

Qualities of Service

Oxford University
Software Engineering Programme
Dec 2014



© Paul Fremantle 2012. Portions © Jeremy Gibbons 2010, © WSO2 2005-2012 used with permission of the author(s).
Licensed under the Creative Commons 3.0 BY-SA (Attribution-Sharealike) license.
See <http://creativecommons.org/licenses/by-sa/3.0/>

Qualities of Service

- Transactions
- Reliable Messaging
- Security



Transactions

- Idealized *indivisible activities* techniques for maintaining the illusion in the face of complexity,
- Ideas arose from distributed databases
- Concurrency, failures
- Underlying finance, logistics, manufacturing. . .
- *Transaction Processing: Concepts and Techniques*, Gray and Reuter, 1993
- (http://books.google.co.uk/books?id=S_yHERPRZScC)



© Paul Fremantle 2012. Portions © Jeremy Gibbons 2010, © WSO2 2005-2012 used with permission of the author(s).

Licensed under the Creative Commons 3.0 BY-SA (Attribution-Sharealike) license.

See <http://creativecommons.org/licenses/by-sa/3.0/>

ACID

- *atomicity*
 - all-or-nothing
- *consistency*
 - integrity-preserving: invariants satisfied
- *isolation*
 - hidden intermediate results: multi-user behaviour consistent with single-user mode
- *durability*
 - permanent committed results



Problems Avoided with ACID

- *lost update*
 - write committed and acknowledged but then discarded
- *inconsistent retrieval*
 - reads of multiple fields at different times
- *non-serializability*
 - loss of single-user abstraction
- *conflict*
 - e.g. simultaneous bookings of the same room

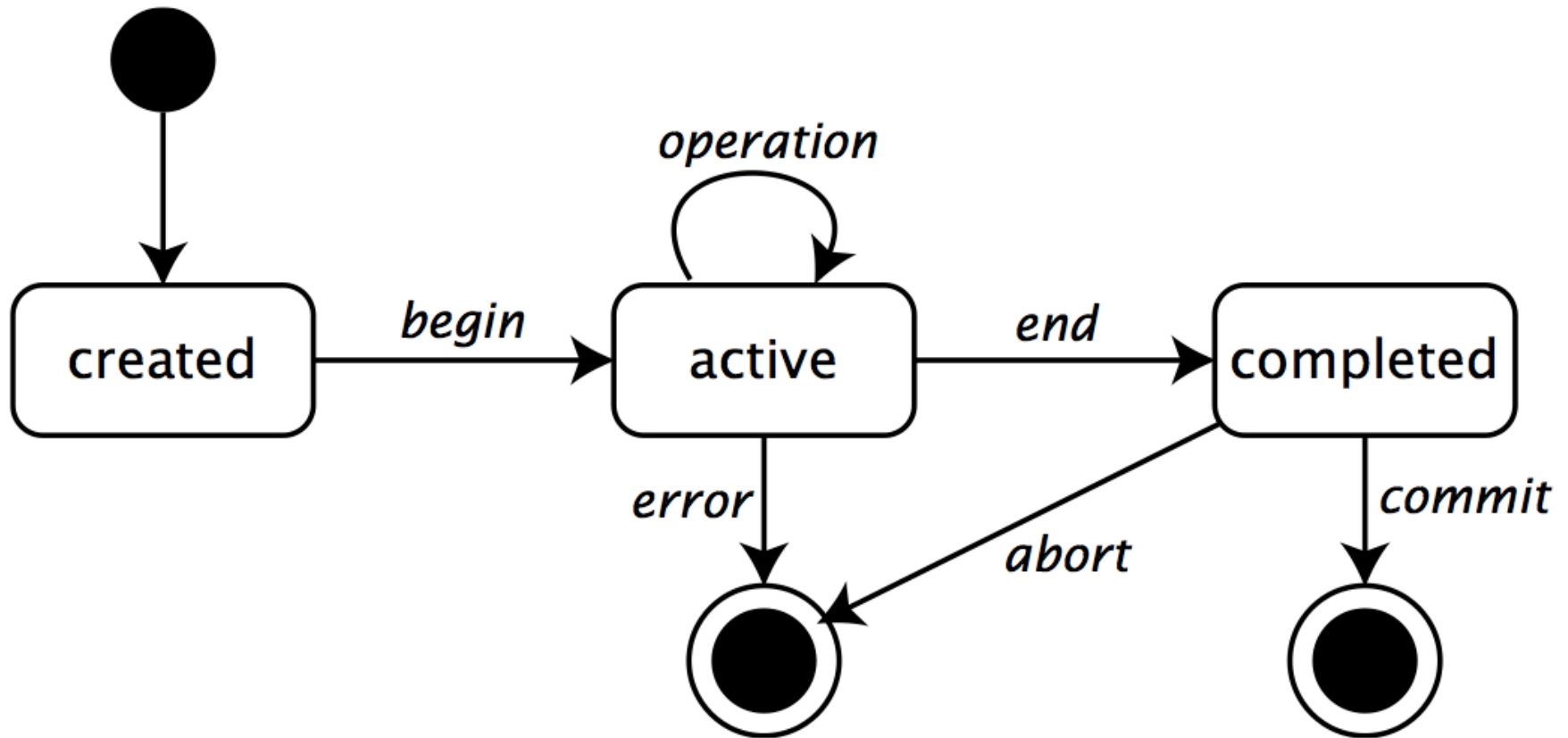


© Paul Fremantle 2012. Portions © Jeremy Gibbons 2010, © WSO2 2005-2012 used with permission of the author(s).

