Network Services

Dr. Manfred Hauswirth
Laboratoire de Systèmes d'Information Répartis
École Polytechnique Fédérale de Lausanne (EPFL)
http://lsirpeople.epfl.ch/hauswirth/

Oct 23, 2002 – Lecture 3

Overview

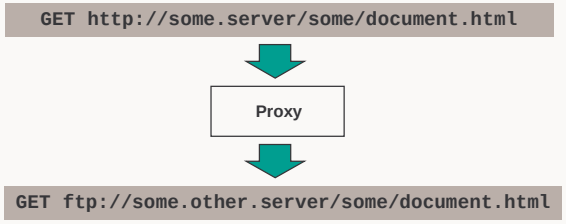
- Caching
- Search engines and robots
- Harvest, X.500, LDAP, DNS, UDDI
- HTML (HyperText Markup Language)
- XML (Extensible Markup Language)
- XSL (Extensible Stylesheet Language)
- XSLT (XSL Transformations)
- CSS (Cascading Style Sheets)
- XLink, XPointer, XPath
- DOM (Document Object Model)
- RDF (Resource Description Framework)
- SOAP (Simple Object Access Protocol)

Proxies, Caches, and Mirrors

- Application protocol gateway
  - "protocol translator"
  - clients have no direct access
  - example: SOCKS, TIS firewall
- Cache = proxy + persistent memory
  - "bandwidth multiplicator"
  - offline work
  - example: Squid
- Mirror
  - copy content at regular intervals
  - example: cron + wget

Proxies

- Applications
  - firewalls
  - clients without DNS
  - clients without FTP, WAIS, ...
  - clients without SSL
- Basic functionality



Caching Issues (1/2)

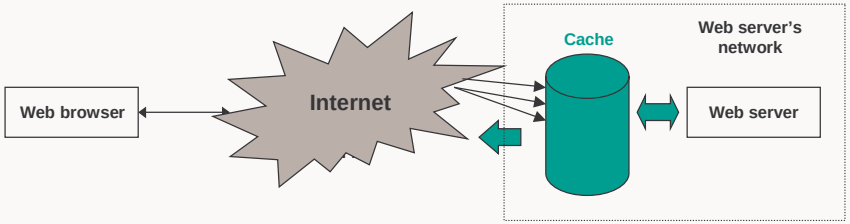
- What is being cached ?
- When to update (consistency vs. performance) ?
  - a posteriori vs. a priori (mirror, preventive)
  - If-Modified-Since: always, interval, never ?
  - Ignore [Cache-control|Pragma]: no-cache ?
- How long to cache ?
  - TTL depends on protocol, MIME type, cache config
  - Expires: header
  - Garbage collection: LRU

Caching Issues (2/2)

- Who can access the cache when and can access what information ?
- Volatile vs. persistent chaching
- Negative caching
- Confidential info (SSL, basic authentication)
- Implementation and tuning
  - hashed access to keys (URLs, ...)
  - when to swap info to disk ?
  - locking
  - pre-forking

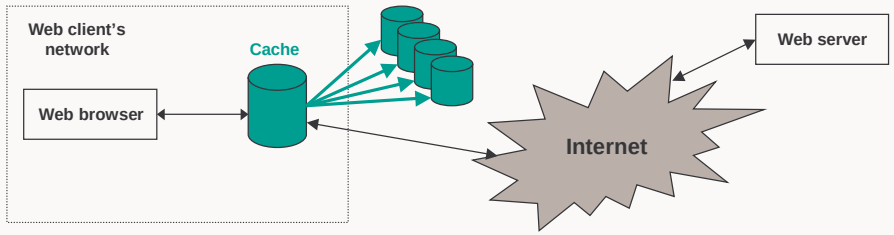
Caching Patterns (1/3)

- Chains (accelerators)
  - take away load from the web server
  - cache incoming requests for outgoing data = what you publish to the world !!



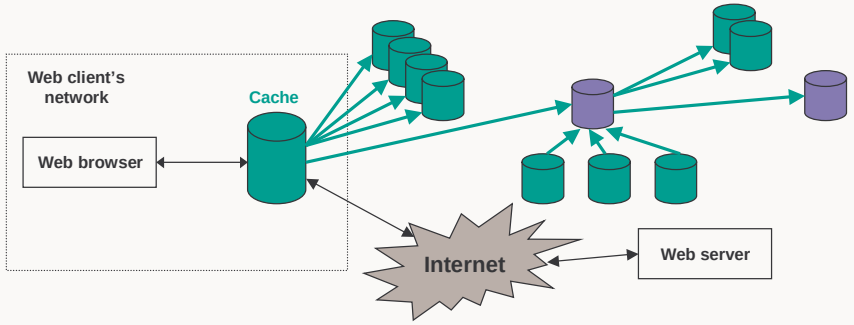
Caching Patterns (2/3)

- Collections
  - local cache miss => request is forwarded to a set of other caches
  - e.g., Internet Cache Protocol (Squid)



Caching Patterns (3/3)

- Hierarchies
  - client cache miss => local cache miss => request is forwarded to a higher level cache
  - frequently combined with cache collections
  - e.g., Internet Cache Protocol (Squid)



HTTP Support for Caching

- Client-side
  - Cache-Control: max-stale=<seconds>
  - If-[Modified|Unmodified]-Since: <date>
- Server-side
  - Cache-control:
    - no-cache, must-revalidate, max-age
  - (Pragma: no-cache)
  - Expires: <date>
  - Last-Modified: <date>
  - Etag: <tag>

Search engines

- (Stationary) robots recursively index documents at regular intervals=> database
  - get document and index it
  - extract references to external documents and index them
  - Example: Scooter (Altavista)
- Web interface to index database provides search access
  - Examples: Google, Altavista, Hotbot, Lycos, etc.

