*:96 Overheads

Part 9: WebDAV, RSS, SOAP (Web applications), Bittorrent

More about this course about Internet application protocols can be found at URL:

`http://dsv.su.se/jpalme/internet-course/Int-app-prot-kurs.html`

Last update: 07-08-11 13.20

Use of HTTP for custom protocols

HTTP is often used as a basis for developing new custom protocols for particular applications.

Pro: Firewalls often stop access using other than the normal ports. Using HTTP over the standard HTTP port 80 will usually bypass firewalls.

Pro: Existing software modules for sending and receiving HTTP can be reused.

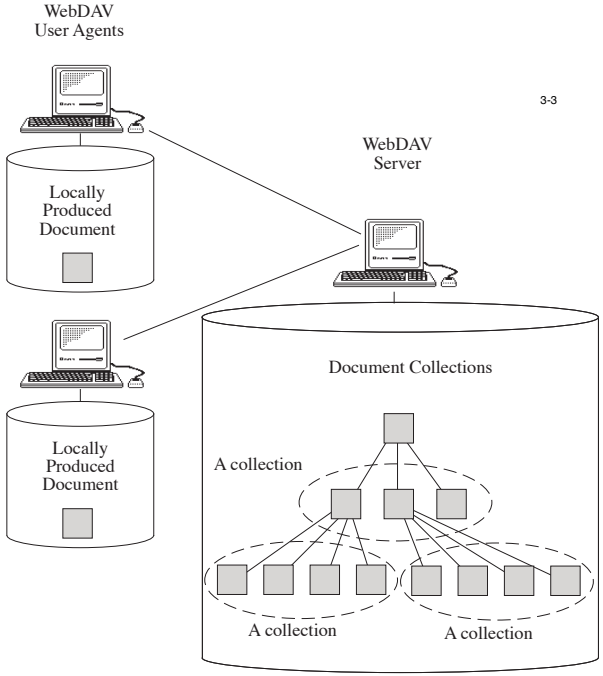
Con: HTTP is a very complex protocol, often has many features not needed for particular protocols, and which can cause interference.

Example: Caching is not needed for particular protocols and can cause problems for them.

In practice, often only a subset of HTTP is used, but often without specifying this restriction.

Example of new specialized protocols using HTTP: WebDAV, RSS and Soap.

WebDAV Model:



WebDAV
HTTP Extensions for Distributed Authoring

WWW and HTTP: Good for delivery of pages (read only). Not good for managing, editing and revising pages.

Problem: Several people working on the revision of the same set of documents, for example web pages. Errors will occur if there is no coordination of what they are doing.

Example 1:

User A downloads a document to modify it.
User B downloads the same document to modify it.
User A uploads the modified document.
User B uploads the modified document.

The result with this example will be that user B will overwrite the changes made by user A, so that these changes will be lost, since user B will base its changes on the document before user A has uploaded its change.

Variation of example of why WebDAV is needed:

Suppose that the names and the prices are stored in separate objects in the data base.

Original text:	Red Orchid:	Price $ 6.75
User A downloads this and changes it to:	Bouquet of five red orchids:	Price $ 33.75
User B downloads this and changes the descriptive text to:	Red or pink orchids	Price $ 6.75
User A saves its change:	Bouquet of five red orchids:	Price $ 33.75
User B saves its change to the description only:	Read or pink orchid:	Price $ 33.75

Result: Unintended five-fold increase to the price of an orchid!

WebDAV =
Web Distributed Authoring And Versioning

Exclusive lock (hard lock).

Shared lock (soft lock).

No support for merging, management of simultaneous edits not fully handled by basic WebDAV standard (RFC 2518).

Data base of objects (resources) with attributes such as title, date created, last revision date, author, etc. etc.

Data base of objects organised into folders and subfolders (directories and subdirectories) in WebDAV named “Collections”.

WebDAV coding methods

A combination of revised and new HTTP headers, specified with ABNF, and bodies of HTTP requests and results, specified in XML.

Why XML: Allows unlimited sets of different attributes, unlimited sets of objects (resources) in directories (collections) and subdirectories.

HTTP method
HTTP header

```
PROPFIND  /container/ HTTP/1.1
Host: www.foo.bar
Content-Length: 117
Content-Type: text/xml; charset="utf-8"
```

Specified with ABNF

HTTP body

```
<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
<D:prop><D:supportedlock/></D:prop>
</D:propfind>
```

Specified in XML

WebDAV HTTP methods

PROPFIND: Get some or all properties of one or more objects (r

PROPPATCH: Modify or remove properties of one or more objects (resources).