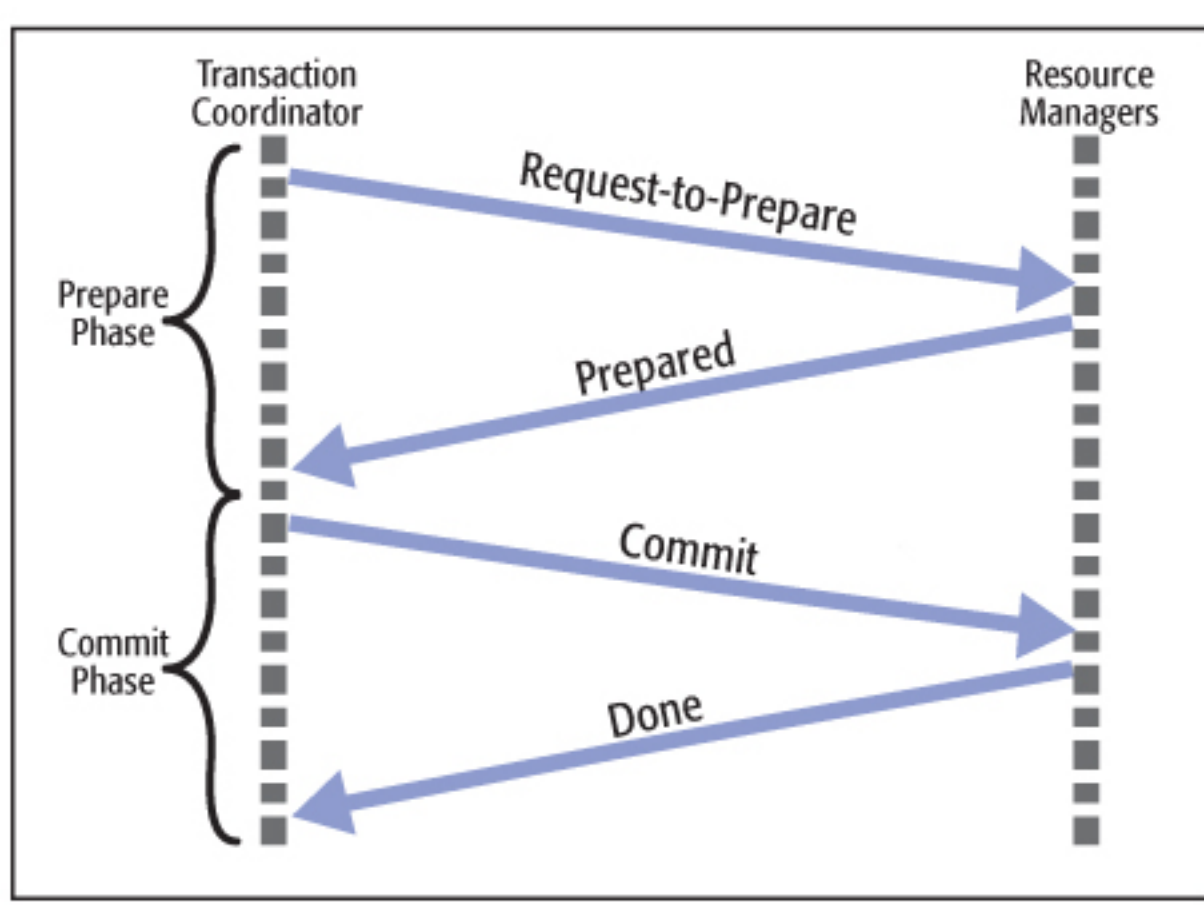
Licensed under the Creative Commons 3.0 BY-SA (Attribution-Sharealike) license.

See <http://creativecommons.org/licenses/by-sa/3.0/>

Transaction Lifecycle



Two Phase Commit (2PC)



Locking

- *serialization mechanisms* for resources, enforcing unique access
- *read locks* (shareable) and *write locks* (exclusive)
- may need to wait for locked resource to be released
- may result in *deadlock*: two parties, each waiting for the other
- lock resources in canonical order (requires foresight), or abort one party (requires rollback)

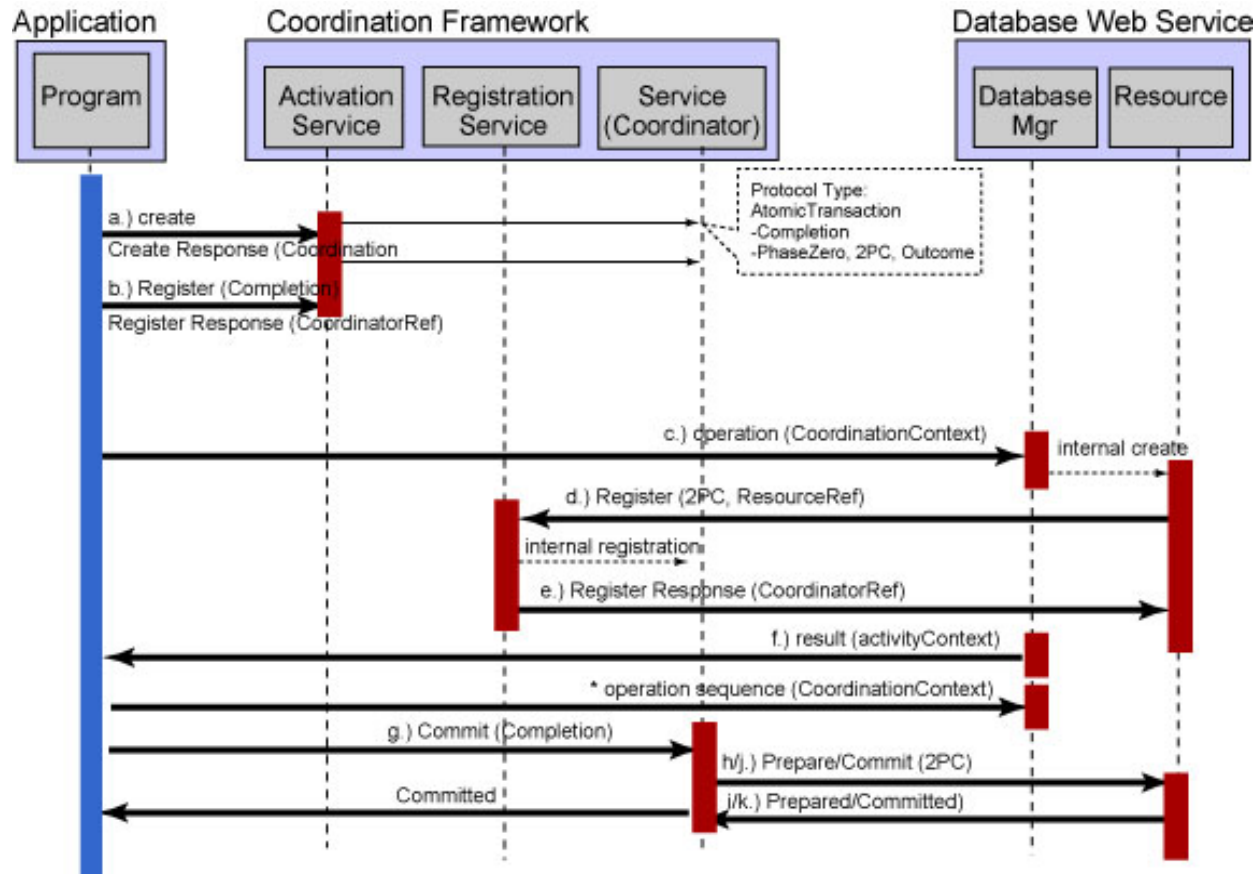


© Paul Fremantle 2012. Portions © Jeremy Gibbons 2010, © WSO2 2005-2012 used with permission of the author(s).