Robots

- Navigate through the network on their own volition and gather (index) information
  - topic-specific
    - worms, spiders, (mobile) agents
    - request ⇨ "trip"
  - document-specific
    - gatherer, indexers
    - (recursively) index documents => index database
    - request ⇨ index database (possibly full-text)

Beware the Robot

- Rapid fire
  - many (parallel) retrievals in a short time
  - server and network are being overloaded
  - even worse if several different robots visit the same site
- Black hole
  - robots is trapped in an "endless" loop
  - e.g., CGI script produces
    /cgi-bin/black/hole/1, /cgi-bin/black/hole/2, ...
- No incremental indexing
- Index information is not exchanged between robots

Robot Exclusion

Robot exclusion (/robot.txt)

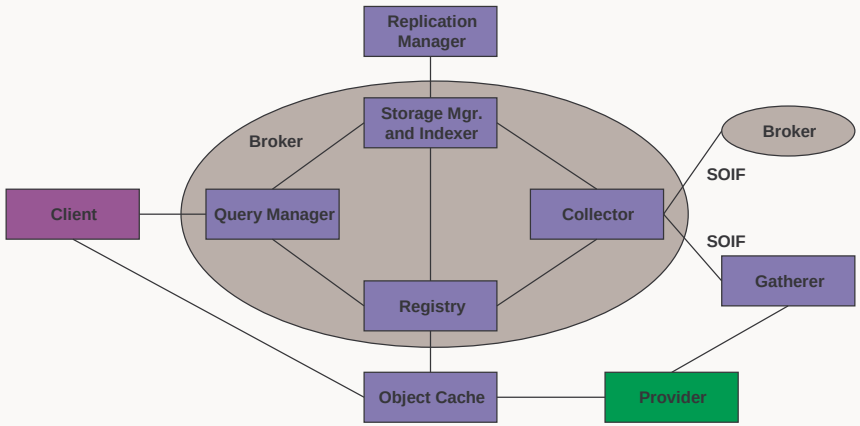
```
User-agent: *
Disallow: /cyberworld/map/  # infinite URLs (black hole)
Disallow: /tmp/             # these will soon disappear
Disallow: /secret.html
```

Robots META tag (HTML files)

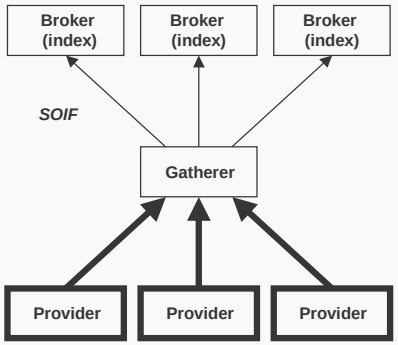
```
<html>
<head>
  <meta name="robots" content="noindex,nofollow">
  <meta name="description" content="This page ....">
  <title>...</title>
</head>
<body>
```

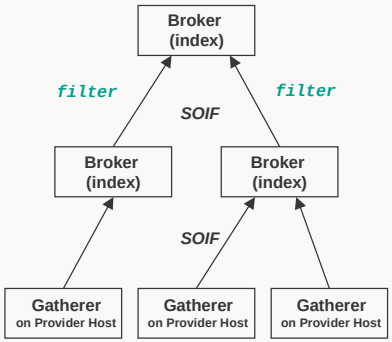

The Harvest System

Scalable, customizable information gathering, indexing, and searching



Harvest Configurations





- A remote Gatherer retrieves documents from Providers => high networking load on Providers
- Info is "essenced" to SOIF which is sent to multiple Brokers
- Local Gatherers provide indexing information to multiple Brokers ⇨ no re-indexing by multiple indexers necessary
- SOIF = Summary Object Interchange Format

Gatherer

- (Recursively) retrieve documents (local, HTTP, FTP, NNTP, ...)
- Intelligent content-type sensitive indexing (Essence)
  - extract index information based on content-type (e.g., index ASCII version of PostScript)
  - extract certain fields and infos (author, abstract, ...)
  - understands structured documents
  - new extractors/summarizers can be added easily
  - produces Standard Object Interchange Format

SOIF Example

```
@UPDATE {
@FILE { ftp://ecrc.de/pub/ECRC_tech_reports/reports/ECRC-93-10.ps.Z
Time-to_live{7}:  9676800
Last-Modification-Time{9}: 774988159
Refresh-Rate{7}: 2419200
Gatherer-Name{50}:  Computer Science Technical Reports - Selected Text
Gatherer-Host{21}:  bruno.cs.colorado.edu
Gatherer-Version{3}:    0.3
Type{10}:   Compressed
Update-Time{9}: 774988159
File-Size{6}:   164373
MD5{32}:    43193942d4d53f5a8e4a7b4bcff7a415
mbed<1>-Nested-Filename{13}:   ECRC-93-10.ps
Embed<1>-Type{10}:  PostScript
Embed<1>-File-Size{6}:  428233
Embed<1>-MD5{32}:   84c123582c3d0754a39a78a7e2fb6d23
Embed<1>-Keywords{105}: technical report ECRC{93{10
Polymorphic Sorts and Types for
Concurrent Functional Programs
Bent Thomsen
}
```


