Enterprise Integration: Patterns and Deployments

Scott Cranton
Principal Solution Engineer
FuseSource
CamelOne 2011

FuseSource
A Progress Software Company

Enterprise Integration

Many flavors of integration - how do we make them fit together?

Apache Camel has to be the Starting Point

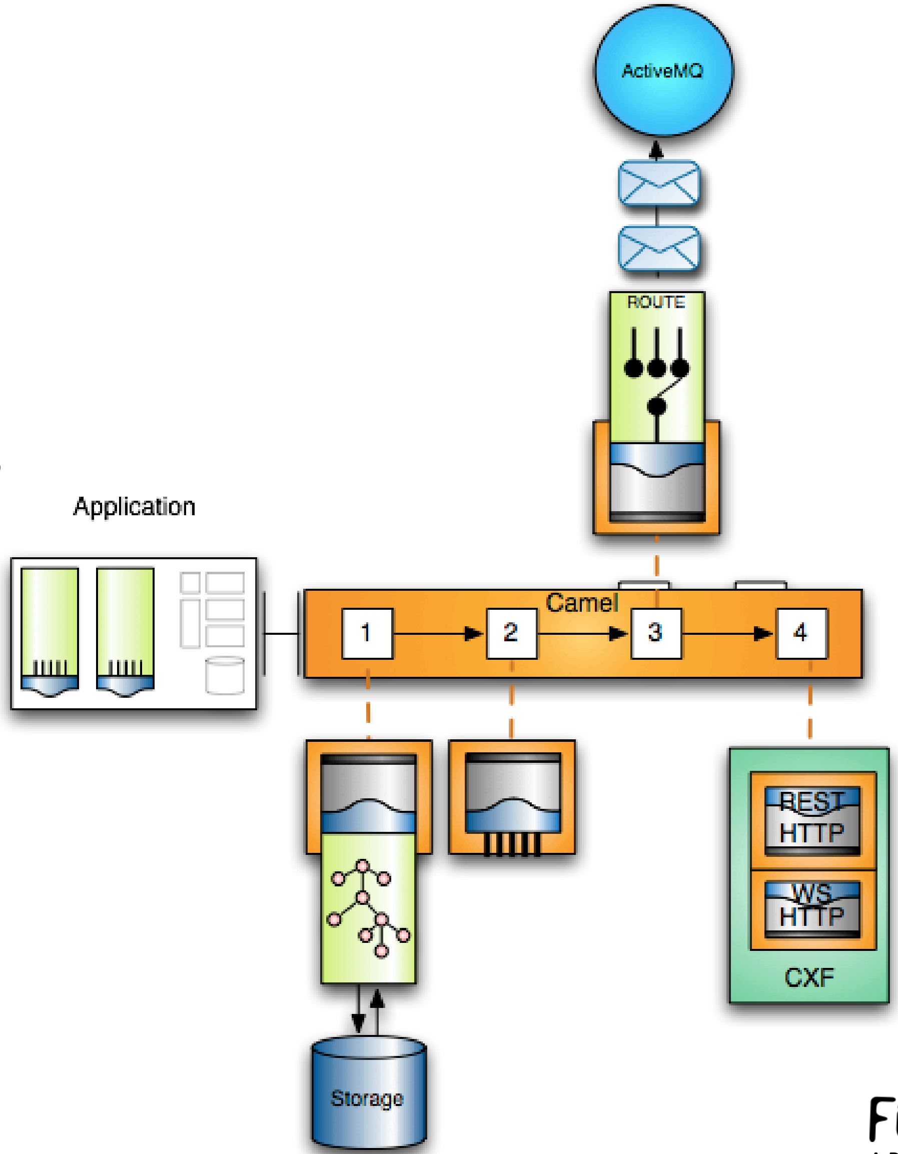
- Its framework, designed to be embedded - which allows us to build out a story with Apache ActiveMQ, ServiceMix and CXF
- Intuitive domain specific language for integration, using Enterprise Integration Patterns (EIP) - which is why its so successful
- Over 90 integration components - and growing
- Wide adoption across open source and closed source (e.g. JBoss, Progress, etc.)

Apache Camel: Integration Glue

Apache Camel provides immense flexibility, and seamless transforms between different message formats

It already integrates well with

CXF and ActiveMQ

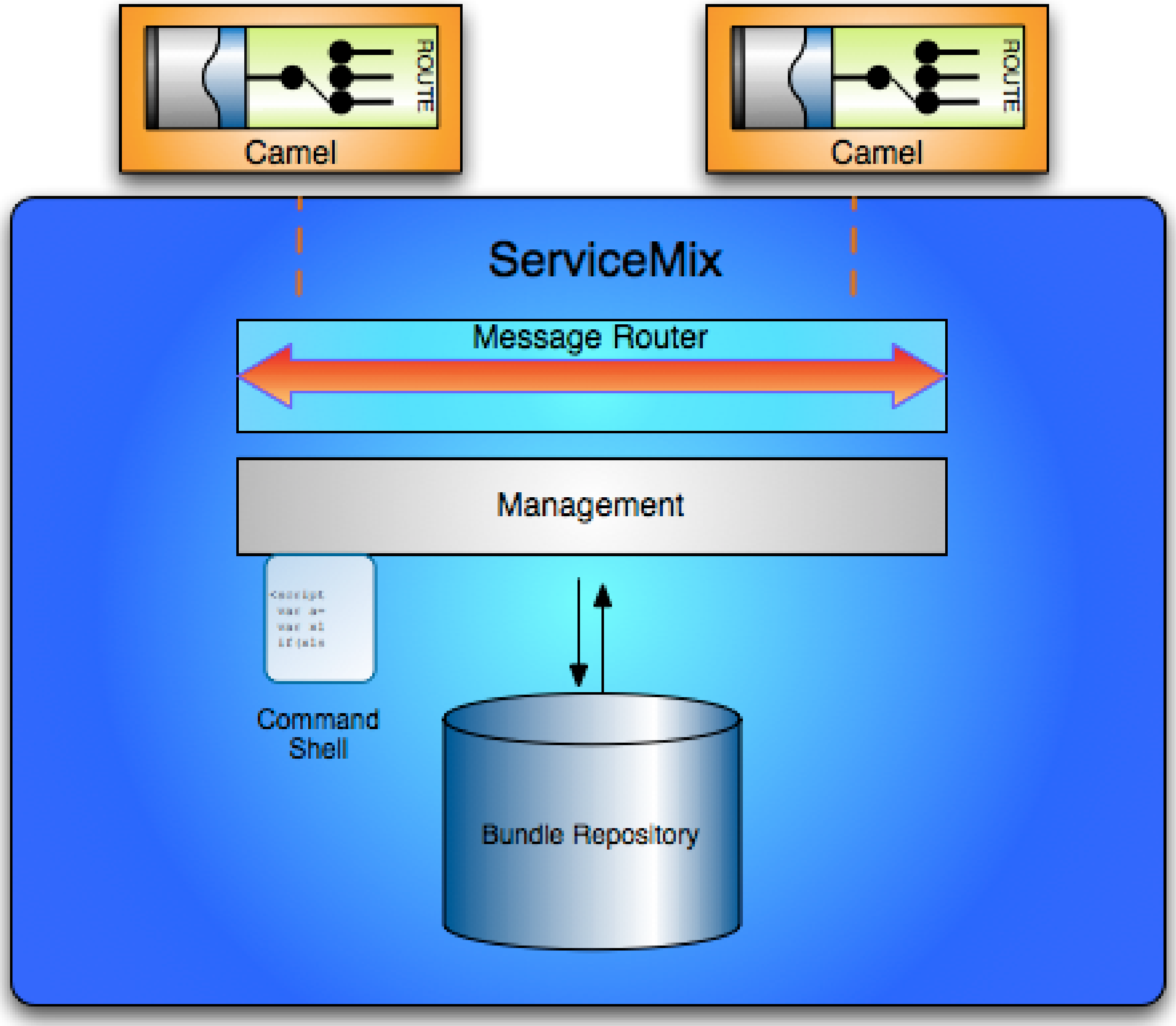


How to do make Camel work in the Enterprise? … ServiceMix

ServiceMix is *the integration container of choice.*

Start with Camel, but

for Enterprise deployments, use ServiceMix



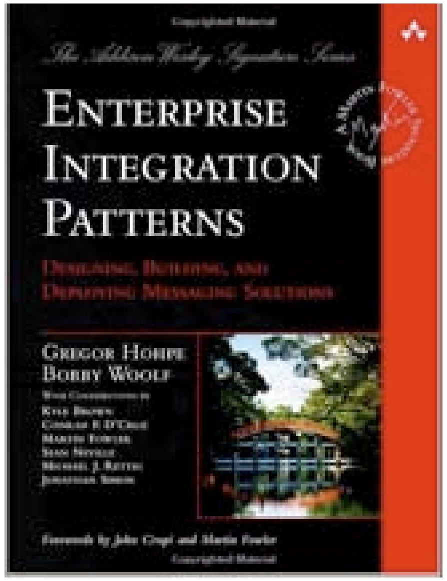
Enterprise Integration

Integration is all about patterns - lets look at:

Apache ActiveMQ …

Enterprise Integration Patterns

- Book by Gregor Hohpe and Booby Woolf
- Patterns and Recipes for common integration problems
- Message Centric
- Used as the basis for all the major integration products
- Should be the the first thing to reference when starting an integration project
- http://www.eaipatterns.com/


The Addison-Wesley Signature Series

ENTERPRISE INTEGRATION PATTERNS

DESIGNING, BUILDING, AND DEPLOYING MESSAGING SOLUTIONS

GREGOR HOHPE
BOBBY WOOLF

Forewords by John Crupi and Martin Fowler

FuseSource
A Progress Software Company

Some Integration Patterns

Message Routing

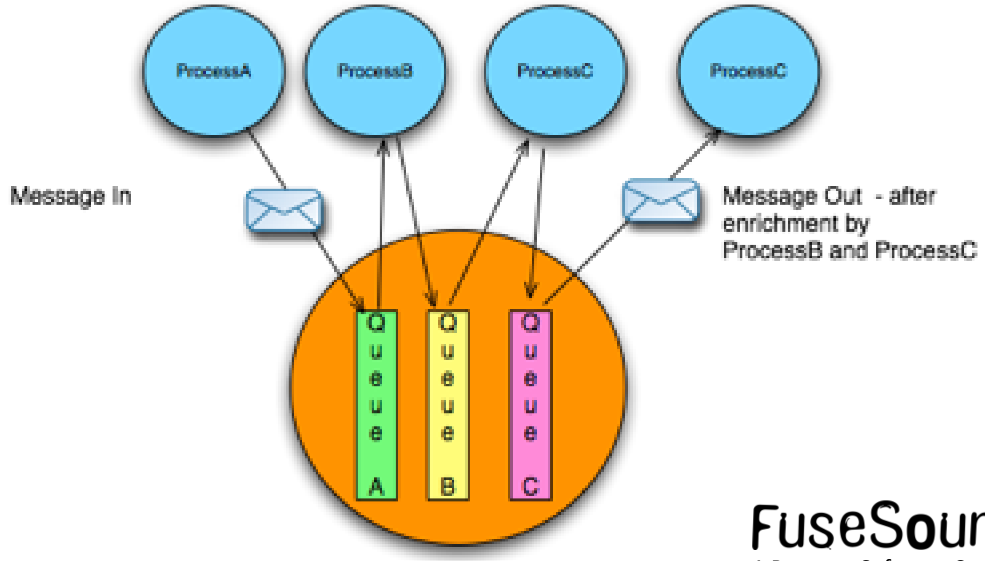
	Content Based Router	How do we handle a situation where the implementation of a single logical function (e.g., inventory check) is spread across multiple physical systems?
	Message Filter	How can a component avoid receiving uninteresting messages?
	Recipient List	How do we route a message to a list of dynamically specified recipients?
	Splitter	How can we process a message if it contains multiple elements, each of which may have to be processed in a different way?
	Aggregator	How do we combine the results of individual, but related messages so that they can be processed as a whole?
	Resequencer	How can we get a stream of related but out-of-sequence messages back into the correct order?
	Throttler	How can I throttle messages to ensure that a specific endpoint does not get overloaded, or we don't exceed an agreed SLA with some external service?
	Delayer	How can I delay the sending of a message?

