Cloud Computing: SOAP based Web Services

Vijay Dialani, PhD
Boise State University
vijaydialani@boisestate.edu

Cloud Computing – Examples for this Lecture

- https://github.com/vdialani/CloudComputing.git
- Look for examples in “chapter2” folder of this repository.

Cloud Computing – SOAP

What is SOAP?

- SOAP (Simple Object Access Protocol) is a lightweight protocol for exchange of objects between services in a distributed system. SOAP uses XML to define an extensible, interoperable and programming model independent format for passing one-way messages between systems.

SOAP has three major characteristics:

1. *Extensibility:* Security and WS-routing are among the extensions under development
2. *Neutrality*: SOAP can be used over any transport protocol such as HTTP, SMTP, TCP, UDP, or JMS and
3. *Independence* , SOAP allows for any programming model.

Why SOAP?

- WSDL specifies the interface for the web service. Once the object has been serialized it needs to be marshaled/routed across the network from a sender to the receiver.

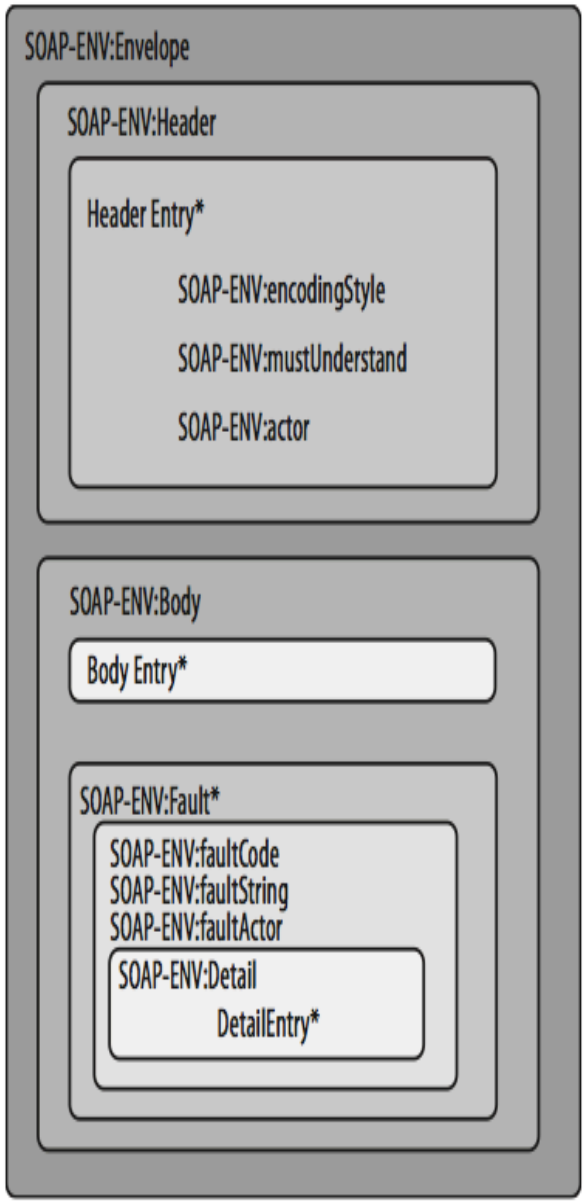
- When the message is being routed on the network, it may require:
  - secure transport
  - multi-cast capability
  - guaranteed delivery
  - callback handling

- SOAP facilitates all of the above mentioned properties of message routing across the networks.

- A SOAP message is composed of three sub-parts (more discussion on this latter.)
  - A SOAP Envelope
  - A SOAP Header
  - A SOAP Body

- SOAP messages allow encryption of the body and the header and the communication between the sender and receiver could be routed through the information in the Envelope.

