Simple Object Access Protocol (SOAP)

Reference:

1. Web Services, Gustavo Alonso et. al., Springer

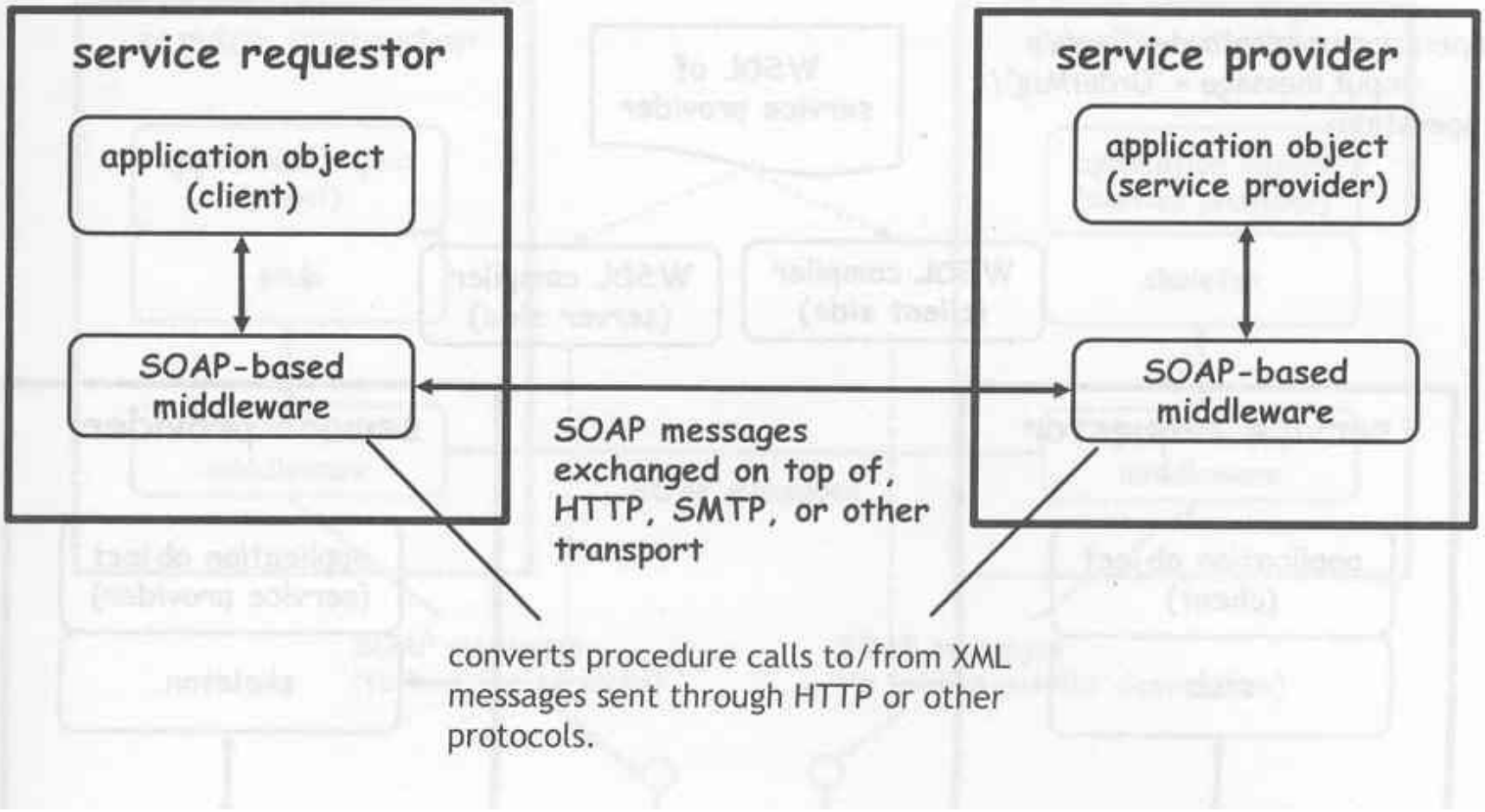
Minimal List

- Common Syntax is provided by XML
- To allow remote sites to interact with each other:

1. A common data format for messages being exchanged
2. A convention for supporting specific forms of interaction (such as messages or RPC)
3. A set of bindings for mapping messages into a transport protocol

Minimal List

- Transport protocol:
  TCP/IP, HTTP, SMTP (e-mail)
- Message interaction:
  loosely coupled
  However, RPC-style interaction to be offered
- Interactions are based on SOAP
  to turn a service invocation into an XML message, to exchange the message and to turn XML message back into an actual service invocation, as shown in the next figure


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.

Minimal List

- The role played by IDL in middleware, is played by WSDL in web services.
- WSDL is an XML-based IDL, and fulfills the same purpose as existing IDLs.
- It introduces a few extensions that are necessary to cope with the lack of a centralized middleware that manages transport and addressing.
- A designer specifies the programming interface of a web service, which is specified in terms of methods supported by Web service, where each method could take one message as input, and return another as output.

