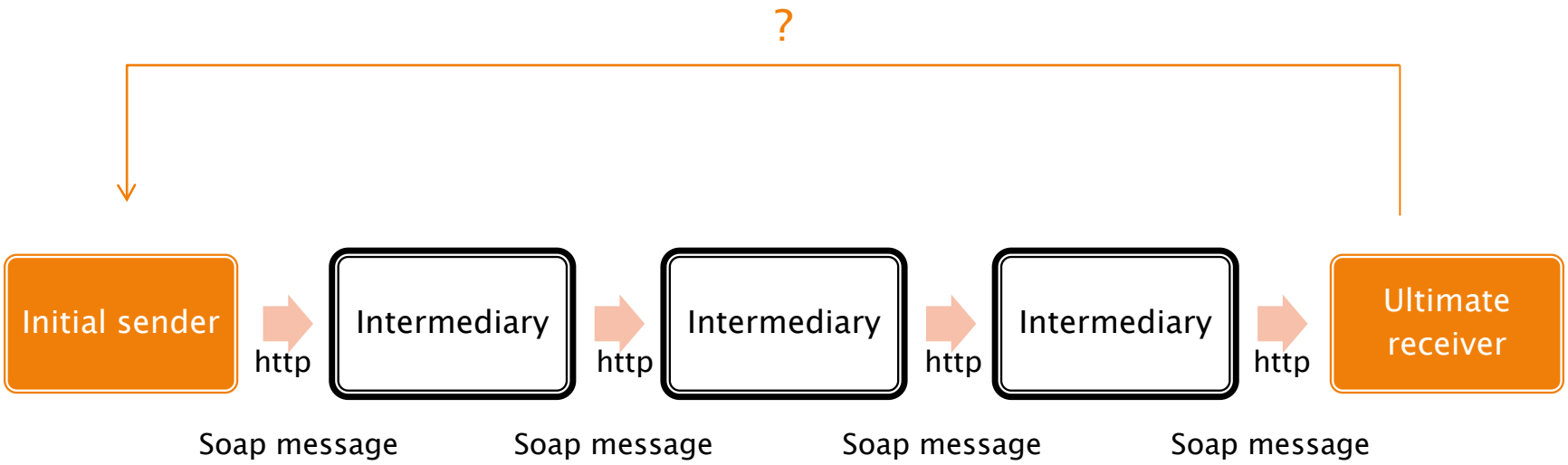
WS-ADDRESSING
WS-RESOURCES
WS-NOTIFICATION
WS-EVENTING

Paul Mestereaga
Mahmoud Abubakr
Kevin Van Gyseghem

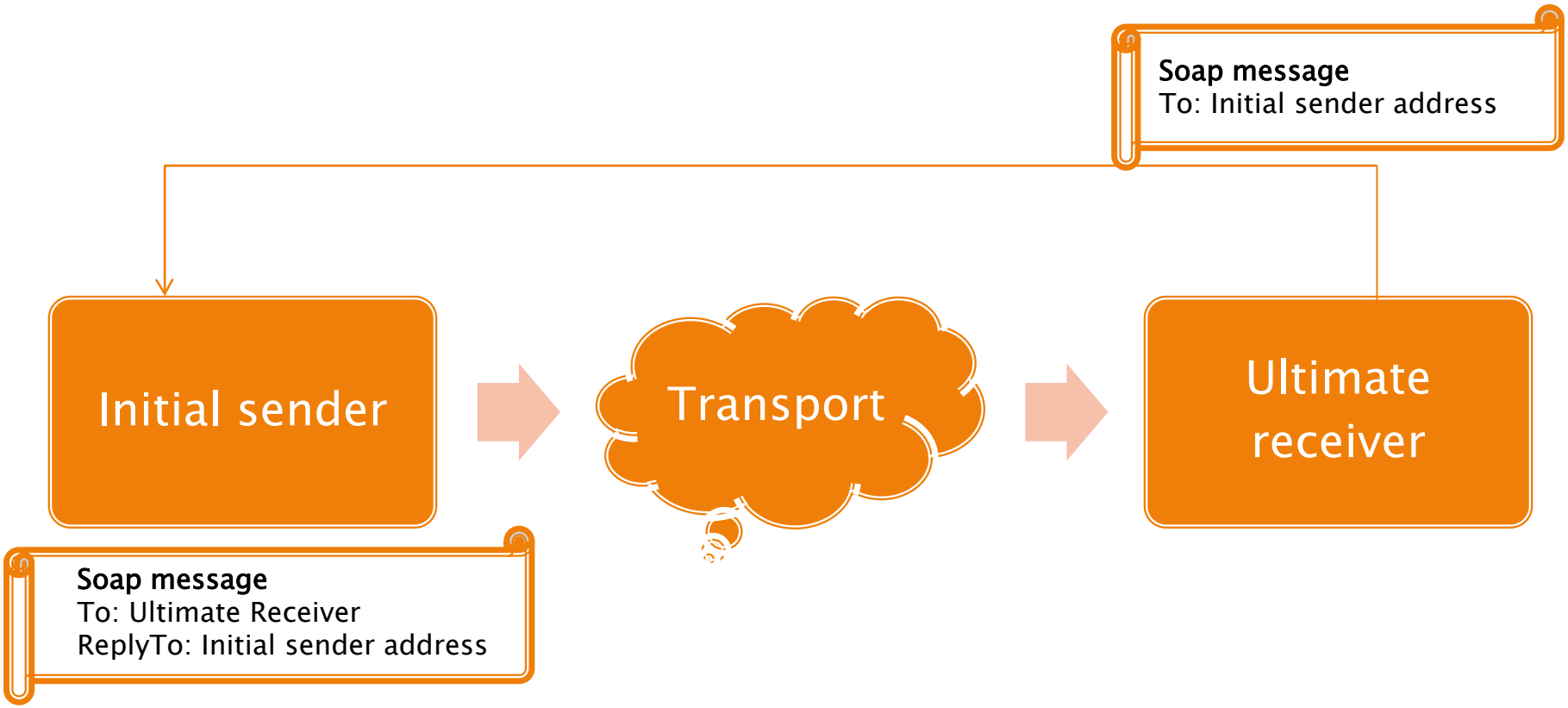
WS-Addressing: Example problem



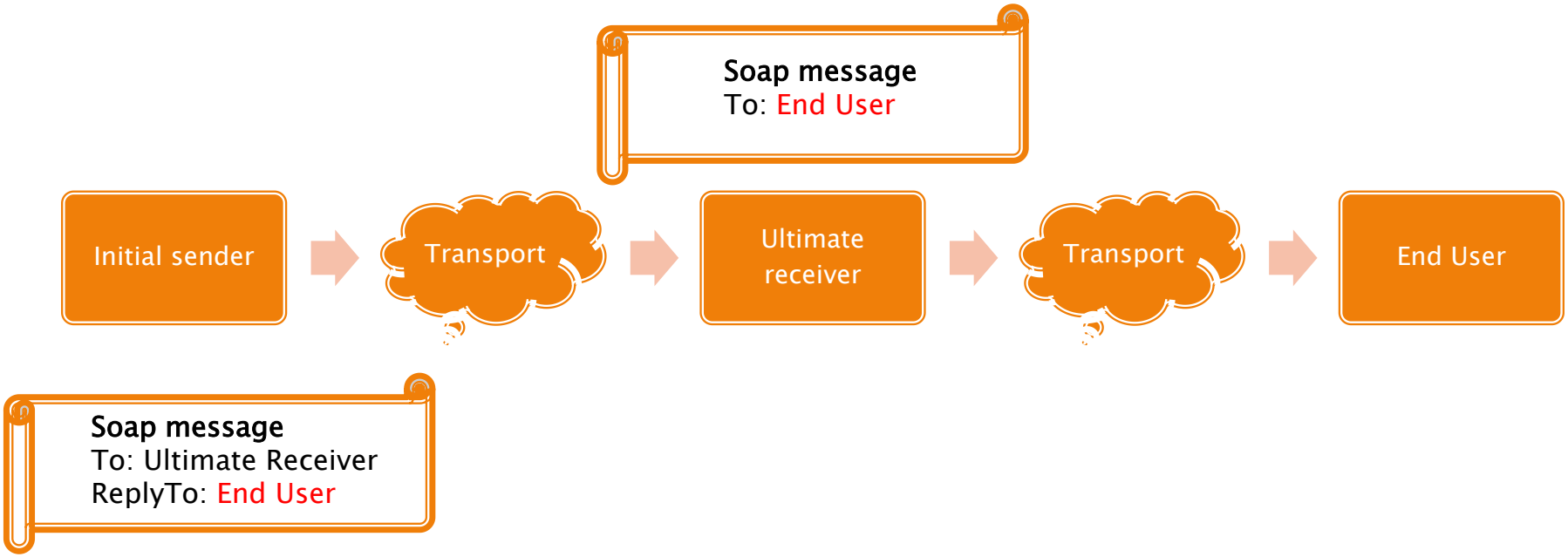
WS-Addressing goal

"Web Services Addressing normalizes the information typically provided by transport protocols and messaging systems in a way that is independent of any particular transport or messaging system."

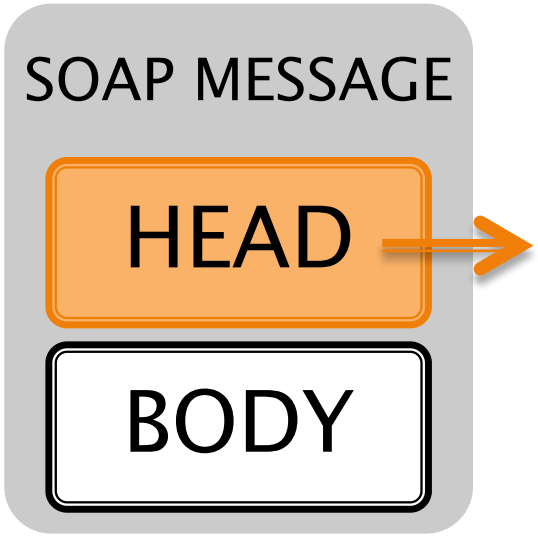
WS-Addressing: Principle



WS-Addressing: Example 2



SOAP Message with Addressing



Element	Content-Type
wsa:To	URI
wsa:From	*EndpointReferenceType*
wsa:ReplyTo	*EndpointReferenceType*
wsa:FaultTo	*EndpointReferenceType*
wsa:Action	URI
wsa:MessageID	URI
wsa:RelatesTo	URI
[reference parameters]*	any

WS-Addressing: To

- What? Address of ultimate receiver
- Content-Type? URI
- Example:

```
<wsa:To>http://www.plastics_suply.com/purchasing</wsa:To>
```

Endpoint Reference Structure

```
<wsa:EndpointReference>
  <wsa:Address>http://supply.com/purchase/wsdl</wsa:Address> *
  <wsa:ReferenceParameters></wsa:ReferenceParameters>
  <wsa:Metadata></wsa:Metadata>
</wsa:EndpointReference>
```

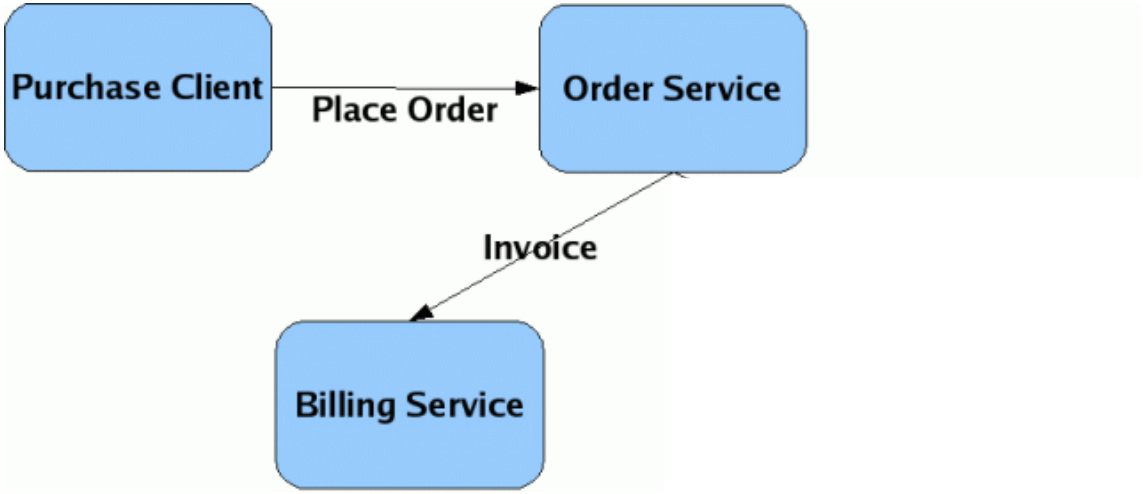
WS-Addressing: From

- What? Reference Type of where the message originated


```
<wsa:From>
  <wsa:Address>http://shop.com/purchase_client</wsa:Address> *
  <wsa:ReferenceParameters></wsa:ReferenceParameters>
  <wsa:Metadata></wsa:Metadata>
</wsa:From>
```

WS-Addressing: ReplyTo

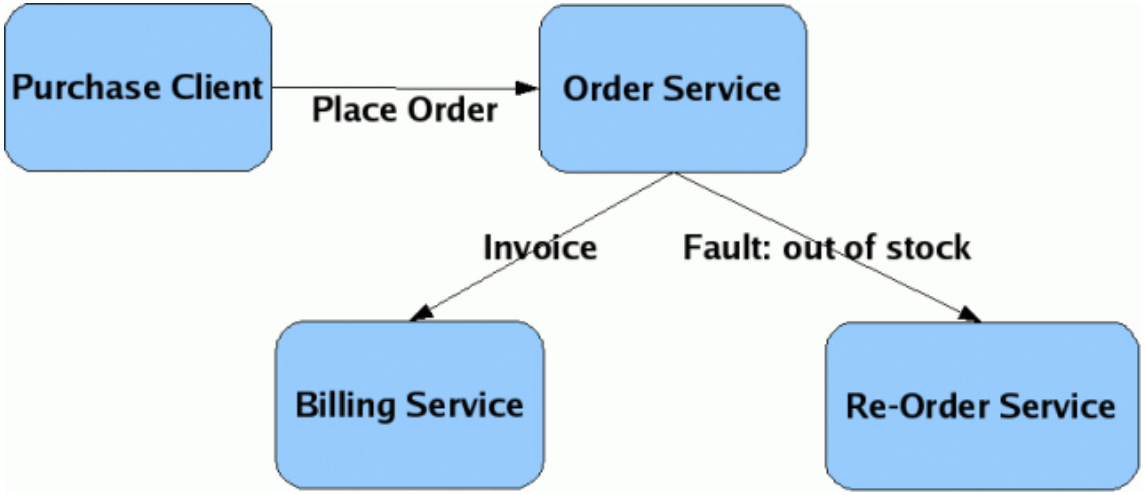
- What? Reference Type of where a response must be sent




```
<wsa:ReplyTo>
  <wsa:Address>http://supply.com/billing_service</wsa:Address> *
  <wsa:ReferenceParameters></wsa:ReferenceParameters>
  <wsa:Metadata></wsa:Metadata>
</wsa:ReplyTo>
```

