

FOLLOW US:
[TWITTER.COM/REDHATSUMMIT](https://twitter.com/redhatsummit)

TWEET ABOUT US:
ADD #SUMMIT AND/OR #JBOSSWORLD TO THE END
OF YOUR EVENT-RELATED TWEET

What's all the buzz about?

Tim Fox
Messaging Lead
JBoss
03 Sep 2009

Change of plan

- This presentation was scheduled to be about JBoss messaging
- This presentation is actually about **“HornetQ”**
<http://jboss.org/hornetq>





HornetQ

Putting the buzz in messaging

- HornetQ is the new name for JBoss Messaging 2
- HornetQ is an open source community project to build a multi-protocol asynchronous messaging system
- HornetQ is designed for performance
- HornetQ is designed with usability in mind
- HornetQ is full featured
- See the wiki for more information:
<http://www.jboss.org/community/wiki/HornetQGeneralFAQs>

How does HornetQ relate to JBoss Messaging?

- We decided to rename JBoss Messaging 2 to HornetQ
- JBM 1.x and 2.x code bases 95%+ different.
- HornetQ is a different beast to JBM 1.x
- HornetQ is not tightly coupled to JBoss Application Server

How is HornetQ licenced?

- Most of HornetQ is licenced using ASL 2.0
- A few files still under LGPL
- Why ASL, not LGPL?

Usability is critical!

- Key design goal of HornetQ is ease of use
- Simple API
- Clear and simple to configure
- Great docs
- Ships with over 65 fully runnable examples out of the box
- Minimal third party dependencies

