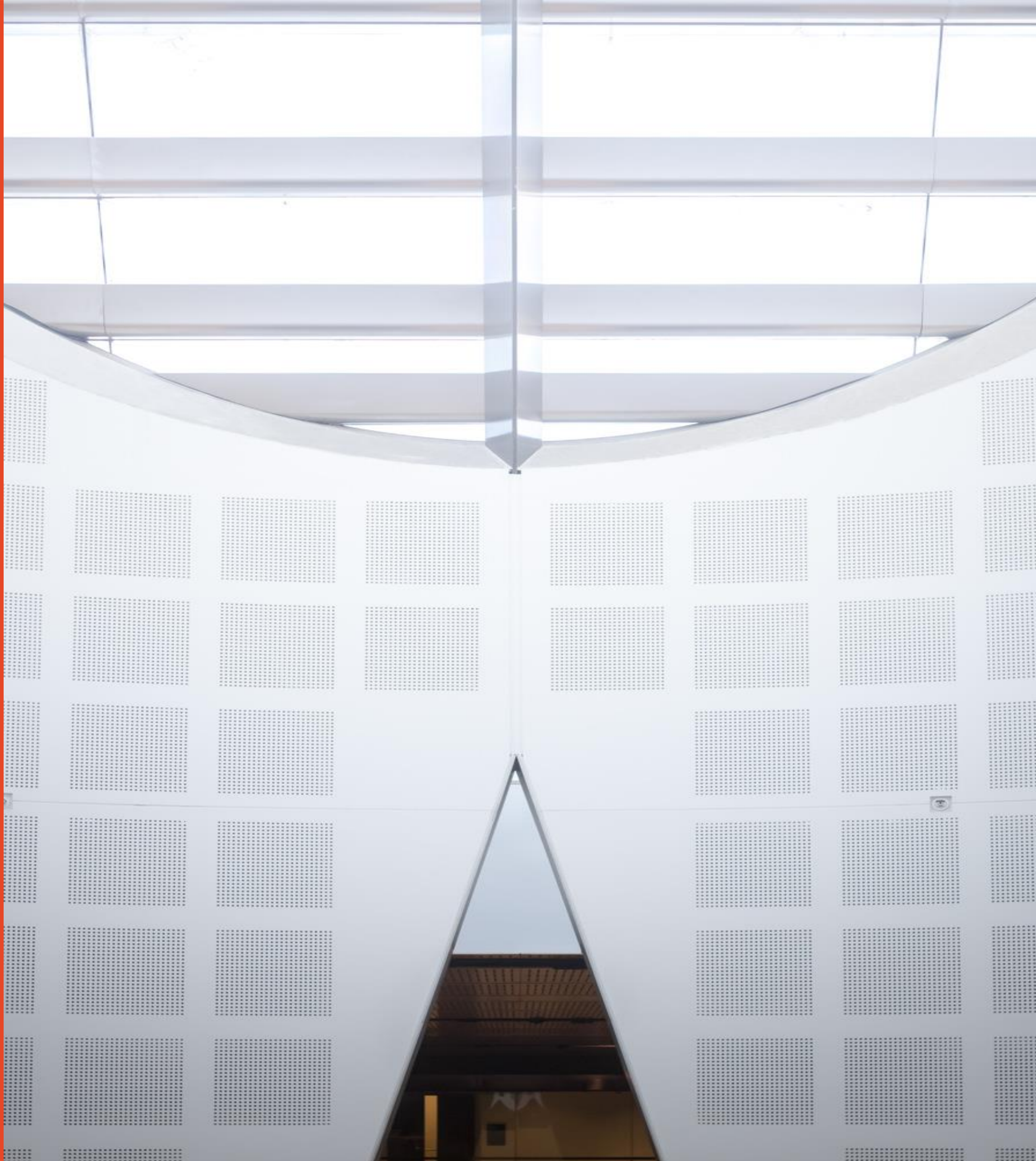
COMP5347 Web Application Development – Advanced Topics
Web Services Policies

Dr. Basem Suleiman
School of Computer Science

THE UNIVERSITY OF SYDNEY



Outline

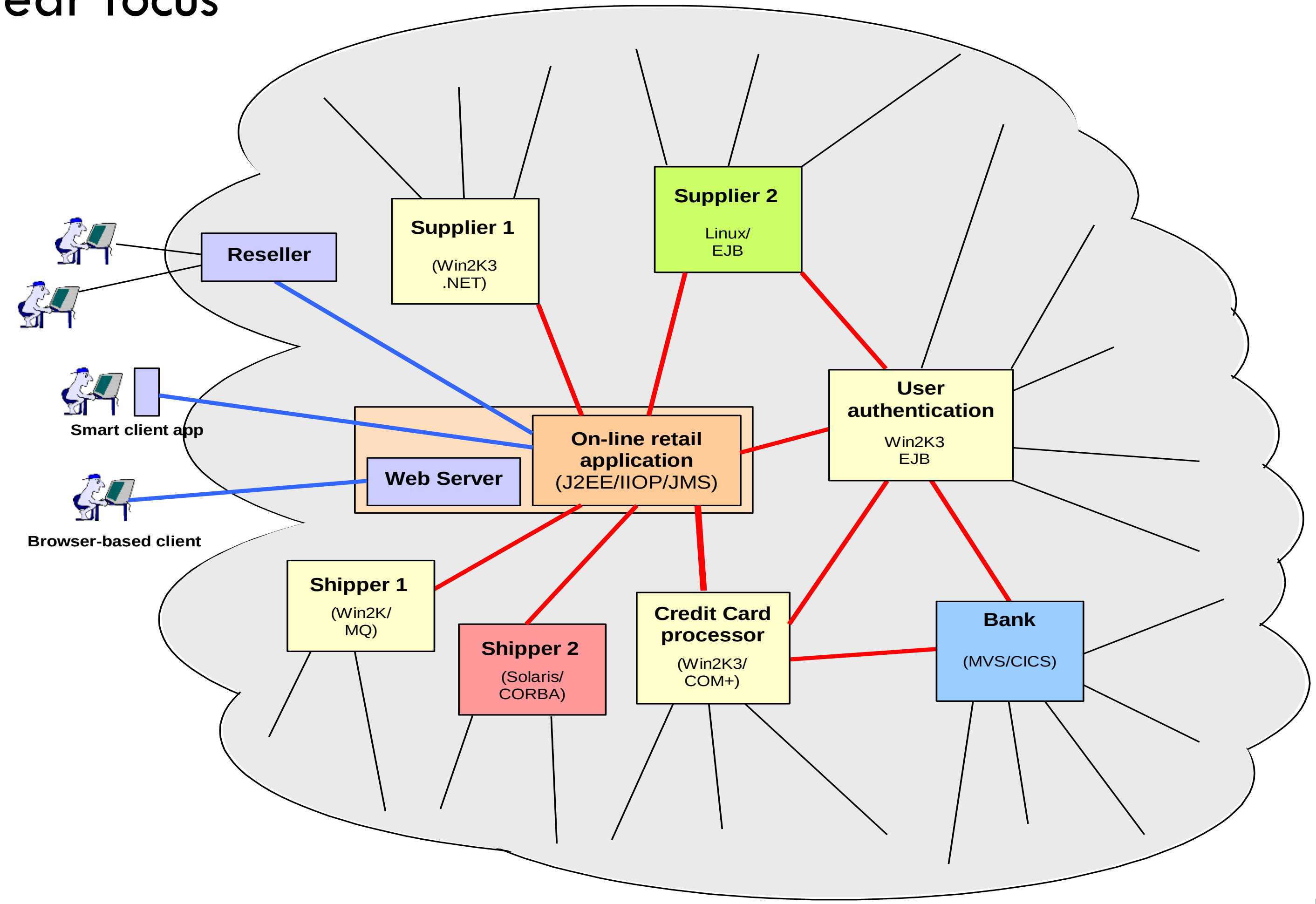
- Service Oriented Architectures
- Web Services
  - Web Services Standards
    - SOAP, UDDI, WSDL
  - Web Service Policy
    - Security
    - Reliability
    - Distributed Transactions

Service-Oriented Architecture – SOA (Revisit)

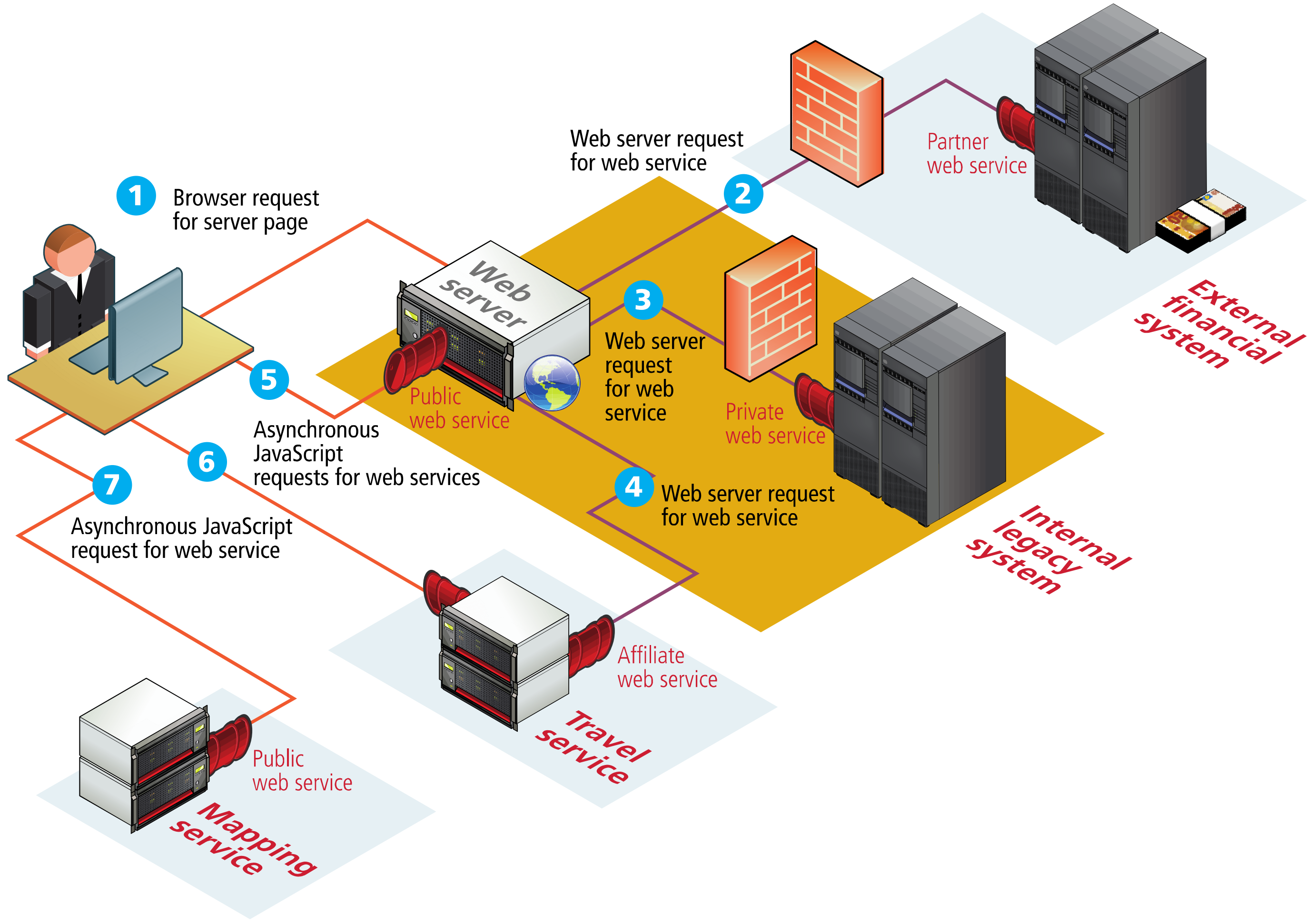
- What are Service-Oriented Architectures?
  - Architectures for a service-based world
  - Design principles
  - Patterns
- Applications provide services
- Applications consume services
- Applications share standards, contracts, schemas
- Applications share nothing else
  - Not implementation technologies, data, classes, …