Structure of a SOAP message

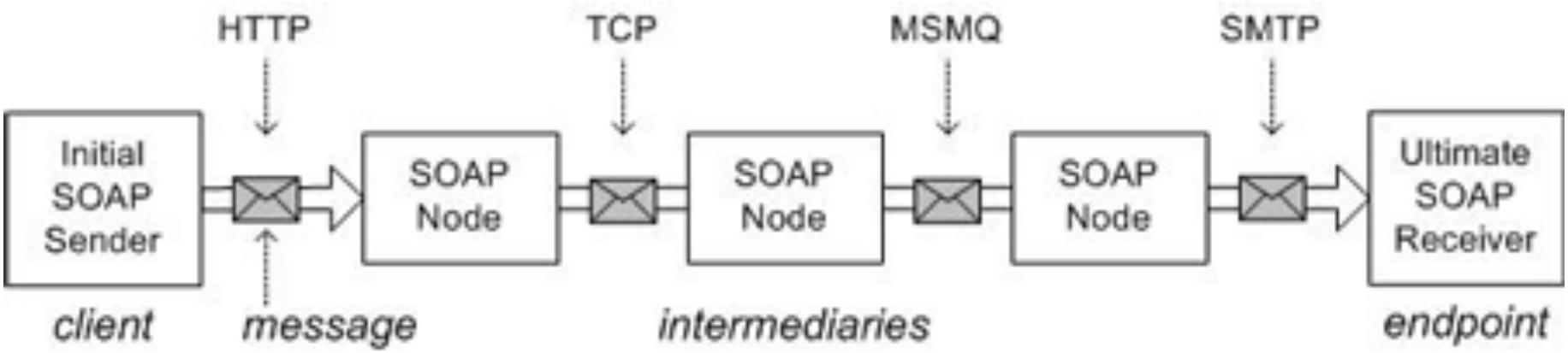


Example SOAP Message

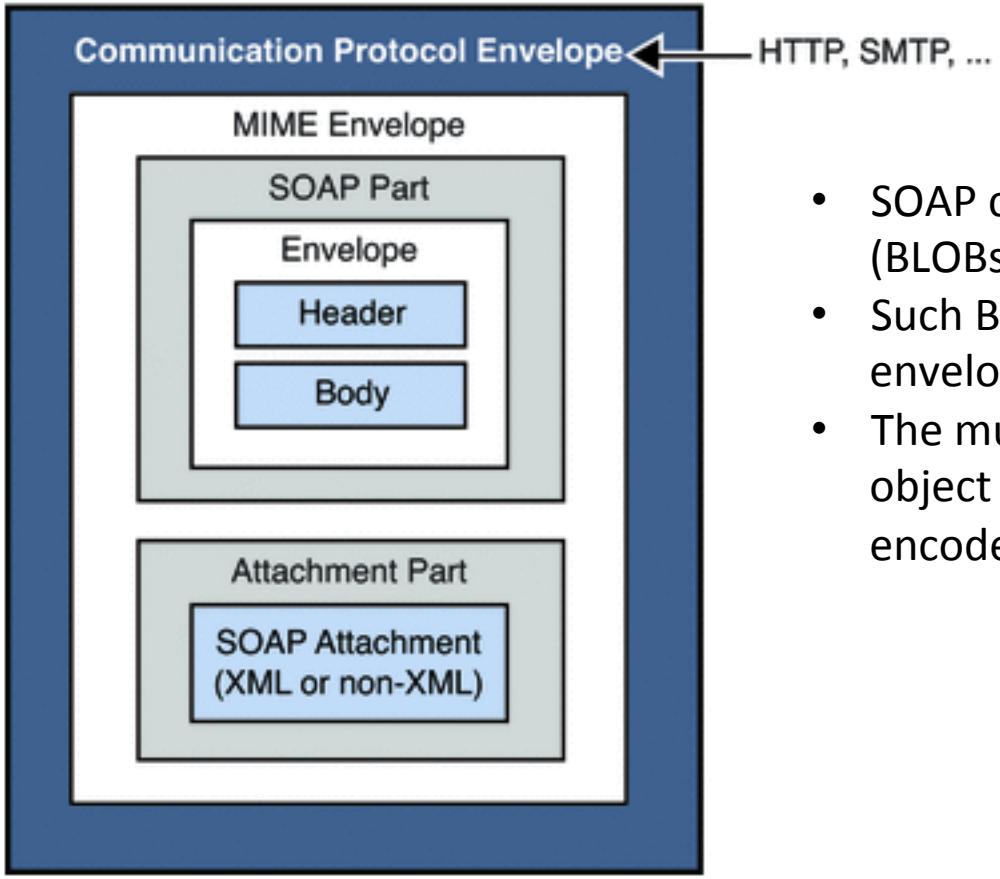
```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 299
SOAPAction: "http://www.w3.org/2003/05/soap-envelope"

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
 <soap:Header>
 </soap:Header>
 <soap:Body>
  <m:GetStockPrice
      xmlns:m="http://www.example.org/stock">
   <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
 </soap:Body>
</soap:Envelope>
```

