

Web Services – Part 2

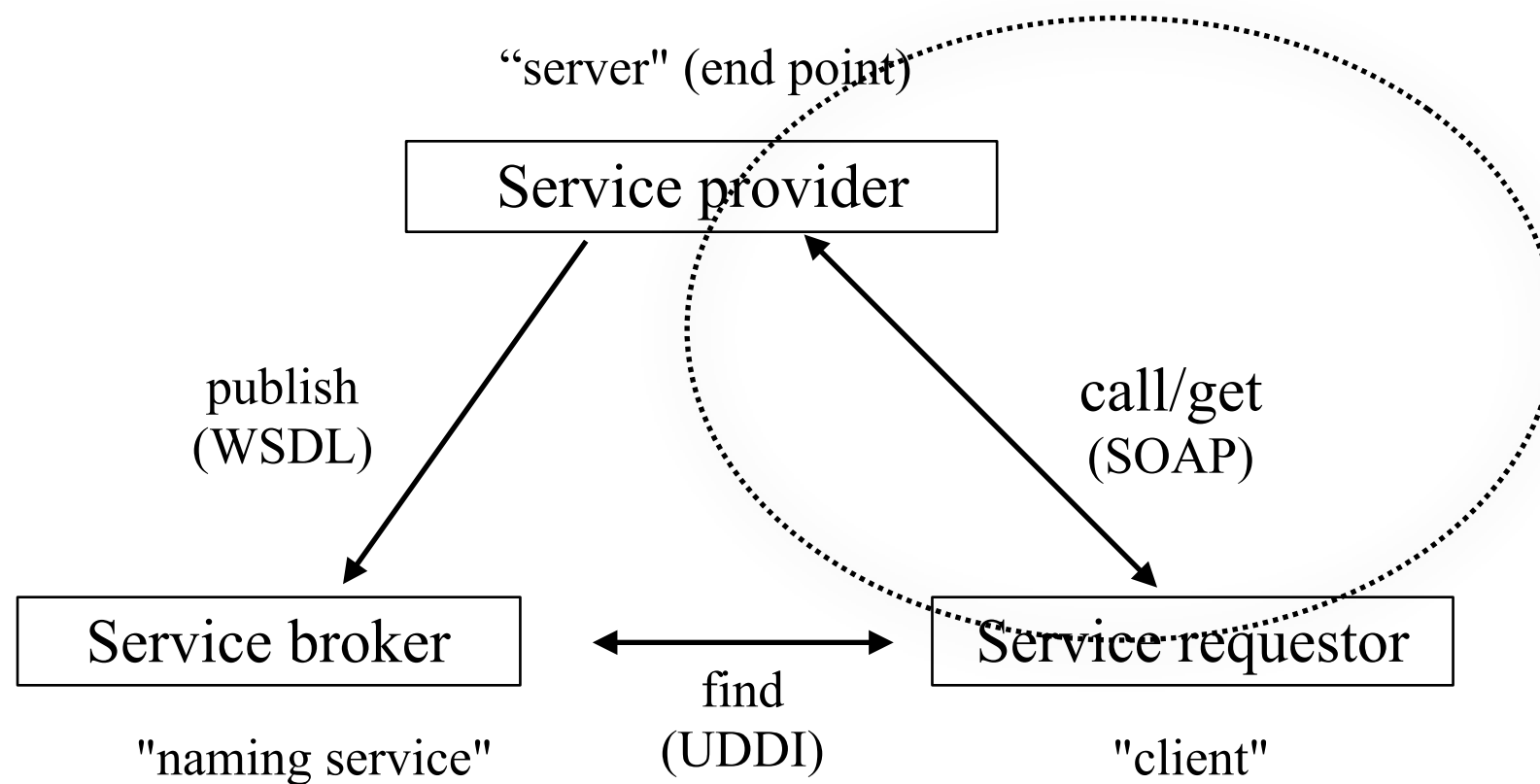
Syed Gillani

Laboratoire Hubert Curien St-Etienne, France

[Simple Object Access
Protocol (SOAP)]

[Recall from Earlier]