SOA – Web Applications

- Building large-scale integrated applications from services
  - Specific enough to have clear focus
- Integration and communication
- Platform independence
- Abstracted from end users
- Avoid rewriting legacy code



A Service-based Application


1 Browser request for server page
Web server request for web service
2
3
Web server request for web service
Partner web service
Web server
Public web service
Private web service
External financial system
5
Asynchronous JavaScript requests for web services
6
4 Web server request for web service
Internal legacy system
7
Asynchronous JavaScript request for web service
Affiliate web service
Travel service
Public web service
Mapping service

SOA Principles (Revisit)

- Building large-scale integrated applications from services
  - Coarse-grained, rich services
    - But specialised enough to have clear focus
  - Think about reliability and robustness
  - Reduce dependencies
  - Avoid rewriting legacy code
  - Keep everything as simple as possible
    - Including integration technologies

SOA Tenets

- Boundaries are explicit
  - Services are external, crossing boundaries has costs and implications
- Services are autonomous
  - May be outside your control, can change
- Share schema and contract
  - Not classes and implementations
- Policy-based compatibility
  - Requirements stated in policies, not in code

Outline

- Service Oriented Architectures
- Web Services
  - Web Services Standards
    - SOAP, UDDI, WSDL
  - Web Service Policy
    - Security
    - Reliability
    - Distributed Transactions

Web Services (Revisit)

- Relatively standardized mechanism by which one piece of software application can connect and communicate with another using web protocols
- It provides a simple and open way of **integrating** functions or data from various systems
- It can be used **within** an organization and/or **across** the public Internet
- At least two implementations: SOAP-based vs. RESTful services
- Original design of Web Services is very **application-centric** in contrast to the **resource-centric** Web and REST style

Web Service

- A component that offers services that can be accessed through messages that follow some particular XML-based standards
  - XML messages in the protocol
  - XML definitions of services
  - Pass around XML documents
  - Building on other XML standards
  - All ‘human readable’ (for some subset of ‘humans’)
- These messages may be carried over HTTP…
  - And possibly not as well
- Standards are managed through W3C
  - Standardise only what goes on the wire
  - No programming models or APIs

Web Service – Common Features

- Many different definitions (wikipedia, standards bodies, textbooks)
- Common features:
  - Coarse-grained chunks of meaningful business functionality provided by IT
  - Service abstracts away implementation details
  - Large-scale applications then built by integrating services
  - Components on an Internet scale

Terminology

- Sender
- Receiver
- Intermediary
  - A receiver that then sends the message onwards
- Ultimate Receiver
  - The receiver that actually provides the service
  - Does not send the message on any further

Web Service Standards (Revisit)

- Simple Object Access Protocol (SOAP): a messaging protocol for transferring information (XML format)

- Web Service Description Language (WSDL): A standard for describing Web services (XML format)
  - Interfaces, methods, parameters, request, response
  - Generated and consumed by tools

- Universal Description, Discovery and Integration (UDDI): a registry and protocol for publishing and discovering web services
  - Like white, Yellow and Green ‘pages’
  - Not really used

- Also, WS-Policy framework for non-functional properties (e.g., security, reliability)

WS Standards

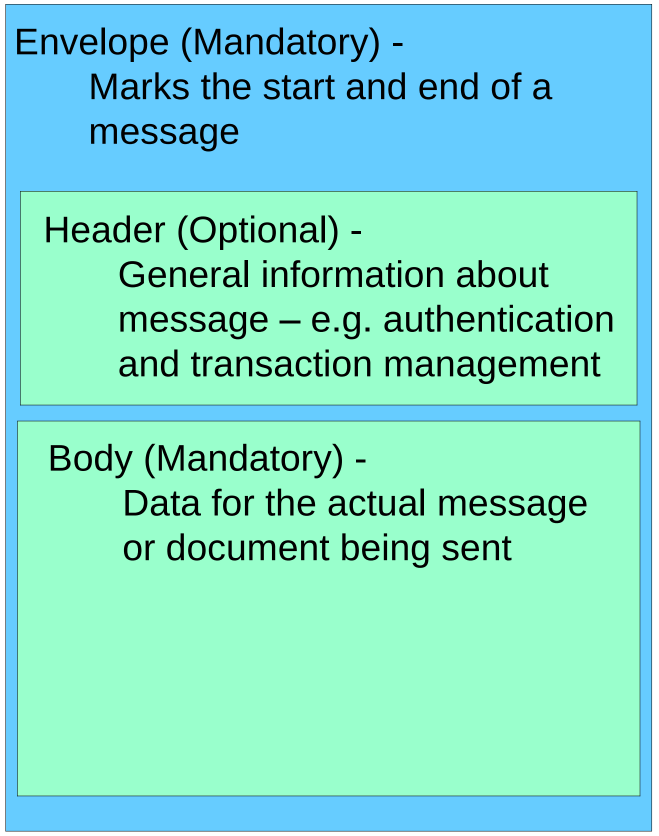
- Full Distributed Computing platform
- Architecture is metadata-driven, policy-based
- Architecture allows intermediaries
- Architecture allows extension
  - Additional aspects (eg security) should not inconvenience legacy code
- Uniform data model for protocols and content
  - All defined in XML using schemas
  - Attributes in extensible SOAP header

Calling Things – Messaging

- SOAP – Simple Object Access Protocol
  - Originally simple remote procedure calls using XML-formatted protocol
    - Can pass messages as well as RPC
  - Envelope, header, body
  - Extensible protocol
    - Add new header elements for new semantics
  - Very widely accepted standard
    - Up to SOAP 1.2 now

Calling A Service – SOAP Message Structure

- SOAP – Simple Object Access Protocol!
  - Originally simple remote procedure calls using XML-formatted protocol
  - Envelope, header, body
  - Extensible protocol
    - Add new header elements for new semantics
  - Very widely accepted standard


Envelope (Mandatory) - Marks the start and end of a message
Header (Optional) - General information about message – e.g. authentication and transaction management
Body (Mandatory) - Data for the actual message or document being sent

SOAP Example – Stock Quote

```
GET /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml
Content-Length: nnnn
SOAPMethodName: Stock-Namespace-URI#GetLastTradePrice
<SOAP:Envelope xmlns:SOAP="urn:schemas-xmlsoap-org:soap.v1">
    <SOAP:Body>
        <m:GetLastTradePrice xmlns:m="Stock-Namespace-URI">
                <symbol>DIS</symbol>
        </m:GetLastTradePrice>
    </SOAP:Body>
</SOAP:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: nnnn
<SOAP:Envelope xmlns:SOAP="urn:schemas-xmlsoap-org:soap.v1">
    <SOAP:Body>
        <m:GetLastTradePriceResponse xmlns:m="Stock-NS-URI">
            <return>34.5</return>
        </m:GetLastTradePriceResponse>
    </SOAP:Body>
</SOAP:Envelope>
```

Encodings for SOAP

- Two common encoding styles
  - Doc/Literal
    - Use particular schema that is defined for this message type
    - Place XML in SOAP body
    - Allows for common xml validators
    - Better for interoperability
    - XML becomes more complex
  - RPC/encoded
    - Now deprecated

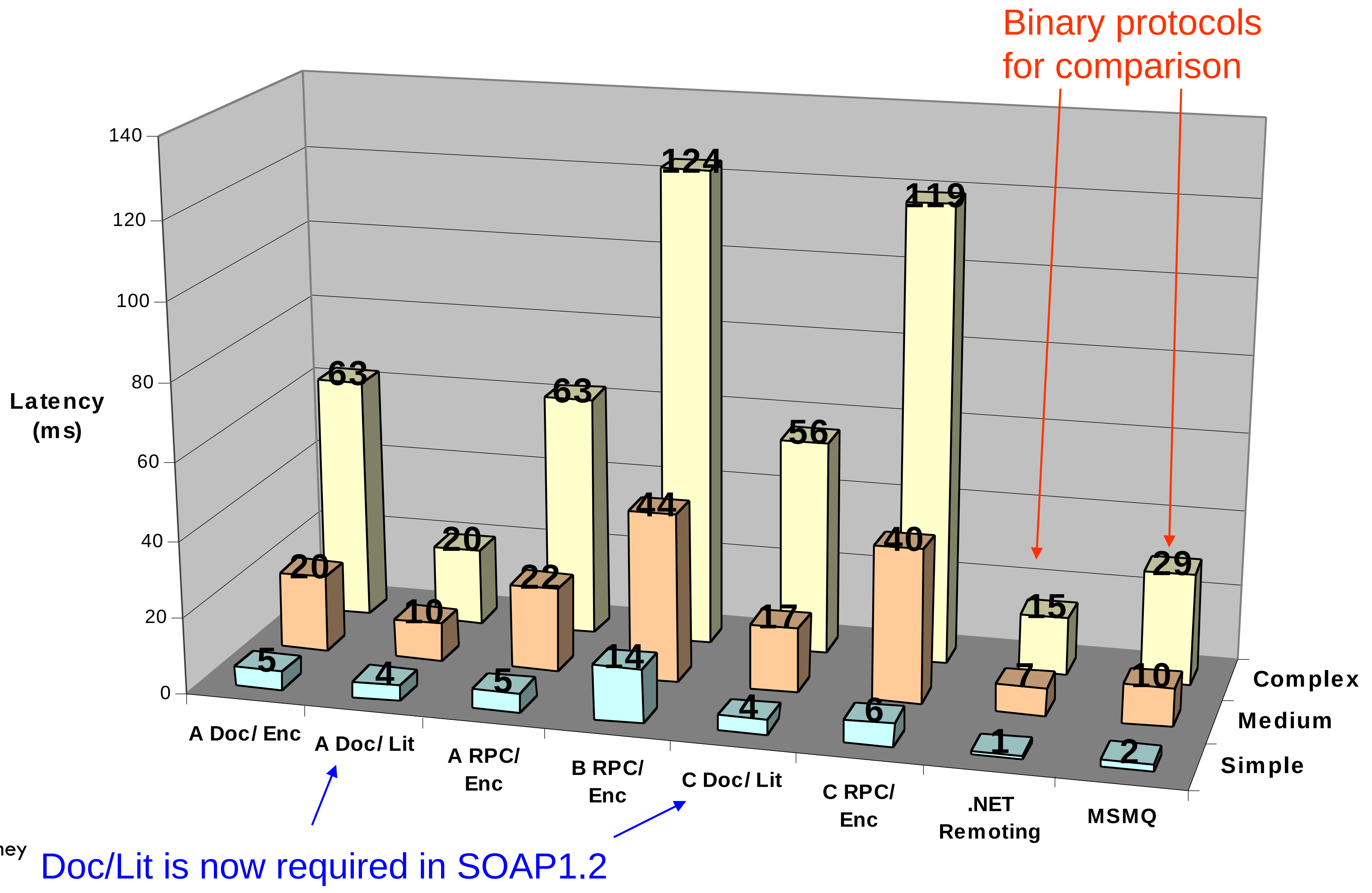
Performance Factors

- Processor time for XML encoding/decoding
  - Is XML slower than binary alternatives?
- Number and size of msgs passed around
  - Does verbosity cost performance?
- Proc time taken by transport protocols
  - TCP/IP and HTTP
- Network delays
  - Switches, routers, …
- Speed of light delays
  - Caused by synchronous message exchanges
  - 1.5mS Sydney-Canberra in glass

Are All SOAPs Equal?