WS-Addressing: FaultTo

- What? Reference Type of where faults must be sent




```
<wsa:FaultTo>
  <wsa:Address>http://supply.com/reorder_service</wsa:Address> *
  <wsa:ReferenceParameters></wsa:ReferenceParameters>
  <wsa:Metadata></wsa:Metadata>
</wsa:FaultTo>
```

WS-Addressing: Action

- What? Action that the receiver must undertake with the message
- Content-Type? URI
- Example:

```
<wsa:Action>
  http://supply.com/SubmitOrder
</wsa:Action>
```

WS-Addressing: MessageID

- What? URI to uniquely identify the message
- Content-Type? URI
- Example:

```
<wsa:MessageID>
  550e8400-e29b-41d4-a716-446655440000
</wsa:MessageID>
```

WS-Addressing: RelatesTo

- What?
  - Specifies a relation between this message and another message
- Example:

```
<wsa:RelatesTo
  RelationShipType="http://www.w3.org/2005/08/addressing/reply">
  550e8400-e29b-41d4-a716-446655440000
</wsa:MessageID>
```

WS-Addressing: Parameters

- What?
  - Add your own elements at will

WS-Addressing: XML example

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <S:Header>
      <wsa:MessageID>
            uuid:6B29FC40-CA47-1067-B31D-00DD010662DA
      </wsa:MessageID
      <wsa:ReplyTo>
            <wsa:Address>
                  http://business456.example/client1
            </wsa:Address>
      </wsa:ReplyTo>
      <wsa:To>http://fabrikam123.example/Purchasing</wsa:To>
      <wsa:Action>http://fabrikam123.example/SubmitPO</wsa:Action>
</S:Header>
<S:Body>
  …
</S:Body>
</S:Envelope>
```

2004 – 2006 specifications

- Elements were removed from Endpoint Ref.
  - PortType
  - ServiceName
  - Policy
- However:
  - 2006 specification leaves room to freely add elements

WS-Addressing and REST

- WS-Addressing abstracts technology from SOAP Messages
- REST uses properties of HTTP

Web services and stateful resources

- **Program state** – ability to remember information between successive computations
- **Stateful interaction** keeps track of client state between method invocations
- **State** – piece of information outside the contents of a request message, which is needed in order to properly process the request

WS-Resource Framework (WS-RF)

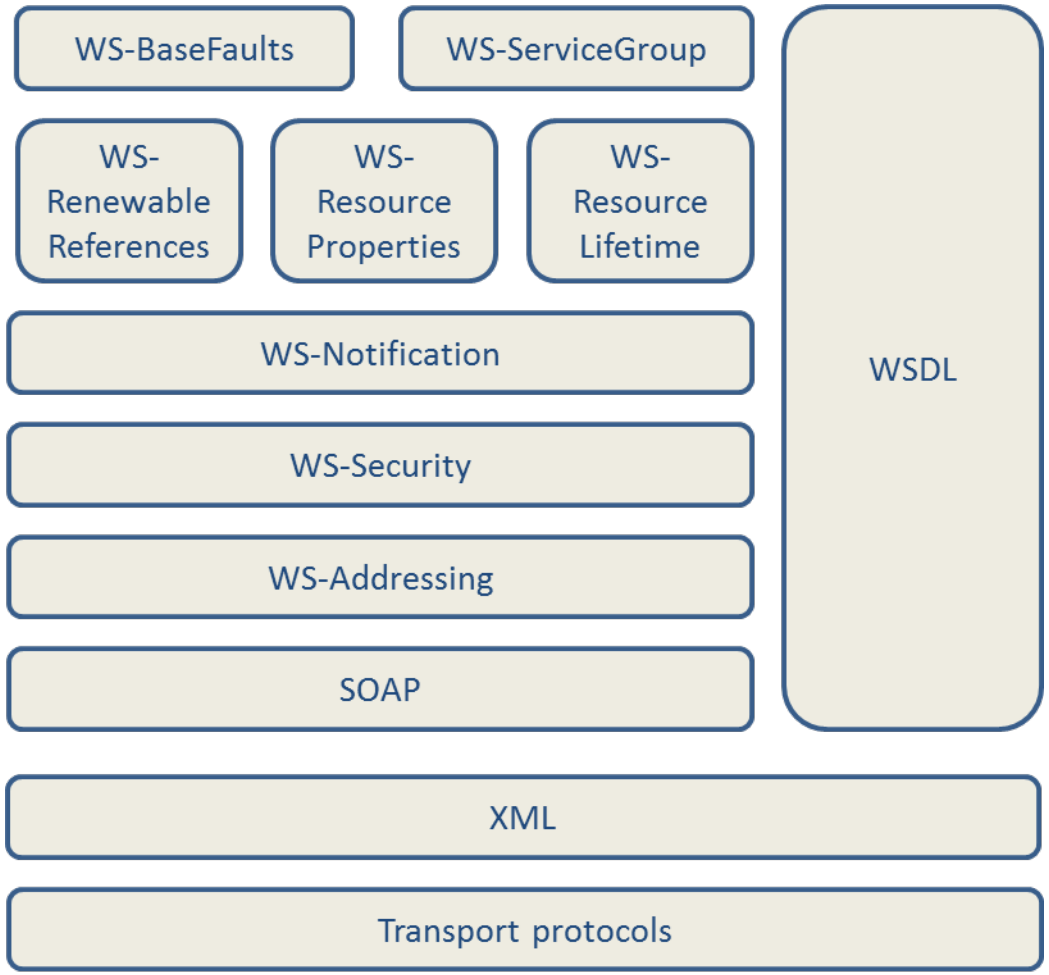
- Collection of specifications that provide the ability to model stateful resources using web services.
- Concerned with **creation**, **addressing**, **inspection** and **lifetime management** of stateful resources
- Relationship between Web services and stateful resources sits at the core of WS-RF

WS-Resource Framework (WS-RF)

- WS-RF specifications:
  - **WS-ResourceProperties**
  - **WS-ResourceLifeTime**
  - **WS-RenewableReferences**
  - **WS-ServiceGroup**
  - **WS-BaseFault**
  - **WS-Notification** family of specifications

- Defines the WS-Resource approach to represent and manage state

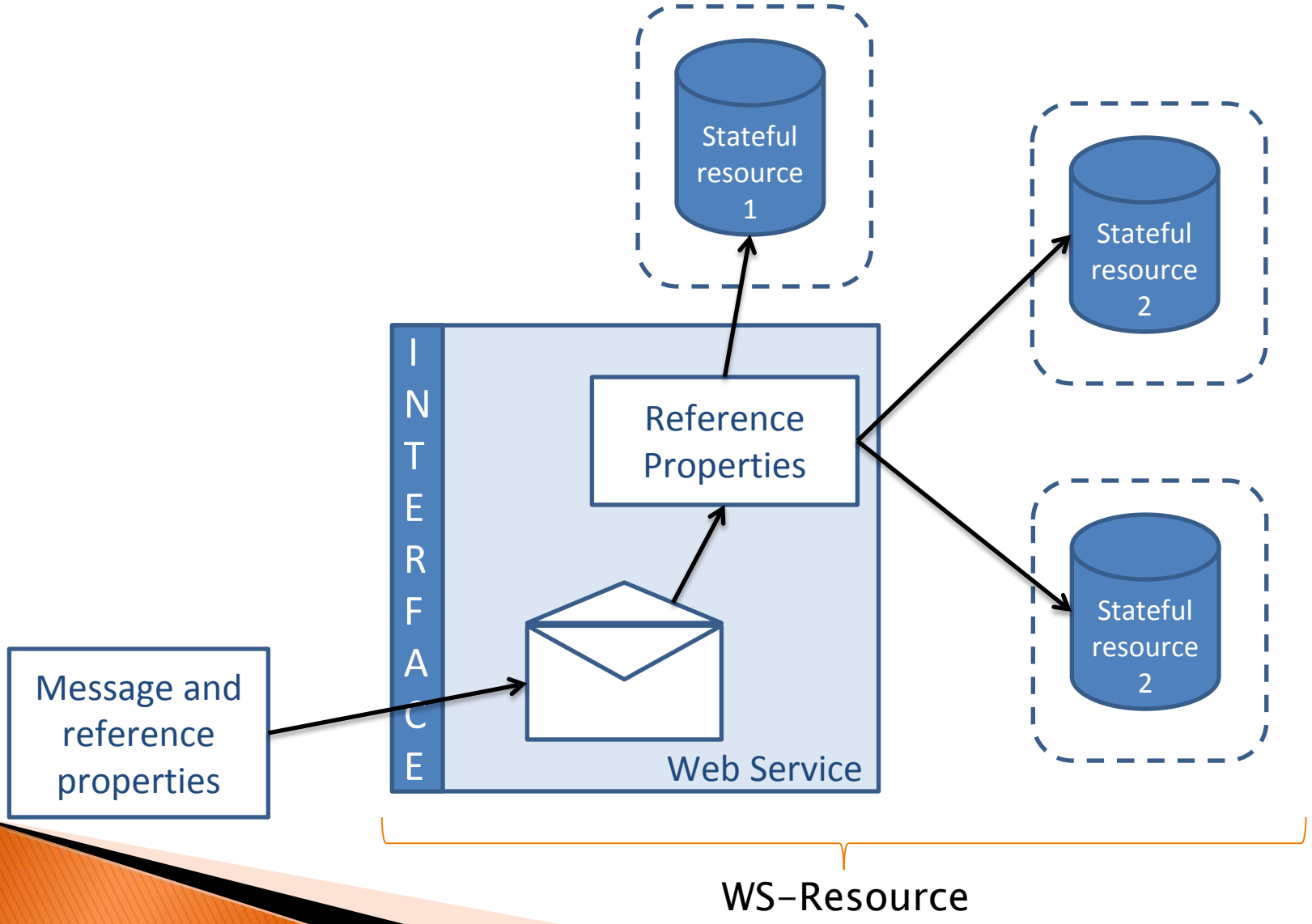
Relationship between WS-RF and other Web services standards



WS-Resource construct

- A combination of a Web service and a stateful resource
- An association of an XML document with a defined type with a Web services <**PortType**> element
- Properties of a WS-Resource are accessible through a Web service interface.

Web service and associated WS-Resources



