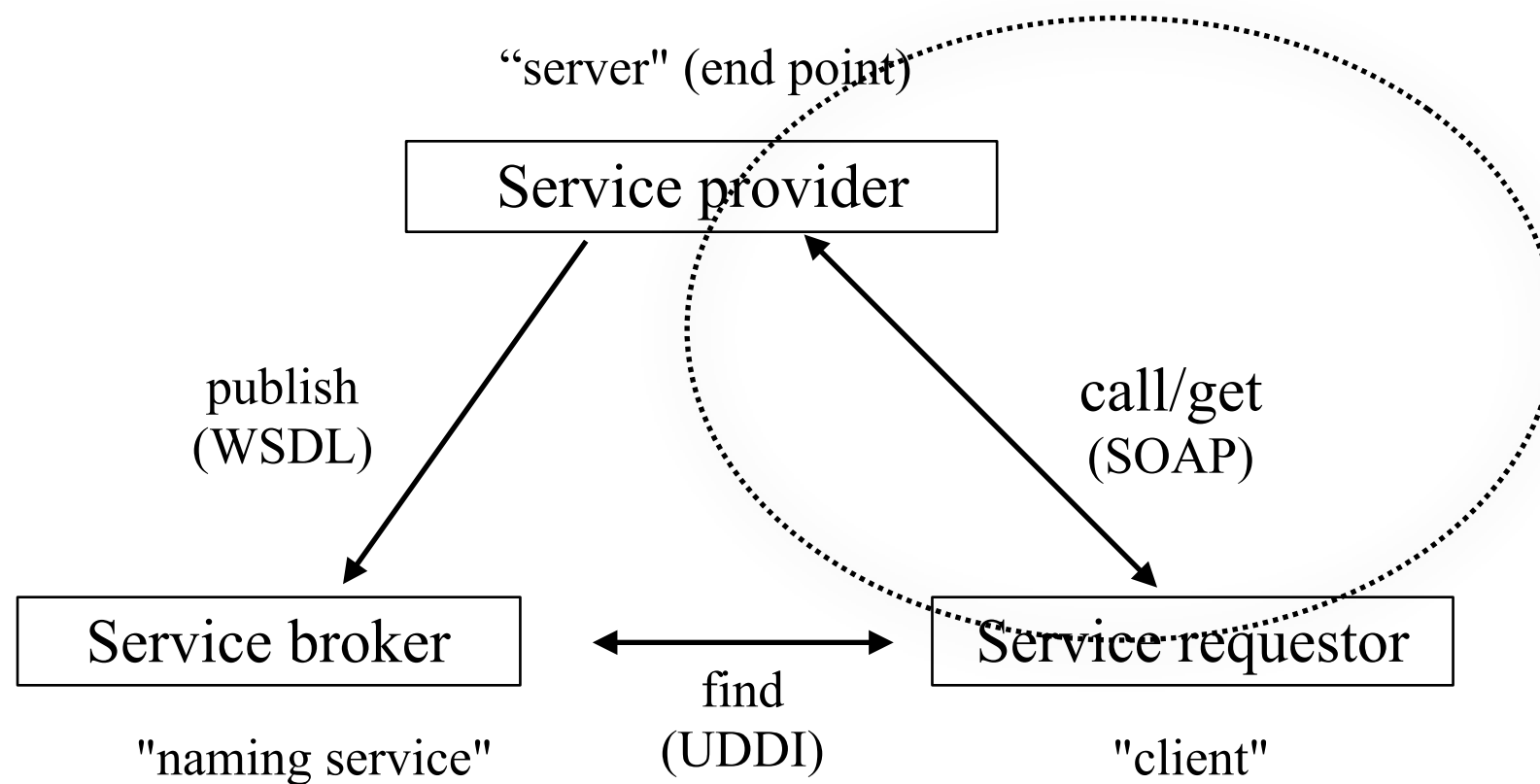# Web Services - Part 2

Syed Gillani
Laboratoire Hubert Curien St-Étienne, 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

"server" (end point)

Service provider

publish
(WSDL)

call/get
(SOAP)

Service broker

find
(UDDI)

Service requestor

"naming service"

"client"

[Simple Object Access Protocol (SOAP)]

▸ What is SOAP?

  ✓ a standard (W3C) messaging protocol for inter application

    communication

▸ SOAP codifies request and response (with XML) using HTTP as a

  means of transport (but can run over others as well e.g. SMTP)