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



What is Apache Camel?

- Top level Apache Software Foundation Project
- Mediation Router/Integration Framework
- Designed to:
  - Have no container dependency
  - But ... work very well with ActiveMQ, ServiceMix and CXF
  - Can integrate seamlessly with Spring
  - Implements All the Enterprise Integration Patterns
  - Breadth of Connectivity Options

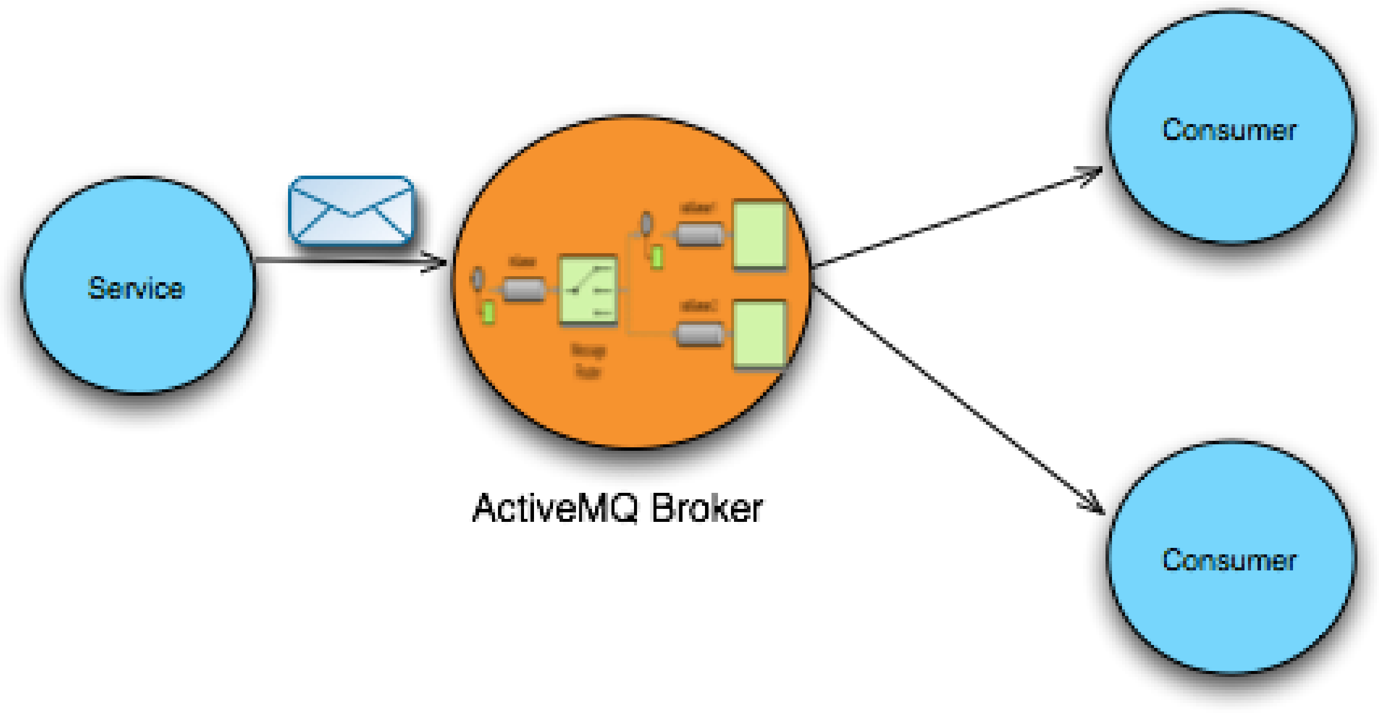
Using Apache Camel to Hide Middleware - Consume Annotation

```
public class Foo {

  @Consume(uri="activemq:cheese")
  public void onCheese(String name) {
    ...
  }

}
```

Using Apache Camel to Hide Middleware - Produce Annotation

```
public interface MyListener {
    String sayHello(String name);
}

public class MyBean {

    @Produce(uri = "activemq:foo")
    protected MyListener producer;

    public void doSomething() {
        // lets send a message
        String response = producer.sayHello("Mom");
    }

}
```

ActiveMQ with Embedded Camel Flexible and Performant



ActiveMQ with Embedded Camel
Import Camel into ActiveMQ broker config

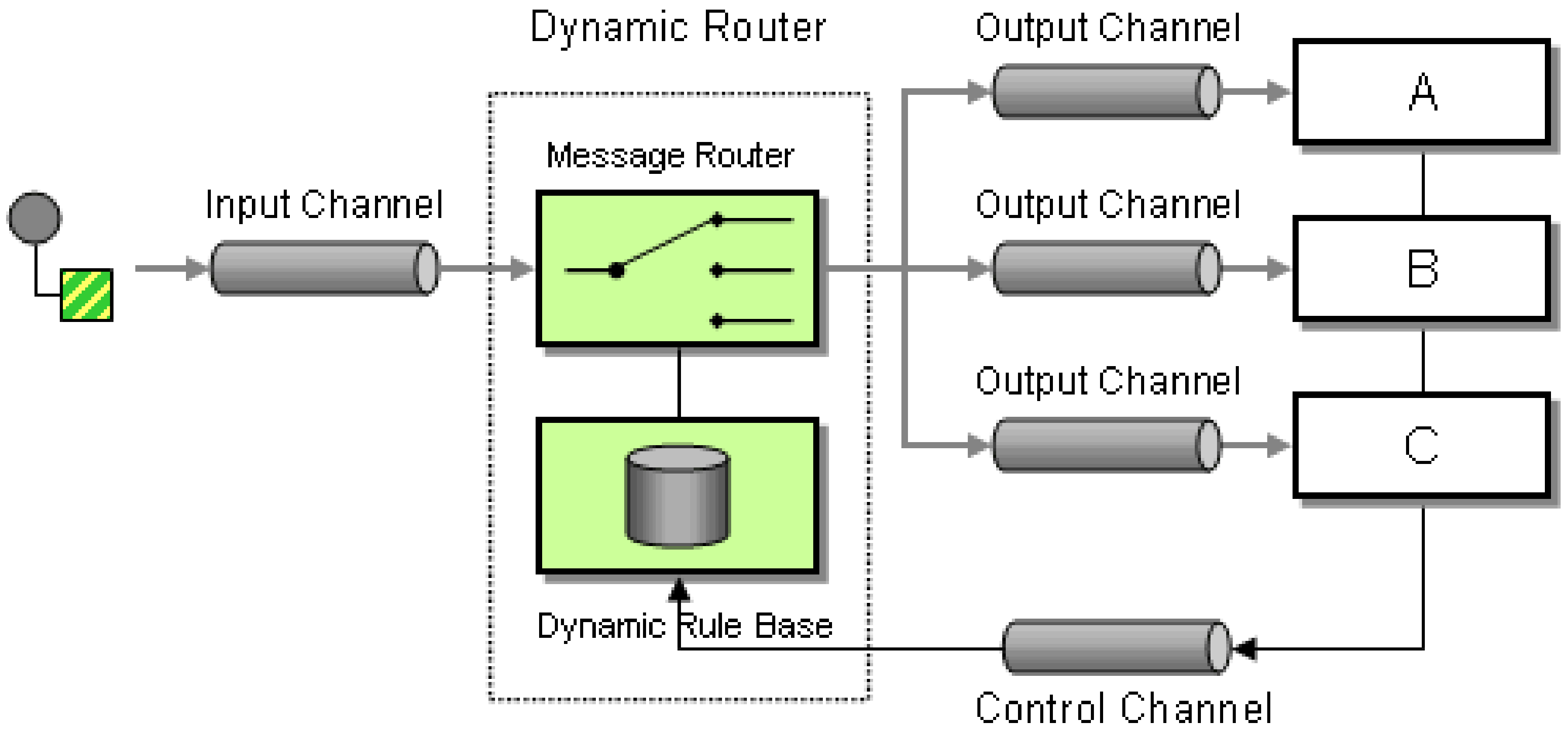
```
<beans>

   <broker brokerName="testBroker"
              xmlns="http://activemq.apache.org/schema/core">
    <transportConnectors>
      <transportConnector uri="tcp://localhost:61616"/>
    </transportConnectors>
  </broker>

  <import resource="camel.xml"/>

</beans>
```

ActiveMQ with Embedded Camel
Setup Camel Context in usual way

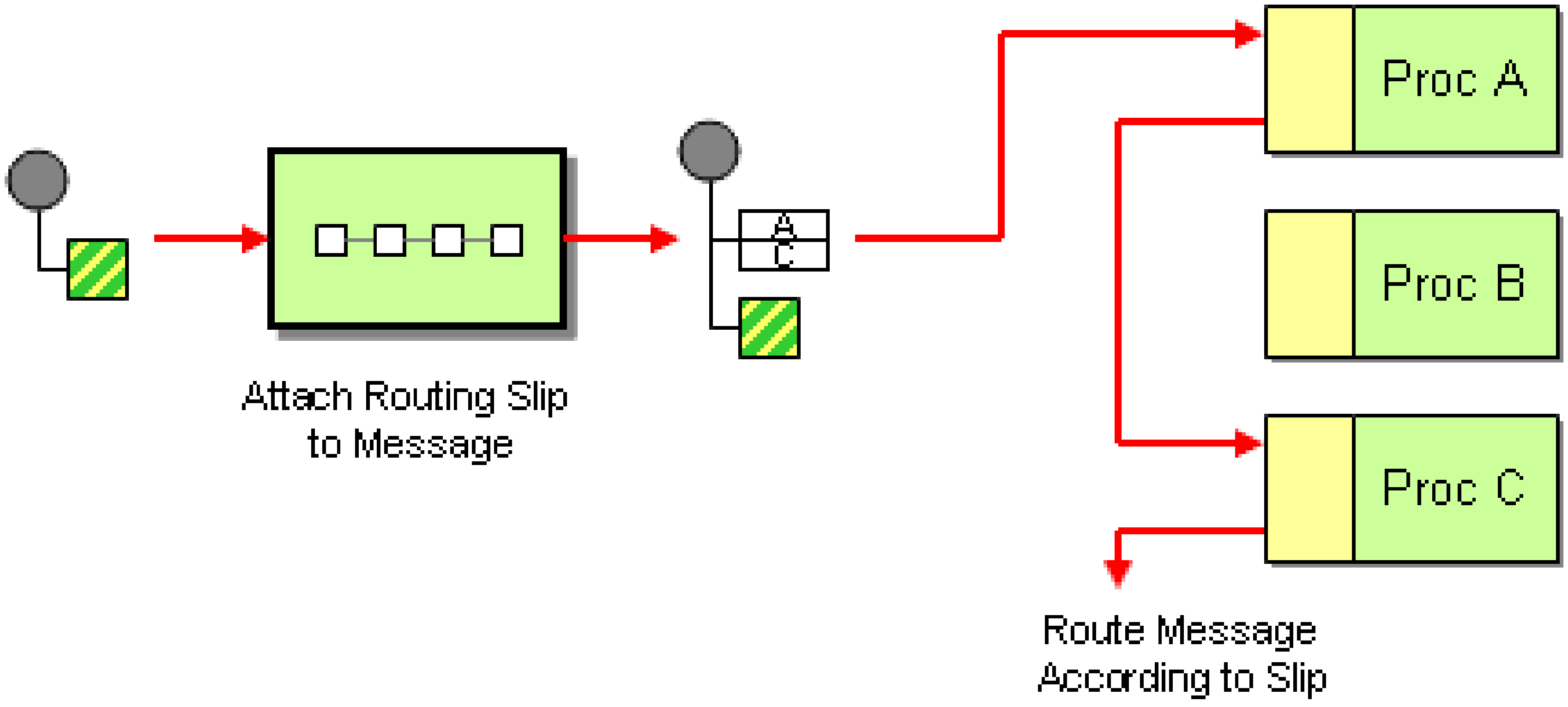
```
<camelContext errorHandlerRef="errorHandler"
         xmlns="http://camel.apache.org/schema/spring">
    <route>
        <from uri="activemq:queue:test.queue"/>
        <choice>
            <when>
                <xpath>$foo = 'bar'</xpath>
                <to uri="activemq:topic:topic.bar"/>
            </when>
            <when>
                <xpath>$foo = 'cheese'</xpath>
                <to uri="activemq:topic:topic.cheese"/>
            </when>
            <otherwise>
                <to uri="activemq:topic:topic.all"/>
            </otherwise>
        </choice>
    </route>
</camelContext>
```

