SOAP – The Information Exchange Protocol for Web Services

Eetu Mäkelä

Semantic Computing Research Group
http://www.cs.helsinki.fi/group/seco/

HELSINKI UNIVERSITY OF TECHNOLOGY
HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY
UNIVERSITY OF HELSINKI

Be wary traveller, for herein we discover:

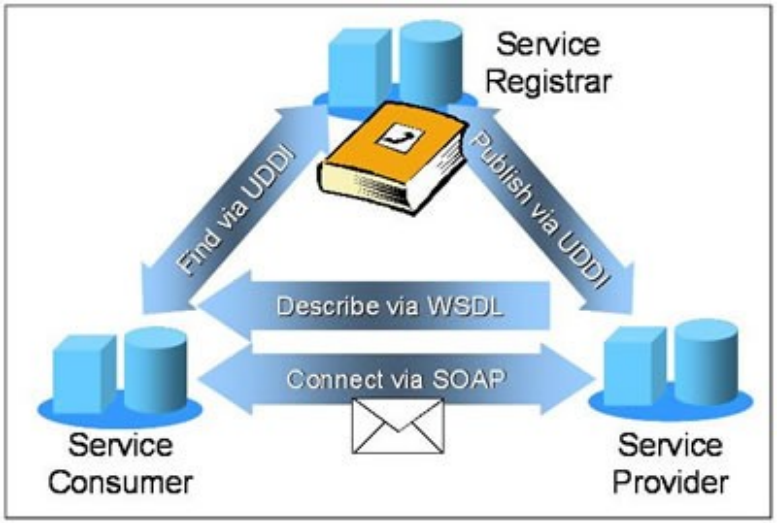
- What is required of a good enterprise application information exchange protocol
- How SOAP fails most of these goals
- And why it has still become the de facto industry standard

Orientation

- Communication between services in general:
  - Find a service via a registry or manually
  - Get a description of that service's parameters and interface
  - Build a request to the service according to the information exchange protocol and service parameters
  - Invoke the service via the information exchange protocol
- Invoking a Web Service:
  - Locate the service manually via UDDI or other registry
  - Get a WSDL description from UDDI or via other means
  - Build a SOAP request according to a WSDL description
  - Call service via a SOAP binding

Orientation

- In other words, information exchange protocols [such as SOAP] describes how applications [Web Services] exchange messages as well as the general formatting of those messages (but not the content, that's for the interface language [such as WSDL] to do)



The History of SOAP

- The Simple Object Access Protocol (SOAP)
  - Initiated by W3C in 1999.
  - SOAP 1.0 was entirely based on HTTP
  - SOAP 1.1 (May 2000), was more generic since it included other transport protocols.
  - The first draft of SOAP 1.2 was presented in 2001 and promoted to a recommendation in 2003. A second edition recommendation was published in April 2007.

Requirements of a Good General Information Exchange Protocol

- Programming/application platform independent
- Able to robustly carry any sort of information
- Routable and transmittable through various network infrastructures and platforms
- Applicable to different messaging patterns
- Secure
- Reliable (transactioned)
- Extensible

Requirements of a Good General Information Exchange Protocol

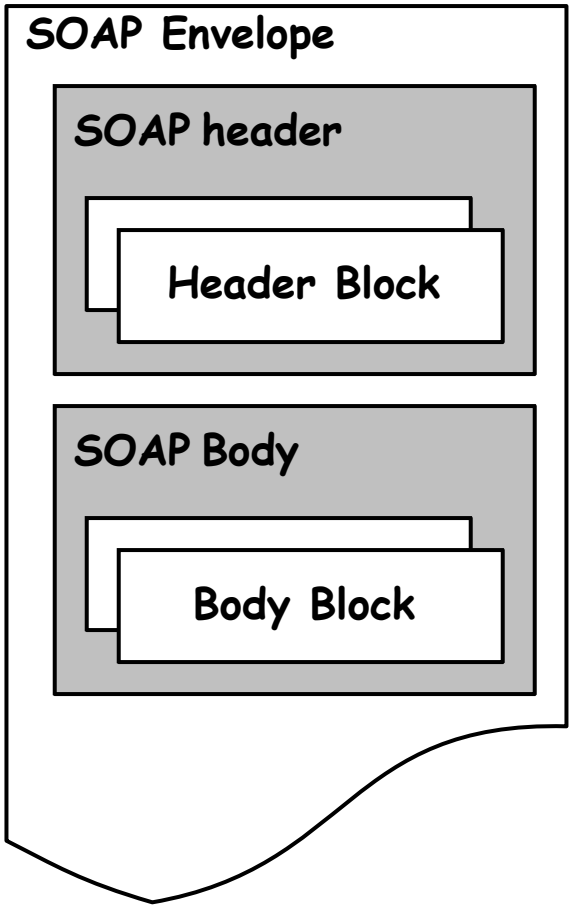
- Programming/application platform independent
- Able to robustly carry any sort of information
- Routable and transmittable through various network infrastructures and platforms
- Applicable to different messaging patterns
- Secure
- Reliable (transactioned)
- Extensible

