads of control within a single socket connection are called “channels”
- For each channel, frames run in sequence
- Each frame consists of header (type, channel id, payload size), payload, frame end packet
- Frames can be protocol methods (commands), structured content (message headers), data

AMQP Model



Virtual Hosts

- Created for administrative purposes
- Access control
- Each connection (and all channels inside) must be associated with a single virtual host
- Each virtual host comprises its own name space, a set of exchanges, message queues and all associated objects

Exchange

- A message routing agent
- Can be durable – lasts till explicitly deleted
- Can be temporary – lasts till server shuts down
- Can be auto-deleted – lasts till no longer used
- There are several types of exchanges, each implements a particular algorithm
- Each message is delivered to each qualifying queue
- “Binding” - a link between queue and exchange

Direct Exchange Type

- Uses string as routing key
- Queue binds to exchange with key K
- Publisher sends message with key R
- Message is passed to this queue if $K=R$
- Direct exchange named “amq.direct” is always pre-created in each virtual host

Fanout Exchange Type

- No routing key
- What goes in must go out
- Can be used for load balancing

Topic Exchange Type

- Uses pattern as routing key (“a.b.c.d”)
- Queues can use wildcard characters in binding
- * matches a single word, # - zero or more words
- “amq.topic” is pre-created in each vhost
- *.stock.# matches usd.stock and eur.stock.db but not stock.nasdaq

Message Queue

- named “weak FIFO” buffer
- FIFO is guaranteed only with 1 consumer
- Can be durable, temporary (private to 1 consumer) or auto-deleted
- A message routed to a queue is never sent to more than one client unless it is being resent after failure or rejection
- You can get server to auto generate and assign queue name for your queue – this is usually done for private queues

One more time...

- Each message received by an exchange will be delivered to each qualifying (matching) queue
- A message routed to a queue is never sent to more than one client unless it is being resent after failure or rejection

Messages

- Message is the atomic unit of processing
- Can be persistent (delivery guarantee in case network failure or server crash)
- Can have a priority level (not yet implemented in RabbitMQ broker)

Message Content

- Messages carry content (header + body)
- Content body is opaque block of binary data
- Broker never modifies content body
- AMQP defines several “content classes,” each with specific syntax (which headers can be used) and semantics (which methods are available for such messages)

Basic Content Type

- Implements regular messaging model
- `basic.consume` – start a queue consumer
- `basic.cancel` – cancel a consumer
- `basic.publish` – publish a message
- `basic.ack` – acknowledge message(s)
- `basic.reject` – reject a message
- `basic.get` – get message (synchronous)

AMQP Specification XML

- Details of each of these methods can be found in AMQP Spec
- Spec defines the protocol (like RFC)
- Spec is meant to be parsed to generate AMQP library code (XML)
- <class> section for every class: connection, channel, access, exchange, queue, basic, file, stream and others
- Look for <method> nodes inside each class

RabbitMQ



Open Source Enterprise Messaging

[News](#) [Download](#) [Documentation](#) [Examples](#) [Services](#) [FAQ](#)

RabbitMQ is an implementation of [AMQP](#), the emerging standard for high performance enterprise messaging.

Features

- ✧ A complete, [conformant](#) and [interoperable](#) implementation of the published AMQP specification
- ✧ Based on a [proven platform](#), offering exceptionally high reliability, availability and scalability
- ✧ Good throughput and latency performance that is predictable and consistent
- ✧ Compact, easily maintainable code base, for rapid customisation and hot deployment
- ✧ Extensive facilities for management, monitoring, control and debugging
- ✧ Licensed under the open source [Mozilla Public License](#)

News



- ✧ The RabbitMQ team is pleased to announce the release of the RabbitMQ .Net/C# AMQP client library 1.4.0.

This release has beta status and focuses on the following areas: bug fixes for a number of race conditions, improved shutdown protocol, bug fix for usage of read timeouts in .Net sockets.

Further details are available [here](#).

Distribution

- ✧ RabbitMQ server, written on top of the widely-used [Open Telecom Platform](#)
- ✧ RabbitMQ [Java client](#)
- ✧ RabbitMQ [.NET/C# client](#), with support for WCF
- ✧ [Experimental bindings](#) supporting HTTP, STOMP, SMTP, POP3, ...
- ✧ Platform-neutral distribution, plus platform-specific packages and bundles for easy installation
- ✧ Several user-contributed packages that extend the core RabbitMQ functionality
- ✧ Extensive [documentation](#), several [demos and examples](#), and a functional/performance test suite
- ✧ [Download Now!](#)

RabbitMQ

- RabbitMQ is a broker written in Erlang
- RabbitMQ team also provides Java and .NET clients
- Implements AMQP 0-8 spec
- Experimental products include AMQP-over-HTTP + Javascript libraries, Erlang client, gateway for STOMP clients, XMPP (Jabber) gateway
- RabbitMQ 1.4 was released July 29, 2008

RabbitMQ.com

- <http://www.rabbitmq.com/download.html>
- <http://www.rabbitmq.com/documentation.html>
- <http://www.rabbitmq.com/api-guide.html> (Java)
- <http://www.rabbitmq.com/examples.html>

