



ActiveMQ In Action

Common Problems and Solutions

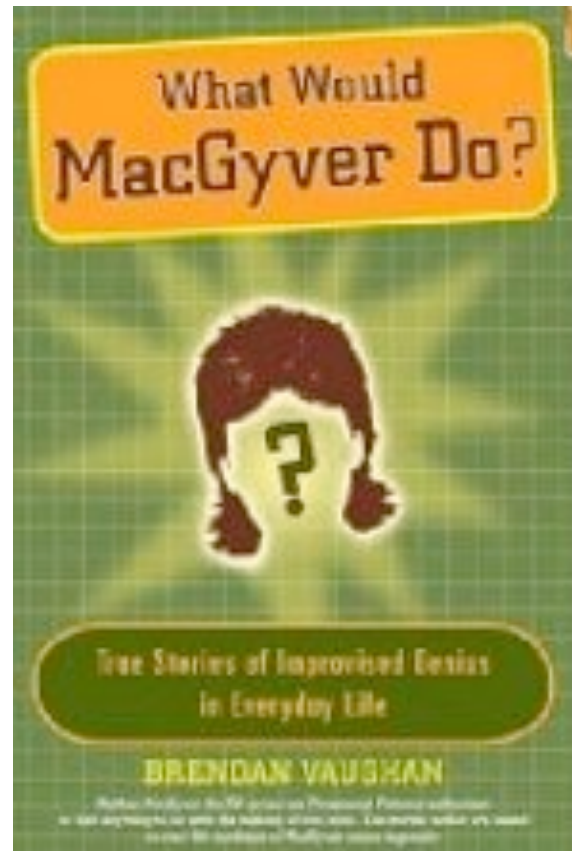
Bruce Snyder
VMware, Inc.
bsnyder@vmware.com

Common Questions

- Should I create my JMS clients from scratch?
- How do I manage connections efficiently?
- How do I consume only certain messages?
- Why is ActiveMQ locking up or freezing?

- **Bonus:** Do I need a network of brokers?
- **Bonus:** Should I use a master/slave configuration?

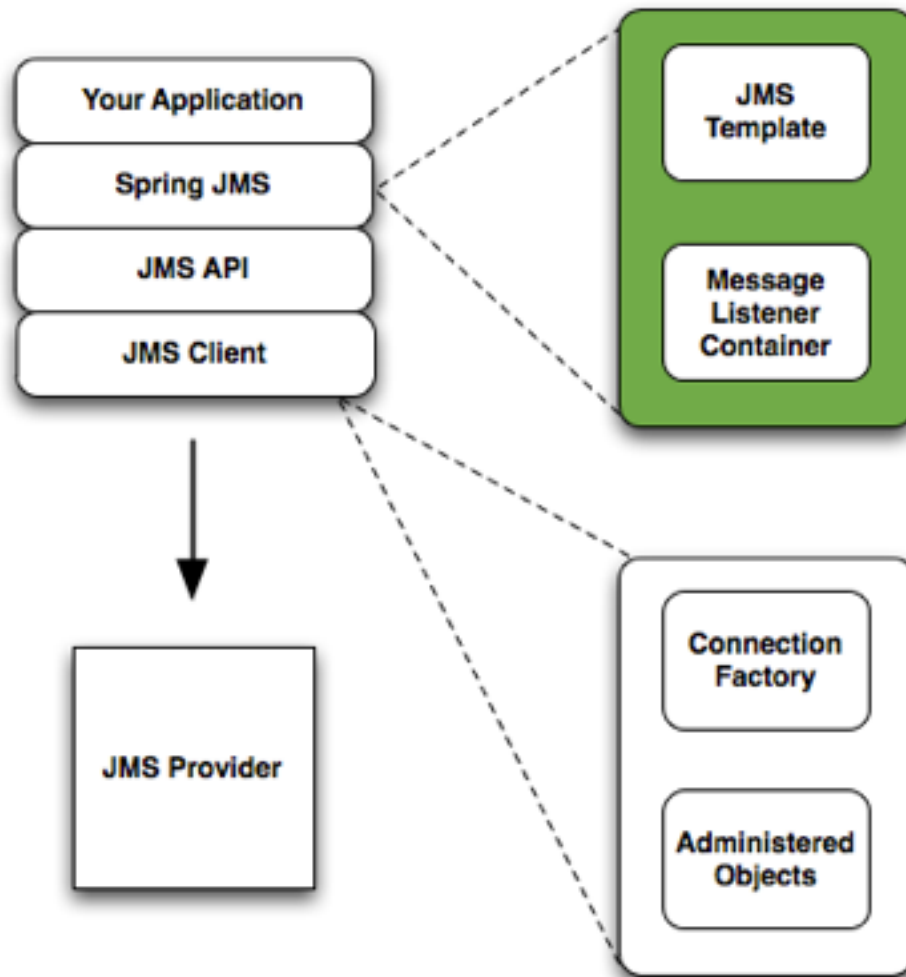
Should I create JMS clients from scratch?



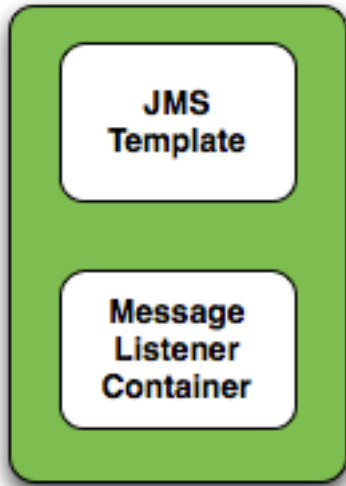
Should I create JMS clients from scratch?