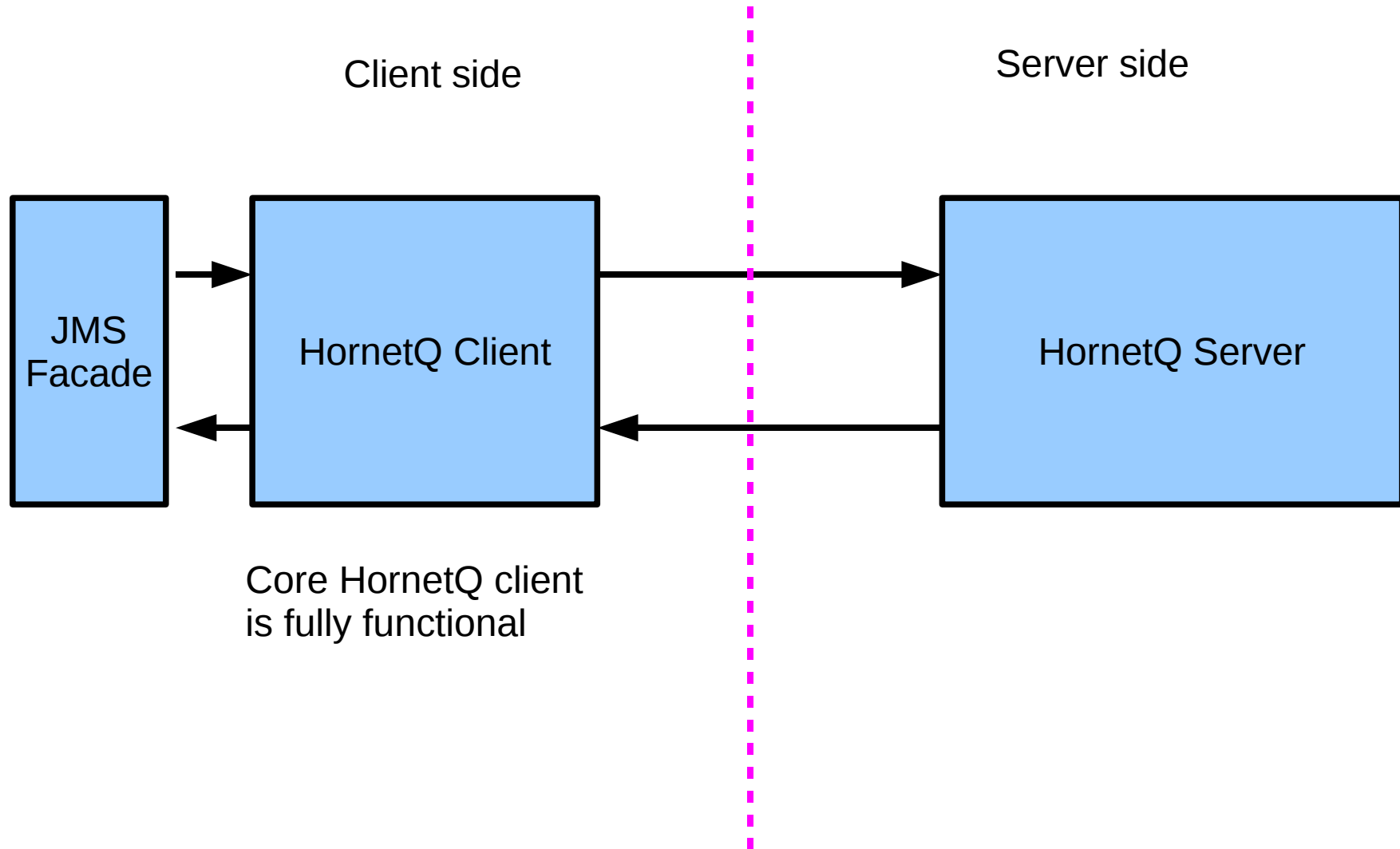
Both standalone and JEE messaging

- HornetQ is a fully functional stand-alone messaging server – if you don't want an app server, don't use an app server
- HornetQ can also be integrated with any JEE application server, e.g. JBoss Application Server 5.0, using its JCA adaptor
- HornetQ can also be used with any dependency injection framework, e.g. JBoss MC, Spring, Google Guice