▸ More powerful than XML-RPC

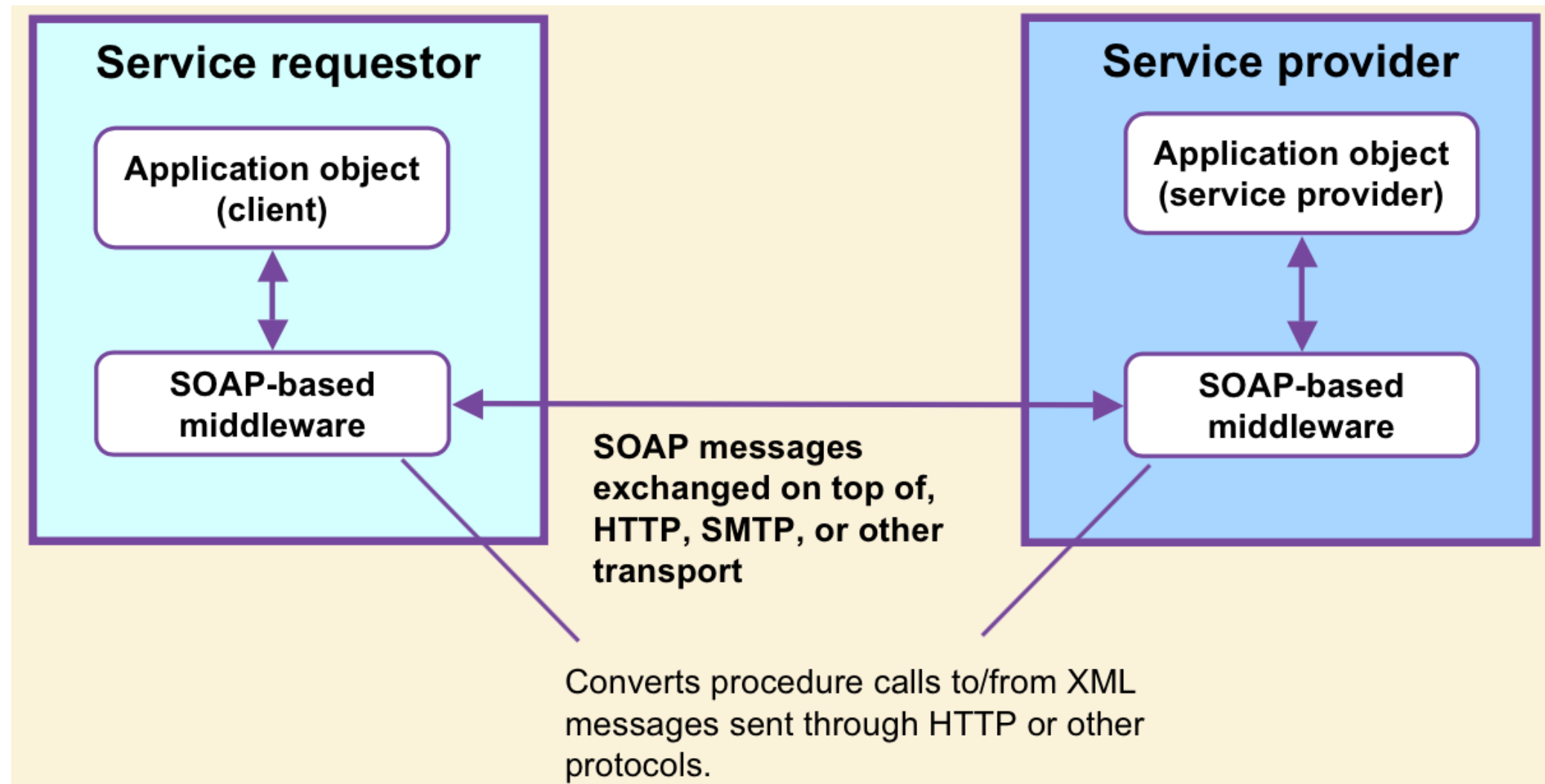 ✓ user-defined data types

  ✓ 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)]

Service requestor

Application object (client)

SOAP-based middleware

Service provider

Application object (service provider)

SOAP-based middleware

SOAP messages exchanged on top of, HTTP, SMTP, or other transport

Converts procedure calls to/from XML messages sent through HTTP or other protocols.

