Introduction to Apache ActiveMQ

Hiram Chirino
Software Fellow
Blog: http://hiramchirino.com/blog/
Twitter: @hiramchirino
GitHub: https://github.com/chirino

FuseSource
integration everywhere

About me

- **Hiram Chirino**

  Blog: http://hiramchirino.com/blog/
  Twitter: @hiramchirino
  GitHub: https://github.com/chirino

- Software Fellow at FuseSource - http://fusesource.com
- Apache Member and ActiveMQ PMC Chair
- Apache Committer on: ActiveMQ, Camel, Karaf, ServiceMix, Geronimo, Felix, and Aries
- Lead of STOMP 1.1 SpecificationCo-Founder of many other OS projects:
  - HawtDispatch, Scalate, LevelDBJNI, Jansi, And many more!

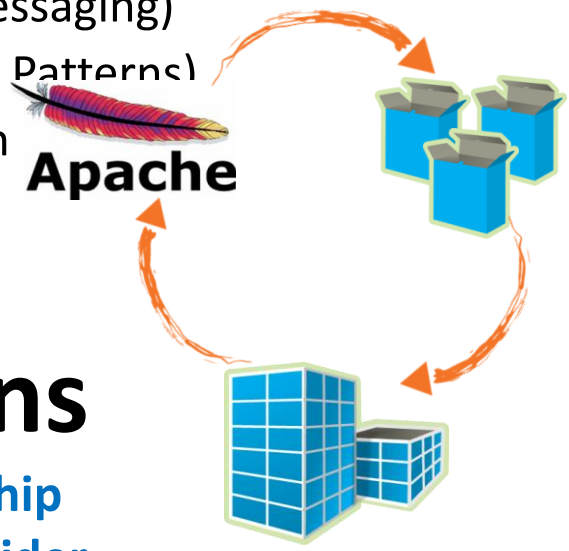
Agenda

- Who's FuseSource?
- ActiveMQ Overview
  - Core capabilities
  - Managing client connections
  - Managing persistence
  - High availability
  - Network of brokers
- What's New in ActiveMQ 5.6
- Demo

Bringing Open Source Integration & Messaging to Enterprise IT

Apache Software Foundation

- ActiveMQ (reliable messaging)
- Camel (Ent. Integration Patterns)
- ServiceMix/Karaf (con



Enterprise OSS Products:

Fuse ESB Enterprise

Fuse MQ Enterprise

- Integrated solutions
- Tested and certified
- Documented

Subscriptions

Collaborative relationship with your software provider

- Enterprise tooling
- Services level agreement
- WW support organization

Training & Consulting

- Expert training on site or via the Web
- Packaged services for all phases of the lifecycle

FuseSource
integration everywhere

FuseSource Subscription:
Collaborative relationship with your software provider

Support

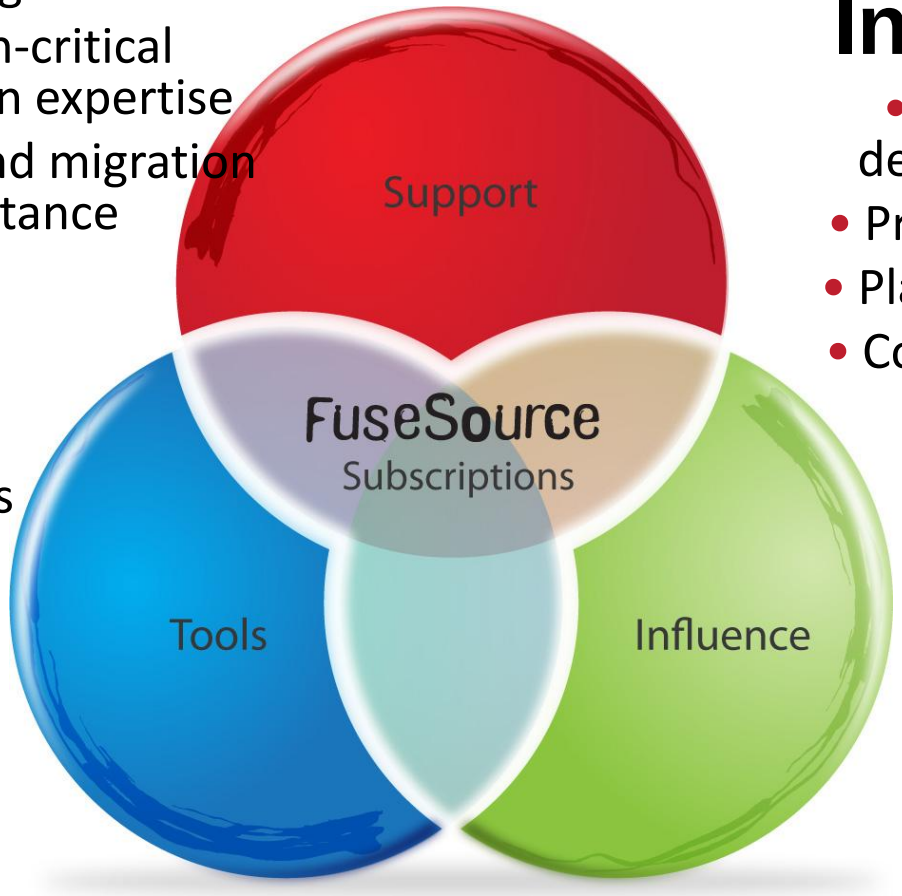
- Enterprise-class 24x7 coverage
- Global organization
- Mission-critical integration expertise
- Updates and migration assistance

Influence

- Access to the development team
- Product roadmaps
- Planning processes
- Conduit to Apache