Platform Independence from the viewpoint of the IE protocol

- Means just that both sides must be able to encode and decode the messages, and understand them
- In practice: a mapping from the platform specific information storage format to a mutually understandable information format and back
- In SOAP, this is accomplished by the use of XML in serializing the information, with defined mappings for all the most widely used basic datatypes and structures

A SOAP message

- Application data is enclosed in a SOAP envelope
- The body contains the call parameters
- The header contains relevant additional information
- The information is serialized in XML



An example SOAP message

```
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    <SOAP-ENV:Header>
      <t:Transaction xmlns:t="some-URI" SOAP_ENV:role=”ultimateReceiver”
     SOAP-ENV:mustUnderstand="1">5</t:Transaction>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
      <m:GetLastTradePrice xmlns:m="Some-URI">
  <m:stockIdentifier>
          <m:symbol>DEF</m:symbol>
      <m:stockExchange>New York</m:stockExchange>
  </m:stockIdentifier>
      </m:GetLastTradePrice>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Requirements of a Good General Information Exchange Protocol

- Programming/application platform independent √
- Able to robustly carry any sort of information
- Routable and transmittable through various network infrastructures and platforms
- Applicable to different messaging patterns
- Secure
- Reliable (transactioned)
- Extensible

The Problem of Binary Information

- Okay, so SOAP uses XML to represent data
- This approach reflects an implicit assumption that what is being exchanged is similar to input and output parameters when calling a procedure in a program
- What about messages with complex binary data that does not easily translate into XML (and there is no reason to do so)
  - Images
  - Word processing documents
  - Sound
  - Proprietary formats
  - Binary data in general

Solution: SOAP with attachments

- Idea: Encode binary data outside of the XML element somehow
- Unfortunately, it's not in the standard:
  - SOAP 1.2 defines an abstract model for attachments
  - SOAP messages with attachments note from 2002 proposes using MIME attachments, is implemented for example in the Apache SOAP 2.2 toolkit, ebXML specification