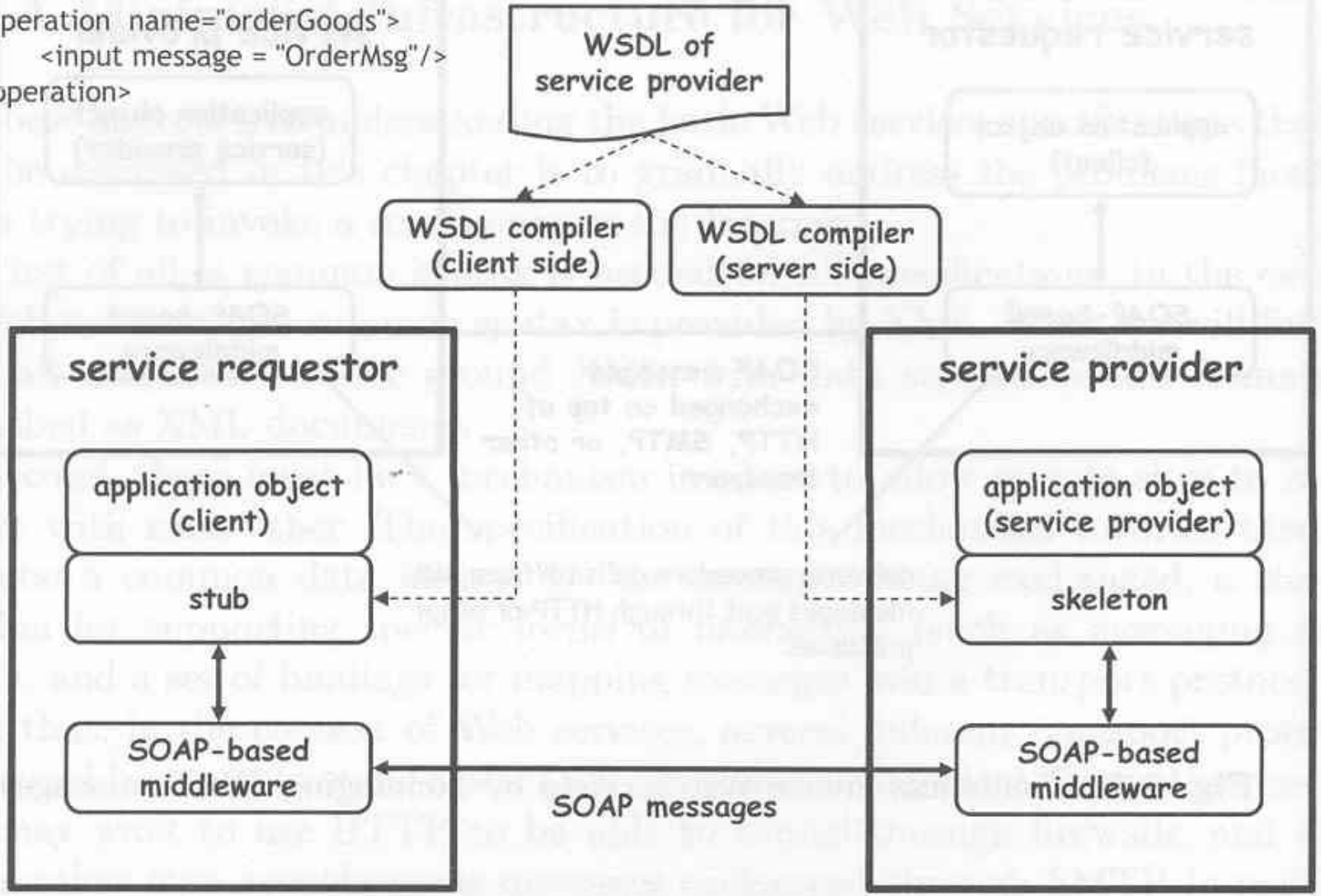
Minimal List

- In case of RPC-style interactions, these messages carry the input and output parameters of a procedure call.
- At each site, WSDL is used like conventional IDLs.
- WSDL file describing an interface can be compiled into the appropriate programming language to generate the stubs and intermediate layer that make calls to the Web services transparent.

Minimal List

- In the same way IDL makes RPC appear as local procedure call, WSDL is used to hide a web service behind some interface, e.g. a method call. That way, users do not need to worry about the fact that they need to invoke a Web service.
- The Web services infrastructure uses WSDL and SOAP to construct the proxy objects at the requestor and provider sides, so that developers can code their applications as if they were making local calls.
- Clients simply invoke a method using the programming language of their choice, and similarly services can be implemented as if they were invoked by a local method.

```
<operation  name="orderGoods">
      <input message = "OrderMsg"/>
</operation>
```



Minimal List

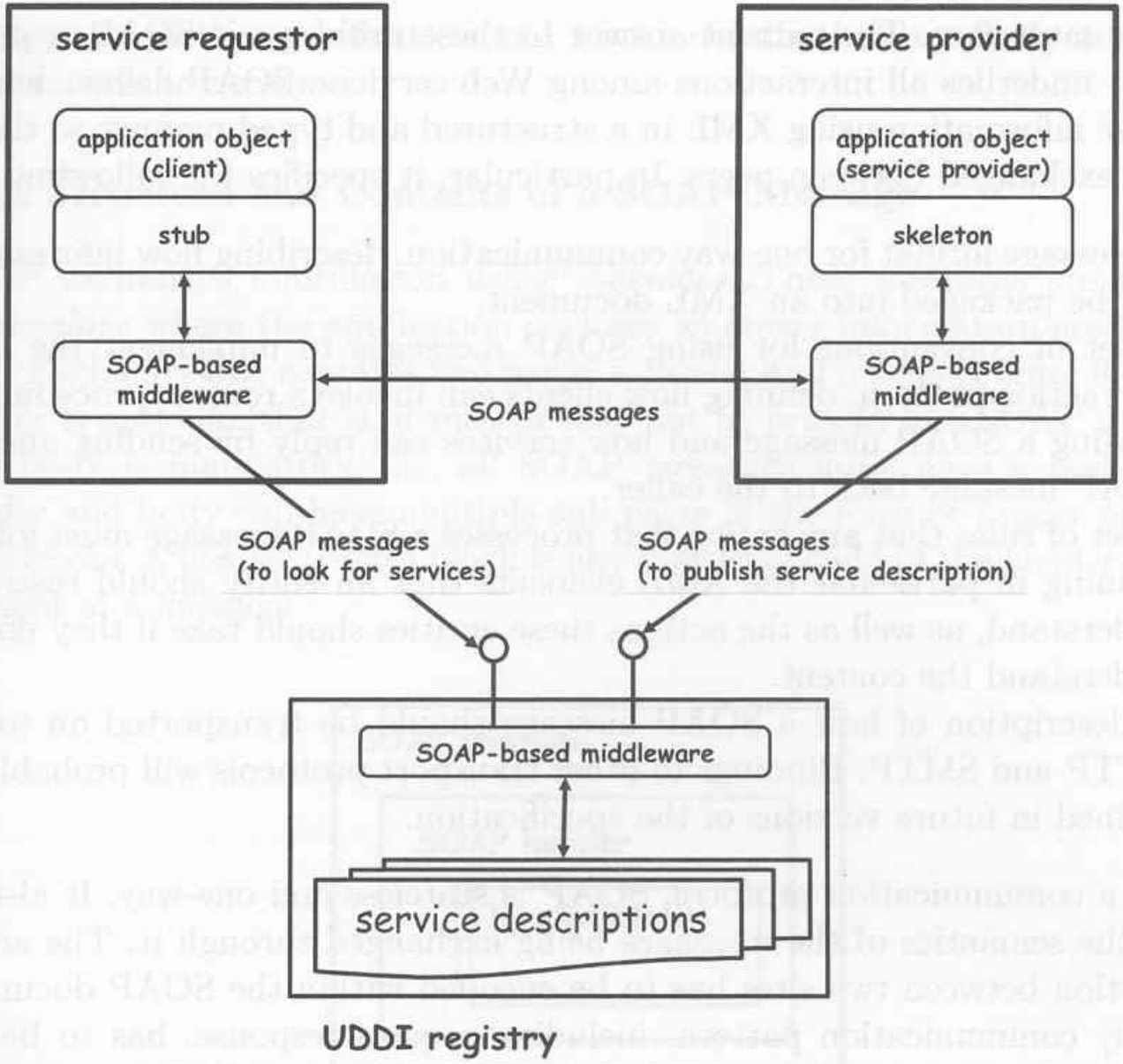
- The difference: absence of a common middleware platform.
- Once the services can be described and invoked, the only missing piece for a basic middleware infrastructure for Web services is a name and directory service.
- Being able to use Web services in a pervasive manner and at a global scale requires a standard way to publish and locate services.

Minimal List

- The requesters can look for services of interest and understand their properties, e.g. their interfaces and the URI at which they are made available.
- There is a requirement to standardize the Web services registry. Such standardization is taking place as part of Universal Description, Discovery, and Integration (UDDI) project.
- This effort comprises two parts: the UDDI registry and UDDI APIs.