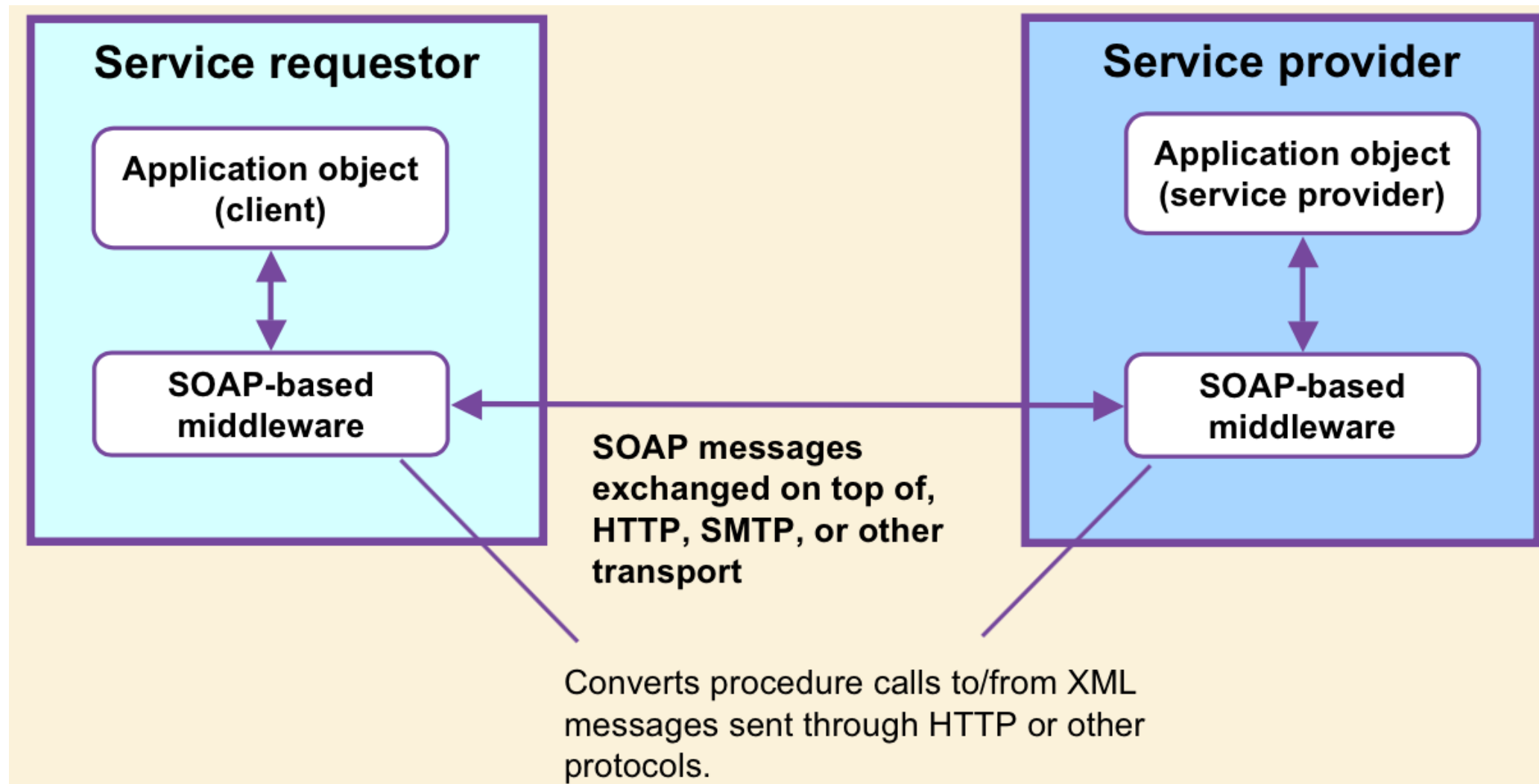
- ▶ Server (endpoint) can describe what it can do (through WSDL)
- ▶ But how a server execute the actions a client asked for: RPC



[Simple Object Access Protocol (SOAP)]

- ▶ What is SOAP?
 - ✓ A standard (W3C) messaging protocol for inter application communication
- ▶ SOAP codifies the use of XML as an encoding scheme for request and response parameters using HTTP as a means of transport (but can run over others as well e.g. SMTP)
- ▶ More powerful than XML-RPC
 - ✓ User-defined data types
 - ✓ Ability to specify specific recipient
 - ✓ Message specific processing control
- ▶ Its a standard, thus:
 - ✓ Every one uses it
 - ✓ Inter communication between proprietary systems running on heterogeneous infrastructures

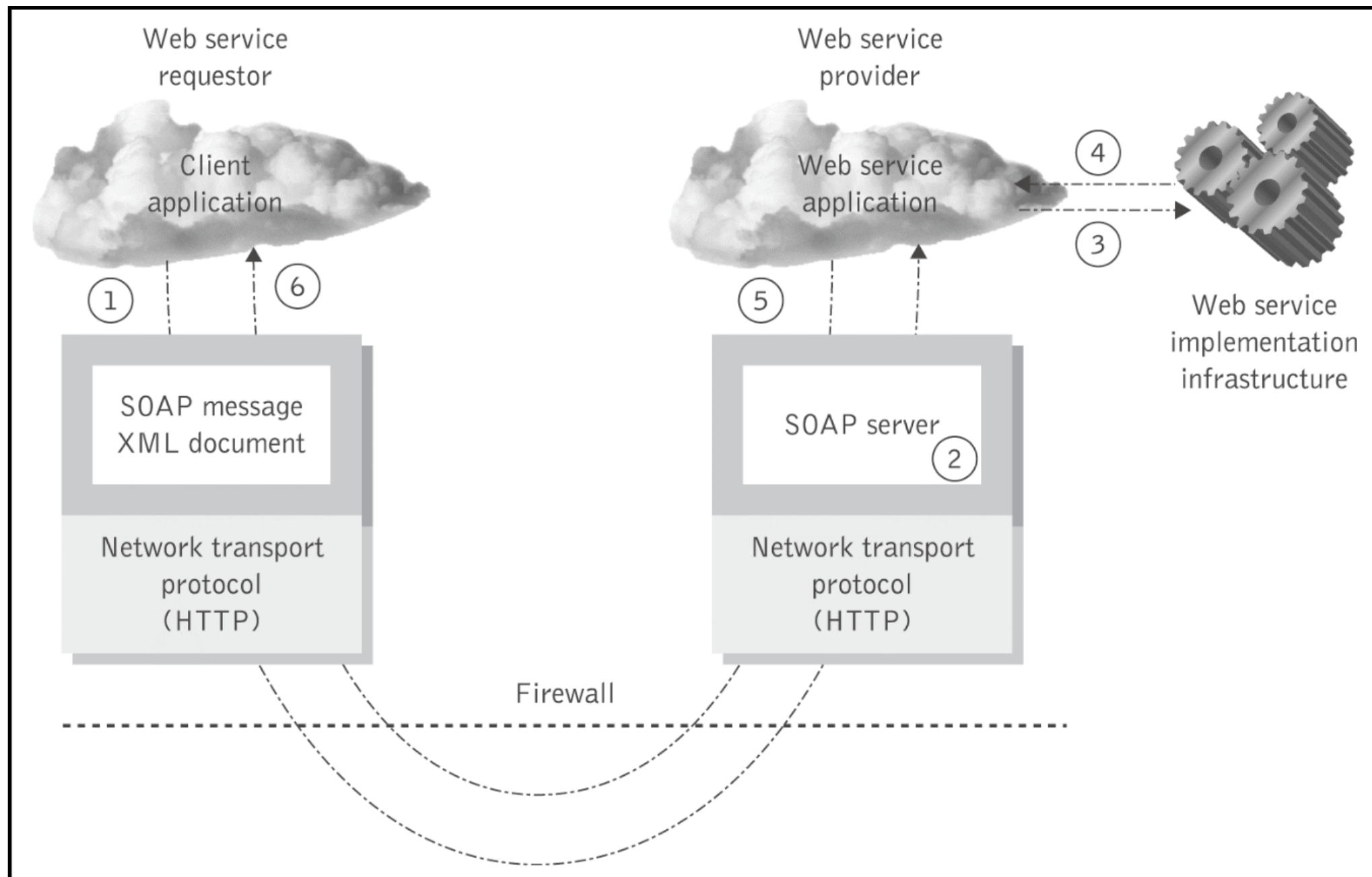
[Simple Object Access Protocol (SOAP)]