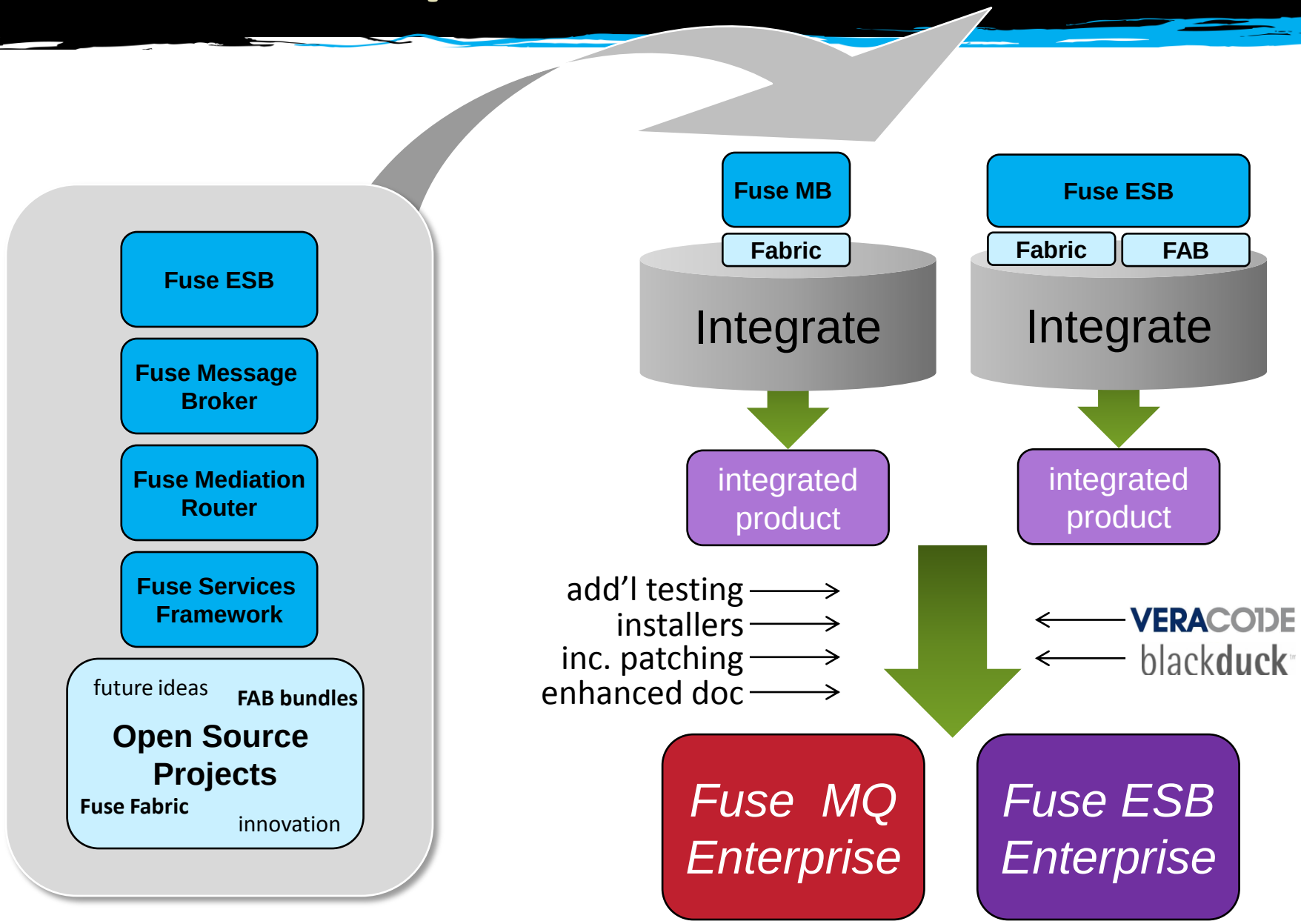
Tools

- Certified distributions
- Dev, ops, and management tools
- Performance tuning
- Documentation



FuseSource
integration everywhere

Where Enterprise Products Come From



FuseSource : Open Source Integration and Messaging

Cost-effective, enterprise-class solutions

- Over 25 active Apache committers on staff
- Business-friendly, Apache open source license
- Backed by stable, global software company

Apache Project	Capabilities
ActiveMQ	Reliable messaging for Java / JMS, C++ and .NET
ServiceMix	ESB combining the best of Apache Integration projects
Karaf	OSGI-based integration server
Camel	Enterprise Integration Pattern framework
CXF	SOAP, XML and RESTful web services

FuseSource : Team that Wrote the Code

No one knows the code, or influences the projects at Apache more than FuseSource:

- Co-founders and PMC members of ServiceMix, Karaf, ActiveMQ, Camel, and others…
- Over 25 active committers on 11 Apache projects