[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]
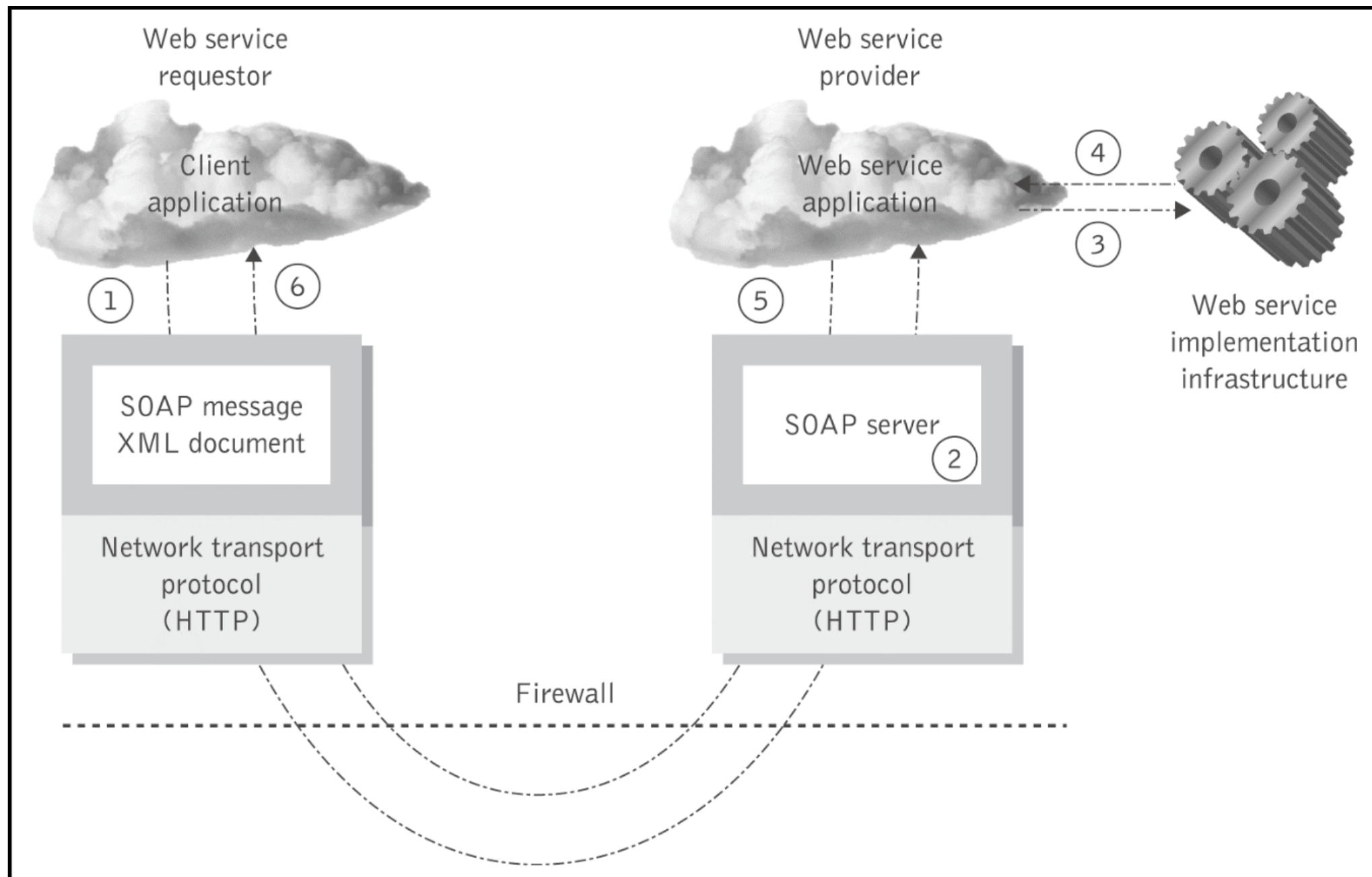
[SOAP Messages]