SOAP: Interoperability

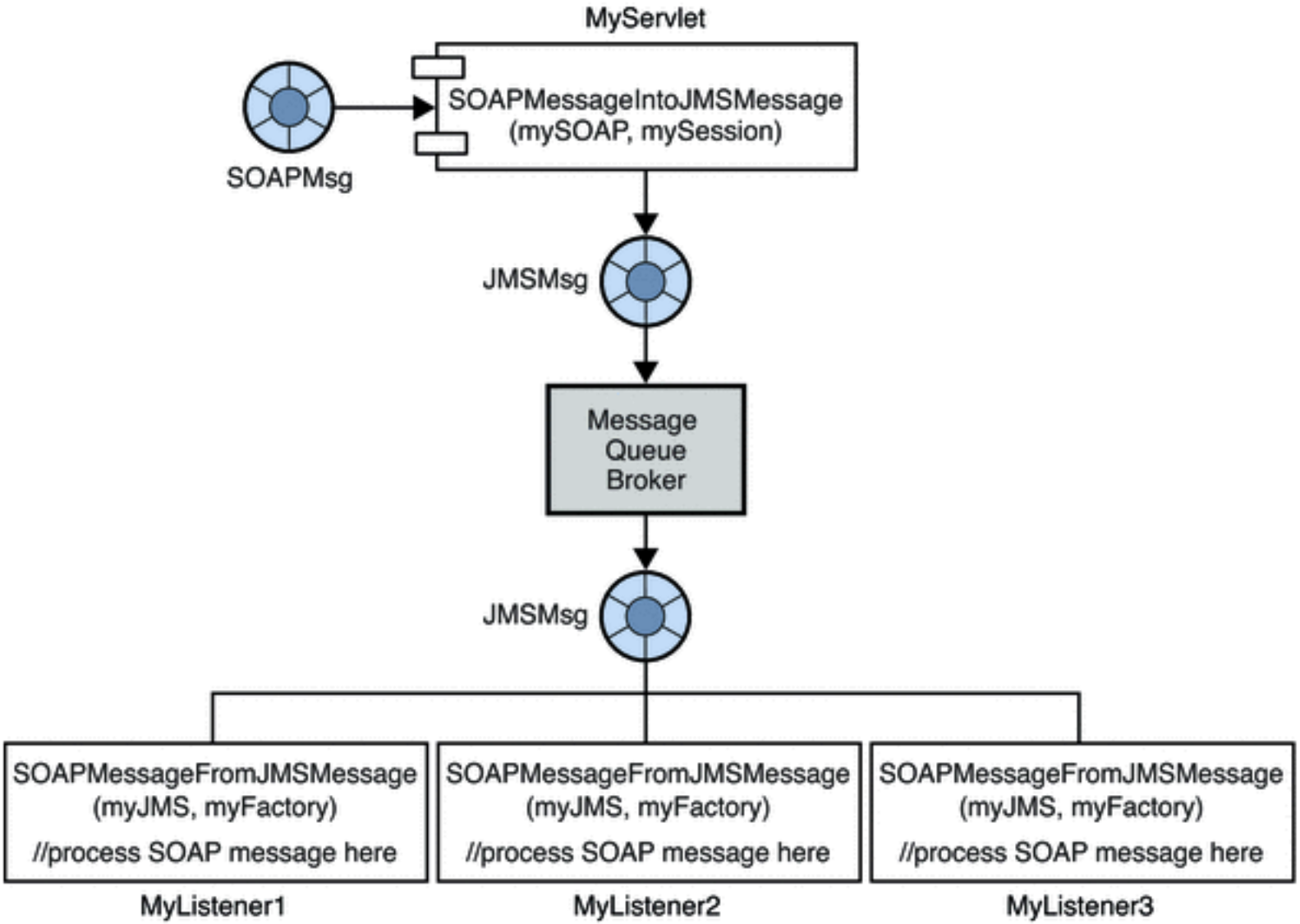


SOAP: MIME messages



- SOAP can be used to transmit Binary Large OBjects (BLOBs).
- Such BLOBs are wrapped in the MIME message envelope used by SMTP and other protocols
- The multipart message will contain the usual SOAP object and could also have parts of the messages encoded for security reasons.