Some patterns that are useful inside ActiveMQ …

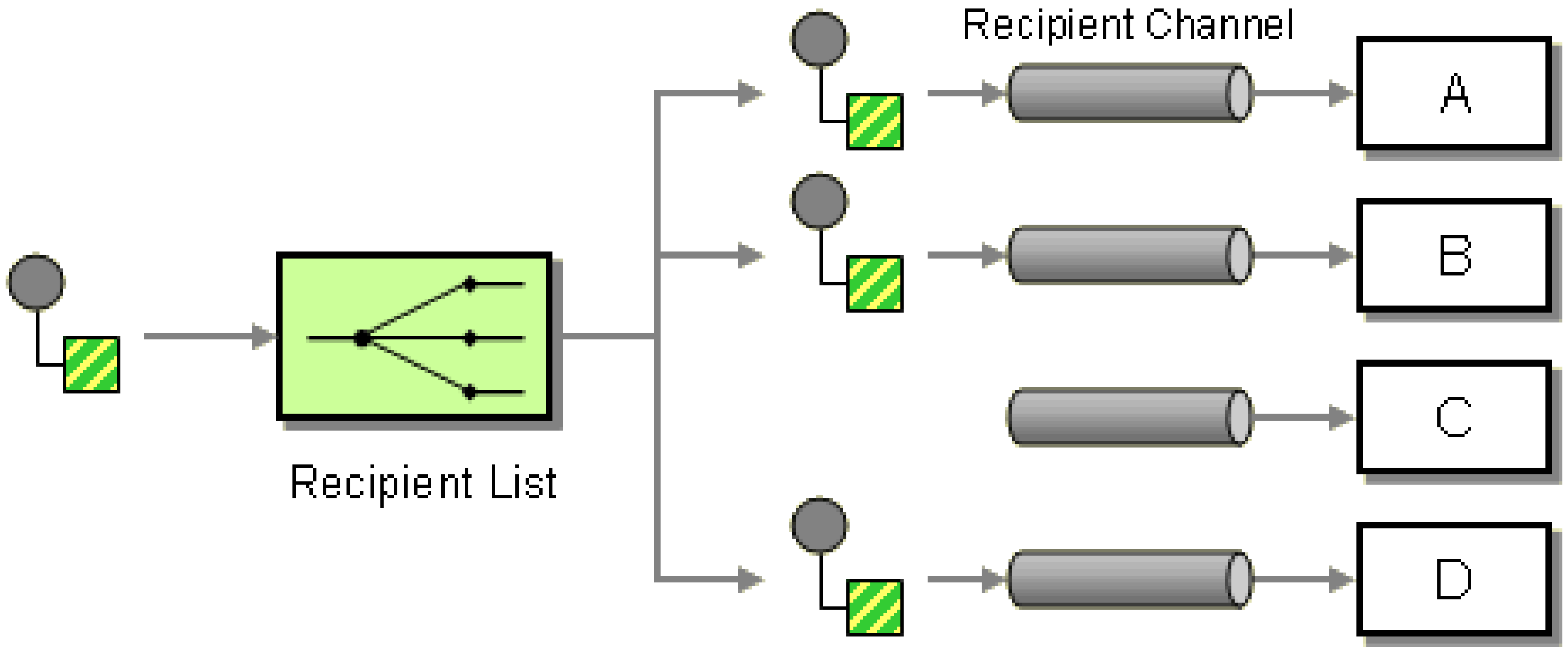
Dynamic Router: Flexible routing



Routing Slip


A
C
Attach Routing Slip to Message
Proc A
Proc B
Proc C
Route Message According to Slip

Recipient List

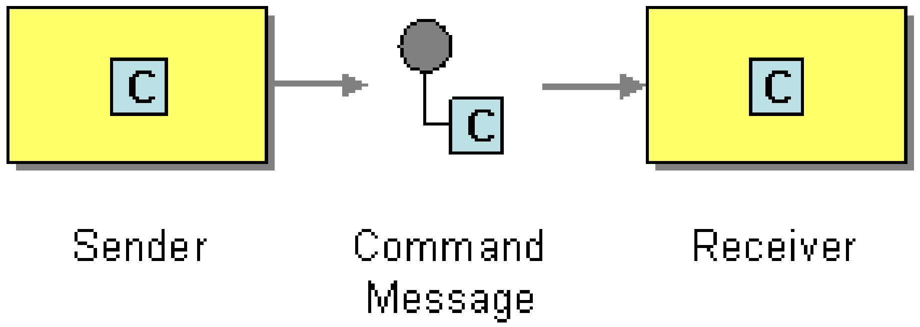


FuseSource
A Progress Software Company

Enterprise Integration

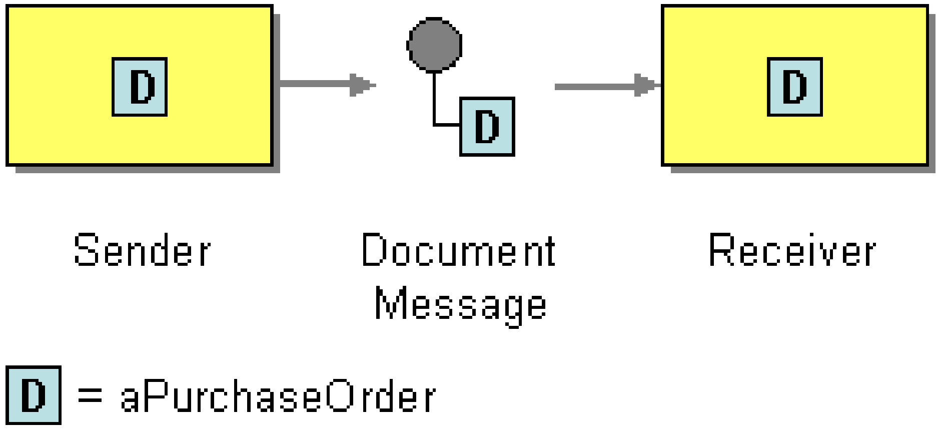
Types of Message and their uses …

Types of Message: Command

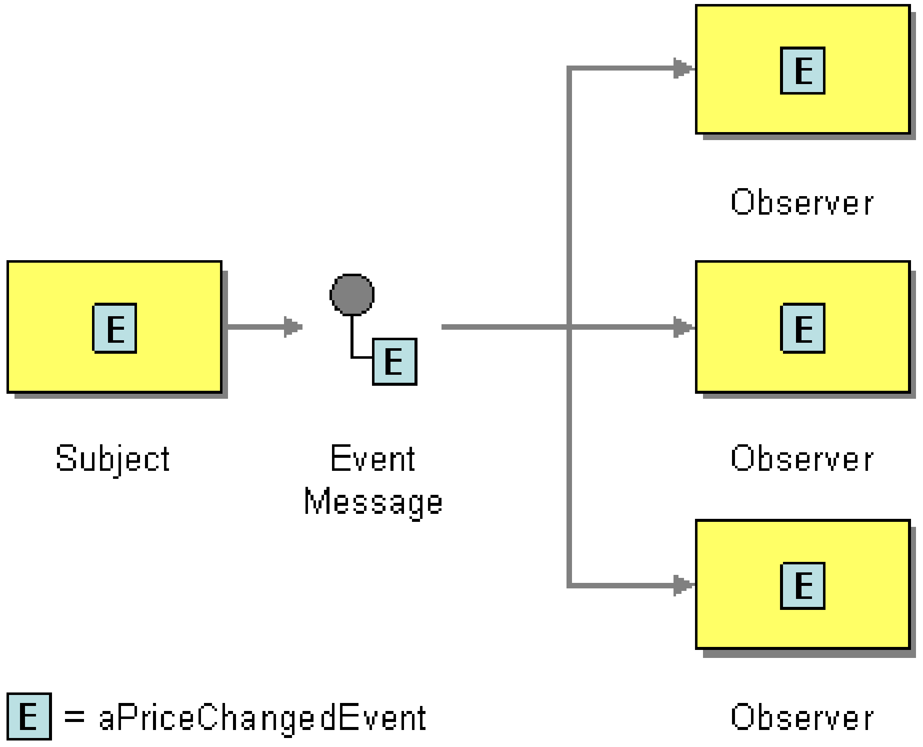


C = getLastTradePrice("DIS");