Minimal List

- The registry is the equivalent name of the name and directory server found on any middleware platform.
- It is the place where service descriptions are published in catalogs that can be searched by users.


service requestor
application object (client)
stub
SOAP-based middleware
service provider
application object (service provider)
skeleton
SOAP-based middleware
SOAP messages
SOAP messages (to look for services)
SOAP messages (to publish service description)
SOAP-based middleware
service descriptions
UDDI registry

Simple Object Access Protocol (SOAP)

- W3C started working on SOAP in 1999.
- SOAP 1.0 was entirely based on HTTP.
- SOAP 1.1, May 2000, on top of variety of transport protocools.
- SOAP 1.2, May 2003, is a joint effort from Canon, IBM, Microsoft and Sun.

Simple Object Access Protocol (SOAP)

- A message format for one-way communication, describing how information can be packaged into an XML document.
- A set of conventions for using SOAP messages to implement RPC interaction pattern
- A set of rules that
  - any entity that processes SOAP messages must follow,
  - defining in particular the XML elements that an entity should read and understand,
  - as well as actions these entities should take if they do not understand the content.

Simple Object Access Protocol (SOAP)

- A description of how a SOAP message should be transported on top of HTTP and SMTP.
- Binding to other protocols will probably be defined in future versions of the specification.
- SOAP is a stateless and one-way.
- It ignores the semantics of the messages being exchanged through it.
- The actual interaction between two sites has to be encoded within SOAP document and any communication pattern, including request-response has to be implemented by the underlying systems.

SOAP (Simple Object Access Protocol)

- Used to exchange messages via HTTP, SMTP, and SIP (Session Initiation Protocol for Internet telephony)
- Originally designed for remote-procedure calls (RPC)
- Works through firewalls on port 80
- Character-based, so easy to encrypt/decrypt and thus easy to secure
- Inefficient due to character, not binary, data and large headers
- Does not describe bidirectional or n-party interaction

SOAP

- SOAP is a simple and lightweight XML-based mechanism for creating structured data packages that can be exchanged between network applications.
- The latest SOAP specification is SOAP 1.2, which passes the recommendation criteria of the W3C organization.

SOAP

The SOAP specification defines the following fundamental components:

- An envelope that defines a framework for describing message structure
- A set of encoding rules for expressing instances of application-defined data types
- A convention for representing remote procedure calls (RPC) and responses

SOAP

- A set of rules for using SOAP with HTTP
- Message exchange patterns (MEP) such as request–response, one-way, and peer-to-peer conversations.

SOAP can be used in combination with a variety of network protocols, such as HTTP, SMTP, BEEP, FTP, and RMI/IIOP.

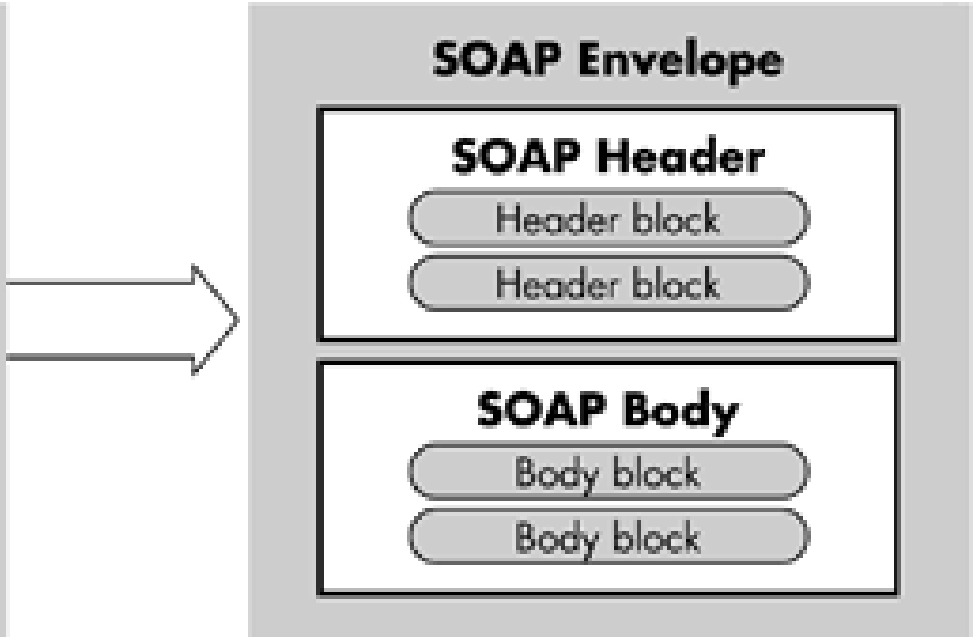
SOAP

- SOAP is currently being used as the de facto standard for XML messaging including enveloping and exchanging messages.
- SOAP provides a simple enveloping mechanism and is proven in being able to work with existing networking services technologies, such as HTTP. SOAP is also very flexible and extensible.
- It provides capabilities to add-on standards and application-defined extensions.
- The wide acceptance of SOAP is based on the fact that it builds upon the XML infoset.

SOAP

- The format of a SOAP message is formally defined in SOAP 1.2; specifically, the Messaging Framework specification.
- The next figure illustrates a simple SOAP message structure, as defined by this specification.

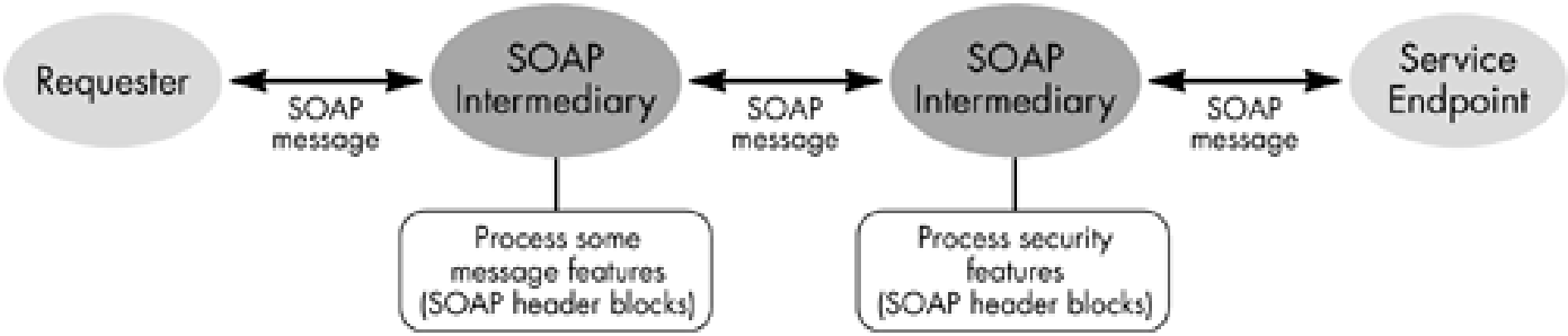
```
<?xml version='1.0'?>
<env:Envelope
        xmlns:env
        ="http://www.w3.org/2003/
        05/soap-envelope">

    <env:Header>
    </env:Header>

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



SOAP message formats

SOAP

- A SOAP message is packaged in a SOAP:Envelope, which consists of zero or more SOAP:Header and exactly one SOAP:Body.
- The SOAP body and header can be further divided into blocks.
- The SOAP body blocks are intended for the final receiver,
- The header blocks can be interpreted by the SOAP intermediaries.
- The next figure illustrates such a message exchange pattern.



SOAP intermediaries

SOAP

- The SOAP header blocks carry information such as
  - security,
  - transactional information,
  - correlation, and so on.
  - These intermediaries may act on these header blocks, add more blocks, change them, or leave them untouched.
- These header blocks can be targeted to some intermediary or final receiver by the use of "roles" attributes ("actors" in SOAP 1.1).

SOAP

- The value (specified by URI) of the "roles" attribute can be an address or a standard role name as defined by the SOAP specification, which are:
- "next"- each SOAP intermediary and the ultimate receiver must act on this role
- "none"- the SOAP nodes must not act in this role
- "ultimateReceiver"- the ultimate receiver must act in this role.

SOAP

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-
   envelope">
<env:Header>
<a:firstheaderBlock xmlns:a="http://ph.com"
   env:role="http://ph.com/example/role">
...
</a:firstheaderBlock>
<q:secondBlock xmlns:q="http://ph.com"
   env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
...
</q:secondBlock>
</env:Header>
<env:Body >
...
</env:Body> </env:Envelope>
```