SOAP Multicast



Simple SOAP Service

Let us walk through the example on git.

WS-Security

WS-Security describes three main mechanisms:

1. How to sign SOAP messages to assure integrity. Signed messages also provide non-repudiation?
2. How to encrypt SOAP messages to assure confidentiality?
3. How to attach security tokens to ascertain the sender's identity?

The specification allows a variety of signature formats, encryption algorithms and multiple trust domains, and is open to various security token models, such as:

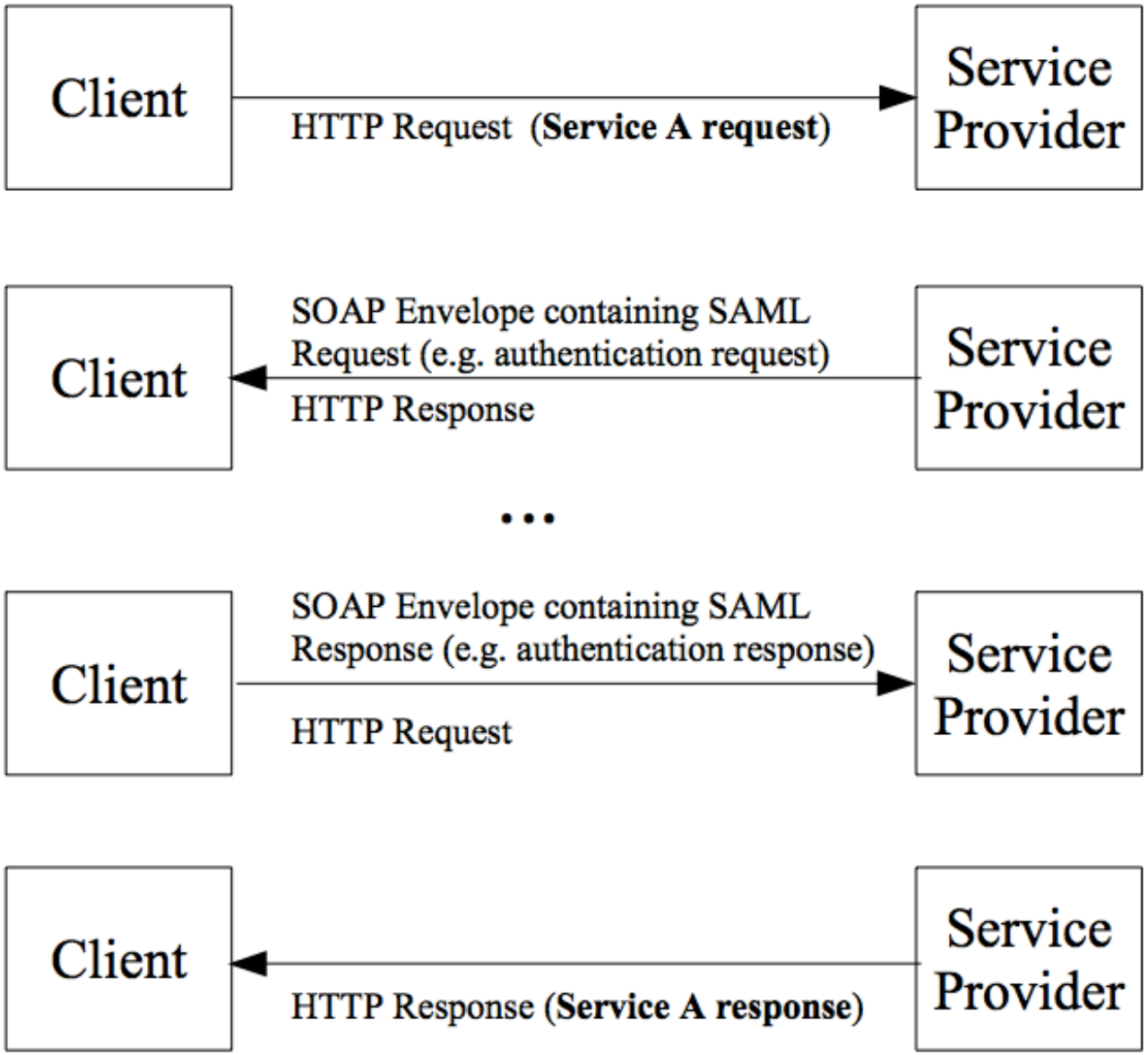
- X.509 certificates,
- Kerberos tickets,
- UserID/Password credentials,
- SAML Assertions, and
- custom-defined tokens.