Types of Message: Document

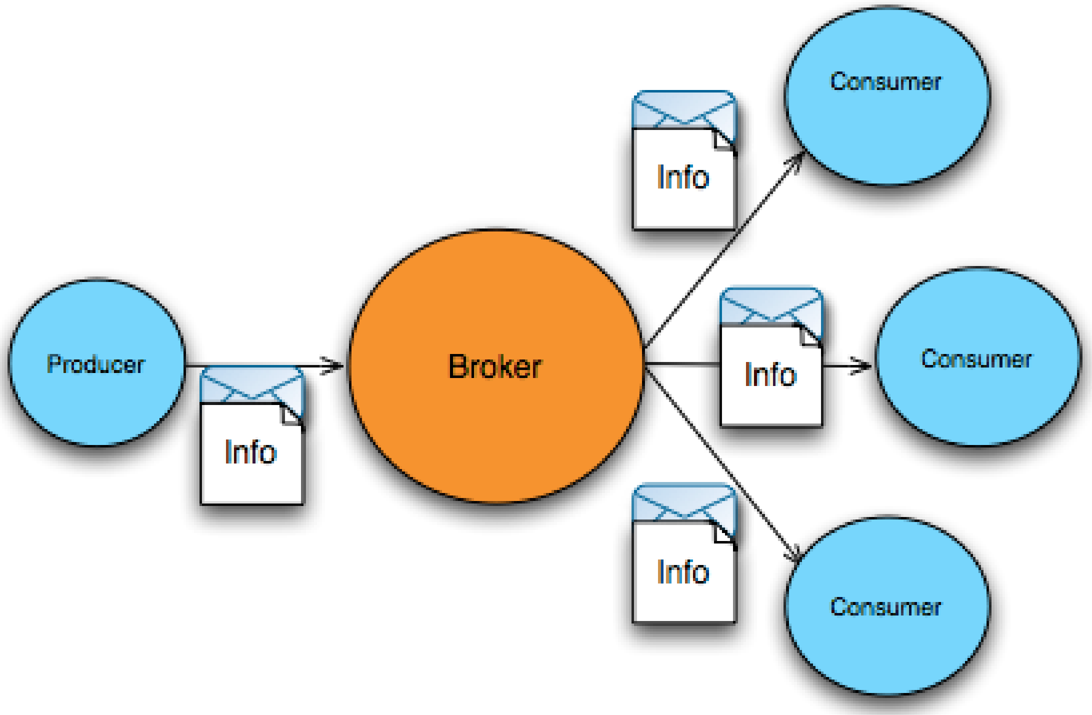


Types of Message: Event

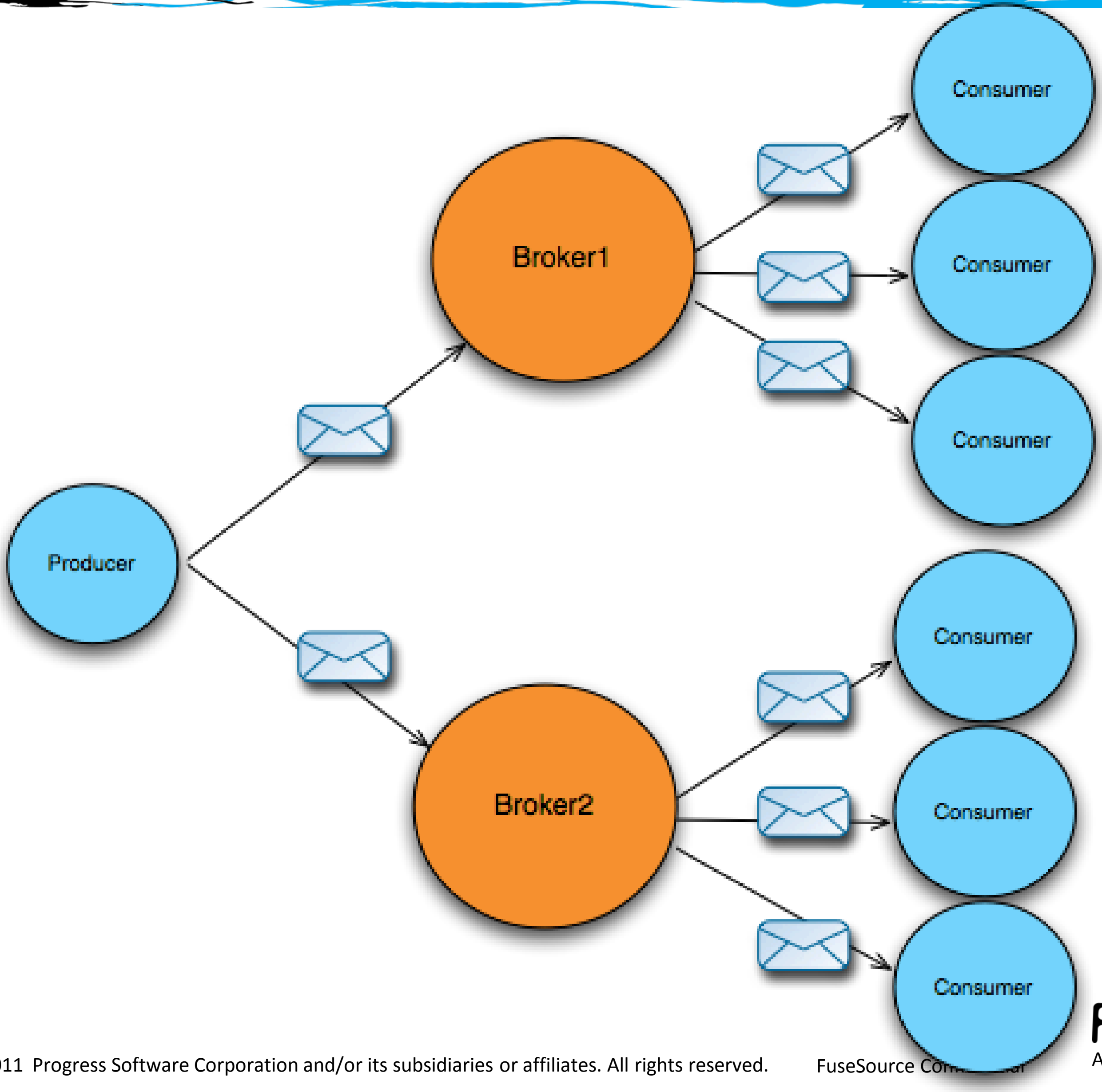


Push Model for Integration

- Typically uses a document message being sent as an event
  - Information about a change (e.g. a price change) is an event
  - Information about a change and the changed information - is an event/document combination



Push Model Using ActiveMQ for Scalability FanOut



Push Model Using ActiveMQ for Scalability FanOut Configuration

ActiveMQ producer connection URI - will connect to all brokers:

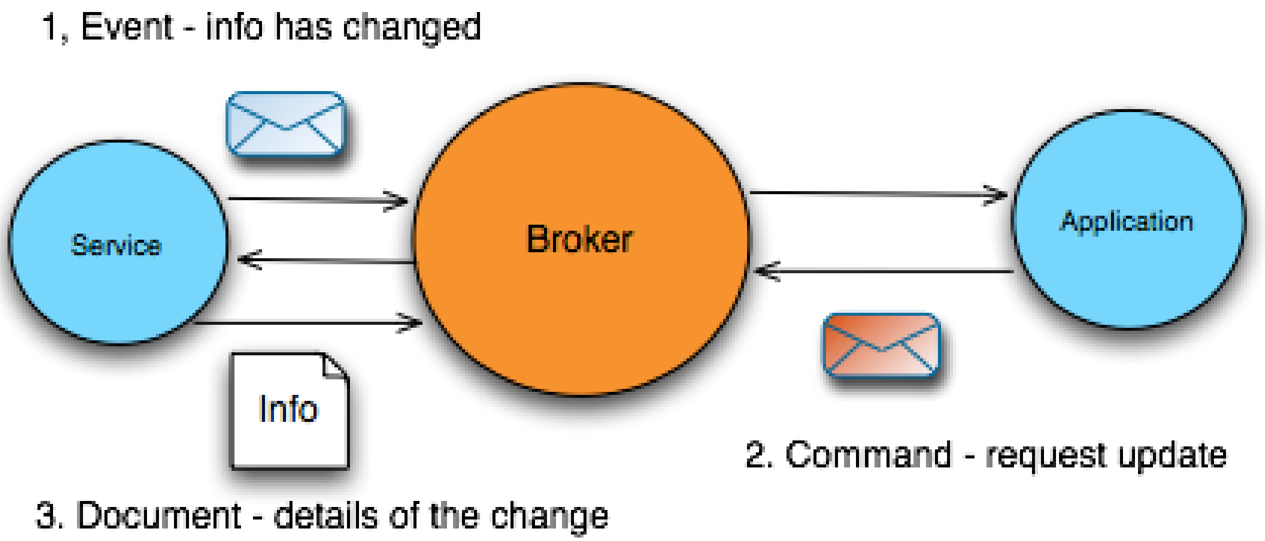
fanout:(static:(tcp://broker1:61616,tcp://broker2:61616))

ActiveMQ Consumers connection URI - will connect to only one broker

failover:(tcp://broker1:61616,tcp://broker2:61616)

Pull Model for Integration

- Three message types used
  - Event message - to notify observers of changes
  - Command message: - to request updated information
  - Document message: - details of the change

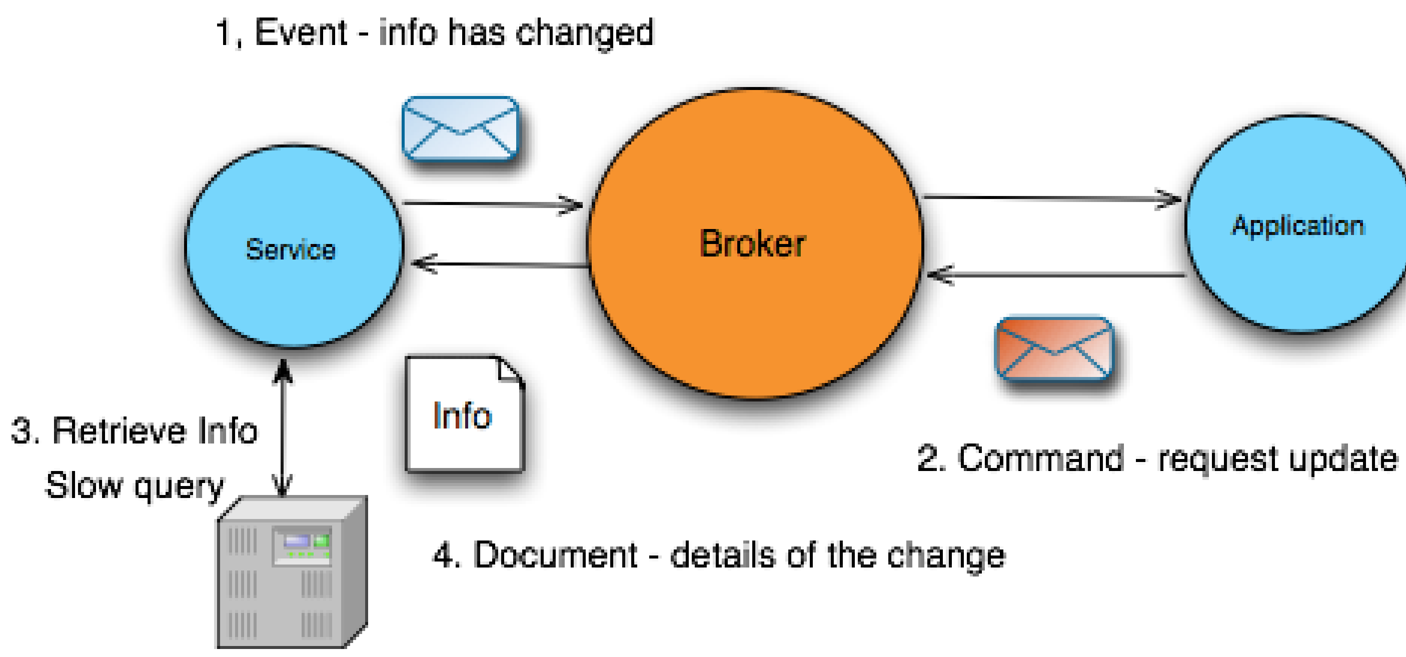

1, Event - info has changed
Service
Broker
Application
Info
2. Command - request update
3. Document - details of the change

Which Model to Use - Push or Pull? It Depends :)