```
<wsdl:message name="POMessage">
        <wsdl:part name="PurchaseOrder" type="tns:POType"/>
        <wsdl:part name="CustomerInfo"
type="tns:CustomerInfoType"/>
</wsdl:message>
<wsdl:message name="InvMessage">
        <wsdl:part name="POEndPointReference" element=
"tns:POReference" />
</wsdl:message>
<wsdl:portType name="PurchaseOrderPortType">
        <wsdl:operation name="SendPurchase">
                <wsdl:input message="tns:POMessage"/>
                <wsdl:output message="tns:InvMessage"/>
         </wsdl:operation>
</wsdl:portType>
```

Important constructs of the endpoint reference

Resource Properties

Requirements for stateful resources

- Determine type of state
- Message exchanges
- Issue read update requests and queries against state components

WS-RF uses XML to define resource property elements

Resource Properties

- A set of components called resource property elements.
- **Resource property element**
  - An atomic element of state that can be read or updated.
- **Resource property document**
  - A set of resource property elements gathered together into an XML document

WS-ResourceProperties document

- A projection of the resource properties of the WS-Resource
- Defines a basis for access to the resource properties through the Web services interface

WS-ResourceProperties document

- **ResourceProperties attribute**
  - specify the XML definition of the state of a WS-Resource
  - Indicates that the application adheres to the WS-ResourceProperties standard.
- **<portType>**
  - Associates the document with a Web services interface
  - Defines the overall type of a WS-Resource