SOAP: Security Alternatives

Security Specification

1. WS-Security
2. XML Signature
3. XML Encryption
4. XML Key Management (XKMS)
5. WS-SecureConversation
6. WS-SecurityPolicy
7. WS-Trust
8. WS-Federation
9. WS-Federation Active Requestor Profile
10. WS-Federation Passive Requestor Profile
11. Web Services Security Kerberos Binding
12. Web Single Sign-On Interoperability Profile
13. Web Single Sign-On Metadata Exchange Protocol
14. Security Assertion Markup Language (SAML)
15. XACML

Security example – SAML over SOAP



Security example – SAML over SOAP -I

```
POST /SamlService HTTP/1.1
Host: www.example.com
Content-Type: text/xml
Content-Length: nnn
SOAPAction: http://www.oasis-open.org/committees/security
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <samlp:AttributeQuery xmlns:samlp:="…"
xmlns:saml="…" xmlns:ds="…" ID="_6c3a4f8b9c2d" Version="2.0"
IssueInstant="2004-03-27T08:41:00Z"
            <ds:Signature> … </ds:Signature>
            <saml:Subject>
            …
            </saml:Subject>
        </samlp:AttributeQuery>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Security example – SAML over SOAP -II

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: nnnn
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <samlp:Response xmlns:samlp="…" xmlns:saml="…" xmlns:ds="…"
ID="_6c3a4f8b9c2d" Version="2.0" IssueInstant="2004-03-27T08:42:00Z">
            <saml:Issuer>https://www.example.com/SAML</saml:Issuer>
            <ds:Signature> … </ds:Signature>
            <Status>
              <StatusCode Value="…"/>
            </Status>

            <saml:Assertion>
                <saml:Subject>
                …
                </saml:Subject>
                <saml:AttributeStatement>
            …
                </saml:AttributeStatement>
             </saml:Assertion>
        </samlp:Response>
    </SOAP-Env:Body>
```