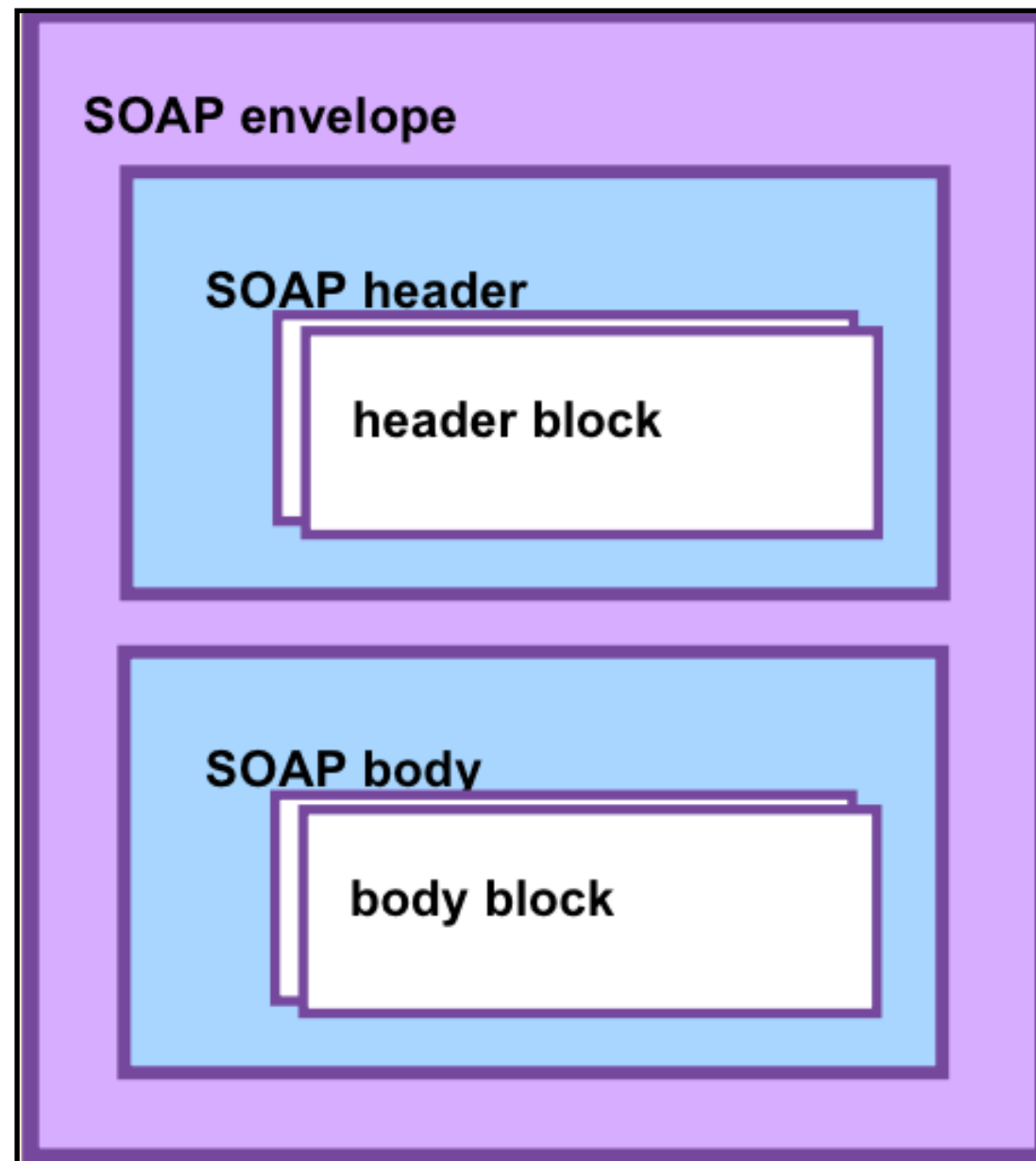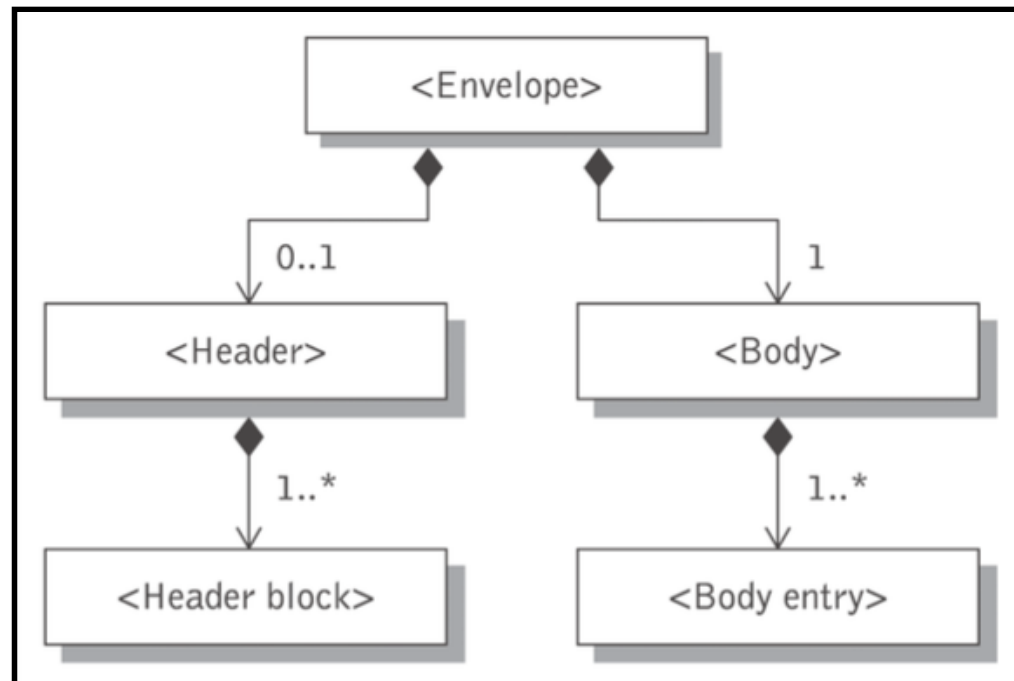SOAP messages are seen as envelopes where the application encloses
the data to be sent