Other Harvest Components

- Broker
  - WWW query interface
  - Glimpse: index & search subsystem
    - 3-7% index size
    - approximate, boolean queries
    - incremental indexing
- Replicator: distribute a Broker's index DB
- Object Cache => Squid

Directory Services

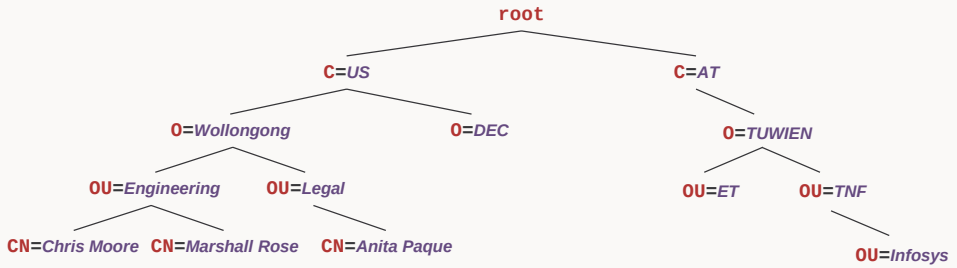
- Highly important as the Internet grows
  - Certificates, identification
  - Resources
- White pages (Example: X.500, DNS)
  - What is the phone number of John Smith in Austin, TX?
  - What is the IP address of www.epfl.ch?
- Yellow pages (Example: X.500)
  - Who offers a certain service in a certain area?
- Example service: http://wp.tuwien.ac.at:8888/

X.500

- Directory services of the OSI model
- Global, distributed database (Directory Information Base - DIB)
- Holds entities consisting of a set of attributes ⇨ object classes (inheritance)
- Entities, attributes and their types are freely definable

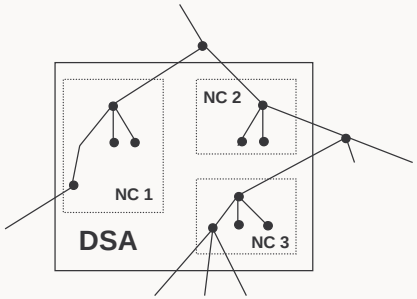
X.500 Directory Information Tree

- Distinguished Name (DN): path in DIT
  `C=US;O=Wollongong;OU=Engineering;CN="Marshall Rose"`
- Aliases, Referrals
- Roles



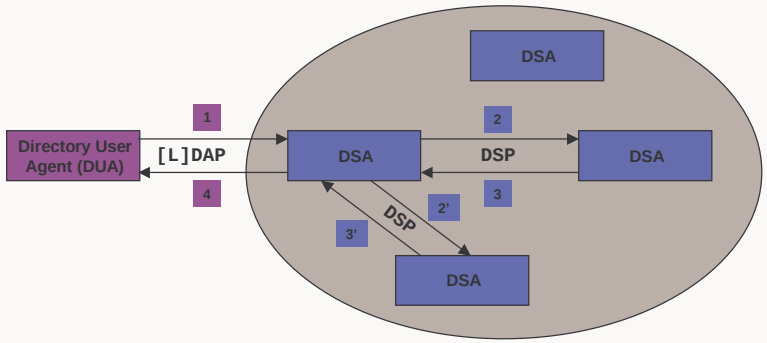
X.500 Directory System Agent

- Directory: set of interacting DSAs
- DSA: holds a set of naming contexts (NCs)



X.500 Directory Access

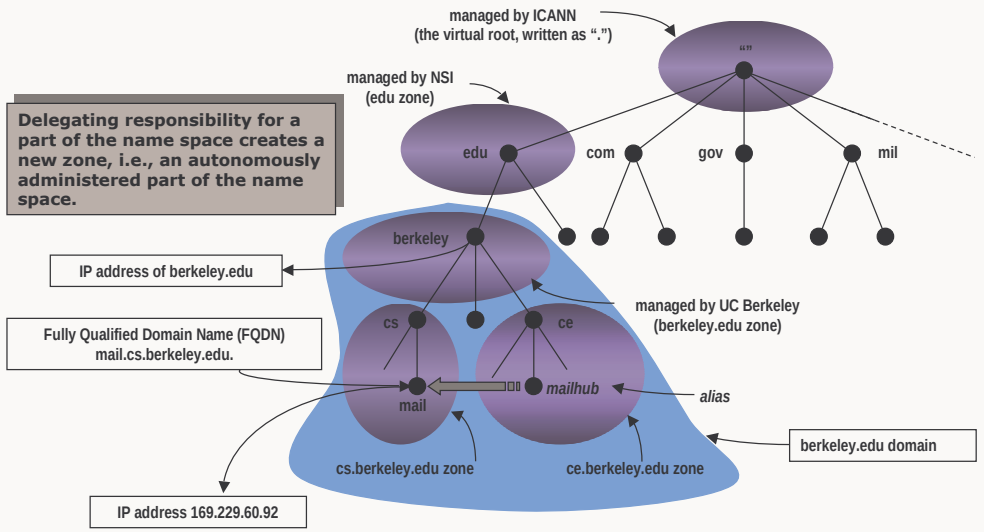
- (Lightweight) Directory Access Protocol ([L]DAP)
  `ldap://ldap.umich.edu/o=University%20of%20Michigan,c=US`
- Directory System Protocol (DSP)