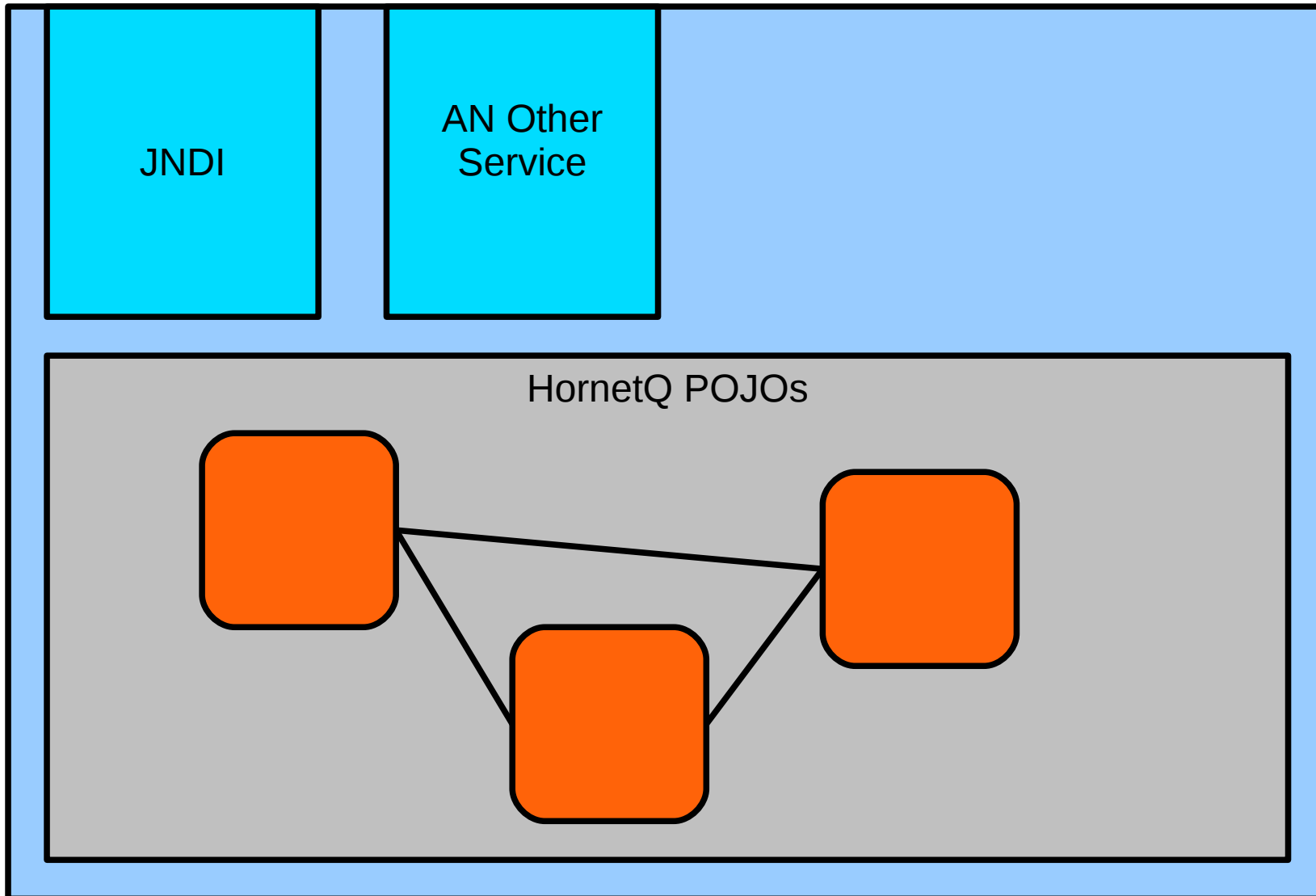
Elegant generic architecture

- Fully functional JMS agnostic messaging system
- No dependencies on JMX, JNDI, JCA, etc.
- Just a set of simple POJOs
- JMS functionality applied as thin facade on the client side
- Much, much more than just JMS!
- Can be embedded in an application that requires messaging internally.