- Question:
 - Would you create a HTTP client from scratch?
 - Would you create a SMTP client from scratch?
- Answer:
 - Sometimes, but mostly no
- Solution:
 - Use Spring JMS

Spring JMS

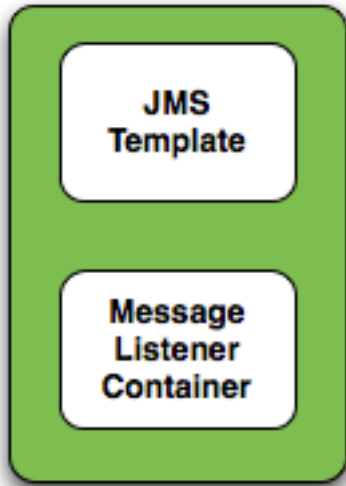


Spring JMS



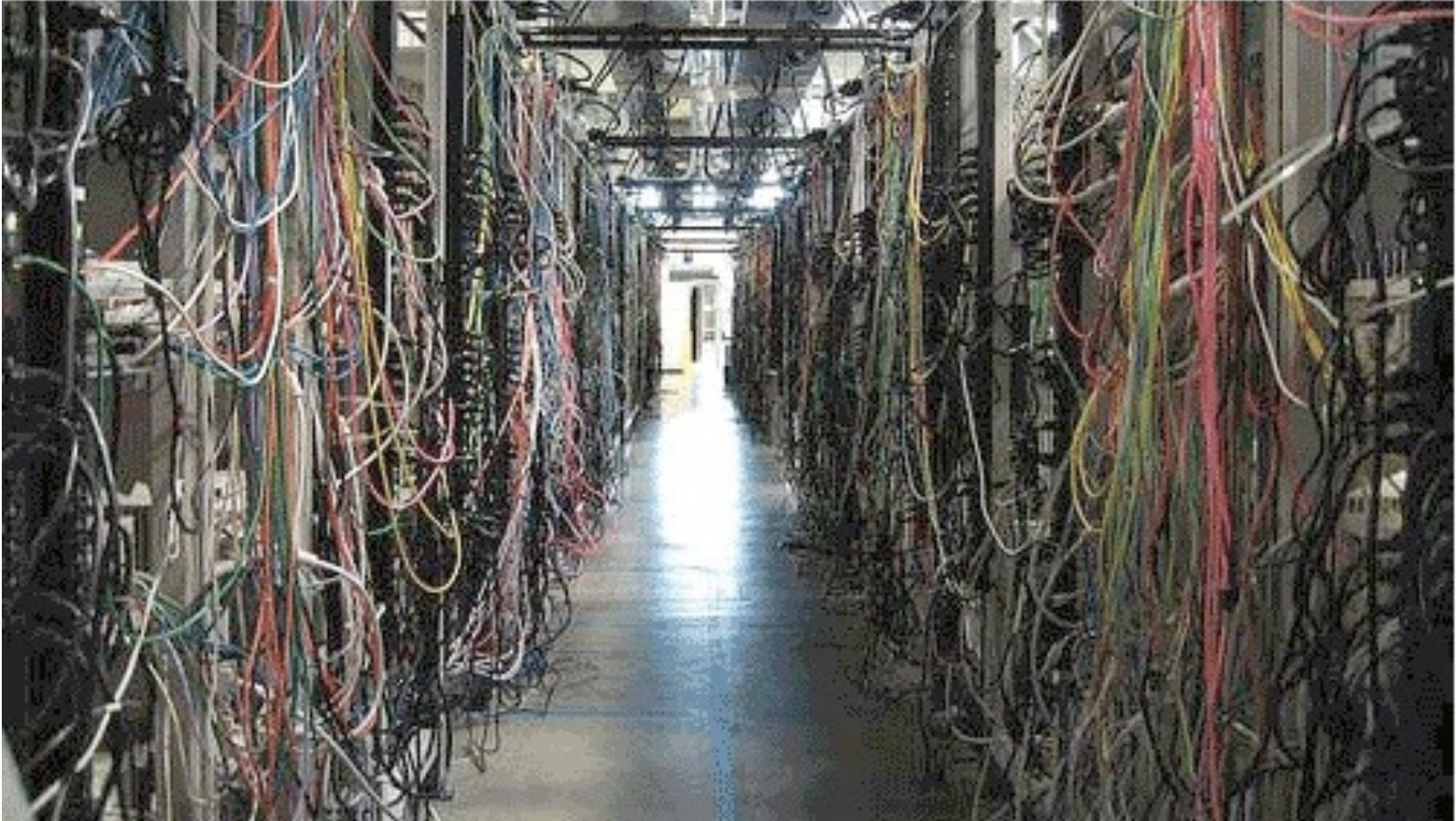
- **JMS Template**
 - Send and receive messages synchronously
- Message Listener Container
 - Receive messages asynchronously
 - Message-Driven POJOs (MDPs)

Spring JMS



- JMS Template
 - Send and receive messages synchronously
- **Message Listener Container**
 - Receive messages asynchronously
 - Message-Driven POJOs (MDPs)

How do I manage connections efficiently?



How do I manage connections efficiently?