Domain Name System

- Distributed hierarchical database of mappings
  - Host names ⇨ IP address ("name resolution")
  - IP address ⇨ host name ("back resolution")
  - Special information, for example, MX records
- Distributed management and maintenace
- Pre-DNS: *HOSTS.TXT* ⇨ */etc/hosts*
  - 1 text file to contain all hosts
  - managed by SRI-NIC
  - Problems:
    - load on SRI-NIC
    - consistency (name clashes)

DNS Overview



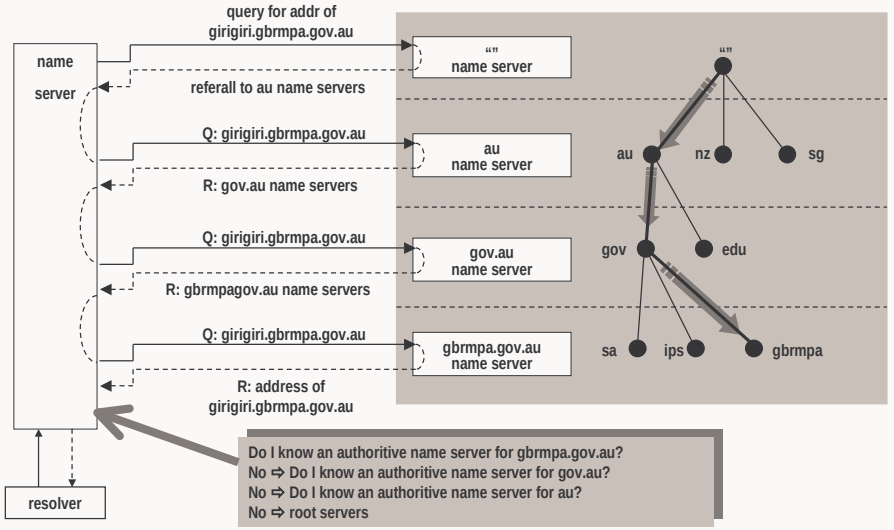
DNS Terminology - 1

- Top-Level Domain (TLD): com, edu, org, ch, at, ...
  - child of root (alternative name: first-level domain)
- Second-level Domain: ac.at, sun.com, ...
  - child (subdomain) of first-level domain
- TLDs are managed by ICANN (Internet Cooperation for Assigned Names and Numbers)
- Standard DNS software: Berkeley Internet Name Domain (BIND)
- Resource record types (types of mappings)
  - *A* (address): `myhost.movie.com. IN A 162.132.17.8`
  - *MX* (mail exchanger): `movie.com. IN MX 0 m1.movie.com`
    `IN MX 10 m2.movie.com`
  - *SOA* (Start Of Authority – mgmt. info), *NS* (authoritative Name Server), *CNAME* (Canonical Name - aliases), ...

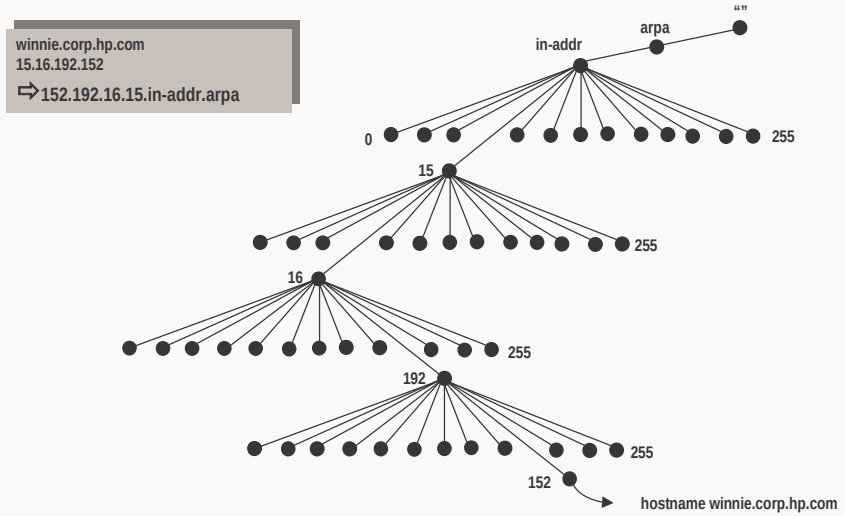
DNS Terminology - 2

- Name servers
  - A *primary master* name server for a zone reads the data for the zone from a file on its host
  - A *secondary master (slave)* name server for a zone gets the zone data from another name server that is *authoritative* for the zone, called its *master server.*
  - *Zone transfer*: a slave name server pulls the zone data
  - *Authoritative* name servers for a zone:
    - primary master
    - slaves
  - 13 root name servers
- Zone data file: files form which the primary master loads the zone data
- Resolver: clients that acess name servers (library)

Name Resolution



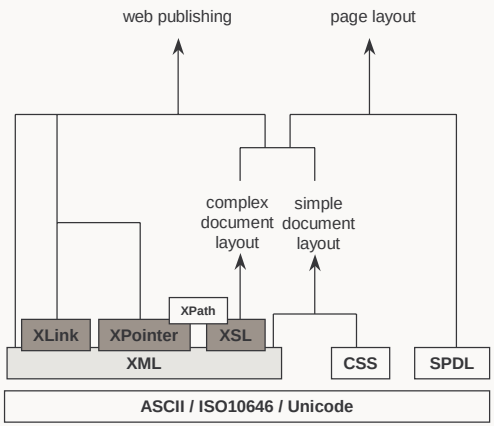
Back resolution (in-addr.arpa domain)