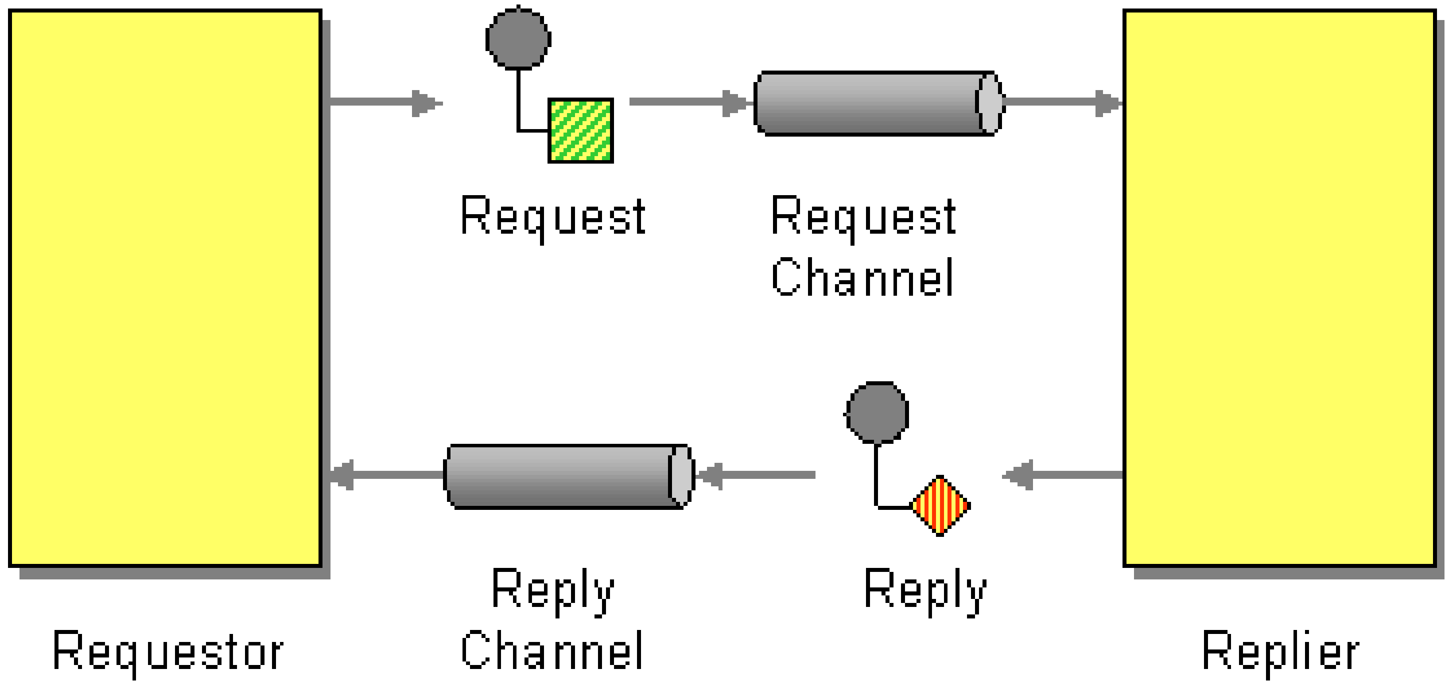
- Push model is good when:
  - When all consumers want details of change
  - Information (Document part) isn't too large
- Push model is bad when:
  - Lots of consumers - but only a few want updated require updated information

- Pull model is good when:
  - Lots of consumers, only a few will be interested in the change
  - Flexibility in the implementation
- Pull model is bad when:
  - Need to reduce traffic - 3 messages versus 1 for push
  - 2 Destinations versus 1 for push

A Bad Use of Pull


1, Event - info has changed
Service
Broker
Application
Info
3. Retrieve Info
Slow query
2. Command - request update
4. Document - details of the change

Two Way Conversation: Request/Reply

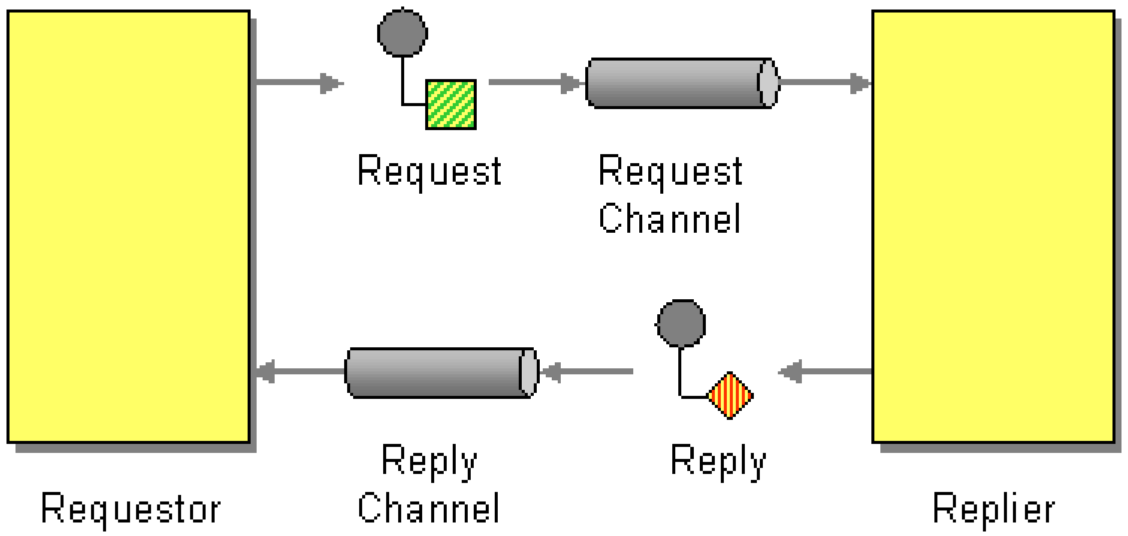


Two Way Conversation: Request/Reply with JMS

- javax.jms has helper classes for Request/Reply pattern
  - QueueRequestor
  - TopicRequestor
- Limitations
  - Requests have to be persistent
  - Request can’t be transacted
  - Requestor is synchronous
  - Uses a temporary destination for response:
    - There maybe a network outage - loose response
    - You may want to load balance responses - so need a Queue

Request/Reply: Camel to the Rescue!

- Camel supports Request/Reply - use an In/Out Exchange pattern
- ActiveMQ can help - destinations can be optionally garbage collected