SOAP

- Here <a:firstheaderBlock> is targeted to SOAP nodes who are acting on the role http://ph.com/example/role
- whereas the <q:secondBlock> must need to be processed by all SOAP nodes in the path, and finally by the ultimate receiver.
- If no "role" is specified, it is assumed for the ultimate receiver.

SOAP

- We must be cautious in the distinction of these roles, in that we must identify the actor on a message header for the correct processing of the message.
- There are two other attributes that we must combine with the "role" attribute.
- They are "mustUnderstand" and "relay."

SOAP

- Thus far, we are discussing topics regarding the format of the message.
- However, SOAP is much more than simply a message format, it also provides a simple framework for extensible messages and message processing features.

The SOAP Processing Model

- The processing of a SOAP message is dependent on the role assumed by the processor.
- As we have mentioned above, the SOAP headers are targeted using a "role" attribute.
- If the SOAP intermediary plays the role as defined by the SOAP message, it can then process the message.

The SOAP Processing Model

- There are two options related to processing.
- If the SOAP header is targeted to this node and specifies a "mustUnderstand" flag set to "true," then the processing node must process that header.
- If there is no such requirement (i.e., mustUnderstand flag is not set), it is up to the processing node to decide on the processing of the message.

The SOAP Processing Model

- Once the processing is completed, the message will be directed to the next node.
- The decision on the next node selection is not specified by the SOAP specification.
- Therefore, it is now the choice of the processing node to make such a decision.
- However, there are some standards that exist to specify common routing mechanisms, such as WS-Routing and WS-Addressing.

The SOAP Processing Model

- Another interesting aspect of this message-forwarding paradigm is the concept of relaying SOAP headers.
- A header can have a "relay" attribute value (i.e., true or false) to indicate that non-processed headers get forwarded to the next node.
- The default value is "false." This indicates a SOAP node, which is targeted by this header, will not forward this header to the next node.

The SOAP Processing Model

- If we refer back to the SOAP 1.1 specification (SOAP1.1), we could see that there is no standard processing rules for SOAP extensions (header) processing.
- This causes a number of interoperability problems.
- However, the SOAP 1.2 specification introduces the following core SOAP constructs in order to support SOAP extensions.

SOAP Features

- A SOAP feature is an extension to the SOAP messaging framework.
- These features are common in distributed computing such as
  - reliability, security, correlation, routing, and message exchange patterns such as request/response, one-way, and peer-to-peer conversations.
- This is an abstract concept with the indication that there is some processing needs to be done in the SOAP nodes but it does not specify how this processing is done.

SOAP Features

A SOAP feature has the following characteristics:

- A unique name used to identify the feature and its properties.
- This enables us to identify whether a SOAP node supports a specific feature.
- For example, if we have a feature called "secure-ssl-channel," then we can ask the SOAP nodes, including the ultimate receiver, whether they support that feature or not.

SOAP Features

- A set of properties associated with a feature that can be used to control, constrain, or identify a feature.
- For example, we can see in the SOAP request-response message exchange pattern there are properties for accessing the inbound or outbound messages, the immediate sender, and next destination.

SOAP Features

It is important to understand that SOAP provides two mechanisms for implementing these features:

SOAP header blocks.

- In this kind of implementation SOAP header blocks are used to specify a feature.
- These headers are processed by the SOAP nodes. The SOAP processing model defines the behaviors of a single processing SOAP node in order to process an individual message.
- The most common example of such a feature is the security features as defined by WS-Security specifications.

SOAP Features

SOAP binding protocol

- In this case the features are directly implemented in the protocol binding level.
- For example, a binding extension to support the SOAP over SSL protocol.

SOAP Features

- In the first case it is more protocol independent and flexible, but this may cause unnecessary processing overhead.
- The second case is protocol dependent and the flexibility is limited.
- In addition to the extensions as specified in the above cases, the SOAP specification defined a standard feature for the message exchange pattern called MEP.
- Let us now explore in further detail exactly how this is defined.

Message Exchange Pattern

- One special type of SOAP feature is the MEP. A SOAP MEP is a template that establishes a pattern for the exchange of messages between SOAP nodes.
- Some examples of MEPs include request/response, one-way, peer-to-peer conversation, and so on.
- To clarify further, we could consider a special MEP, a request for proposal (RFP) MEP, where we could define a message exchange pattern for the RFP such as submitting a request to the service provider, getting a response from the provider at a later period, and other subpatterns. We can envision building these kinds of message exchange patterns.

Message Exchange Pattern

- MEP, similar to other features, is implemented either as headers or using protocol bindings.
- A MEP may be supported by one or more underlying protocol binding instances either directly or indirectly with support from the software.
- This software implements the required processing to support the SOAP feature, expressed as a SOAP module.

SOAP Modules

- The combined syntax and semantics of a set of SOAP headers are known as a SOAP module.
- A SOAP module realizes one or more SOAP features.
- This enables us to specify a more general-purpose concept such as a secure purchase order, with a combination of one or more features, including the purchase order MEP as described above, the security feature, and more.

SOAP Modules

- So far we have discussed message packaging, transmission, and processing in a transport in a rather independent manner.
- These SOAP messages can be transported through different transport bindings, such as HTTP, TCP, and BEEP.
- The SOAP defined some protocol bindings for its transport methods.
- The most notable one is SOAP over HTTP utilizing the GET and POST operations.

Processing a SOAP message

- The division between header and body already provides some indication of how SOAP expects messages to be processed by the different nodes along the message path.
- These nodes will, in most cases, constitute the different tiers of the Web services middleware.
- Nodes processing a message can play one or more roles. Each block in a SOAP header may include the definition of the role for which it is intended.