[What is SOAP (continued)]

- ▶ SOAP covers the following four main areas:
 - ✓ A message format for one-way communication describing how a message can be packed into an XML document.
 - ✓ A description of how a SOAP message should be transported using HTTP (for Web-based interaction) or SMTP (for e-mail-based interaction).
 - ✓ A set of rules that must be followed when processing a SOAP message and a simple classification of the entities involved in processing a SOAP message. – A set of conventions on how to turn an RPC call into a SOAP message and back.

[Distributed Messaging using SOAP]



[SOAP Messages]

SOAP messages are seen as envelopes where the application encloses the data to be sent

<Envelope>

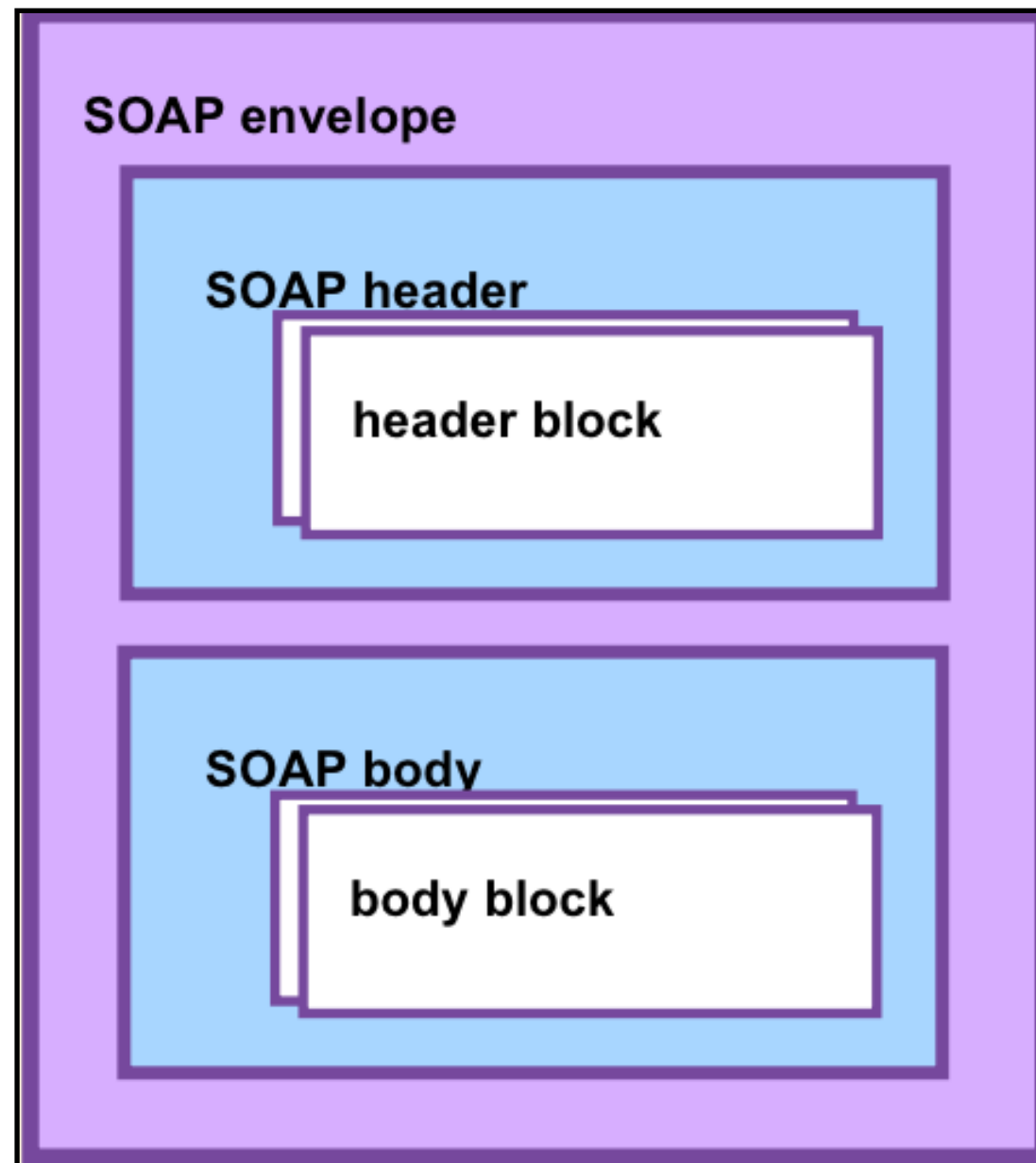
<Header>

</Header>

<Body>

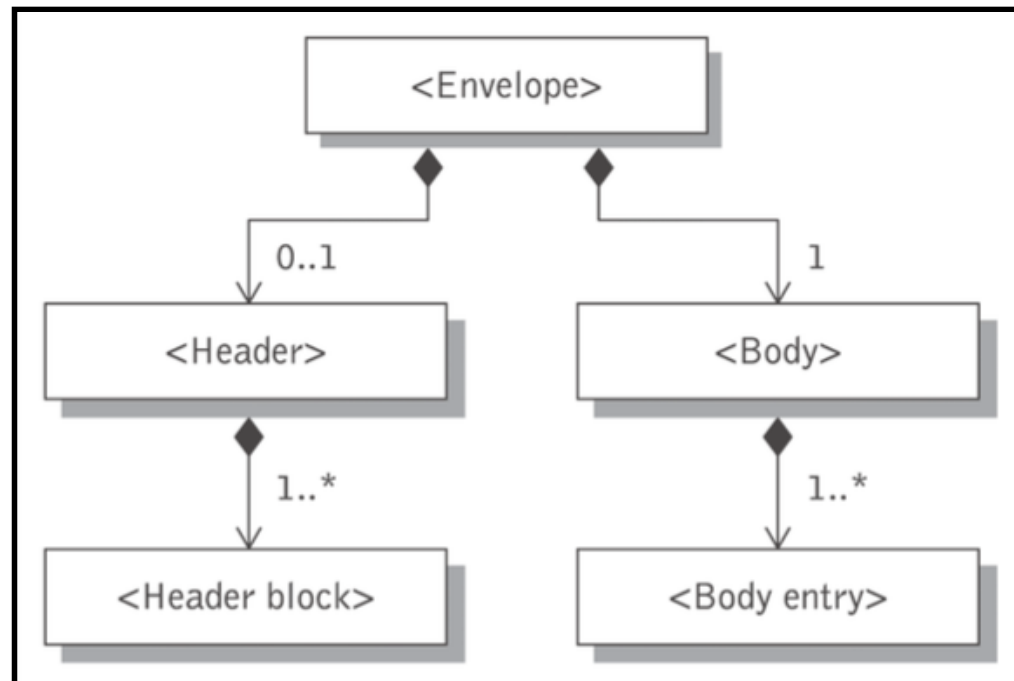
</Body>

</Envelope>



*Headers are optional

[SOAP Messages]



- ▶ **Envelope**: a root element
- ▶ **Header**: contains blocks of information how the message is to be processed (message is for application or directly for the SOAP engine)