Universal Description, Discovery & Integration

- Standard for describing, publishing and finding web services
  - Still evolving, based on Simple Object Access Protocol (SOAP)
  - Use XML-based description files for services
  - Like WSDL, but different -> complex mapping
- Main components
  - White pages: basic contact information about an organization
  - Yellow pages: classification of organization based on industrial categorization
  - Green pages: technical description of services offered by registered organisations
- Access of UDDI Registry
  - Standard UDDI API
  - Web browser
- Data Structures
  - Business entity: general information + business services
  - Business services: business level description + binding templates
  - Binding templates: access point + tModel (service types)
  - tModel: abstract definition of a web service

W3C Publishing Standards



Web Publishing

- Static documents
  - HTML, ASCII text, Postscript, PDF
  - GIF, JPEG, MOV, Quicktime, AVI
  - AU, WAV, MP3, RealAudio
- Dynamic documents
  - executable content
  - Java, Javascript, Active-X, Dynamic HTML
- Variable documents
  - Dynamically-generated (on-the-fly)
  - CGI, FastCGI, JSP, PHP

HyperText Markup Language

- Document structure description
  - (sub-)sections
  - headings
  - tables
- No (?) layout information
  - style sheets
  - font mapping
- Defined in SGML / XML (XHTML)
  - Document Type Definition (DTD)

HTML History

- HTML 2.0
  - 1st version conforming to SGML
  - Core HTML (lists, forms, headings, fonts, image maps, ...)
- HTML 3.2 (no HTML 3.0 & HTML 3.1 !!)
  - many elements get additional attributes
  - font sizes, font faces, colors
  - form-based file upload
  - client side image maps
  - tables
  - APPLET tag
  - STYLE and SCRIPT tags (placeholders, no exact mechanisms)

HTML 4.01

- Common attributes for many elements: Id, Title, Style, Class
- Improved tables (Thead, Tbody, Tfoot, column handling, formatting)
- Improved forms
- Incremental rendering of tables
- STYLE element now encloses style sheet instructions (CSS)
- Style sheets: separation of content and styles (CSS)
- Frames
- Embedded documents (IFRAME)
- APPLET element deprecated (OBJECT)
- Intrinsic events (onclick, ondblclick, onmouseover, onmouseout, ...)
- OBJECT element (IMAGE, APPLET, IFRAME)
- SCRIPT element (client-side scripting - JavaScript, VBScript, ...)

OBJECT Element Examples

```
<OBJECT data="http://some.server.com/me.png type="image/png">
   A picture of me.
</OBJECT>
```

```
<OBJECT codetype="application.java"
        classid="java:AudioItem.start
        codebase="http://some.server.com/java/AudioStuff/
        width="15" height="15">
   <PARAM name="sound" value="Welcome.au">
   Play a welcoming sound.
</OBJECT>
```

```
<OBJECT data="embed_me.html">
   Warning: embed_me.html could not be embedded.
</OBJECT>
```


Client-side Scripting (1/2)

- Languages: JavaScript, VBScript, Python, Tcl, ...
- Security ?

```
<SCRIPT type="text/vbscript" src="http://someplace.com/vbcalc">
</SCRIPT>
```

```
<INPUT NAME="userName" onblur="validUserName(this.value)">
```

```
<INPUT NAME="edit1" size="50">
<SCRIPT type="text/vbscript">
   Sub edit1_changed()
     If edit1.value = "abc" Then
       button1.enabled = True
     Else
       button1.enabled = False
     EndIf
   EndSub
</SCRIPT>
```


Client-side Scripting (2/2)

```
<INPUT NAME="num"
       onchange="if (!checkNum(this.value, 1, 10)) {
                   this.focus(); this.select(); }
                 else {thanks()}"
       VALUE="0">
```

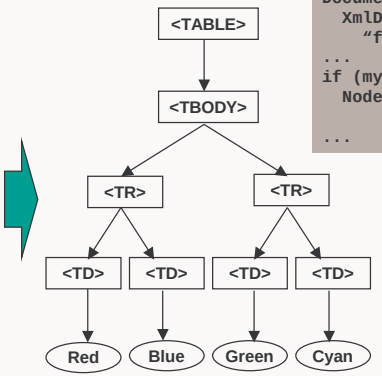
```
<BUTTON type="button" name="mybutton" value="10">
<SCRIPT type="text/javascript">
   function my_onclick() {
         ...
   }
   document.form.mybutton.onclick = my_onclick
</SCRIPT>
</BUTTON>
```


Document Object Model

- Parsers (XML/HTML) build a (tree) model of a document
- DOM supports access to this model, i.e. the parts (fragments) of a document
- Platform- and language-neutral interface which allows programs and scripts to dynamically access and update
  - content of documents (XML/HTML)
  - structure of documents
  - style of documents
- OMG IDL specification
- Language bindings for Java and ECMAScript

DOM Example

```
...
<TABLE>
<TBODY>
<TR>
<TD>Red</TD>
<TD>Blue</TD>
</TR>
<TR>
<TD>Green</TD>
<TD>Cyan</TD>
</TR>
</TBODY>
</TABLE>
...
```




```
Document myDoc=
  XmlDocument.createXmlDocument(
    "file:/doc.html", true);
...
if (myNode.hasChildNodes()) {
  Node child1 =
        myNode.getFirstChild();
...
```