<Envelope>

<Header>

SOAP envelope

SOAP header

header block

</Header>

<Body>

SOAP body

body block

</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
<?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>
```
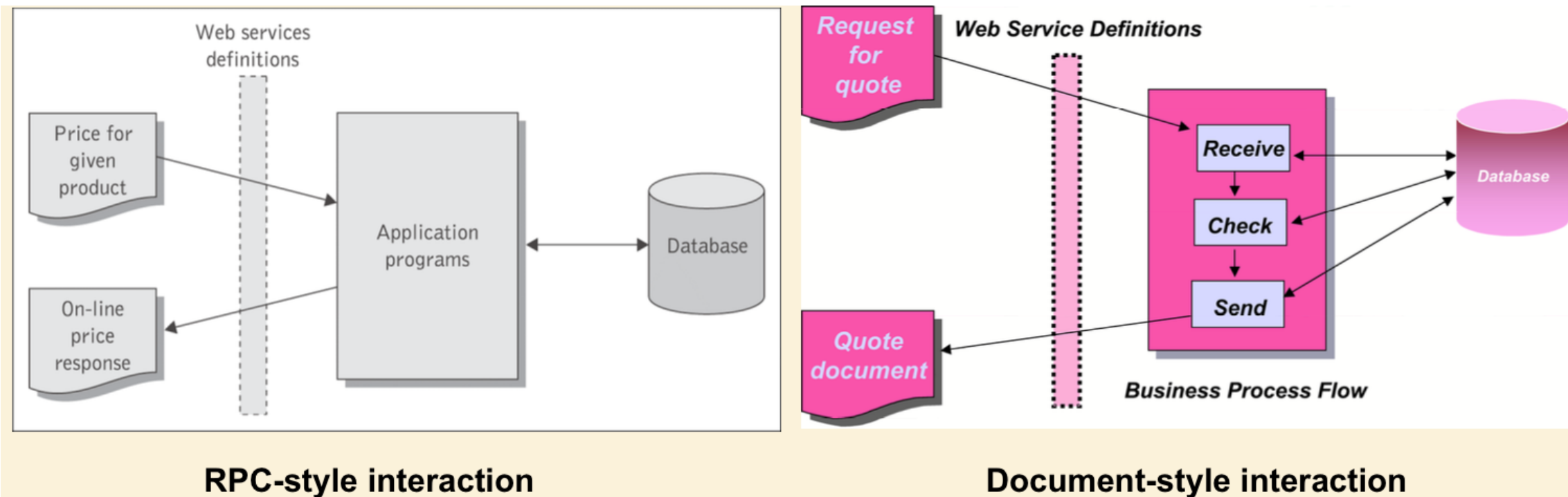
Envelope

Header

Blocks

Body

[The SOAP Communication Model]

‣ SOAP supports two possible communication styles:

  ✓ Remote Procedure Call (RPC)

  ✓ Document (or simple messages)

RPC-style interaction