- ▶ **Body**: main end-to-end information is conveyed, the method class, parameters, etc.

[SOAP Messages: Example]

```
<?xml version='1.0' ?>
```

```
<env:Envelope xmlns:env="http://www.w3.org/2002/06/soap-envelope" >
```

```
<env:Header>
```

```
<t:transactionID
```

```
  xmlns:t="http://intermediary.example.com/procurement"
```

```
  env:role="http://www.w3.org/2002/06/soap-envelope/role/next"
```

```
  env:mustUnderstand="true" >
```

```
  57539
```

```
</t:transactionID>
```

```
</env:Header>
```

```
<env:Body>
```

```
<m:orderGoods
```

```
  env:encodingStyle="http://www.w3.org/2002/06/soap-encoding"
```

```
  xmlns:m="http://example.com/procurement">
```

```
<m:productItem>
```

```
  <name>ACME Softener</name>
```

```
</m:productItem>
```

```
<m:quantity>
```

```
  35
```

```
</m:quantity>
```

```
</m:orderGoods>
```

```
</env:Body>
```

```
</env:Envelope>
```

Envelope

Header

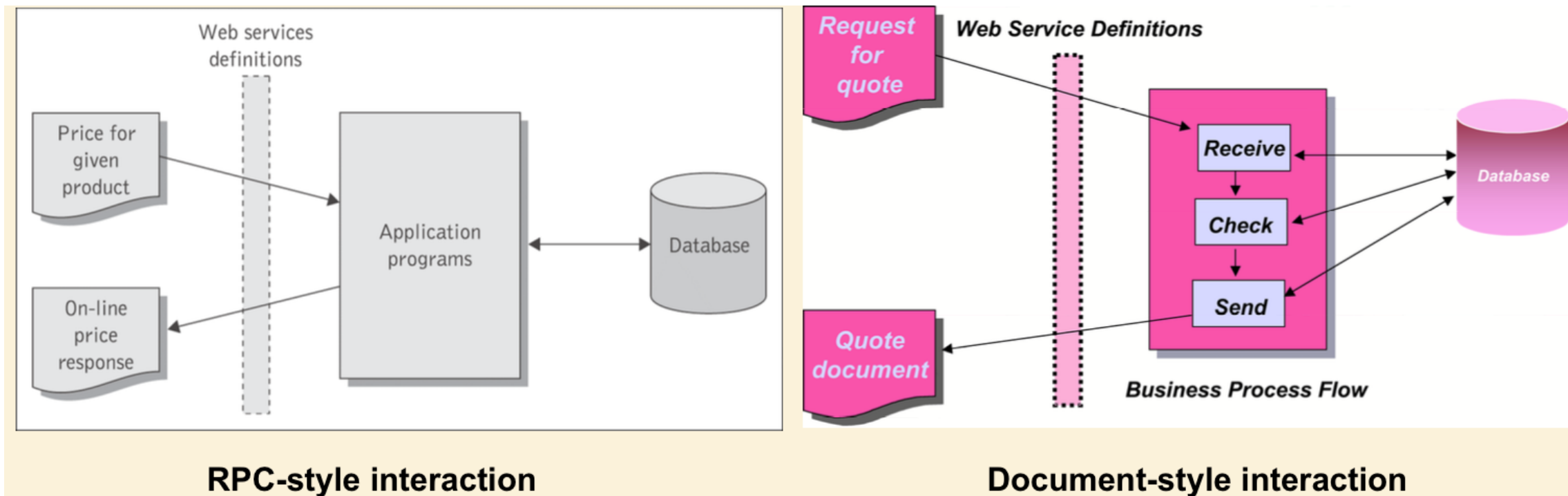
Blocks

Body

[The SOAP Communication Model]