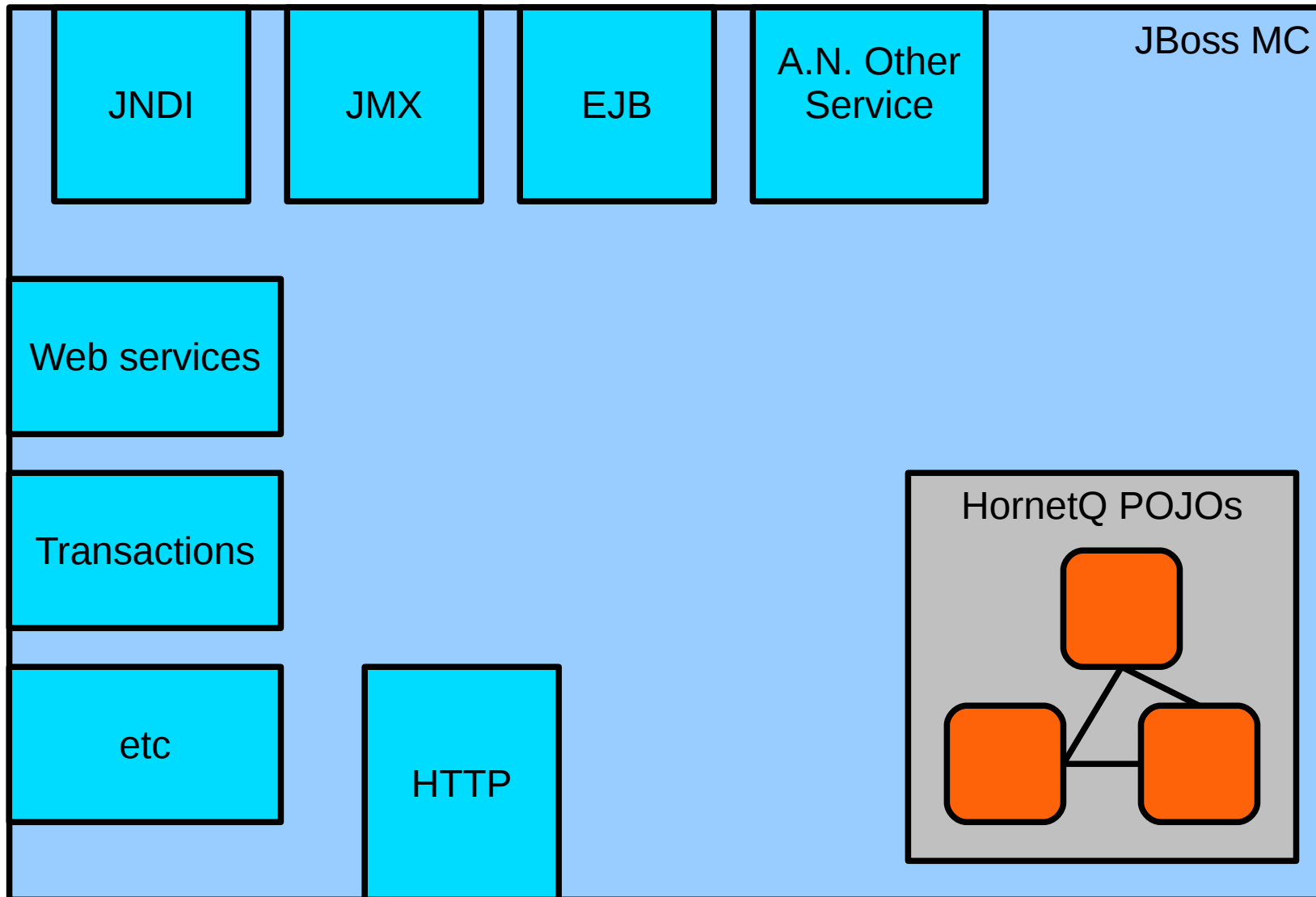
Generic core and JMS facade



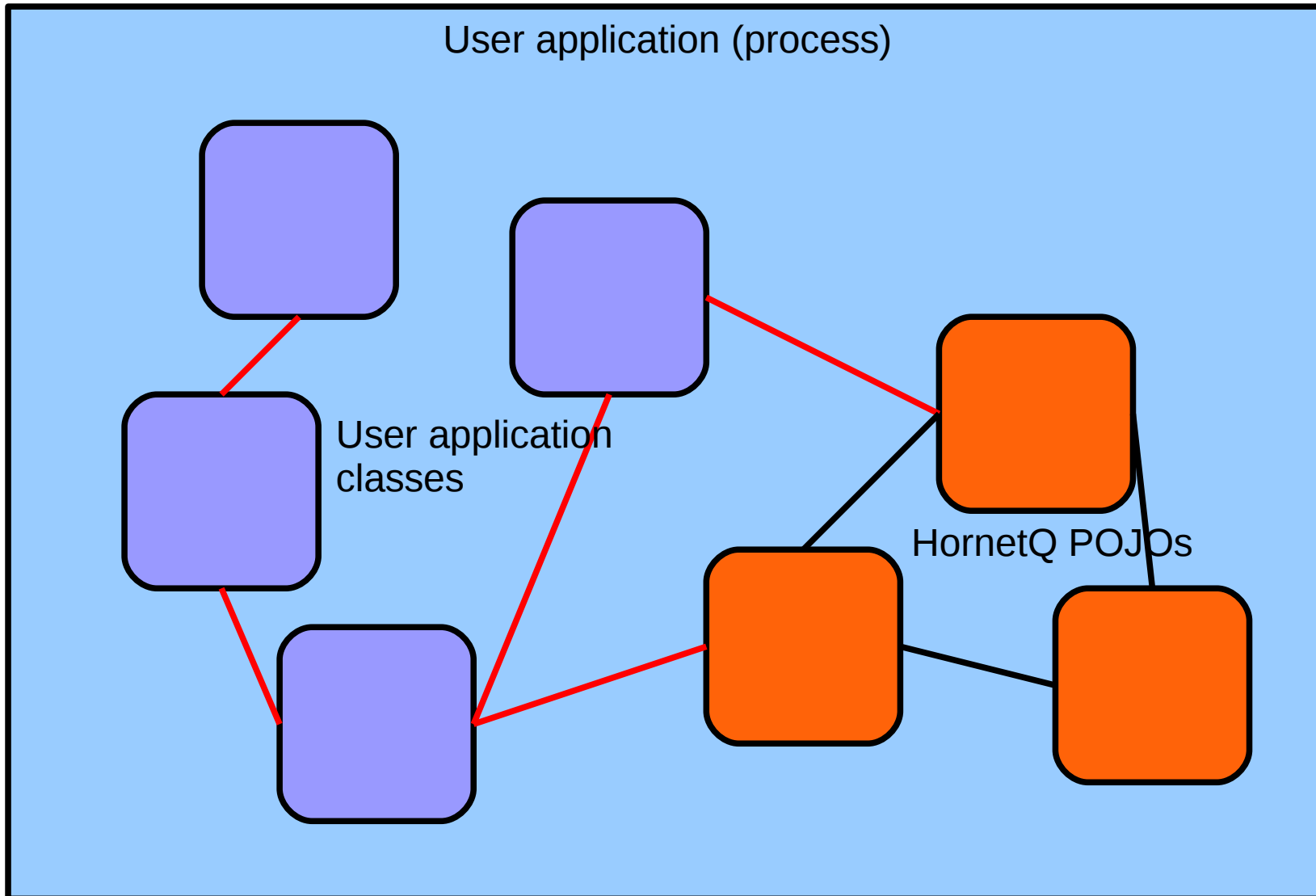
Stand-alone deployment using JBoss Micro-container