Cascading Style Sheets

- Separation of content (HTML, XML documents) and presentation style (CSS)
  - simplified Web authoring
  - easier Web site maintenance
- CSS vs. XSL
  - CSS was defined earlier
  - XSL is still a draft while CSS is already supported by browsers
  - XSL is more powerful => too complex for many users/applications

Style and HTML

```
<HTML>
<HEAD>
  <STYLE TYPE="text/css">
    H1 { color: blue;
         font-style: italic;
         font-family: helvetica }
    .verde { color: #00FF00 }
  </STYLE>
</HEAD>
<BODY>
  <H1>Heading 1</H1>
  <H2 class="verde">Heading 2</H2>
  <P STYLE="color: red">
    paragraph text
  </P>
  yet another paragraph
</BODY>
</HTML>
```

Heading 1

Heading 2

paragraph text

yet another paragraph

External CSS and Cascasding

```
<HTML>
<HEAD>
  <TITLE>Document with Cascading Style Sheets</TITLE>
  <LINK rel="alternate stylesheet" title="compact"
        href="small-base.css" type="text/css">
  <LINK rel="alternate stylesheet" title="compact"
        href="small-extras.css" type="text/css">
  <LINK rel="alternate stylesheet" title="big print"
        href="bigprint.css" type="text/css">
  <LINK rel="stylesheet" href="common.css"
        type="text/css">
</HEAD>
<BODY>
  ...
</BODY>
</HTML>
```

```
@import "fineprint.css" print;
@import url("bluish.css") projection, tv;
```

Without Style

Previous Next Up Down

Heading 1

He brewed a song of love and hatred – oblique suggestions – and he waited.
-- Jethro Tull -- Minstrel in the Gallery

Heading 2

Heading 3

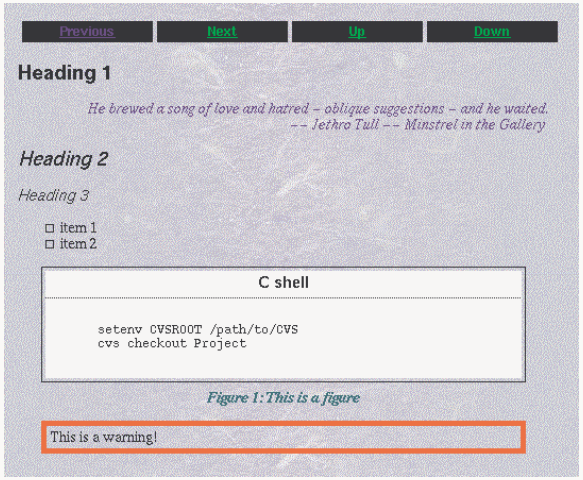
- item 1
- item 2

C shell

```
setenv CVSROOT /path/to/CVS
cvs checkout Project
```

Figure 1: This is a figure
This is a warning!

Better do it stylish !



CSS by Example (1/3)

```
BODY {
  background: #def url("recbg.jpg");
  color: black;
  margin: 0.5em;
}

A:link { color: #00ff00; }
A:visited { color: #0000ff; }

TABLE.navigation {
  border-style: none;
}

TD.navigation {
  background: black;
  text-align: center;
  font-weight: bold;
  font-family: helvetica, sans-serif;
  padding: 0.2em;
  vertical-align: top;
}
```

```
<HTML>
<HEAD>
<LINK rel="STYLESHEET"
  href="MyStyle.css" type="text/css">
</HEAD>
<BODY>
<TABLE class="navigation" COLS=4>
<TR>
<TD class="navigation">
<A HREF="...">Previous</A></TD>
...
```

CSS by Example (2/3)

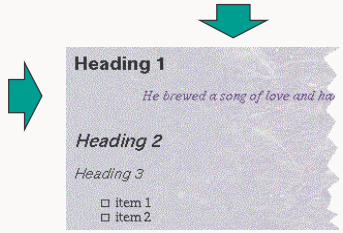
```
H1, H2, H3, H4, H5, H6 {
  font-family: helvetica, sans-serif;
  text-align: left;
  font-weight: normal;
  font-style: italic;
}

H1 {
  font-weight: bold;
  font-style: normal;
}

DIV.quote {
  text-align: right;
  font-style: italic;
  color: blue;
}

LI {
  list-style: square;
}
```

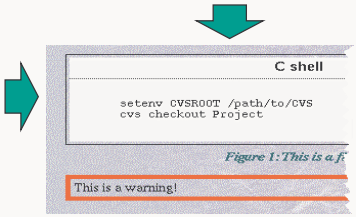
```
<H1>Heading 1</H1>
<DIV class="quote">
  He brewed a song of love and ...
</DIV>
<H2>Heading 2</H2>
<H3>Heading 3</H3>
<UL>
  <LI>item 1</LI>
  <LI>item 2</LI>
</UL>
```