Guillaume Nodet

James Strachan

Rob Davis

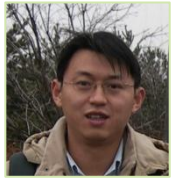
Hiram Chirino

Jon Anstey

Gary Tully

Dejan Bosanac

Gert Vanthienen



Willem Jiang

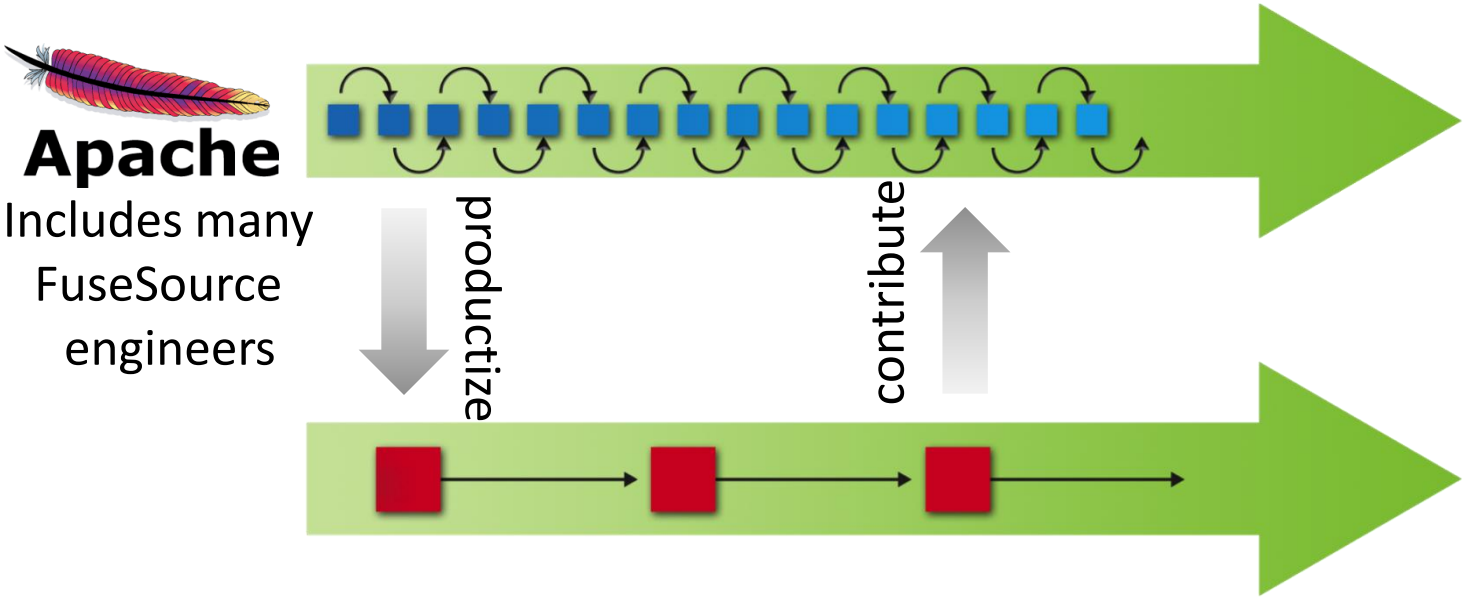
Claus Ibsen

FuseSource
integration everywhere

FuseSource : Alignment with Apache

Not just a team of hackers – FuseSource drives the products

- No one knows the internals of the projects better
- FuseSource has access to product road maps
- Customer patches are contributed to Apache
- Customer feedback drives project direction

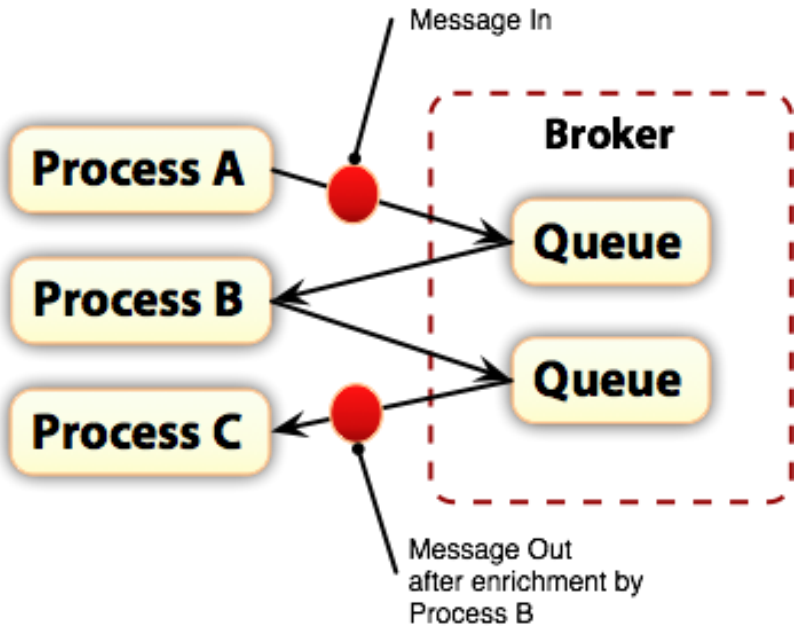


What is Apache ActiveMQ?

- Top level Apache Software Foundation project
- Wildly popular, high performance, reliable message broker
  - Supports JMS 1.1; adding support for AMQP 1.0 and JMS 2.0
  - Clustering and Fault Tolerance
  - Supports publish/subscribe, point to point, message groups, out of band messaging and streaming, distributed transactions, …
- Myriad of connectivity options
  - Native Java, C/C++, and .NET
  - STOMP protocol enables Ruby, JS, Perl, Python, PHP, ActionScript, ...
- Embedded and standalone deployment options
  - Pre-integrated with open source integration and application frameworks
  - Deep integration with Spring Framework and Java EE

Why use Messaging?

- Reliable remote communication between applications
- Asynchronous communication
  - De-couple producer and consumer (loose coupling)
- Platform and language integration
- Fault tolerant - processing can survive Processor outage
- Scalable - multiple consumers of each queue
  - Distributes processing