```
<wsdl:definitions name="PurchaseOrder" xmlns:tns="http://supply.com/PurchaseService/wsdl" ... >
<!— Type definitions -->
<wsdl:types>
        <xsd:schema targetNamespace="http://supply.com/PurchaseService/poResource.wsdl">
            ... ...
            <xsd:element name="ErrorMessage" type="xsd:string" />
            <xsd:element name="poResourceID" type="xsd:positiveInteger" />
        </xsd:schema>
</wsdl:types>
<!-- Message definitions -->
<wsdl:message name="ErrorMessage">
        <wsdl:part name="ErrorMessage" element="poRefProp:ErrorMessage" />
</wsdl:message>
<wsdl:message name="GetInvoiceRequest"> ... ... </wsdl:message>
 <wsdl:message name="GetInvoiceResponse">
      <wsdl:part name="GetInvoiceRequest" element="inv:invoice" />
</wsdl:message>
 <!-- Association of resource properties document to a portType -->
<wsdl:portType name="PurchaseOrderPortType" wsrp:ResourceProperties="poRefProp:poResourceProperties">
         <!—Operations supported by the PO PortType -->
       <wsdl:operation name="getInvoice">
            <wsdl:input name="GetInvoiceRequest" message="poRefProp:GetInvoiceRequest" />
            <wsdl:output name="GetInvoiceResponse" message="poRefProp:GetInvoiceResponse" />
            <wsdl:fault name="FaultyInvoice" message="poRefProp:ErrorMessage" />
       </wsdl:operation>
       <wsdl:operation name="dispatch-order"> ... </wsdl:operation>
       <wsdl:operation name="cancel-order"> ... </wsdl:operation>
       <!—- WS-RF operations supported by this portType -->
       <wsdl:operation name="GetResourceProperty"> ... </wsdl:operation>
       <wsdl:operation name="GetMultipleResourceProperties"> ... </wsdl:operation>
       <wsdl:operation name="QueryResourceProperties"> ... </wsdl:operation>
       <wsdl:operation name="SetResourceProperties"> ... </wsdl:operation>
        :
</wsdl:portType>
</wsdl:definitions>
```

Resource Lifecycle

- The period between creation and destruction
- Static or dynamic
- WS-RF addresses the following aspects of entity lifecycle:
  - Creation
  - Identity assignment
  - Destruction

Resource Creation

- Factory pattern
- WS-Resource factory
  - Web service capable of creating a WS-Resource and assign an identity to it
  - Results an endpoint reference containing a WS-Resource context that refers to the new WS-Resource.