HornetQ inside JBoss AS 5.x



HornetQ embedded in 3rd party application



HornetQ embedded code example

```
01. MessagingServer server = new MessagingServerImpl();
02. server.start();
03. ClientSessionFactory sf = new ClientSessionFactoryImpl(...);
04. ClientSession sess = sf.createSession();
05. sess.createQueue("address1", "queue1");
06. sess.start();
07. ClientProducer prod = sess.createProducer("address1");
08. Message message = new ClientMessageImpl(false);
09. message.getBody().writeString("hello world");
10. prod.send(message);
11. ClientConsumer cons = sess.createConsumer("queue1");
12. Message received = cons.receive();
13. System.out.println("Got message " + received.getBody().readString());
14. sess.close();
15. server.stop();
```

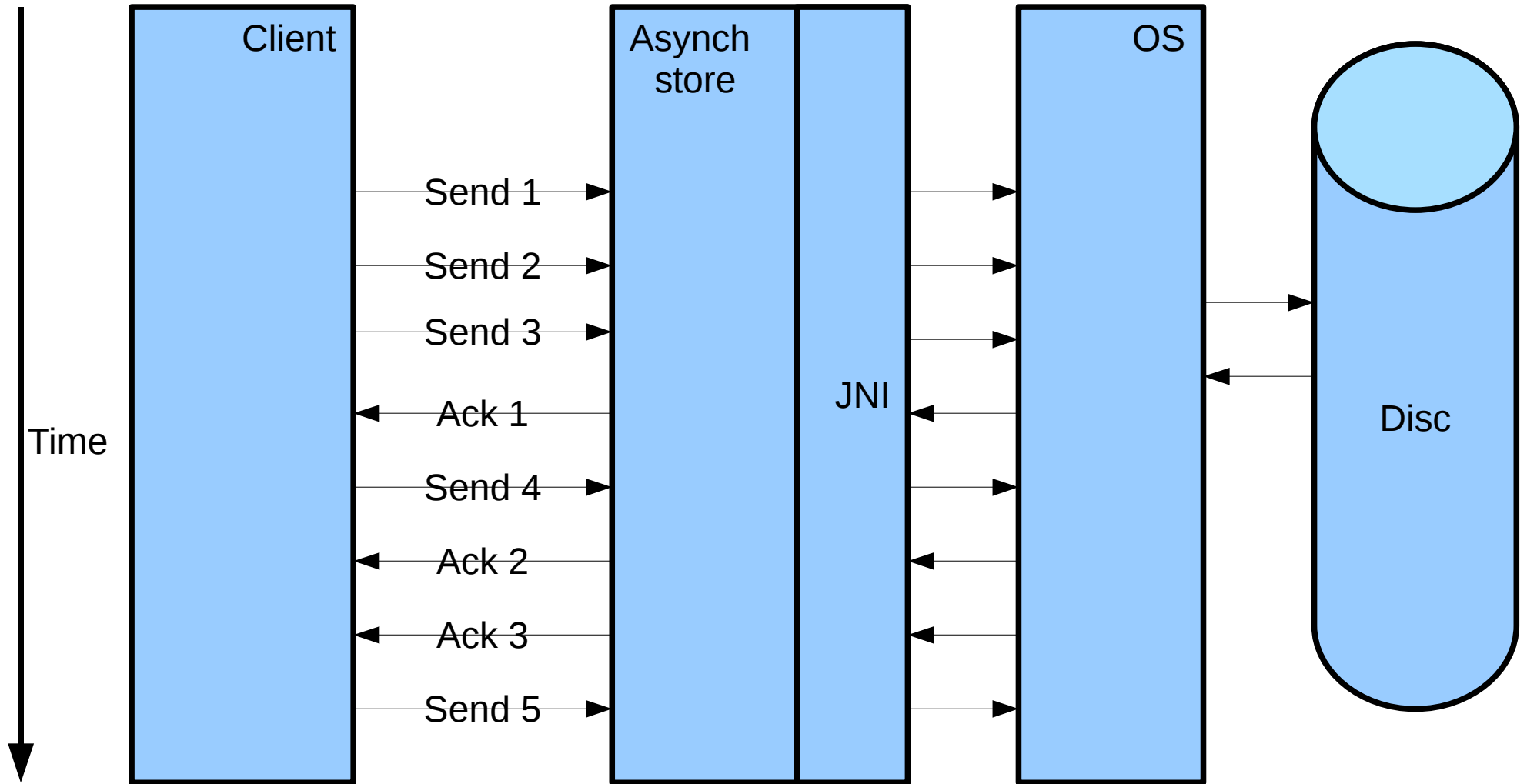
HornetQ features

- Very high performance journal
- Support for huge queues and huge messages with small server and client footprint
- Pluggable transport system
- Seamless High Availability (HA)
- Massively flexible clustering
- Extensive management API
- Lots, lots, more, but no time here! See the wiki for a full list:
<http://www.jboss.org/community/wiki/HornetQFeatures>