- JMS connections are expensive to constantly create and destroy
- Create a group that never closes, i.e., pooling
- **Solutions:**
 - ActiveMQ PooledConnectionFactory
 - Spring CachingConnectionFactory

ActiveMQ PooledConnectionFactory

- Based on Apache Commons Pool
 - Generic object pooling framework from the ASF
- Highly configurable
 - Instantiate your own custom GenericObjectPool
- Could be improved
 - Upon hitting pool limit, grow the pool instead of blocking
 - Throw exception when the pool is exhausted
- Caches JMS Sessions and MessageProducers

Spring CachingConnectionFactory

- Based on Spring SingleConnectionFactory
 - Ignores calls to Connection.close()
- Caches JMS Sessions and MessageProducers
- By default only one session is cached
 - Increase the sessionCacheSize property!
- Consumers are not closed until Session is closed
 - NOTE: Cache strategy uses the JMS selector as a key



Quick Tip: Monitor the Broker

- Q: How can I monitor the message broker while I am developing?
- A: Manually via JMX
- A: Use the web console (backed by JMX)
- A: Use the advisory messages
 - Especially powerful
 - <http://activemq.apache.org/advisory-message.html>

How do I consume only certain messages?



ActiveMQ is not a database!

How do I consume only certain messages?

- ActiveMQ is for sending and receiving events
- ActiveMQ is NOT a message store
- Solutions:
 - Use message selectors
 - Correct application design

JMS Selectors

- Allows a client to filter messages from a destination
- Filters message headers only, not payload
- Conditional expressions using a subset of SQL
- Provide boolean evaluation of message headers

Literals	Booleans TRUE/FALSE; numbers such as 5, -10, +34; numbers with decimal or scientific notation such as 43.3E7, +10.5239
Identifiers	A header field
Operators	AND, OR, LIKE, BETWEEN, =, <>, <, >, <=, =>, +, =, *, /, IS NULL, IS NOT NULL

JMS Selector Examples

```
// Select messages with a header named symbol whose value is APPL
```

```
String selector = "symbol = 'APPL'";
```

```
// Create a consumer that only receives messages about Apple Computer stock
```

```
MessageConsumer consumer =
```

```
    session.createConsumer(someDestination, selector);
```

```
// Select messages with a header named symbol whose value is APPL
```

```
// and with a header named price that is greater than the previous price
```

```
String selector = "symbol = 'APPL' AND price > " + getPreviousPrice();
```

```
// Create a consumer that only receives messages about Apple Computer stock
```

```
// that has increased in price
```

```
MessageConsumer consumer =
```

```
    session.createConsumer(someDestination, selector);
```

JMS Selectors

- Very powerful, but like a sharp knife
- Applied to every message on a destination
 - Can cause unnecessary overhead

Correct Application Design

- ActiveMQ is for sending and receiving events
- ActiveMQ is NOT a message store

- Phase one, consume the messages
 - Lightweight processing
- Phase two, conduct further processing
 - Heavyweight processing

