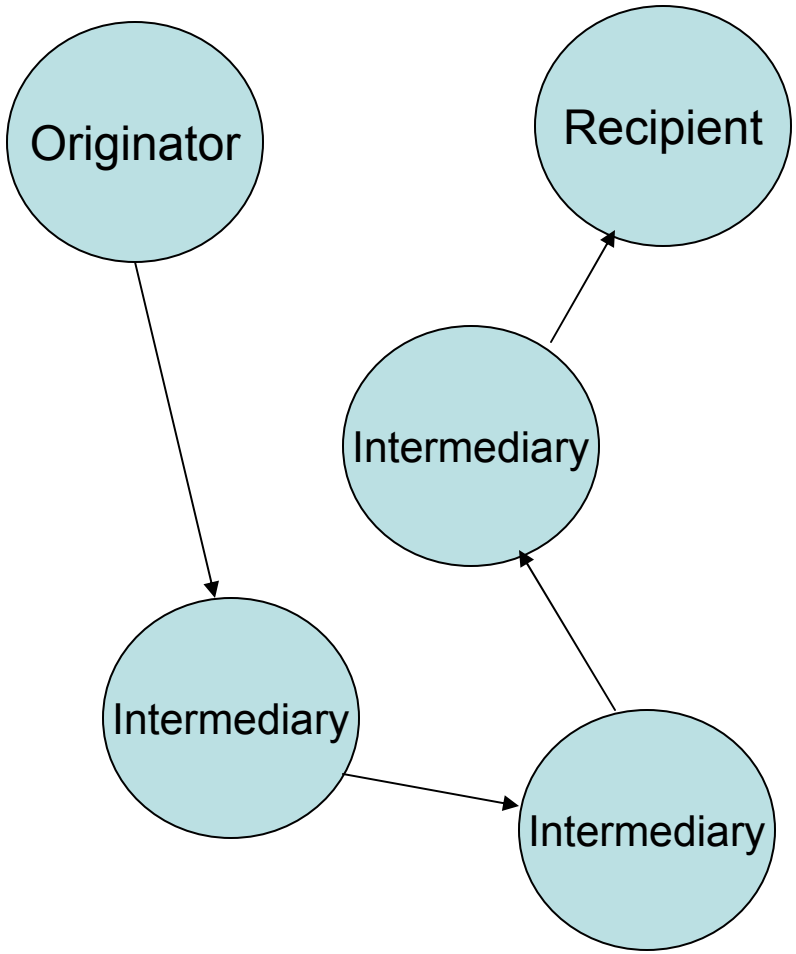
SOAP Routing and Processing Concepts

Marlon Pierce, Bryan Carpenter, Geoffrey Fox
Community Grids Lab
Indiana University
mpierce@cs.indiana.edu
http://www.grid2004.org/spring2004

SOAP Processing Assumptions

- SOAP assumes messages have an *originator*, one or more *ultimate receivers*, and zero or more *intermediaries*.
- The reason is to support distributed message processing.
- That is, we can go beyond client-server messaging.



Processing and SOAP Structure

- SOAP processing rules are directly related to the SOAP message envelope:
  - The Body is only for final recipients.
  - Header sections may be processed by one or more *intermediaries* as well as final recipient nodes.
  - SOAP headers are the extensibility elements for defining other features.
- The Header therefore has three optional attributes:
  - Role (called *actor* in SOAP 1.0 and 1.1): Determines is a header should process a particular header.
  - mustUnderstand: If set to “true”, the node must know how to process the header.
  - Relay: Indicates whether or not an unprocessed header block should be forwarded.

Example Uses of Headers

- Security: WS-Security and SAML place additional security information (like digital signatures and public keys) in the header.
- Quality of Service: SOAP headers can be used if we want to negotiate particular qualities of service such as reliable message delivery and transactions.
  - Reliable Message is one example.
- Session State Support: Many services require several steps and so will require maintenance of session state.
  - Equivalent to cookies in HTTP.
  - Put session identifier in the header.