- In 2005 CSIRO took 3 commercial SOAP products
- Compare to 2 non-SOAP alternatives
- Effects of different encoding styles?
- Measured single-thread call latency…
  - Client call to response received
  - Small, medium and complex messages
    - 1.5KB, 7.5KB, 13KB (call & response)
  - 4x Xeon servers, 100M LAN
- Measure factors in processor usage
  - XML encoding/decoding, HTTP, …

Are All SOAPs Equal?



	A Doc/ Enc	A Doc/ Lit	A RPC/ Enc	B RPC/ Enc	C Doc/ Lit	C RPC/ Enc	.NET Remoting	MSMQ
Complex	63	20	63	124	56	119	15	29
Medium	20	10	22	44	17	40	7	10
Simple	5	4	5	14	4	6	1	2

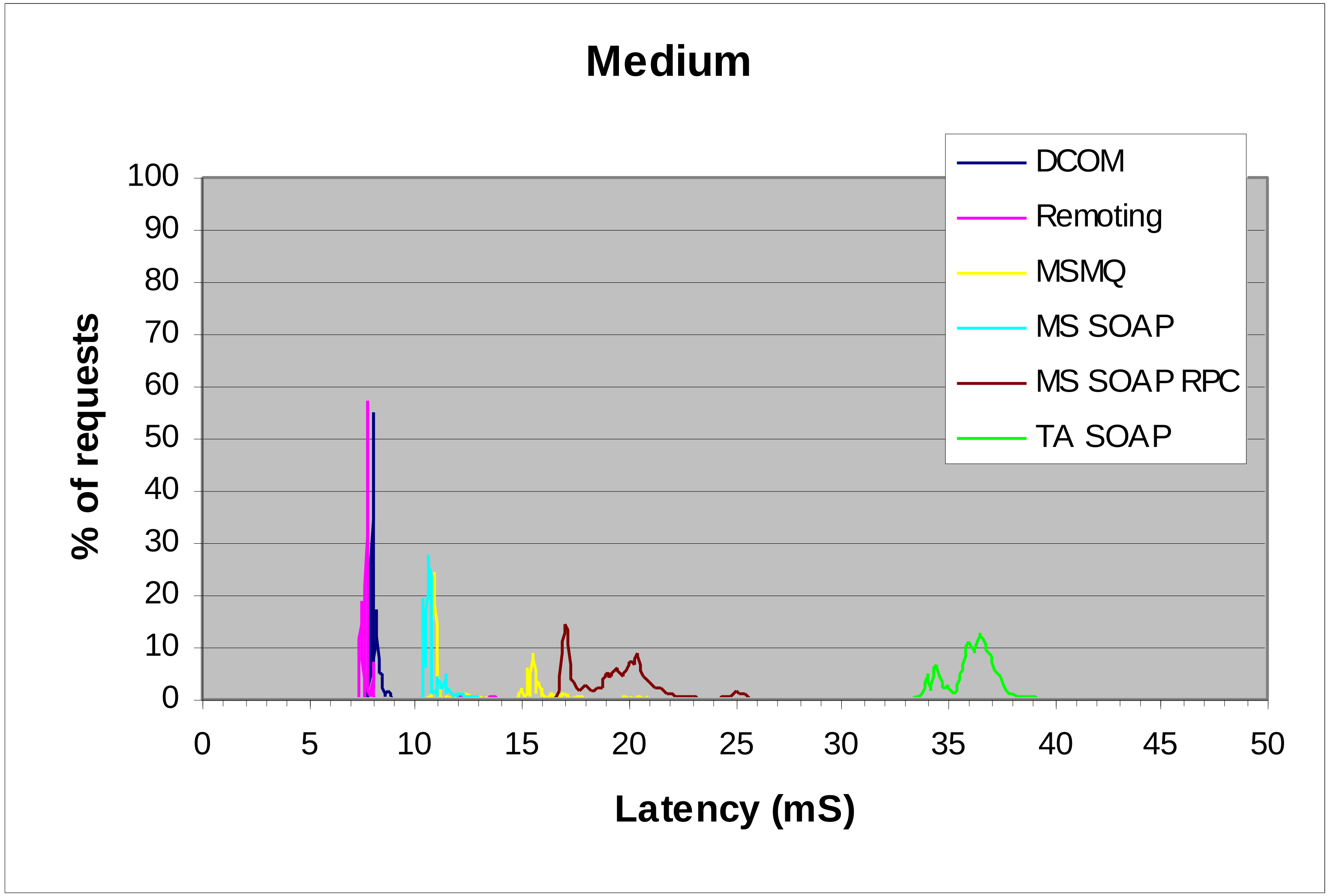
SOAP Findings

- Best implementations competitive with non-SOAP alternatives
  - 4-20 mS vs. 1-15mS
- Encoding style very important
  - ‘doc/lit’ style much faster
- Considerable differences between products
  - Factors of 4-6 times in performance
- What about WAN performance?

SOAP Over WAN

- Sydney-Canberra over CeNTIE network
  - 100M locally, 1G for rest of network
  - Lightly-loaded, low contention network
  - More realistic than you'd think…
- Single-threaded clients
  - 2.8GHz Dell desktop systems
  - Windows 2003 Server, .NET client
- Initial tests
  - SOAP (Microsoft & Apache)
  - Alternatives (MS DCOM, .Net remoting)
  - Very simple call/return + app call

Latency measurements



Scalability Lessons

- XML message size may cause…
  - Latency problems on slower networks
  - Higher network loads
    - And overheads increase with new features such as security and reliability
  - Be aware of impact of encoding style

- Processor utilisation
  - Best implementations comparable to binary alternatives
    - And others have room to improve…

WS-Addressing

- Addresses of services
  - Basically just extended URIs
  - Addresses shipped within messages
- Addresses are <u>transport-neutral</u>
  - Not just HTTP
- No constraints on addresses
  - Services can be constructed, partitioned, named, addressed in arbitrary fashion

WSA Endpoint Reference

```
<a:EndpointReference
  xmlns:a='http://schemas.xmlsoap.org/ws/2004/08/addressing'
  xmlns:m='http://example.net/ws/weather/forecasts'
  xmlns:p='http://schemas.xmlsoap.org/ws/2002/12/policy' >
  <a:Address>http://example.org/weather/us</a:Address>
  <a:ReferenceProperties>
   <m:Info>Services.Weather.Wind</m:Info>
  </a:ReferenceProperties>
  <p:Policy>
  . . .
  </p:Policy>
<a:/EndpointReference>
```

Network address of resource

Policy for this resource

Other addressing information

Finding Things

- UDDI - a directory of services
  - Universal Description, Discovery and Integration
  - A Web Service in its own right!
  - White, Yellow and Green ‘pages’
- Vision of a single universal directory?
  - Reality is probably many directories
  - Accepted in theory but has never taken off in practice
  - Most systems use other repositories

UDDI Pages

- White pages
  - General information; business name, contact details and addresses
- Yellow Pages
  - Information about business services specific to a particular industry
  - UDDI taxonomies (e.g., NAICS-1997 and UNSPSC-7.03)
- Green Pages
  - Technical details about services
    - Service address (location)
    - How to call it
    - tModel to describe services

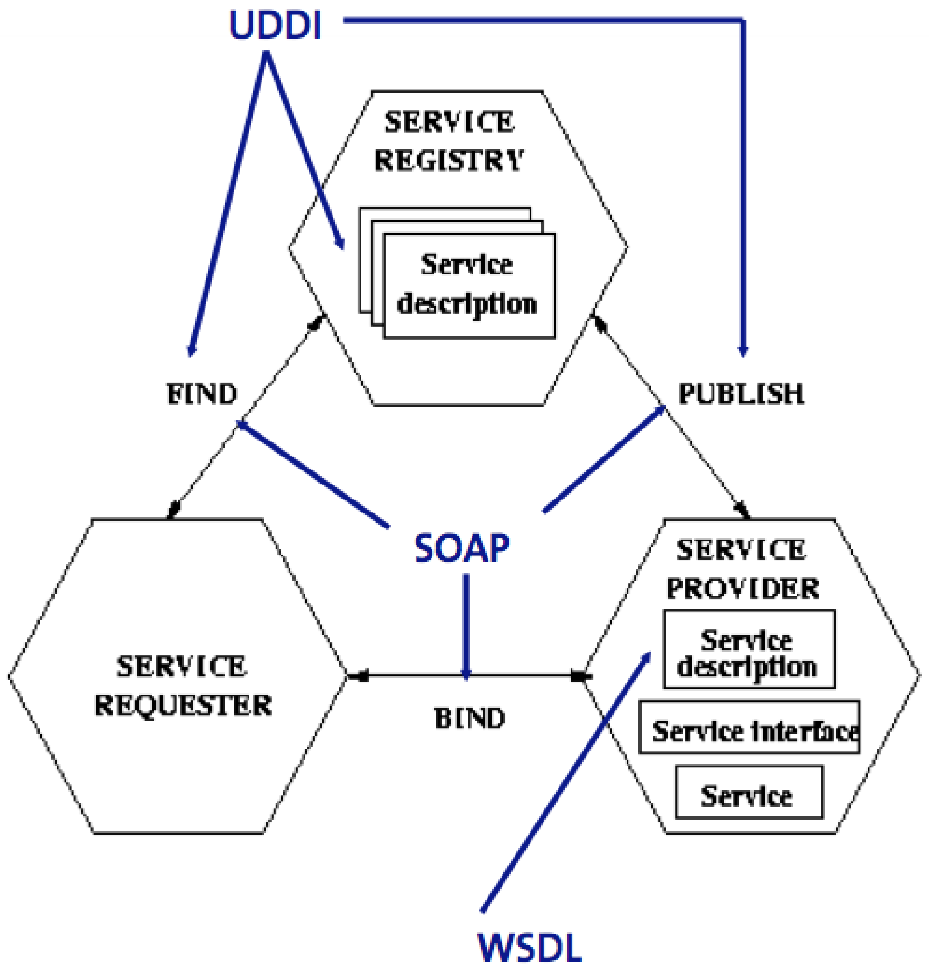
Finding About Things

- WSDL - Web Services Description Language
- XML description of a service
  - Interfaces, methods, parameters
  - Request, response
  - How to bind to the service
  - Widely accepted standard
  - Suffering from the problems faced by all ‘standards’
- Generated and consumed by tools
  - Microsoft Visual Studio, IBM WebSphere, Sun’s Java, …