Licensed under the Creative Commons 3.0 BY-SA (Attribution-Sharealike) license.

See <http://creativecommons.org/licenses/by-sa/3.0/>

WS-Atomic Transactions



WS-AT

- Relies on WS-Addressing and WS-Coordination
 - WS-Coordination is a generalized coordination protocol for multiple parties
 - Also used by WS-BusinessActivity
- How does WS-AT fit with?
 - Loose Coupling
 - Service Oriented Architecture
 - Asynchronous calling



CAP Theorem

Brewer's Theorem

- Consistency
 - ACID
- Availability
 - High Available
- Partition Tolerance
 - Can work at Internet scale across Datacenters

You can only guarantee two of these three
While this is true, its also misleading.

Recommend you read

<http://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed> by Eric Brewer

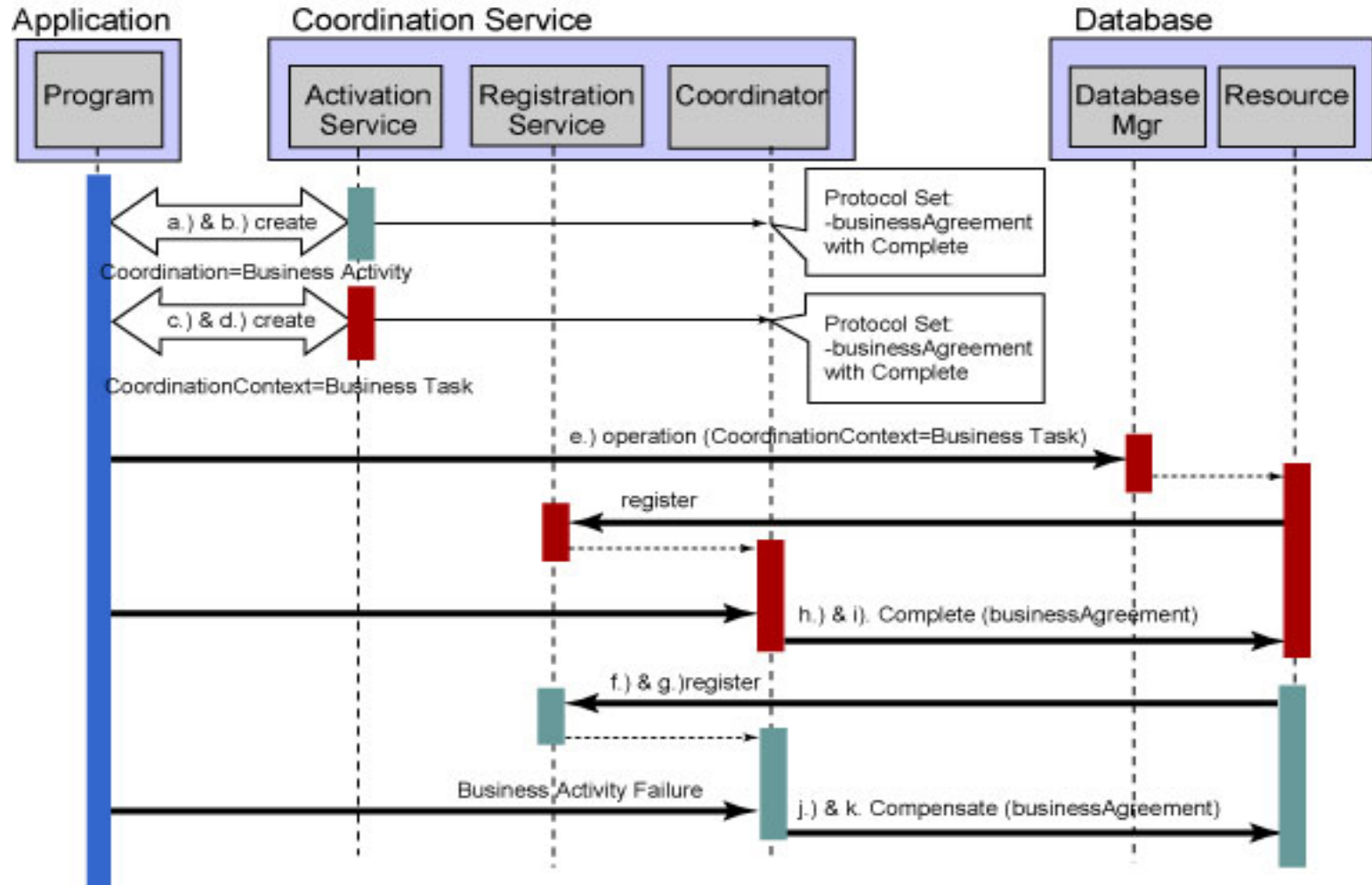


Eventually Consistent

- Despite the arguments, etc about CAP
- Eventually Consistent / NoSQL databases are gaining huge mindshare (and a lot of marketshare of new greenfield apps too)
 - e.g. Cassandra vs MySQL w/50Gb
 - MySQL r/w 300ms/350ms
 - Cassandra r/w 15ms/0.15ms
- Also scalability, elasticity, no master



WS-BusinessActivity



WS-BusinessActivity

- Suitable for long running loosely coupled scenarios
- Can use WS-AT to handle system exceptions – WS-BA handles business exceptions
- Business logic must support the cancel/compensate logic
- Not ACID
 - For example, no clear isolation as locks are not held
- Can take place even if not all participants are always available