  - WS-Attachments standard of the WS-I organization

Requirements of a Good General Information Exchange Protocol

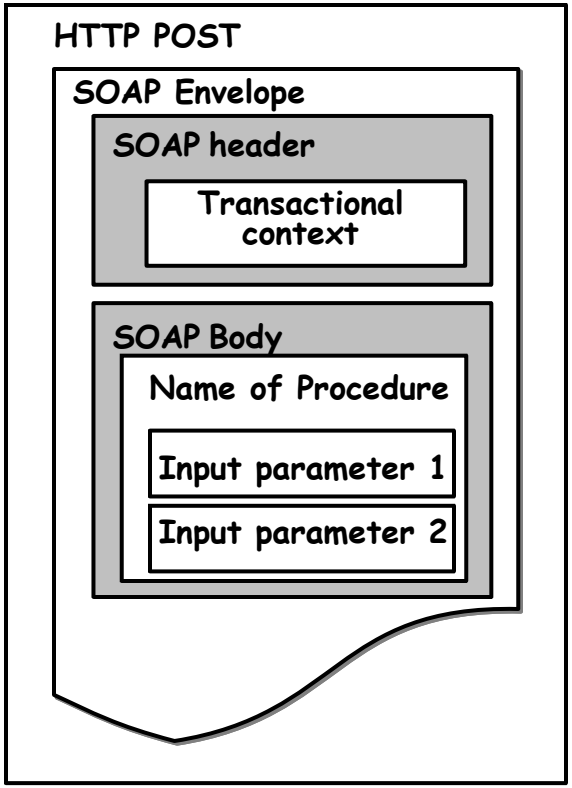
- Programming/application platform independent √
- Able to robustly carry any sort of information † (√ if you and your communication partner bet on the same horse)
- Routable and transmittable through various network infrastructures and platforms
- Applicable to different messaging patterns
- Secure
- Reliable (transactioned)
- Extensible

Routing and message transport on the Internet

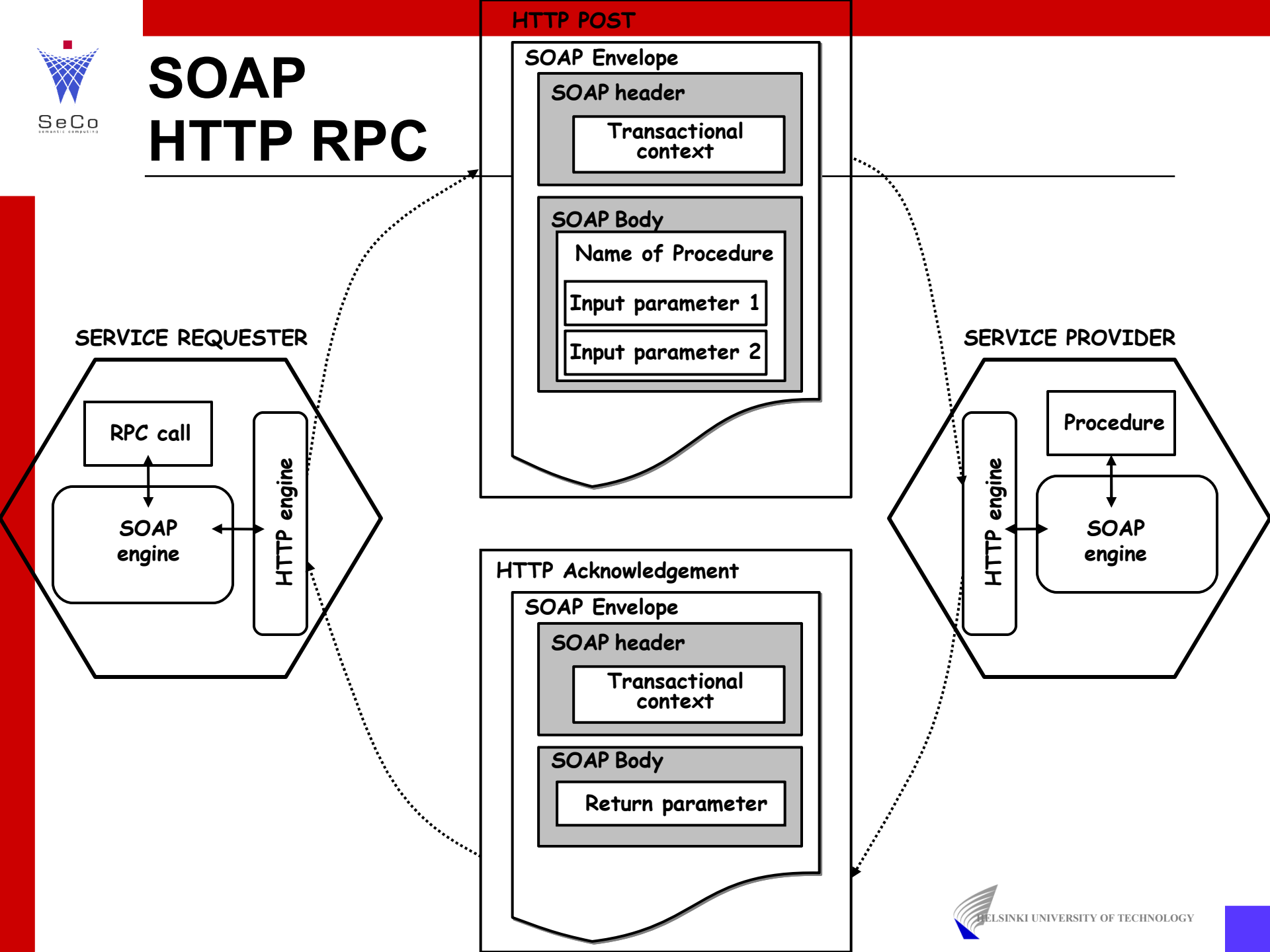
- The architecture of the Internet is very heterogeneous, with different links between hosts supporting different transport protocols, particularly with firewalls blocking different transport protocols in different places
- The two protocols most likely to penetrate firewalls are HTTP (web surfing) and SMTP (e-mail)
- SOAP has a functional HTTP binding
- SOAP theoretically has an e-mail binding, as well as a possibility to create additional bindings. In practice, there are many problems

SOAP and HTTP

- A binding of SOAP to a transport protocol describes how a SOAP message is to be sent using that transport protocol
- The HTTP binding of SOAP can use HTTP GET or POST, but with GET, the request is actually not a SOAP message!



SOAP HTTP RPC



In text (a request)

POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
     <m:GetLastTradePrice xmlns:m="Some-URI">
     <symbol>DIS</symbol>
     </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

In text (the response)

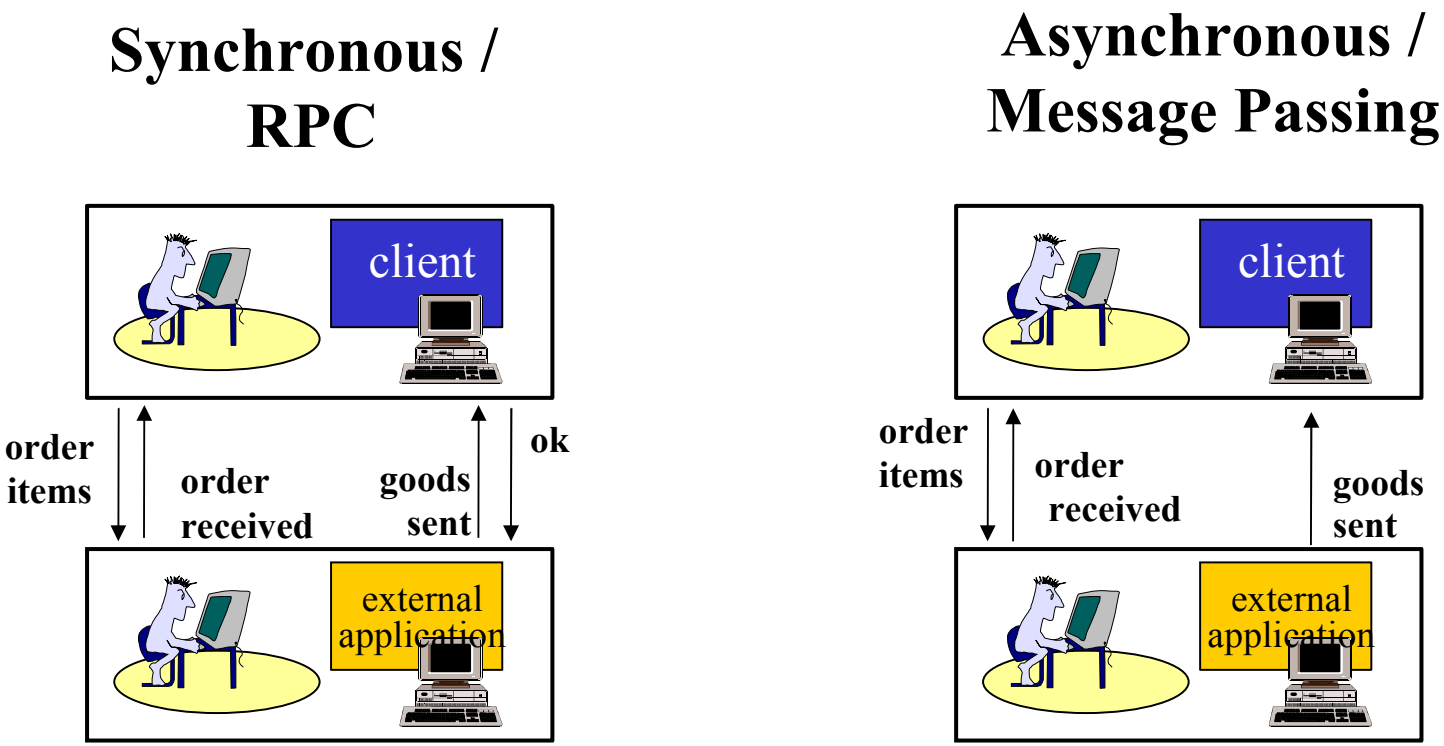
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  <SOAP-ENV:Body>
     <m:GetLastTradePriceResponse xmlns:m="Some-URI">
     <Price>34.5</Price>
     </m:GetLastTradePriceResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Requirements of a Good General Information Exchange Protocol

- Programming/application platform independent √
- Able to robustly carry any sort of information † (√ if you and your communication partner bet on the same horse)
- Routable and transmittable through various network infrastructures and platforms √/†
- Applicable to different messaging patterns
- Secure
- Reliable (transactioned)
- Extensible

Different messaging patterns between applications