WSDL Structure

```
<definitions>
 <types>
  definition of types........
 </types>

<message>
  definition of a message....
 </message>

 <portType>
  definition of a port.......
 </portType>

 <binding>
  definition of a binding....
 </binding>

</definitions>
```

The Role of WSDL and UDDI



[Alonso2010]

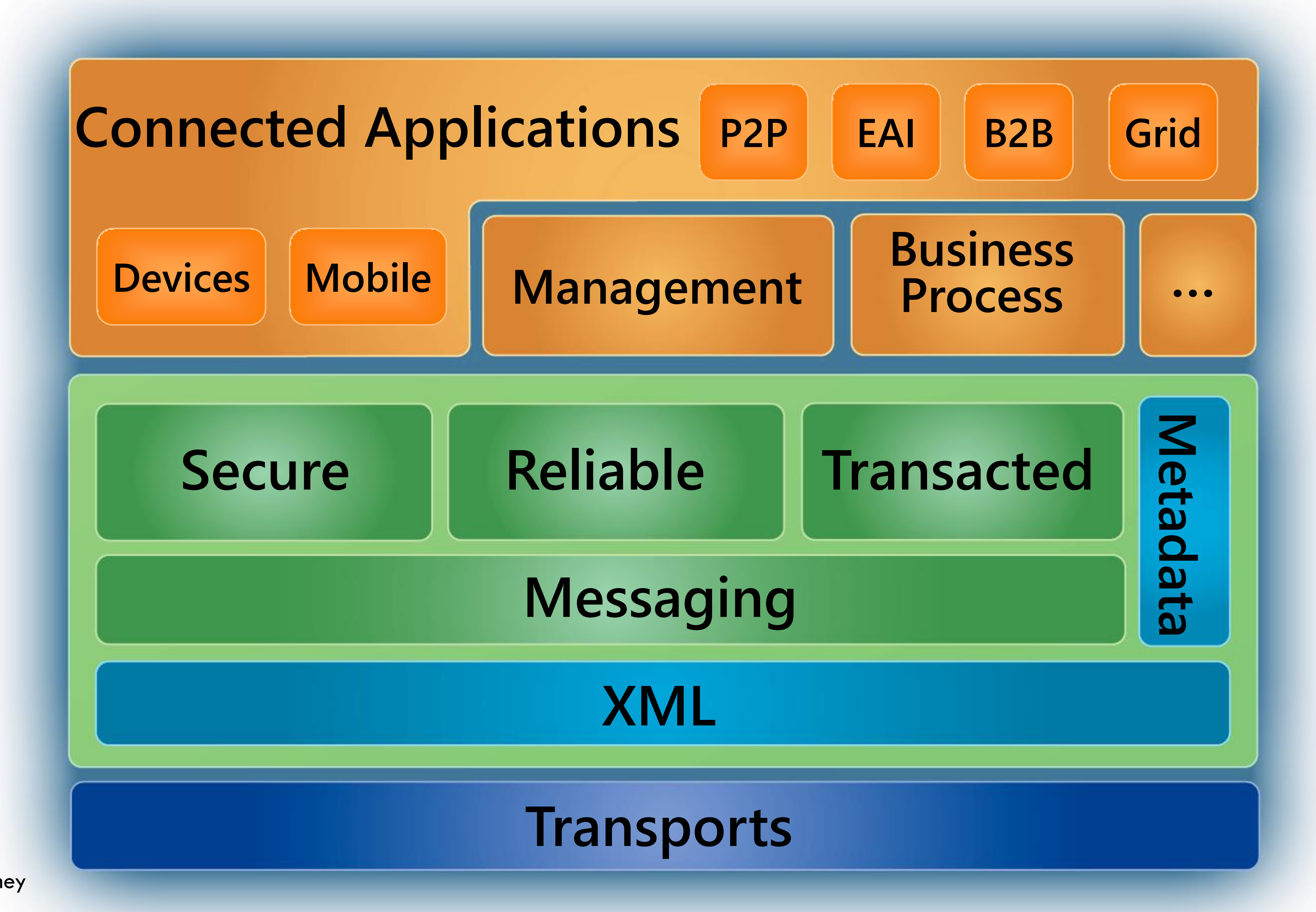
Limits of WSDL

- WSDL is focused on request-response interactions
  - An IDL for WS
  - It does not specify in which order to send msgs!
  - Can theoretically augment with BPEL for conversations
  - In practice tool support is limited, approach is verbose and complex

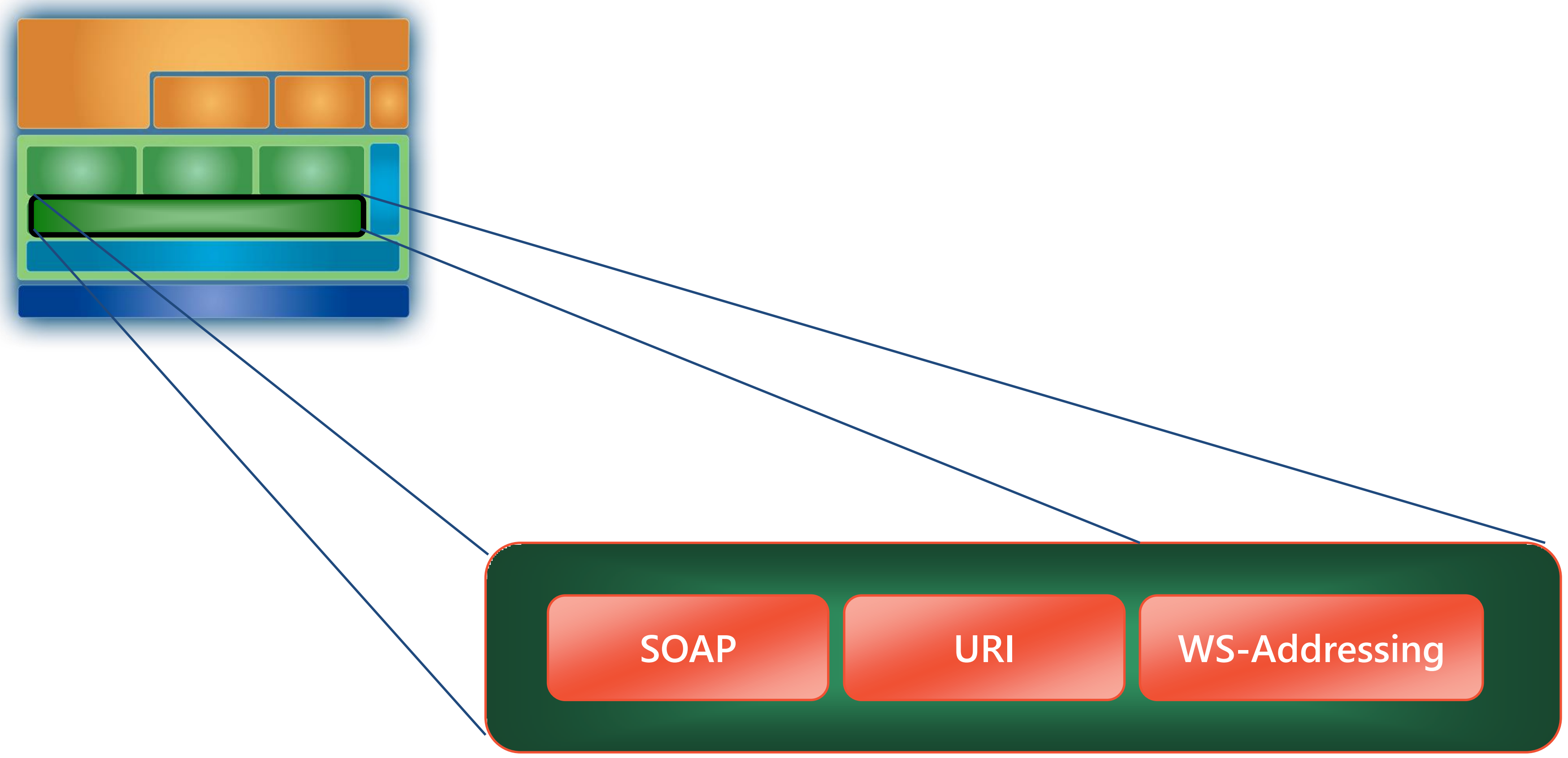
Summary

- Core WS standards give the essential support for distributed computing
  - Just like many other standards
  - But these are widely supported
  - And they have good tool support
    - Automated generation of WSDL from a component, or vice versa
    - Automated generation of stubs from WSDL
  - Good design for extensibility
  - The messages are verbose (performance needs attention),
  - Messages are not very human-friendly
    - But they don’t need to be!