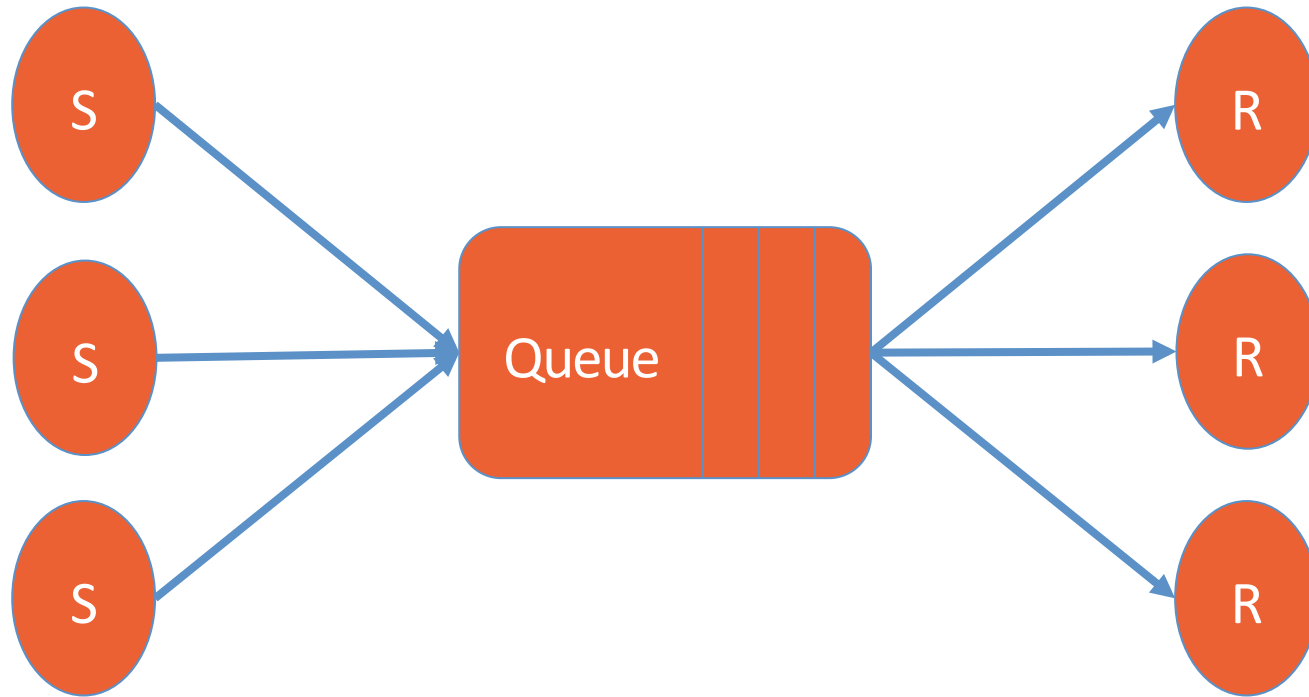


REST Transactions

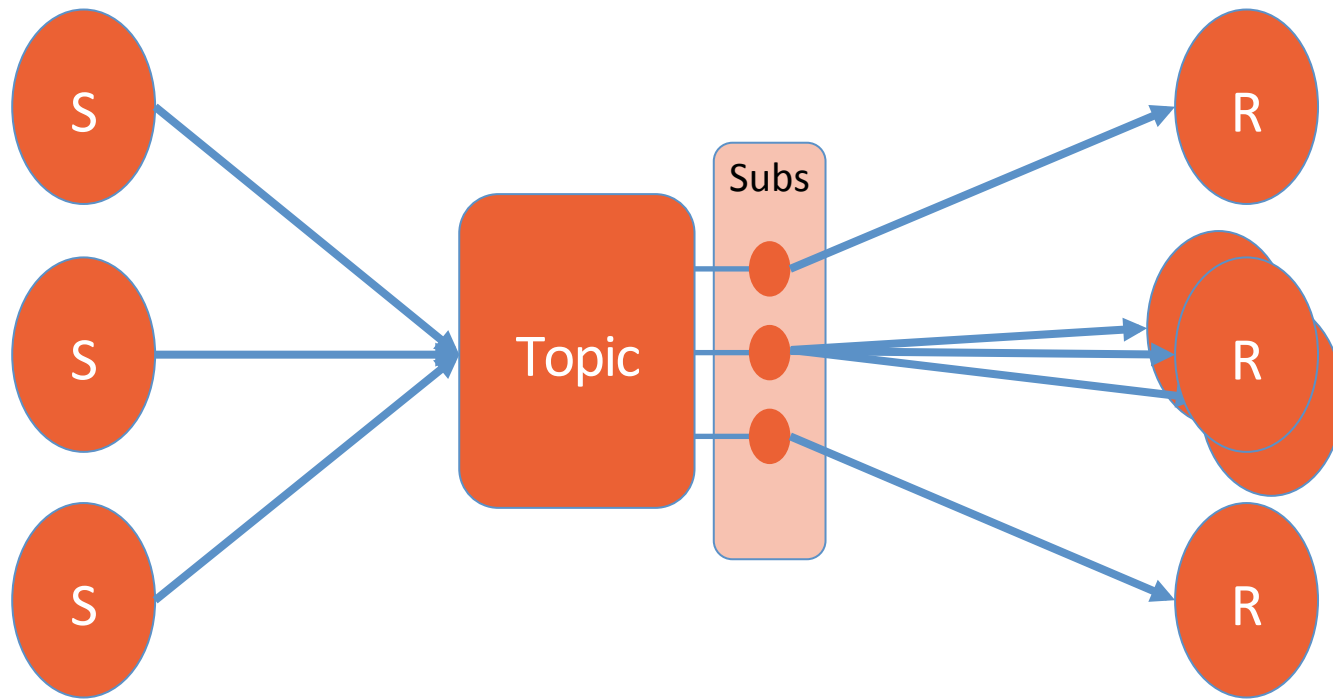
- Transactions
 - Various ideas: RETRO
 - <http://freo.me/TLnyWp>
- Patterns
 - Towards Distributed Atomic Transactions over RESTful Services, Pardon and Pautasso, REST From Research to Practice, ISBN: 978-1-4419-8302-2 (Print) 978-1-4419-8303-9 (Online)