Resource Destruction

- Immediate destruction
  - request resource's immediate destruction
  - defined in the WS-ResourceLifetime specification.
- Scheduled destruction
  - a time-based mechanism for managing the destruction of a WS-Resource.
  - establish and renew a scheduled termination time for the WS-Resource
  - defines the circumstances under which a service requestor can determine that the termination time has elapsed

Service Groups

- Standard mechanism for creating a collection of Web services
- A service group entry is a WS-Resource.

WS-ServiceGroup

- Defines means by which WS and WS-Resources can be grouped for a domain-specific purpose.

Service Groups

- Uses **resource property model** to express membership rules, constraints and classifications
- Groups – a collection of members
- ServiceGroup – maintains information about a collection of Web services
- Web services represented in the collection may be a component of a WS-Resource

WS-RF and REST-style architecture

- Using WS-RF with REST is in dispute

In REST:

- everything is a resource
- actions – a limited and standardized set of operations
- URL is all that is needed to address the resource
- no need for the complexity of the WS-Addressing ReferenceParameters
- callbacks/notifications do not go through firewalls

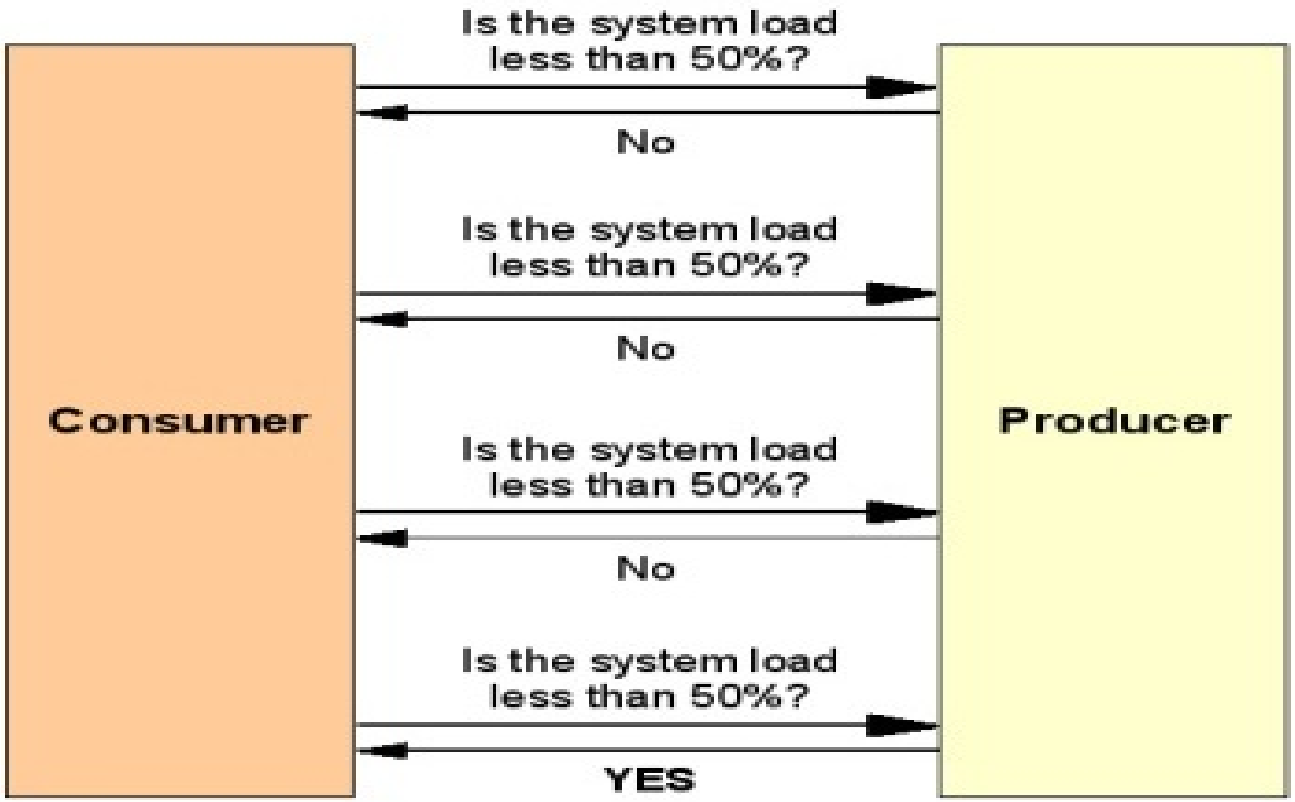
WS-Notification

- WS-Notification is a family of specifications that standardise the way web services interact between each others using “Notifications” or “Events” based on “Publish/Subscribe” pattern.

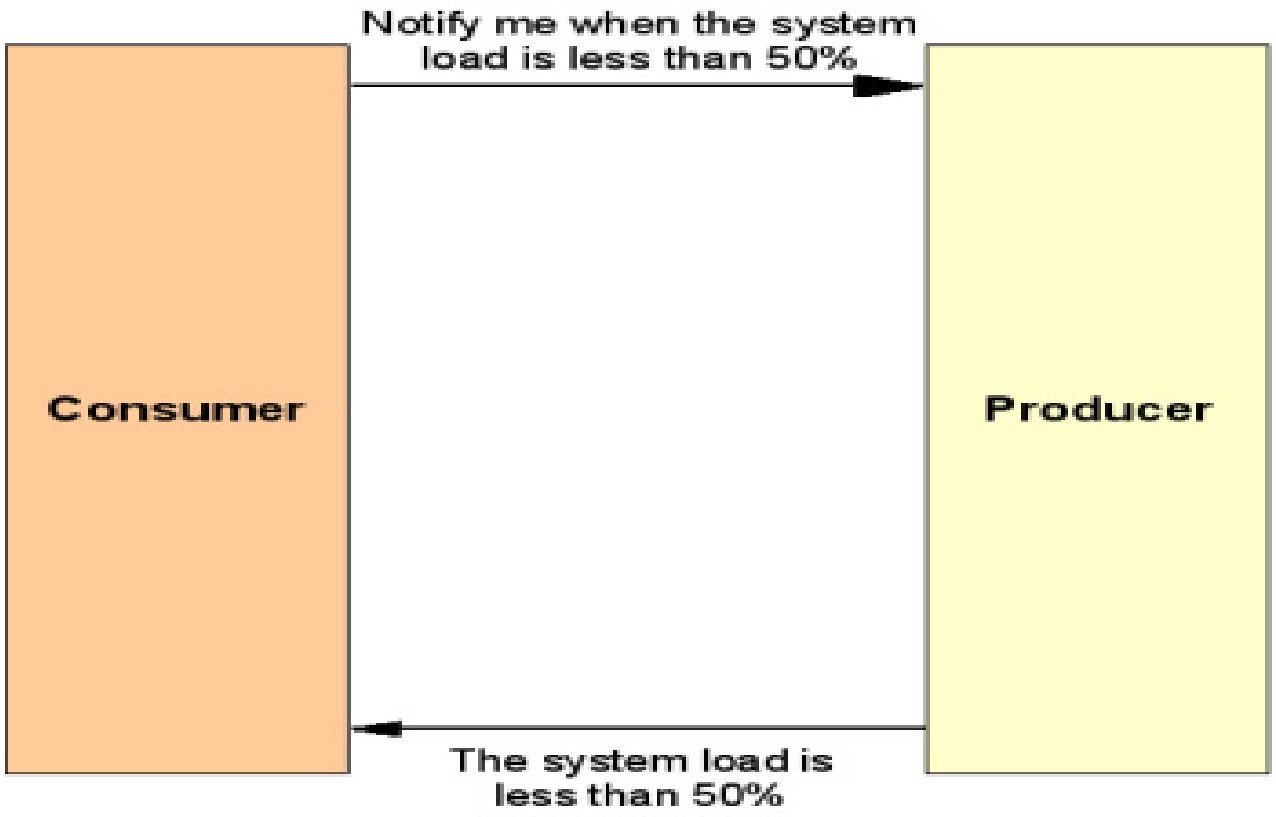
- WS-Notification’s specifications:

  - WS-BaseNotification
  - WS-Topics
  - WS-BrokeredNotification

WS-Notification



WS-Notification



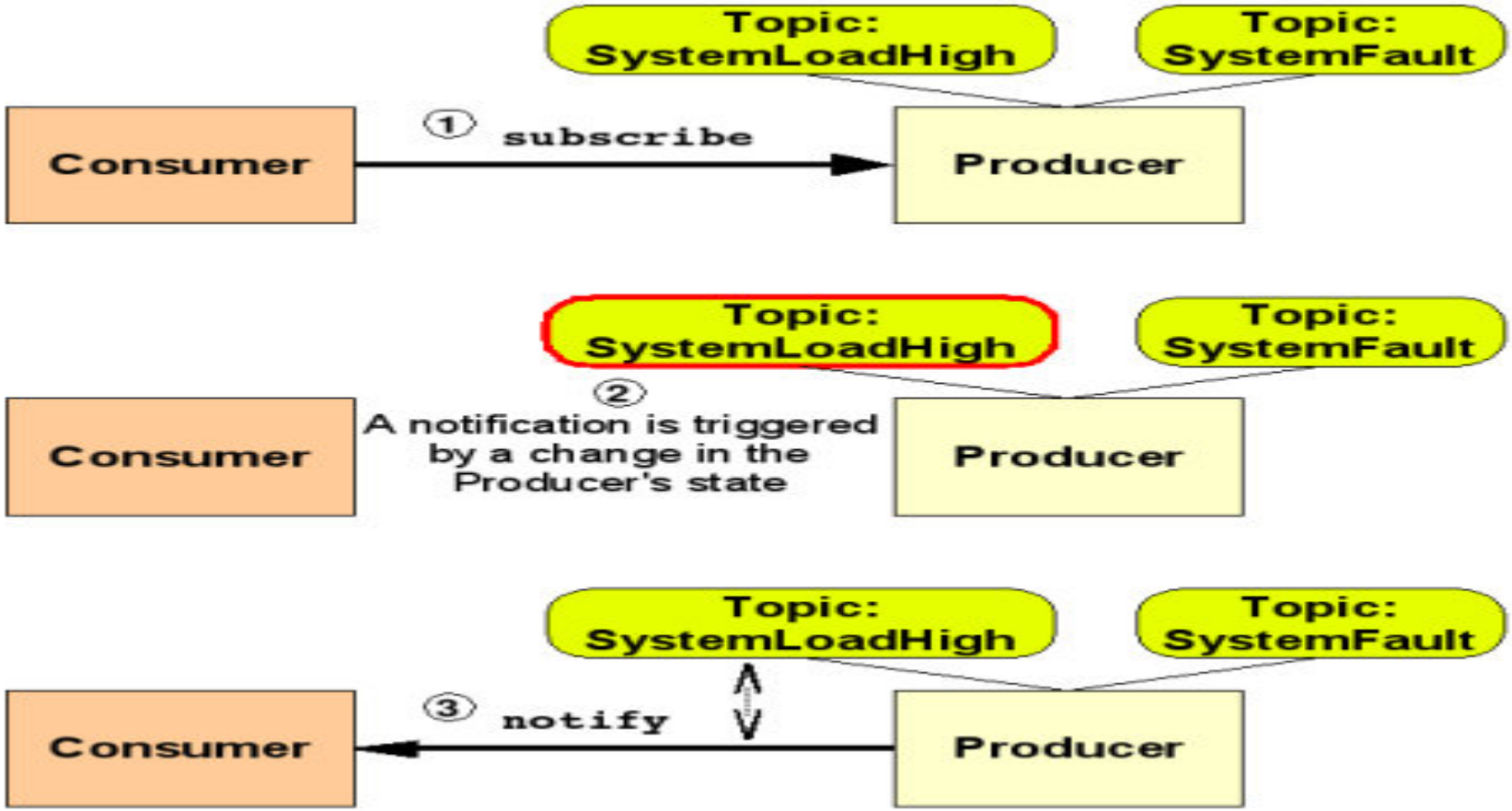
WS-Notification

- WS-Notification family of standards currently outlines only the push deliver mode for notifications, which is based on a publish/subscribe mode.
- The push model is one in which notifications are pushed to the consumer once they are available.

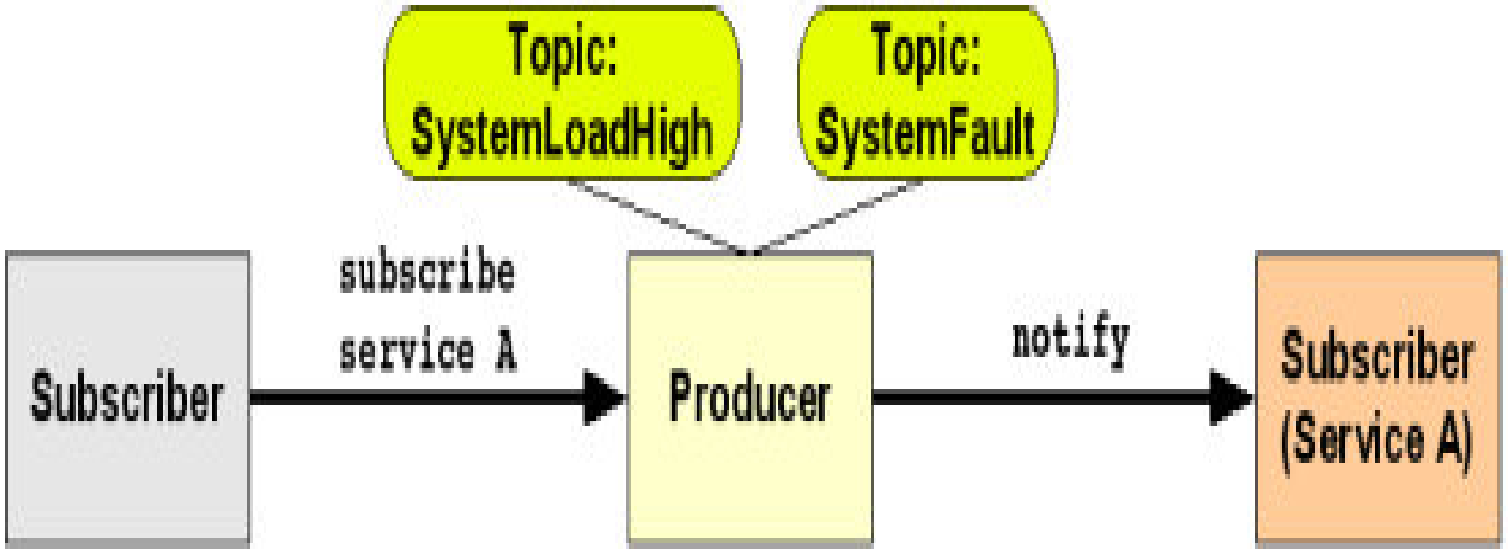
WS-BaseNotification

- The WS-BaseNotification specification defines the standard interfaces of a notification consumers and producers.