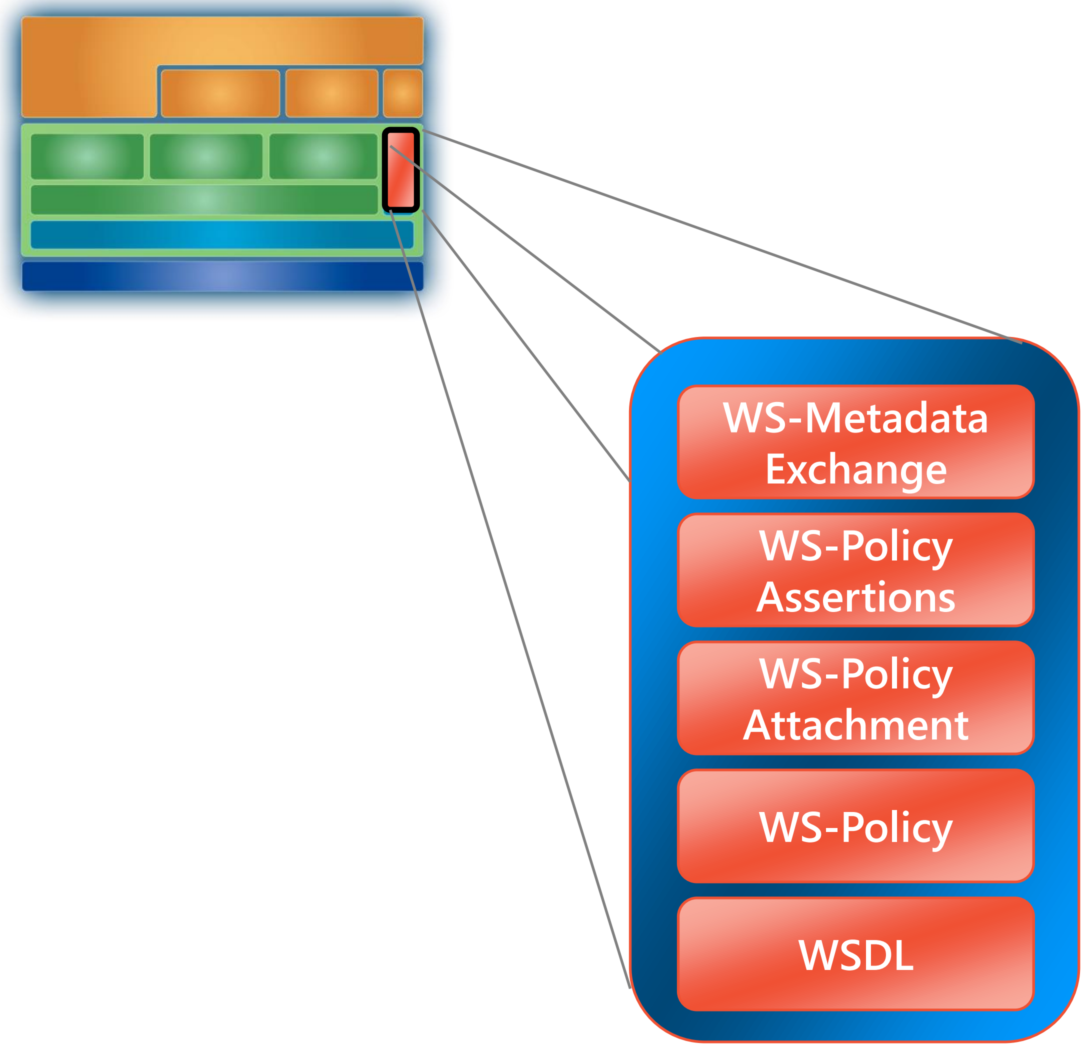
Web Services Stack



Messaging



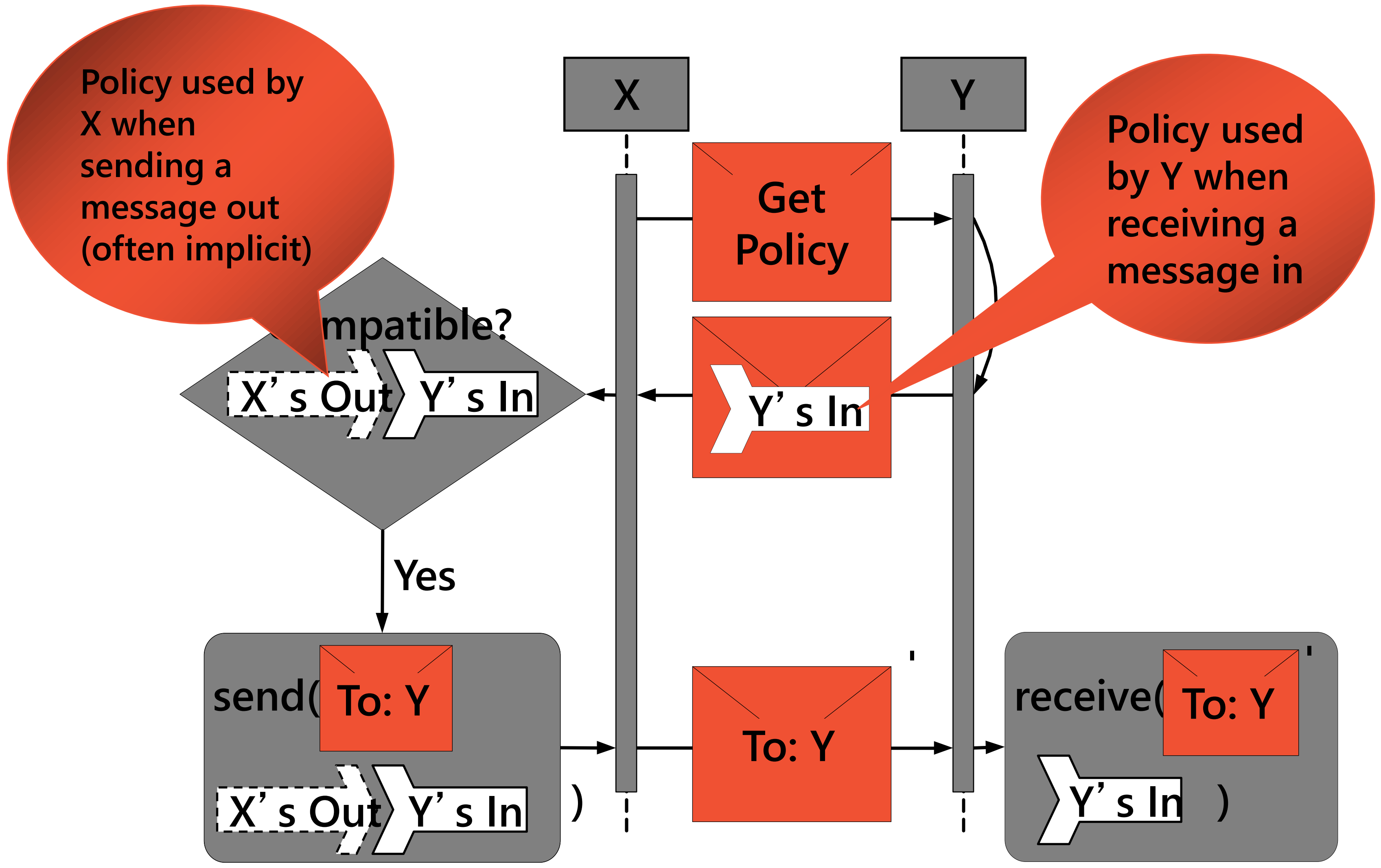
Metadata


WS-Metadata Exchange
WS-Policy Assertions
WS-Policy Attachment
WS-Policy
WSDL

Policy

- WS-Policy: A framework for making statements about resource capabilities & requirements
  - Used to express receiver requirements for incoming messages (e.g., transports, security)
  - Can be used to match requirements to capabilities at runtime
- WS-PolicyAssertions: Some predefined basics
- WS-PolicyAttachment: Attaching policy expression to a subject
  - WSDL, UDDI, whatever
- WS-MetadataExchange: Discovery and retrieval of metadata

Metadata Driven Architecture

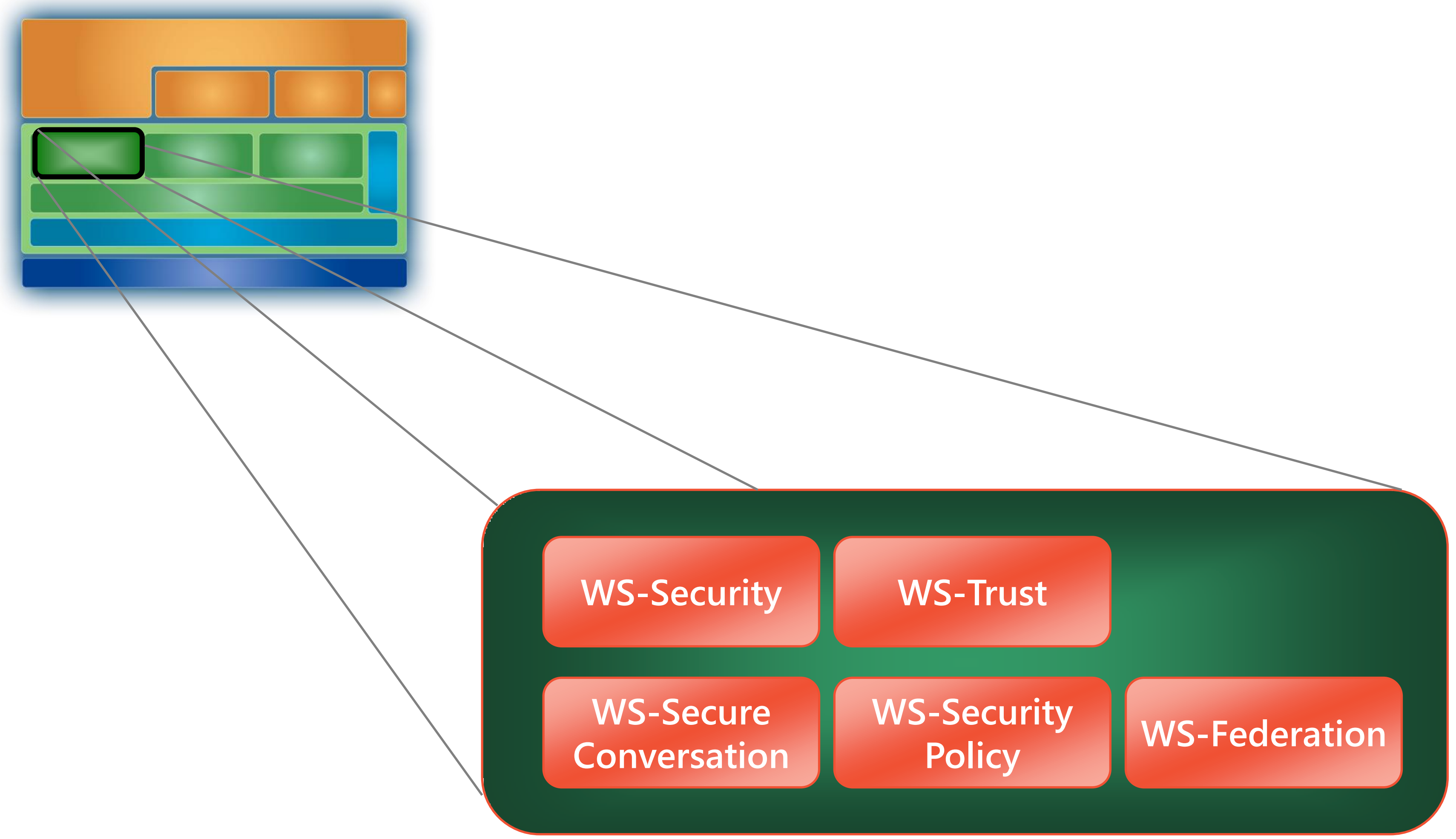

Policy used by X when sending a message out (often implicit)
X
Y
Get Policy
Policy used by Y when receiving a message in
mpatible?
X's Out
Y's In
Y's In
Yes
send(To: Y
X's Out Y's In)
To: Y
receive(To: Y
Y's In)

WS-Policy Example

```
<wsp:Policy>
   <wsp:ExactlyOne>
      <wsse:SecurityToken>
         <wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
      </wsse:SecurityToken>
      <wsse:SecurityToken>
         <wsse:TokenType>wsse:X509v3</wsse:TokenType>
      </wsse:SecurityToken>
   </wsp:ExactlyOne>
   <wssx:Audit wsp:Optional="true" />
</wsp:Policy>
```

- ➔ Must use either Kerberos or X.509 security tokens
- ➔ Can use Auditing as well