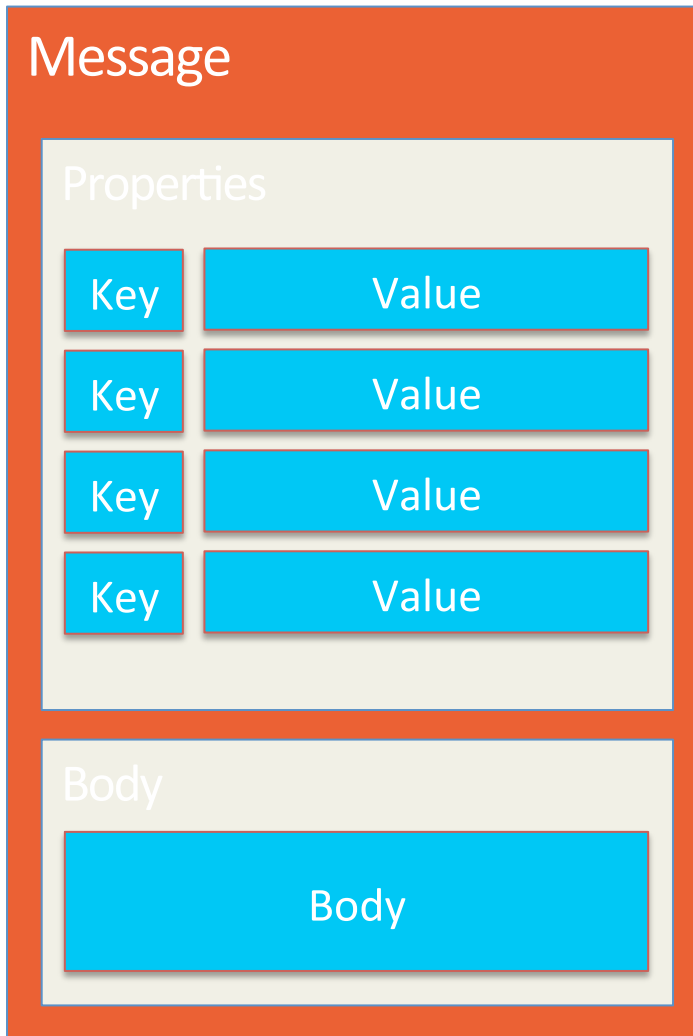
Messaging concepts: queuing



Messaging concepts: pub/sub



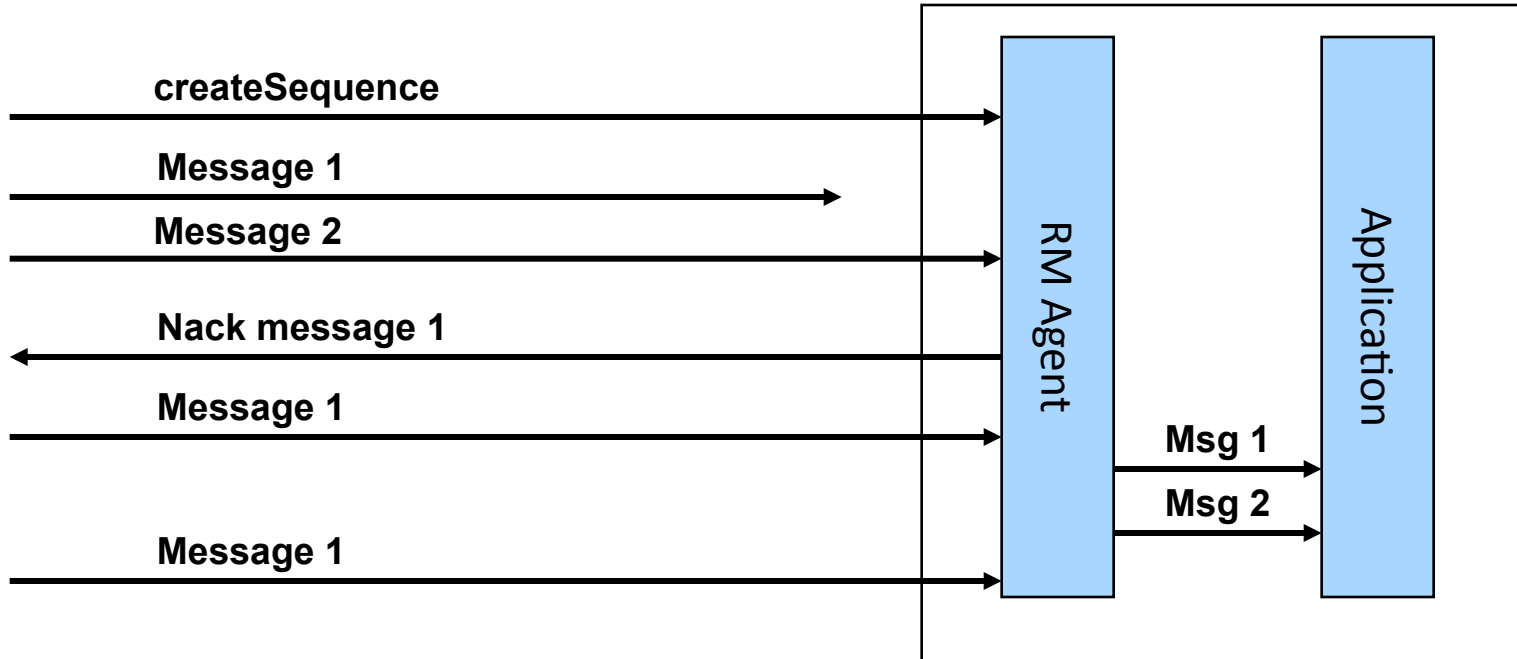
Messaging concepts: message



- **Properties**
 - Key/value pairs exposed to the broker
 - Subscription rules can filter based on properties
- **Body**
 - Opaque payload not exposed to the broker
 - Can be used for encrypted data

WS-ReliableMessaging

- Provides an MQ-like model for Web Services
 - Redelivers missing messages
 - Numbers messages so they can be delivered InOrder and ExactlyOnce



Other approaches to reliability

- ebMS Reliable Messaging
- AMQP 1.0
- MQTT
- STOMP
- REST patterns
 - Some believe you don't need RM
 - <http://www.infoq.com/articles/no-reliable-messaging>



Security

- Dealt with at length in a separate lecture!