Processing a SOAP message

Three roles: none, next, ultimateReceiver

- If a block is assigned to a none role, it means that such a block should not be processed by any node receiving the message, although the block might be read if it contains information important for processing other blocks.
- If a block is assigned to the utilimateReceiver role, that block is solely intended for the recipient of the message, not for any intermediate node.

Processing a SOAP message

- If a block is assigned to the next role, every node receiving the message can process that block.
- The ultimateReceiver is also included in the the set of next nodes.
- The body of the message does not have a role associated to it.
- Its role defaults to ultimateReceiver.

Processing a SOAP message

Different Operations

- Processing can involve:
  - removing blocks for the header,
  - performing some action like logging the message, extend the header with additional information, removing the header and introducing a new one and more.
- A block may contain a mustUnderstand flag, indicating that a node playing the role indicated by the block must process the block as needed.

Processing a SOAP message

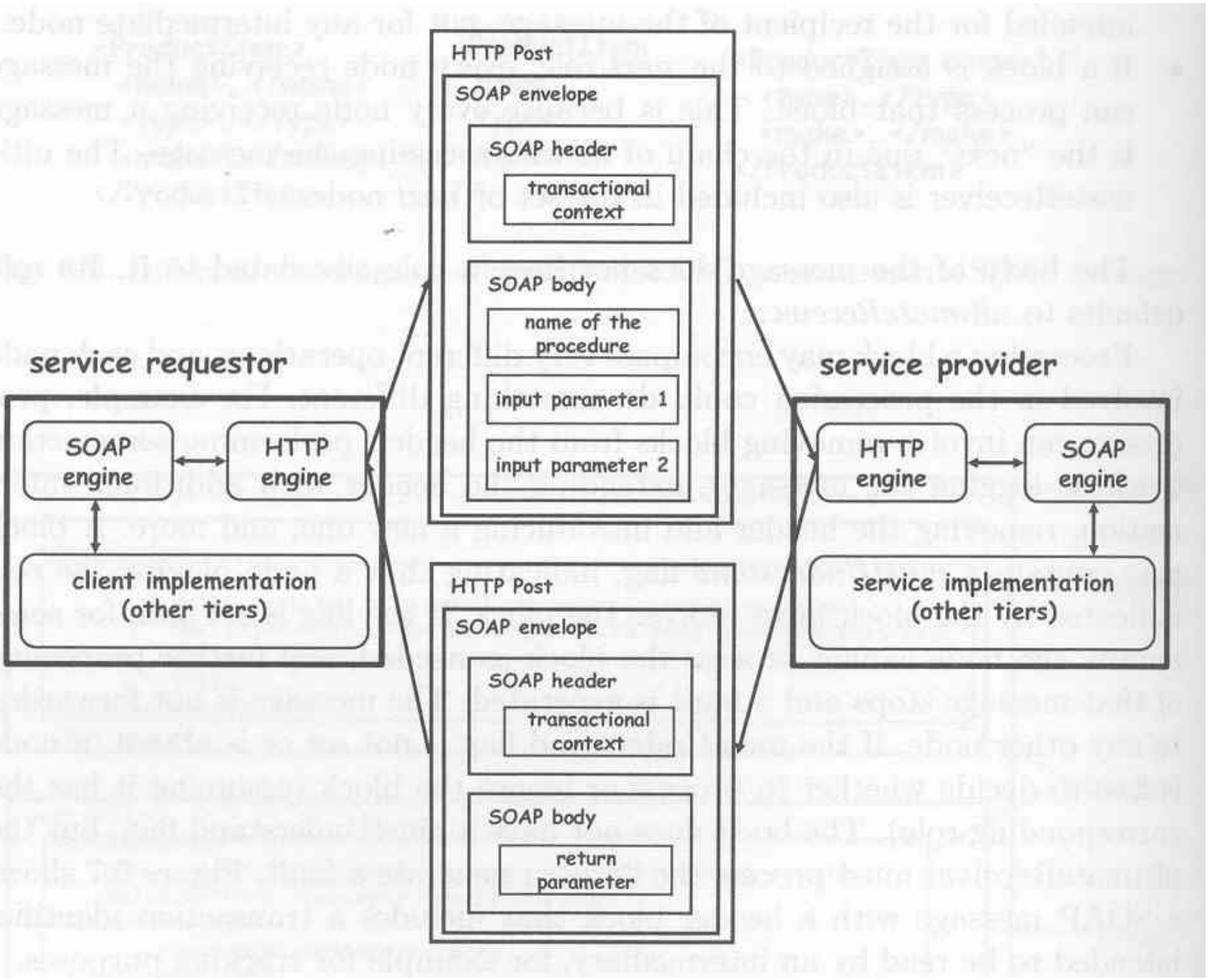
- If the flag is set and for some reason, the node cannot process the block as needed, any further processing of that message stops and a fault is generated. The message is not forwarded to any other node.
- If the flag is not set or absent, a node is free to decide whether to process or ignore.
- The body does not have a mustUnderstand flag. The utimateReceiver must process the body or generate a fault.

Processing a SOAP message

- The previous figure shows a SOAP message with a header block that includes a transaction identifier intended to be read by an intermediary, for example for tracking purposes.

Binding SOAP to a Transport Protocol

- SOAP is typically associated with HTTP but it can also be used with SMTP.
- The specification of which protocol to use is called binding. It defines how a message is wrapped within a transport protocol and how the message has to be treated using primitives of the transport protocol used.
- When SOAP is used over HTTP, SOAP can get GET, POST or other HTTP primitives. POST can be used to implement the RPC invocation.


HTTP Post
SOAP envelope
SOAP header
transactional context
SOAP body
name of the procedure
input parameter 1
input parameter 2
service requestor
SOAP engine
HTTP engine
client implementation (other tiers)
service provider
HTTP engine
SOAP engine
service implementation (other tiers)
HTTP Post
SOAP envelope
SOAP header
transactional context
SOAP body
return parameter

Ex. SOAP Request

```
POST /temp HTTP/1.1
Host: www.socweather.com
Content-Type: text/xml; charset="utf-8"
Content-Length: xxx
SOAPAction: "http://www.socweather.com/temp"

<!-- The above are HTTP headers -->
<?xml version="1.0"?>
<env:Envelope
 xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
 env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
 <env:Body>
  <m:GetTemp xmlns:m="http://www.socweather.com/temp.xsd">
    <m:City>Honolulu</m:City>
    <m:When>now</m:When>
  </m:GetTemp>
 </env:Body>
</env:Envelope>
```