- I.e., a proper service-oriented architecture



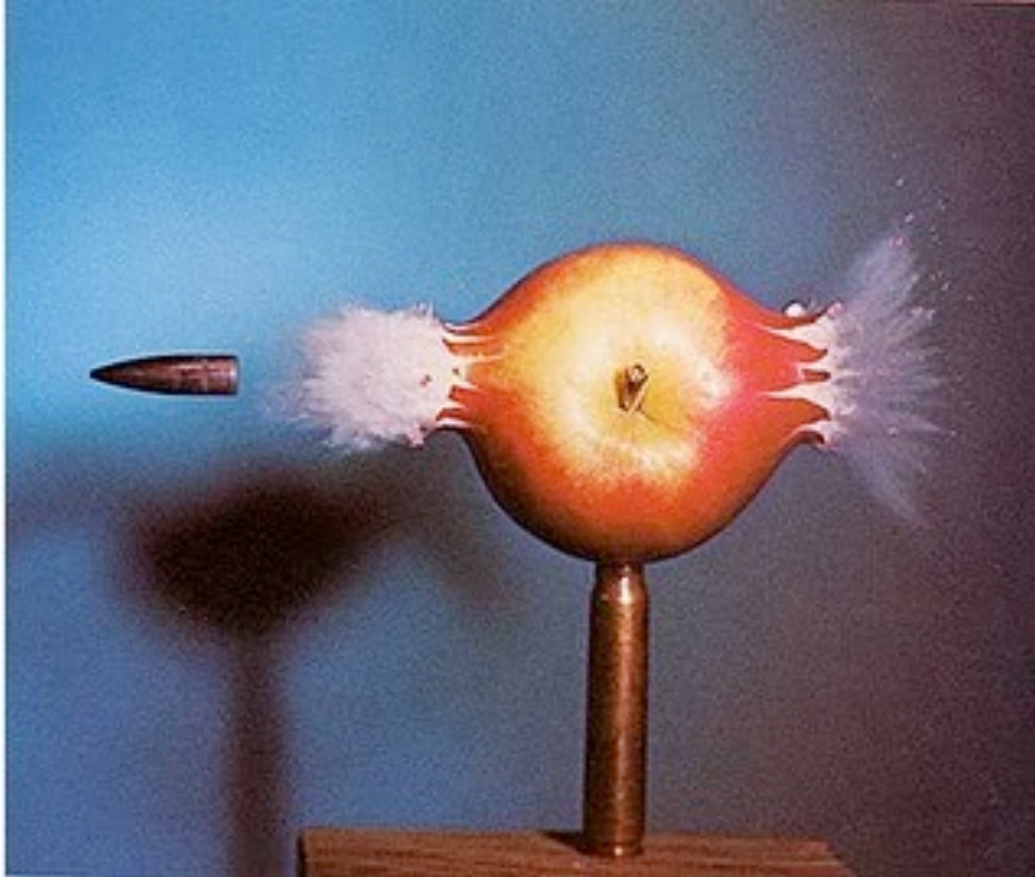
Quick Tip: Clearing the DLQ

- **Q:** Is there a way to automatically clear messages in the DLQ?
- **A:** Use the DiscardingDLQ plugin or create a custom consumer

```
<broker brokerName="myBroker" ...>
  <plugins>
    <discardingDLQBrokerPlugin dropAll="true" dropTemporaryTopics="true"
      dropTemporaryQueues="true" />
  </plugins>
</broker>
```

```
<broker brokerName="myBroker" ...>
  <plugins>
    <discardingDLQBrokerPlugin
      dropOnly="TEST.FOO.[1-9] EXAMPLE.TOPIC"
      reportInterval="5000" />
  </plugins>
</broker>
```

Why is ActiveMQ is locking up or freezing?



Why is ActiveMQ is locking up or freezing?

- JVM memory
- Broker memory
- Prefetch limit
- Producer flow control
- Message cursors

JVM Memory

- ActiveMQ start script
 - In 5.4.x, JVM is given 256mb of memory (min and max)
- You may need to increase this!

Broker Memory

- ActiveMQ controls how much memory it can use
- Will not automatically use all the JVM memory
- Configurable but commented out by default

Broker Memory Example

```
<broker brokerName="myBroker" ...>
...
<systemUsage>
  <systemUsage>
    <memoryUsage>
      <memoryUsage limit="64 mb" />
    </memoryUsage>
    <storeUsage>
      <storeUsage limit="100 gb" />
    </storeUsage>
    <tempUsage>
      <tempUsage limit="10 gb" />
    </tempUsage>
  </systemUsage>
</systemUsage>
...
</broker>
```

Prefetch Limit

- Prevents a consumer from being flooded with messages
- Applied on a per client basis
- Incorrect prefetch limit + slow consumer = messages remain in a queue unconsumed
- Results in some consumers being starved of messages
- NOTE: Be careful with connection pools

Prefetch Limit Example

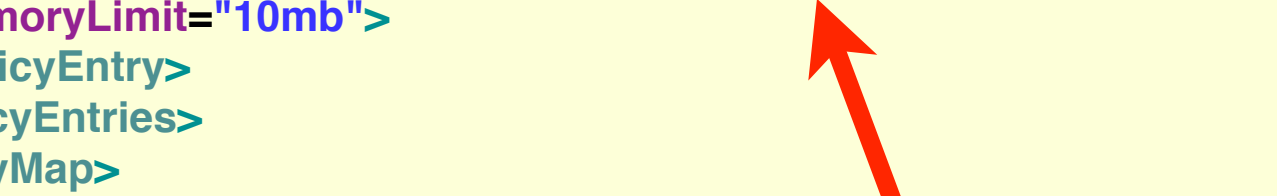
```
...  
<bean id="connectionFactory"  
  class="org.apache.activemq.ActiveMQConnectionFactory"  
  p:brokerURL="tcp://localhost:61616"  
  p:prefetchPolicy-ref="prefetchPolicy"/>  
  
<bean id="prefetchPolicy"  
  class="org.apache.activemq.ActiveMQPrefetchPolicy"  
  p:queuePrefetch="1" />  
...
```

Producer Flow Control

- Prevents producer from flooding broker
- If memory exceeds limit, a producer will be paused
- NOTE: This setting is enabled by default

Broker Memory Example

```
<broker brokerName="myBroker" ...>
...
<destinationPolicy>
  <policyMap>
    <policyEntries>
      <policyEntry topic="" producerFlowControl="true"
        memoryLimit="10mb">
        <pendingSubscriberPolicy>
          <vmCursor />
        </pendingSubscriberPolicy>
      </policyEntry>
      <policyEntry queue="" producerFlowControl="true"
        memoryLimit="10mb">
      </policyEntry>
    </policyEntries>
  </policyMap>
</destinationPolicy>
...
</broker>
```