Classroom exercise

Quiz

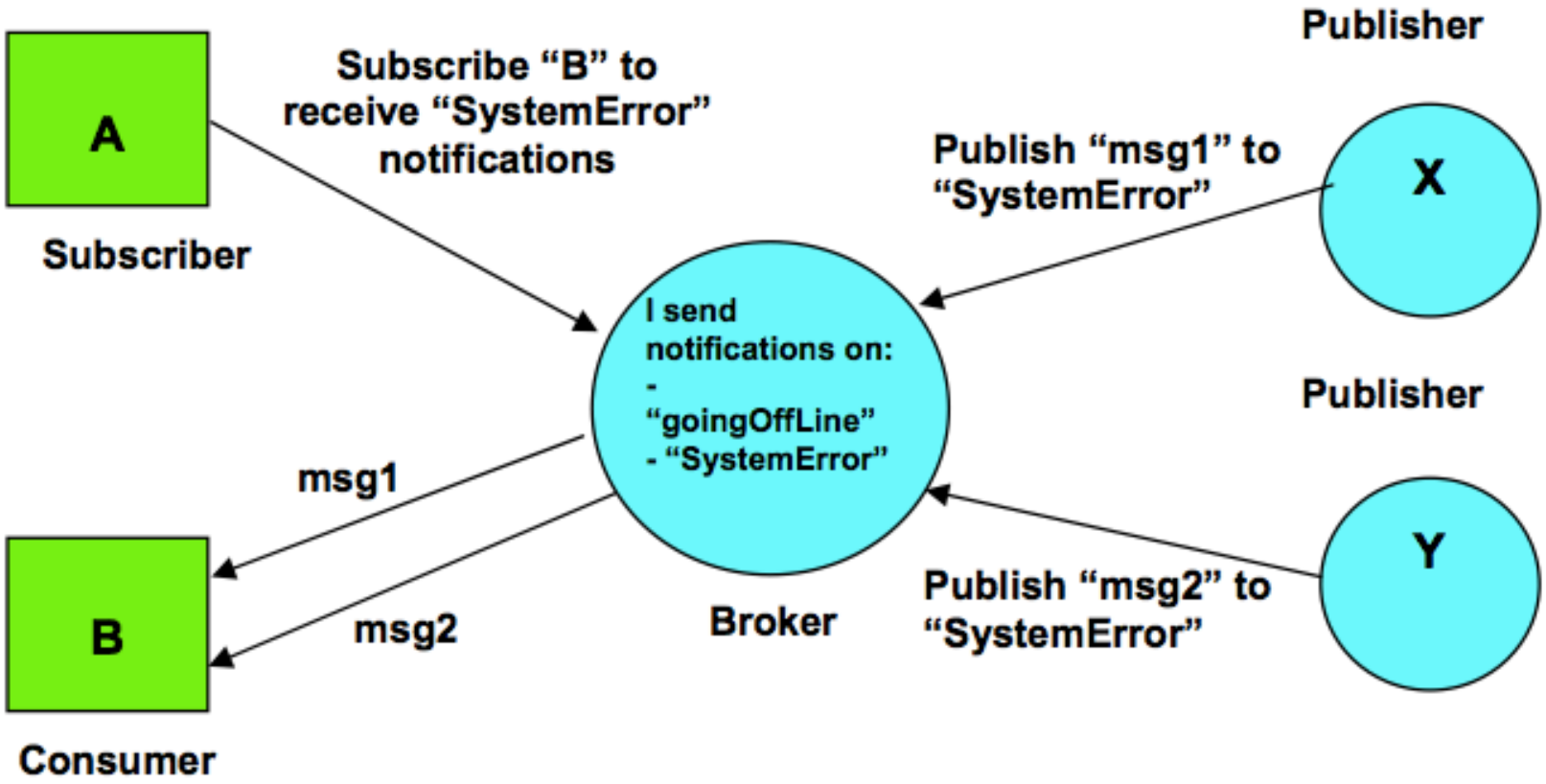
SOAP – messaging specifications

1. SOAP Message Transmission Optimization Mechanism
2. WS-Notification
   - WS-BaseNotification
   - WS-Topics
   - WS-BrokeredNotification
3. WS-Addressing
4. WS-Transfer
5. WS-Eventing

There are numerous messaging protocols, many are to do with the possible ways of transmitting the messages in async message queue frameworks.