- The configuration where a NotificationConsumer is subscribed directly to a NotificationProducer is referred to as peer-to-peer, direct, or point-to-point notification pattern.

WS-BaseNotification



WS-BaseNotification



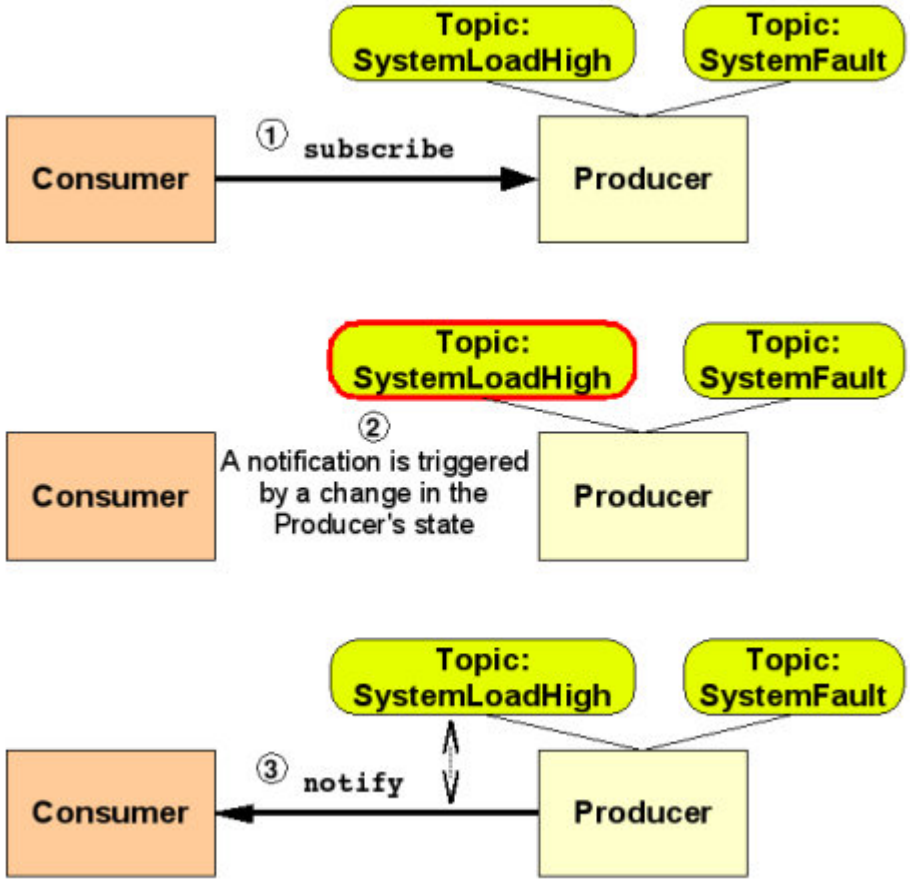
WS-BaseNotification

- A subscription is an entity that represents the relationship between a notification consumer and a notification producer.
- Each notification producer holds a list of active subscriptions, and when it has a notification to perform it matches this notification against the interest registered in each subscription in its list.
- To create a subscription, a subscriber must send specific information to a notification producer.

WS-Topics

- The WS-Topics specification defines the features required to allow application to work with topic-oriented notification systems.
- When a subscriber creates a subscription, it associates the subscription with one or more topics to indicate which kind of notifications it is interested in.
- The notification producer is responsible for matching notifications to the list of subscriptions and for sending the notification to the appropriate subscribed consumer.

WS-Topics

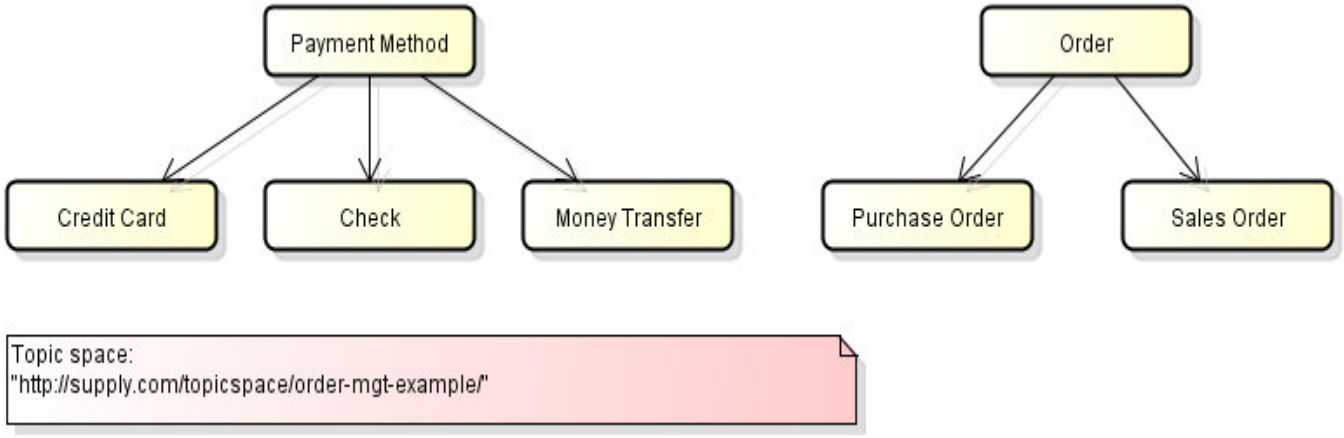

Topic: SystemLoadHigh
Topic: SystemFault
Consumer
① subscribe
Producer
Topic: SystemLoadHigh
Topic: SystemFault
Consumer
② A notification is triggered by a change in the Producer's state
Producer
Topic: SystemLoadHigh
Topic: SystemFault
Consumer
③ notify
Producer

WS-Topics

- Topic are organised hierarchically into a topic tree, where each topic may have zero or more child topics and a child topic can itself contain further child topics.
- A subscriber can subscribe to an entire topic tree or to a subset of the topics in a topic tree.
- By subscribing to a topic, a subscriber automatically receives notifications from all the descendant topics (without having to manually subscribe to each of them).

WS-Topics

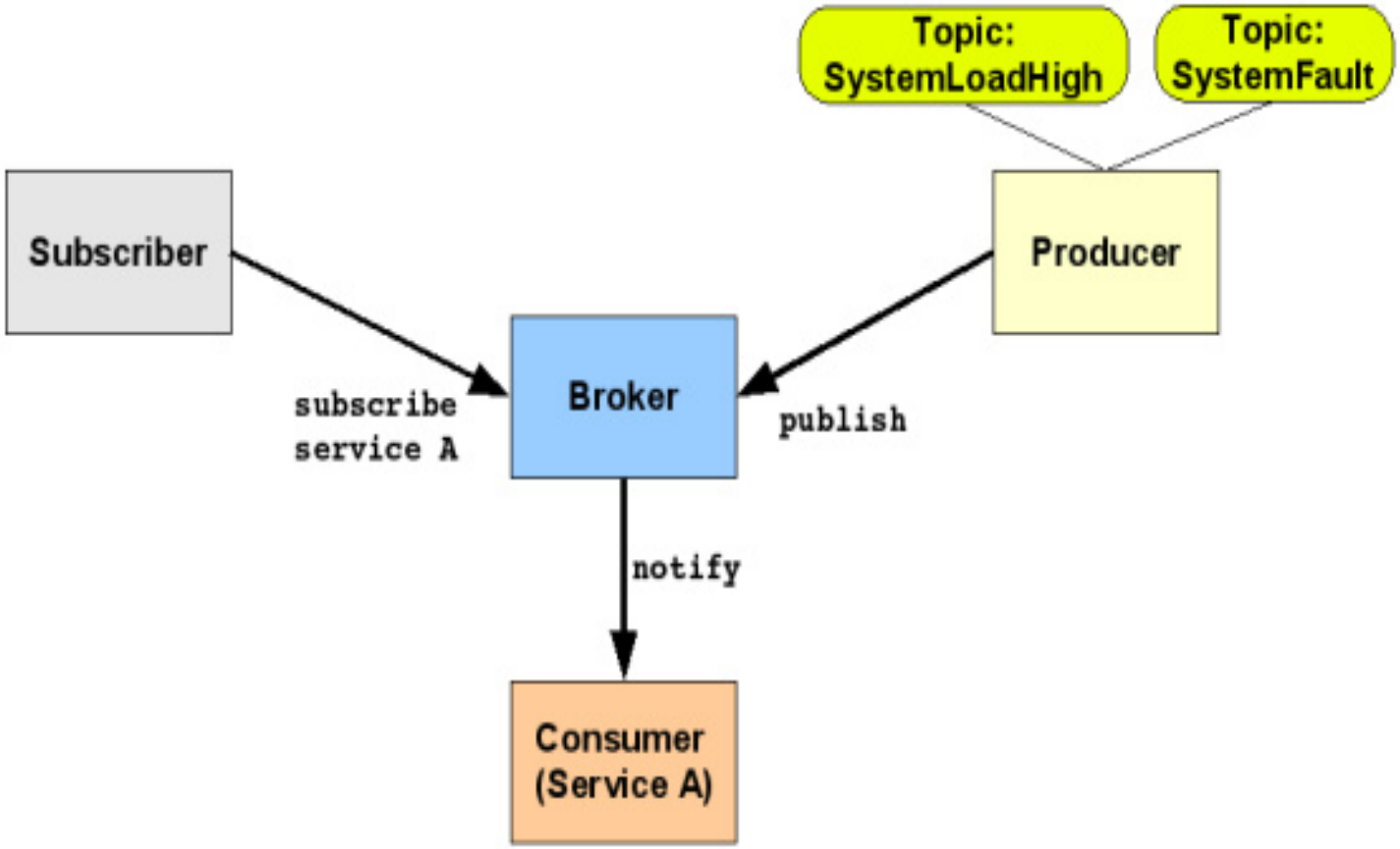
- Topics are arranged within topic spaces, which uses XML namespaces to avoid topic-definition clashes.
- A topic space is a collection (forest) of topic trees.



WS-BrokeredNotification

- The WS-BrokeredNotification introduces an additional role called a notification broker.
