Bitstream Syntax Description Language: Application of XML-Schema to Multimedia Content Adaptation

Myriam Amielh & Sylvain Devillers

Philips Research France

Presentation Plan

- Content Adaptation
- Scalable Coding formats
- XML description of a multimedia bitstream
- Bitstream Syntax Description Language (BSDL)
- XMLtoBin
- System approach / Demo
- Conclusion and future works

Content adaptation: system features

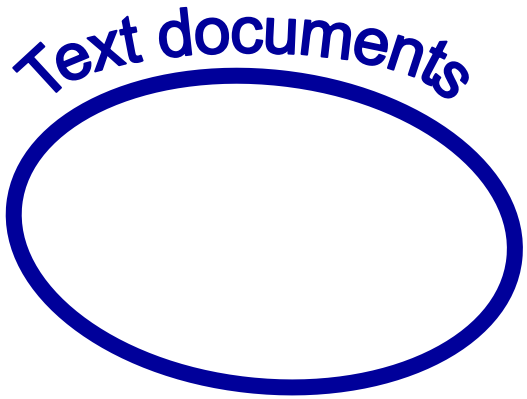
Content servers:
audio, video, MPEG, JPEG2000, H263, proprietary formats

Device capabilities:
display, CPU, memory

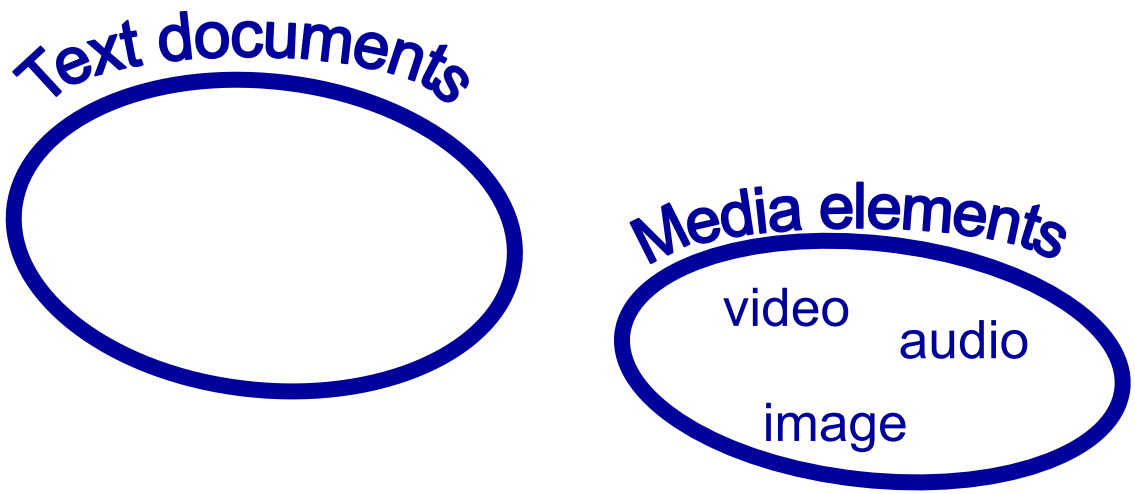
Wireless networks

Content adaptation: content types

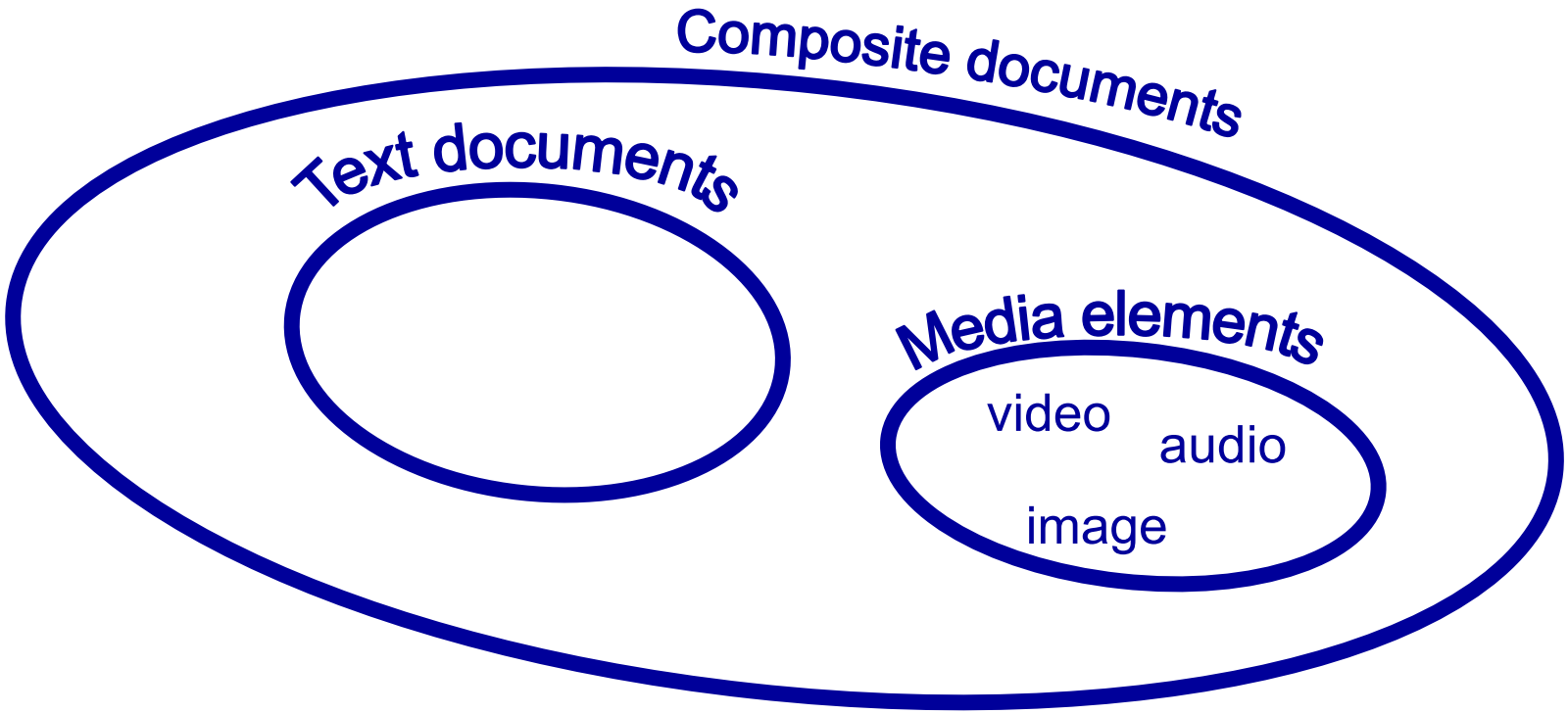
Content adaptation: content types



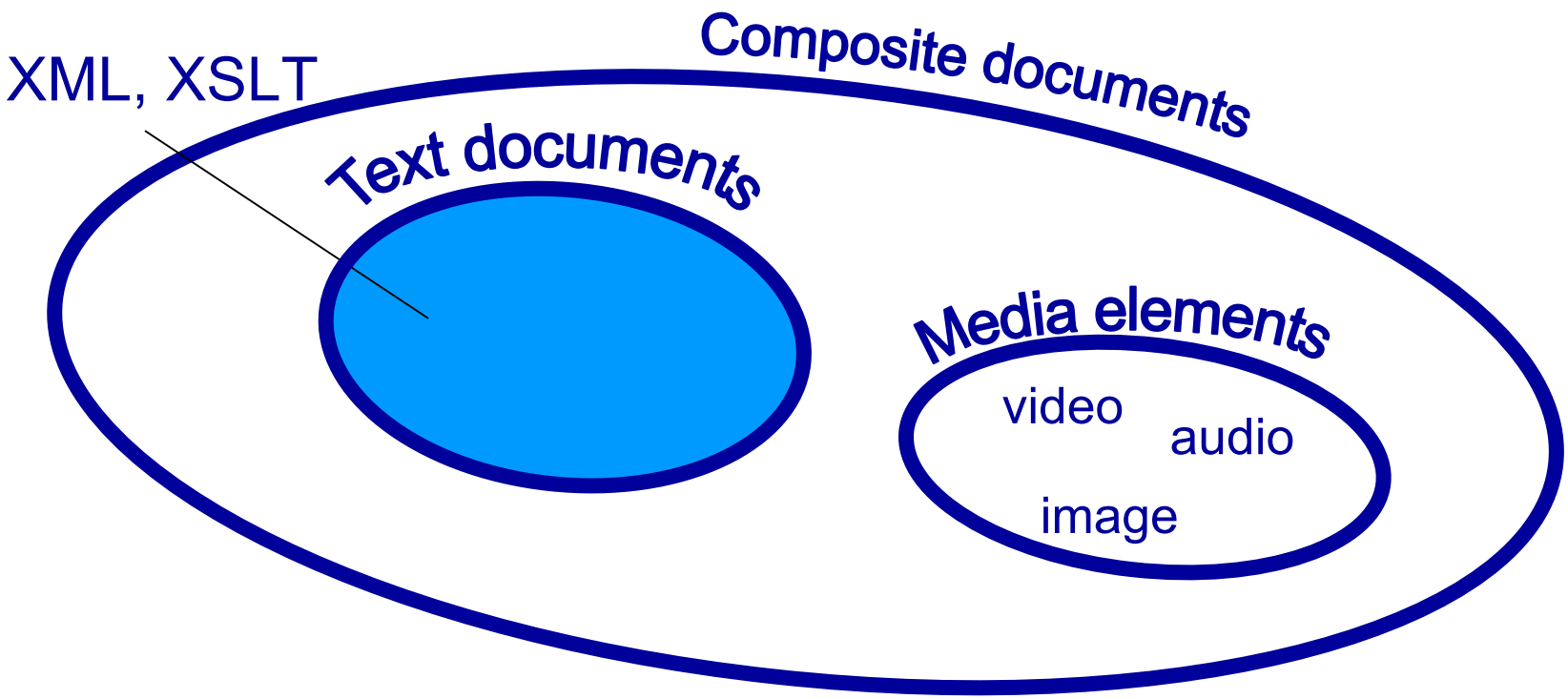
Content adaptation: content types



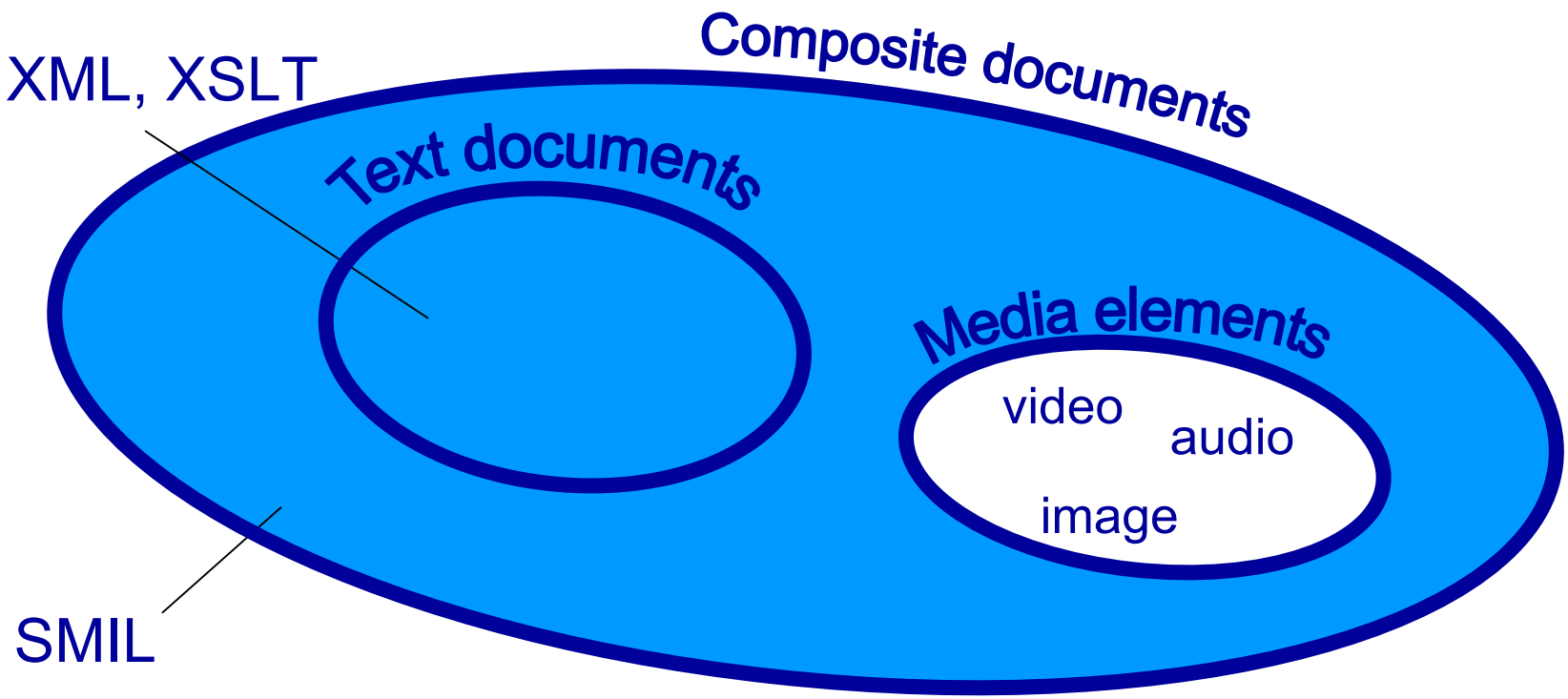
Content adaptation: content types



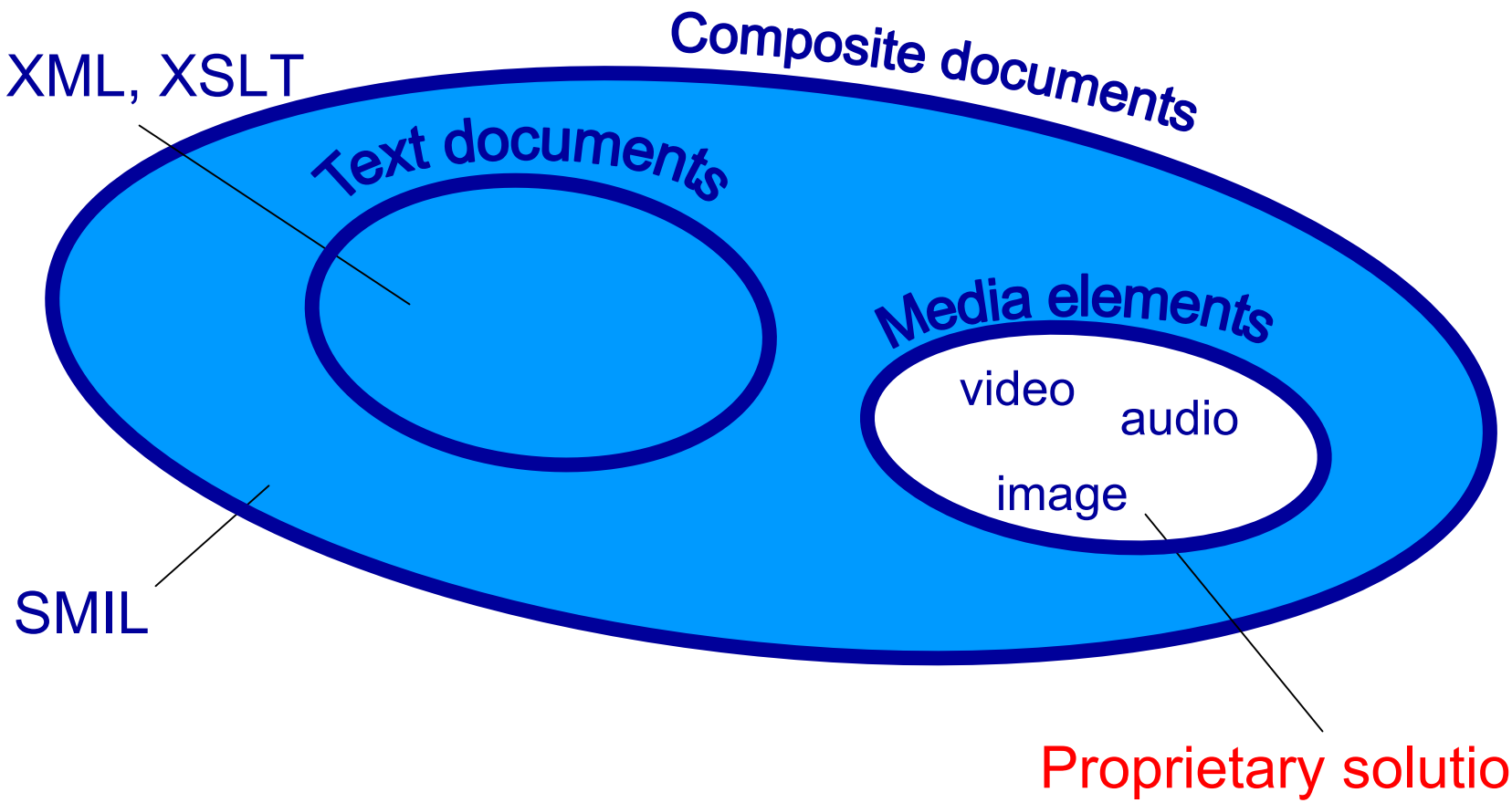
Content adaptation: content types



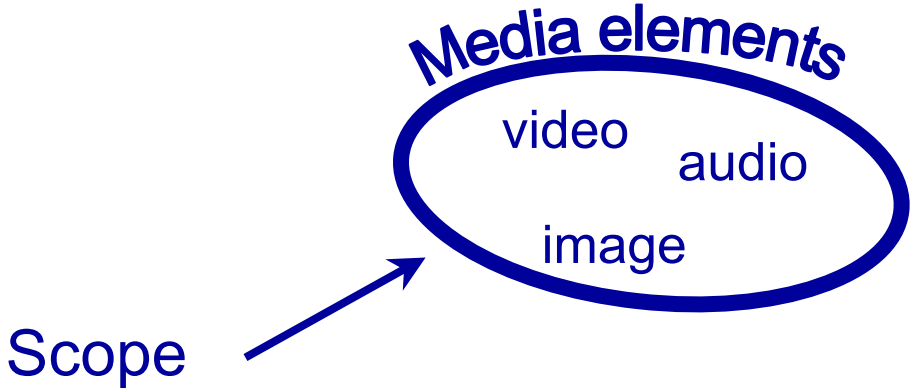
Content adaptation: content types



Content adaptation: content types

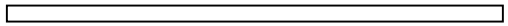


Content adaptation: content types

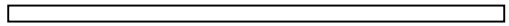


Scalable Coding formats

Scalability by quality (SNR)

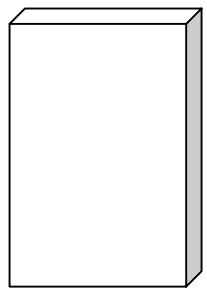


Spatial scalability



Temporal scalability

3D scalable scenes



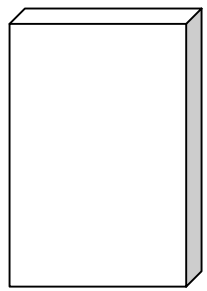
Scalable Coding formats

Scalability by quality (SNR)

Spatial scalability

Temporal scalability

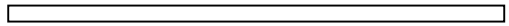
3D scalable scenes



Scalable Coding formats

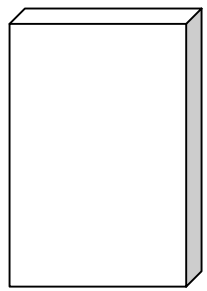
Scalability by quality (SNR)

Spatial scalability



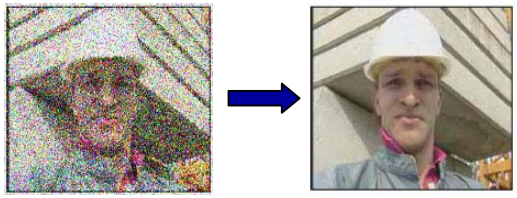
Temporal scalability

3D scalable scenes



Scalable Coding formats

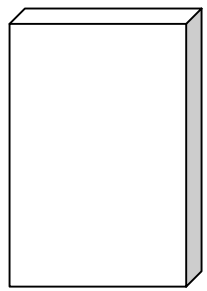
Scalability by quality (SNR)



Spatial scalability

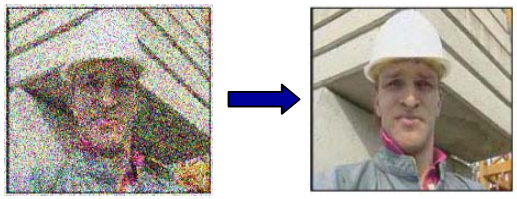
Temporal scalability

3D scalable scenes



Scalable Coding formats

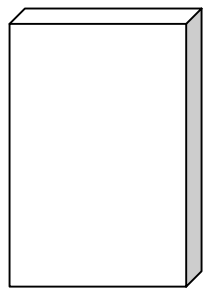
Scalability by quality (SNR)



Spatial scalability

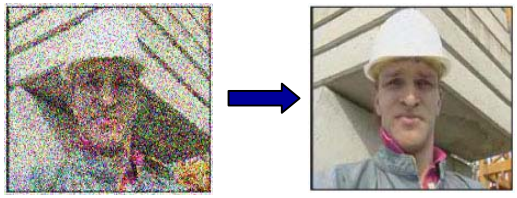
Temporal scalability

3D scalable scenes



Scalable Coding formats

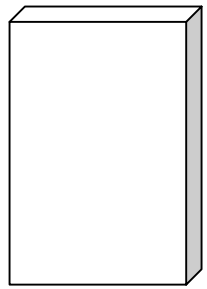
Scalability by quality (SNR)



Spatial scalability

Temporal scalability

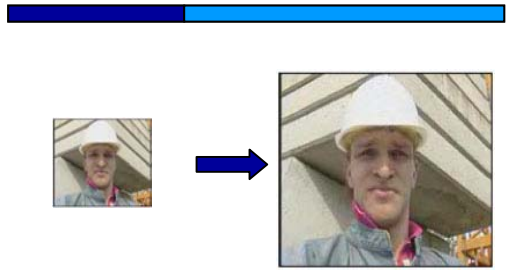
3D scalable scenes



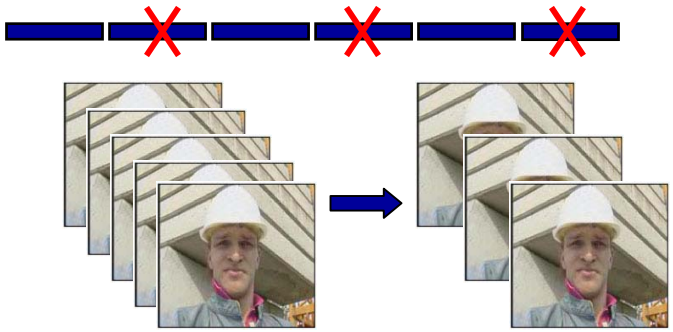
Scalable Coding formats

Scalability by quality (SNR)

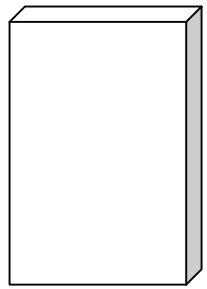
Spatial scalability



Temporal scalability



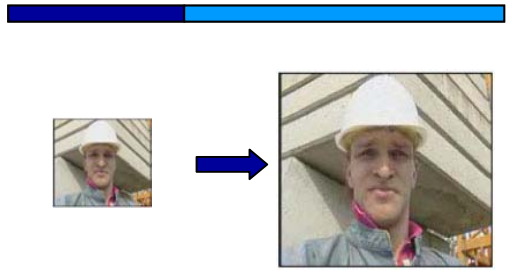
3D scalable scenes



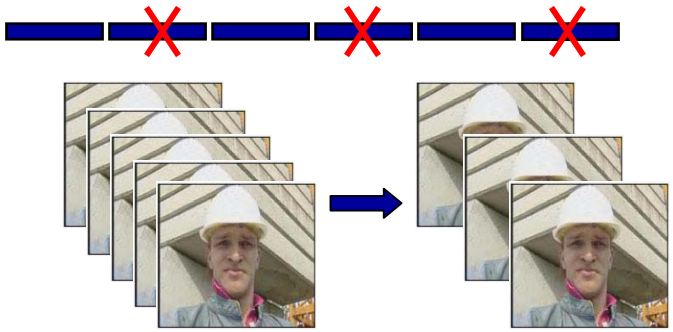
Scalable Coding formats

Scalability by quality (SNR)

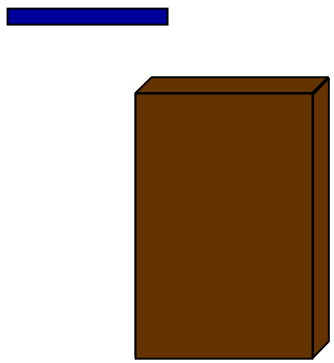
Spatial scalability



Temporal scalability



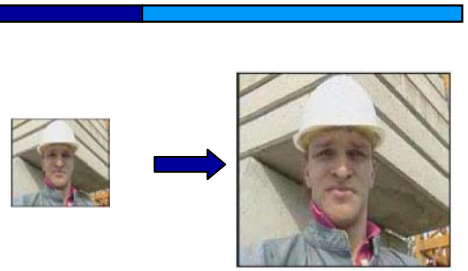
3D scalable scenes



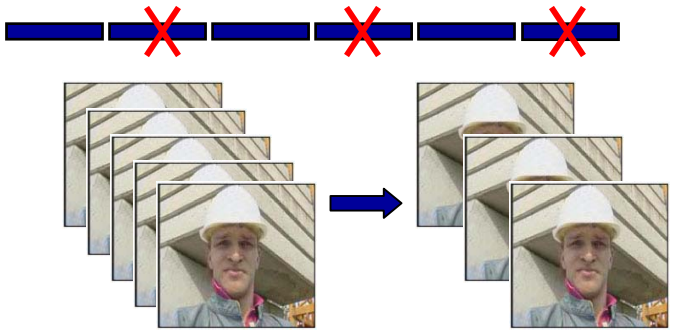
Scalable Coding formats

Scalability by quality (SNR)

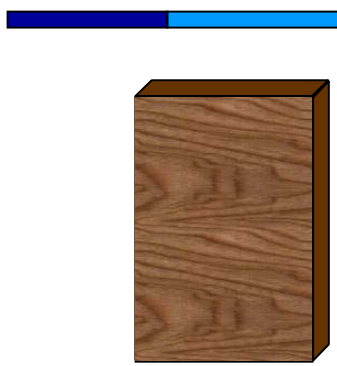
Spatial scalability



Temporal scalability



3D scalable scenes



Dynamic Content Adaptation

- Issue: how to fully exploit scalability for content adaptation ?
- In a client-server application:
  - The client requests a scalable content available on a server
  - The server establishes a content negotiation stage
  - The server edits the bitstream, and sends it to the client

⇨ Need of a software to parse and edit the bitstream

- Issue: each coding format has its own data structure
  - one software per available format
  - update the software if new version

⇨ Generic approach: use a common structuring language to describe the bitstream syntax

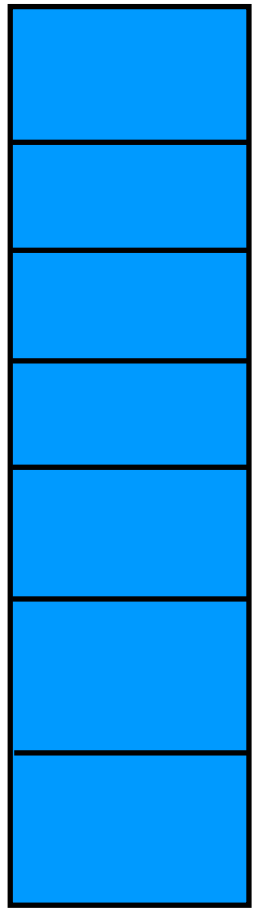
➡ Use XML !

XML Description of a Multimedia Bitstream

- *Not* an alternative format, but additional layer describing the high-level structure (i.e. in packets or layers)
- May be stored / transmitted along with content
- Data may be
  - embedded in the XML description
    - in a readable format
    - in binary format (hexadecimal or base64)
  - in an external entity (pointed by the XML description)

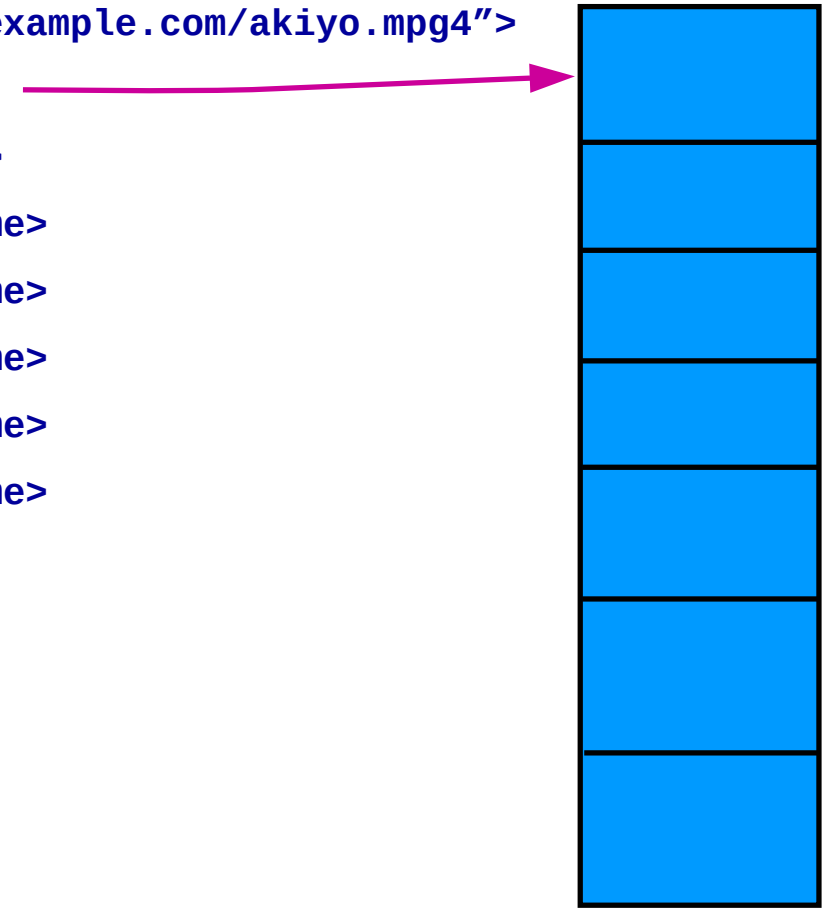
XML description

```
<Bitstream
  xml:base="http://www.example.com/akiyo.mpg4">
  <Header>0-17</Header>
  <frame>18-4658</frame>
  <frame>4659-4756</frame>
  <frame>4757-4772</frame>
  <frame>4773-4795</frame>
  <frame>4796-4973</frame>
  <frame>4974-5026</frame>
</Bitstream>
```



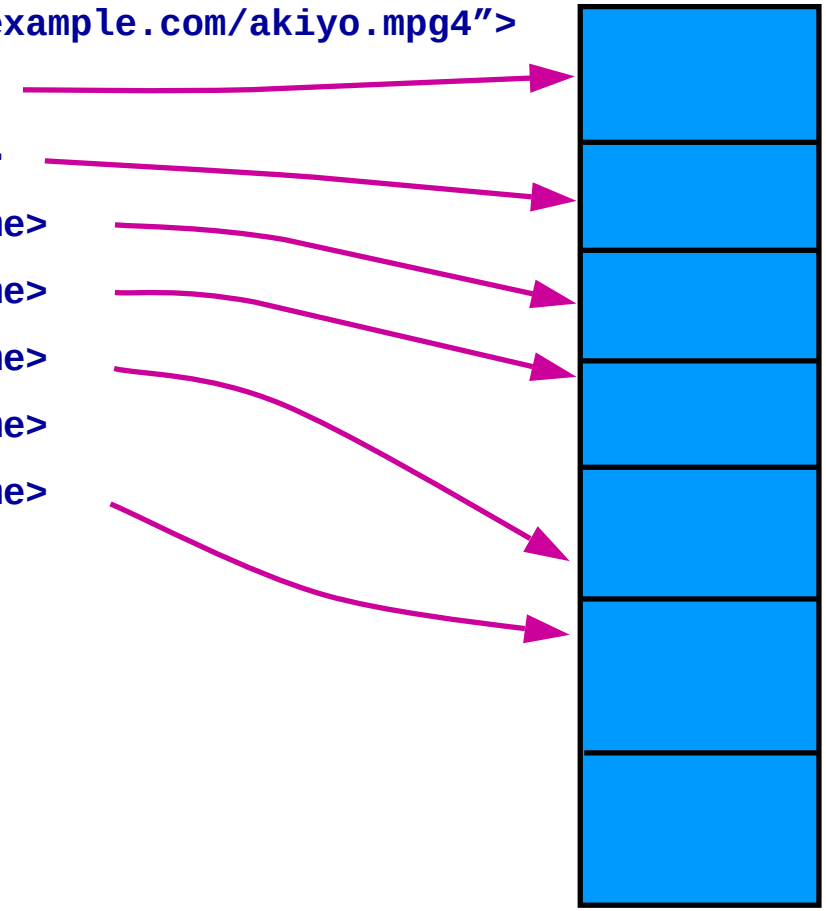
XML description

```
<Bitstream
  xml:base="http://www.example.com/akiyo.mpg4">
  <Header>0-17</Header>
  <frame>18-4658</frame>
  <frame>4659-4756</frame>
  <frame>4757-4772</frame>
  <frame>4773-4795</frame>
  <frame>4796-4973</frame>
  <frame>4974-5026</frame>
</Bitstream>
```



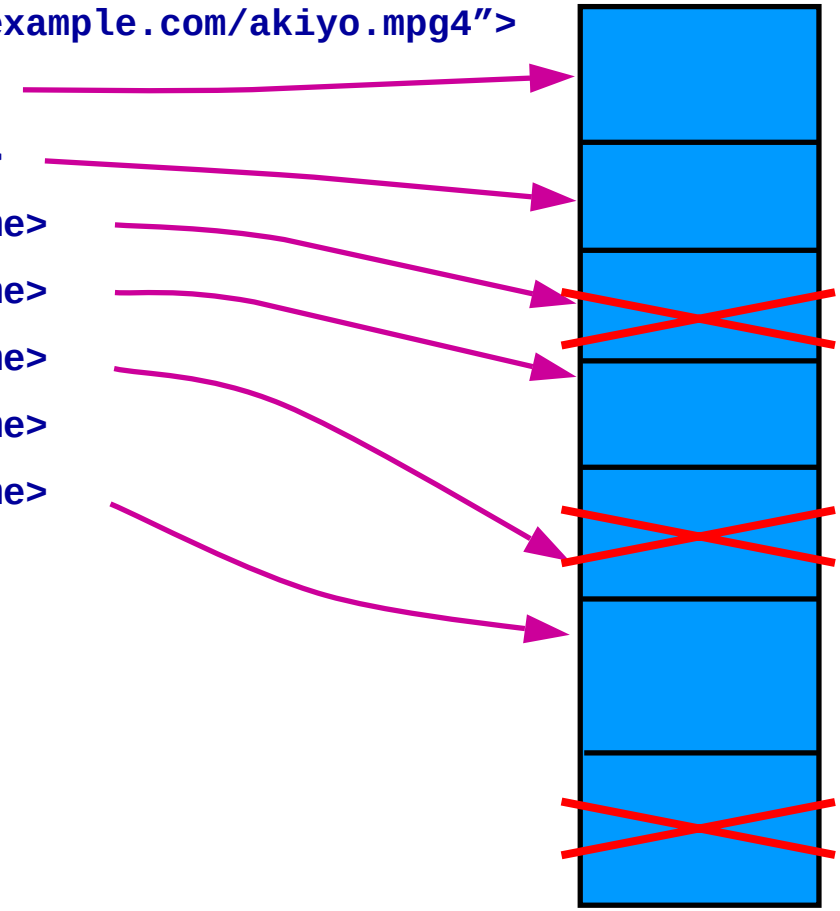
XML description

```
<Bitstream
  xml:base="http://www.example.com/akiyo.mpg4">
  <Header>0-17</Header>
  <frame>18-4658</frame>
  <frame>4659-4756</frame>
  <frame>4757-4772</frame>
  <frame>4773-4795</frame>
  <frame>4796-4973</frame>
  <frame>4974-5026</frame>
</Bitstream>
```



XML description

```
<Bitstream
  xml:base="http://www.example.com/akiyo.mpg4">
  <Header>0-17</Header>
  <frame>18-4658</frame>
  <frame>4659-4756</frame>
  <frame>4757-4772</frame>
  <frame>4773-4795</frame>
  <frame>4796-4973</frame>
  <frame>4974-5026</frame>
</Bitstream>
```



XML description

```
<Bitstream
  xml:base="http://www.example.com/akiyo.mpg4">
  <Header>0-17</Header>
  <frame>18-4658</frame>
  <frame>4659-4756</frame>
  <frame>4757-4772</frame>
  <frame>4773-4795</frame>
  <frame>4796-4973</frame>
  <frame>4974-5026</frame>
</Bitstream>
```

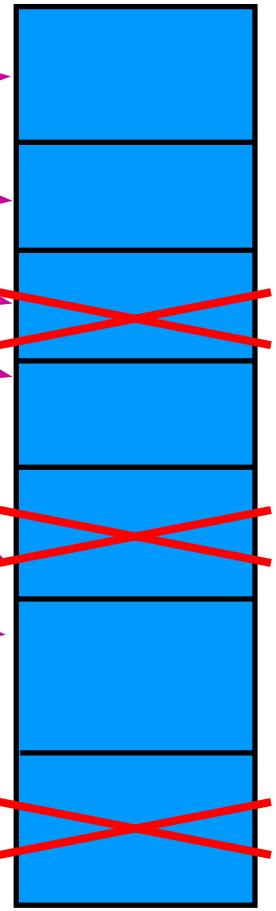
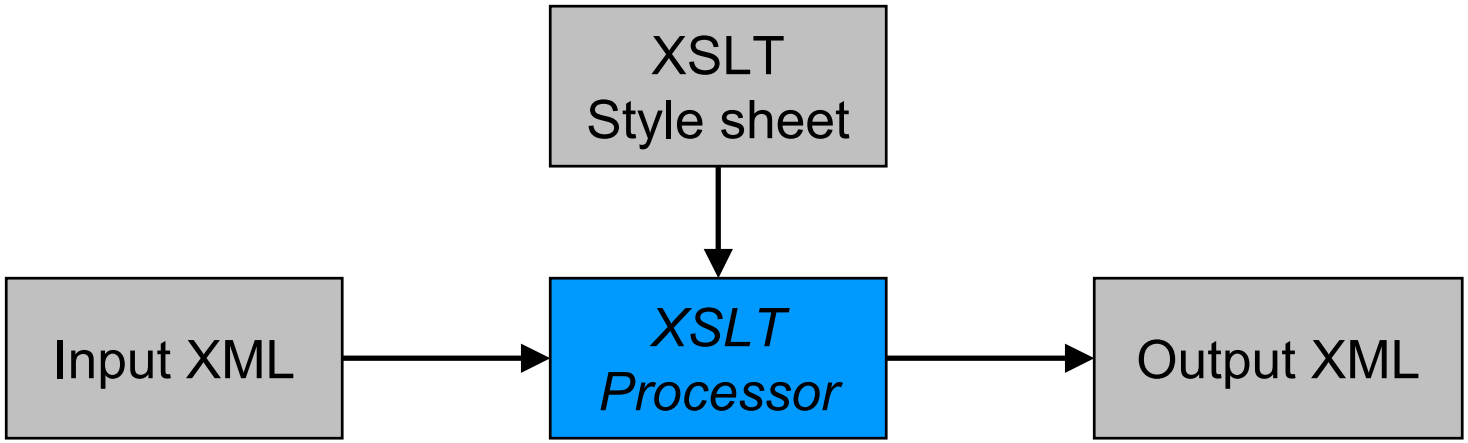


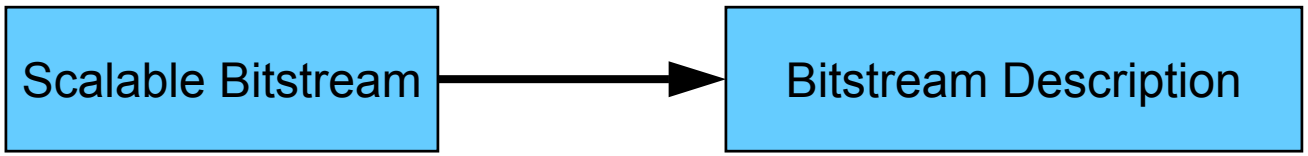
Image Transformation with XSLT

- eXtensible Stylesheet Language Transformations (XSLT): W3C language specifying XML-to-XML transformations

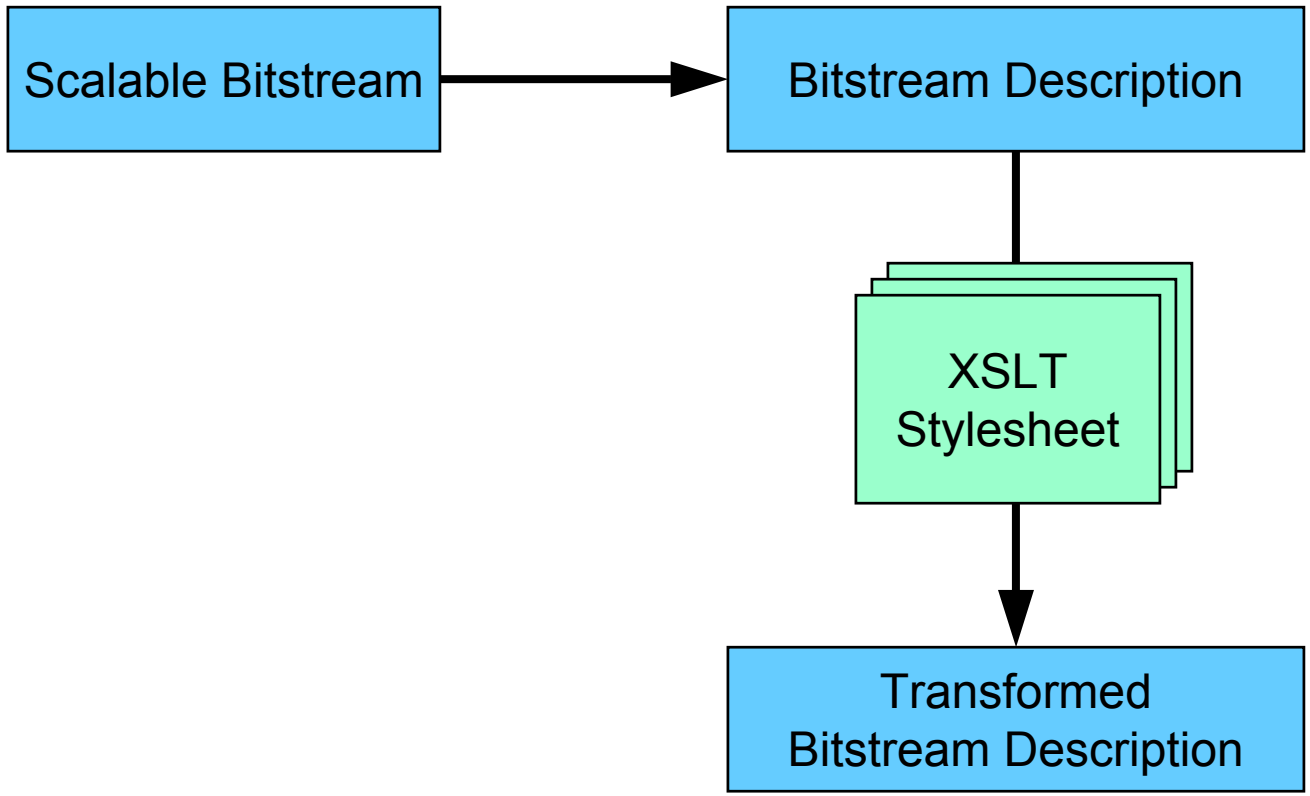


Scalable Content Adaptation with XML

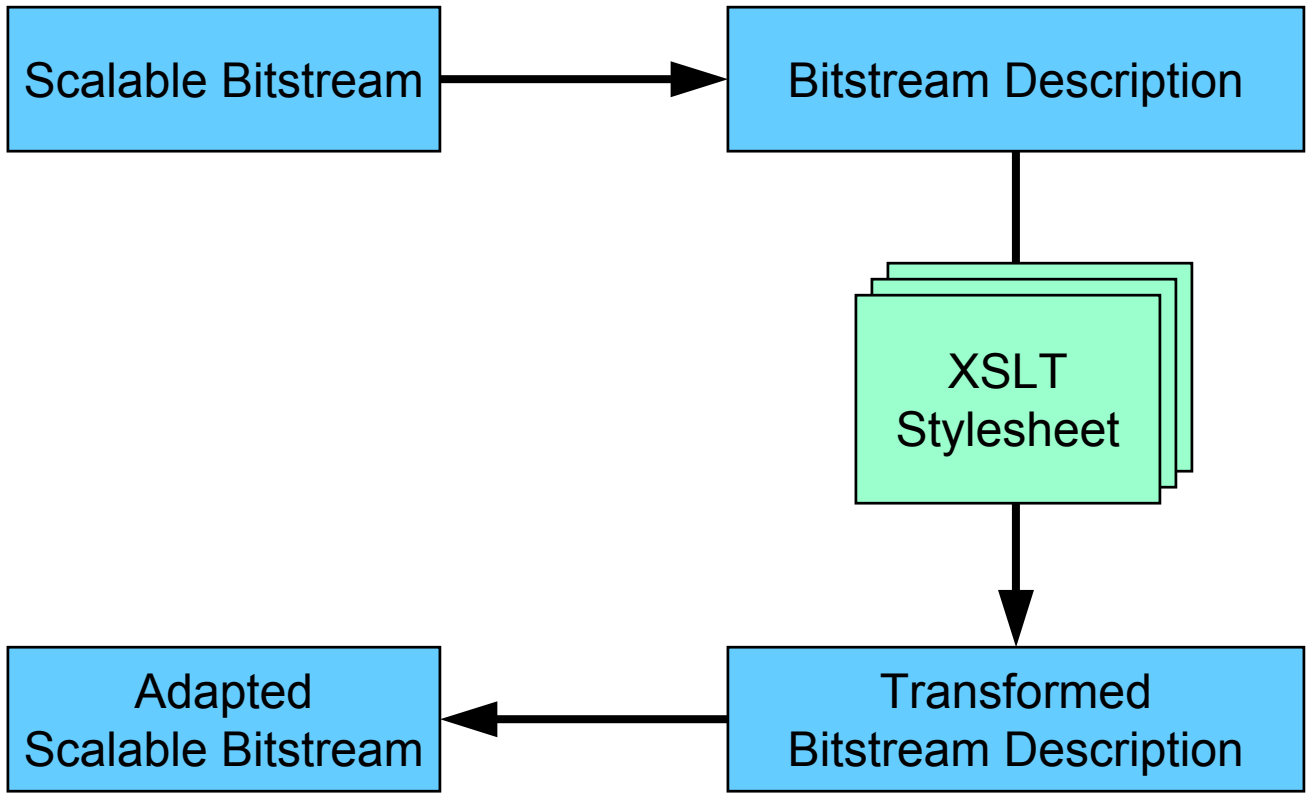
Scalable Content Adaptation with XML



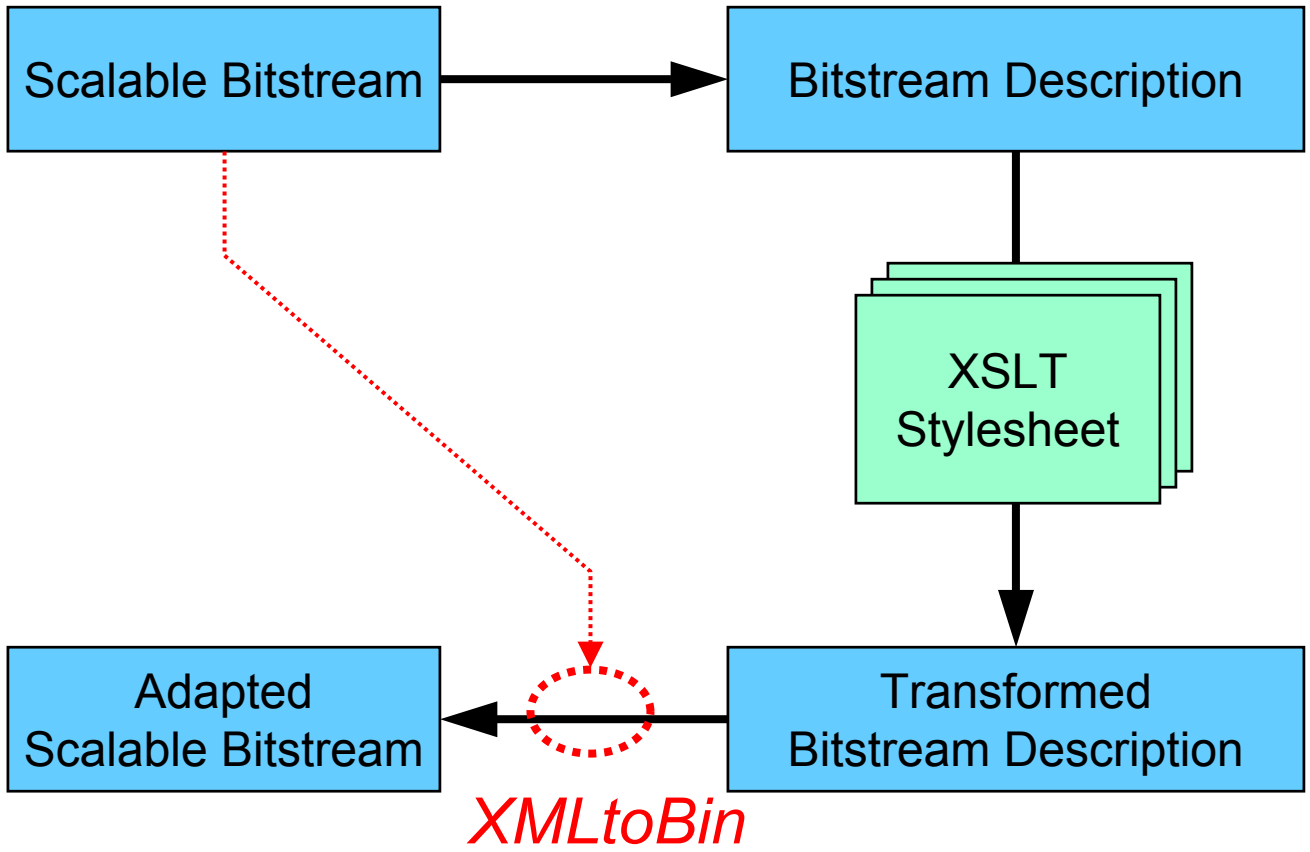
Scalable Content Adaptation with XML



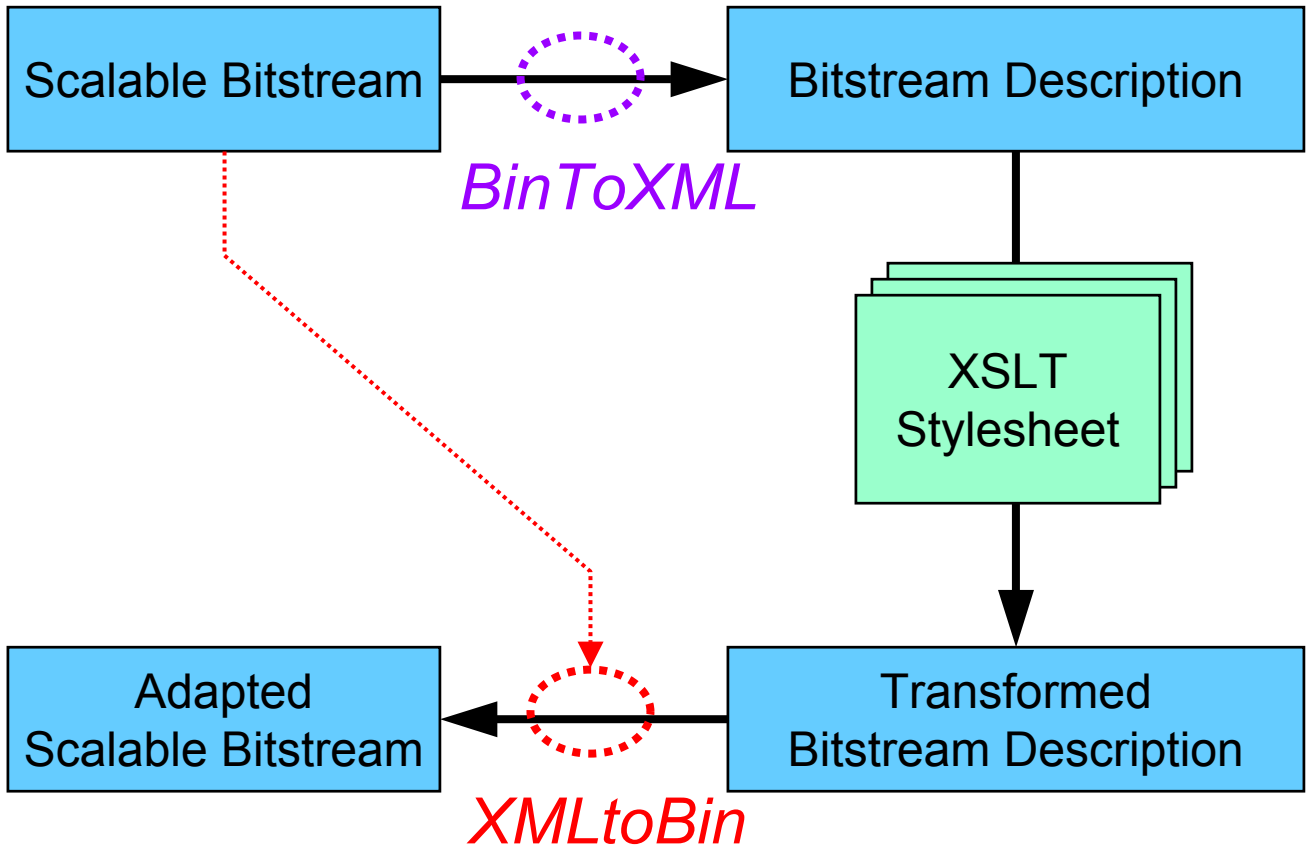
Scalable Content Adaptation with XML



Scalable Content Adaptation with XML



Scalable Content Adaptation with XML



Need for a document model

XML Bitstream Description

```
<Codestream>
  <MainHeader>
    <SOC>
      <Marker>ff4f</Marker>
    </SOC>
    <SIZ>
      <Marker>ff51</Marker>
      <Lsiz>47</Lsiz>
      <Xsiz>512</Xsiz>
    </SIZ>
  </MainHeader>
</Codestream>
```

Need for a document model

XML Bitstream Description

```
<Codestream>
  <MainHeader>
    <SOC>
      <Marker>ff4f</Marker>
    </SOC>
    <SIZ>
      <Marker>ff51</Marker>
      <Lsiz>47</Lsiz>
      <Xsiz>512</Xsiz>
    </SIZ>
  </MainHeader>
</Codestream>
```

- Some parameters need to be written in a readable format (for XSLT transformations)

Need for a document model

XML Bitstream Description

```
<Codestream>
  <MainHeader>
    <SOC>
      <Marker>ff4f</Marker>
    </SOC>
    <SIZ>
      <Marker>ff51</Marker>
      <Lsiz>47</Lsiz>
      <Xsiz>512</Xsiz>
    </SIZ>
  </MainHeader>
</Codestream>
```

How many bits in the bitstream ?

- Some parameters need to be written in a readable format (for XSLT transformations)
- How to binarize these values ?

Need for a document model

XML Bitstream Description

```
<Codestream>
  <MainHeader>
    <SOC>
      <Marker>ff4f</Marker>
    </SOC>
    <SIZ>
      <Marker>ff51</Marker>
      <Lsiz>47</Lsiz>
      <Xsiz>512</Xsiz>
    </SIZ>
  </MainHeader>
</Codestream>
```

How many bits in the bitstream ?

- Some parameters need to be written in a readable format (for XSLT transformations)
- How to binarize these values ?

→ Need for a document model

→ Efficient for data type specification

Need for a document model

XML Bitstream Description

```
<Codestream>
  <MainHeader>
    <SOC>
      <Marker>ff4f</Marker>
    </SOC>
    <SIZ>
      <Marker>ff51</Marker>
      <Lsiz>47</Lsiz>
      <Xsiz>512</Xsiz>
    </SIZ>
  </MainHeader>
</Codestream>
```

How many bits in the bitstream ?

- Some parameters need to be written in a readable format (for XSLT transformations)
- How to binarize these values ?

→ Need for a document model

→ Efficient for data type specification

→ XML-schema

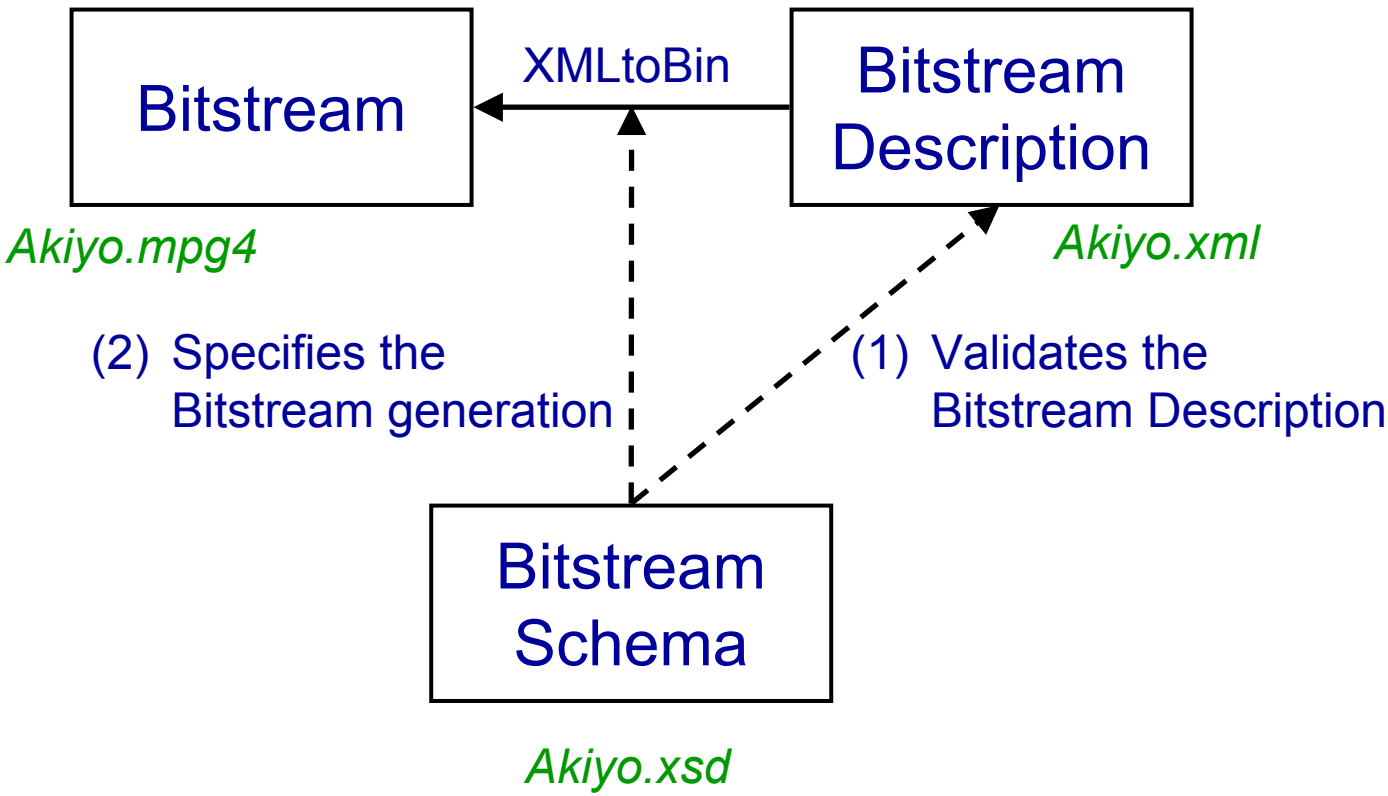
Bitstream Syntax Description Language

To design a new technical framework based on XML for describing the structure of bitstreams,

we define a “schema language”:
“Bitstream Syntax Description Language”,

based on W3C XML-Schema (restrictions + extensions)

BSDL Functionality



BSDL: Embedded Data

Input BSDL Description

```
<Codestream>
  <MainHeader>
    <SOC>
      <Marker>ff4f</Marker>
    </SOC>
    <SIZ>
      <Marker>ff51</Marker>
      <Lsiz>47</Lsiz>
      <Xsiz>512</Xsiz>
    </SIZ>
  </MainHeader>
</Codestream>
```

BSDL Schema

```
<xsd:complexType name="SOCType">
  <xsd:sequence>
    <xsd:element name="Marker"
      type="xsd:short"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="SIZType">
  <xsd:sequence>
    <xsd:element name="Marker"
      type="xsd: short "/>
    <xsd:element name="Lsiz"
      type="xsd:unsignedShort"/>
    <xsd:element name="Xsiz"
      type="xsd:unsignedInt"/>
  </xsd:sequence>
</xsd:complexType>
```

Output bitstream: `ff4f ff51 002f 00000200 ...`

BSDL: Embedded Data

Input BSDL Description

```
<Codestream>
  <MainHeader>
    <SOC>
      <Marker>ff4f</Marker>
    </SOC>
    <SIZ>
      <Marker>ff51</Marker>
      <Lsiz>47</Lsiz>
      <Xsiz>512</Xsiz>
    </SIZ>
  </MainHeader>
</Codestream>
```

BSDL Schema

```
<xsd:complexType name="SOCType">
  <xsd:sequence>
    <xsd:element name="Marker"
      type="xsd:short"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="SIZType">
  <xsd:sequence>
    <xsd:element name="Marker"
      type="xsd: short "/>
    <xsd:element name="Lsiz"
      type="xsd:unsignedShort"/>
    <xsd:element name="Xsiz"
      type="xsd:unsignedInt"/>
  </xsd:sequence>
</xsd:complexType>
```

Output bitstream: ff4f ff51 002f 00000200 ...

BSDL: Embedded Data

Input BSDL Description

```
<Codestream>
  <MainHeader>
    <SOC>
      <Marker>ff4f</Marker>
    </SOC>
    <SIZ>
      <Marker>ff51</Marker>
      <Lsiz>47</Lsiz>
      <Xsiz>512</Xsiz>
    </SIZ>
  </MainHeader>
</Codestream>
```

BSDL Schema

```
<xsd:complexType name="SOCType">
  <xsd:sequence>
    <xsd:element name="Marker"
      type="xsd:short"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="SIZType">
  <xsd:sequence>
    <xsd:element name="Marker"
      type="xsd: short "/>
    <xsd:element name="Lsiz"
      type="xsd:unsignedShort"/>
    <xsd:element name="Xsiz"
      type="xsd:unsignedInt"/>
  </xsd:sequence>
</xsd:complexType>
```

Output bitstream: `ff4f ff51 002f 00000200 ...`

BSDL: Embedded Data

Input BSDL Description

```
<Codestream>
  <MainHeader>
    <SOC>
      <Marker>ff4f</Marker>
    </SOC>
    <SIZ>
      <Marker>ff51</Marker>
      <Lsiz>47</Lsiz>
      <Xsiz>512</Xsiz>
    </SIZ>
  </MainHeader>
</Codestream>
```

BSDL Schema

```
<xsd:complexType name="SOCType">
  <xsd:sequence>
    <xsd:element name="Marker"
      type="xsd:short"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="SIZType">
  <xsd:sequence>
    <xsd:element name="Marker"
      type="xsd: short "/>
    <xsd:element name="Lsiz"
      type="xsd:unsignedShort"/>
    <xsd:element name="Xsiz"
      type="xsd:unsignedInt"/>
  </xsd:sequence>
</xsd:complexType>
```

Output bitstream: `ff4f ff51 002f 00000200 ...`

BSDL: Embedded Data

Input BSDL Description

```
<Codestream>
  <MainHeader>
    <SOC>
      <Marker>ff4f</Marker>
    </SOC>
    <SIZ>
      <Marker>ff51</Marker>
      <Lsiz>47</Lsiz>
      <Xsiz>512</Xsiz>
    </SIZ>
  </MainHeader>
</Codestream>
```

BSDL Schema

```
<xsd:complexType name="SOCType">
  <xsd:sequence>
    <xsd:element name="Marker"
      type="xsd:short"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="SIZType">
  <xsd:sequence>
    <xsd:element name="Marker"
      type="xsd: short "/>
    <xsd:element name="Lsiz"
      type="xsd:unsignedShort"/>
    <xsd:element name="Xsiz"
      type="xsd:unsignedInt"/>
  </xsd:sequence>
</xsd:complexType>
```

Output bitstream: `ff4f ff51 002f 00000200 ...`

BSDL: External Entity

Input BSDL Description

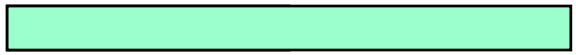
```
<Bitstream xml:base="myImage.jp2">
<PacketData>122-363</PacketData>
<PacketData>370-861</PacketData>
```

BSDL Schema

```
<xsd:element name="PacketData"
     type="bsdl:byteRange"/>
```

Input bitstream:

Output bitstream:



BSDL: External Entity

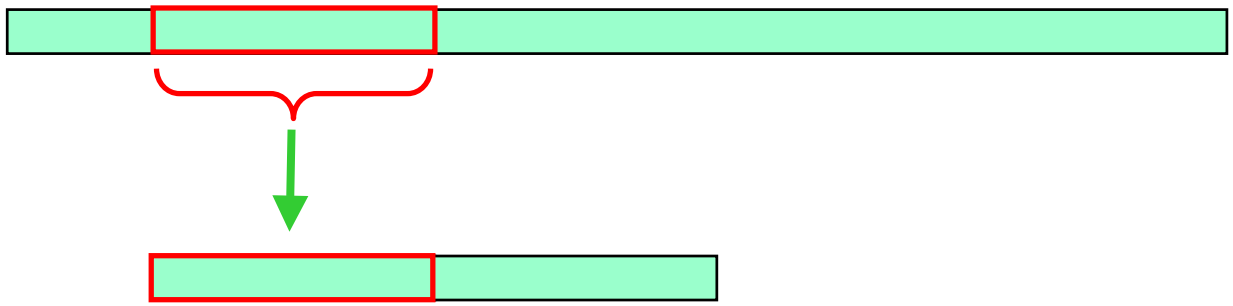
Input BSDL Description

```
<Bitstream xml:base="myImage.jp2">
<PacketData>122-363</PacketData>
<PacketData>370-861</PacketData>
```

BSDL Schema

```
<xsd:element name="PacketData"
     type="bsdl:byteRange"/>
```

Input bitstream:



Output bitstream:

BSDL: External Entity

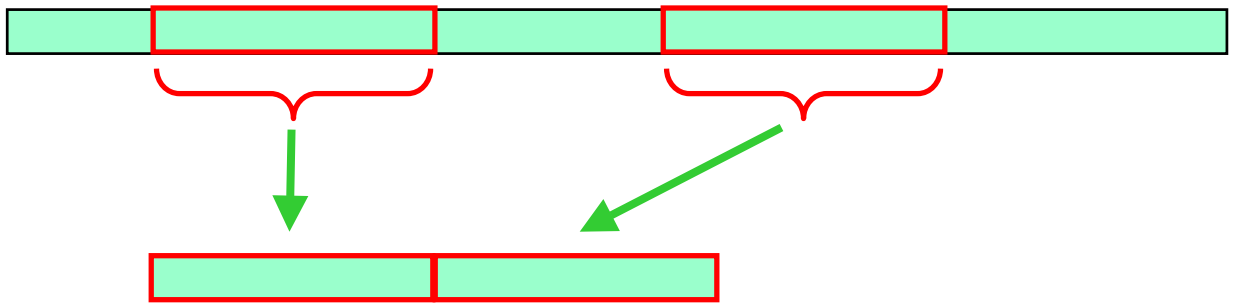
Input BSDL Description

```
<Bitstream xml:base="myImage.jp2">
<PacketData>122-363</PacketData>
<PacketData>370-861</PacketData>
```

BSDL Schema

```
<xsd:element name="PacketData"
     type="bsdl:byteRange"/>
```

Input bitstream:



Output bitstream:

BSDL: Restrictions on Structures

- Data shall be embedded in elements (and not in attributes)
- A type must be assigned to each element of the instance

BSDL: Restrictions on Datatypes

- Use only XML-Schema datatypes having an implicit binary representation.

  `xsd:integer` = unbounded integer
  ➔ excluded

  `xsd:int` = bounded integer
  ➔ included, binary representation on 4 bytes

BSDL: Extensions on Datatypes

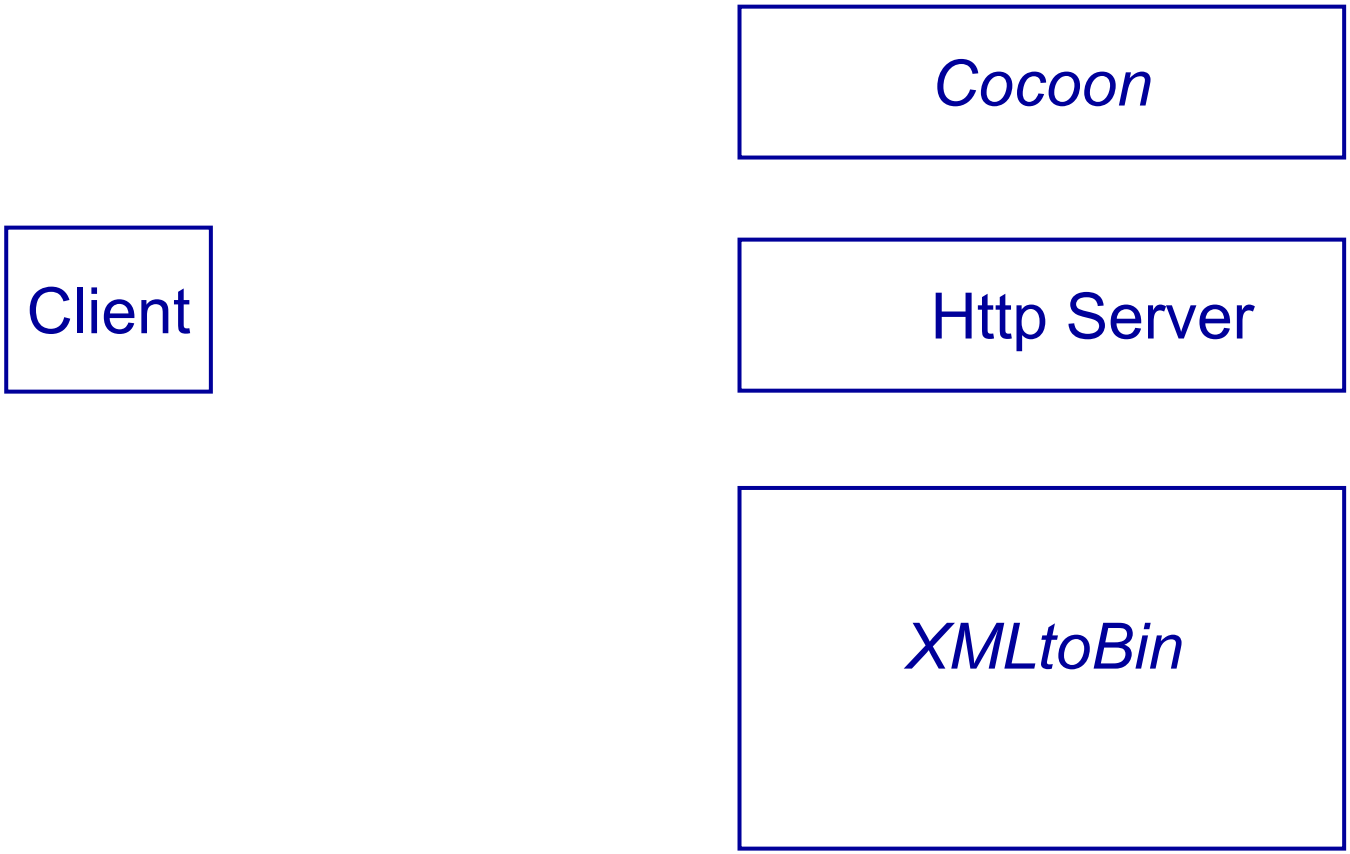
- BSDL Pseudo-facets

  bsdl:bitsLength = indicates the number of bits on which the element should be encoded

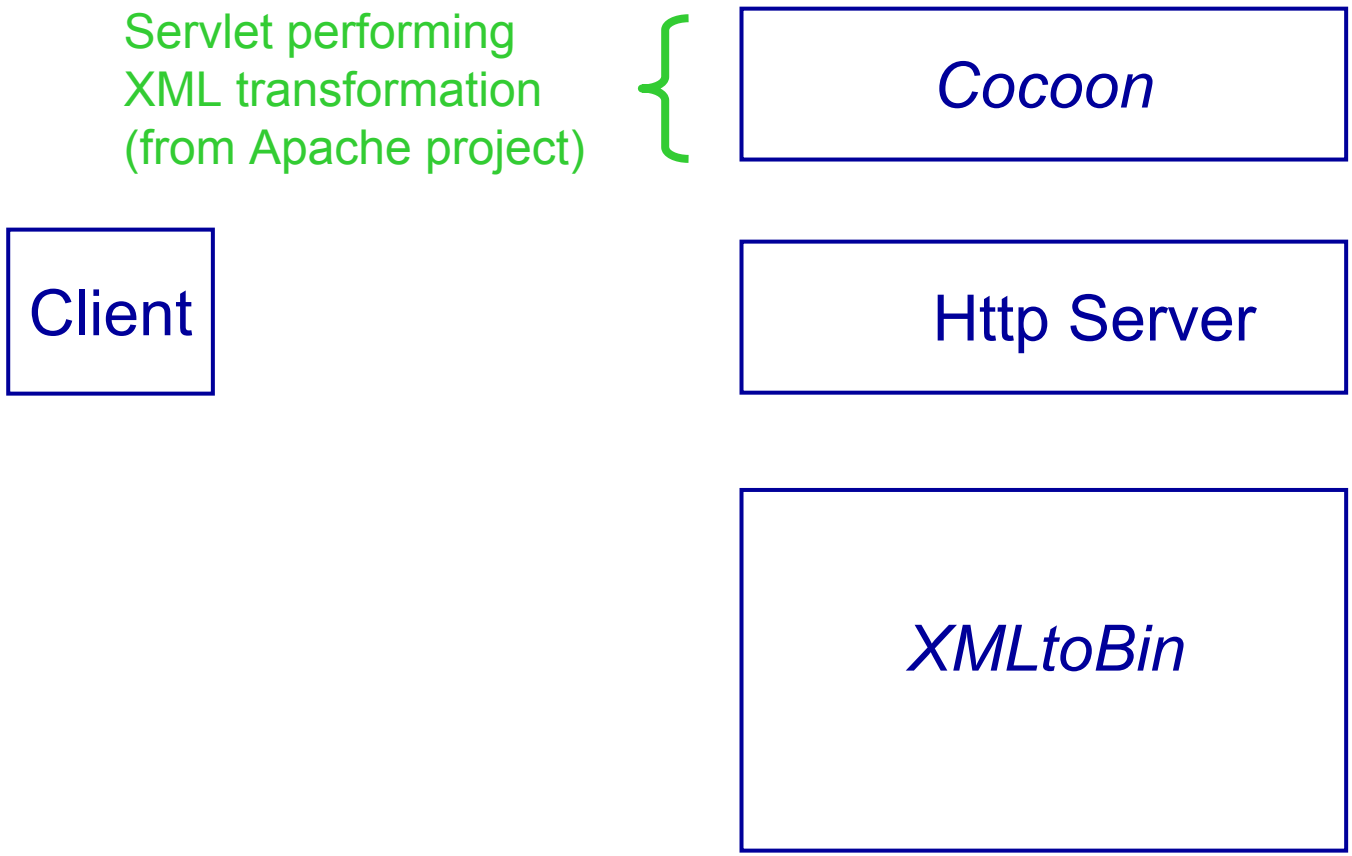
- XML-Schema extension mechanism

```
<xsd:simpleType>
  <xsd:restriction base="xsd:unsignedByte">
     <xsd:annotation><xsd:appinfo>
        <bsdl:bitsLength>
     </xsd:appinfo></xsd:annotation>
  </xsd:restriction>
</xsd:simpleType>
```

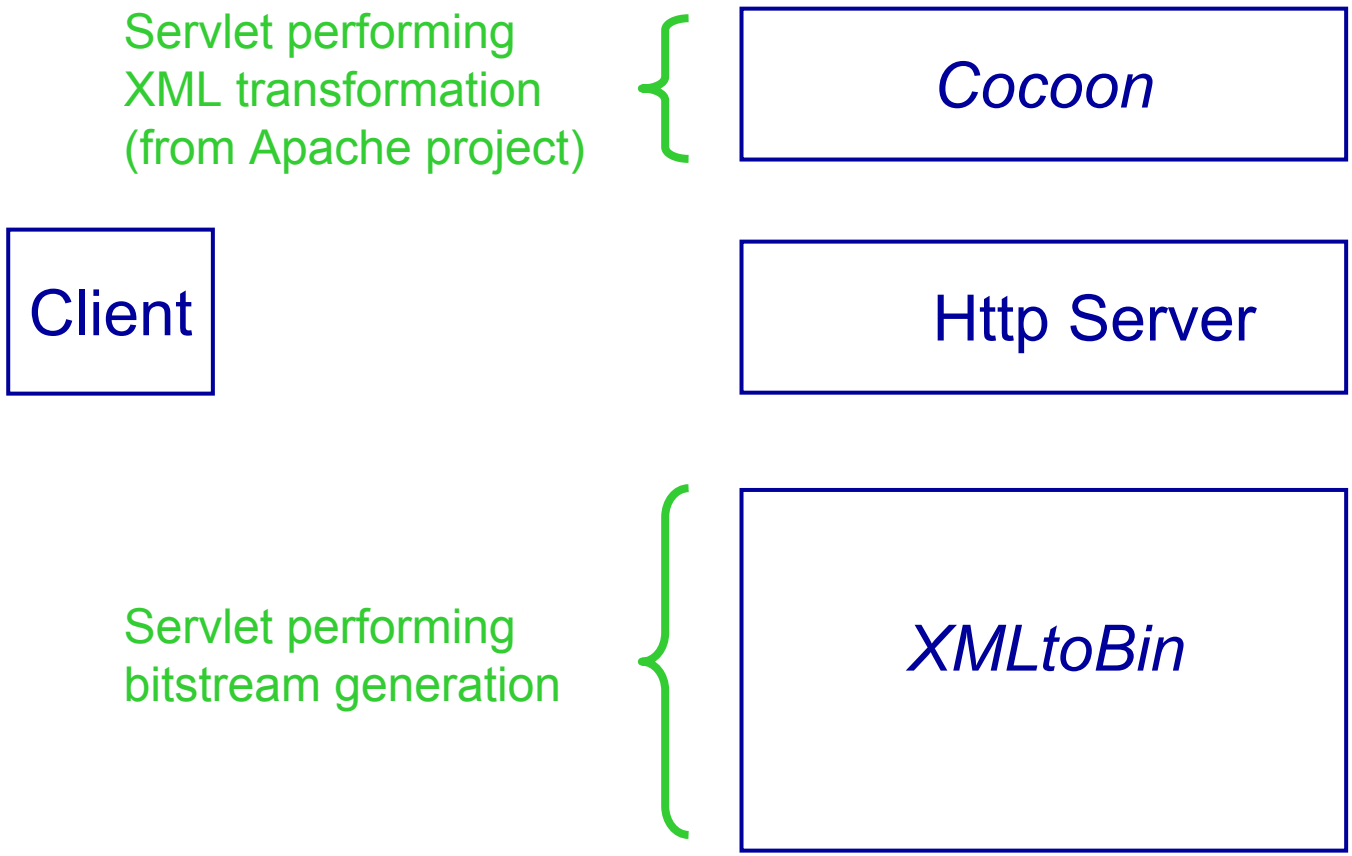
Demo: Implementation on a Web Architecture



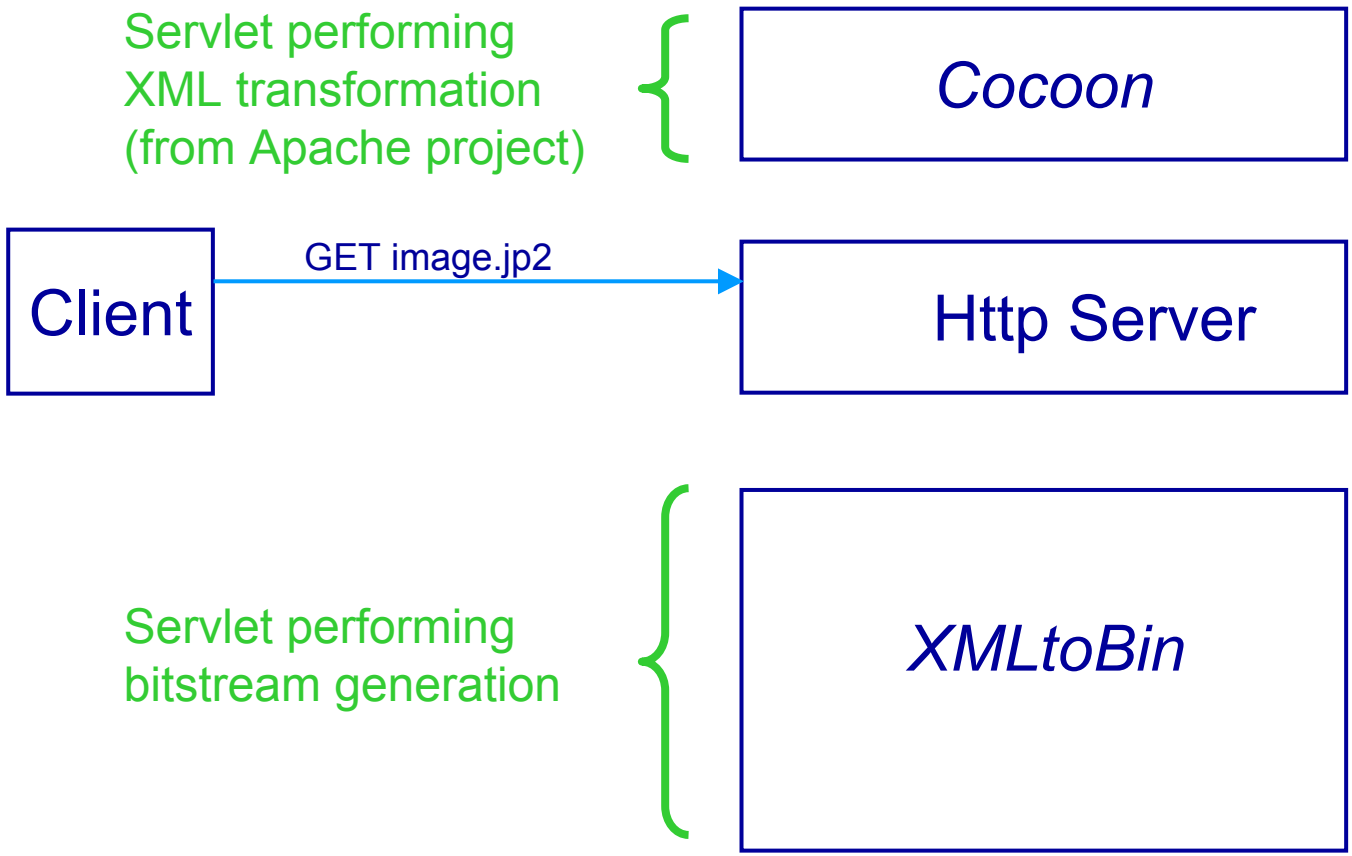
Demo: Implementation on a Web Architecture



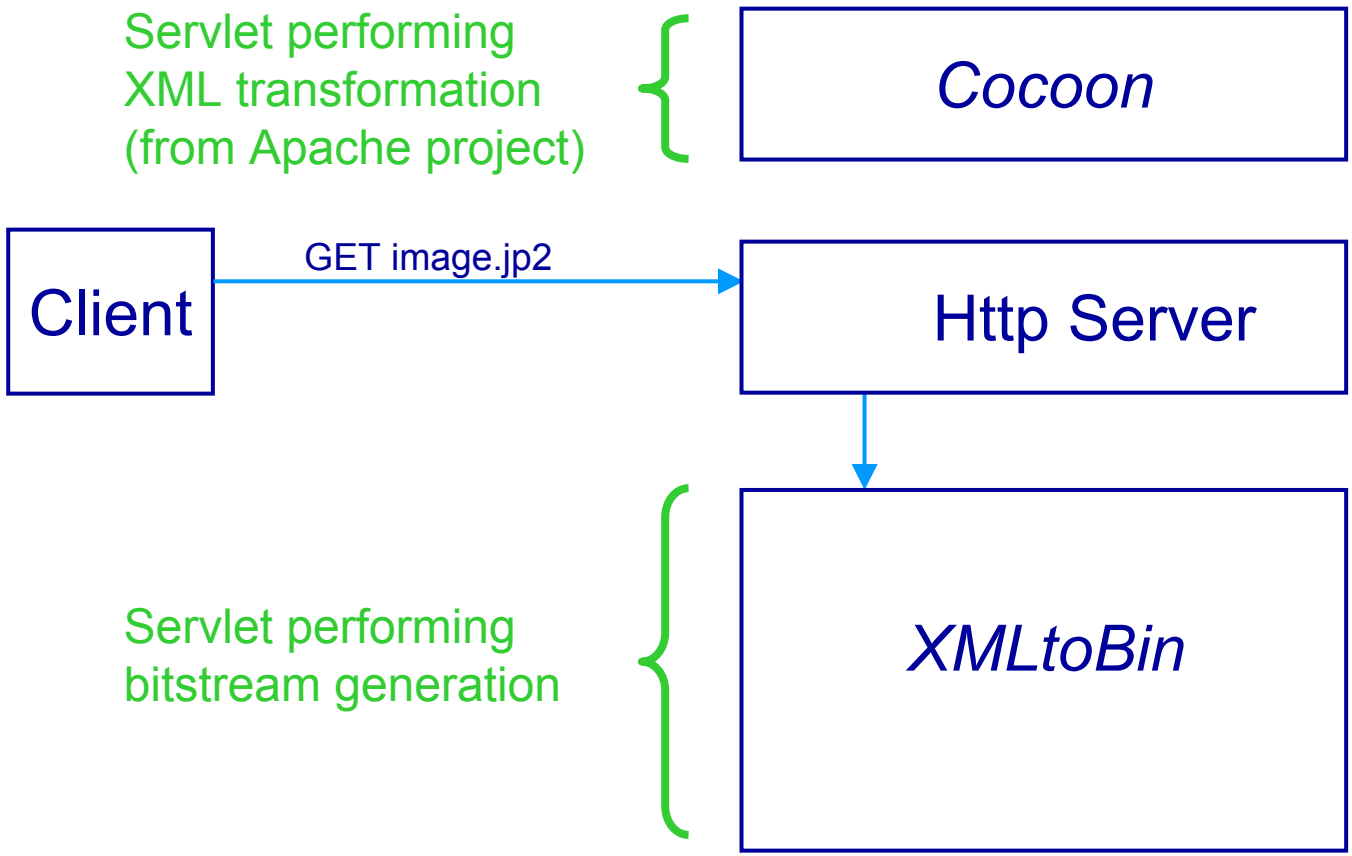
Demo: Implementation on a Web Architecture



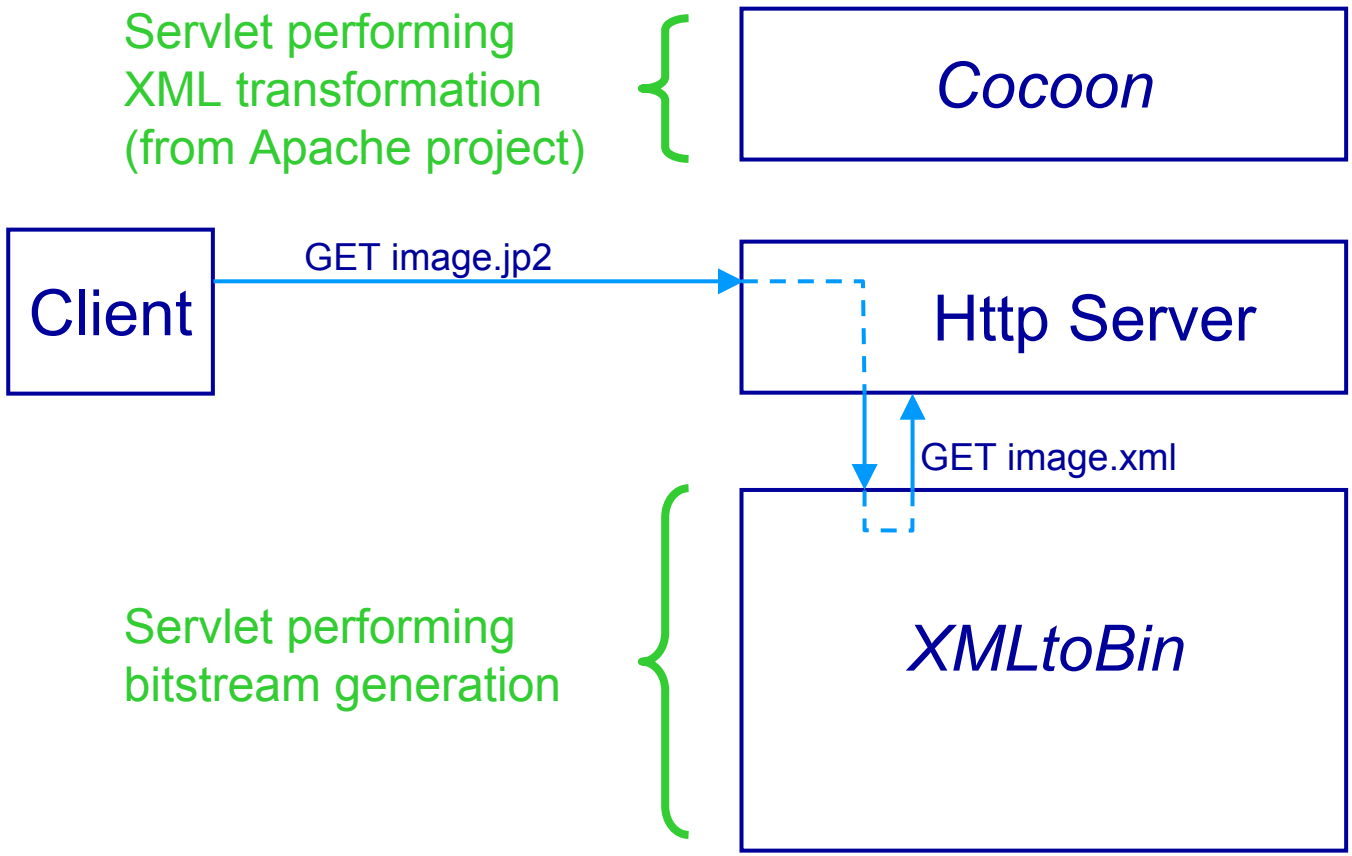
Demo: Implementation on a Web Architecture



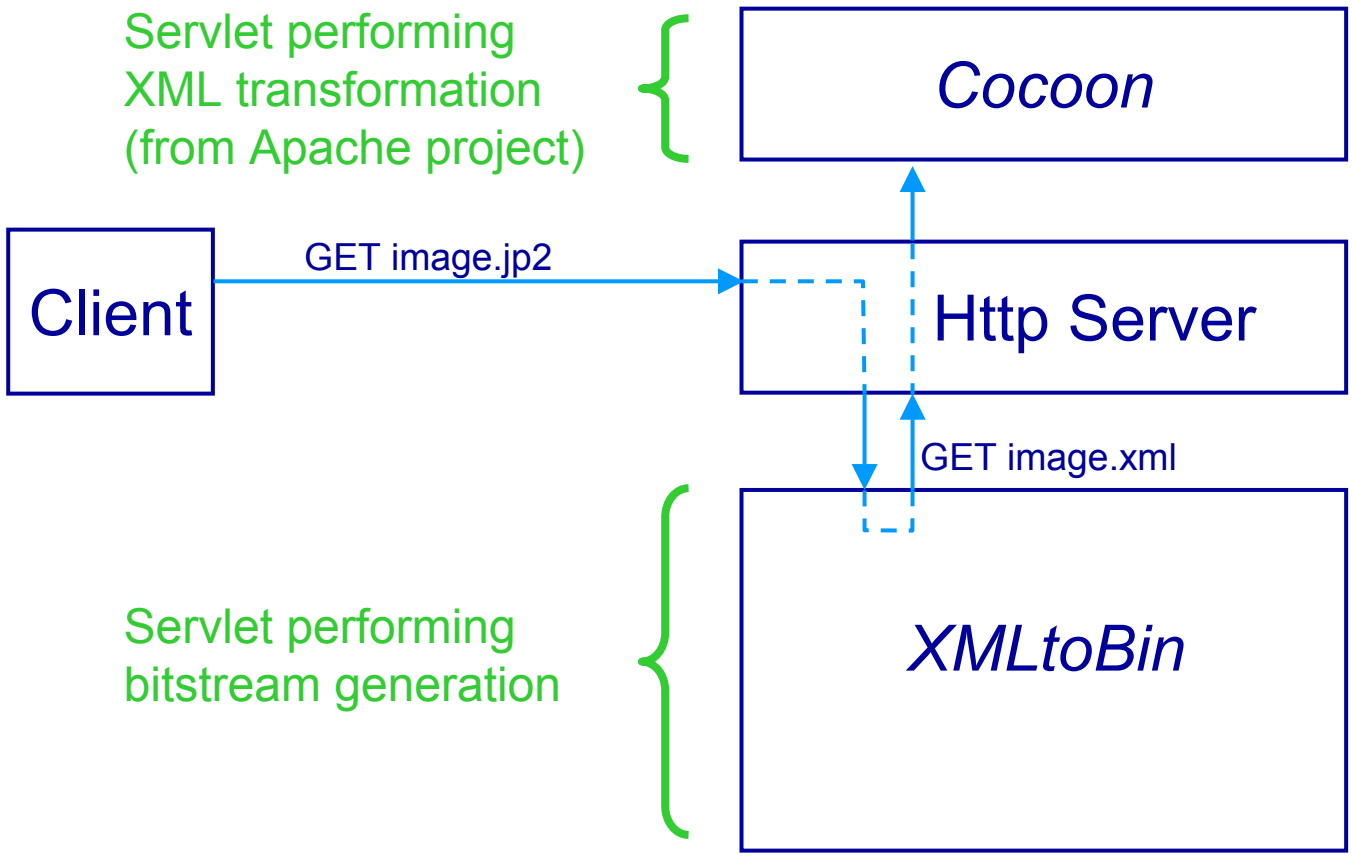
Demo: Implementation on a Web Architecture



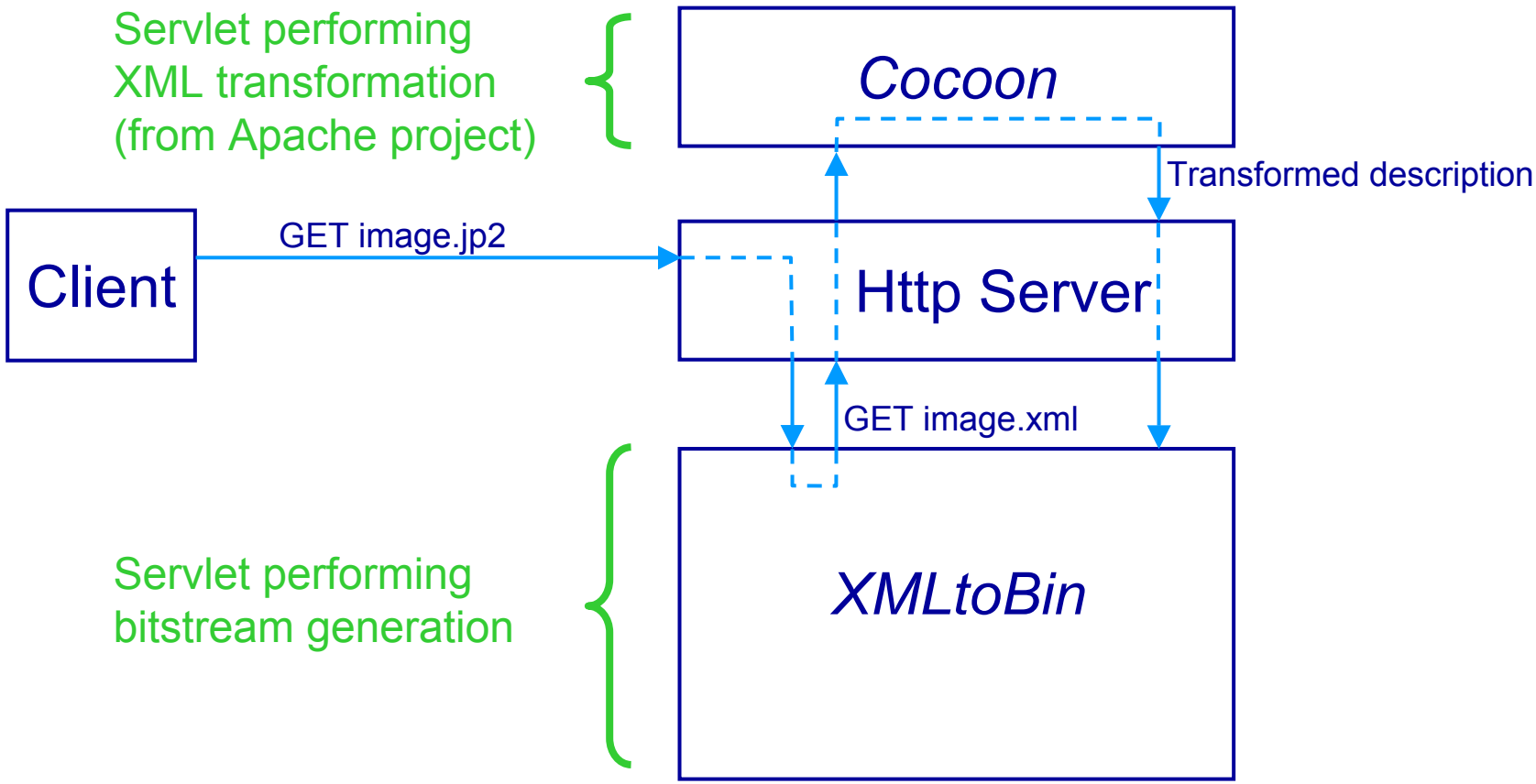
Demo: Implementation on a Web Architecture



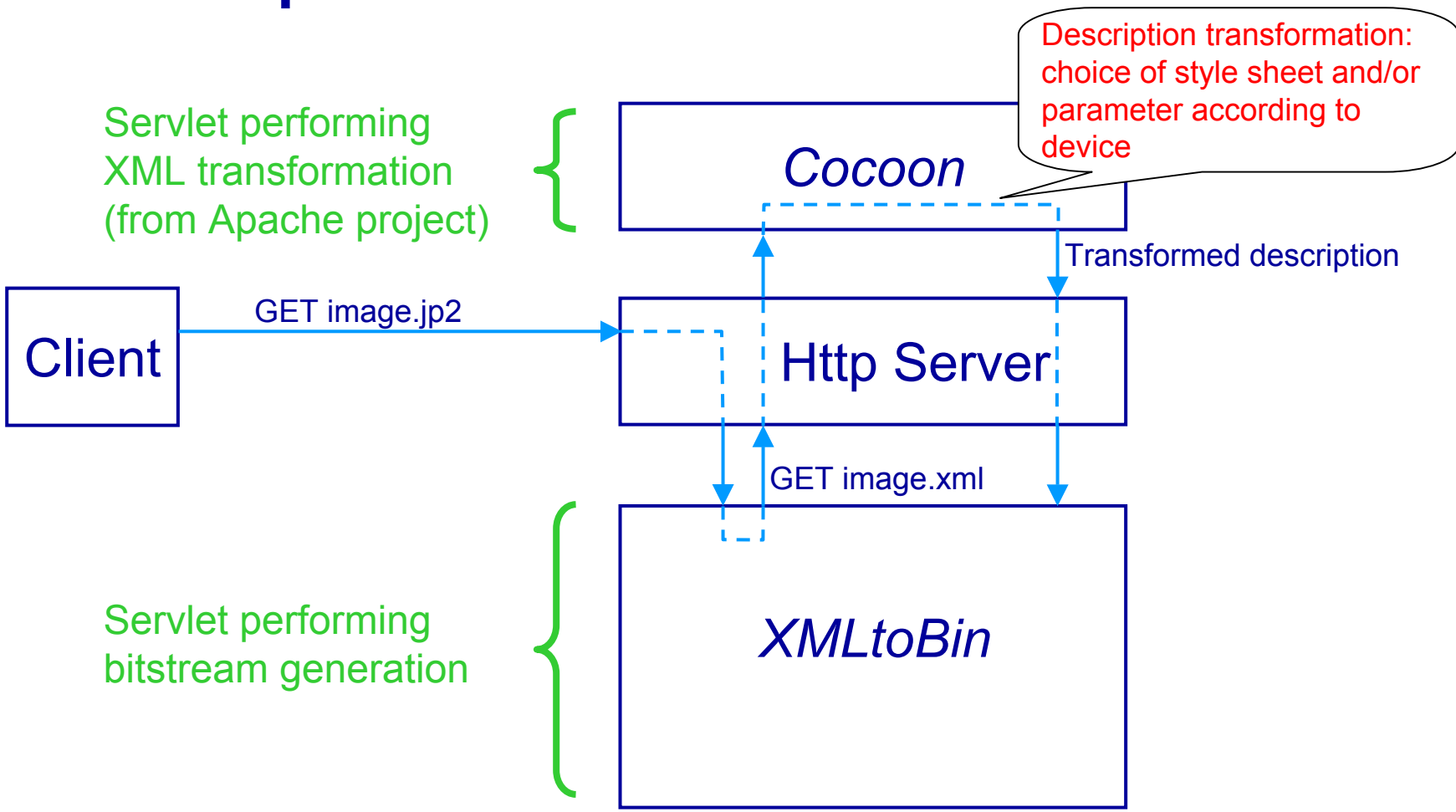
Demo: Implementation on a Web Architecture



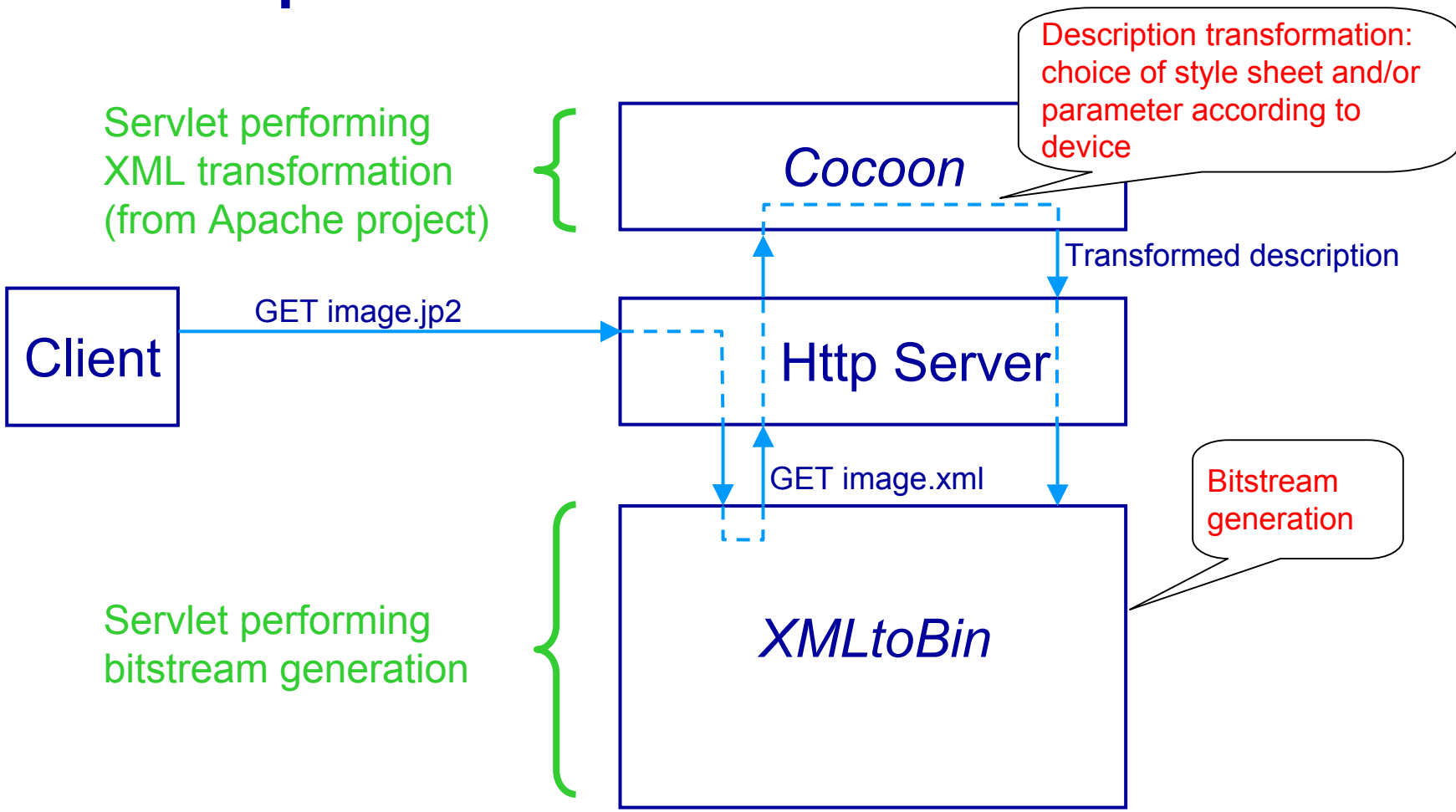
Demo: Implementation on a Web Architecture



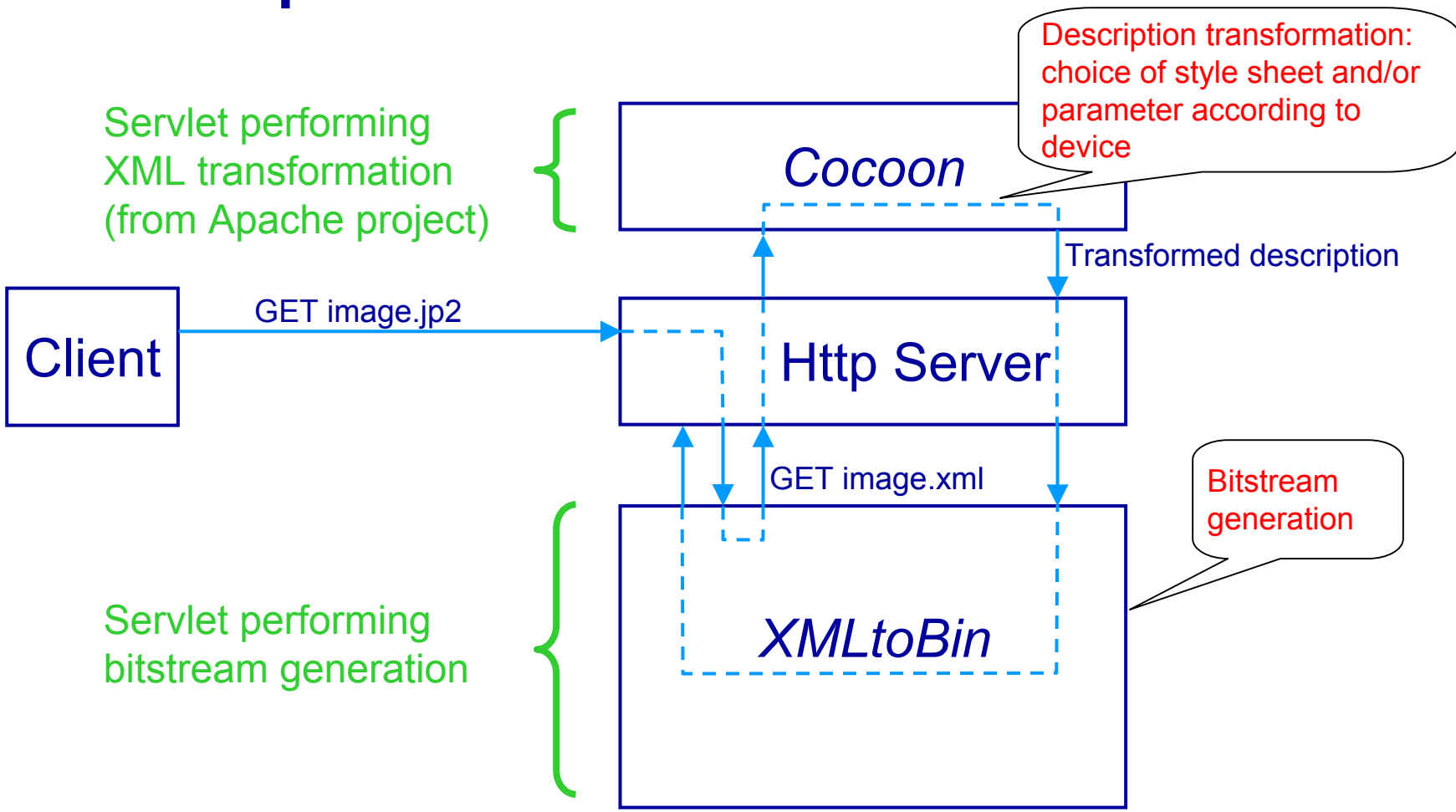
Demo: Implementation on a Web Architecture



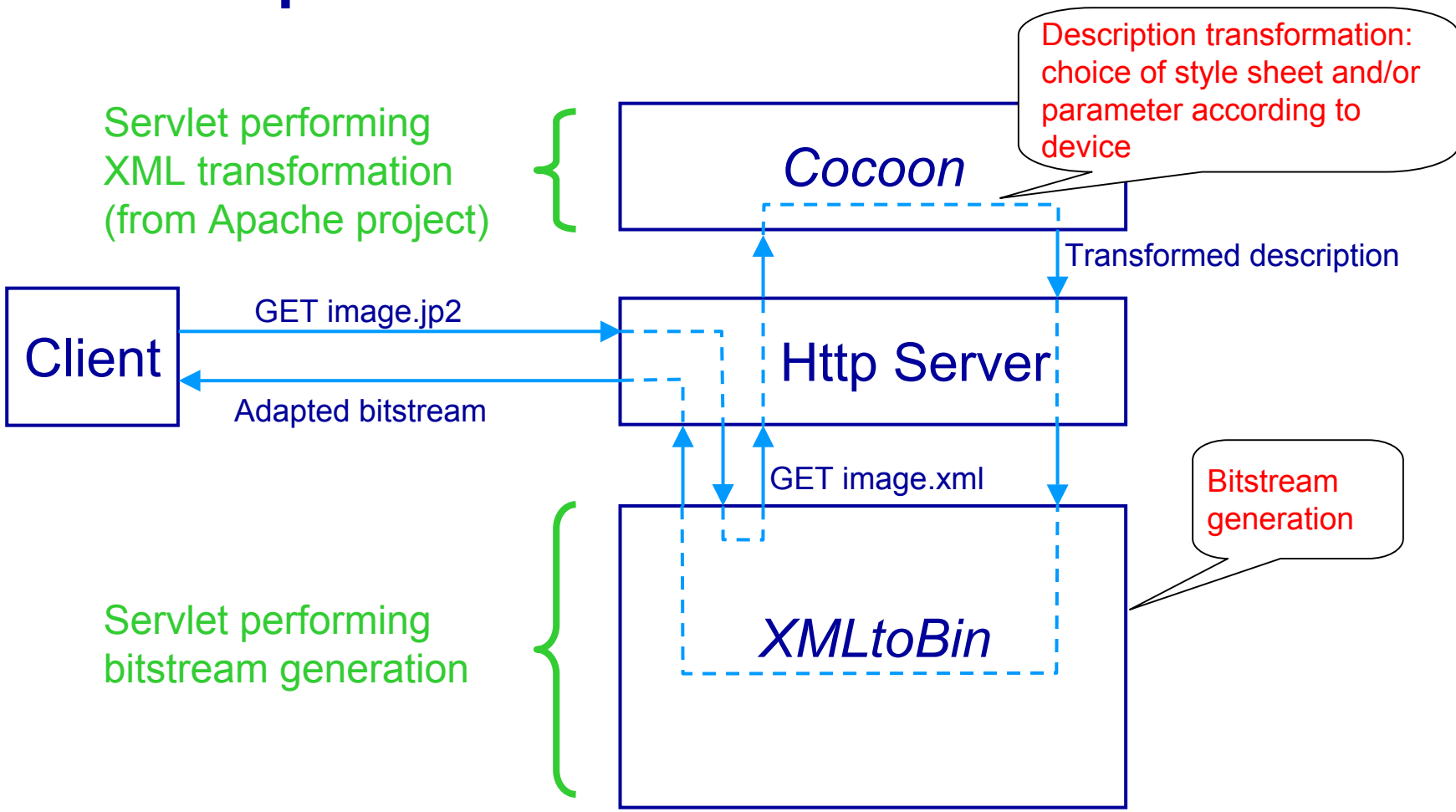
Demo: Implementation on a Web Architecture



Demo: Implementation on a Web Architecture



Demo: Implementation on a Web Architecture



Conclusion and future works

- Cross-standard and flexible approach to handle scalable content
- Generic software XMLtoBin
- Scenario for a Client/Server system
- Further steps for BSDL: provide a grammar to parse a bitstream and to generate an XML description

Thank you for your attention

Philips Research France

Appendix

Bitstream Syntax Description Language

- New language based on W3C XML-Schema (restrictions + extensions)
- Allows to:
  - Validate (in the XML-Schema meaning) the instance (Bitstream Description) against its schema (Bitstream Schema)

    Level 0
  - Parse a Description and generate the Bitstream (XML to Bin)

    Level 1
  - Parse a Bitstream and generate its Description (Bin to XML)

    Level 2

BSDL level 2: Extensions on Structures

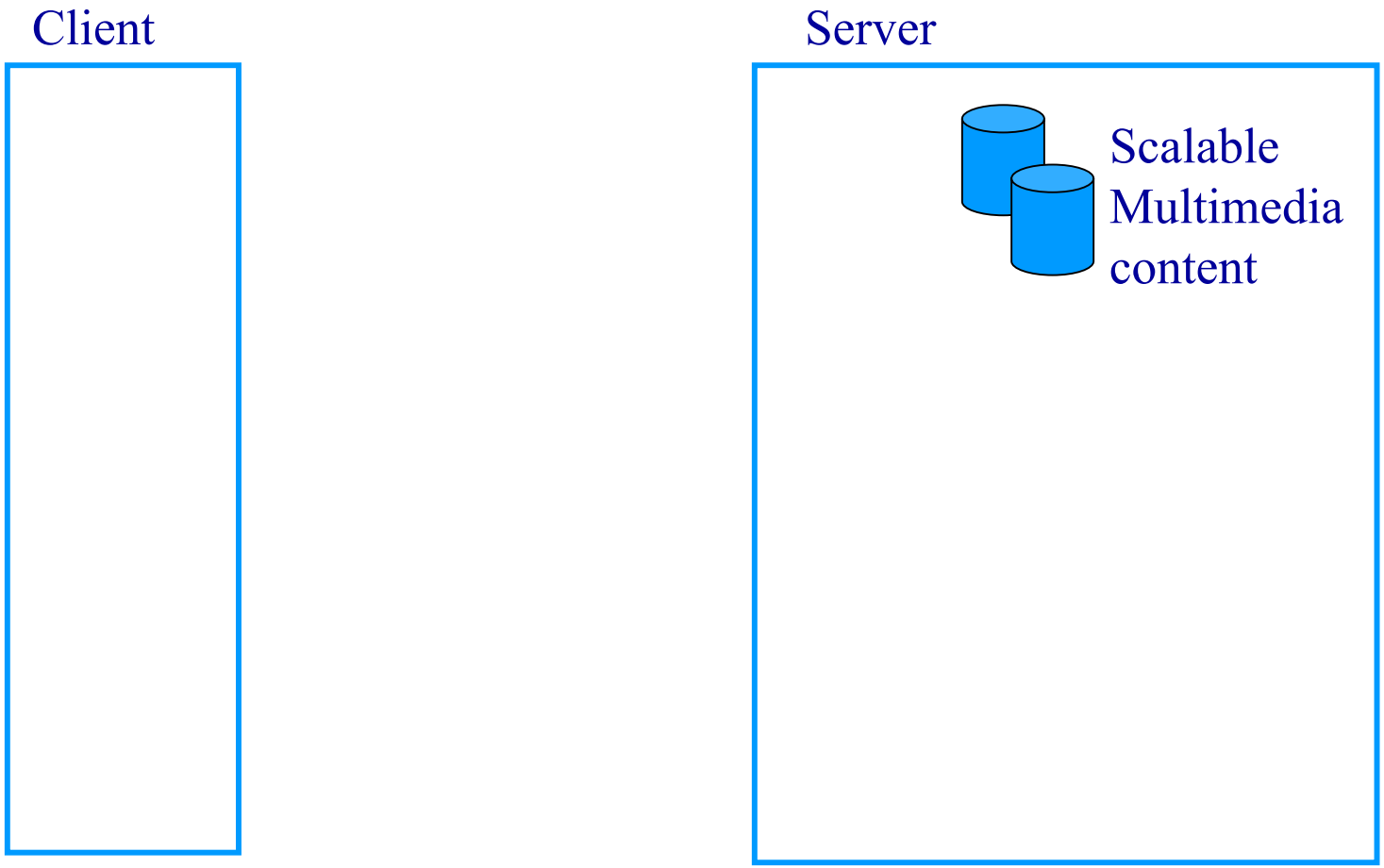
- Variables, to specify the occurrence number of a particle

```
<xsd:element name="NrOfPackets" type="xsd:short"/>
<xsd:element name="Packet" type="packetType"
        bsdl:nOccurs="NrOfPackets"/>
```

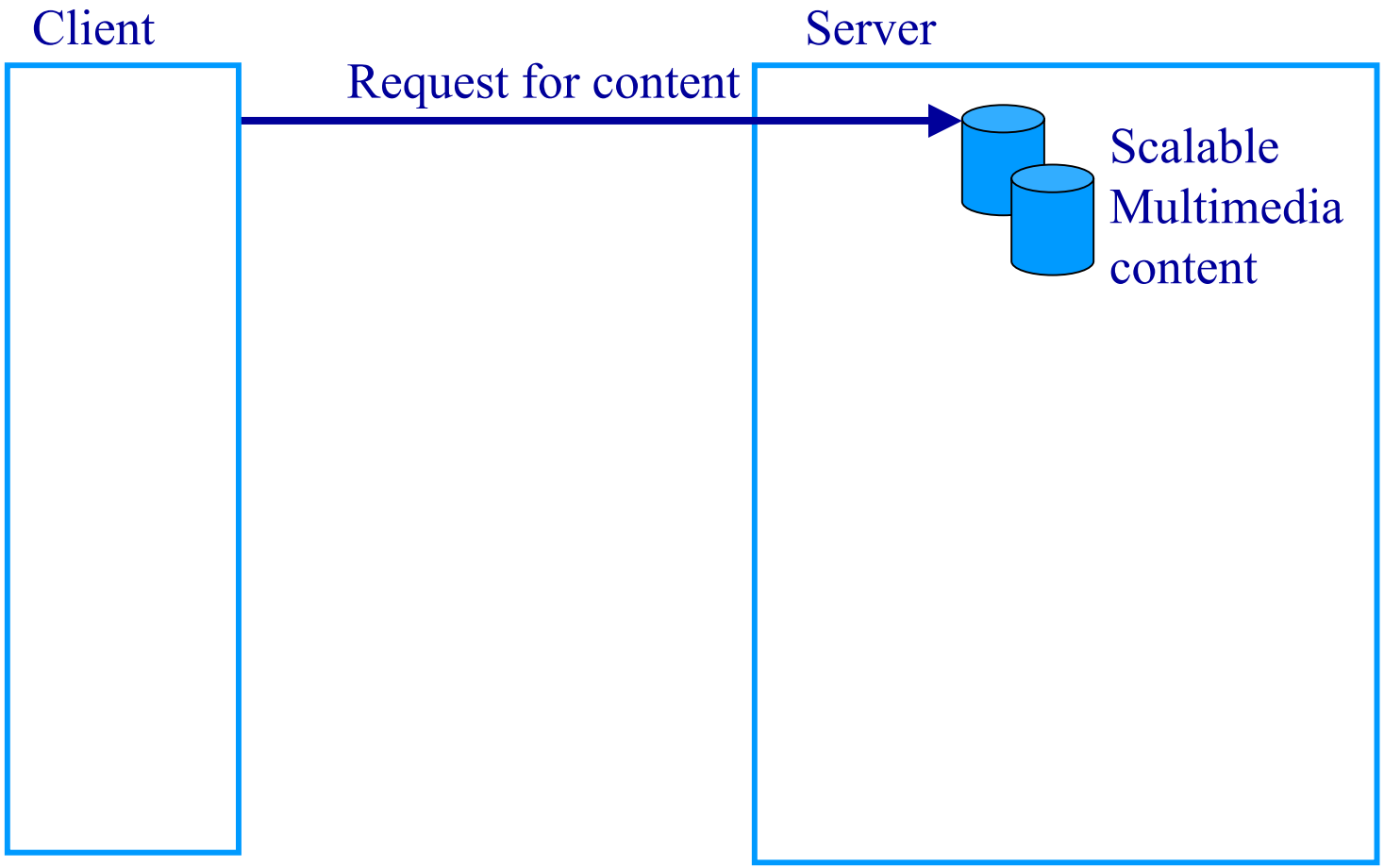
- Conditional statements

```
<xsd:element name="param" type="xsd:short"/>
<xsd:element name="elt"
        bsdl:if="param=3" minOccurs="0">
```

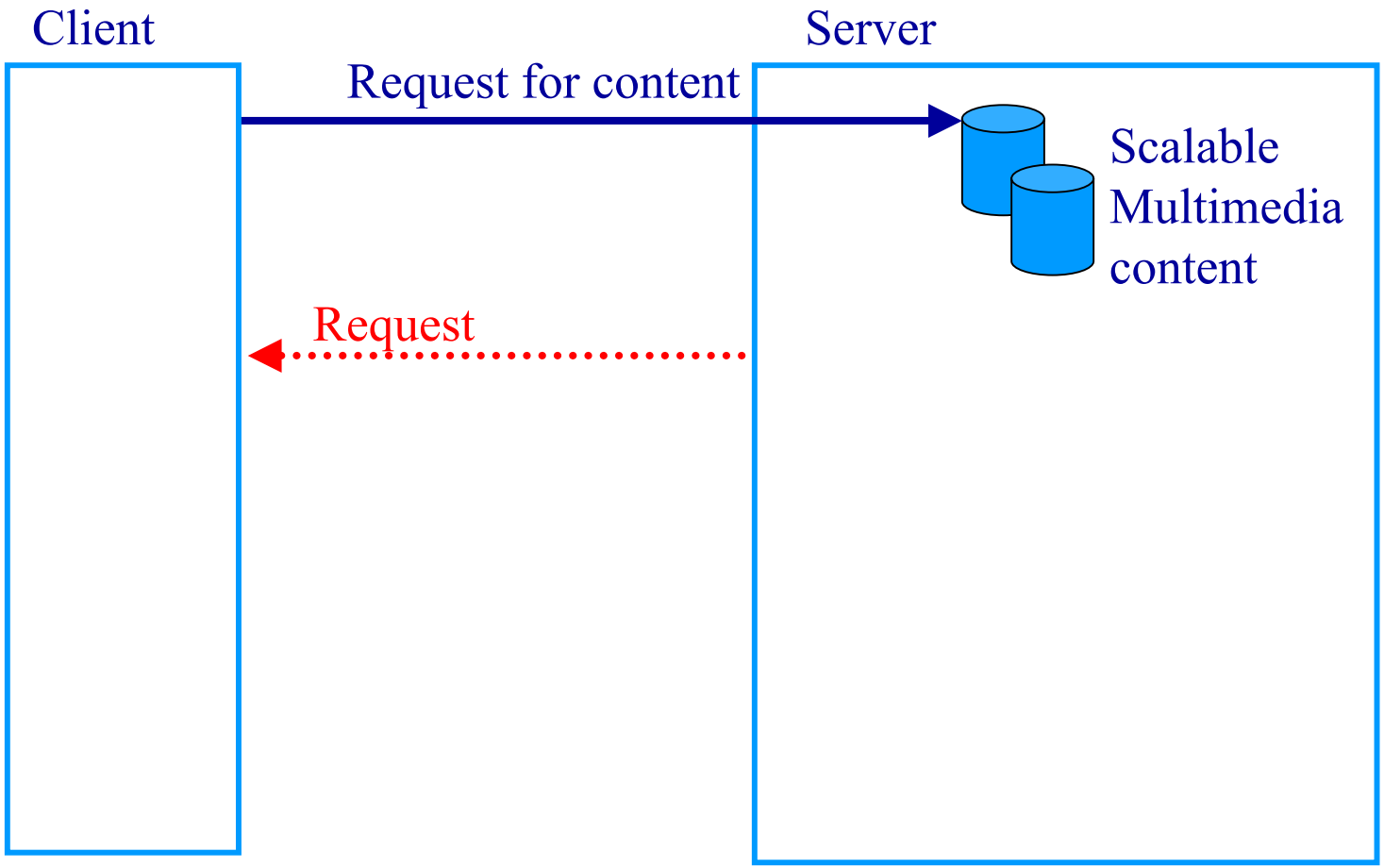
System Approach: a complete scenario



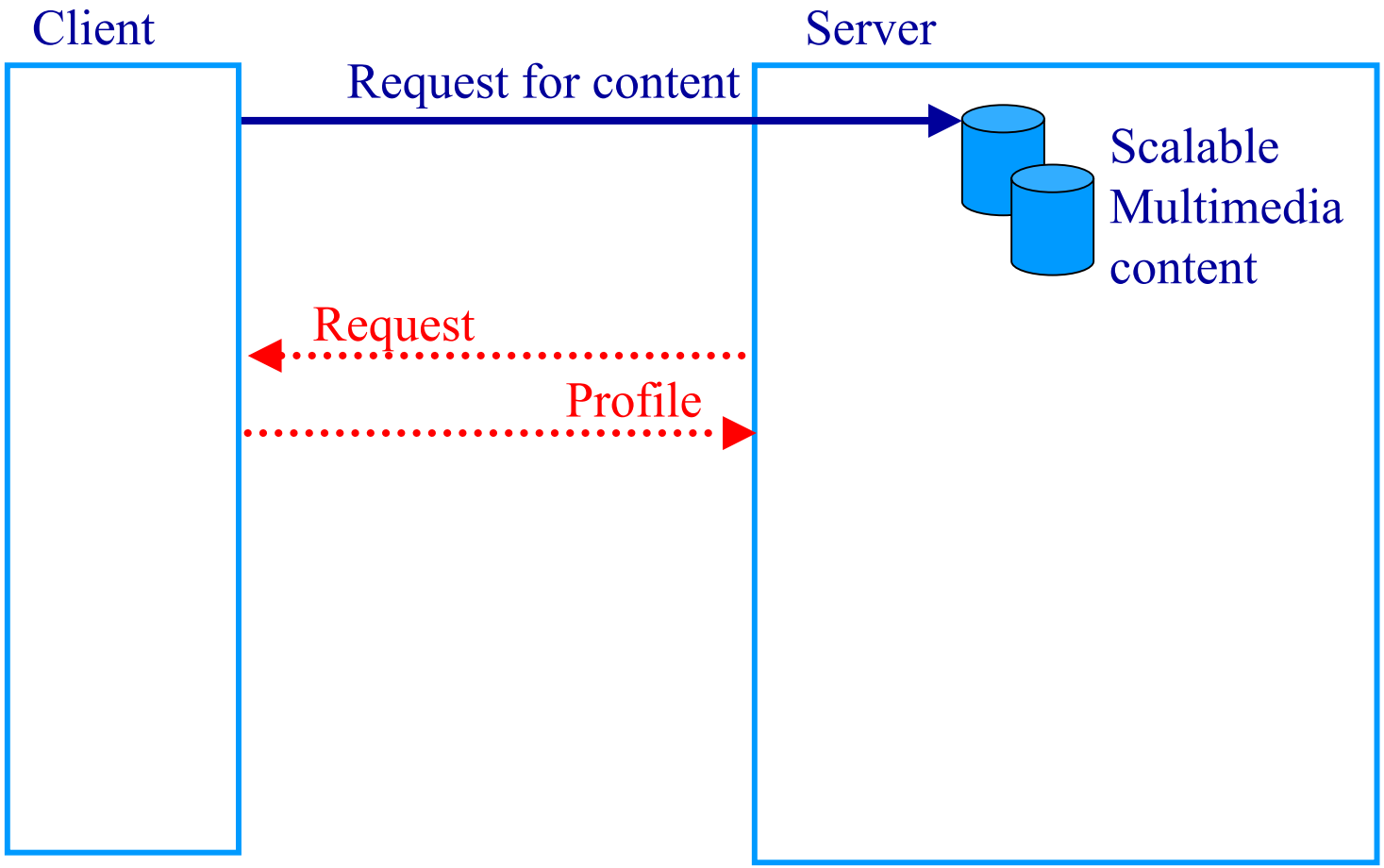
System Approach: a complete scenario



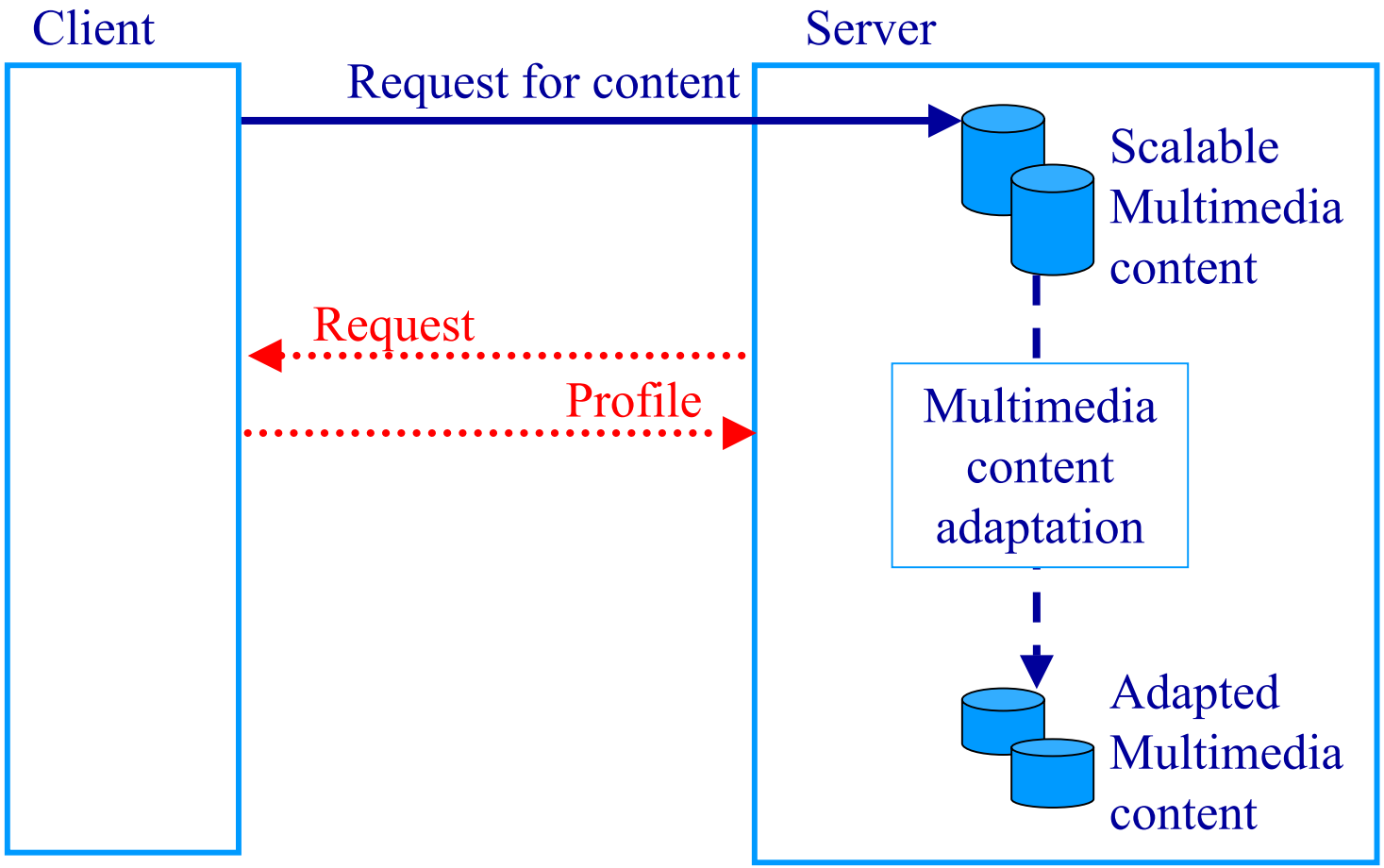
System Approach: a complete scenario



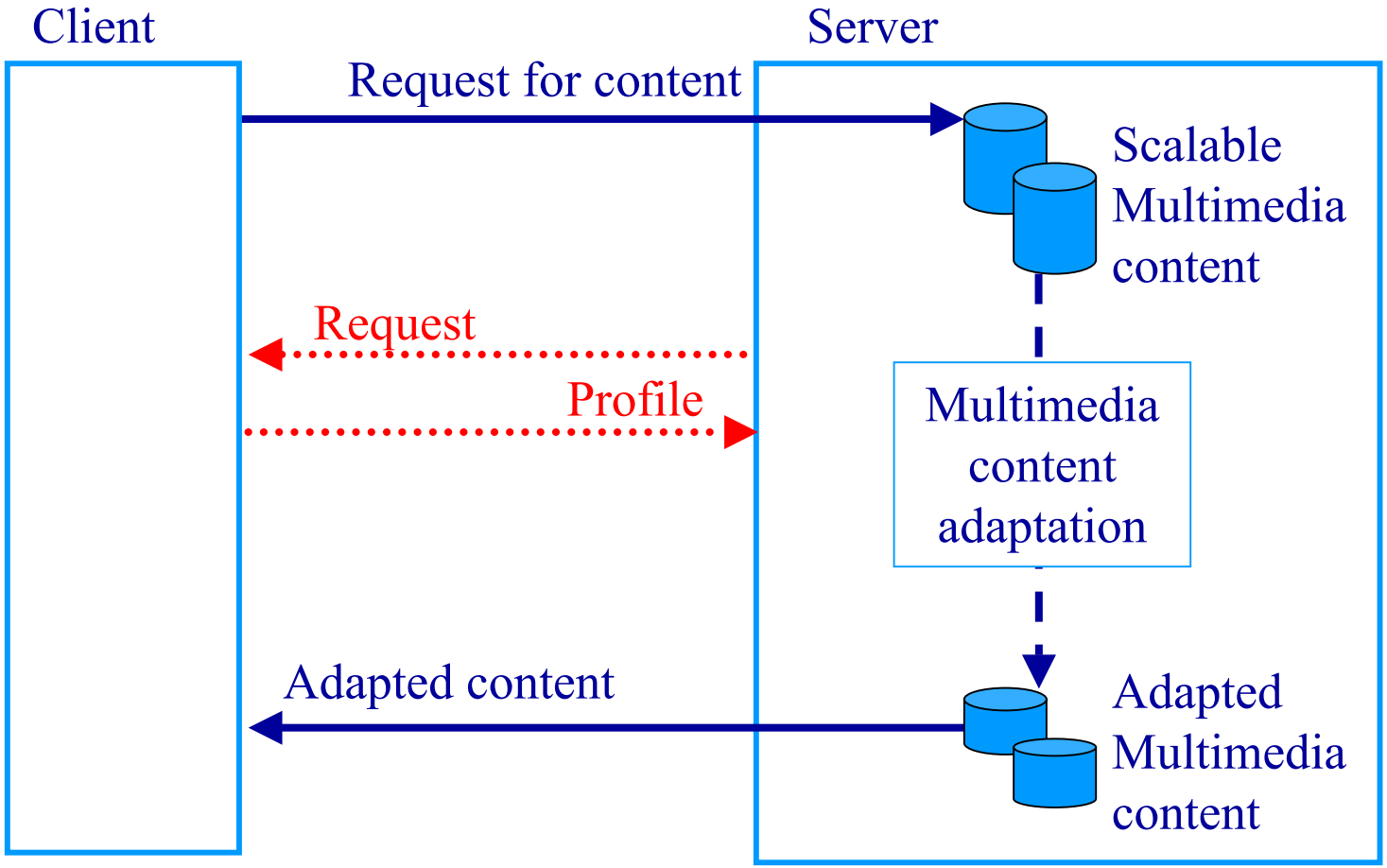
System Approach: a complete scenario



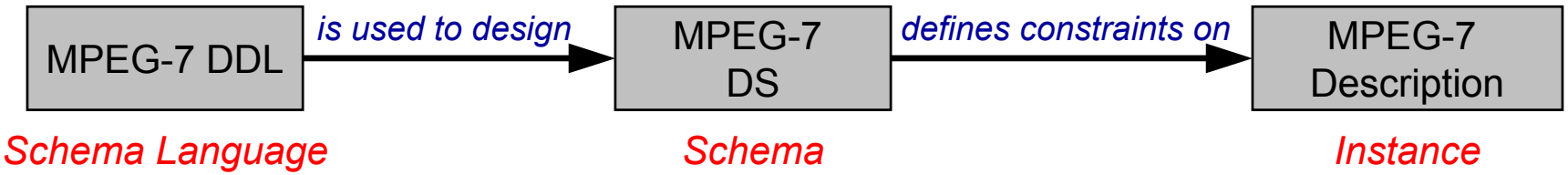
System Approach: a complete scenario



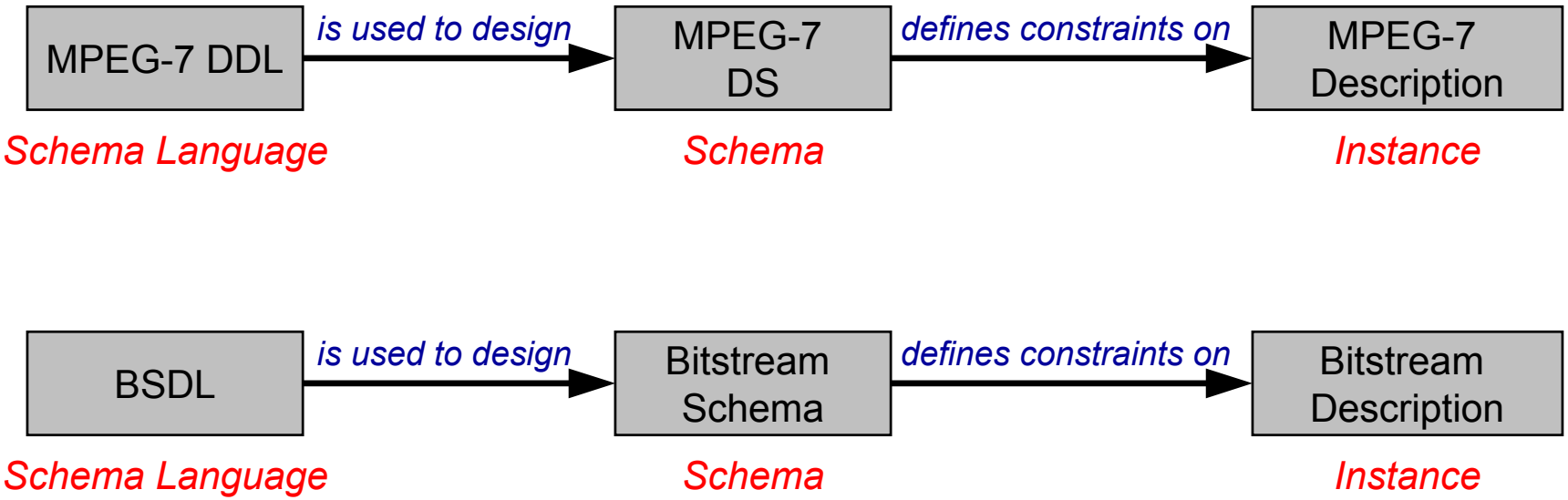
System Approach: a complete scenario



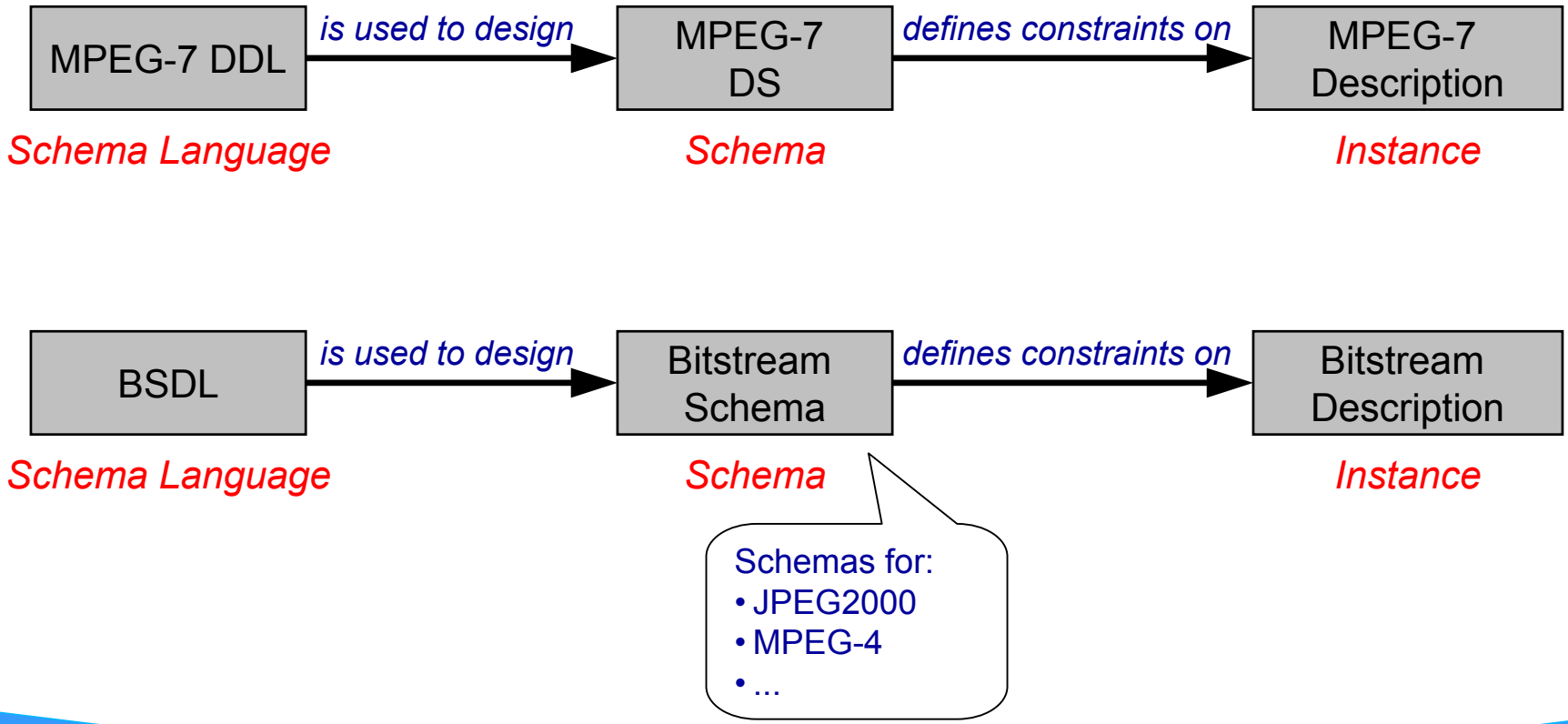
Analogy with MPEG-7



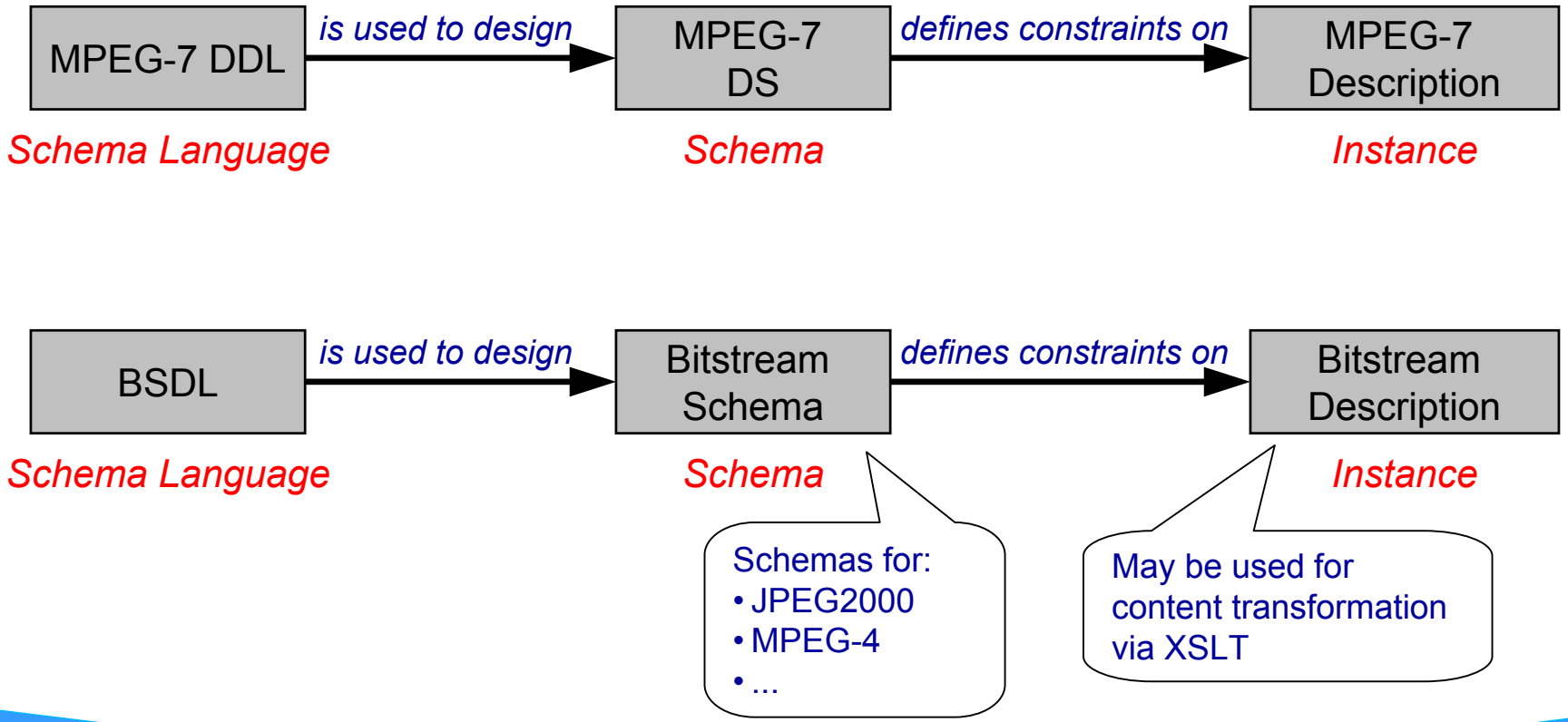
Analogy with MPEG-7



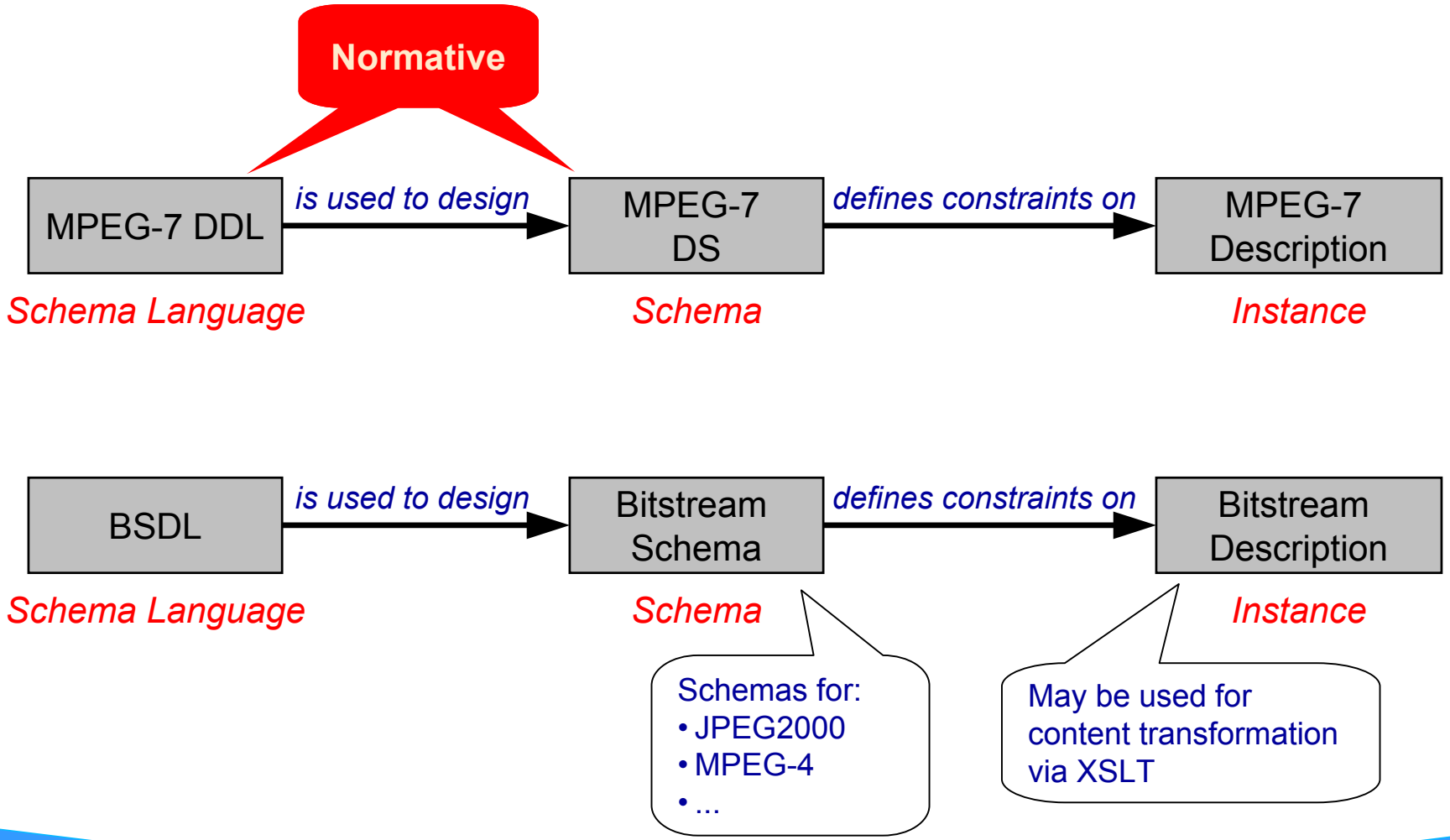
Analogy with MPEG-7



Analogy with MPEG-7



Analogy with MPEG-7



Analogy with MPEG-7