Security


WS-Security
WS-Trust
WS-Secure Conversation
WS-Security Policy
WS-Federation

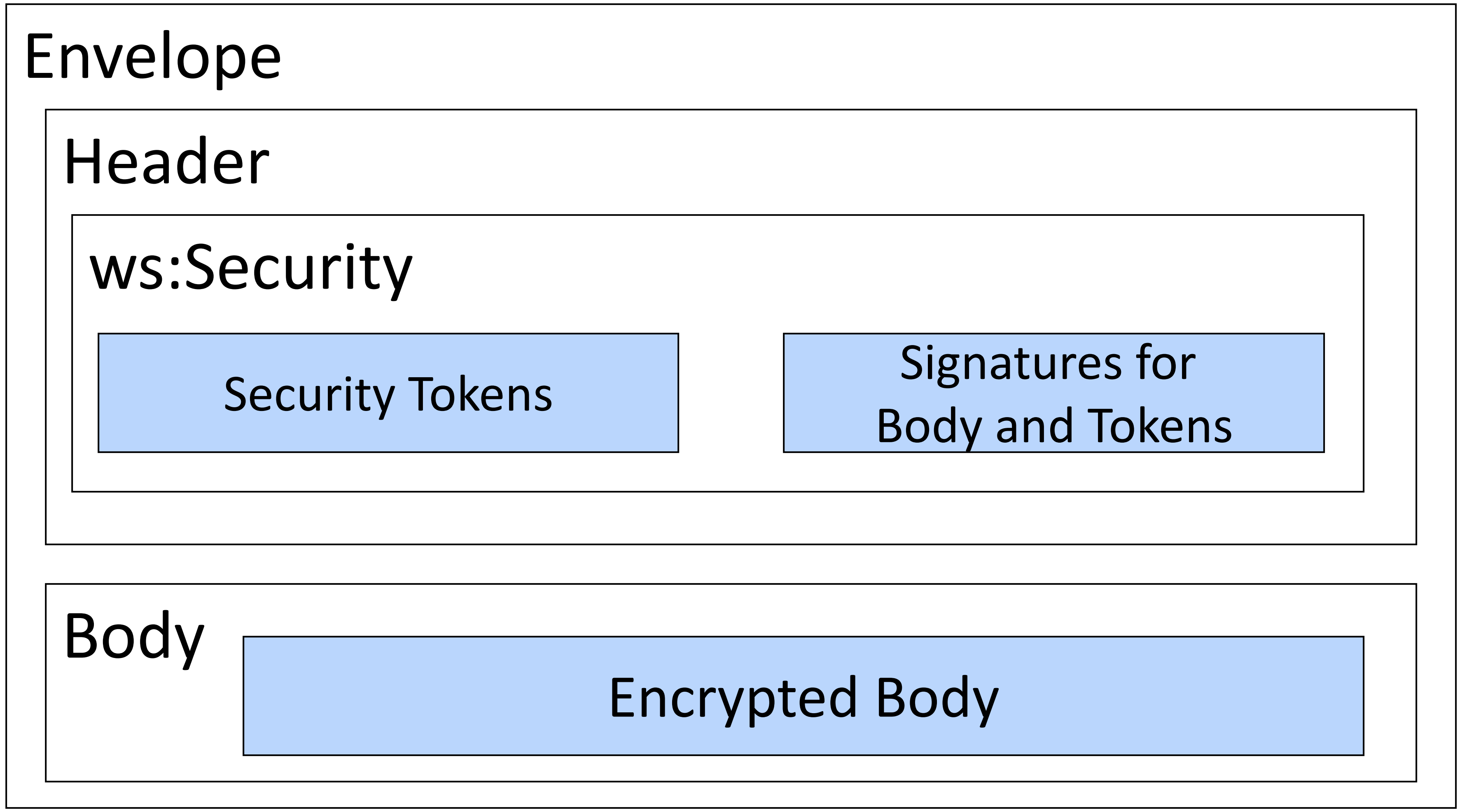
WS-Security

- Framework for using security protocols
  - applies XML security technologies (XML encryption and XML signature) to implement secure SOAP message exchange across multiple trust domains
- Goals:
  - security (integrity; confidentiality) at the message level
  - End-to-end security:
    From <u>initial sender</u>, through 0-n intermediaries to <u>final receiver</u>
- Solution:
  - encryption and signatures within a SOAP message:
    parts of the message body can be encrypted, signatures are stored in the header; propagation of security tokens
  - Support for pluggable algorithms
    (Encryption, Digest, Signature, Canonicalisation, Transforms)

Protecting SOAP Messages

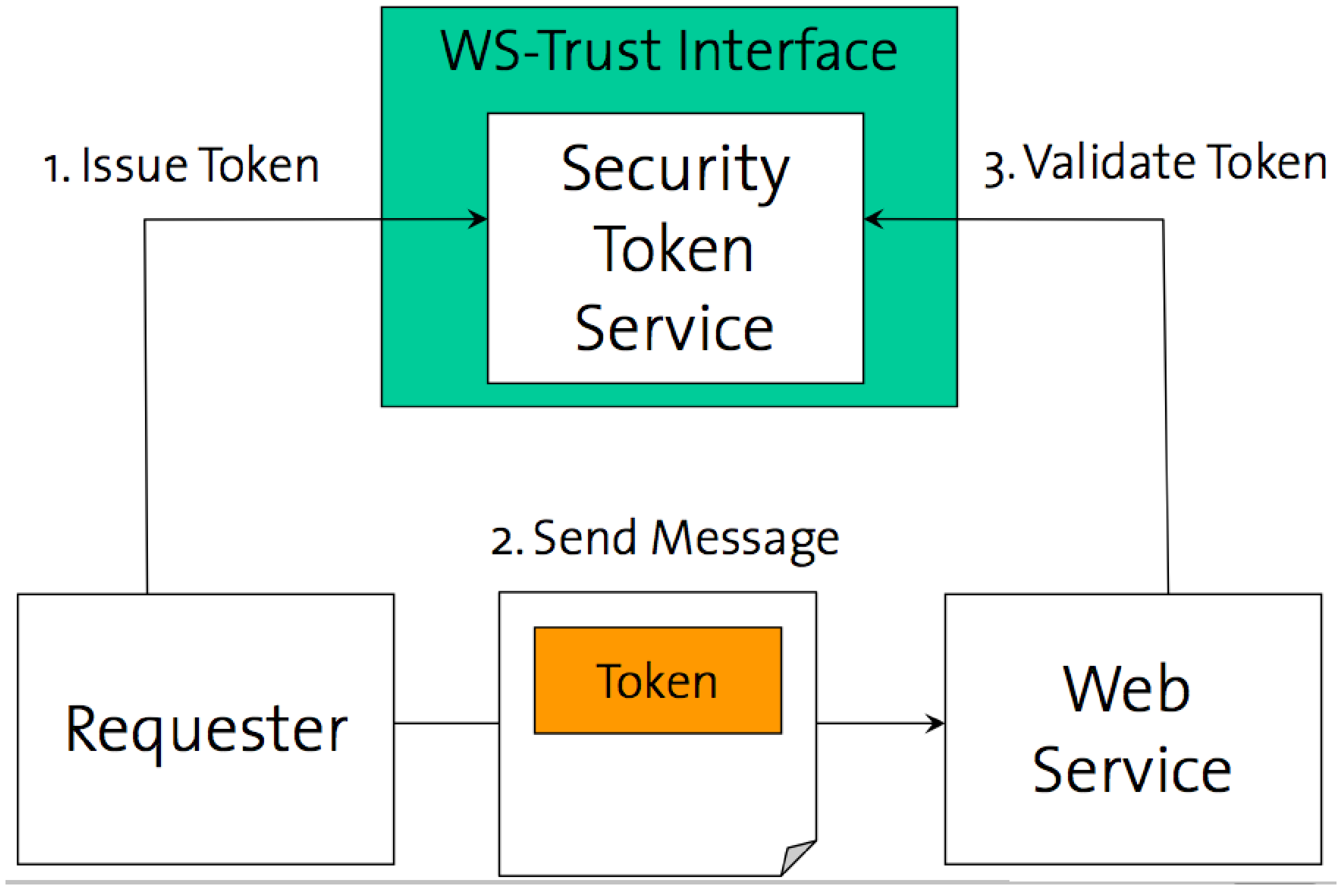
- Security Threats to a SOAP message:
  - A message could be read by an attacker
  - A message could be modified by an attacker
  - A message could be sent by an attacker
- To address these threats, WS-Security applies a combination of:
  1. Encryption (Ensure the confidentiality of the message)
  2. Signatures (Verify the origin and the integrity of a message)
  3. Security Tokens (Authorize the processing of the message based on the credentials associated with the message)
- Messages with invalid signatures and incorrect or missing tokens are rejected.

A Secure SOAP Message



Overview

WS-Security supports a variety of authentication and authorization Mechanisms, from simple username/pw to X.509 certificates etc.



WS-Security Policy

- A set of policy assertions related to other WS-Sec* specs

- Allows participants to specify
  - Authentication token types
  - Whether integrity and/or confidentiality are required
  - Algorithms for the above
  - Which message parts need signing or encrypting

Policy-based Security

- Take security out of the code and…
- Express in policies interpreted by frameworks
  - Declarative, administrative model
  - Policy says ‘signed’, ‘encrypted’
  - Signatures checked, message decrypted on the way to the application… and again on the way out
  - With no user code involved at all

Security!

- “SOAP is firewall friendly”
  - Normally transported over HTTP
  - Firewalls expect HTTP to be Web requests, not procedure calls
    - Fetch the brochure…
    - … not update my bank account…
  - Forcing good security
    - Tunnelling via http means we have to rely on strong security rather than weaker physical security.

Message Security vs. Transport Security

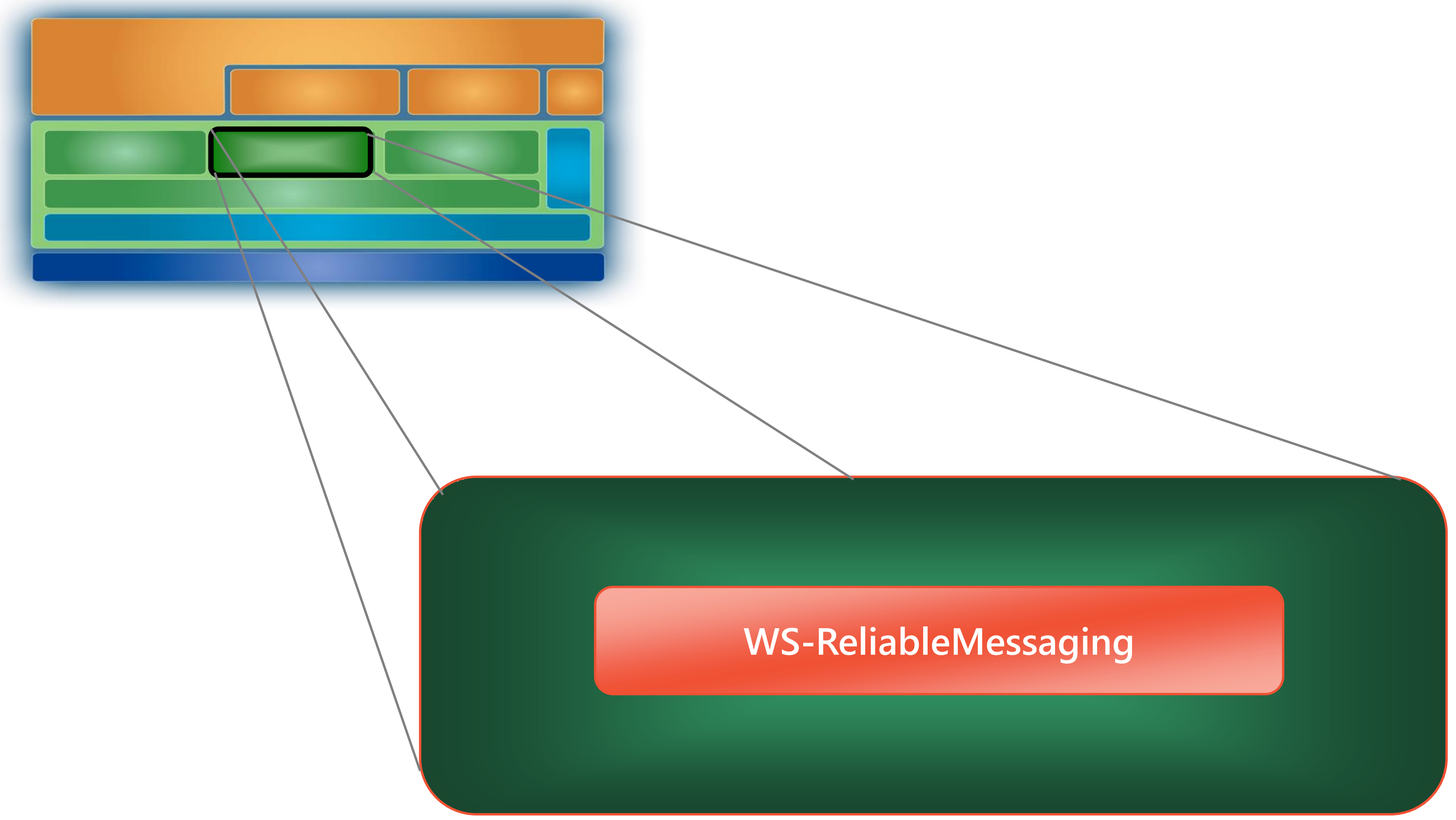
Message Security

- **Disadvantages**
  - Standards are only partially supported by existing tools
  - Securing XML is complicated
- **Advantages**
  - End-to-End: Message encrypted throughout the whole way
  - Asymmetric: different security mechanisms can be applied to request and response
  - Different parts of a message can be secured in different ways
  - Self-protecting message (transport independent)