Composability



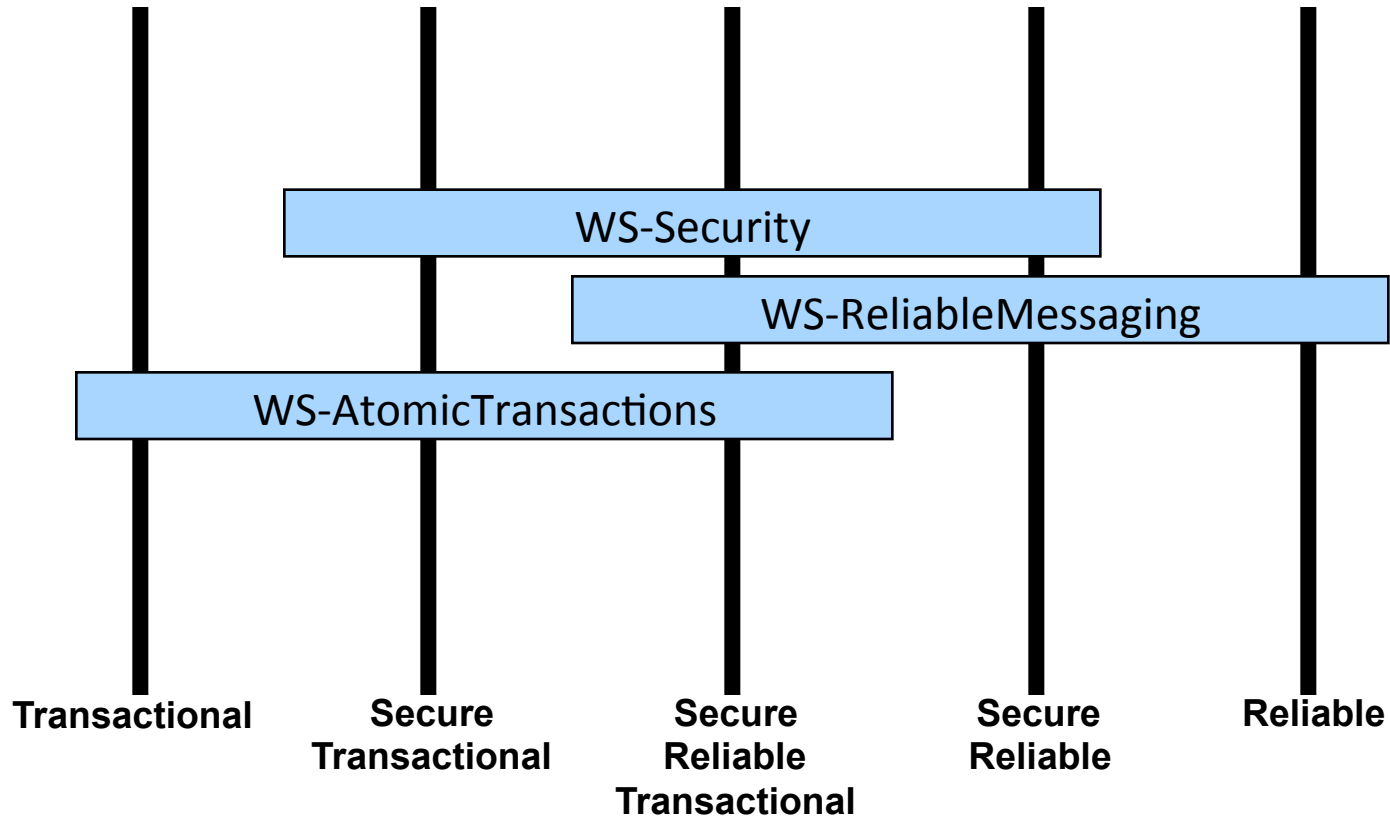
© Paul Fremantle 2012. Portions © Jeremy Gibbons 2010, © WSO2 2005-2012 used with permission of the author(s).
Licensed under the Creative Commons 3.0 BY-SA (Attribution-Sharealike) license.
See <http://creativecommons.org/licenses/by-sa/3.0/>

Composability

- The whole architecture of WS-* has been designed around this model
 - Use what you need
 - And no more
 - Don't replicate technology in different standards
 - All the standards should work together