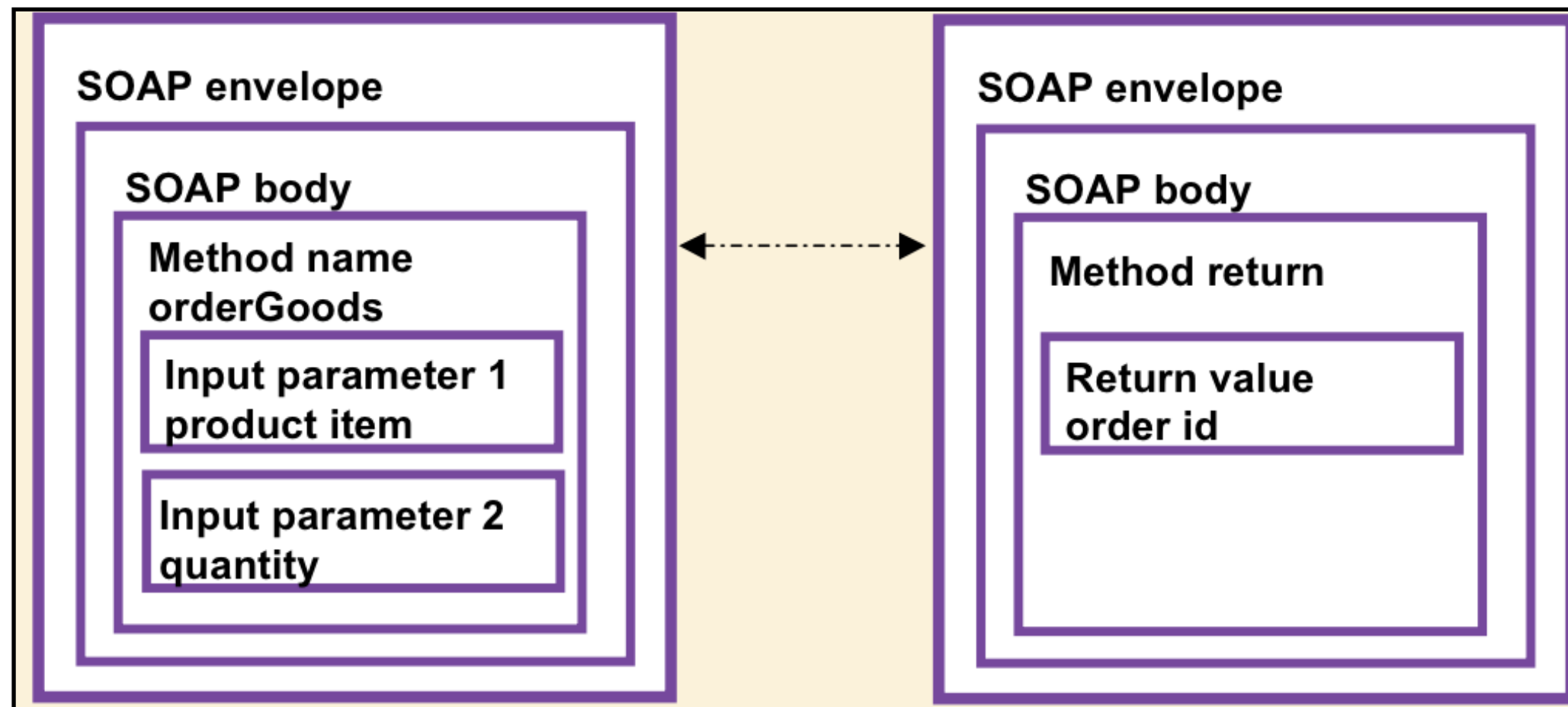
► SOAP supports two possible communication styles:

- ✓ Remote Procedure Call (RPC),
- ✓ Document (or simple messages)



[RPC-style SOAP Services]

- ▶ RPC-style web service appears as remote object to client
- ▶ RPC-style web services centre's around a service-specific interface
- ▶ Client express their requests as a method call with a set of arguments
- ▶ service returns a response containing a return value



[RPC-style SOAP Services]