Ultra high performance journal

- HornetQ persistence is very fast
- Very fast store using Linux asynchronous IO.
- Up to 100+ MiB/s on a single node!
- JNI interface to aio library (libaio), encapsulated in Java package.
- Automatic switches to Java NIO when not running on Linux

Asynchronous IO on Linux



Huge queues and messages

- HornetQ supports huge queues – far bigger than can fit in available RAM
- Run Terabyte queues while the server is only running in 50MiB of RAM!
- Send and receive huge multi-gigabyte messages with all the normal transactional semantics
- Effectively, only limit to message size is available disk space. We have tested messages up to 8 GiB in size.

Configurable Transport system

- Fully pluggable transport
- Ships with default implementation using JBoss Netty
<http://jboss.org/netty/>
- TCP transport
- SSL transport
- HTTP transport
- Servlet transport
- In-VM transport

HornetQ High Availability

- Transparent reconnection on failure
- 100% guaranteed no loss or duplication of messages
- Fail-over via store on SAN
- Replicated journal (Shared nothing approach)

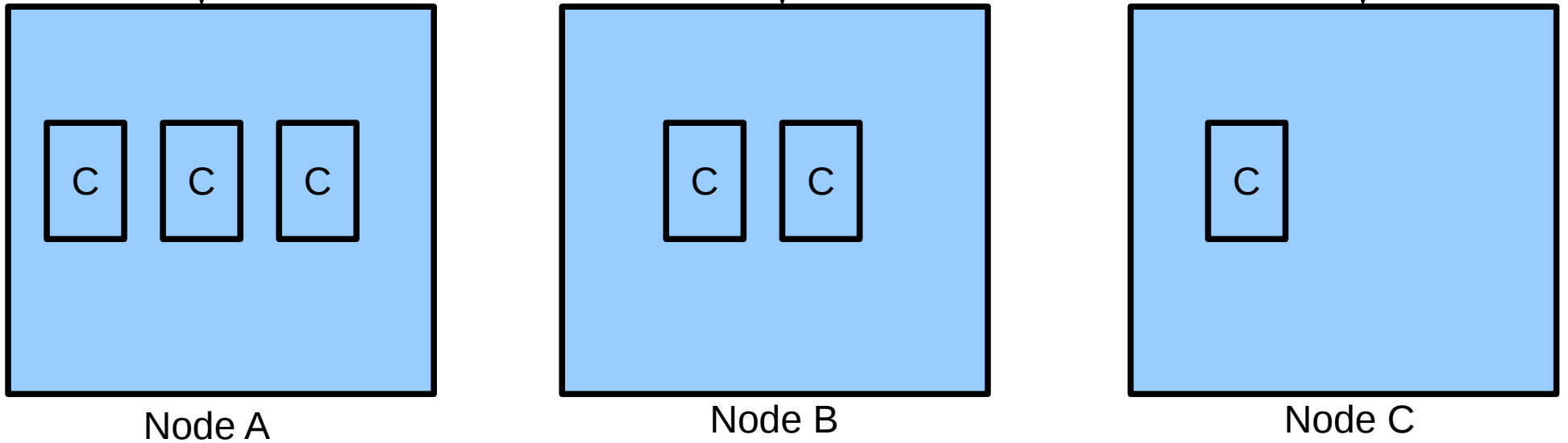
HornetQ Clusters

- HornetQ servers can be grouped into clusters
- Clusters are a way of balancing message processing across several nodes.
- Messages arriving on cluster are balanced to different nodes to spread the load – default balancing is round robin.
- Balancing takes into account selectors and consumers on each node