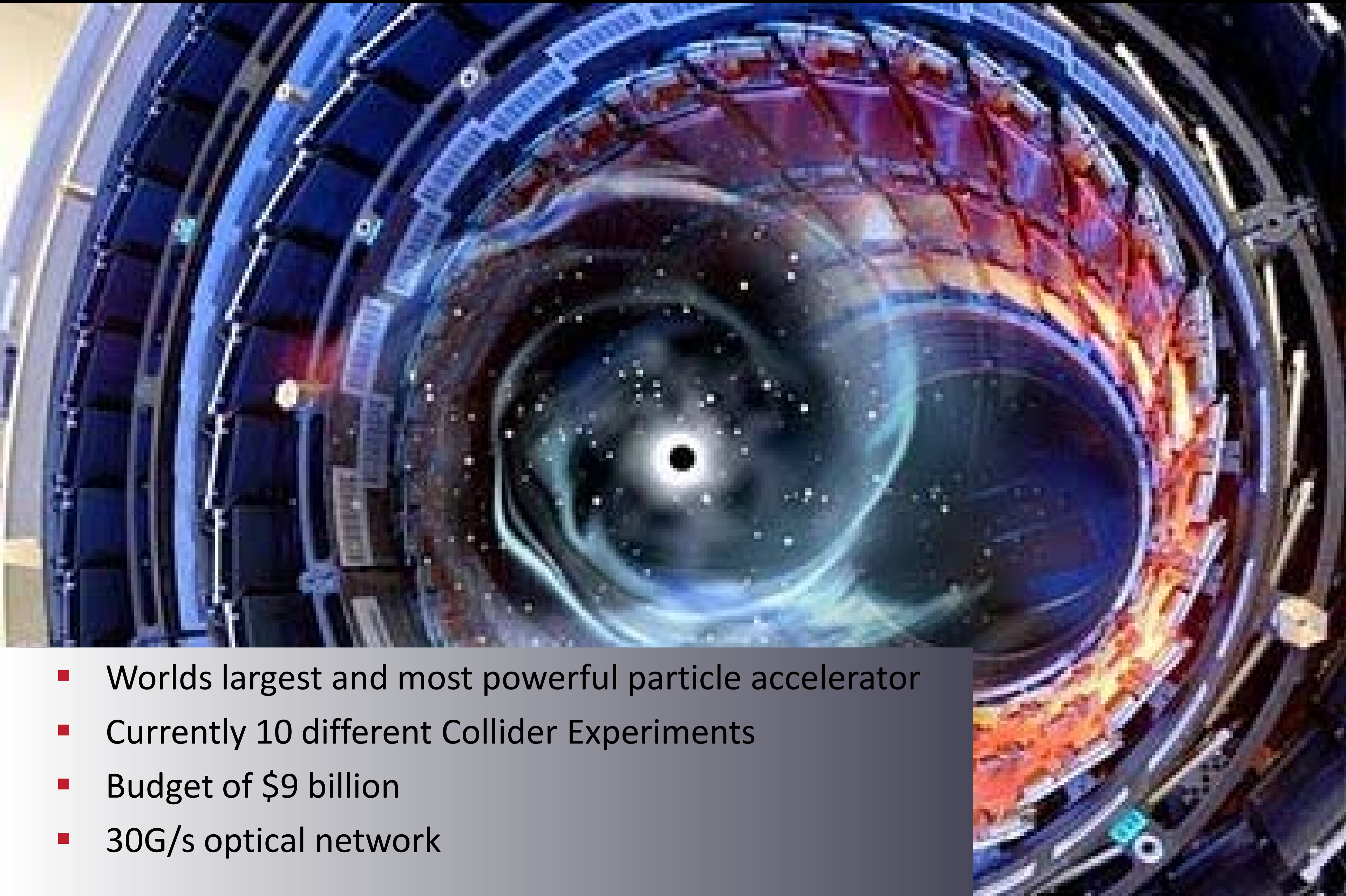
Enterprise Integration

Lets look at some challenges for integration - first ActiveMQ

Messaging Challenges

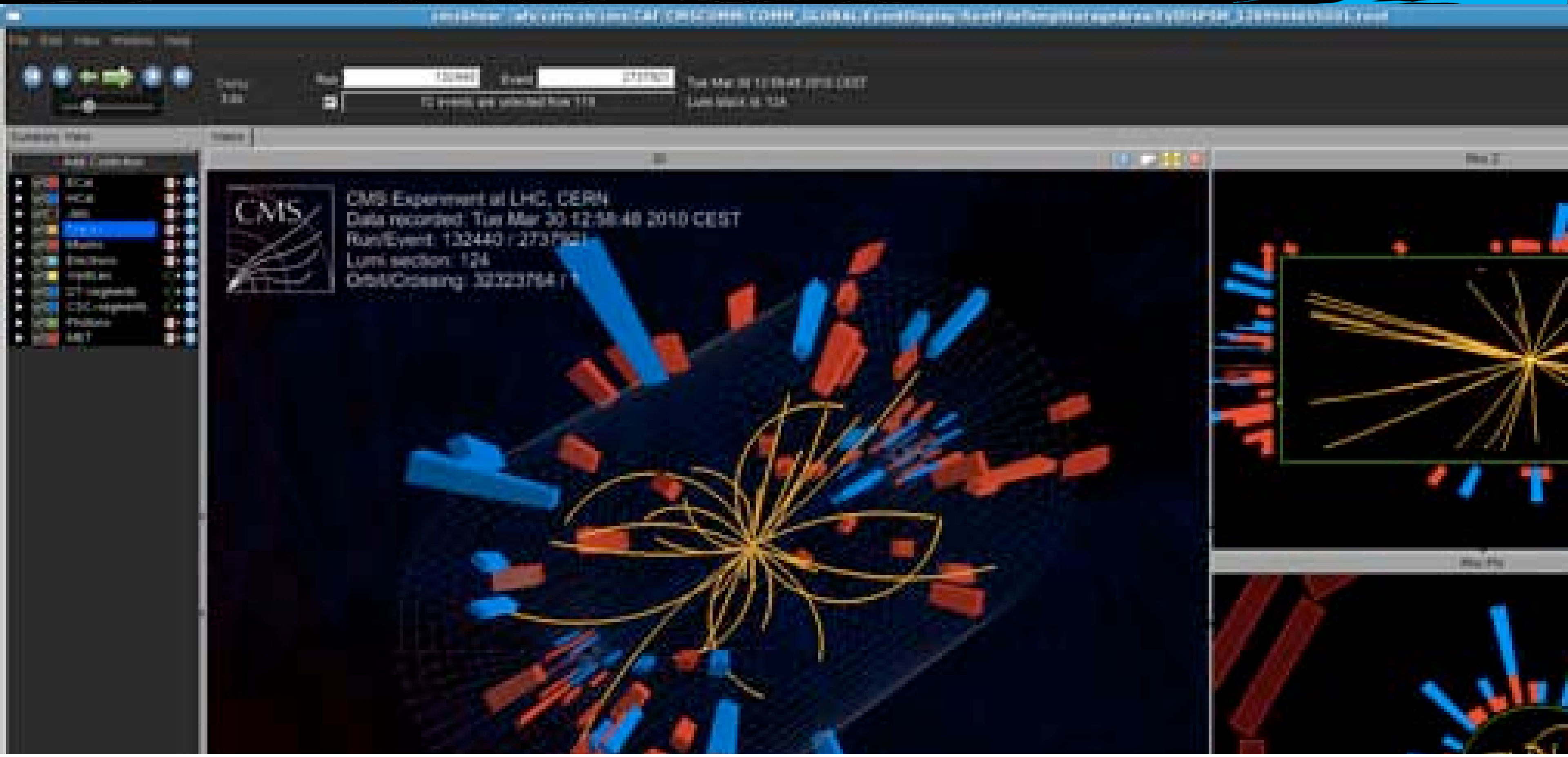
- Scalability
  - Vertical scaling - how do we support a 100,000 destinations?
  - Horizontal Scaling - how can we linear scale greater than 100k destinations ?
- Performance - everything needs to be faster - ActiveMQ - should be the fastest open source messaging
- Continuous availability (active active clustering)
- Protocol support – there's a range of choices - ActiveMQ should support them

CERN Large Hadron Collider



- Worlds largest and most powerful particle accelerator
- Currently 10 different Collider Experiments
- Budget of $9 billion
- 30G/s optical network

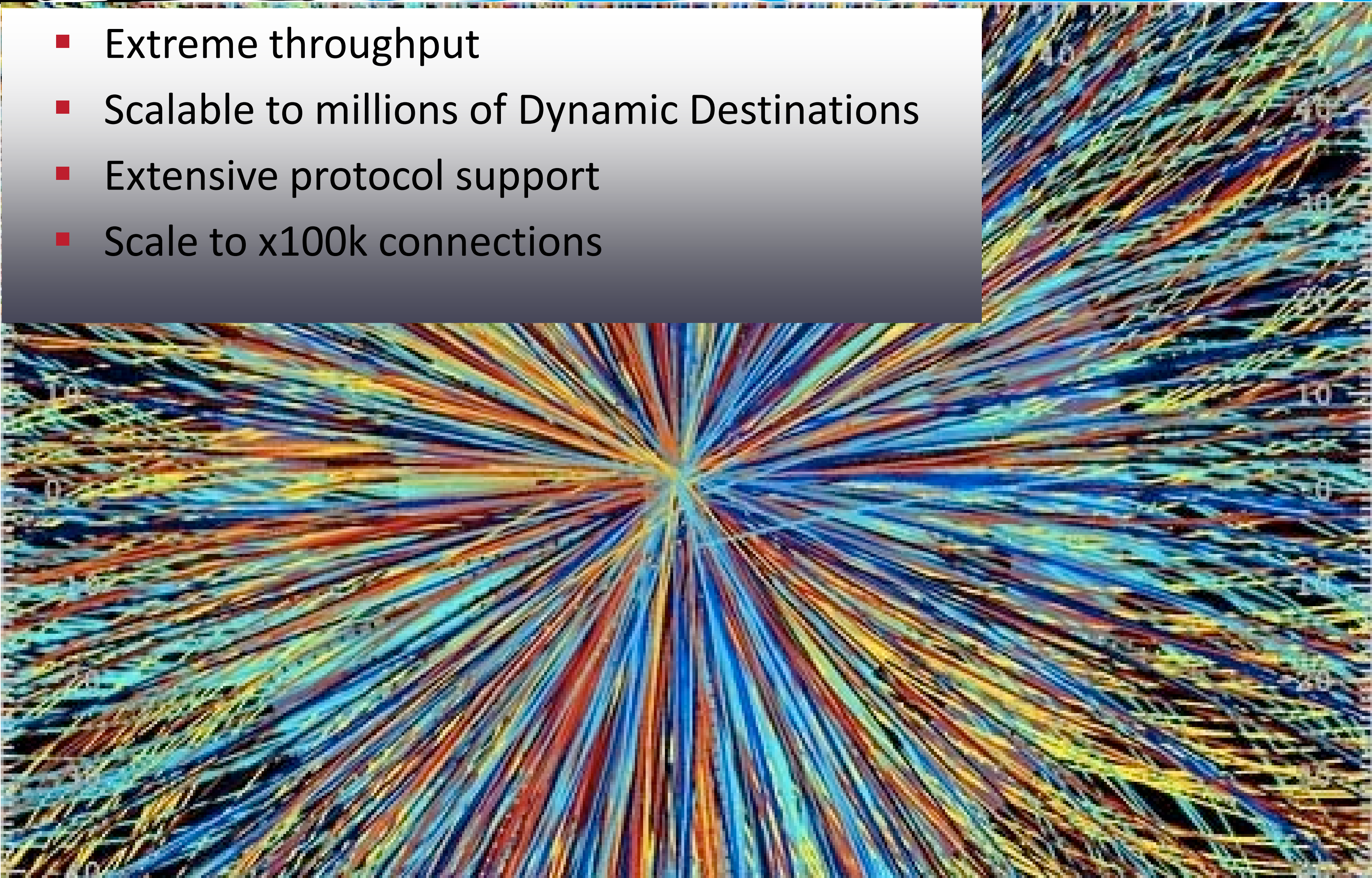
CERN Large Hadron Collider


CMS Experiment at LHC, CERN
Data recorded: Tue Mar 30 12:58:48 2010 CEST
Run/Event: 132440 / 2737921
Lumi section: 124
Orbit/Crossing: 32323764

- Produces 15 petabytes of data annually
- WLCG – 34 different countries
- Lots of Data – lots of destinations
- Requires next generation messaging to cope with information demand

Need New Messaging Architecture

- Extreme throughput
- Scalable to millions of Dynamic Destinations
- Extensive protocol support
- Scale to x100k connections



Introducing ActiveMQ Apollo

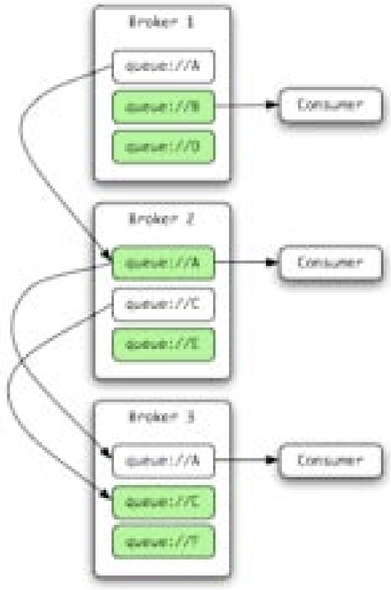
- Scala based core for very fast, scalable dispatching
- Modular design – independent lifecycle support
- Apache Karaf core for standardization and flexibility
- Enhanced Queues
- More Protocols
- Richer REST based Management
- Intelligent Clustering

More Protocols than OpenWire and STOMP

- MQTT
  - IBM developed open protocol
    - Supported by Websphere MQ, Mosquitto, and now Apollo
  - Publish/Subscribe and Queues (version 5 spec)
  - Designed to be used from embedded devices all the way up to applications
- Beanstalk
  - Short lived tasks
  - Sender needs to take action if a job is not consumed
  - Sender needs to know job is on the Queue
  - Very scalable
- AMQP 1.0
  - First Enterprise Version
  - Supports distributed transactions
  - Supports reliable messaging
  - Flow Control should now work

Automatic Destination Partitioning (Clustering)

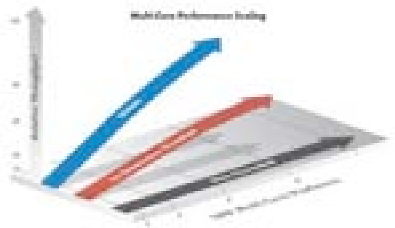
- Uniform load across multiple brokers
- Clients automatically connect to the correct broker(s)
- Massive scalability
- Reduce network hops



Apollo Implemented: HawtDispatch

HawtDispatch

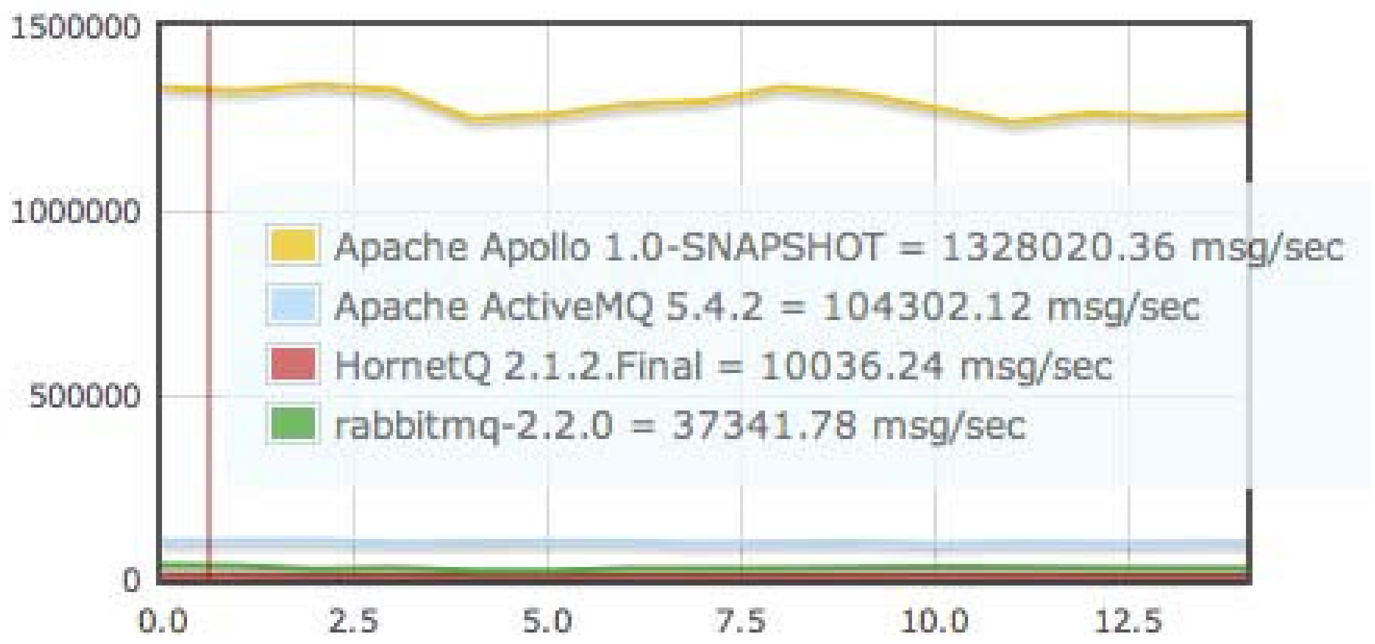
- Based on Grand Central Dispatch (from OS X)
- Fixed size thread pool = processing cores
- Eliminates the need for Java Synchronization blocks
- Actor style thread architecture
  - == easier to maintain code
- Requires all code to be non-blocking
  - You end up methods using callbacks or a continuation passing style