Example of RPC-style SOAP body

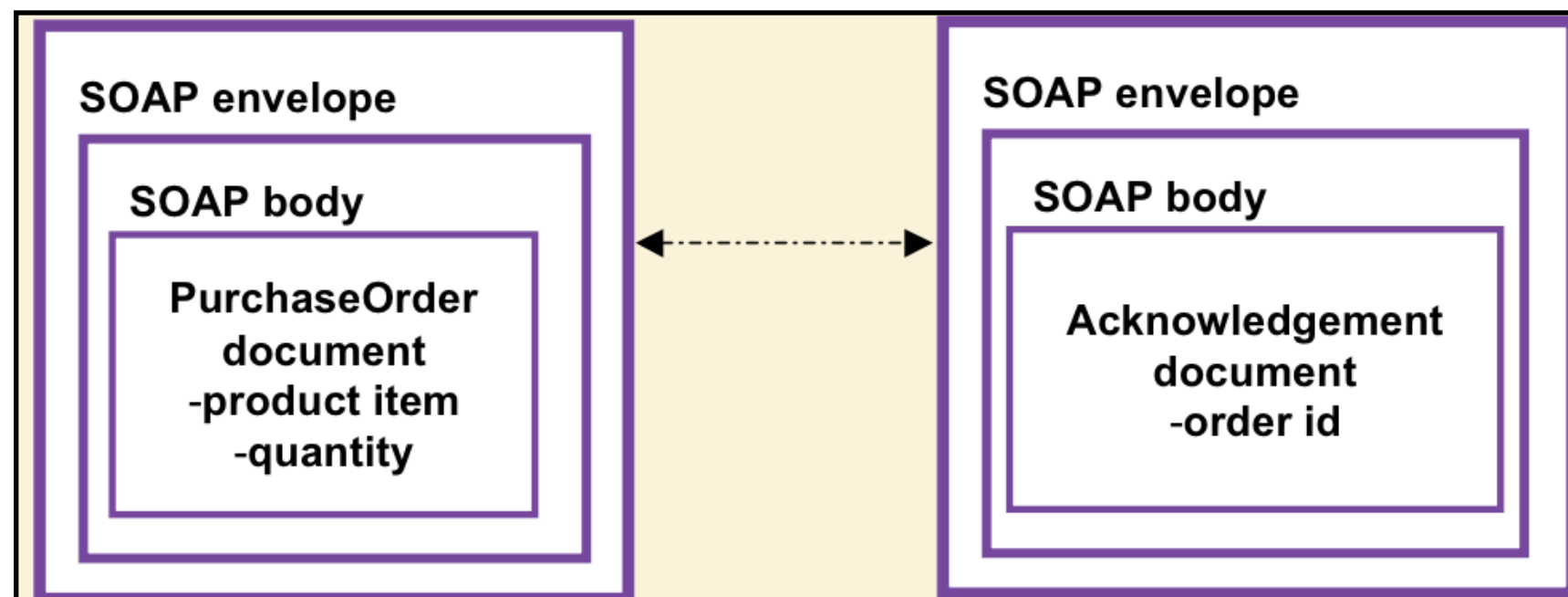
```
<env:Envelope
  xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
  xmlns:m="http://www.plastics_supply.com/product-prices">
  <env:Header>
    <tx:Transaction-id
      xmlns:t="http://www.transaction.com/transactions"
      env:mustUnderstand='1'>
      512
    </tx:Transaction-id>
  </env:Header>
  <env:Body>
    <m:GetProductPrice>
      <product-id> 450R60P </product-id >
    </m:GetProductPrice >
  </env:Body>
</env:Envelope>
```

Example of RPC-style SOAP response message

```
<env:Envelope
  xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
  xmlns:m="http://www.plastics_supply.com/product-prices">
  <env:Header>
    <!--! - Optional context information -->
  </env:Header>
  <env:Body>
    <m:GetProductPriceResponse>
      <product-price> 134.32 </product-price>
    </m:GetProductPriceResponse>
  </env:Body>
</env:Envelope>
```

[Document(Message)-style SOAP Services]

- ▶ Document-style of messaging, the SOAP <Body> contains an XML document fragment. The <Body> element reflects no explicit XML structure
- ▶ The SOAP run-time environment accepts the SOAP <Body> element as it stands and hands it over to the application it is destined for unchanged. There may or may not be a response associated with this message



[Document(Message)-style SOAP Body: Example]

```
<env:Envelope
  xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">

  <env:Header>
    <tx:Transaction-id
      xmlns:t="http://www.transaction.com/transactions"
      env:mustUnderstand='1'>
      512
    </env:Header>
  <env:Body>
    <po:PurchaseOrder oderDate="2004-12-02"
      xmlns:m="http://www.plastics_supply.com/POs">
      <po:from>
        <po:accountName> RightPlastics </po:accountName>
        <po:accountNumber> PSC-0343-02 </po:accountNumber>
      </po:from>
      <po:to>
        <po:supplierName> Plastic Supplies Inc. </po:supplierName>
        <po:supplierAddress> Yara Valley Melbourne </po:supplierAddress>
      </po:to>
      <po:product>
        <po:product-name> injection molder </po:product-name>
        <po:product-model> G-100T </po:product-model>
        <po:quantity> 2 </po:quantity>
      </po:product>
    </ po:PurchaseOrder >
  </env:Body>
</env:Envelope>
```

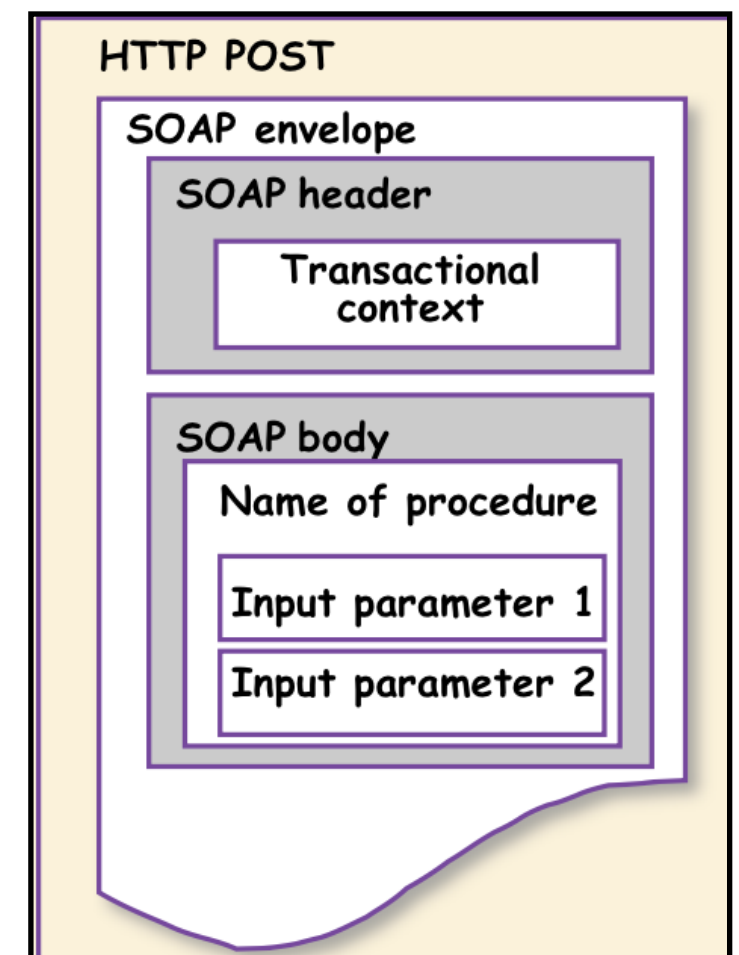
[SOAP Fault Element]