MKCOL: Create folder/directory/collection.

GET, HEAD for collections: Unspecified in standard HTTP, can in standard HTTP retrieve a list of files, or a start page (index.html).

POST for collections.

DELETE of objects (resources) and collections.

PUT on objects (resources) and folders/directories/collections.

COPY and MOVE on objects (resources) and folders/directories/collections.

LOCK and UNLOCK on objects (resources) and folders/directories/collections.

Example of new or modified HTTP headers:

Dav: Specifies that this is a WebDAV operations, and indicates version of WebDAV.

Depth: Operation on only a collection as a whole, or on its members and submembers to limited or unlimited depth. Usual values: “0” or “infinity”.

Destination: Destinations for operations such as COPY and MOVE.

If: Existing HTTP header extended.

Lock-token: Unique identifier of a lock.

Overwrite: Should overwriting be allowed?

Time-out: Maximum duration of a lock

Example of use of XML: A source Property

```
<?xml version="1.0" encoding="utf-8" ?>
<D:prop xmlns:D="DAV:" xmlns:F="http://www.foocorp.com/Project/">
     <D:source>
          <D:link>
               <F:projfiles>Source</F:projfiles>
               <D:src>http://foo.bar/program</D:src>
               <D:dst>http://foo.bar/src/main.c</D:dst>
          </D:link>
          <D:link>
               <F:projfiles>Library</F:projfiles>
               <D:src>http://foo.bar/program</D:src>
               <D:dst>http://foo.bar/src/main.lib</D:dst>
          </D:link>
          <D:link>
               <F:projfiles>Makefile</F:projfiles>
               <D:src>http://foo.bar/program</D:src>
               <D:dst>http://foo.bar/src/makefile</D:dst>
          </D:link>
     </D:source>
</D:prop>
```

“D:” = Dav namespace, “F:” = own namespace, "src"=Source, "dst"=Destination

WebDAV specification RFCs

RFC 2518 Basic WebDAV: Locks, properties, collections.

RFC 3648 Versioning Extensions to WebDAV:

- Version history
- Parallel development (bug corrections versus extensions)
- Forking, merging

RFC 3648 Ordered Collections - not based alone on property values

Examples: Recommended access order, Revision history order, Pages of a book, Overheads in a lecture

RFC 3744 Access Control Protocol: Access control lists, who can do what?

RFC 4316 (experimental) Datatypes for properties (XML has rather few datatypes): Passing data type with value, Schema, Mandatory property types, Updating parts of a structured property.

RFC 4331 Quota and Size Properties

RFC 4437 (experimental) Redirect Reference Resources (authoring redirect commands)

RFC 4709 (informational) Seeing WebDAV data in ordinary HTML interfaces

RFC 4791 Calendaring (iCalendar format): Accessing calendars

RFC 4918 Revised version of RFC2518

RSS and Podcasting – Push and Pull

Push: Protocols where the sender controls what to send to whom.

Pull: Protocols where the recipient controls what to downlo

Protocol	Push	Pull
E-mail	Strongly push oriented, causing problems with information overload and spamming.	Recipients can sometimes chose to subscribe or unsubscribe to mailing lists.
Usenet News and Forum systems	Senders decide what to send to a newsgroup.	Recipients decide which newsgroups/forums to subscribe to.
Moderated groups in News and Forums	Moderator controls what is sent.	Recipients are protected by the moderator from getting what they do not want.
RSS and Podcasting	Owner controls what is sent.	

RSS and Podcasting main principle

RSS and Podcasting are mainly used for allowing users to subscribe to for example news items from a newspaper or new radio programs from a radio station. The main principle of RSS and Podcasting is a web page called the seed. The seed is in XML format, and contains some general data about the channel, followed by a list of items. The list of items is updated every time a new radio program or news item is available. The RSS readers run in the background on the user's personal computer and checks for news. News are downloaded automatically, some RSS readers will even automatically copy the news to an MP3 player.

Versions of RSS

There are three main branches of RSS:

RDF Site Summary, Rich Site Summary: RSS 0.90, RSS 1.0 and RSS 1.1.

Really Simple Syndication: RSS 0.92 and 2.0.

Atom: IETF RFC 4287.