Document-style interaction

[RPC-style SOAP Services]

‣ RPC-style web service appears as remote object to client

‣ RCP-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
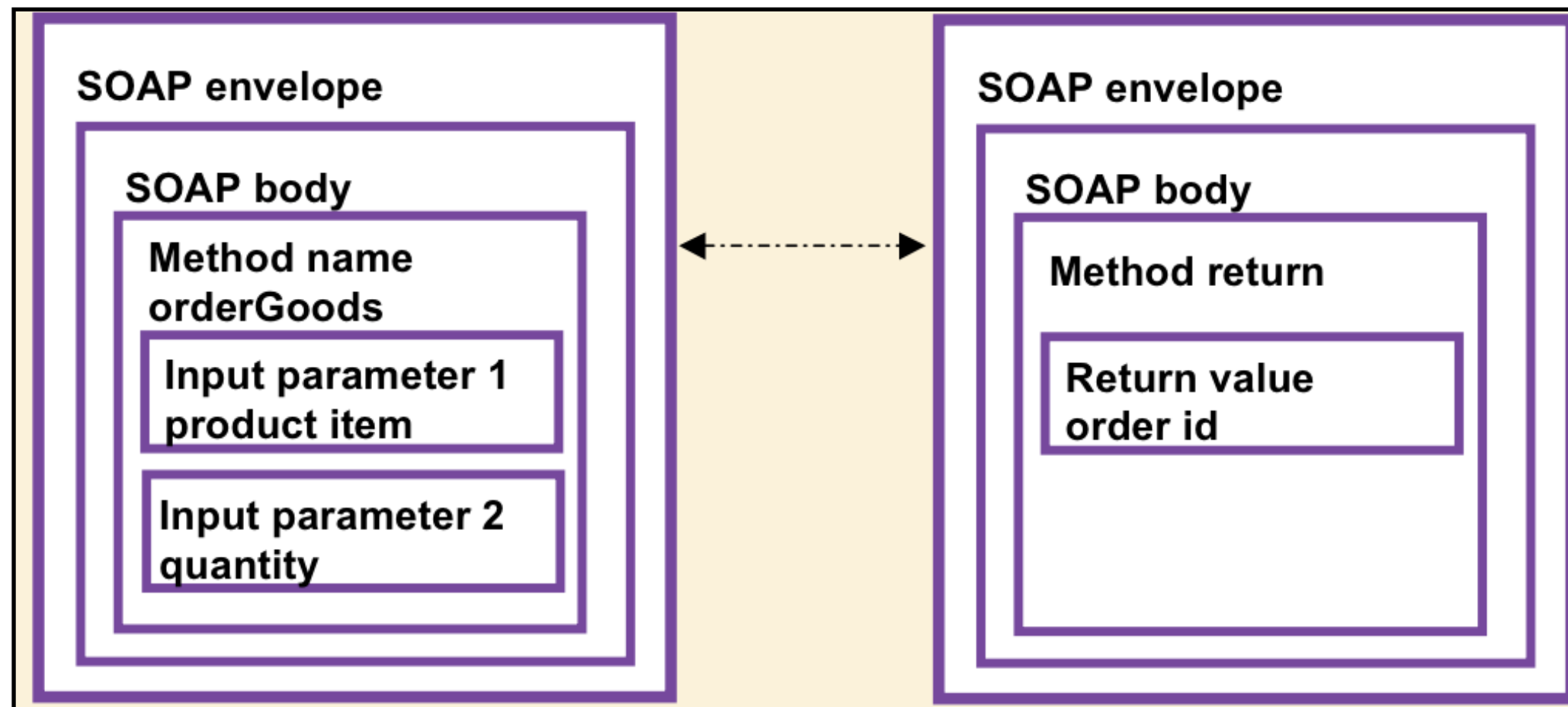
[RPC-style SOAP Services]

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>  450R6OP  </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>
```
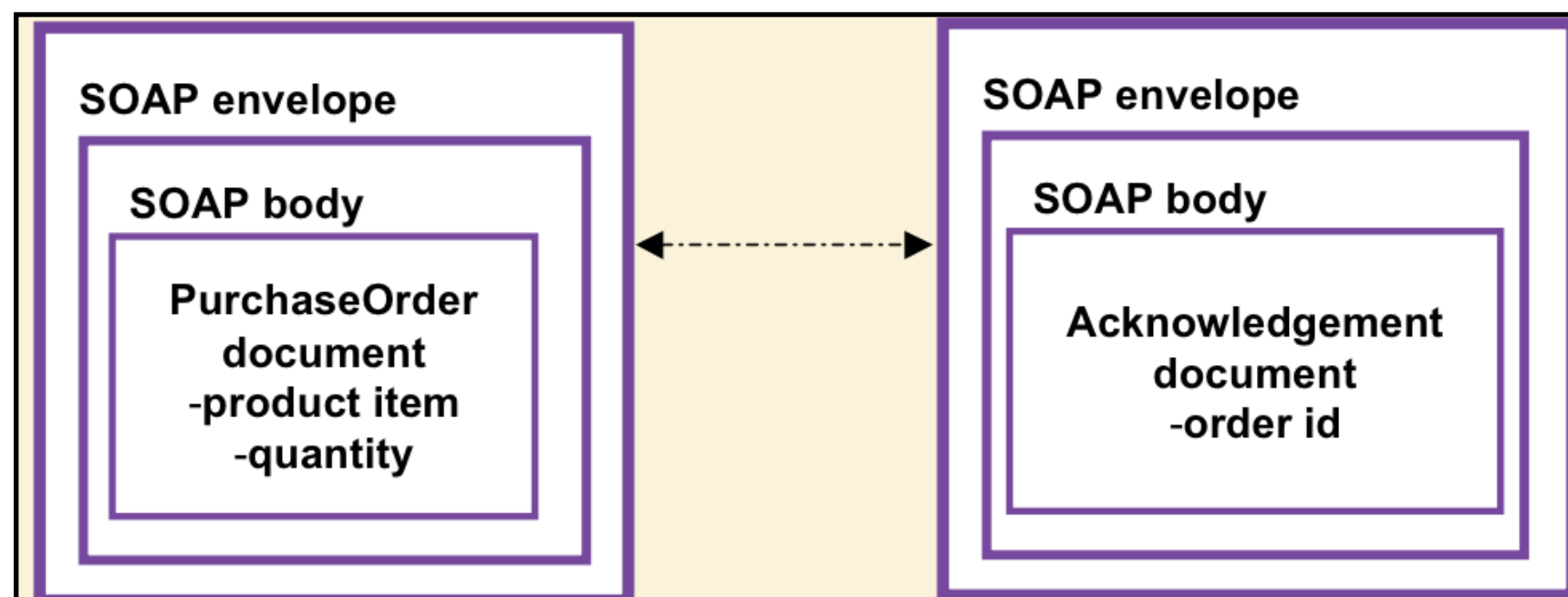
[Document(Message)-style SOAP Services]