- ▶ SOAP provides a model for handling faults (e.g. entries do not exist in the database)
- ▶ The SOAP body contains the fault message

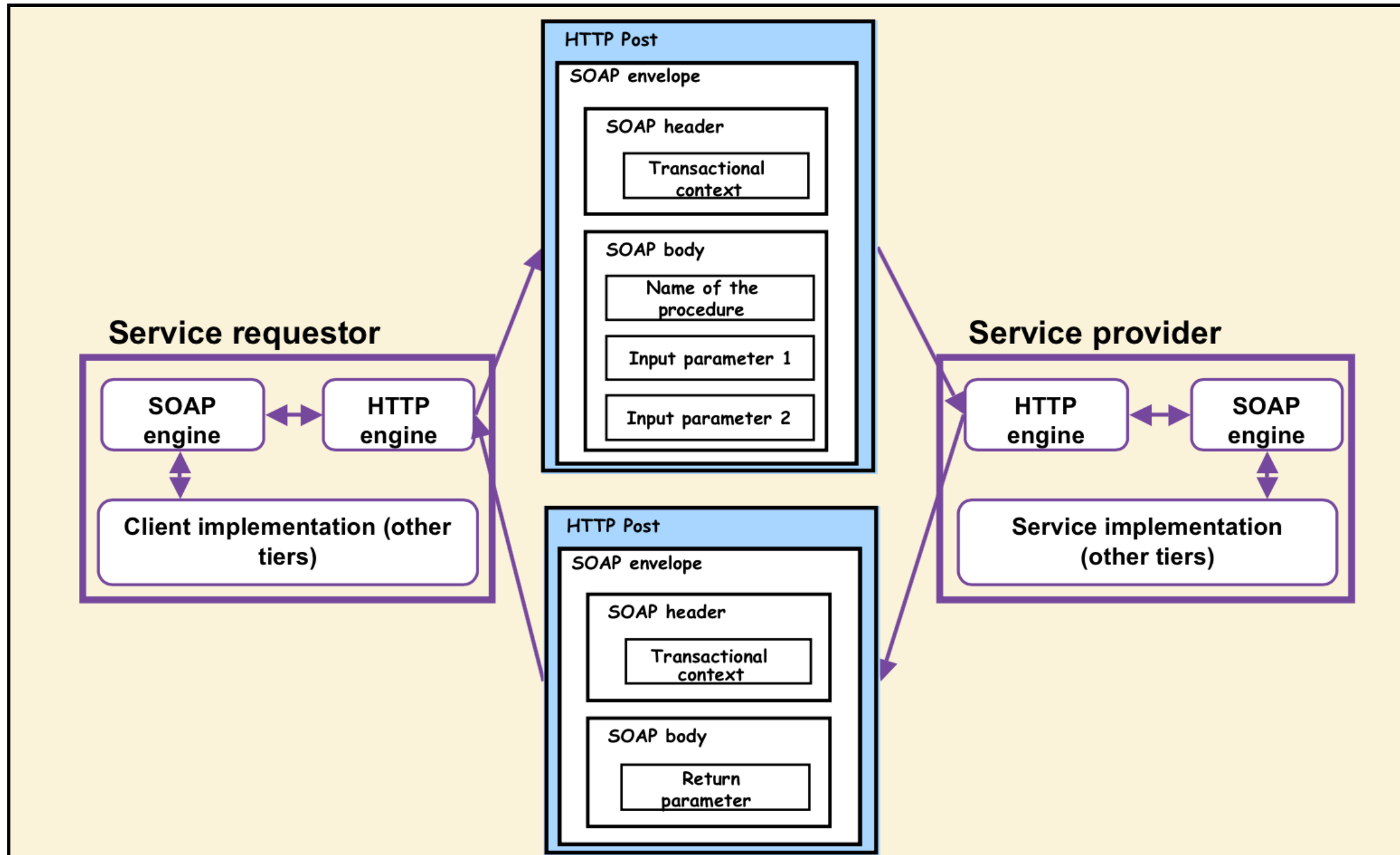
```
<env:Envelope
  xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
  xmlns:m="http://www.plastics_supply.com/product-prices">
  <env:Header>
    <tx:Transaction-id
      xmlns:t="http://www.transaction.com/transactions"
      env:mustUnderstand='1'>
      512
    </tx:Transaction-id>
  </env:Header>
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Sender</env:Value>
        <env:Subcode>
          <env:Value> m:InvalidPurchaseOrder </env:Value>
        </env:Subcode>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-UK"> Specified product did not exist </env:Text>
      </env:Reason>
      <env:Detail>
        <err:myFaultDetails
          xmlns:err="http://www.plastics_supply.com/faults">
          <err:message> Product number contains invalid characters </err:message>
          <err:errorCode> 129 </err:errorCode>
        </err:myFaultDetails>
      </env:Detail>
    </env:Fault>
  </env:Body>
</env:Envelope>
```


[SOAP Over HTTP]

- ▶ Typical binding for SOAP is HTTP
- ▶ A binding of SOAP to a transport protocol is a description of how a SOAP message is to be sent
- ▶ SOAP can use GET and POST of HTTP
- ▶ With GET the request is not a SOAP message but the response is
- ▶ With POST both request and response is SOAP message (use of POST is recommended over HTTP)