Example Header from SOAP Primer

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-
    envelope">
  <env:Header>
    <m:reservation xmlns:m="…"
          env:role="http://www.w3.org/2003/05/soap-
    envelope/role/next"    env:mustUnderstand="true">
      <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d
      </m:reference>
      <m:dateAndTime>2001-11-29T13:20:00.000-05:00
      </m:dateAndTime>
    </m:reservation>
    <n:passenger xmlns:n="…"
          env:role="http://www.w3.org/2003/05/soap-
    envelope/role/next"    env:mustUnderstand="true">
          <n:name>Åke Jógvan Øyvind</n:name>
    </n:passenger>
  </env:Header>
```

SOAP Nodes and Roles

- Originators, recipients, and receivers of SOAP messages are all called SOAP Nodes.
  - Each node is labeled with a URI
- For a particular message, the Node can act in one or more SOAP Roles.
  - Each role is labeled with a URI
  - The following table list predefined roles.
- You can define your own roles
  - “Log message” role
  - “Check authorization” role
- When a node receives a message, it must examine the message for a role definition and process the headers as required.
- The SOAP specification itself does not specify how you assign a role to a node.
  - This depends upon the implementation.

Standard SOAP 1.2 Roles

Short-name	**Name**	**Description**
next	"http://www.w3.org/2003/05/soap-envelope/role/next"	Each SOAP intermediary and the ultimate SOAP receiver MUST act in this role.
none	"http://www.w3.org/2003/05/soap-envelope/role/none"	SOAP nodes MUST NOT act in this role. That is, the header block should not be directly processed. It may carry supplemental information.
ultimateReceiver	"http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver"	The ultimate receiver MUST act in this role. If no role is specified in a header, it is treated as being in this role.

Understanding Headers

- SOAP role definitions may require SOAP nodes to process headers.
- In a distributed processing model, it is possible that certain nodes will not have the required capability to process the header.
- We must therefore identify a header as optional or required.
- We do this with the mustUnderstand attribute.
  - If true, the node must process the header or else stop processing and return a Fault message.
  - If false, the header is optionally processed, depending on the role of the node. This is the default value.
- The SOAP specification requires that a node identify all required headers and determine if they are understood before any processing takes place.

Relaying SOAP Messages

- As we have seen, SOAP headers may or may not be processed by an intermediate node.
  - mustUnderstand and role attributes determine this.
- Processed headers must be removed from the SOAP message before forwarding.
- But there are times when a node role indicates processing, but processing is optional.
  - Role is “next” but mustUnderstand=“false”
- What happens to these headers?
- SOAP 1.2 defines an optional attribute called “relay” to resolve this.
  - Relay is a boolean attribute.

Summary of Relay Forwarding

Role		**Header block**	
Short-name	**Assumed**	**Understood & Processed**	**Forwarded**
next	Yes	Yes	No, unless reinserted
		No	No, unless relay ="true"
user-defined	Yes	Yes	No, unless reinserted
		No	No, unless relay ="true"
	No	n/a	Yes
ultimateReceiver	Yes	Yes	n/a
		No	n/a
none	No	n/a	Yes

SOAP Intermediaries

- Forwarding Intermediaries:
  - Are used to route messages to other SOAP nodes, based on header information.
  - May do additional processing as described in a SOAP header.
- Active Intermediaries do additional processing to a message that is NOT described in any of the message headers.
  - For example, may insert additional headers needed for additional processing, or may encrypt parts of the message for security.

SOAP Forwarding Intermediaries

- As we have seen, a forwarding intermediary must do the following:
  - Process any headers as required by its role and mustUnderstand.
  - Relay any unprocessed headers.
- It is also required by the spec to
  - Remove all processed header blocks.
  - Remove all unprocessed and non-relayable header blocks.
- Forwarding Intermediaries may also insert new headers.
  - This may be a reinsertion of a processed header, for example.
  - Oddly, there seems to be no built-in way to label a header as “persistent”.

SOAP + HTTP

A Quick HTTP Lesson

- HTTP is an ASCII request and response protocol.
- You can easily send HTTP messages to your favorite website and get a response.

- Type this:
  - telnet www.cnn.com 80
- Then type
  - GET / HTTP/1.0
- Hit enter twice.
- You'll get back the HTML for CNN's home page.

Putting SOAP into HTTP

- Assume that I know the port of a particular HTTP server that speaks SOAP.
- Then I can easily construct an HTTP message with a SOAP payload.
- Then write the message to the remote socket.

```
POST /axis/service/echo
   HTTP/1.0
Host: www.myservice.com
Content-Type: text/xml;
   charset="utf-8"
Content-Length: nnn
SOAPAction=""
<SOAP:env>
   …
</SOAP:evn>
```

What Does It Mean?

- The POST line specifies that we will use the POST method and assume HTTP 1.0 (not HTTP 1.1).
  - /axis/services/echo is the relative path part of the URL.
  - Host is in on a separate line.
- Host: specifies the name of the host.
- Content-Type: Type of content we are sending.
  - We must use text/xml for SOAP.
  - In general these are called mime-types.
- Content-Length: number of characters in the HTTP payload.
- SOAPAction: Recall this from our WSDL Binding example.

SOAPAction

- In SOAP 1.0 this is required by all HTTP request messages that transmit SOAP.
- It is optional in SOAP 1.1, deprecated in 1.2.
- It's intended use is to tell the Web Server some specific intended use.
  - The server could use this to short circuit SOAP message processing if the requested service was unavailable.
- SOAPAction="" means that the intended service is identical to the relative path of the POST line.
  - /axis/services/Echo