Ex. SOAP Response

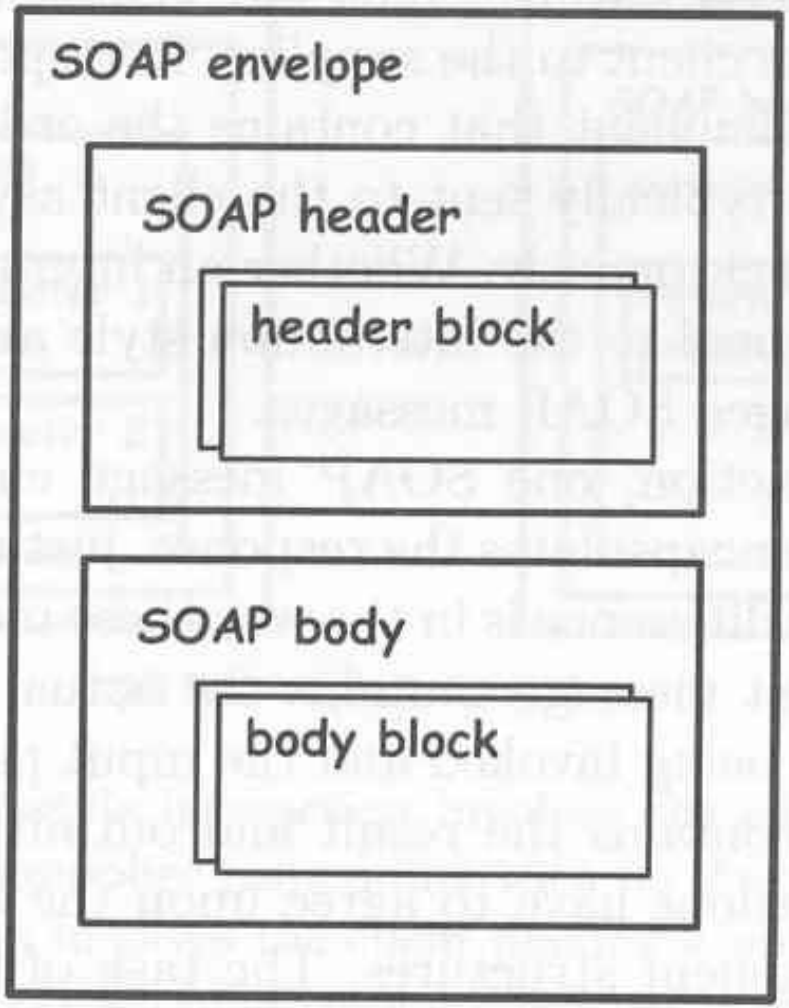
```
HTTP/1.1  200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: xxx
SOAPAction: "http://www.socweather.com/temp"

<?xml version=“1.0”?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
    env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   <env:Body>
      <m:GetTempResponse xmlns:m="http://www.socweather.com/temp.xsd">
         <DegreesCelsius>30</DegreesCelsius>
      </m:GetTempResponse>
   </env:Body>
</env:Envelope>
```

Simple Object Access Protocol (SOAP)

- SOAP is created to support loosely-coupled applications that interact by exchanging one-way asynchronous messages with each other.
- Any further complexity in the communication pattern such as two-way synchronous messaging of RPC-style interaction requires SOAP to be combined with underlying protocol or middleware that has some additional properties.
- RPC-style: to encode the input parameters, call to the procedure into one SOAP message. The response of the procedure to be encoded in another SOAP message. HTTP should be used to transport two messages.

Structure and Contents of a SOAP message

- The messages are used as envelope.
- Each envelope contains two parts: a head and a body.
- The header is optional, the body is mandatory, both can have multiple sub-parts in the form of header blocks or body blocks.

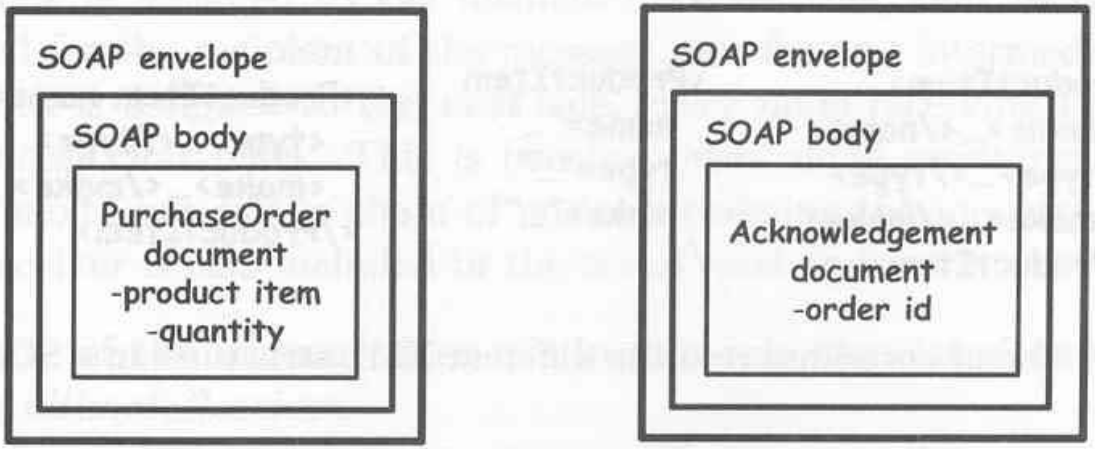

SOAP envelope
SOAP header
header block
SOAP body
body block

Structure and Contents of a SOAP message

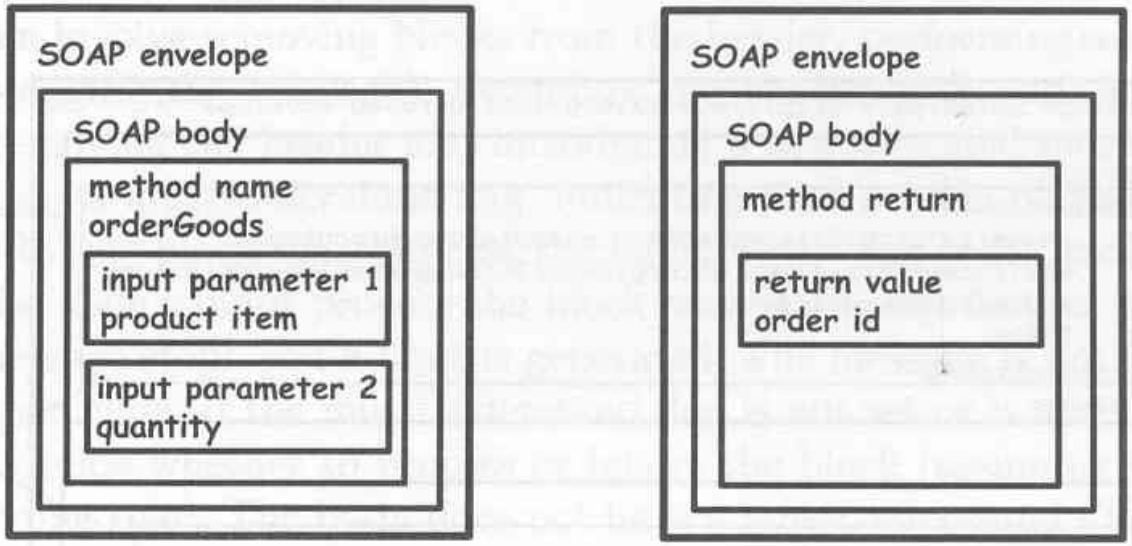
- SOAP assumes that every message has a sender and ultimate receiver, and arbitrary number of intermediaries (or nodes) that process the message and route it to the receiver.
- The core information the sender wants to transmit to the receiver should be in the body of the message. Any additional information necessary for intermediate processing or added value services like transactional interaction, security etc. goes into the header.
- The header contains information that can be processed by intermediate nodes. Intermediaries may process the header, add new headers, keep track of the message etc. This is required for multi-tier architecture.