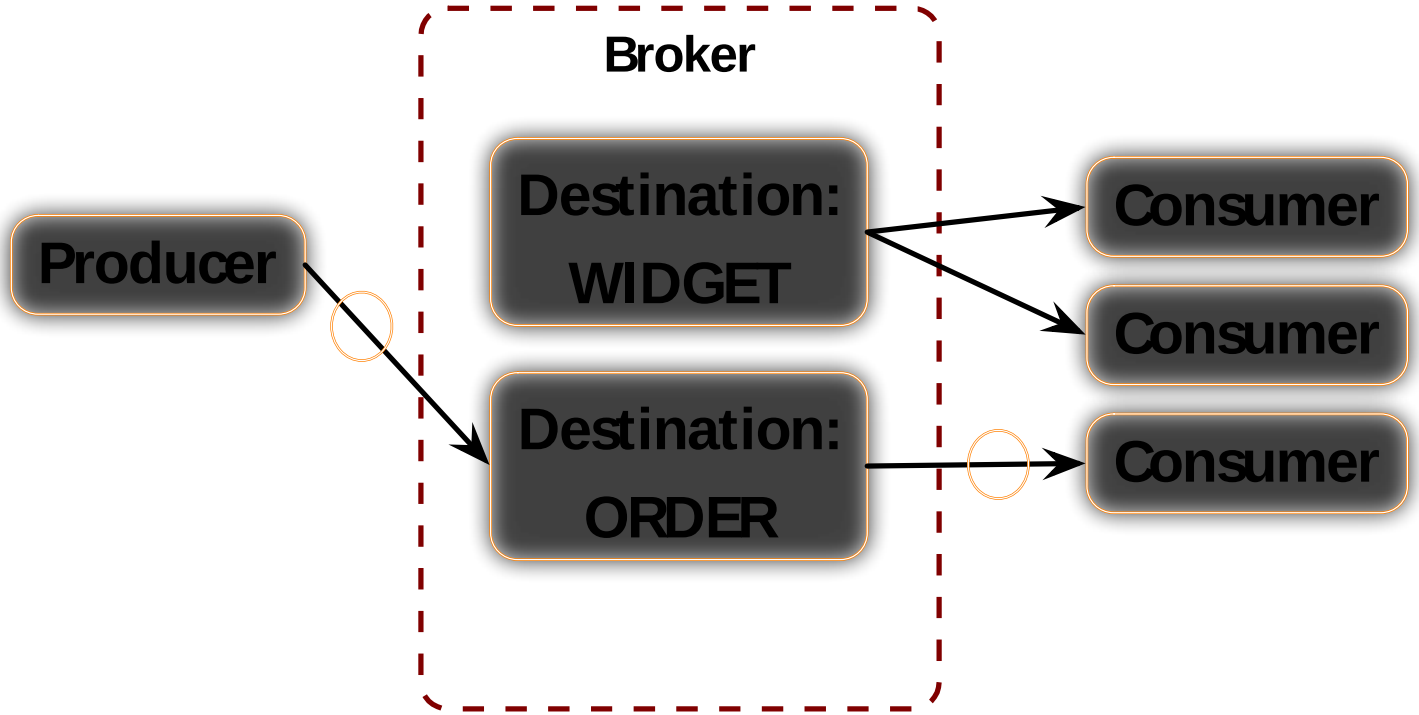


FuseSource
integration everywhere

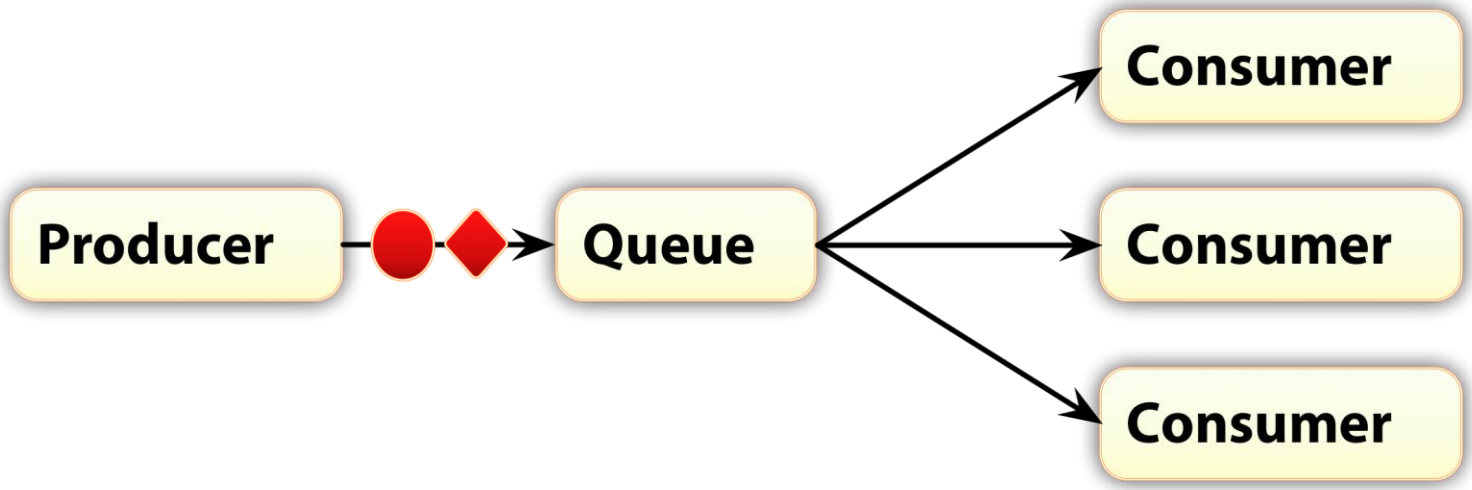
Message Channels and Routing

- Message Channels
  - Named communication between interested parties
  - JMS calls them ‘Destinations’
- Can fine-tune message consumption with selectors
- Can route a message based on content