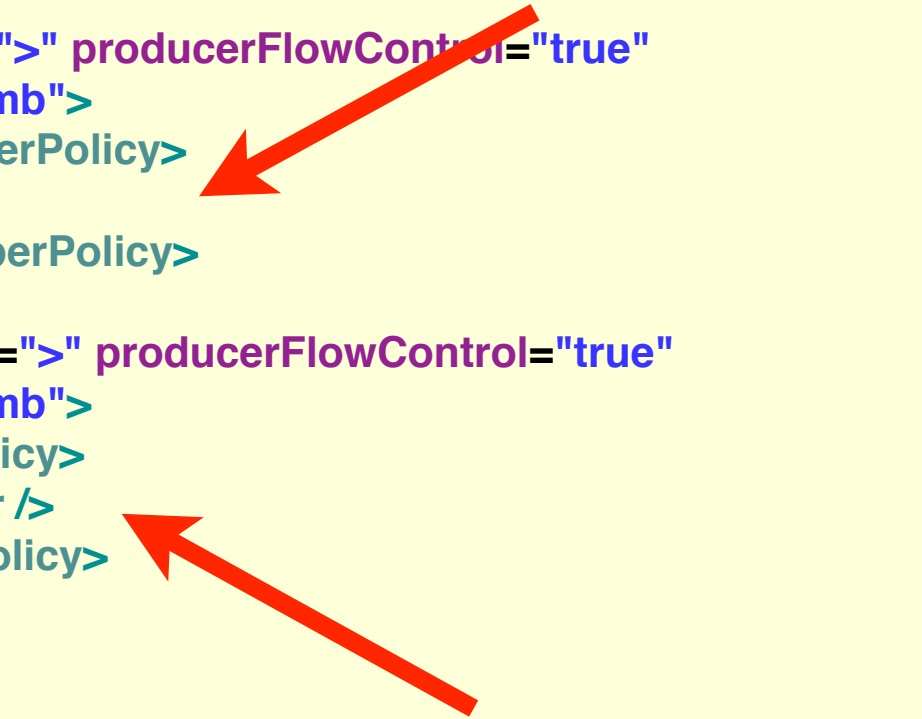


Message Cursors

- Only so many messages can be held in memory
- Message cursors provide a configurable message paging
- Two types of cursors
 - VM cursors
 - Holds only message reference in memory
 - File cursors
 - Flushes both message and reference to disk
- <http://activemq.apache.org/how-do-i-configure-activemq-to-hold-100s-of-millions-of-queue-messages-.html>

Message Cursor Example

```
<broker brokerName="myBroker" ...>
...
<destinationPolicy>
  <policyMap>
    <policyEntries>
      <policyEntry topic="" producerFlowControl="true"
        memoryLimit="10mb">
        <pendingSubscriberPolicy>
          <vmCursor />
        </pendingSubscriberPolicy>
      </policyEntry>
      <policyEntry queue="" producerFlowControl="true"
        memoryLimit="10mb">
        <pendingQueuePolicy>
          <fileQueueCursor />
        </pendingQueuePolicy>
      </policyEntry>
    </policyEntries>
  </policyMap>
</destinationPolicy>
...
</broker>
```





Quick Tip: Rebalancing Clients

- **Q:** After restarting one of my message brokers, how can I force clients to connect to that broker?
- **A:** Enforce a deterministic lifetime on a connection by setting the **expiryTimeout** on the pool of connections

```
<bean id="connectionFactory"
  class="org.apache.activemq.pool.PooledConnectionFactory"
  <property name="brokerURL" value="tcp://localhost:61616" />
  <property name="expiryTimeout" value="10000" />
  <property name="connectionFactory">
    <bean id="connectionFactory"
      class="org.apache.activemq.ActiveMQConnectionFactory"
      p:brokerURL="tcp://localhost:61616" />
    </property>
  </bean>
```

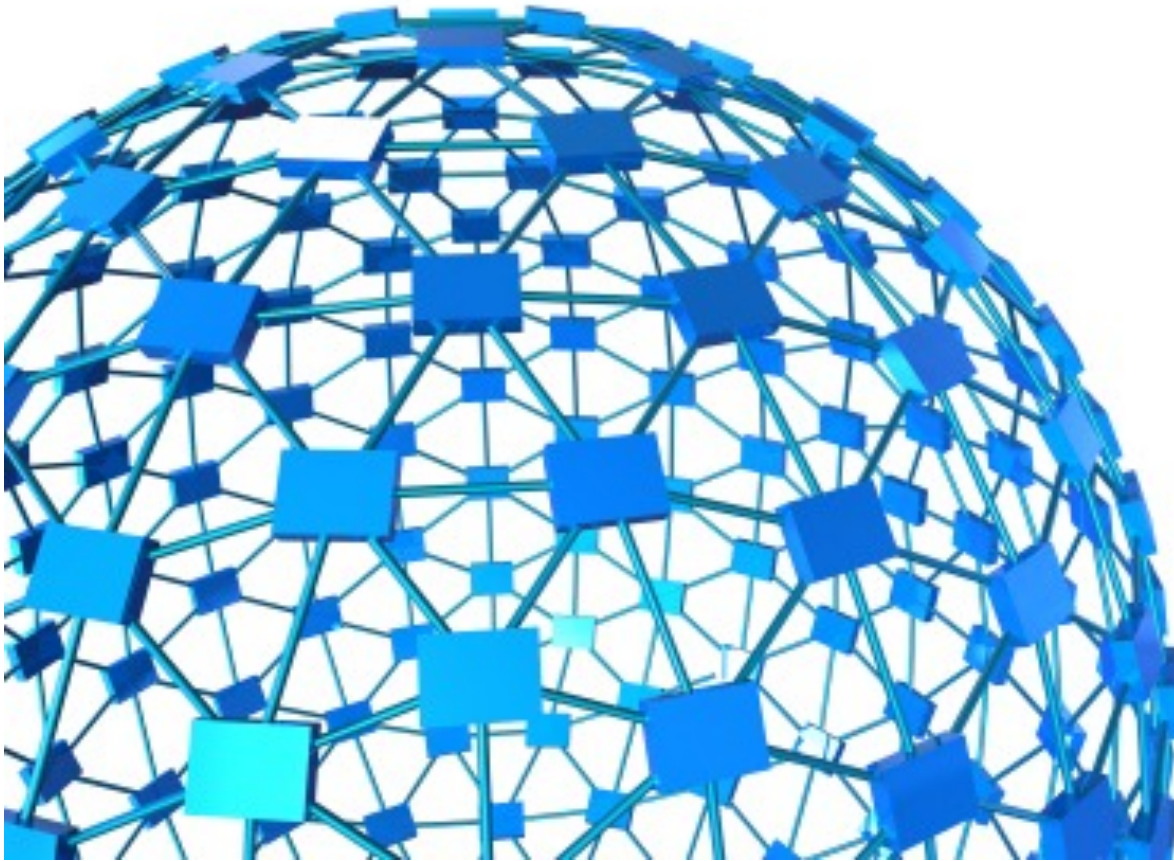
Thank You!