CSS by Example (3/3)

```
DIV.listing { border: solid thin;
  margin-top: 0.5em;
  margin-right: 1.5em;
  margin-left: 1.5em;
  margin-bottom: 0.5em;
  background: white; }
SPAN.ltitle { text-align: center;
  font-family: helvetica, sans-serif;
  font-style: normal;
  font-weight: bold; font-size: 100%;
  margin-top: 0.25em; }
DIV.warning { border: solid thick;
  border-color: red;
  margin-top: 0.5em;
  margin-right: 1.5em;
  margin-left: 1.5em;
  margin-bottom: 0.5em; }
CAPTION { text-align: center;
  color: #088; font-style: italic;
  font-weight: bold; font-size: 100%;
}
```

```
<DIV CLASS="listing">
  <SPAN CLASS="ltitle">C shell</SPAN>
  <HR>
  <PRE>
    setenv CVSROOT /path/to/CVS
    cvs checkout Project
  </PRE>
</DIV>
<CAPTION>Figure 1: This ...</CAPTION>
<DIV CLASS="warning">
  This is a warning!
</DIV>
```



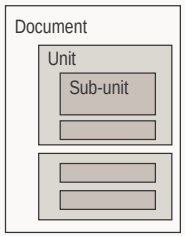
Extensible Markup Language

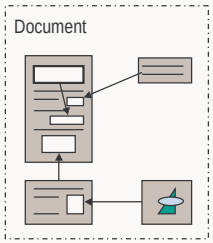
- XML (1998) is an application of SGML
  - Standard Generalized Markup Language (1986): ISO8879
  - influenced by HTML (SGML Document Type Definition)
- Structure description language
- Meta-language: language to describe other languages
- Tags enclose identifiable parts of a document => markup (type-setting systems)

```
<warning>
<para>This substance is <emph>hazardous</emph> to health</para>
<para>See procedure 12A.7 for information on protective clothing.</para>
<image .../>
</warning>
```


XML Documents

- logical structure:
  - division of documents into name units and sub-units
  - unit = element
- physical structure
  - components of the document
  - can be named and stored separately
  - component = entity





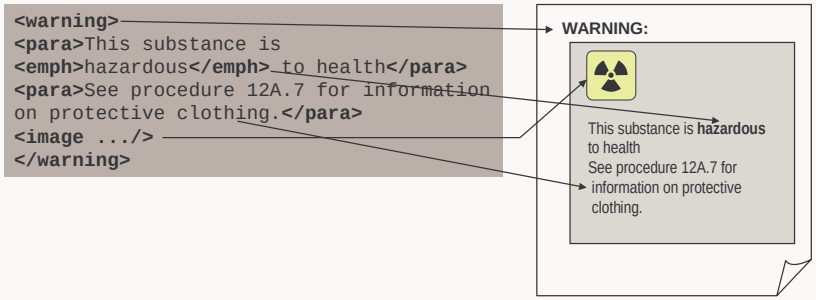
Document Type Definition

- DTD defines the elements allowed
- A parser compares the DTD rules against a given XML document => validation
- XML DTDs can be applied for data-type definitions (XML-RPC), data exchange (EDI, push, RDBMS), etc.

```
<!ELEMENT warning   (para*, image?)>
<!ELEMENT para      (#PCDATA | emph)*>
<!ELEMENT image     EMPTY>
<!ATTLIST image     url CDATA #REQUIRED>
<!ELEMENT emph      (#PCDATA)*>
```

XML Document Presentation

- Style sheets specify output format
- 1 XML document, *n* alternative style sheets depending on audience, media, etc.



XML by Example (1/2)

```
<!-- DTD for user groups in JSEF -->
<?xml encoding="US-ASCII"?>
<!ELEMENT usergroups (user)*>
<!ATTLIST usergroups lastChanged  CDATA  #REQUIRED
                     changedBy    CDATA  #REQUIRED>
<!ELEMENT user  (addgroups | subgroups)*>
<!ATTLIST user  userName ID  #REQUIRED>
<!ELEMENT addgroups (group)+>
<!ELEMENT subgroups (group)+>
<!ELEMENT group EMPTY>
<!ATTLIST group groupName  CDATA  #REQUIRED>
```


XML by Example (2/2)

```
<?xml version="1.0"?>
<!DOCTYPE usergroups SYSTEM "usergroups.dtd">
<usergroups lastChanged="10/21/1999" changedBy="sysadmin">
  <user username = "Charly Brown">
    <addgroups>
      <group groupname = "User" />
      <group groupname = "InternalUser" />
    </addgroups>
    <subgroups>
      <group groupname = "Developer" />
    </subgroups>
  </user>
  <user username = "Hugo Boss">
    <addgroups>
      <group groupname = "Admin" />
    </addgroups>
  </user>
</usergroups>
```

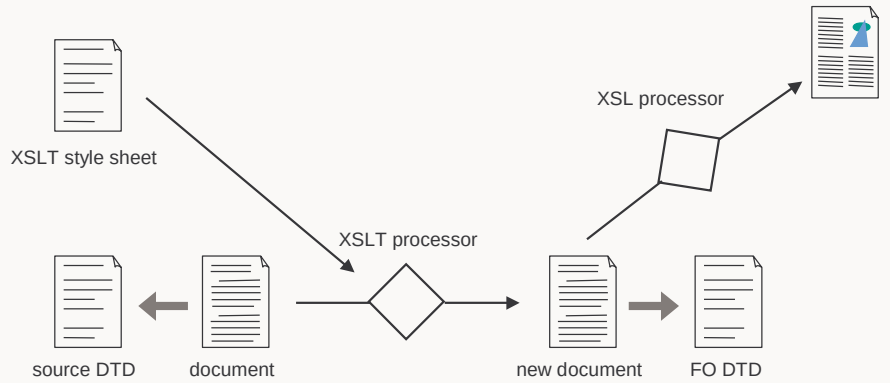

Extensible Stylesheet Language

- XSL is a language for expressing stylesheets and consists of
  - a language for transforming XML documents (XSLT)
  - an XML vocabulary for specifying formatting semantics (Formatting Objects DTD - FO DTD)
- CSS < XSL < DSSSL (SGML's Document Style Semantics and Specification Language)
- Style sheets
  - target specific elements => closely related with DTDs
  - 1 XML document can be associated with n style sheets (target audience, media, etc.)