Basic Composability

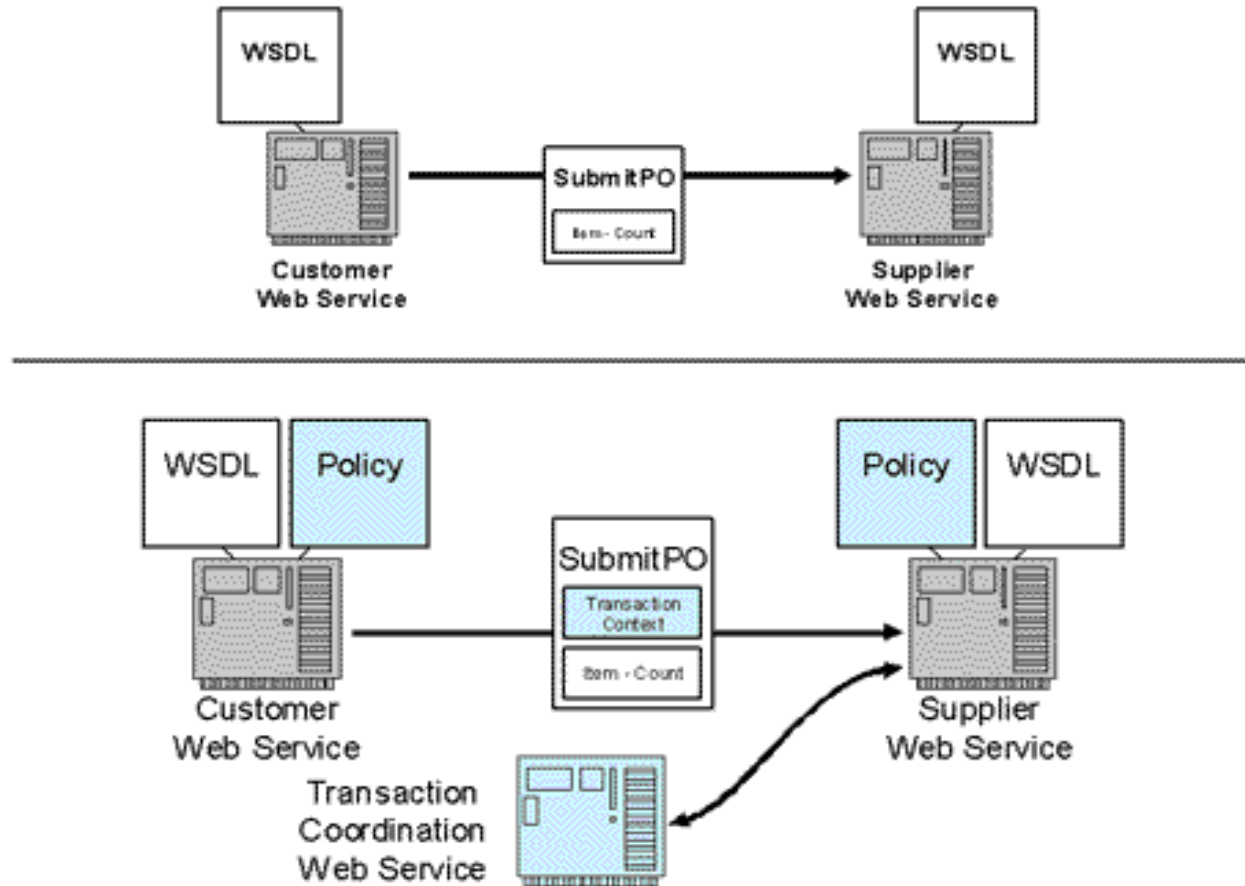


Composability

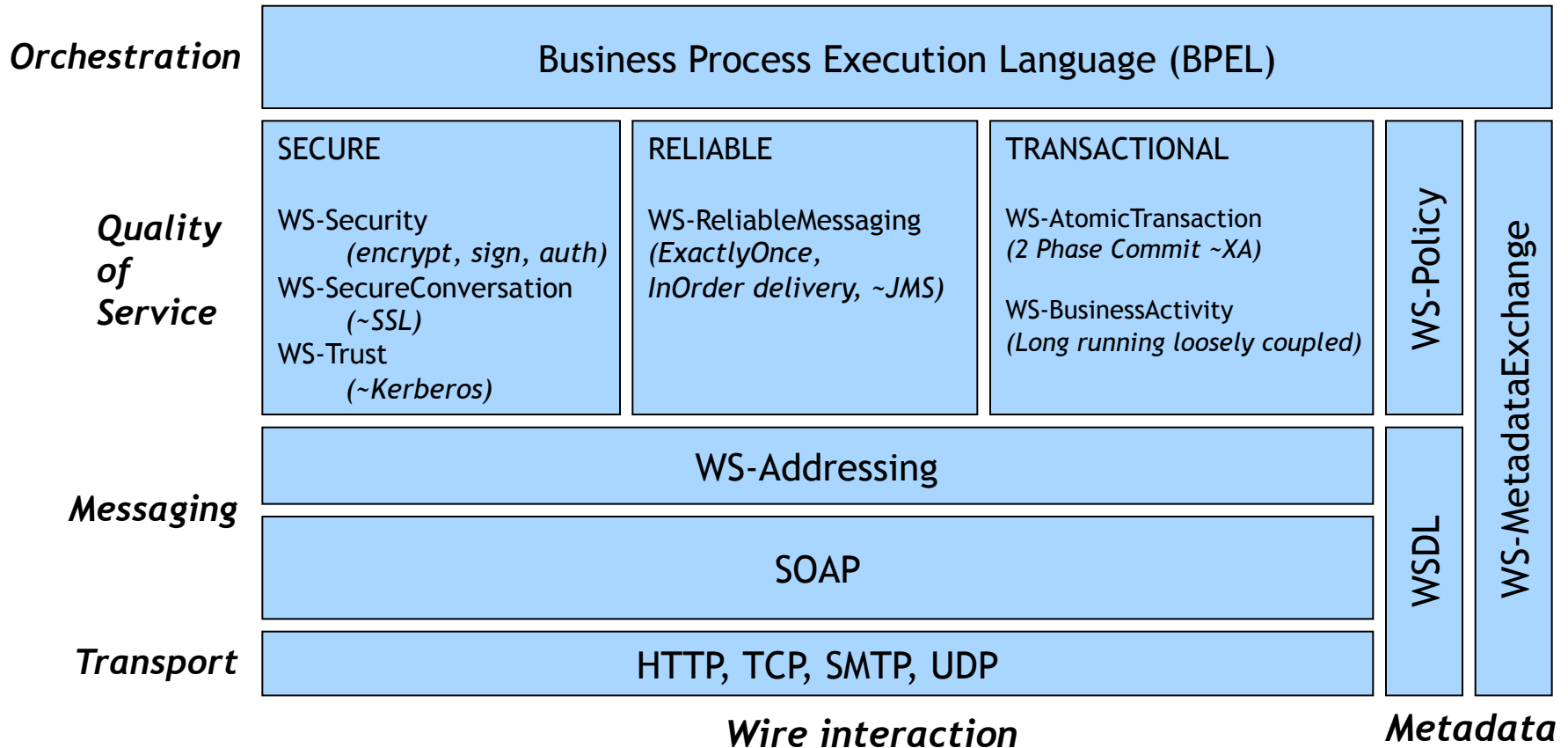
- SOAP Headers that work together
- Common usage of base components
 - SOAP
 - WS-Addressing
- The right building blocks, e.g.
 - Reliable pub/sub
 - WSRM + WS-Eventing
 - Secure Single Sign-on
 - WS-Trust + WS-Security



Composability example



Key Web Services Standards



The Web services platform forms a complete framework for open standards enterprise middleware



WS-Policy Framework

- A way of publishing requirements and capabilities for service endpoints
 - A simple language to specify combinations
 - Plus a way of attaching to WSDL, etc
 - Plus a set of Policy Assertion Languages
- For example:
 - WS-SecurityPolicy
 - WS-ReliableMessagingPolicy
 - etc
- More in later chapter