Clusters

Messages

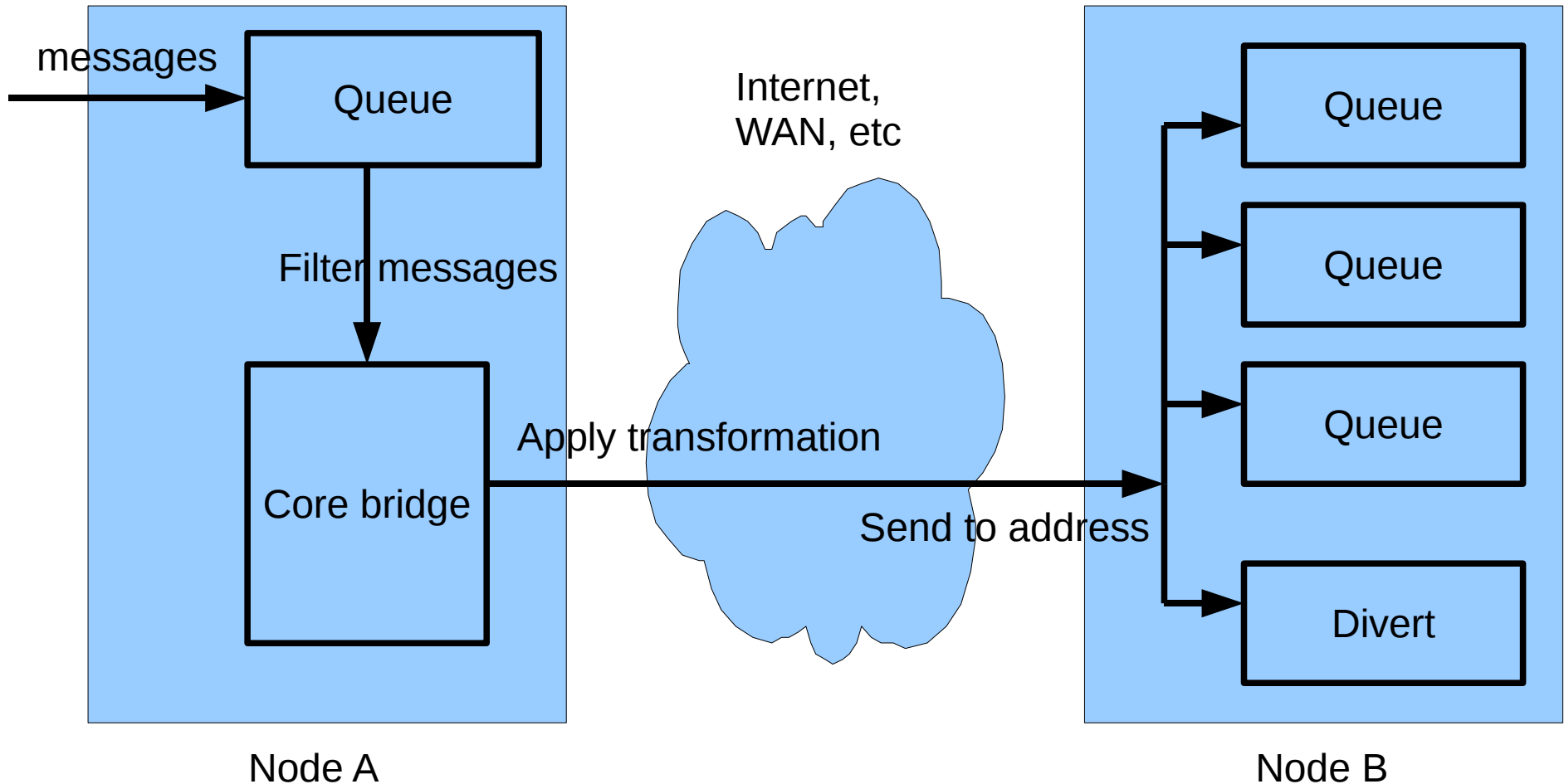
Messages balanced according to number of matching consumers



HornetQ Core Bridges

- Core bridges take messages from one queue and forward them to another remote address
- Core bridges can use filters to select only certain messages
- Core bridges can apply transformations
- Core bridges are high performance
- Core bridges can work with unreliable connections
- Core bridges are not JMS bridges