XML Everywhere

- XML has become the general data description and exchange format
  - UDDI
  - XML-RPC
  - SOAP
  - ....

XSL and XSLT Processing



XSL/XSLT Example 1

```
An <emph>emphasized</emph> word.
```

```
<template match="emph">
  <fo:inline-sequence
       font-weight="bold">
     <apply-templates/>
  </fo:inline-sequence>
</template>
```

An **emphasized** word.

XSL/XSLT Example 2

```
<xsl:template match="/">
 <html>
  <head><title>Customers</title></head>
  <body>
   <table>
    <tbody>
     <xsl:for-each select="customers/customer">
      <tr>
       <th>
        <xsl:apply-templates select="name"/>
       </th>
       <xsl:for-each select="order">
        <td>
         <xsl:apply-templates/>
        </td>
       </xsl:for-each>
      </tr>
     </xsl:for-each>
    </tbody>
   </table>
  </body>
 </html>
</xsl:template>
```

```
<customers>
 <customer>
    <name>...</name>
    <order>...</order>
    <order>...</order>
  </customer>
  <customer>
    <name>...</name>
    <order>...</order>
    <order>...</order>
  </customer>
</customers>
```

an HTML document containing a table with a row for each `customer` element

XLL = XLink + XPointer

- Extensible Linking Language = XML Linking Language + XML Pointer Language
- XLink defines how one document links to another document (URL)
- XPointer defines how individual parts of a document are addressed

XLL vs. HTML Links

- HTML
  - URLs point at a single document
  - More granularity requires anchors => write access to target
  - URLs have no browsing history (=> browser => not very reliable)
  - URLs are unidirectional
- XLL
  - XLL ⊃ URL
  - Multidirectional links
  - Any element can become a link (not just the A element)
  - Links do not have to be stored in the same files as the documents they link
  - Links to arbitrary positions in a XML document

A simple XLink

- Simple XLinks = HTML links

```
<FOOTNOTE xmlns:xlink="http://www.w3.org/1999/xlink/namespace/"
          xlink:type="simple"
          xlink:href=" http://www.w3.org/fn/note7.xml">7</FOOTNOTE>
```


A more complex XLink

```
<?xml version="1.0"?>
<WEBSITE xmlns:xlink="http://www.w3.org/1999/xlink/namespace/"
         xlink:type="extended" xlink:title="Cafe au Lait">
  <NAME xlink:type="resource" xlink:role="source">
    Cafe au Lait</NAME>
  <HOMESITE xlink:type="locator"
            xlink:href="http://metalab.unc.edu/javafaq/"
            xlink:role="us"/>
  <MIRROR xlink:type="locator"
          xlink:title="Cafe au Lait Swedish Mirror"
          xlink:role="se"
          xlink:href="http://sunsite.kth.se/javafaq"/>
  <MIRROR xlink:type="locator"
          xlink:title="Cafe au Lait Swiss Mirror"
          xlink:role="ch"
          xlink:href="http://sunsite.cnlab-switch.ch/javafaq/"/>
  <xlink:arc from="source" to="ch" show="replace" actuate="onRequest"/>
  <xlink:arc from="source" to="us" show="new"     actuate="onRequest"/>
  <xlink:arc from="source" to="se" show="replace" actuate="onRequest"/>
</WEBSITE>
```

XPointer

- Defines an addressing scheme for individual parts of an XML document

```
xpointer(id("ebnf"))

xpointer(/child::FAMILYTREE/child::PERSON[position()=3]/child::NAME)

xpointer(/descendant::HUSBAND[position()=1]/parent::*)
```


Resource Description Framework

- RDF is a framework for meta-data
  - interoperability of meta-data
  - machine understandable semantics of meta-data

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://www.yoyodine.com/joe/doc.html"
      dc:creator="Joe Smith"
      dc:title="My document"
      dc:description="Joe's ramblings about his summer vacation."
      dc:date="1999-09-10" />
</rdf:RDF>
```


SOAP – Simple Object Access Protocol

- Lightweight messaging framework based on XML
- Supports simple messaging and RPC
- SOAP consists of
  - SOAP envelope construct: defines the overall structure of messages
  - SOAP encoding rules: define the serialization of application data types
  - SOAP RPC: defines representation of remote procedure calls and responses
- SOAP messages consist of
  - Envelope: top element of XML message (required)
  - Header: general information on message such as security (optional)
  - Body: data exchanged (required)

Example: SOAP Message

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   <SOAP-ENV:Header>
       <t:Transaction
           xmlns:t="some-URI"
           SOAP-ENV:mustUnderstand="1">
               5
       </t:Transaction>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
       <m:GetLastTradePrice xmlns:m="Some-URI">
           <symbol>DEF</symbol>
       </m:GetLastTradePrice>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Transport binding

Envelope

Header

Body

Ahead

- Peer-to-peer systems
  - Napster
  - FastTrack
  - Gnutella
  - Freenet