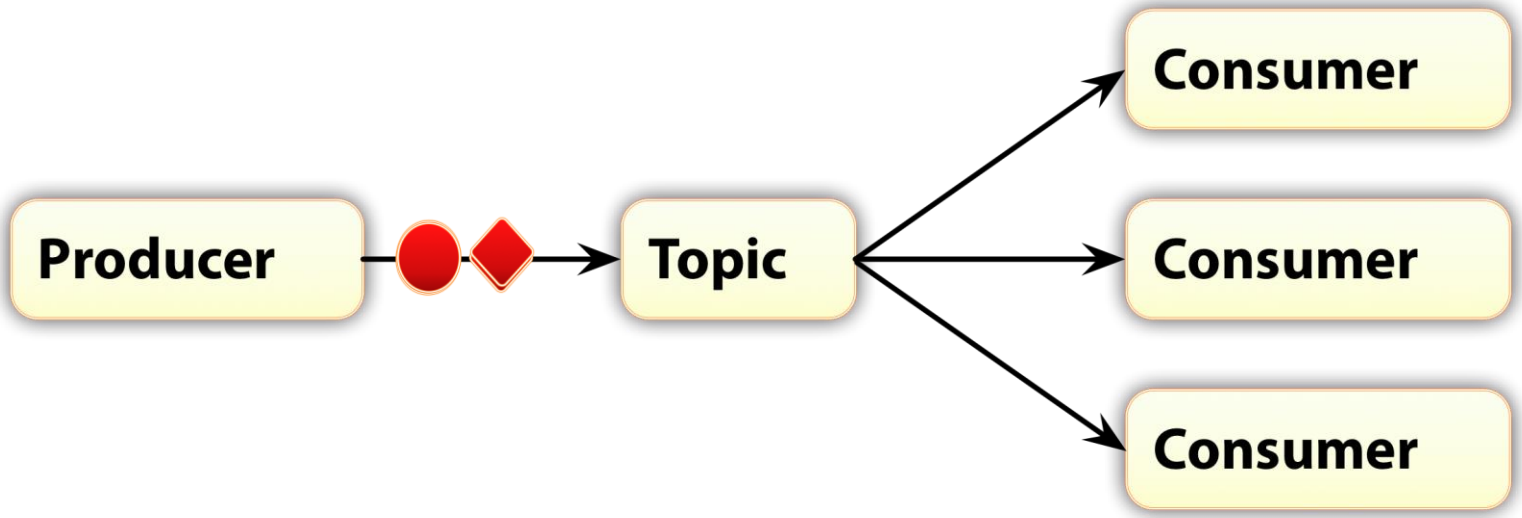
Message Channels = JMS Destinations



Point-to-Point Channel : JMS Queues

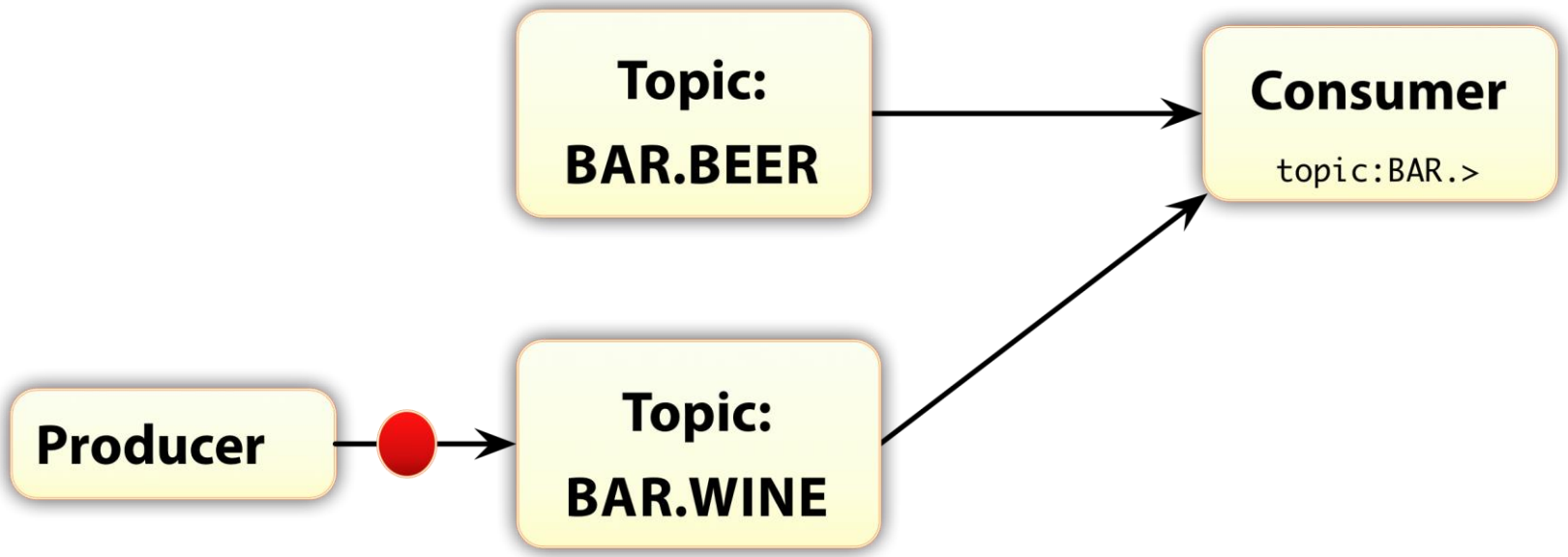


Publish/Subscribe Channel : JMS Topics



Message Routing : Selectors

Message Routing : Destination Wildcards



Managing Client Connections : Transport Connectors

- Configured in broker for client connections

- TCP – most used; socket connections using binary Openwire protocol
- NIO – like TCP, excepts uses Java NIO to reduce number of threads managing all connections
- SSL – secure TCP connection
- STOMP – text based protocol; facilitates multiple language integration
- VM – enables efficient in-process connections for embedded broker

- Examples
  - <transportConnector uri="tcp://0.0.0.0:61616"/>
  - <transportConnector uri="nio://0.0.0.0:61616"/>
  - <transportConnector uri="stomp://0.0.0.0:61617"/>
  - <transportConnector uri="stomp+nio://0.0.0.0:61617"/>

Managing Client Connections : Wrapper Transports

- Augment / wrap client side connections
- Failover – automatic reconnection from connection failures
- Fanout – simultaneously replicate commands and message to multiple brokers
- Example – client connection URI
  - tcp://master:61616
  - failover:(tcp://master:61616,tcp://slave:61616)
  - failover:(tcp://virtualIp:61616)
  - fanout:(static:(tcp://host1:61616,tcp://host2:61616))

Managing Client Connections : Configuring Transports

- tcp://hostname:port?key=value

- Examples
  - tcp://myhost:61616? trace=false&soTimeout=60000
  - failover:(tcp://master:61616?soTimeout=60000,tcp://slave:61616)?randomize=false

- Lot more details at
  - http://fusesource.com/documentation/fuse-message-broker-documentation/
  - http://activemq.apache.org/configuring-transports.html

Managing Persistence : Persistence Adapters

- File system based
  - kahaDB – recommended; improved scalability and quick recovery
  - amqPersistenceAdapter – legacy; fast, but slow recovery
- RDBMS based
  - jdbcPersistenceAdapter – quick and easy to setup
  - journaledJDBC – faster than pure JDBC; file journaling with long term JDBC storage
- Memory based
  - memoryPersistenceAdapter – testing only; same as
    - <broker persistent="false">

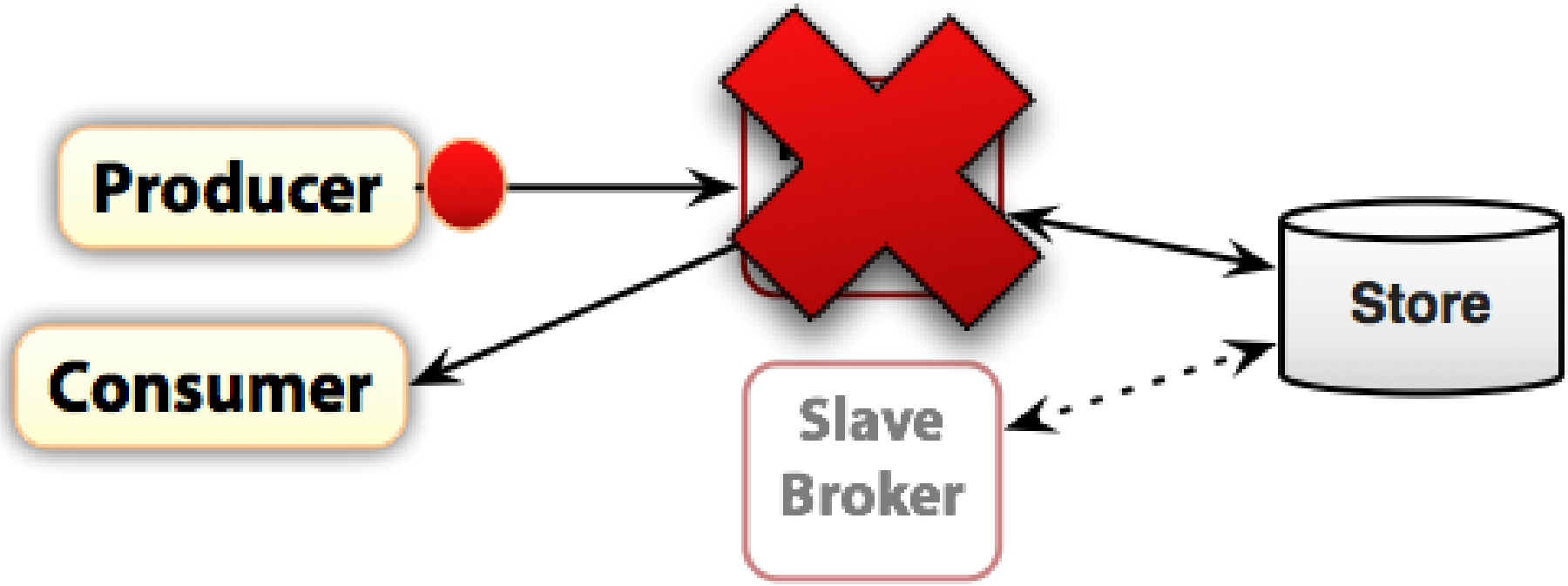
High Availability

- Two complementary approaches:
  - Master/Slave – access to persistent messages after broker failure
  - Network of Brokers – Scale out message processes - next slides…