Transport Security

- **Advantages**
  - Widely available, mature technologies (SSL,TLS,HTTPS)
  - Understood by most sys admins
- **Disadvantages**
  - Point-to-Point: The complete message is clear after each hop
  - Symmetric: Request and response messages must use same security prioperties
  - Transport specific

Reliability



https://docs.oasis-open.org/wsrm/ws-reliability/v1.1/wsrm-ws_reliability-1.1-spec-os.pdf

Reliability in Distributed Computing

The Eight Fallacies of Distributed Computing:

1. **The network is reliable**
2. Latency is zero
3. Bandwidth is infinite
4. The Network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

[Peter Deutsch, 1991]

– **As web services run on top of standard networks, those assumptions don't hold for web services either.**

Example: HTTP and Reliability

- Although HTTP is build on top of TCP, it is not a reliable message transport as compared to ‘real’ messaging queues such as JMS or MQ

- Problem:
  - The transfer of a SOAP message via HTTP may be interrupted at any time (client or server crash, network failure etc)
  - The application should not have to deal with these low-level transport failures
- Solution:
  - Let the WS stack deal with the problem and provide a reliable transport mechanism that guarantees the application that a message will be delivered with certain properties

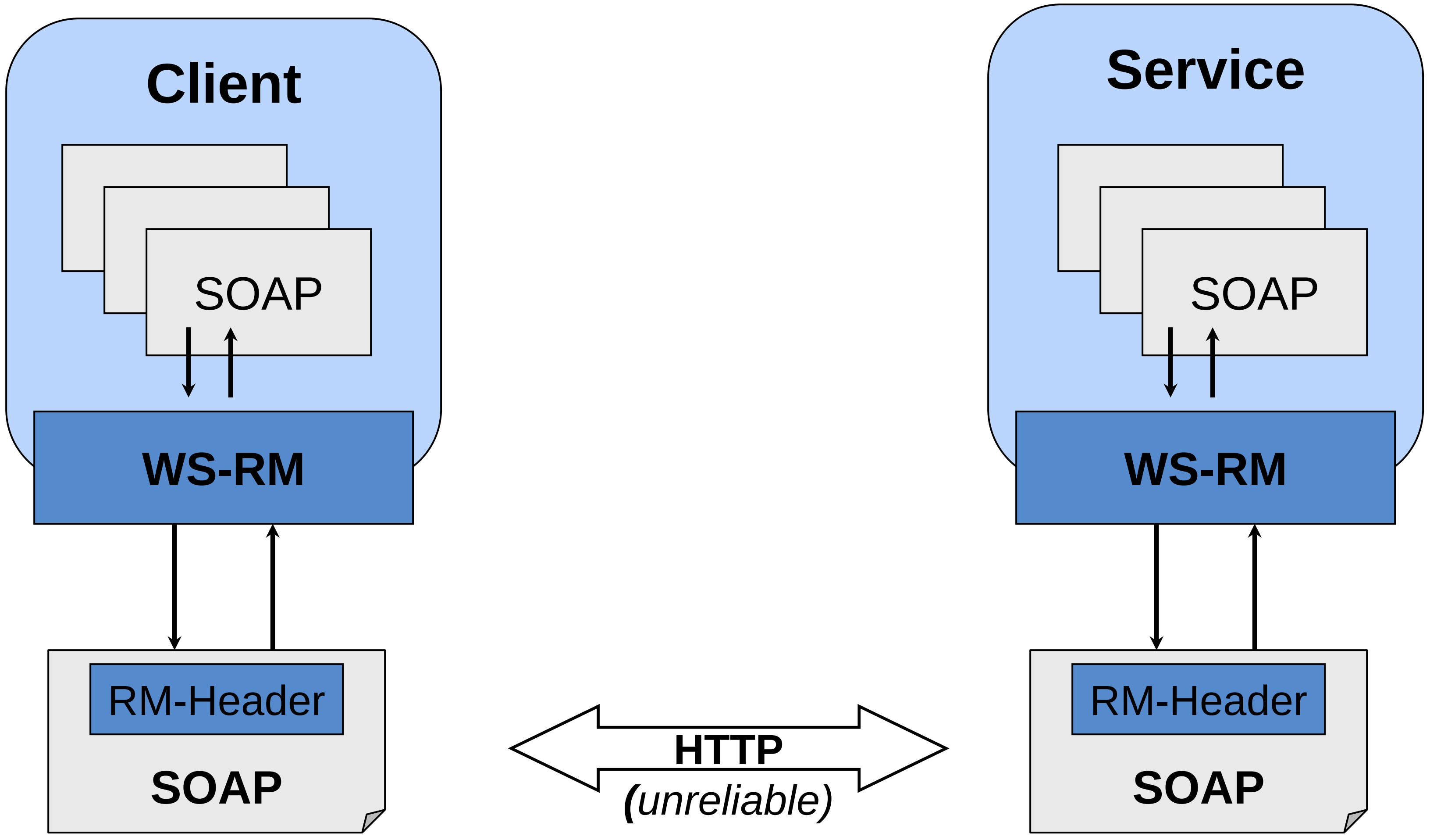
WS-ReliableMessaging

- WS-ReliableMessaging defines QoS over unidirectional message sequences
  - Exactly once, in-order
  - No persistent guarantees though

- Simplifies application programming
  - No need to defend against lost, duplicated or delayed messages

- Acknowledgements sent upon receipt
  - Uses message sequence numbers

WS-RM - The Big Picture



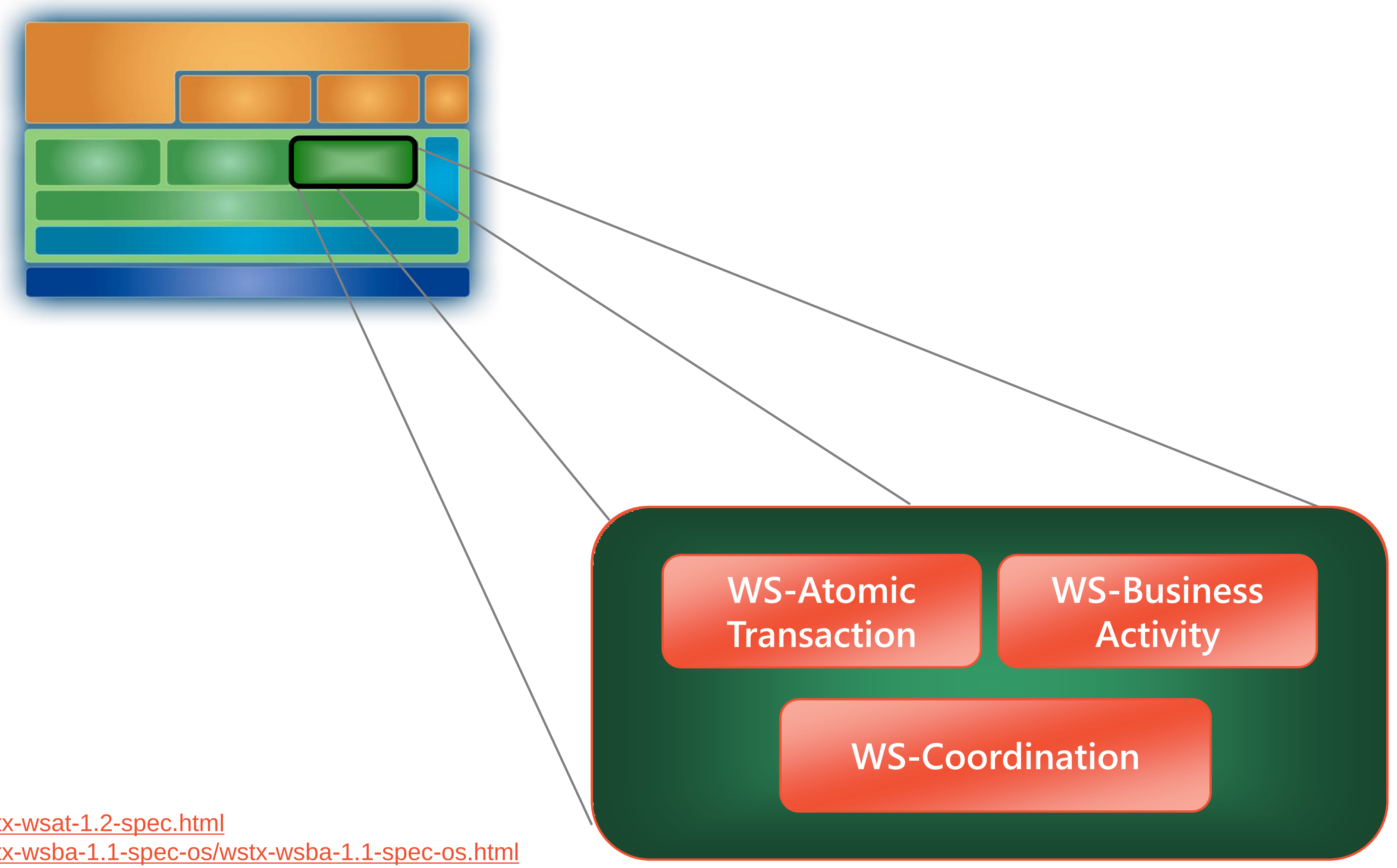
Sequence Example

```
<s:Envelope
  xmlns:s='http://www.w3.org/2003/05/soap-envelope'
  xmlns:wsu='http://schemas.xmlsoap.org/ws/2002/07/utility'
  xmlns:wsrm='http://schemas.xmlsoap.org/ws/2003/03/wsrm' >
 <s:Header>
  <wsa:Action>CreateCoordinationContext</wsa:Action>
  <wsa:To>http://localhost/TestCoordinator</wsa:To>
  <wsrm:Sequence>
    <wsu:Identifier>
      http://fabrikam123.com/abc
    </wsu:Identifier>
    <wsrm:MessageNumber>10</wsrm:MessageNumber>
    <wsrm:LastMessage/>
  </wsrm:Sequence>
 </s:Header>
 <s:Body> . . . </s:Body>
</s:Envelope>
```