Q&A

A close-up photograph of several ears of yellow corn. The corn cobs are arranged diagonally across the frame. The kernels are bright yellow and appear plump. Green husks are visible, partially covering the cobs. The background is dark, making the yellow corn stand out.

Bonus Material!

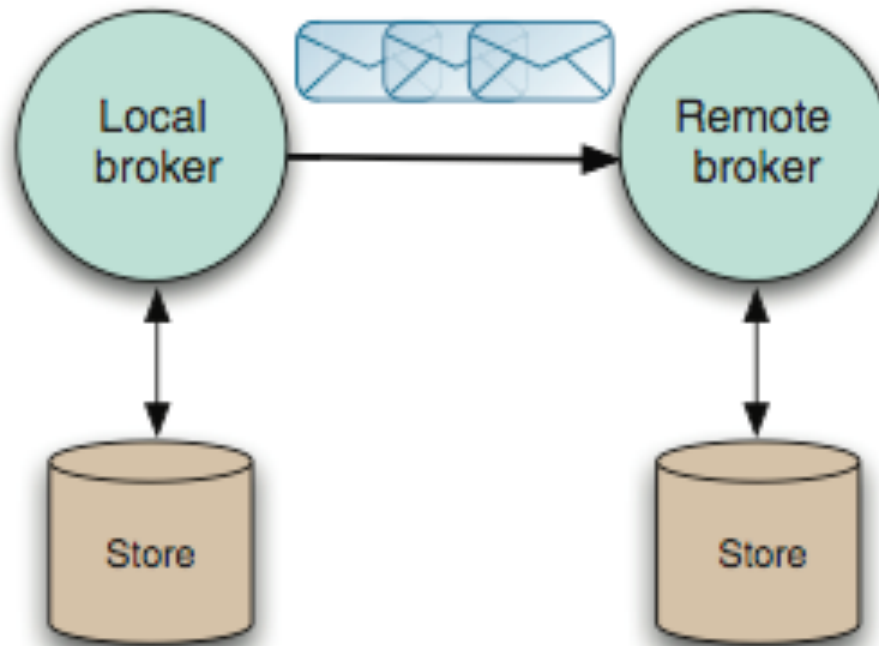
Do I need a network of brokers?



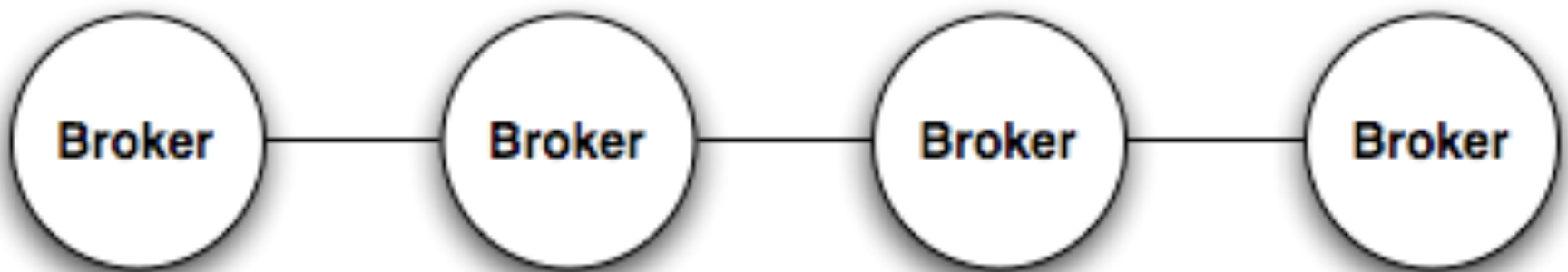
Do I need a network of brokers?

- What is a network of brokers?
 - Clustered ActiveMQ instances
- How are they clustered?
 - They pass messages between broker instances
 - Send a message to one broker, consume the message from a different broker
- Where might this be useful?
 - Situations where a centralized broker is not suitable
- How does this work?
 - Using store and forward