Structure and Contents of a SOAP message

- SOAP can be used in in two interaction styles:
  Document-style and
  RPC-style



(a) Document-style interaction



(b) RPC-style interaction

Structure and Contents of a SOAP message

- Two interacting applications have to agree upon RPC method signature.

RPC-style:

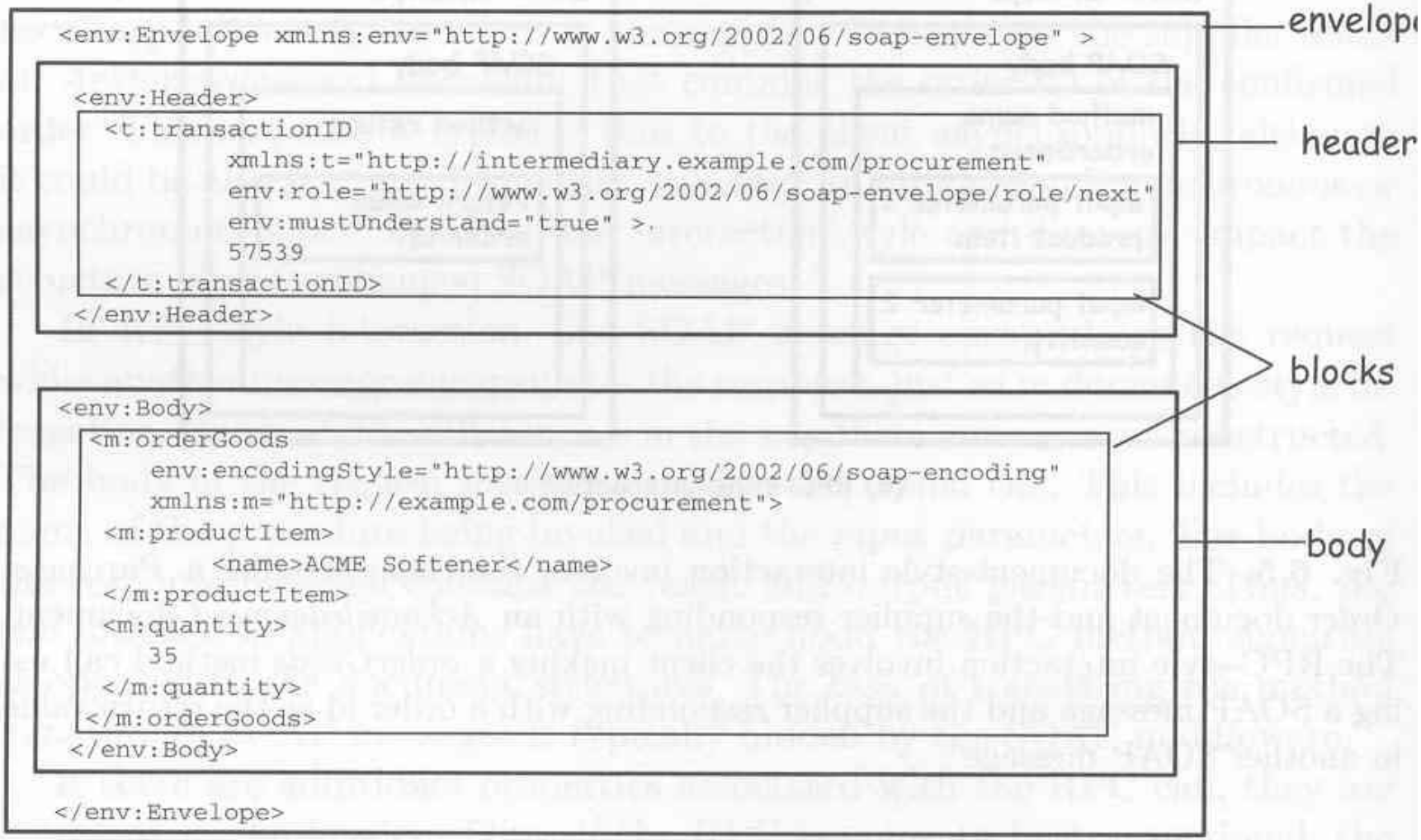
- Transactional, the header would include the necessary transactional context

Structure and Contents of a SOAP message

- The structure of a SOAP message is also influenced by encoding rules, which define how a particular entity or data structure is represented in XML.
- In order for the client and server to interoperate, it is essential that they agree how the contents of a SOAP message are encoded.
- SOAP 1.2 defines a particular form of encoding called SOAP encoding. This defines how data structures including basic types such as integers and strings as well as complex types such arrays and structures can be serialized into XML.

Structure and Contents of a SOAP message

- SOAP 1.2 does not impose any specific form of encoding. Applications are free to ignore SOAP encoding and choose a different one.
- Two applications can simply agree upon an XML Schema representation of a data structure as the serialization format for that data structure. This is referred as literal encoding.
- Next figure shows a complete SOAP message in RPC-style interaction using SOAP encoding.


<?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>
envelop
header
blocks
body

Binding SOAP to a Transport Protocol

- As a part of HTTP protocol, the receiving node has to acknowledge the POST request, indicating whether it received the request and whether it could appropriately process the request.
- As part of this response, the receiving site includes the RPC response or the corresponding fault message.

Binding SOAP to a Transport Protocol

- SOAP transport bindings have another implicit functionality: addressing.
- Identification of ultimate receiver's address is not part of a SOAP message, it is resolved by including the SOAP message as part of an HTTP request or as part of SMTP message.
- In case of HTTP, the URL of the target resource describes the receiver of the SOAP message.
- In case of SMTP, the "to" address in the e-mail header also describes the SOAP receiver.