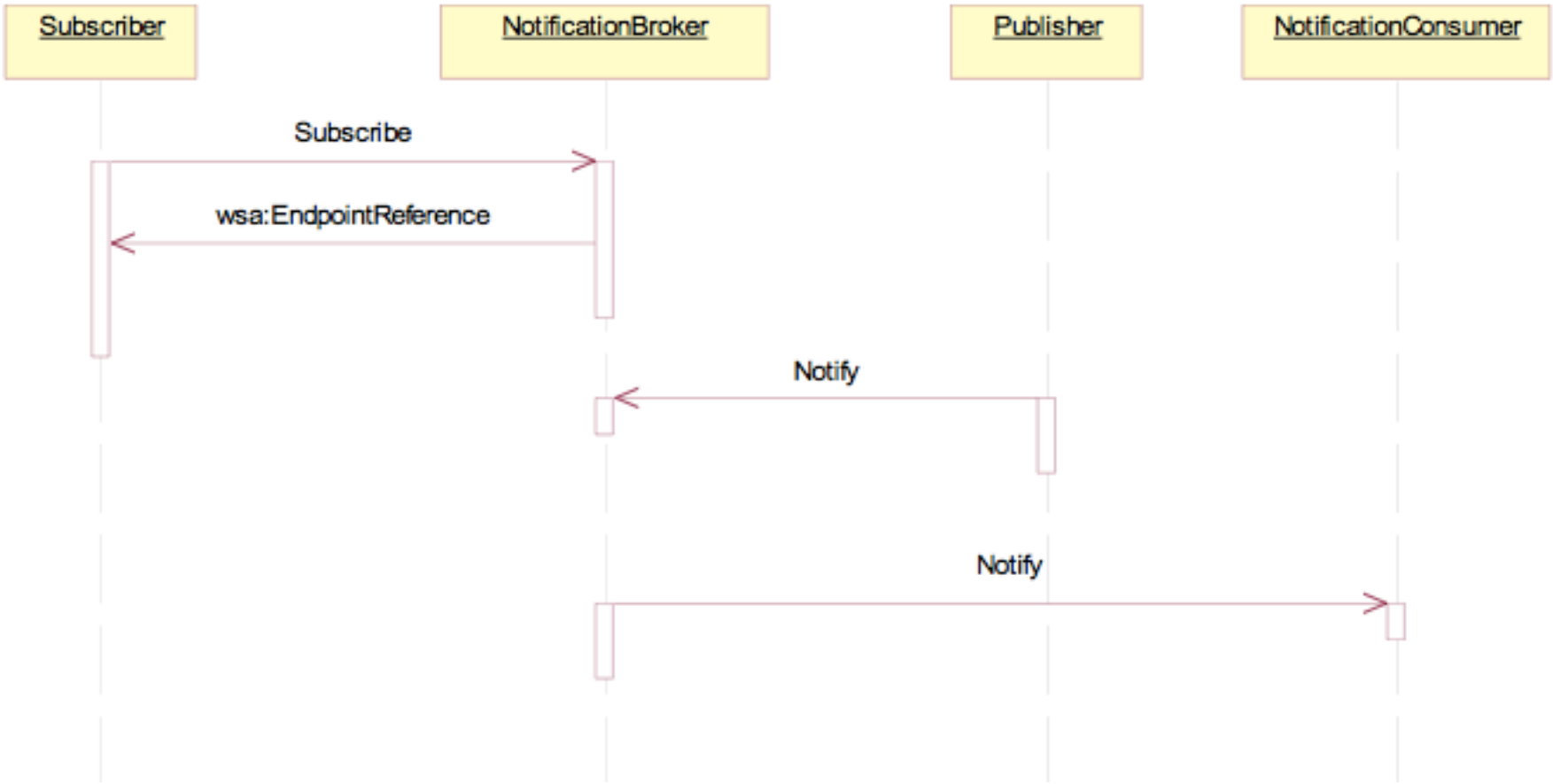
Here we will be focusing on WS-Addressing

SOAP – WS-Notification



http://www.ibm.com/developerworks/library/ws-pubsub/WS-PubSub.pdf

SOAP – WS-Notification



http://www.ibm.com/developerworks/library/ws-pubsub/WS-PubSub.pdf

Callbacks – WS Addressing

WS-Addressing is a standardized way of including message routing data within SOAP headers.

Instead of relying on network-level transport to convey routing information, a message utilizing WS-Addressing may contain its own dispatch metadata in a standardized SOAP header.

The network-level transport is only responsible for delivering that message to a dispatcher capable of reading the WS-Addressing metadata. Once that message arrives at the dispatcher specified in the URI, the job of the network-level transport is done.

Callbacks – WS Addressing – Endpoint Reference

An endpoint reference (EPR) is an XML structure encapsulating information useful for addressing a message to a Web service. This includes the destination address of the message, any additional parameters (called reference parameters) necessary to route the message to the destination, and optional metadata (such as WSDL or WS-Policy) about the service.

Message addressing properties communicate addressing information relating to the delivery of a message to a Web service:

- Message destination URI
- Source endpoint -- the endpoint of the service that dispatched this message (EPR)
- Reply endpoint -- the endpoint to which reply messages should be dispatched (EPR)
- Fault endpoint -- the endpoint to which fault messages should be dispatched (EPR)
- Action -- an action value indicating the semantics of the message (may assist with routing the message) URI
- Unique message ID URI
- Relationship to previous messages (A pair of URIs)

Callbacks – WS Addressing – Example

Let us look at the example in git

Web Services: SOAP

Understanding SOAP
http://msdn.microsoft.com/en-us/library/ms995800.aspx

Java SOA CookBook (available on blackboard)

Working with SOAP
http://docs.oracle.com/cd/E19798-01/821-1796/aeqex/index.html

SAML v2.0
http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf