Optimized Communication for Mobile Applications

Thesis Proposal

Sangyoon Oh
[ohsangy@cs.indiana.edu]

Contents

Introduction

Why Interesting?

- Recent prominent data sharing of mobile computing with other distributed system.
  - e.g. Packet-based, Always-on Cellular
- SOAP Interoperability → Connect disparate and distributed resources.
- Prospective of popularity
  - Open Standard Community (W3C), Industry (Nokia, Sun..)
  - W3C's XML Binary Characterization WG

    ← result of the Binary Interchange Workshop.
- Mobile computing demands support for high-performance data exchange in current Web Services Environment.

Problems

- Performance Overhead in Web Services
  - XML – uses considerable resources for transmit and parsing.
    - Information storing as Text
    - Tagged Element in document (Universal End-to-End presentation format)

      → Parsing and Increase Size
- Mobile environment
  - Battery-constraint, Limited Computation
  - High-Latency, intermittent communication
- Compression of XML (GZip XML) would increase a processing time and the size for small size message

Research Statement

"There will be significant performance and reliability improvements in mobile computing when we use binary data representation and reliable messaging while preserving the SOAP Infoset."

Purpose of Research

- Purpose of Research:

  To design and develop an optimized communication framework preserving SOAP semantics for mobile computing.

- Prospect of Research
  - Separate data presentation format and data content
  - Interoperability: platform and language ←XML's universality
  - Interoperability and performance
  - Loss of self description
  - Saving parsing computation and data binding >> Latency Saving

    ←High Latency of Cellular Wireless

Related Literature Survey

Related Literature Survey

- Sun’s Fast Web Services and Fast Infoset Project
- MTOM/XOP of W3C
- DFDL of Global Grid Forum
- *Extreme! Lab’s research on SOAP Performance*
- *Cross-Format Schema Protocol*

Classification of SOAP Alternatives

- Some data doesn’t fit in XML serialization

  → MTOM/XOP
  - Media data
    - Standard format with compression
    - JPEG, GIF, MP3
  - Data that includes digital signature
    - Binary integrity would not be preserved after serialized into XML format
- Some Application domain use a binary representation to avoiding verbose XML encoding.

  → Fast Web Services/Fast Infoset, Extreme! Lab
  - Send pre-parse data over binary communication

Summary of Projects

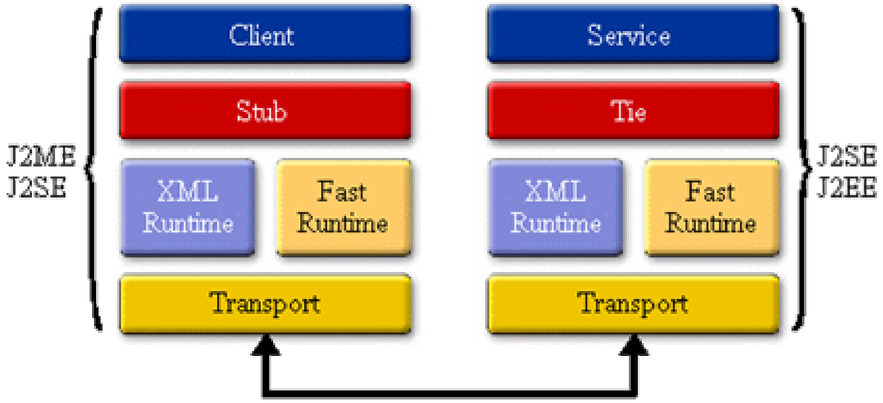
	Extreme! Lab	***MTOM/XOP***	***Fast Web Services***
Properties	Negotiation, Binary Application Protocols	Support binary contents	GZip for XML
Target	Scientific Data	Media File, Data that includes digital signature	Java Application (No changes on Application level)
Lack of mobile issues	QoS	Performance, Optimized message scheme for in-memory representation	Fast Infoset is not interoperable, QoS

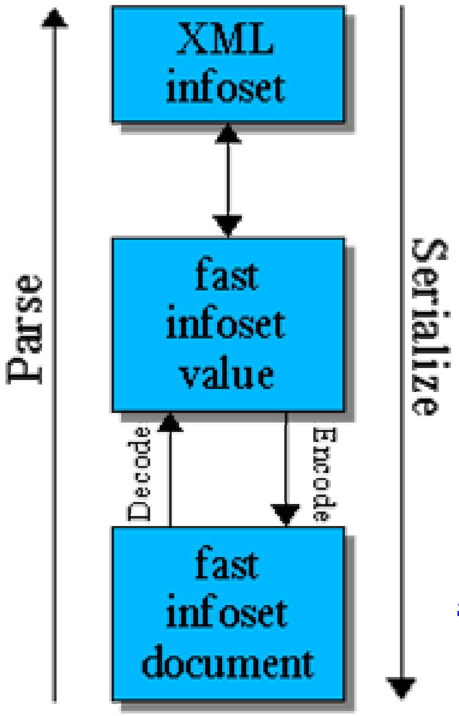
Sun’s Fast Web Services and Fast Infoset Project

- Fast Web Services make use of Fast Infoset (Binary encoding of XML Infosets) for carrying SOAP messages.
  - Fast Infoset: specifies a representation of an instance of the XML Infoset using binary encoding.
  - Use ASN. 1. for binary encoding
    - Binary Encoding → Packed Encoding Rule (PER) – Octet
  - Alt: Map SOAP 1.2 Schema into ASN. 1.

Fast Infoset

- No end tags
- Indexing repeated string
- Indexing qualified names





PERed

Fast Infoset

Fast Web Services

Fast Infoset: Example

```
<root>
    <tag>one</tag>
    <tag>two</tag>
    <anotherTag>one</anotherTag>
</root>
```

{0}<root>
{1}<tag>**{0}**one
[1]<>**{1}**two
{2}<anotherTag>**[0]**

	Local Name	***Content***
0	root	one
1	tag	two
2	anotherTag	

MTOM/XOP

- MTOM (Message Transmission Optimization Mechanism) and XOP (XML-binary Optimized Packaging)
  - By W3C XML Protocol Working Group
  - MTOM describes how XOP is layered into SOAP-HTTP.
  - Serialization of XML: Looks like MIME
  - Still Preserve “XML structure”
    - Binary Data + XML Fragment

XOP: Example

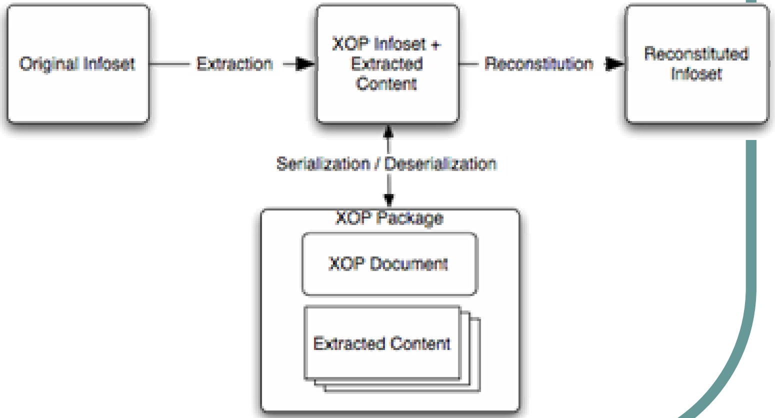
```
--MIME_Boundary
Content-ID: <mymainpart@crf.canon.fr>
Content-Type: application/xop+xml;charset=UTF-8;type="application/soap+xml"
Content-Transfer-Encoding: binary

<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:xmlmime="http://www.w3.org/2004/06/xmlmime" xmlns:xop="http://www.w3.org/2004/08/xop/include">
    <soap:Header></soap:Header>
    <soap:Body><ns1:EchoTest                    xmlns:ns1="http://example.org/mtom/data">
          <ns1:Data><xop:Include
          href="cid:thismessage:/frog.jpg"></xop:Include>
          </ns1:Data></ns1:EchoTest>
    </soap:Body>
</soap:Envelope>

--MIME_Boundary
Content-ID: <thismessage:/frog.jpg>
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary

ÿØÿà┼JFIF┌┌┌┌ÿÛC◘—●–│◘●●● ◘ ♀¶ ♀

    ♀├↕‼☼¶ →    →   $.' ",#  (7),01444 '9=82<.342ÿÛC┌ ♀♀↑
--MIME_Boundary--
```

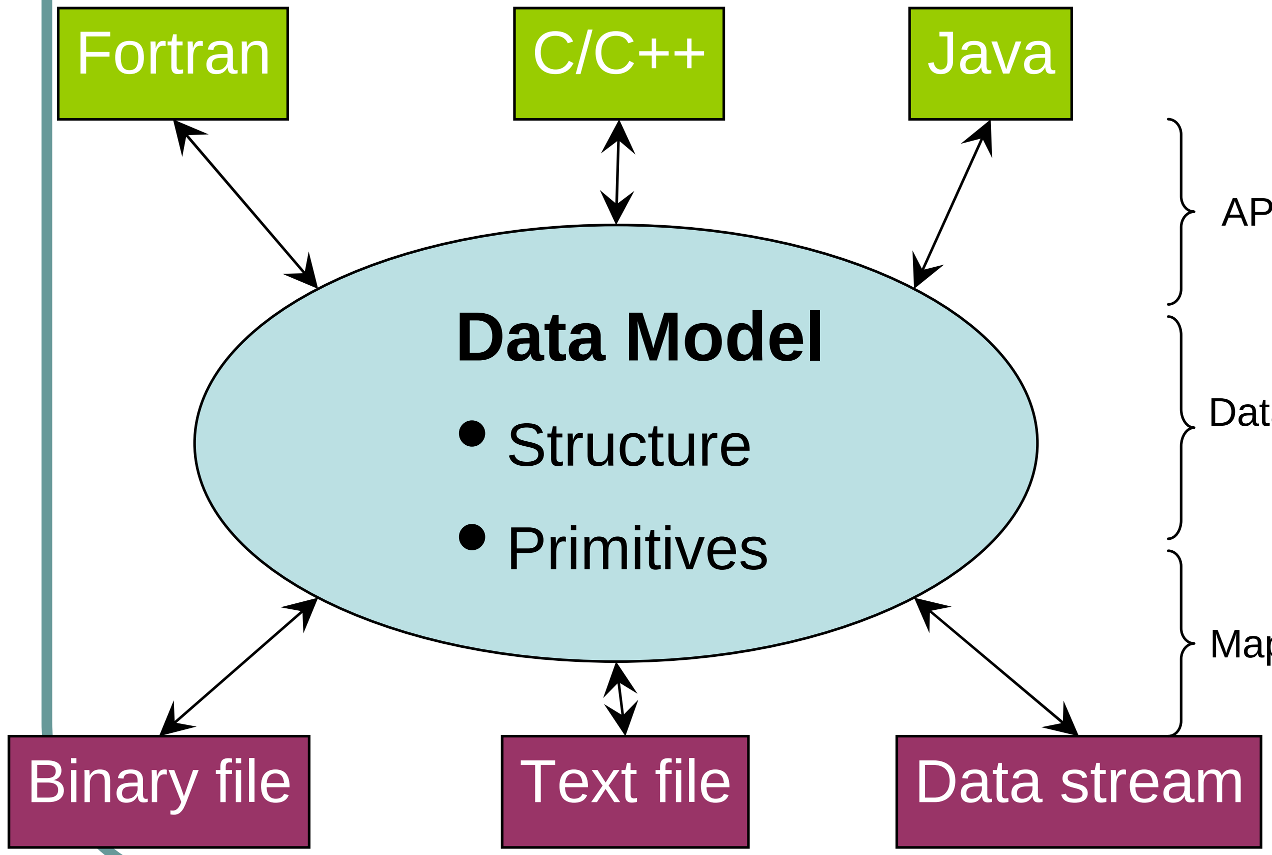


- ***HTTP Header has removed***

Data Format Description Language (DFDL)

- Define an XML-based language for describing the structure of binary and character encoded data
  - DFDL Syntax allows a description of abstract data model using XSD.
- Data file or data stream
- Provide Processing Library
  - Maybe as Web Services, e.g. OGSA Virtual Data Service.
- Process binary data to structured output XML Abstract Data Model
  - Data structure, Complex-type ← schema
  - Lower Layer (Mapping): physical representation, base-type (4 byte integer, little-endian format…)
  - Upper Layer: Programming API

DFDL Overview



Example

Vector myspectrum = DFDL.getElement("spectrum");

A structure called "spectrum" that is a 200 value vector of integers

Integers should be read as 4 bytes, little endian format

DFDL: Example

Example XML Representation A_1

```
<?xml version="1.0" encoding="UTF-8" ?>
        <DFDL xmlns="DFDL" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:noNamespaceSchemaLocation="C:\RECORDS\projects\BFD-DFDL\jdm1\NewDFDLType.xsd">
        <vector>
                    <point>
                                <real>1.23</real>
                                <imaginary>2.34</imaginary>
                    </point>
                    <point>
                                <real>3.45</real>
                                <imaginary>4.56</imaginary>
                    </point>
        </vector>
</DFDL>
```

DFDL: Example

Example Description File in DFDL for XML A1

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified" attributeFormDefault="unqualified" xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns="DFDL">
        <!-- New DFDL Type -->
        <xs:complexType name="complexNumber">
                <xs:sequence>
                        <xs:element name="real" type="xs:float"/>
                        <xs:element name="imaginary" type="xs:float"/>
                </xs:sequence>
        </xs:complexType>
        <xs:element name="DFDL">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name="vector">
                                        <xs:complexType>
                                                <xs:sequence>
                                                        <xs:element name="point" type="complexNumber" maxOccurs="10"/>
                                                </xs:sequence>
                                        </xs:complexType>
                                </xs:element>
                        </xs:sequence>
                 </xs:complexType>
        </xs:element>
        <!-- Notes:
                        New DFDL types are simply new types defined in XML Schema
                        To be usable, new types must be associated with one or more transforms. By default, a transform that is the
                         equivalent of reading constituent elements is available.
                        In this example, complexNumber can be created by sequentially reading two floats using the default transform,
                        so no explicit specification of the transform is needed (see ExplicitTransformSpecification.xsd for this case).
         -->
</xs:schema>
```

SOAP Performance Research Extreme! Lab

- Research of SOAP performance for scientific computing
  - Schema-specific parsing
  - Persistence Connection and Chunking (Streaming) of messages
  - ASCII-Double Conversion ← the most critical overhead
- Target: Large Files with many small objects (million floats)
- High performance communication Approach (Recommendation)
  - Use SOAP as an initial mechanism for negotiating faster protocol
  - Use multiple communication protocol to handle wide range of scientific computing requirements

Handheld Flexible Representation

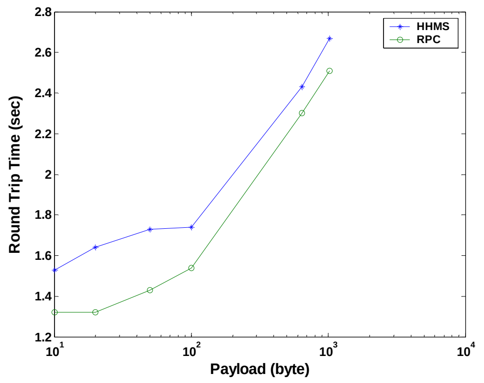
Architecture Requirements

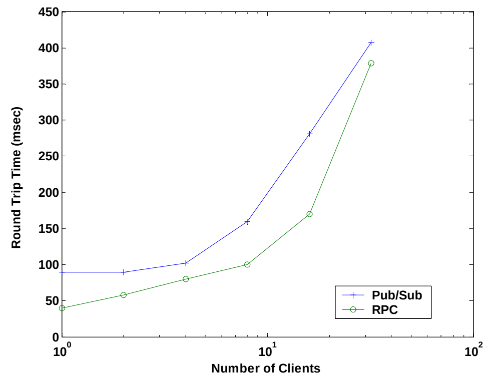
- Mobile computing issues
  - Reliable Communication Model
  - Simpler Security Model
  - Compatible with J2ME platform
- Flexible Representation preserving SOAP Semantics
  - Binary encoding
  - Negotiation
  - Quality of Service (Reliability and Security)

  → Expecting to reduce parse time, bandwidth usage
- Work best for a stream
  - Designed for on high-frequency small messages
- Not limited to proxy architecture
  - Demonstrate on current HHMS-NB framework

Handheld Messaging Service

- General mobile communication framework by CGL
- Provides a core pub/sub API for mobile application.
  - Lightweight user library
  - Server-side gateway
  - TCP and HTTP transport
- HHFR is a part of HHMS

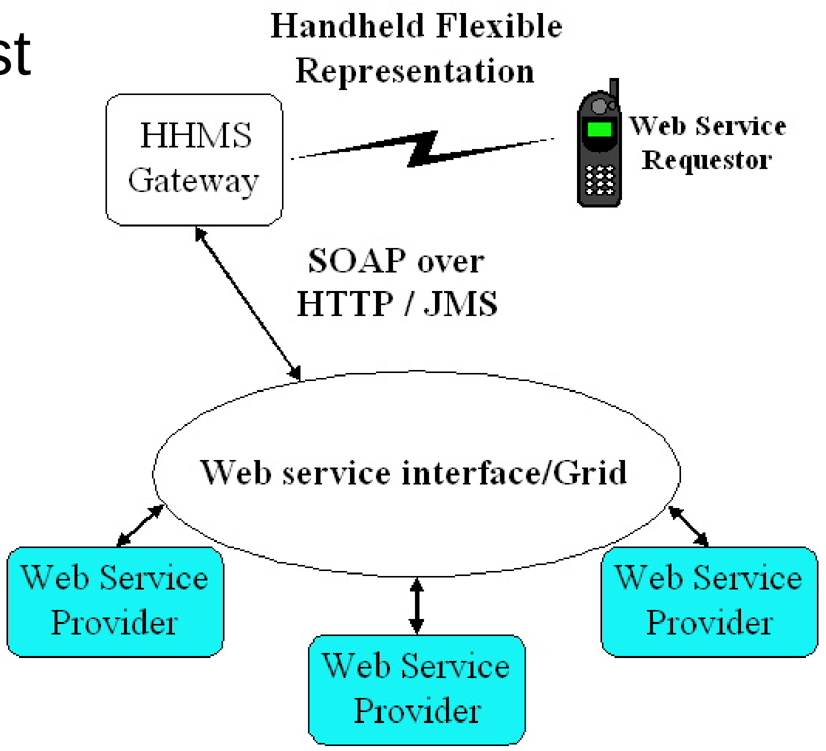




Overview Architecture

Simple Scenario of a Stream

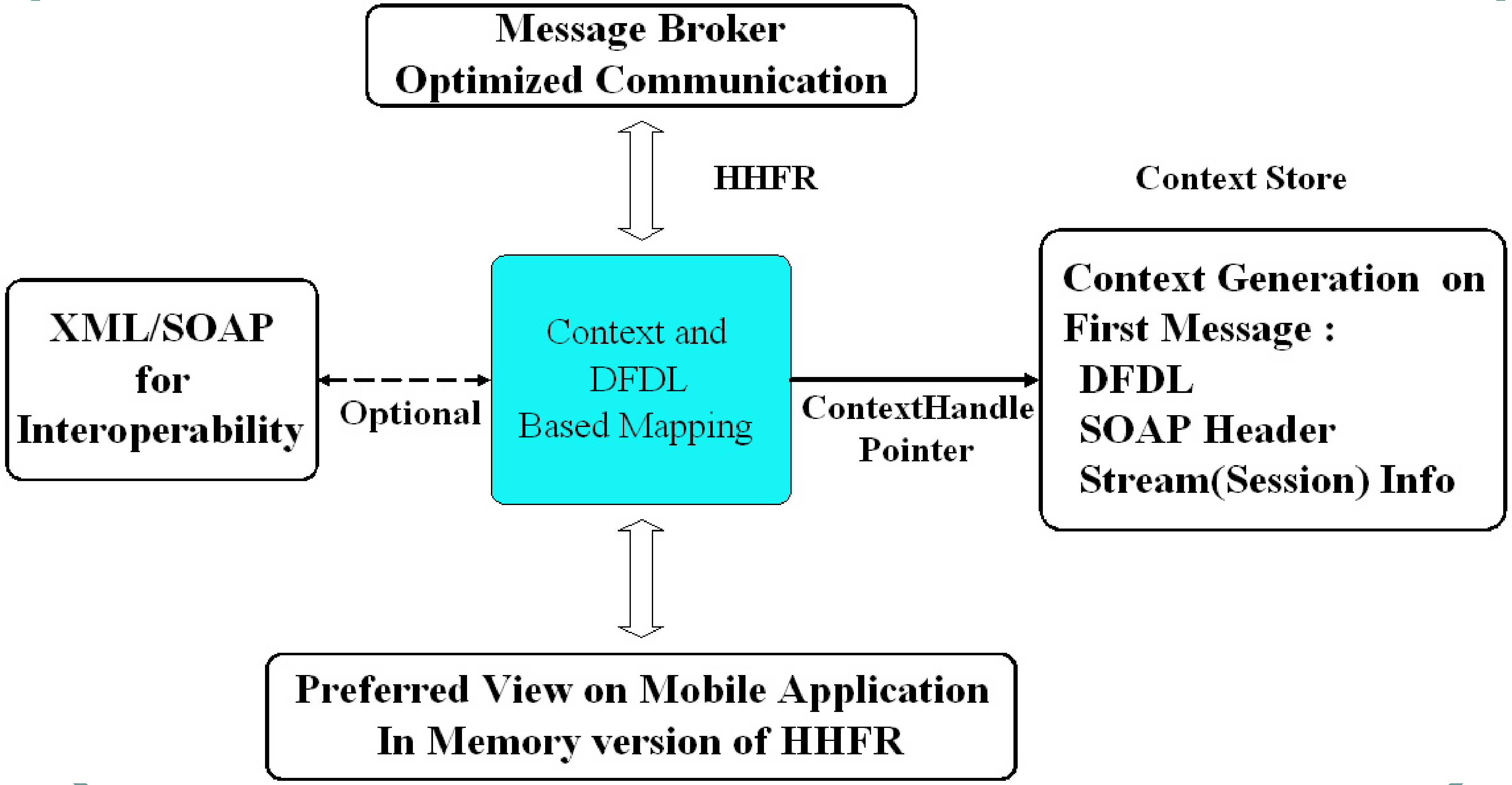
1. Mobile application request Negotiation for stream characteristics
2. Gateway response
3. Gateway generates encoder/decoder
4. Establish Flexible Representation stream
5. Terminate a stream



Message Generation

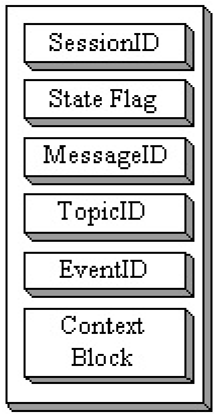
- Works best in a Stream in Web Services
  - Unchanged SOAP Header used throughout the stream
  - Data Structure and SOAP message header ← beginning of stream (Negotiation)
  - SOAP body of changing content ← Binary encoding
- Use DFDL to define message structure
- XML/SOAP serialization and parsing ← use DFDL API and library

Message generation

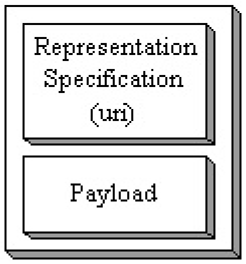

Message Broker
Optimized Communication
HHFR
Context Store
XML/SOAP for Interoperability
Optional
Context and DFDL Based Mapping
ContextHandle Pointer
Context Generation on First Message :
DFDL
SOAP Header
Stream(Session) Info
Preferred View on Mobile Application
In Memory version of HHFR

Message Generation

- Pointer (URI)
  1) Pointer to Context Store (Representation Specification)
  2) Pointer to Method used e.g. HHFR (In SOAP Header)
- Context store
  - SOAP Header
  - Stream Info
  - DFDL schema



Message block



Context block

Negotiation

- Two end-points exchange characteristics of stream.
- Negotiate in a conventional SOAP message.
  - Data structure in DFDL, reliable messaging scheme, security model, and transport protocol
  - In pre-defined negotiation schema
- Current ad-hoc scheme to store Context Information
  - WS-Context for dynamic meta-data

Handlers

- Context Handler
  - Filter: mapping presentation format
  - Context store: SOAP Header, Stream (Session) Info and DFDL
- Reliability Handler
  - Reliable Messaging processor
  - WS-Reliable Messaging compatible
  - Trace messages and assure message delivery
    - Optimized ACK and NAK

Handlers

- Security Handler
  - Binary encoding
    - WS-Security → very careful to apply XML level security
  - Traditional encryption → HTTPS or Message block encryption with lightweight library
    - E.g. Bouncy Castle (http://www.bouncycastle.org/)

Research Issues

Research Issues

- Having on-the-wire representation of SOAP Infoset for wireless communication
- Performance vs. Interoperability
- Break-even point
  - SOAP and Handheld Flexible Representation
- Performance vs. Quality of Service
  - reliability and security
- Application Domain
  - A/V Conferencing

Summary of Contributions

Contributions

- We will show how to improve SOAP message transmission performance without sacrificing its interoperability.
- A novel approach to optimize message transmission by designing scheme which is message-based and capitalizing SOAP message structure.
- We will experiment the idea - Handheld Flexible Representation (HHFR) by building a prototype, for mobile applications in stream specific application domains.
- Also, we will examine gains and losses of QoS in HHFR.