- ‣ 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>
```
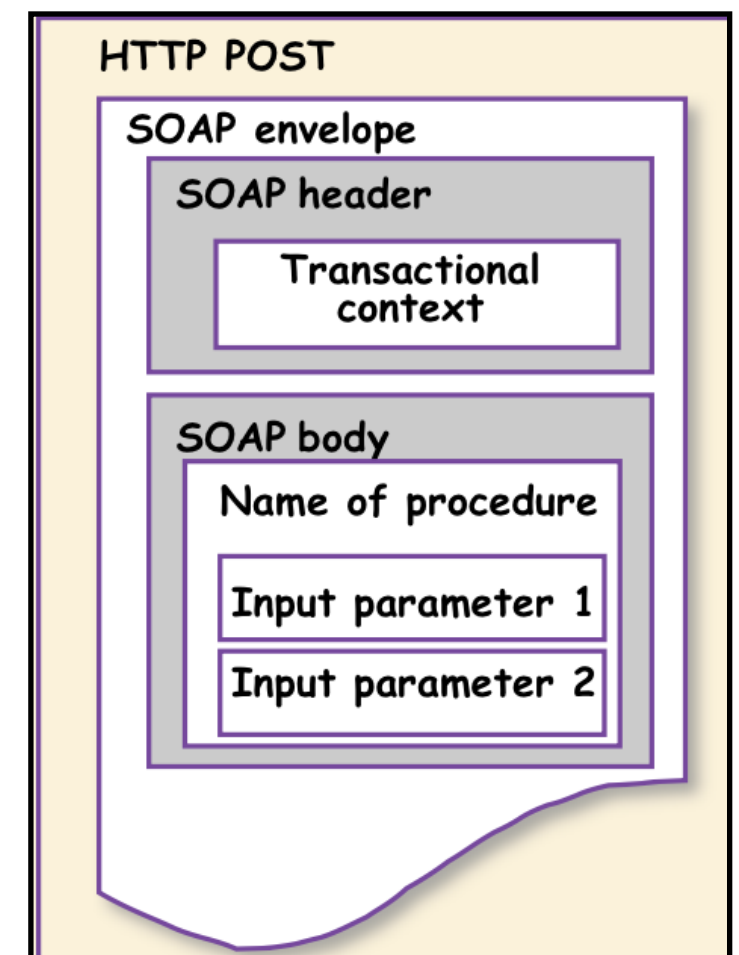
[SOAP Fault Element]

‣ SOAP provides a model for handling faults (e.g. entries

do not exists in the database)

‣ The SOAP body contain the fault message
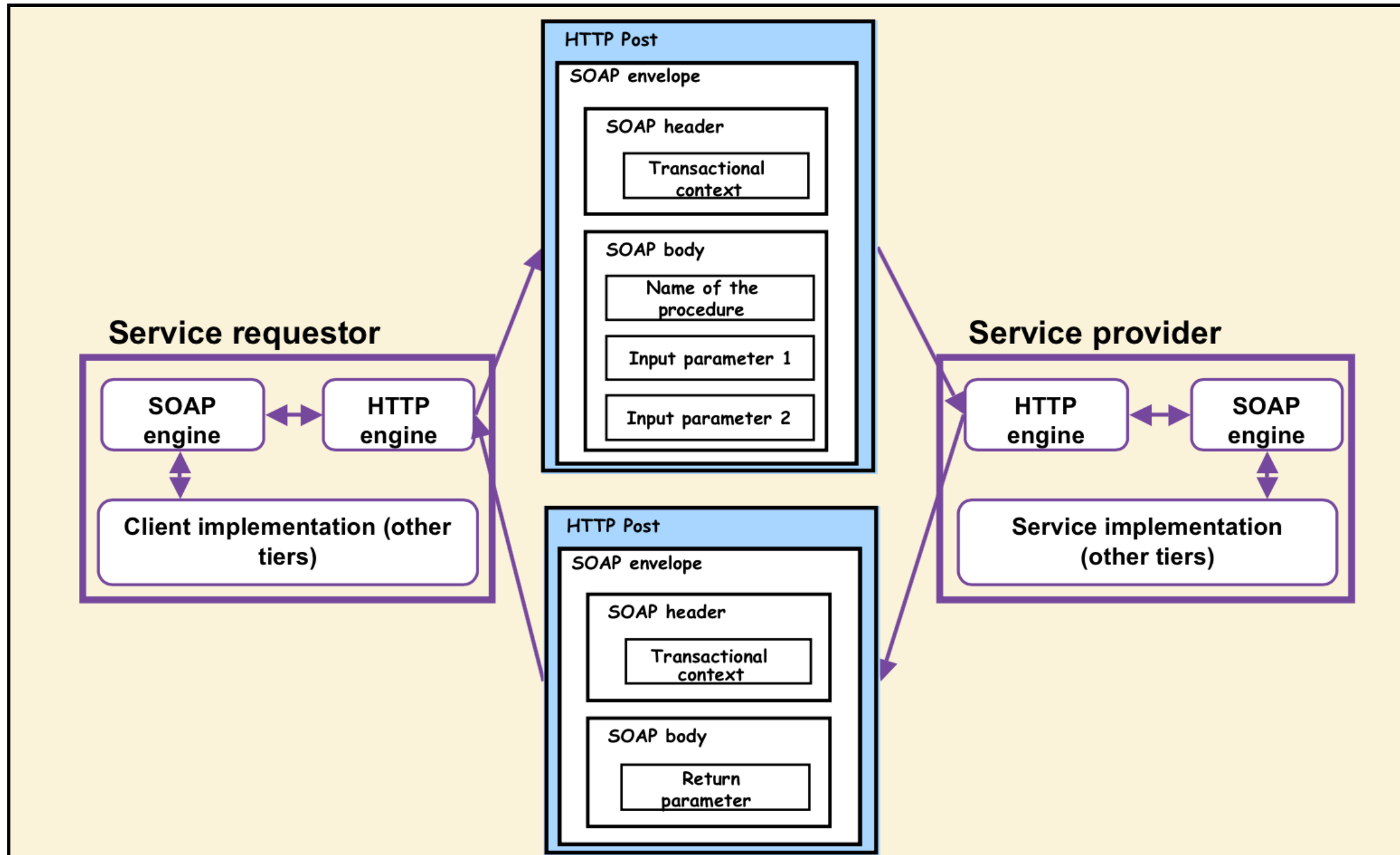
```
<env:Envelope
 xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
 xmlns:m="http://www.plastics_supply.com/product-prices">
    <env:Header>
               <tx:Transaction-id
          xmlns:t="http://www.transaction.com/transactions"
                 env:mustUnderstand='1'>
             512
             </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 os 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)

