**SOAP Protocol**

CS 183 – Hypermedia and the Web

SOAP is most typically viewed as a remote procedure call technology. A *remote procedure call* involves a client machine making a request (a procedure call) of a server machine that is located somewhere on the network. Historically, remote procedure call technologies (e.g., ONC RPC, used by the Network File System (NFS)) have worked best on local area networks, and have not behaved well on wide area networks (situations of high latency).

Distinguishing characteristics of SOAP versus other remote procedure call technologies:

- Uses XML for message format, making it easier for humans to understand messages, and to have multiple organizational units contribute to a single message.
- Has significant support for using HTTP (via the POST method) as the dominant transport, and hence leverages the HTTP transport infrastructure (useful for getting through firewalls)
- Has an internet-accessible service discovery mechanism

Primary value of SOAP is as an integration technology for knitting together information technology services within an organization. When IT services support SOAP interfaces, it is easier to write code to interact with them since:

- The message format is open, and can be easily inspected (no obscure binary formats to decipher, though XML can be obfuscated as well)
- Existence of multiple independent implementations of SOAP client APIs, some open source, so there are no expensive developer’s kits to buy, and there are multiple language and platform choices available.

SOAP specifications are divided into two pieces. From the SOAP Version 1.2 Primer (http://www.w3.org/TR/2003/REC-soap12-part0-20030624/):

[SOAP Part1] defines the SOAP envelope, which is a construct that defines an overall framework for representing the contents of a SOAP message, identifying who should deal with all or part of it, and whether handling such parts are optional or mandatory. It also defines a protocol binding framework, which describes how the specification for a binding of SOAP onto another underlying protocol may be written.

[SOAP Part2] defines a data model for SOAP, a particular encoding scheme for data types which may be used for conveying remote procedure calls (RPC), as well as one concrete realization of the underlying protocol binding framework defined in [SOAP Part1]. This binding allows the exchange of SOAP messages either as payload of a HTTP POST request and response, or as a SOAP message in the response to a HTTP GET.
One of the goals of the SOAP protocol is to separate the representation of protocol messages from their specific instantiation in a protocol. If such a separation can be maintained, then it is possible to have the SOAP messages be transported over multiple protocols, potentially with non-RPC semantics. This explains why you see sentences like this in the SOAP specifications:

    SOAP is fundamentally a stateless, one-way message exchange paradigm, but applications can create more complex interaction patterns (e.g., request/response, request/multiple responses, etc.) by combining such one-way exchanges with features provided by an underlying protocol and/or application-specific information.

However, the dominant use of SOAP today is for remote procedure call processing.

Basic SOAP transport model:

SOAP sender ➔ SOAP node 1 ➔ SOAP node 2 ➔ … ➔ SOAP node n ➔ SOAP receiver

The SOAP nodes in the middle are known together as SOAP intermediaries. There are frequently no intermediaries.

A SOAP protocol message contains an *envelope* which contains a *header* and a *body*. All are expressed in XML.
An example (from the SOAP primer):

```xml
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
        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="http://mycompany.example.com/employees"
        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>
  <env:Body>
    <p:itinerary xmlns:p="http://travelcompany.example.org/reservation/travel">
      <p:departure>
        <p:departing>New York</p:departing>
        <p:arriving>Los Angeles</p:arriving>
        <p:departureDate>2001-12-14</p:departureDate>
        <p:departureTime>late afternoon</p:departureTime>
        <p:seatPreference>aisle</p:seatPreference>
      </p:departure>
      <p:return>
        <p:departing>Los Angeles</p:departing>
        <p:arriving>New York</p:arriving>
        <p:departureDate>2001-12-20</p:departureDate>
        <p:departureTime>mid-morning</p:departureTime>
        <p:seatPreference/>
      </p:return>
    </p:itinerary>
    <q:lodging xmlns:q="http://travelcompany.example.org/reservation/hotels">
      <q:preference>none</q:preference>
    </q:lodging>
  </env:Body>
</env:Envelope>
```

General structure:
- Envelope
  - Header
    - reservation, passenger
  - Body
    - itinerary, lodging
SOAP header is optional, and contains information that might be subject to processing by intermediate SOAP nodes. When designing applications that use SOAP, information that might be useful to intermediaries should be placed in the header. In this example, the passenger information, and type of request are placed in the header, since these are pieces of information that might be useful to an intermediary, such as a firewall (should I let this operation through?) or perhaps a fraud detection server (perhaps this passenger has had their credit cards stolen recently, and hence the request perhaps shouldn’t be processed by a computer without a human double-check).

The SOAP body is mandatory, and contains the payload of the message, which is intended for the final SOAP receiver, and is not intended for processing by intermediate nodes.

A SOAP message sent in response to the previous example message:
There is then a follow-in clarification message:
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <m:reservation
      xmlns:m="http://travelcompany.example.org/reservation"
      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>2001-11-29T13:36:50.000-05:00</m:dateAndTime>
    </m:reservation>
    <n:passenger
      xmlns:n="http://mycompany.example.com/employees"
      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>
  <env:Body>
    <p:itinerary
      xmlns:p="http://travelcompany.example.org/reservation/travel">
      <p:departure>
        <p:departing>LGA</p:departing>
      </p:departure>
      <p:return>
        <p:arriving>EWR</p:arriving>
      </p:return>
    </p:itinerary>
  </env:Body>
</env:Envelope>