- Master/Slave Context
  - A given message is in one and only one broker (persistence store)
  - If a broker instance fails, all persistent messages are recoverable upon broker restart
  - Master/Slave allows a 2nd broker instance (slave) to be ready to process persistent messages upon master (1st broker) failure
  - Clients should use Failover transport for automatic connect to slave
    - failover:(tcp://master:61616,tcp://slave:61616)?randomize=false

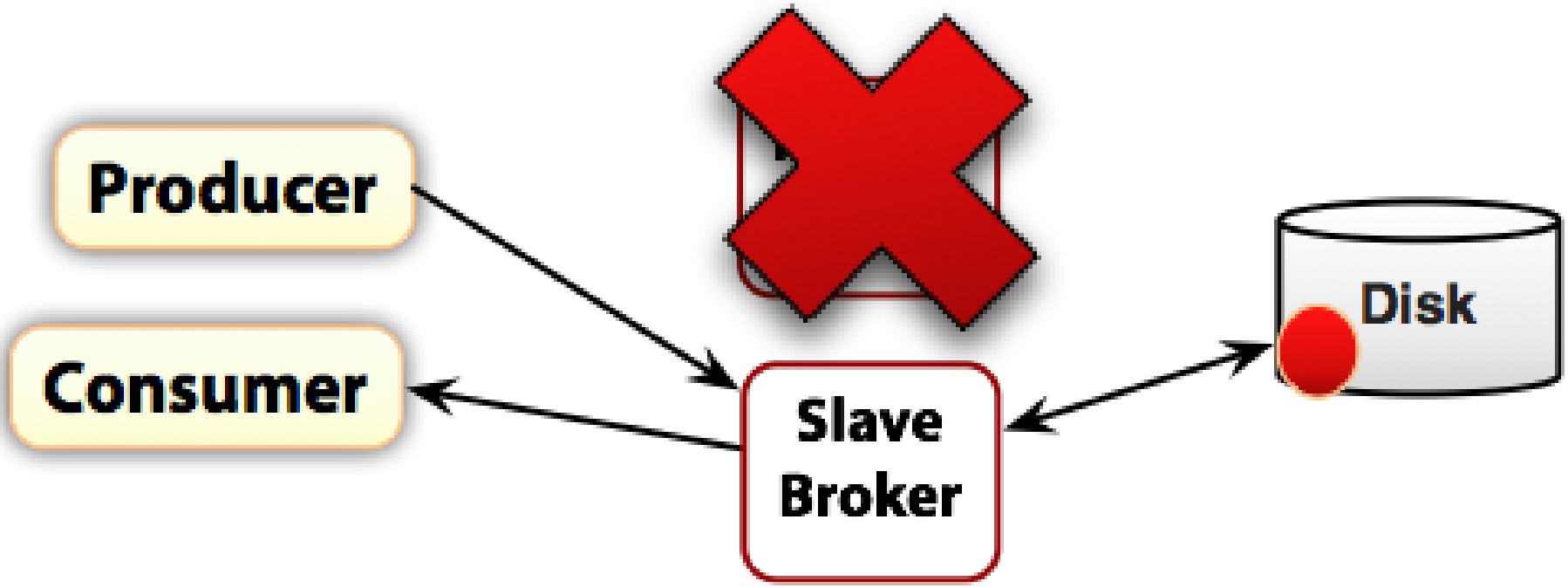
High Availability : Master/Slave

- failover:(tcp://master:61616,tcp://slave:61616)?randomize=false



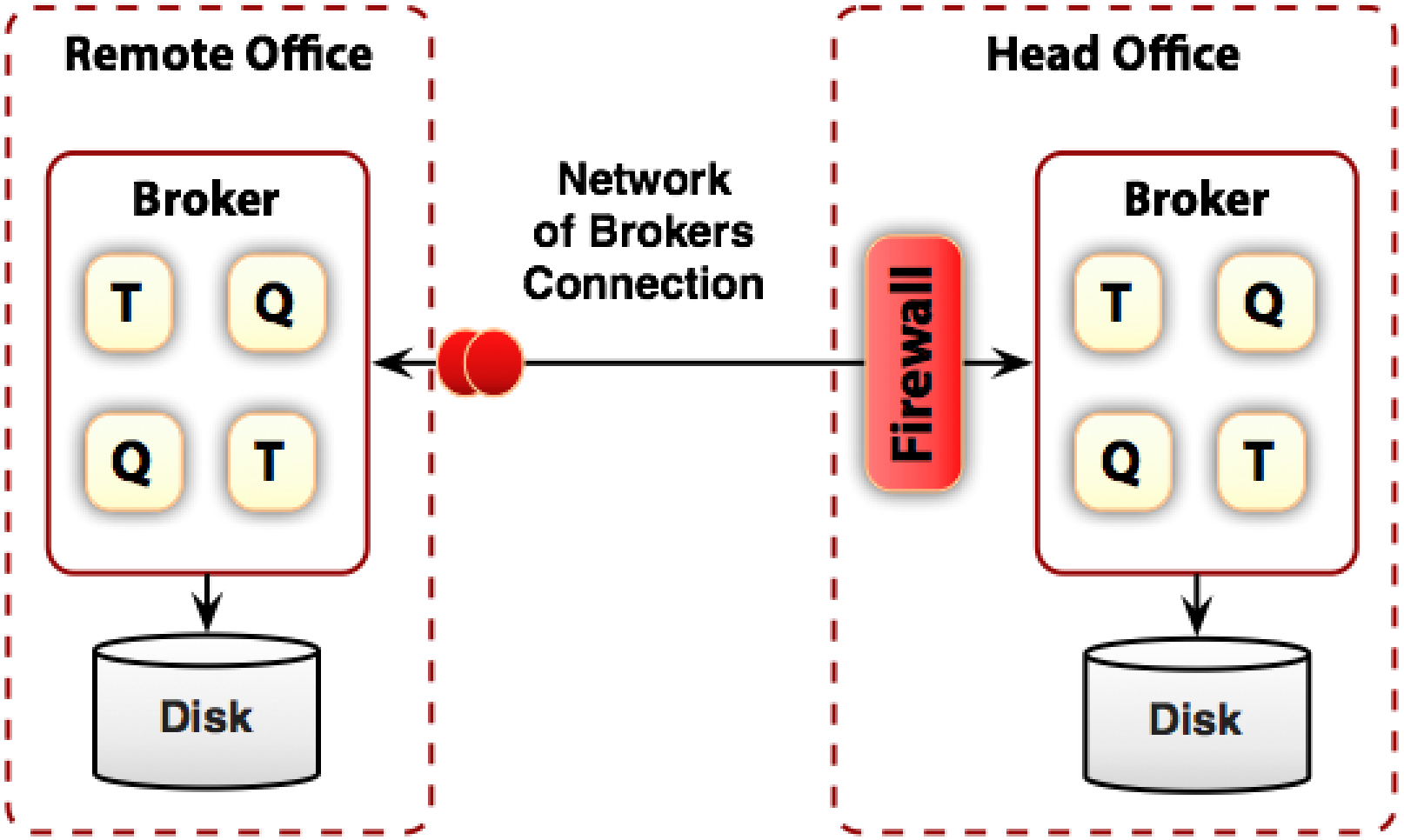
High Availability : Master/Slave

- failover:(tcp://master:61616,tcp://slave:61616)?randomize=false