FuseSource
A Progress Software Company

The Proof is in the Benchmarks

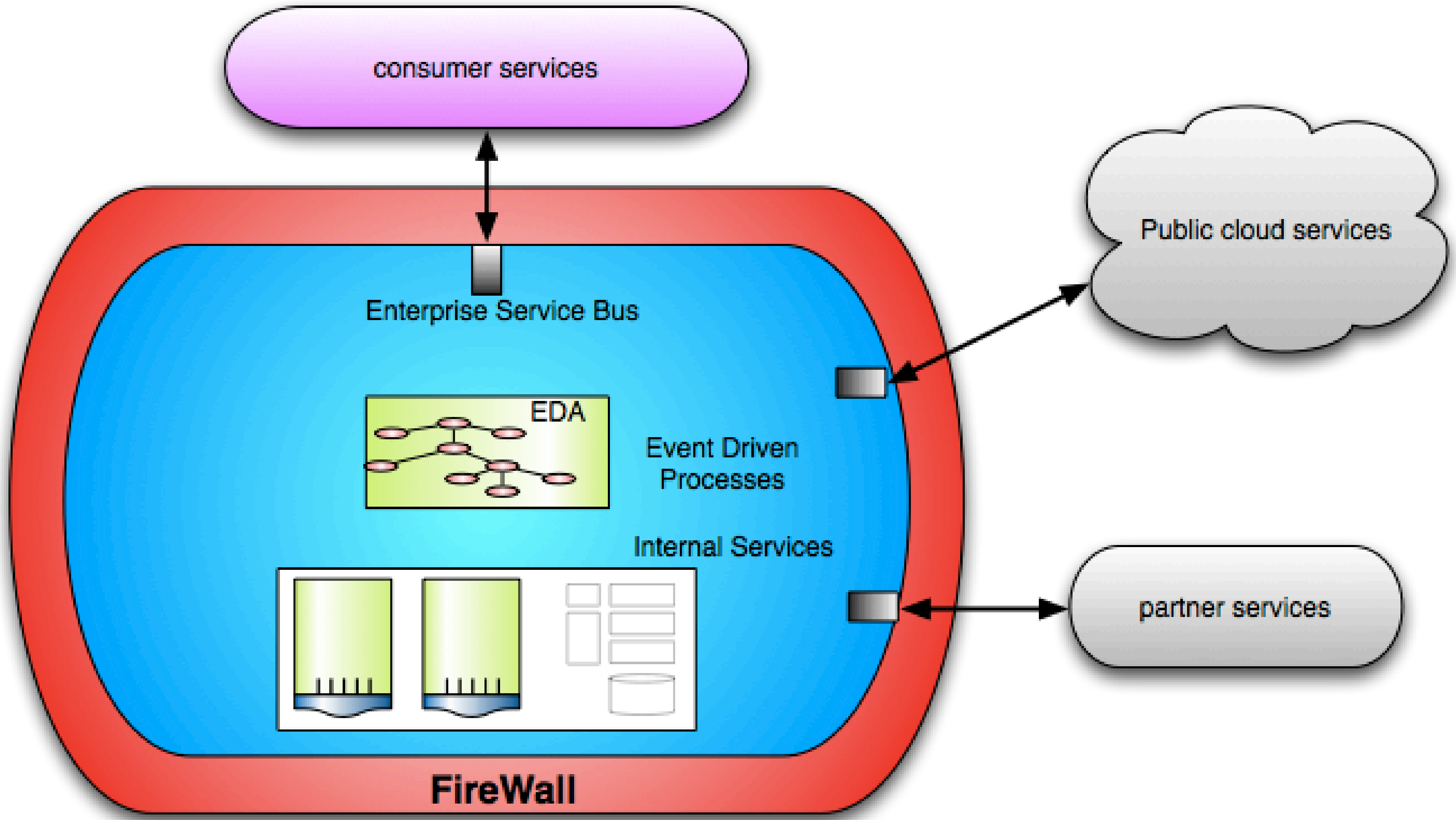
- https://github.com/chirino/stomp-benchmark
- Total Topic Consumer Rate for:
  - Non Persistent & 20 byte message content
  - 10 Producers
  - 10 Consume



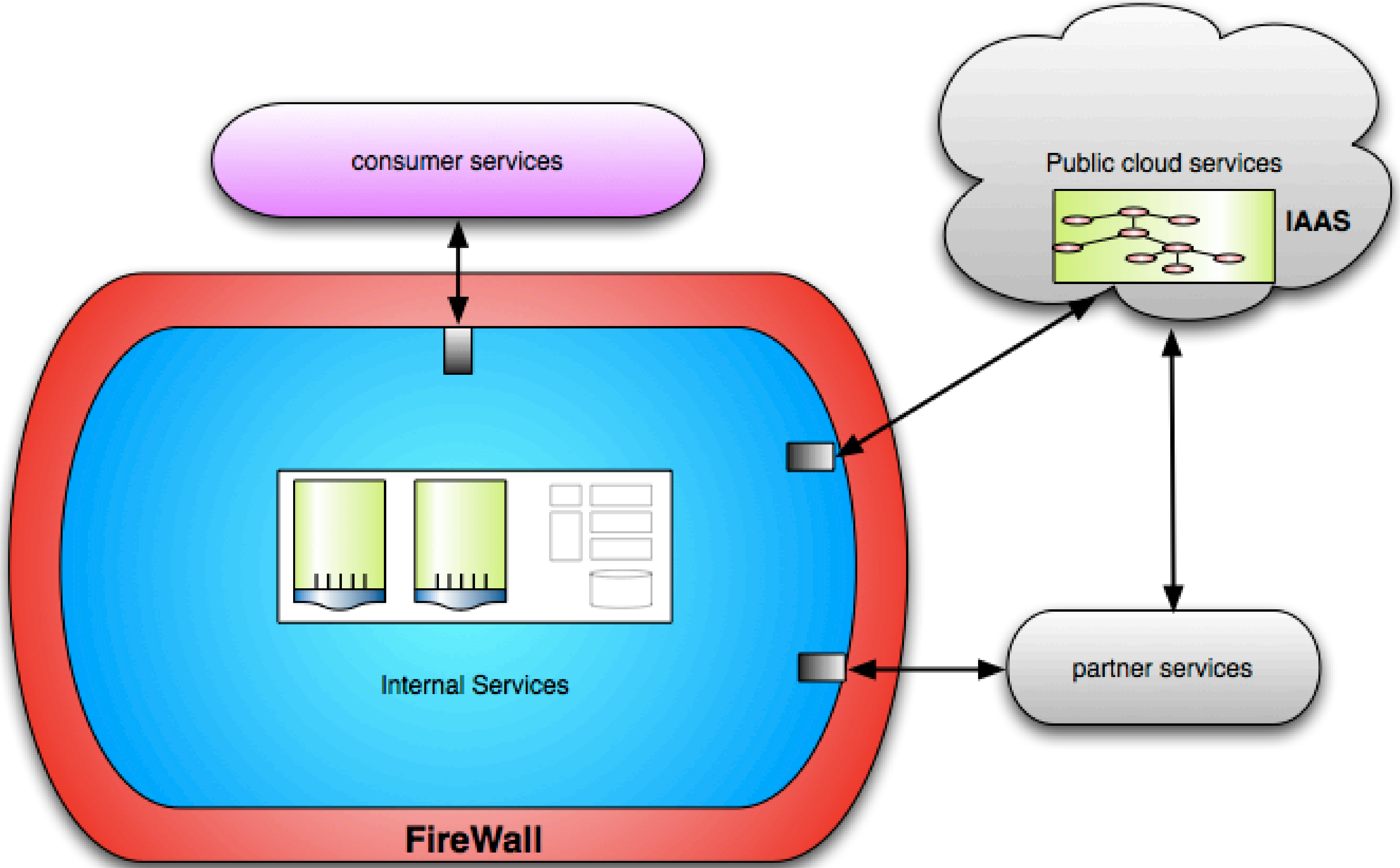
Enterprise Integration

What about deployment?

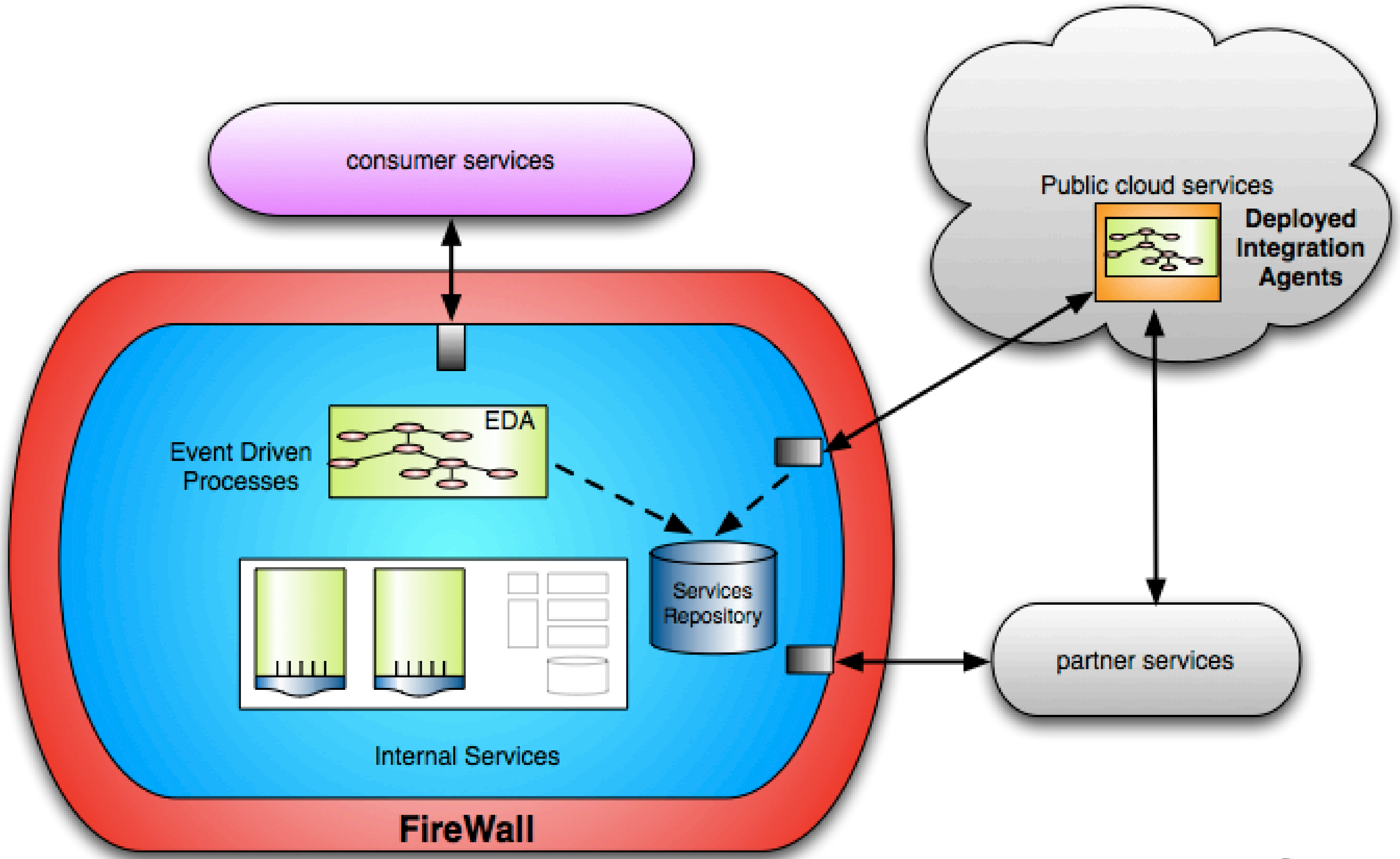
Enterprise Integration - On Premise to Cloud