- A notification broker is an intermediary web service that is designed to provide scalable notification handling that decouples notification consumers from publishers.
- Brokers can improve system scalability by handling subscriptions, filtering, efficient delivery to multiple consumers, and message persistence.

WS-BrokeredNotification



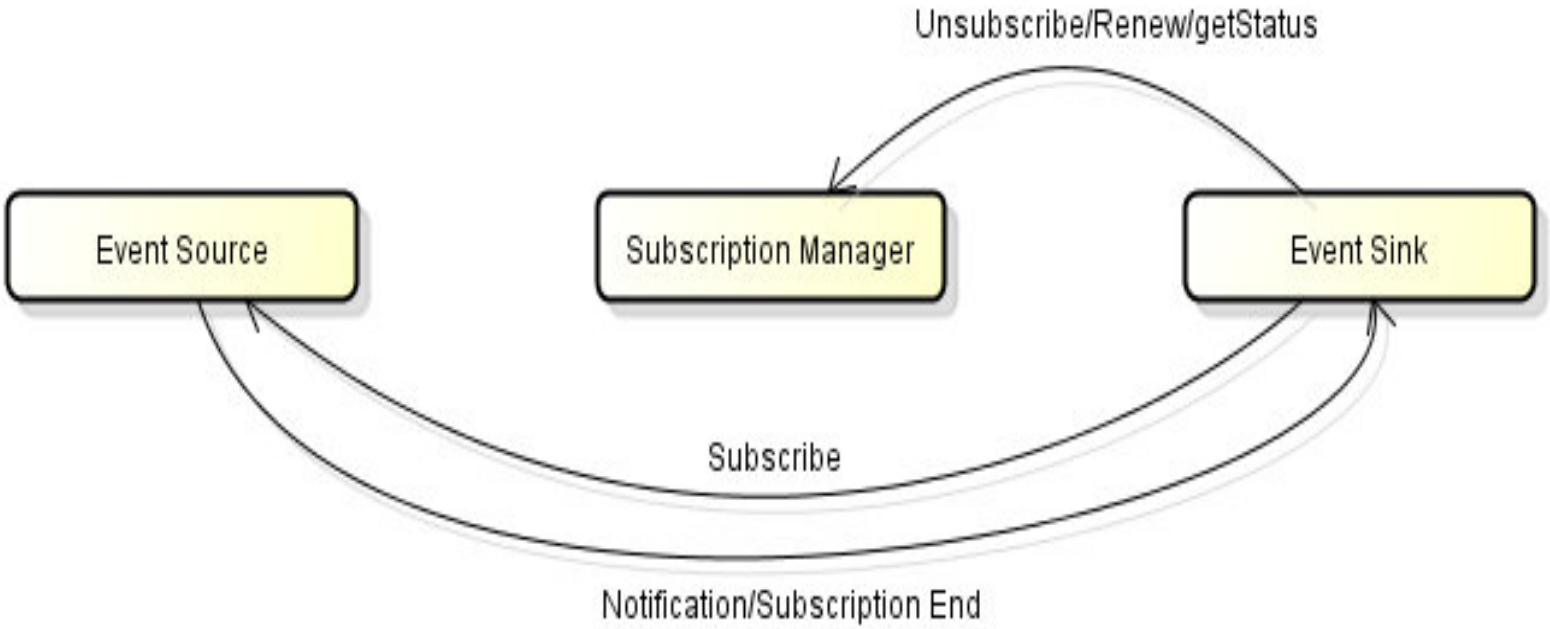
WS-Eventing

- The WS-Eventing specification is intended to define a baseline set of operations that allow web services to provide simple asynchronous notifications of events between web services to interested parties.
- WS-Eventing is a set of protocols, message formats, and interfaces that allow a web service to subscribe or accept subscriptions for event notifications.

WS-Eventing

- The WS-Eventing specification defines a protocol for one web service (a subscriber) to register interest (subscription) with another web (event source) in receiving messages about events (notifications or event messages).
- The subscriber may manage the subscription by interacting with a special-purpose web service called the subscription manager designated by the event source.
- The subscription manager accepts requests to manage, get the status of, renew, and/or delete subcriptions on behalf of an event source.

WS-Eventing


Unsubscribe/Renew/getStatus
Event Source
Subscription Manager
Event Sink
Subscribe
Notification/Subscription End

WS-Notification VS WS-Eventing

- The major difference between WS-Eventing and WS-Notification is that the model of WS-Eventing requires tight coupling between "event filter" declaration and the format of the streaming content, which use XPath expressions, while WS-Notification uses a more loosely coupled topic-based subscription method.

- Another difference is that WS-Notification supports intermediary brokering technologies for publish-and-subscribe web services paradigms, while WS-Eventing currently does not.

Questions

Thank You

References

- Addressing:
- http://www.w3.org/TR/ws-addr-core/
- M. P. Papazoglou, Web Services: Principles and Technology, chapter 7, pages 215-250.
- http://wso2.com/library/2605/

- WS-Notification and WS-Eventing
- M. P. Papazoglou, Web Services: Principles
- http://acs.lbl.gov/projects/gtg/projects/pyGridWare/doc/tutorial/html/x1713.html