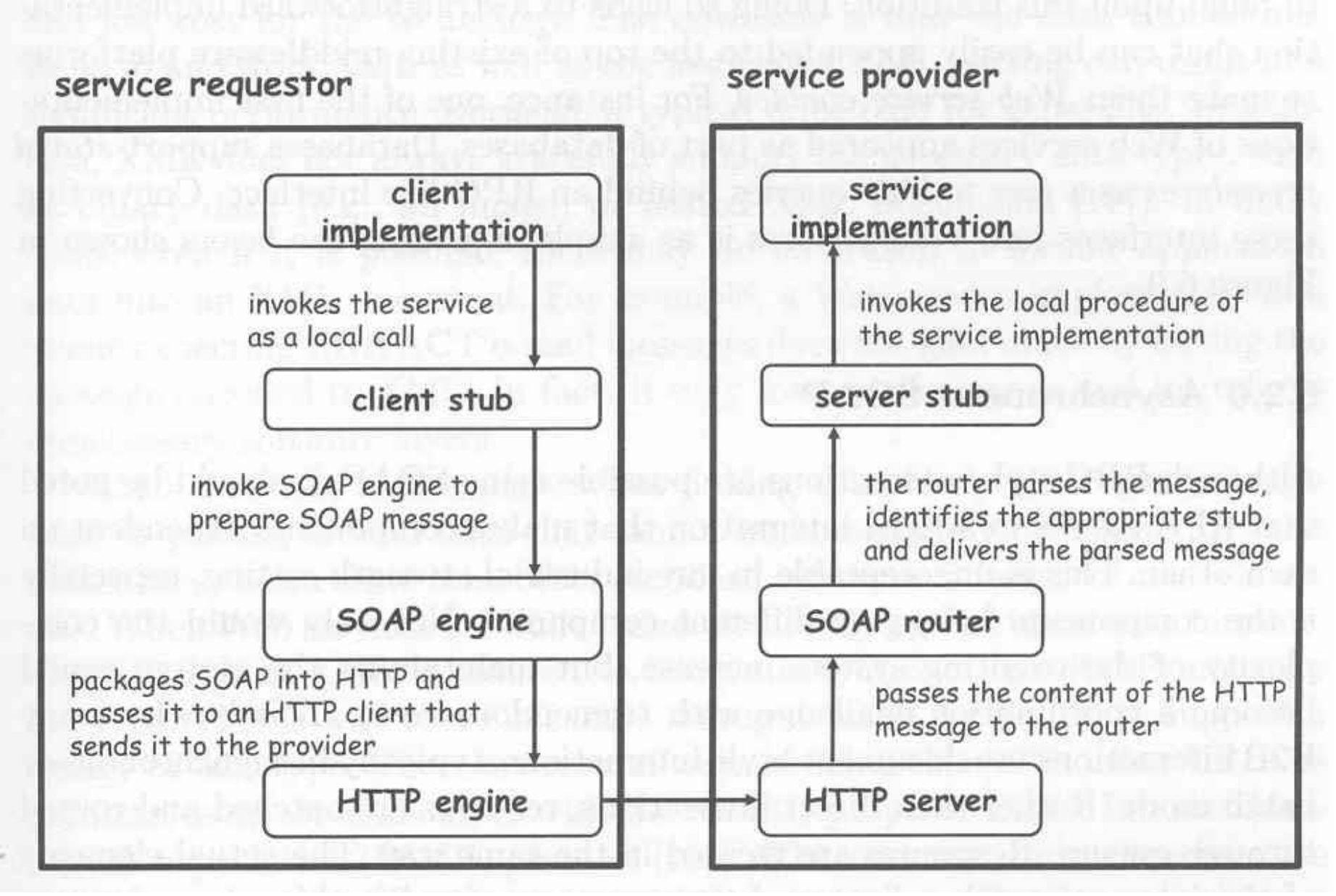
Binding SOAP to a Transport Protocol

Routing

- Though SOAP describes the notion of a SOAP path as comprising of a set of nodes, there is no mechanism for describing a SOAP path as part of a SOAP message.
- The path of a SOAP message is essentially the same as the route taken by the underlying protocol message.
- Some emerging Web services standards that propose alternative solutions to addressing and routing.

Simple implementation of SOAP

- The next figure shows how to implement SOAP-based interaction. It follows the same principles as the implementation of RPC.
- The call is in reality an invocation of a proxy procedure located in a stub appended to the client at compile time.
- The client stub re-routes the call to a SOAP subsystem (which can be an independent module or a part of the stub) that will transform the call into an XML document formatted according to the SOAP message specifications.



Simple implementation of SOAP

Simple implementation of SOAP

- It will also wrap the resulting SOAP message into HTTP format. Once this is done, the HTTP document is passed to an HTTP module (which again can be independent, e.g. a Web server, or a part of the stub) that will forward the request to the remote location.
- At the remote location this procedure is reversed. An HTTP module receives the request and passes it along to a SOAP subsystem. There, the HTTP wrap is removed and the XML document is extracted and analyzed to retrieve its contents.

Simple implementation of SOAP

- These contents are then forwarded to the server stub that will call the corresponding procedure. The return parameters of the call are treated in a similar manner.
- Again the SOAP engine and the stub can be combined or they can be independent modules. If the SOAP related part is independent, then it is often referred to as SOAP router, since one of its tasks is to route the call to the appropriate object.

Simple implementation of SOAP

- The implementations may differ.
- The main differences across implementations are in the specific instances of middleware being adopted, in whether the SOAP system is an independent module provided by a Web services middleware infrastructure or attached to the client and server as a library (dynamic or static), and in whether the HTTP support is provided by a library or a complete Web or application server.

Simple implementation of SOAP

- Simplicity if one of the main goals of SOAP.
- Web services appeared as a way to solve the problem of connecting conventional middleware platforms across the Internet.
- Conventional middleware is strongly tied to RPC and it is therefore only natural for SOAP to build upon this tradition.
- Implementation that can be easily appended to the top of existing middleware platforms to make them Web service enabled.
- One of the first implementations of Web services appeared as part of databases, as databases support stored procedures as a way to hide queries behind RPC-like interface. Converting these interfaces into Web services is as simple as adding the layers shown in the previous figure.

Asynchronous SOAP

- RPC results in a integration that makes components dependent on each other.
- B2B interactions use document-style interactions, typically in asynchronous or batch mode.
- Use of EDI and SWIFT for B2B systems.
- Use of SMTP to exchange SOAP messages.
- Another way of achieving asynchronous SOAP interactions involves setting separate threads, one with main logic and another in charge of making each SOAP call.
- The main thread can make the request by passing the data to the second thread and then continue processing.
- The second thread places call and waits for the response.

Asynchronous SOAP

- When the response arrives, a callback occurs and the main thread is notified of the arrival of the response.
- Alternatively, the response can be placed in a buffer until the main thread decides to act upon it.
- There are no technical difficulties involved as the mechanisms are identical to those used for implementing asynchronous RPC.
- Many MOM platforms are implemented on top of RPC.
- A message is placed in a queue and a daemon makes an RPC call to another remote daemon, which takes the message and places it on the receiving queue, SOAP can be used in exactly the same way.

Binary Data and SOAP

- XML is a blessing as a syntax standard.
- It allows one to build generic parsers that can be used in a multitude of applications, thereby ensuring robustness and low cost for the technology.
- The downside is that the data transformations to and from XML as well as the associated XML parsing can result in a significant performance overhead, a typical price paid for generality.

Binary Data and SOAP

- XML does not always gracefully support the necessary data types, such as binary data (e.g., an image) or nested XML documents.
- In many cases, even if it is possible, there may be no reason to format application data into an XML application.
- XML encoding makes sense then linking completely heterogeneous systems or parsing around data that cannot be immediately interpreted.
- Web services are built based on already agreed upon data formats, then the role of XML is reduced to being the syntax of the SOAP messages involved. There is a strong demand for SOAP to support a binary or blob type.

Binary Data and SOAP

- There are several ways of doing this:
- Using URLs as pointers
- As an attachment, or
- With the recently proposed Direct Internet Message Encapsulation (DIME) protocol.
- Considerable amount of Web services is likely to have a traffic to contain binary data rather then explicit XML.