RSS 2.0 is mostly used, but many RSS readers will accept data in all three versions of RSS.

Format of an RSS seed

```
<channel>
   <title>Liftoff News</title>
   <language>en-us</language>
   <item>
    <title>Star City</title>
    <link>http://liftoff.msfc.nasa.gov/news/2003/news-starcity.asp</link>
    <description>...</description>
    <pubDate>Tue, 03 Jun 2003 09:39:21 GMT</pubDate>
    <guid>http://liftoff.msfc.nasa.gov/2003/06/03.html#item573</guid>
   </item>
    <item>
    ...
   </item>
   <item>
    ...
   </item>
</channel>
```


Information which is specified only once for a whole channel:

Element name	Description
<title>	Name of the channel.
<link>	URL to its website.
<description>	A short description of the channel.
<language>	Language for the channel.
<rating>	A PICS rating for the channel.
<cloud>	Reference to a facility, where the cloud connects to your computer and tells it when there is news.
<ttl>	"time to live" - information for caching.
<textInput>	A facility where people can send comments on a channel..
Element name	Description
<title>	Name of the channel.
<link>	URL to the website for the channel.

Information which is specified separately for each item:

Element name	Description
<title>	Title of the item.
<link>	URL from which the item can be downloaded. Note that <enclosure> is more commonly used for this.
<description>	Synopsis if the item.
<author>	Email address of the author of the item.
<comments>	Comments area, where comments from viewers of the item can be published.
<pubDate>	When the item was published.
<guild>	A globally unique identifier of the item. Used by RSS readers to know which items they have downloaded and not downloaded>.
<enclosure>	See next slide.

RSS <enclosure> element

The <enclosure> element is where an RSS 2.0 item usually indicates from where it can be downloaded. Example:

```
<enclosure url="http://www.scripting.com/mp3s/weatherReportSuite.mp3"
length="12216320" type="audio/mpeg" />
```

In addition to the url, the length and the data type is specified, which makes it easier for RSS readers to know how to download the item.

HTML in RSS

The HTML text "this is bold" can in RSS be encoded in two ways:

Method 1: HTML Entity encoding:

<description>this is bold</description>

Method 2: Encoding HTML within a CDATA Section:

<description><![CDATA[this is bold]]></description>

BitTorrent

Problem: Downloading large files (video, audio, etc.) puts a large burden on the server from which the file is available.

Solution: The large file is split into many small pieces. Each piece can be downloaded in any order. All computers (both those getting the file, and those providing the file) share the burden of providing the file. As soon as one computer has got one piece of the file, it can provide this piece to other computers.

See figure on next slide.

BitTorrent Model concepts:

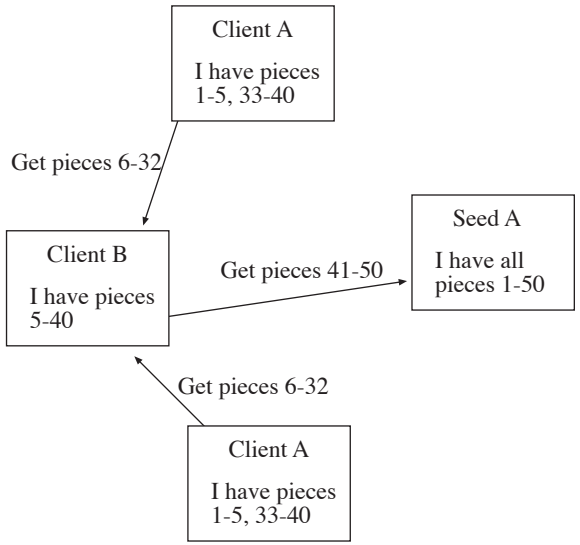
Tracker= directory of torrents.

Torrent = description, no storage of contents, who has what parts, torrents.

Seed = Computer which has all the pieces.

Client = Computer downloading)and uploading) a file.

Swarm = All the computers involed in the transfer.



Peer to Peer (P2P) Protocols

No central storage, many computers provide all or parts of files to each other.

Bittorrent is not a cleanP2P Protocol, since it involves a Torrent which knows which computer has what parts and guides the dowloading process. Other, more P2P clean proocols, are Gnutella.

Illegal downloading

1. Holding a torrent is not illegal in many countries.
2. The file can be available from many peers
3. Each peer may only hold one or a few files.

Difficult to find and prosecute involved users.