- Asynchronous message passing is a more general pattern, more robust

Messaging patterns in SOAP

- SOAP was originally conceived as the minimal possible infrastructure for performing RPC through the Internet
  - Use of XML and HTTP
  - Very simple message structure
  - Designed to be layered atop existing middleware platforms
- HTTP, and as a consequence SOAP, is inherently request-response
- Also sheds light on why the binding for the inherently one-way E-mail protocol is so problematic

The Light at the End of the Tunnel

- WS-ReliableMessaging specification from Microsoft and IBM
- ebXML Messaging specification
- Whole new reliable messaging architectures built on top of SOAP RPC
- Only, of course, they're not standardized!

Requirements of a Good General Information Exchange Protocol

- Programming/application platform independent √
- Able to robustly carry any sort of information † (√ if betting on the same horse)
- Routable and transmittable through various network infrastructures and platforms √/†
- Applicable to different messaging patterns † (√ if betting on the same horse)
- Secure
- Reliable (transactioned)
- Extensible

Web Services Security

- If we are to have secure Web Services, taking security into account must start at the very bottom
  - Message encryption
  - Authentication and signature handling
- SOAP doesn't handle any of this
- ebXML defines its own security extensions
- The WS-I organization has defined extensive WS-Security standards

Requirements of a Good General Information Exchange Protocol

- Programming/application platform independent √
- Able to robustly carry any sort of information † (√ if betting on the same horse)
- Routable and transmittable through various network infrastructures and platforms √/†
- Applicable to different messaging patterns † (√ if betting on the same horse)
- Secure † (√ if betting on the same horse)
- Reliable (transactioned)
- Extensible

Web Services Reliability

- If we are to have reliable Web Services, taking reliability into account must start at the very bottom
  - Reliable message passing
  - Transactions, fault recovery and fallback
- SOAP defines faults, but not how to recover from them
- ebXML defines its own reliable messaging extensions
- So do WS-Transaction and WS-ReliableMessaging by Microsoft and IBM