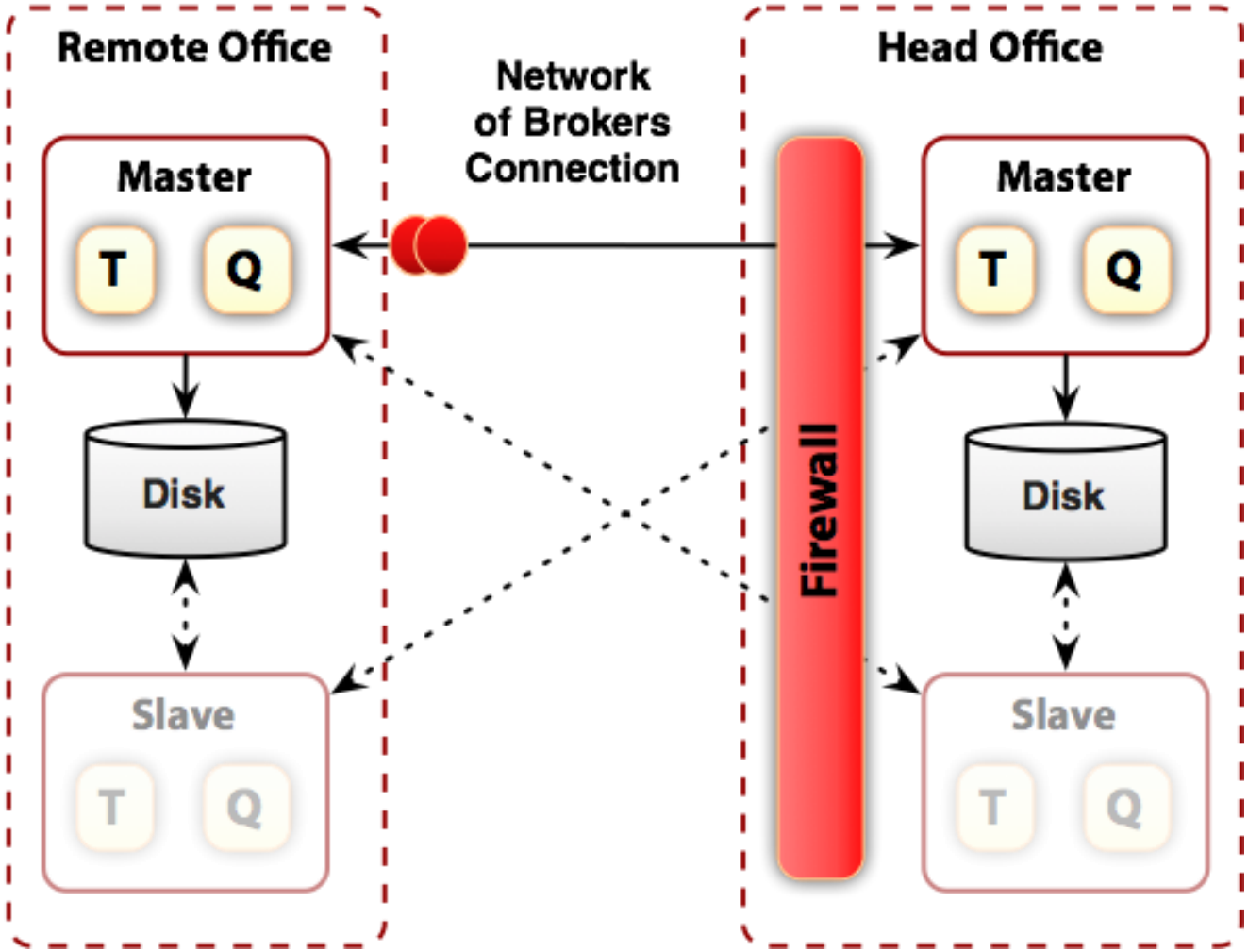


Network of Brokers : Geographically Dispersed

Network of Brokers : Geographically Dispersed



Network of Brokers : Network with Master/Slave



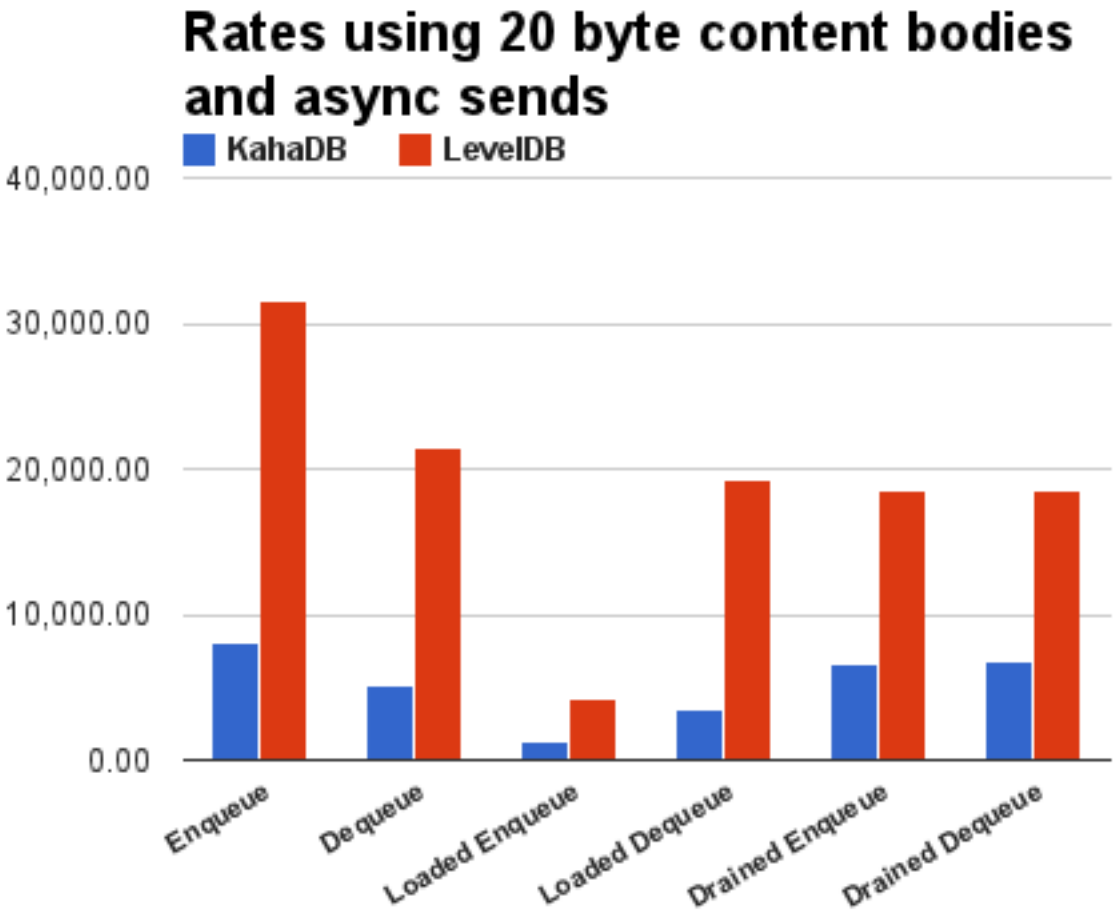
What's New In ActiveMQ 5.6

- LevelDB Store
- MQTT transport
- New LDAP security module
- Stomp Enhancements
- Multi KahaDB persistence
- Priority Failover URIs
- Automatic client rebalance in broker cluster

LevelDB Store vs KahaDB

- Fewer index entries per message than KahaDB
- Faster recovery when a broker restarts
- LevelDB index out-perform Btree index at sequential access .
- LevelDB indexes support concurrent read access.
- Pauseless data log file garbage collection cycles.