RabbitMQ Clustering

- Unique feature of RabbitMQ broker
- Implemented with Erlang distributed nodes
- Data/state replication with full ACID properties
- Exception: queues (visible/reachable from everywhere, reside on node which created them – to be changed in future versions)
- 1 client = 1 socket. Cluster helps scale!
- High Availability

Client libraries

- RabbitMQ – Java, C#, Erlang, Javascript
<http://www.rabbitmq.com/download.html>
- Apache Qpid – java, c++, python, ruby, C#
<http://cwiki.apache.org/qpid/download.html>
- With Qpid, remember to use official spec XML!
- Pyamqplib - python
<http://barryp.org/software/py-amqplib/>
- Non-blocking sockets addon for pyamqplib
<http://lists.rabbitmq.com/pipermail/rabbitmq-discuss/2008-March/000937.htm>

Client libraries (continued)

- Ruby - <http://www.github.com/tmm1/amqp>
- ActionScript 3 (Flash) -
<http://github.com/0x6e6562/as3-amqp/tree/master>
- Want to write your own client? See how easy it is!
<http://hopper.squarespace.com/blog/2008/6/21/build-your-own-amqp-client.html>
- Ruby (Qpid) + RabbitMQ example
<http://somic-org.homelinux.org/blog/2008/06/24/ruby-amqp-rabbitmq-example/>

Community

- rabbitmq-discuss mailing list
- <http://lists.rabbitmq.com>
- Not a single thread ignored by core team!
- Excellent signal to noise ratio
- <http://groups.google.com/group/rabbitmq-discuss>
- The best support mailing list I have ever been a part of!

SCM

- <http://hg.rabbitmq.com/rabbitmq-server>
- <http://hg.rabbitmq.com/rabbitmq-java-client/>
- <http://hg.rabbitmq.com/rabbitmq-codegen/>
- Get Mercurial from
<http://www.selenic.com/mercurial>
- `hg clone repo_url`
- <http://lists.rabbitmq.com/pipermail/rabbitmq-discuss/2008-July/001372.html>

Broker Performance

- <http://www.lshift.net/news.20071116intelrabbit>
- Achieved throughput of 1.3 million msg/sec
- http://news.cnet.com/8301-13846_3-9983286-62.html
- XMPP + RabbitMQ = Twitter That Doesn't Go Down

Best Way to Read the Spec

- Start with specification PDF 0-9 or 0-10
- Do not start with PDF from 0-8!
- Figure out fundamentals and concepts first
- Most client API auto generate method code
- pyamqpplib has nice docstrings
- All AMQP commands are <method> in XML
- Methods are grouped in classes
- RabbitMQ currently implements AMQP 0-8

Broker on Your Laptop

- You can get erlang to run on Windows
- Watch out for firewall interfering with epmd (tcp 4369)
- Check out Windows Bundle available on RabbitMQ.com download page

RabbitMQ Elastic Server

- If you prefer Linux
- <http://elasticserver.com>
- Assemble a RabbitMQ virtual machine
- Requires a free VMWare player or deploy to Amazon EC2 cloud
- <http://es.cohesiveft.com/site/rabbitmq> (ver. 1.2)
- Or get a VM with auto updater from this url:
 - <http://es.cohesiveft.com/server/details/3243-rabbitmq-server-with-auto-updater-10-1217356768-VMware>

RabbitMQ Elastic Server

```
rabbitmq-server-with-auto-updater-10-1217356768
su: Authentication failure
Sorry.
cftuser@rabbitmqserverwithautoupdater:~$
cftuser@rabbitmqserverwithautoupdater:~$ sudo su

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

Password:
rabbitmqserverwithautoupdater:/home/cftuser# rabbitmqctl status
Status of node rabbit@rabbitmqserverwithautoupdater ...
[{running_applications,[{rabbit,"RabbitMQ","1.4.0"},
                        {mnesia,"Mnesia CXC 138 12","4.3.3"},
                        {os_mon,"CPO CXC 138 46","2.1.1"},
                        {sas1,"SASL CXC 138 11","2.1.4"},
                        {stdlib,"ERTS CXC 138 10","1.14.2"},
                        {kernel,"ERTS CXC 138 10","2.11.2"}]},
 {nodes,[rabbit@rabbitmqserverwithautoupdater]},
 {running_nodes,[rabbit@rabbitmqserverwithautoupdater]}]
done.
rabbitmqserverwithautoupdater:/home/cftuser# _

VMware Tools is out of date. Choose the Virtual Machine > Install VMware Tools menu.
```

Using Rabbit at Cohesive

- GUI “sidekick” - loads same data models (MVC) as GUI, actions are triggered by messages instead of user clicks
- Web front end dispatches jobs
- Backend sends status updates
- Some actions triggered from cron
- Can have many distributed “sidekicks”

Thank you!



Im in yr serverz,
queueing yr messagez

Photo credit: <http://flickr.com/photos/53366513@N00/67046506/>