- Starting to see a pattern here?

Requirements of a Good General Information Exchange Protocol

- Programming/application platform independent √
- Able to robustly carry any sort of information † (√ if same horse)
- Routable and transmittable through various network infrastructures and platforms √/†
- Applicable to different messaging patterns † (√ if same horse)
- Secure † (√ if same horse)
- Reliable (transactioned) † (√ if same horse)
- Extensible

Extensibility and SOAP

- By now, you probably have some clue as to why SOAP has become the de facto standard despite its many shortcomings:
  - Keep It Simple Stupid: While the current SOAP specification has grown complex, people are only using the simple HTTP-RPC core of it, which accounts for the most common use case anyway. To top, that core is easy to understand and implement as well.
  - Its damn extensible. While it itself doesn't provide for much, it does provide a rock solid base for extensions
  - And that's good programming practice!
  - ... if only we'd get those damn extensions standardized

Requirements of a Good Enterprise Information Exchange Protocol

- Programming/application platform independent √
- Able to robustly carry any sort of information † (√ if same horse)
- Routable and transmittable through various network infrastructures and platforms √/†
- Applicable to different messaging patterns † (√ if same horse)
- Secure † (√ if same horse)
- Reliable (transactioned) † (√ if same horse)
- Extensible √√√

To Recap: So What Does SOAP Actually Define

- An XML-based message format for transmitting information
- A description of how the message should be transmitted using HTTP
- A set of conventions on how to turn an RPC call into a SOAP message and back as well

“And How do I Actually Use It?”, You Ask

- Fortunately, it's dead easy!
  - There are automated tools to turn program procedures/functions into SOAP accepting Web Services for practically all the programming platforms
  - Calling a Web Service is equally easy
  - The only thing you have to worry about is that your objects are serializable into XML
  - In the Java implementations, this usually means that the objects you use are either JavaBeans or that you provide a custom serialization

Thank You

- Any questions or comments?
- Next: RPC interface description using WSDL