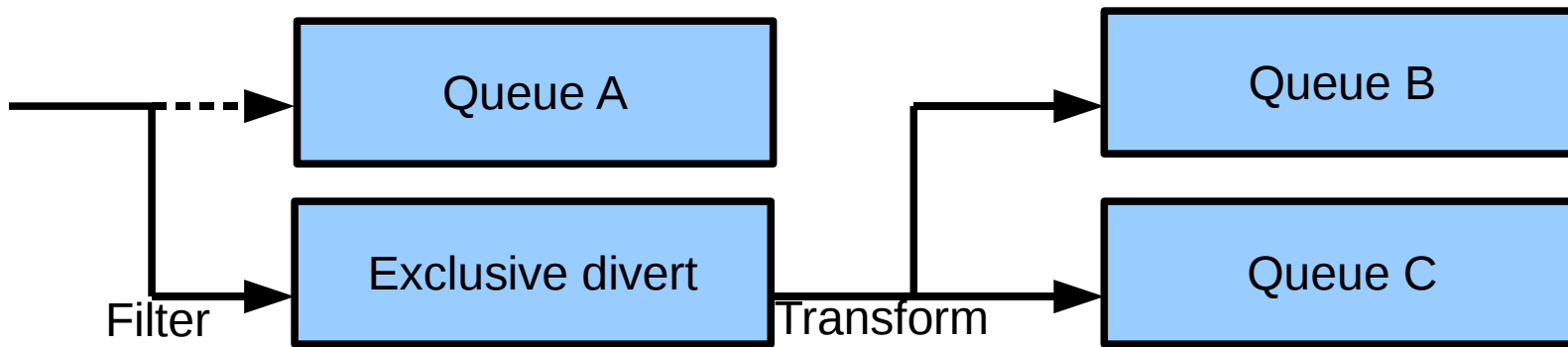
Core bridges



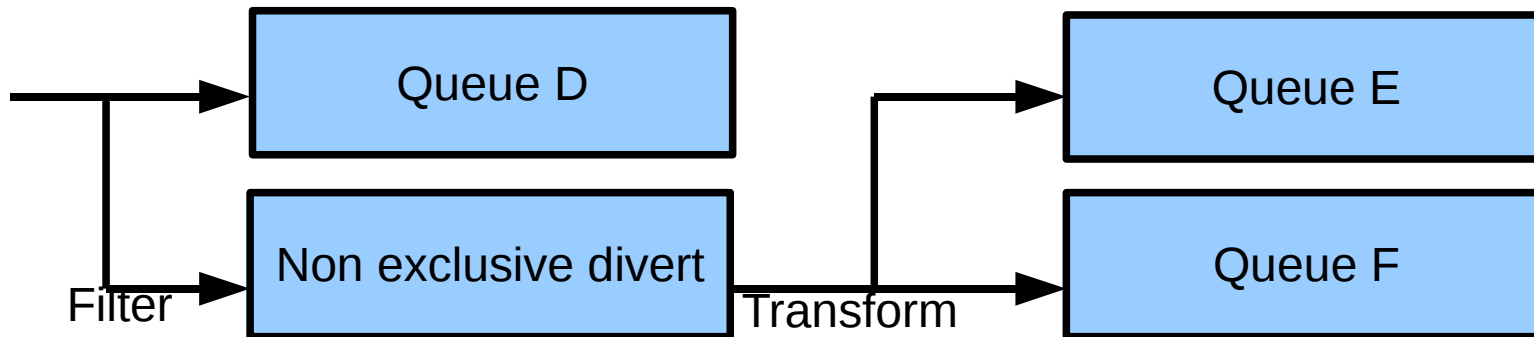
HornetQ diverts

- Diverts are a *routing table* for messaging
- Diverts take messages destined for one address and divert them to another address
- Diverts can be exclusive or not. Non exclusive diverts copy messages.
- Diverts can apply transformations
- Diverts can use filters to only select certain messages
- Diverts are high performance

Diverts



Exclusive diverts can be used to “divert”



Non exclusive diverts can be used to “siphon” or “snoop”

Global scale messaging fabric

- Combine clusters, bridges and diverts and you can create a massively configurable global messaging fabric to run your business. All with zero message loss and zero message duplication guarantee.
- Add HA to provide unbroken up-time.
- Use HornetQ as the messaging fabric together with JBoss ESB, and benefit from all the 3rd party connectors provided by JBoss ESB.

Going ahead

- Interoperability – REST, AMQP, STOMP, XMPP, Ajax/Comet
- Server-less mode
- Performance bench-marks
- See the road map here:
<http://www.jboss.org/community/wiki/Roadmap>

Interoperability

- REST
Provide a simple RESTful interface for HornetQ over HTTP.
See REST-* project
- AMQP – HornetQ will implement the AMQP protocol
- STOMP
Native STOMP support provides access to HornetQ by many STOMP clients written in different languages
- XMPP
- Async web support (Ajax/comet/bayeaux)

Join us!

- HornetQ is a community project - come and get involved!
- We are always looking for new developers to help out – there is a lot of cool new stuff to implement.
- Find us on irc:
<irc://freenode.net:6667#hornetq>

QUESTIONS?

TELL US WHAT YOU THINK:
[REDHAT.COM/JBOSSWORLD-SURVEY](http://redhat.com/jbossworld-survey)

Check out the HornetQ web site:
<http://jboss.org/hornetq>

Follow HornetQ on twitter:
<http://twitter.com/hornetq>

Visit the blog:
<http://hornetq.blogspot.com>