WS-MetadataExchange

- How to query a service at runtime for its policy, WSDL, schema, etc



WS-Eventing

- Guess what! There are two specs:
 - WS-Notification (IBM, HP, Oracle, Tibco and Globus)
 - WS-BaseNotification
 - WS-BrokeredNotification
 - WS-Topics
 - WS-Eventing (Microsoft, BEA, IBM, Tibco)
- BaseNotification and Eventing are very similar

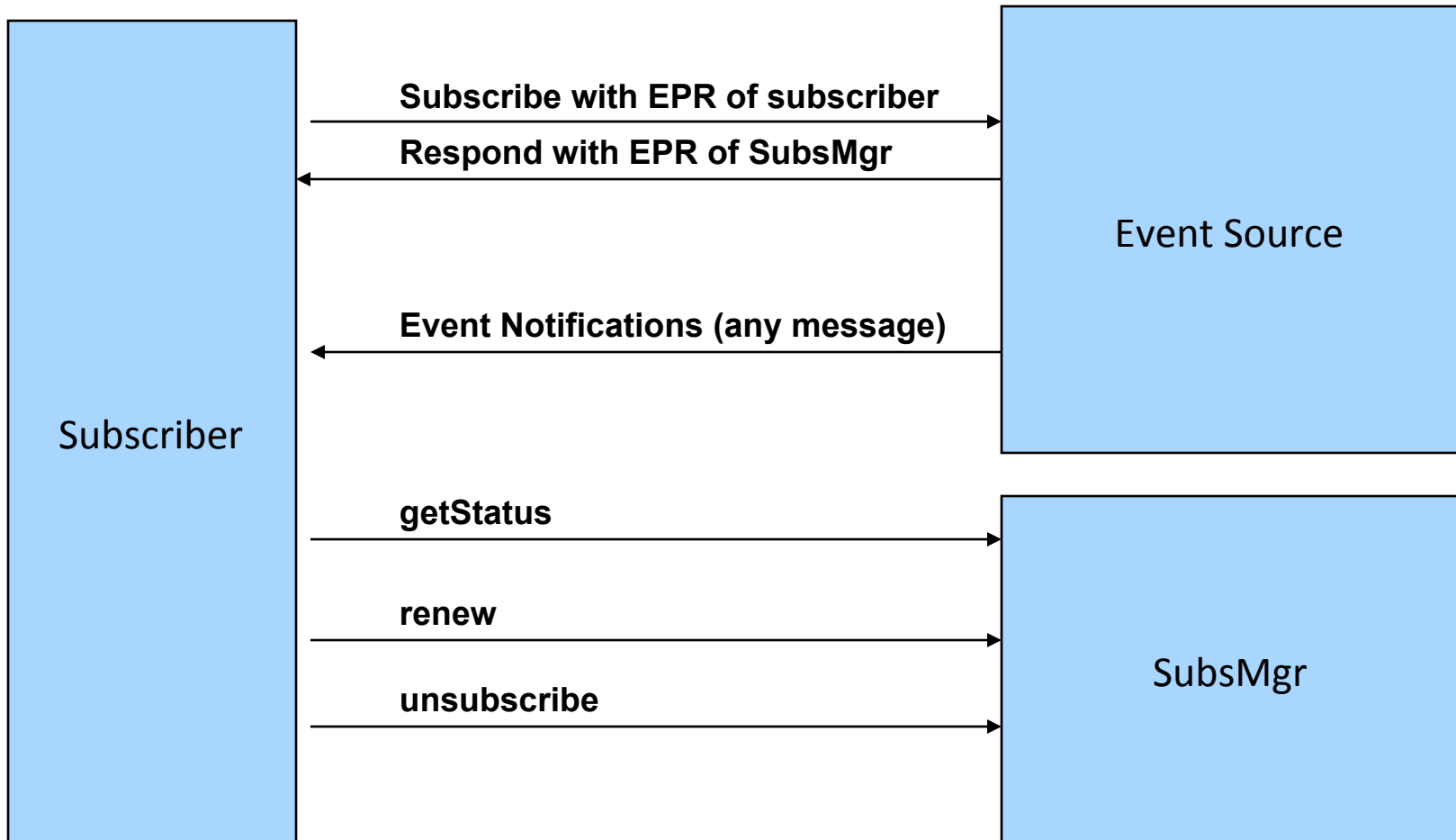


WS-Eventing

- Publish-Subscribe in Web services/XML
- Create, Delete, Renew and expire subscriptions
- Subscriptions may have XPath-based filters
- Specify how event messages should be delivered
- Compose with other standards for secure, reliable transacted delivery



WS-Eventing Example



Event Driven Architecture in REST

- Comet
 - Not considered good!
- WebSockets
 - Not particularly RESTful, but W3C approved
- EventSource
 - https://developer.mozilla.org/en-US/docs/Server-sent_events/EventSource
- WebHooks
 - Web Callbacks



Local / Remote Transparency

“

The hard problems in distributed computing concern dealing with partial failure and the lack of a central resource manager. . . differences in memory access paradigms between local and distributed entities.

Many aspects of robustness can be reflected only at the protocol/interface level. . . An interface design issue has put an upper bound on the performance. . . Part of the definition of a class of objects will be the specification of whether those objects are meant to be used locally or remotely.

”

Jim Waldo et al, ‘A Note on Distributed Computing’, 1994



© Paul Fremantle 2012. Portions © Jeremy Gibbons 2010, © WSO2 2005-2012 used with permission of the author(s).

Licensed under the Creative Commons 3.0 BY-SA (Attribution-Sharealike) license.

See <http://creativecommons.org/licenses/by-sa/3.0/>

One more thing to note

- Service Level Agreements and active management have become *much* more important than RM and Transactions
 - Widespread APIs over the web
 - DoS attack prevention
 - Guaranteed availability
- More when we look at API Management



Questions?



© Paul Fremantle 2012. Portions © Jeremy Gibbons 2010, © WSO2 2005-2012 used with permission of the author(s).
Licensed under the Creative Commons 3.0 BY-SA (Attribution-Sharealike) license.
See <http://creativecommons.org/licenses/by-sa/3.0/>