- Fewer IOPS to load stored messages.
- It exposes it's status via JMX for monitoring

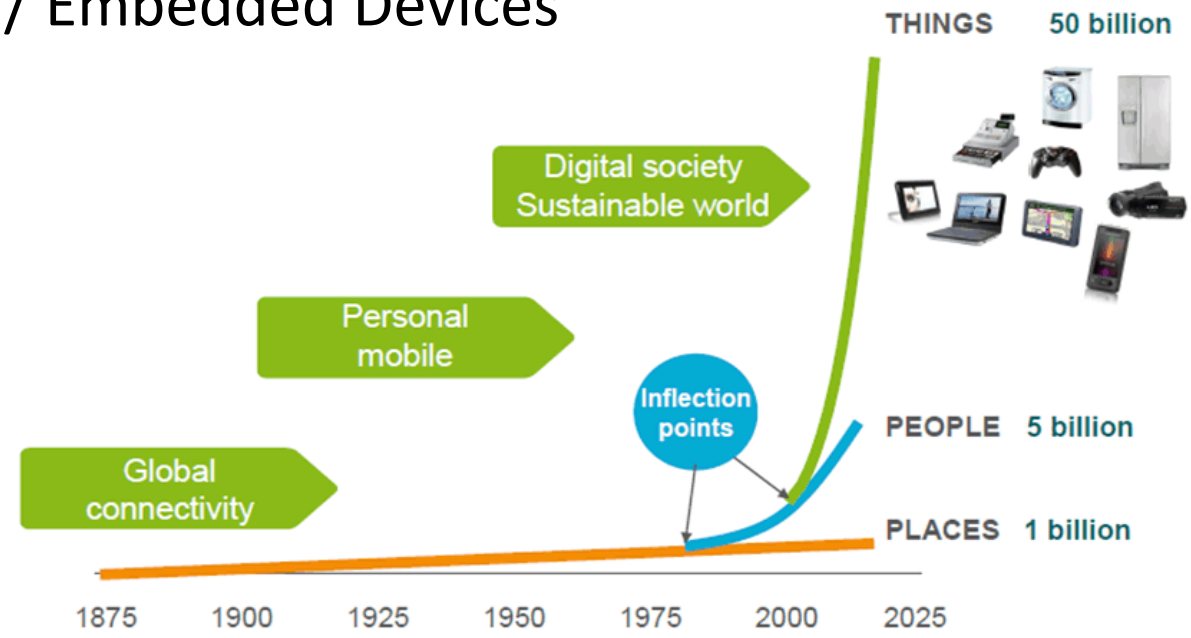
ActiveMQ 5.6: LevelDB Store



FuseSource
integration everywhere

Protocol: MQTT

- Focused on:
  - Pub/Sub
  - Unreliable, low bandwidth networks
  - Small footprint / Embedded Devices
- Interoperates with Apollo, WebsphereMQ, Mosquitto, …



Source: Ericsson AB, "Infrastructure Innovation - Can the Challenge be met?," Sept 2010

FuseSource
integration everywhere

ActiveMQ 5.6 STOMP Enhancements

- STOMP 1.1 Support
  - Protocol Version
  - Heartbeat
  - NACK frames
- Additional STOMP Extensions
  - Queue Browsing
  - Numeric Selectors
  - stomp+nio+ssl://0.0.0.0:61614

DEMO