Enterprise Integration - Integration As a Service



Enterprise Integration - Hybrid Approach



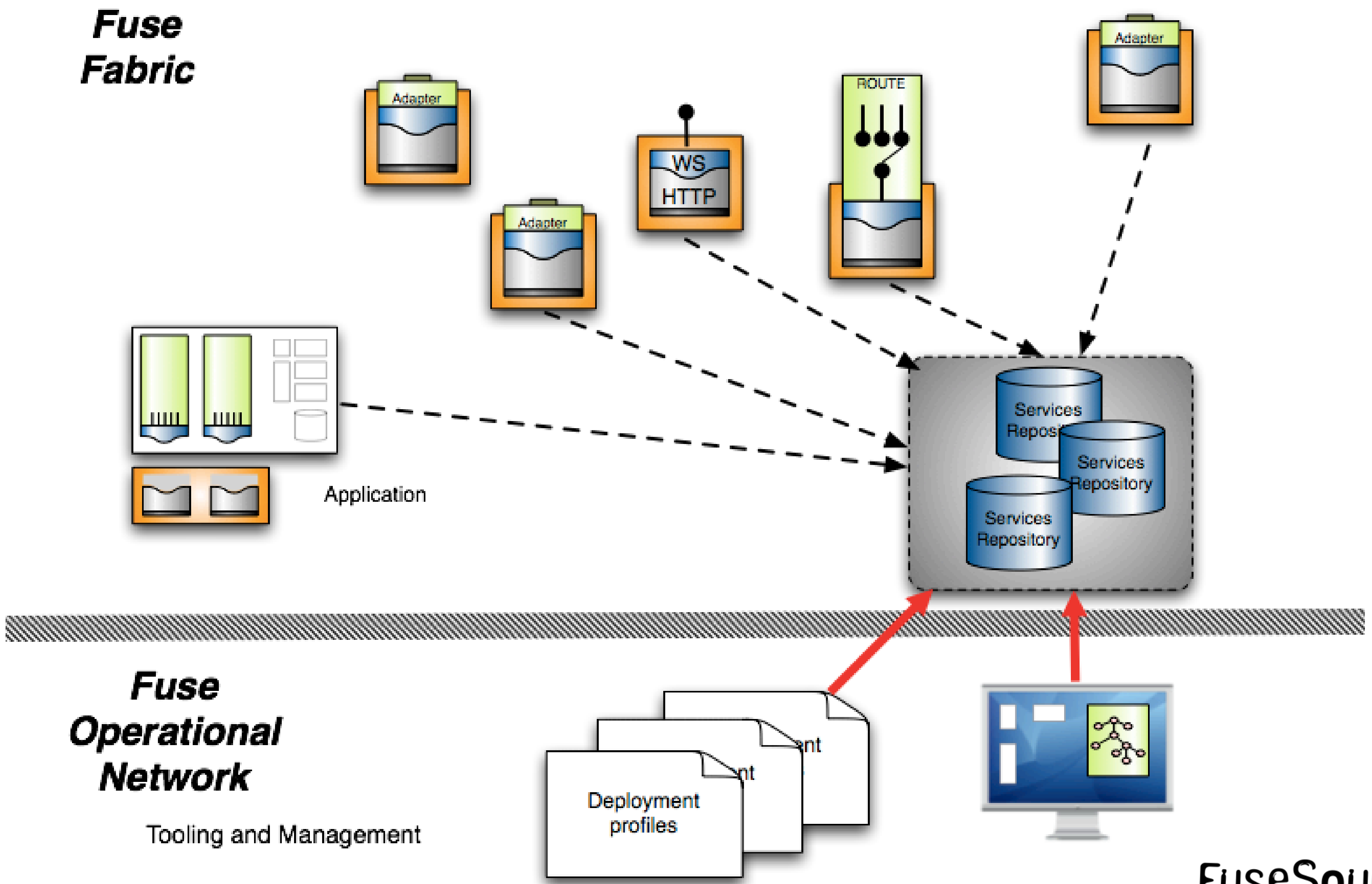
How to Support Hybrid Deployments?

- Location transparency in Endpoints
  - endpoints can be relocated
  - endpoints can be load balanced
  - endpoints can be elastic
  - endpoints can be highly available
- Distributed Configuration
  - Configuration has to be accessed across multiple domains
  - Configuration has the highly available

How to Support Hybrid Deployments (continued)?

- Discovery and Inventory - automatically discover deployed resources (runtime registry)
- Intelligent clustering
  - Hot standby / load balancing
  - Singletons
  - Non-linear elastic deployments
- Distributed Management
  - Distributed control
  - Distributed monitoring

Introducing Fuse Fabric and Fuse Operational Network (FON)



Enterprise Integration

How FuseSource can help …

FuseSource Corporation

The Leaders in Open Source Integration and Messaging

Team behind the projects

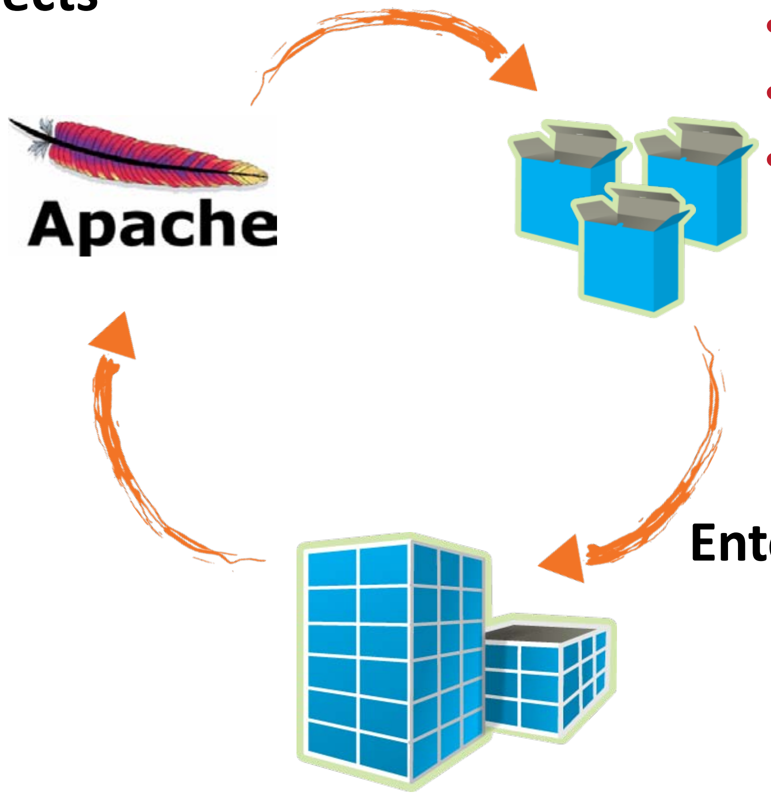
- Leaders at Apache
- Product roadmaps
- Code contributions

Productized distributions

- Integrated
- Tested
- Tooling

Enterprise support

- Subscriptions
- Training
- Consulting



FuseSource Subscription = Long Term Success

Support

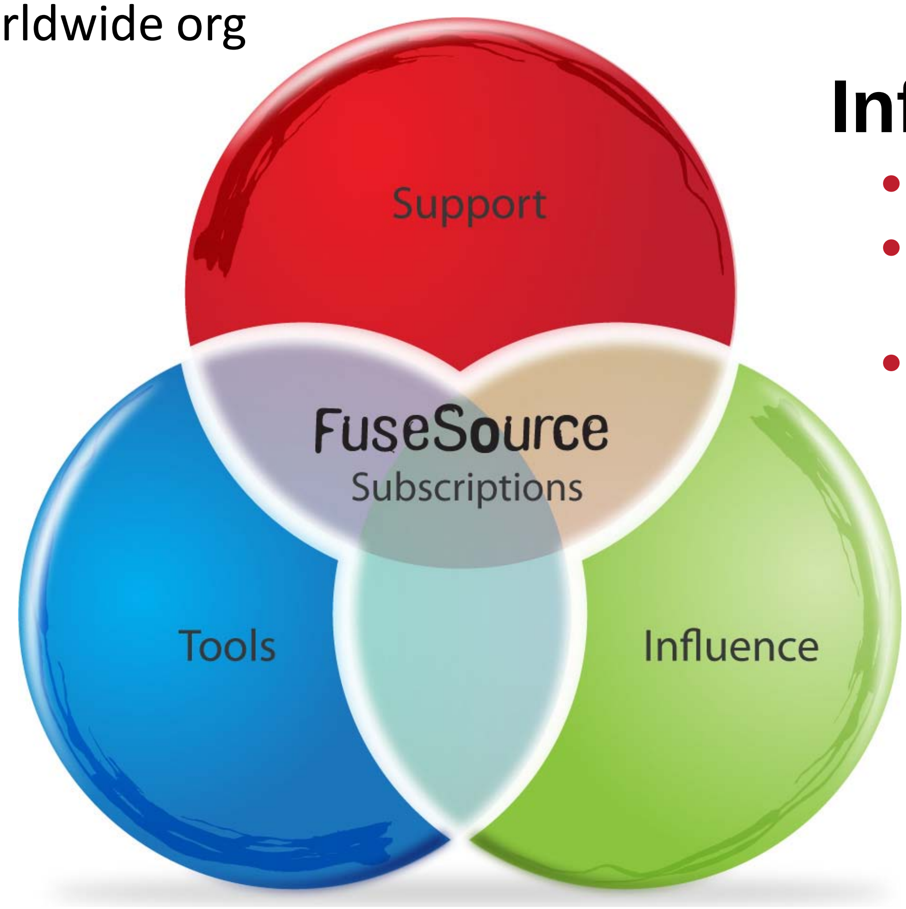
- From the project leaders
- Enterprise-class
- Worldwide org

Influence

- Product knowledge
- Effect product direction
- Partner with the developers

Tools

- Pilot projects
- Development
- Deployment



Any Questions?

Innovation for Integration

Free to redistribute

Enterprise class……

FuseSource

A Progress Software Company

Enterprise ActiveMQ - More Information:

- http://fusesource.com/
- http://activemq.apache.org/
- http://camel.apache.org/
- http://hawtdispatch.fusesource.org/
- http://fabric.fusesource.org/