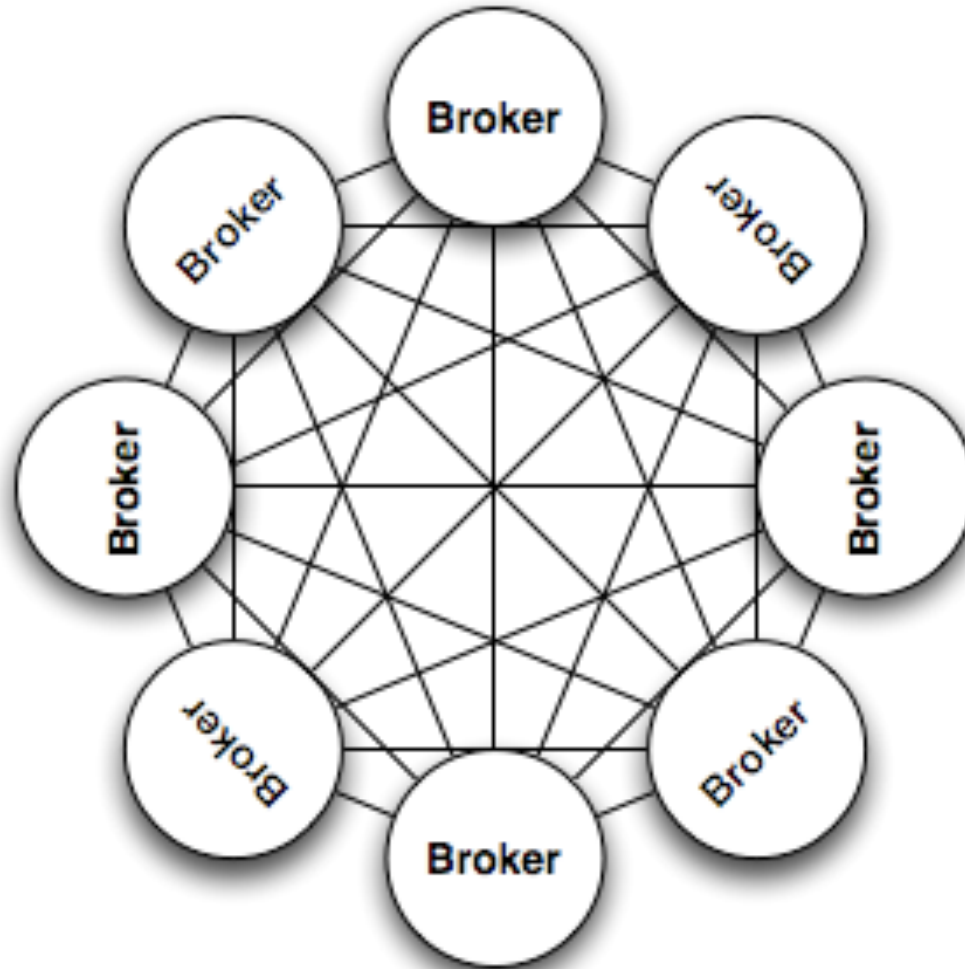
Store and Forward



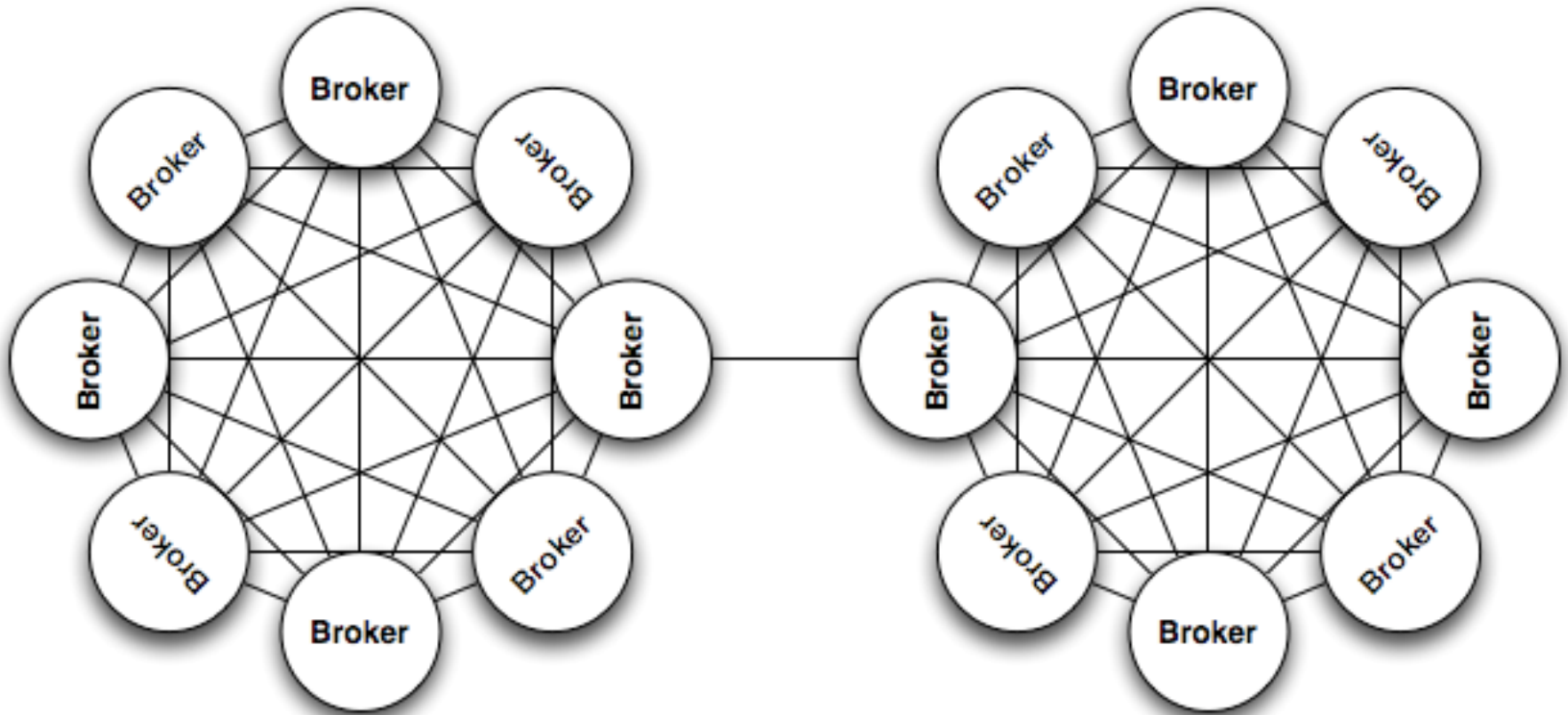
Topology Example



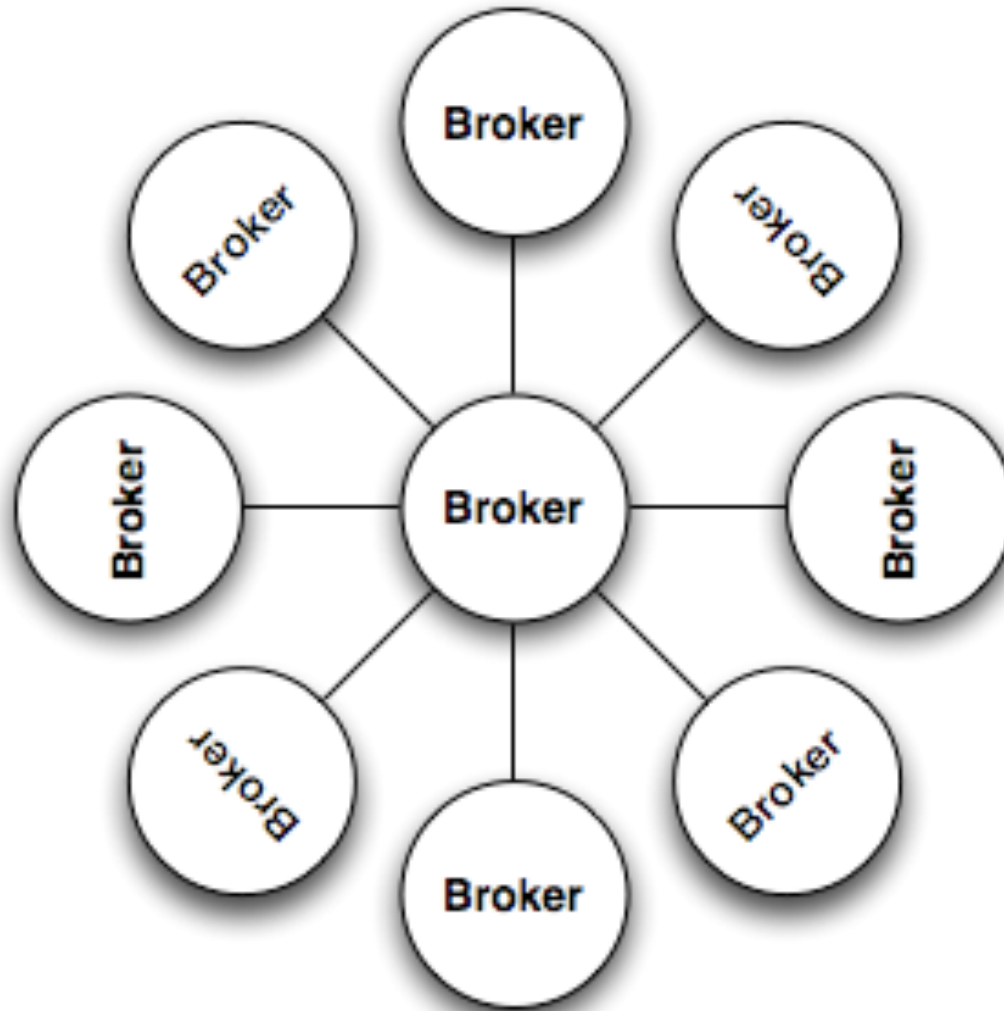
Topology Example



Topology Example



Topology Example



Topology Example



Should I use a master/slave config?



Should I use a master/slave config?

- What is a master/slave configuration?
 - It helps to provide high availability for ActiveMQ
- What does that mean?
 - ActiveMQ brokers are configured for warm failover
 - If one broker fails or becomes unreachable, another one takes over
- Where might this be useful?
 - In situations that need highly available message brokers
- How does this work?
 - Depends on the type of master/slave

Types of Master/Slave

- Shared nothing master/slave
- Shared storage master/slave
 - Shared database
 - Shared file system

Shared Nothing Master/Slave

- Sometimes called pure master/slave
- Uses a fully replicated data store
 - Does not depend on database or file system
- Slave broker consumes all message states from the Master broker (messages, acks, tx states)
- Slave does not start any networking or transport connectors
- Master broker will only respond to client when a message exchange has been successfully passed to the slave broker

Shared Nothing Master/Slave

- If the master fails, the slave optionally has two modes of operation:
 1. Start up all it's network and transport connectors
 - All clients connected to failed Master resume on Slave
 2. Close down completely
 - Slave is simply used to duplicate state from Master
- Clients should use failover transport:

```
failover://(tcp://masterhost:61616, tcp://slavehost:61616)?randomize=false
```

Shared Database Master/Slave

- Uses tables in a relational database to store data
- No restriction on the number of brokers
- Simple configuration (JDBC URL)
- Clustered database mitigates single point of failure
- One master selected at random

- Clients should use failover transport:

```
failover://(tcp://masterhost:61616, tcp://slavehost:61616)?randomize=false
```

Shared File System Master/Slave

- Utilizes a directory on a shared file system to store data
- No restriction on number of brokers
- Simple configuration (point to the data dir)
- Shared file system mitigates single point of failure
- One master selected at random
- Clients should use failover transport:

```
failover://(tcp://masterhost:61616, tcp://slavehost:61616)?randomize=false
```

Should I use a master/slave config?

- Are you trying to provide high availability?
 - Then, yes
- Which one should I use?
 - It depends on your situation