[RPC call using SOAP over HTTP]



[Advantages / Disadvantages of SOAP]

► Advantages:

- ✓ Much more flexible and powerful than XML-RPC
- ✓ Portable
- ✓ Firewall Friendly
- ✓ Use of open standards
- ✓ Interoperability
- ✓ Universal acceptance (W3C standard)

► Disadvantages

- ✓ Too much reliance on HTTP
- ✓ Stateless
- ✓ Serialisation (marshalling) by value not reference

[Other RPC Protocols]

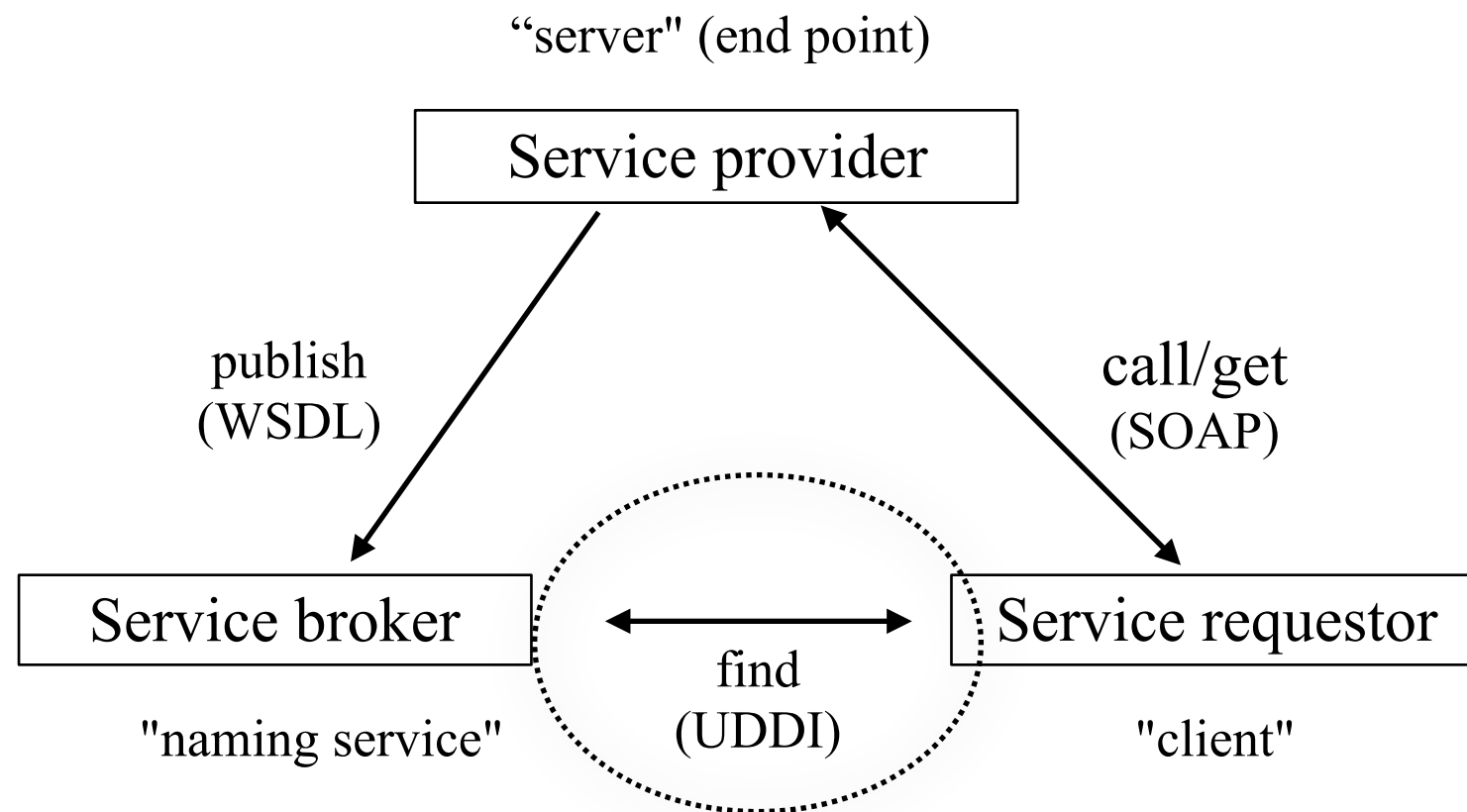
- ▶ Not discussed

- ✓ CORBA, COM (usually for distribution of data)

- ✓ Protocol Buffers, Thrift (by Google and Facebook, and lightning fasts, use JSON)

They are evolved from SOAP, so study it
by yourself

[Last Point]



[UDDI]

[UDDI]

- ▶ Basic idea of UDDI:
 - ✓ UDDI was conceived as a universal business registry similar to search engines (Google et. al.), where services can be located based on different criteria
- ▶ Servers that provide public UDDI registry and lookup service are called nodes
- ▶ UDDI is a technical specification for building a distributed directory of businesses and web services
 - ✓ Uses XML
 - ✓ Includes APIs for searching existing data and creating new

The vision was that service consumers would be linked to service providers through a public brokerage system

[UDDI]

- ▶ UDDI business registration provides 3 distinct sets of information

White Pages: General information about a specific company: address phone number etc.

Yellow Pages: Classification data for either the company or the service offered. This data may include industry, product, or geographic codes based on standard taxonomies

Green Pages: Technical information about a web service, an address for invoking the web service.

[The reality of UDDI]

- ▶ UDDI did not gain widespread use as yet even though it had the backing of large companies like IBM and Microsoft
- ▶ UDDI is mostly used in limited environments (inside companies). For that purpose, UDDI is too complicated and most of the data provided by UDDI is not needed
- ▶ Microsoft, IBM and SAP shut down their public UDDI nodes (servers) in 2006

[F i n]