HTTP POST

SOAP envelope

SOAP header

Transactional context

SOAP body

Name of procedure

Input parameter 1

Input parameter 2

[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

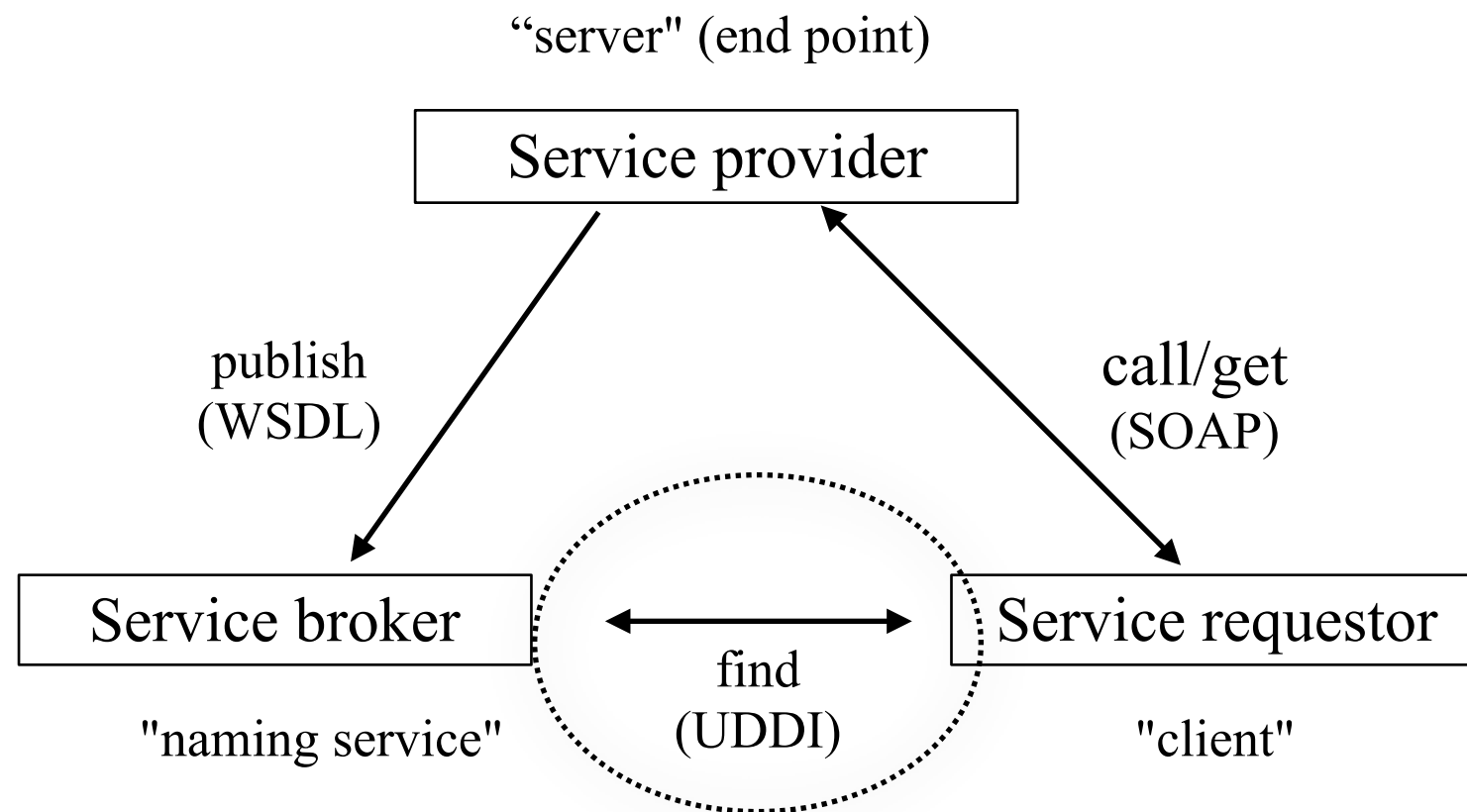   ✓ CORBA, COM (usually for distribution of data)

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

They are evolved from SOAP, so study it
by yourself

[RPC Summary]

▸ Its the backbone of the Web Services

▸ RPC technology focuses on programming use and aims to:

✓ make distributed communication similar to local calls

✓ support protocol evolution

✓ make it hard to get it wrong

▸ Semantics are challenging:

✓ can't really hide the network and make it look like local

▸ Performance is the Key

▸ You have learned about few technologies which you might use in the future

[Last Point]

"server" (end point)

Service provider

publish
(WSDL)

call/get
(SOAP)

Service broker

find
(UDDI)

Service requestor

"naming service"

"client"

[Universal Description Discovery and Integration (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 s 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

[Concluding Remarks]

‣ Web services are every where, and Today's world can't be realised without them

‣ Three main components of web services are WSDL, SOAP and UDDI

‣ SOAP is mainly adopted due to its flexibility and efforts of W3C

‣ These days its more about the performance than flexibility (thats why Facebook and Google has its own protocols)

‣ Its time to move to next phase of the Web, the Web 2.0

‣ Web 2.0 includes the semantics and the data structure that is able to interpret and reason about the selection and orchestration of the web services

[Fin]