Acknowledgement Example

```
<s:Envelope
  xmlns:s='http://www.w3.org/2003/05/soap-envelope'
  xmlns:wsu='http://schemas.xmlsoap.org/ws/2002/07/utility'
  xmlns:wsrm='http://schemas.xmlsoap.org/ws/2003/03/wsrm‘ >
 <s:Header>
  <wsa:Action>CreateCoordinationContext</wsa:Action>
  <wsa:To>http://localhost/TestCoordinator</wsa:To>
  <wsrm:SequenceAcknowledgement>
    <wsu:Identifier>
      http://fabrikam123.com/abc
    </wsu:Identifier>
    <wsrm:AcknowledgementRange Upper=’10’ Lower=‘1’ />
  </wsrm:SequenceAcknowledgement>
 </s:Header>
 <s:Body> . . . </s:Body>
</s:Envelope>
```

Coordination & Consistency

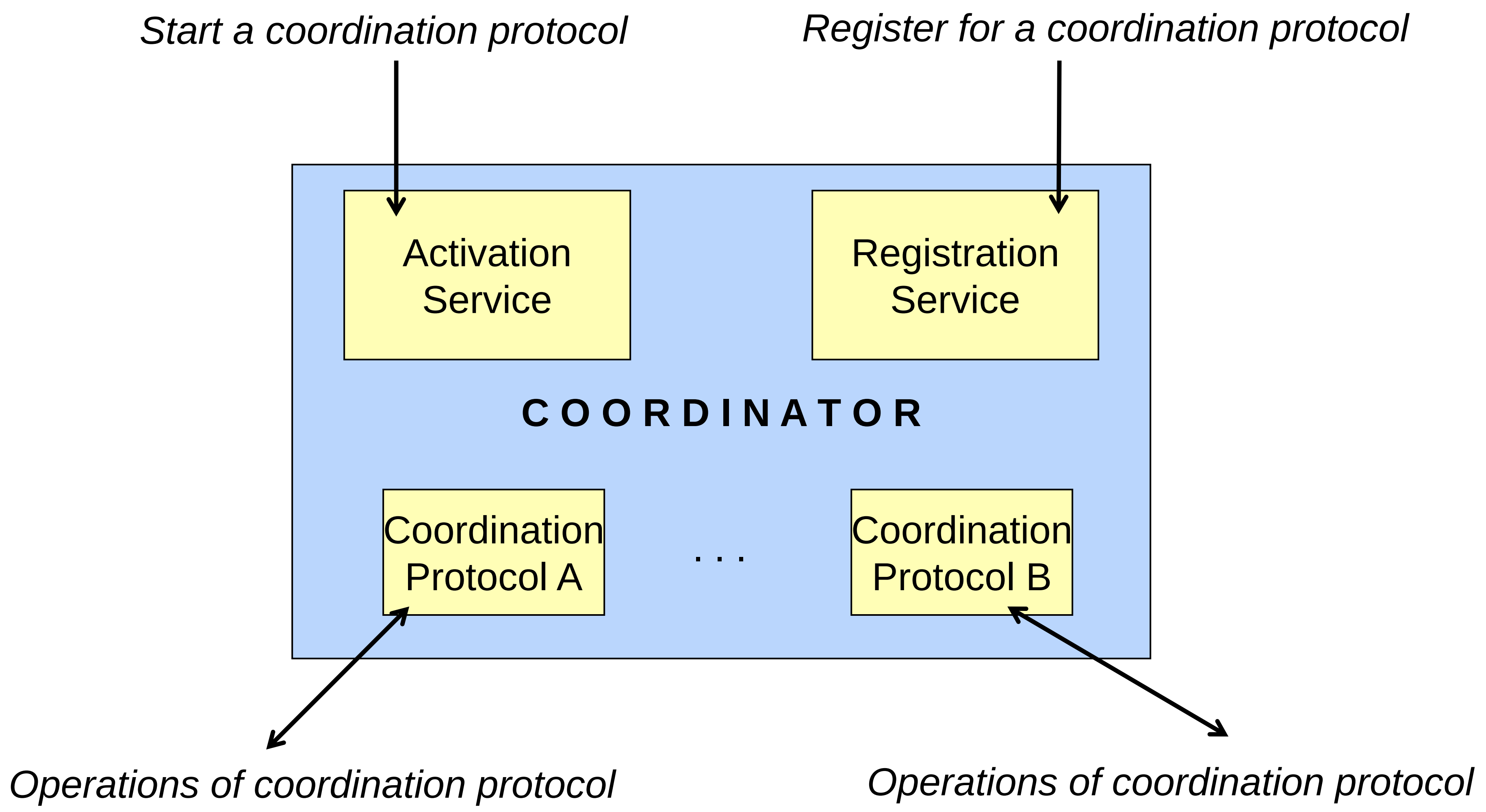


http://docs.oasis-open.org/ws-tx/wstx-wsat-1.2-spec.html
http://docs.oasis-open.org/ws-tx/wstx-wsba-1.1-spec-os/wstx-wsba-1.1-spec-os.html
http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec.html

WS-Coordination

- WS-Coordination is intended as a generic infrastructure to implement *coordination protocols* between Web Services
  - main goal: generic platform for implementing advanced transaction models; WS-AtomicTransaction (AT) and WS-BusinessActivity (BA) are based on it
  - but it can be used to implement a wide variety of coordination protocols between services (AT & BA are the only ones yet though)
- WS-Coordination APIs and behaviours:
  - Registers transactions
  - Generates coordination context
  - Passes data to register for transactions

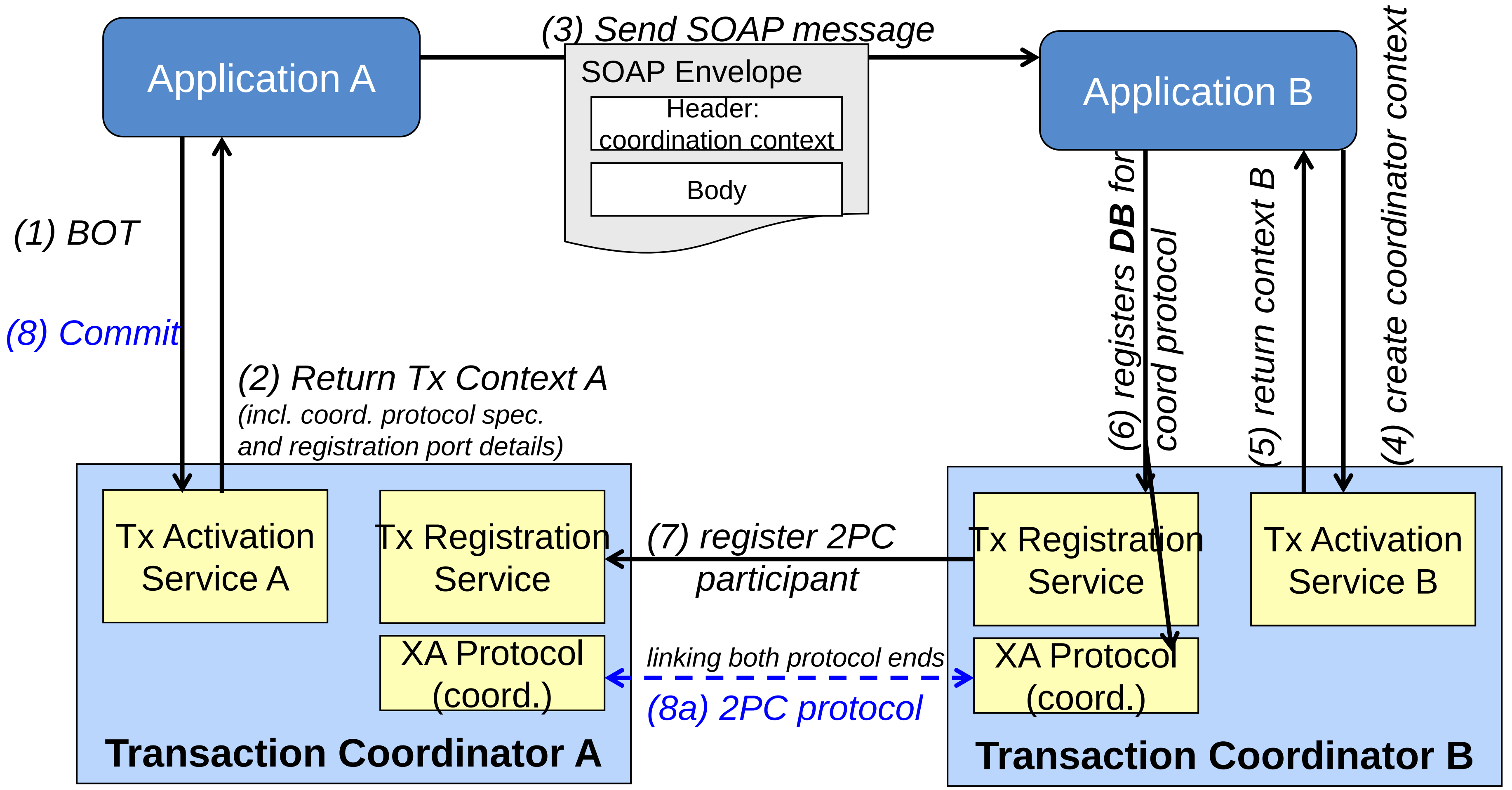
WS-Coordination Overview


Start a coordination protocol
Register for a coordination protocol
Activation Service
Registration Service
COORDINATOR
Coordination Protocol A
. . .
Coordination Protocol B
Operations of coordination protocol
Operations of coordination protocol

WS-AtomicTransaction

- Used for 2 Phase Commit ACID protocol
  - Prepare
  - Commit/Rollback
- Intended for short-lived atomic transactions
  - All resources must be up (synchronous)
  - All-or-nothing (complete agreement)
  - Uses asynchronous messages
  - Resources are locked – easy programming
- Appropriate for tightly coupled scenarios (shared assumptions about latency/trust)

WS-AT over WS-Coordination



http://msdn.microsoft.com/en-us/library/ms996526.aspx

Vendor Support

- Microsoft
  - Web Services and XML at core of .NET
    - .NET remoting & WCF based on SOAP & Web services standards
    - UDDI as local service directory if wanted
    - Transparent use of Web Services from Visual Studio (SOAP and WSDL)
- IBM & Sun
  - Extensive tools support in WebSphere etc
  - Java support for many standards from many vendors

References

– "Essential Software Architecture" by Ian Gorton (Springer 2006)
– "SOA In Practice" by Nicolai Josuttis (O'Reilly, 2007)
– www.infoq.com has many articles, talks, blogs
– "Enterprise Integration Patterns" by Gregor Hohpe and Bobby Woolf (Addison-Wesley, 2004)
– "Web Services: Concepts, Architecture and Applications" by Gustavo Alonso, Fabio Casati, Harumi Kuno and Vijay Machiraju (Springer 2004)
– Lecture material on "EAI", Gustavo Alonso, ETH Zurich, 2010. http://www.systems.ethz.ch/education/hs10/eai/lectures/Chapter-6-EAI-2010-Cloud-HA2.pdf
– More on WCF Reliable Messaging: http://msdn.microsoft.com/en-us/library/aa480191(d=printer).aspx

Industry/Guest Lectures Canvas (Recording)

Front-end Development beyond the Basics - Imad Sader, Software Engineering Manager eBay

Client-side Framework React and Vu –

Muhit Anik, Engineering Team

Lead Nasdaq