Abstract

In today's world networking has become one of the most important ways of data interchange. Now one does not just want to read, but also write and particularly store documents on the net, overcoming geographical distances and being flexible in choice of client device is probably the most important aspect hereby.

WebDAV and especially its extensions are a relatively young development in networking protocols. In my presentation I will give an overview and explain the main protocols, also there will be information about existing implementations.

1 Introduction

WebDAV and its supplements are extensions to HTTP. First I will give a short overview of HTTP, then explain the extensions. Afterward, there will be a presentation of existing implementations and at the end I will draw a conclusion.

2 HTTP

HTTP (HyperText Transfer Protocol) is a network protocol. It cannot be assigned to a single layer in the ISO/OSI Network Model, because it has at least features of the application as well as the presentation layer. The underlying TCP is used for connection-management.

2.1 History

The first version was specified with RFC1945 [4] in 1996. One year later the digest authentication was added [9]. It introduced a new authentication scheme that does not require the password being sent as plain text over the network (contrary to the basic authentication scheme in [4]). The last large change was done in 1999 with RFC2616 [7]. It introduced version 1.1 of HTTP, which includes now support for hierarchical proxies, virtual hosts and the ability to reuse a TCP connection. At the same time both authentication schemes (basic and digest) were united in RFC2617 [8]. There were also smaller improvements done.

2.2 Example

Figure 1 shows a simple HTTP request. First there is a method (e.g., GET) that declares, what to do. Then follows the resource (e.g., /index.html) onto that the method should be applied. After the first line there may be header fields (e.g., Host: www.example.net) which specify
GET /index.html HTTP/1.1
Host: www.example.net

Figure 1: HTTP request

parameters for the method or additional information. There are request methods that submit data. The data then follows after the header fields as request body.

HTTP/1.1 200 OK
Server: Apache/1.3.29 (Unix) PHP/4.3.4
Content-Length: xxxx (size of content in bytes)
Content-Language: de
Content-Type: text/html
Connection: close

<html> ... (content of index.html)

Figure 2: HTTP response

A response as shown in Figure 2 starts with a returncode that informs about success or failure of the request. There may then be header fields for additional information. If the response contains data, it is sent after the header fields as response body.

2.3 Details

The following methods are specified:

OPTIONS request available methods on a resource
GET get content of resource
PUT write content of resource
DELETE remove resource
HEAD GET without body
POST send data that should be posted on that resource
TRACE echo content of request back to client
CONNECT reserved for connecting to proxy that acts as tunnel endpoint

These groups of returncodes exist:
1xx provisional (server is still processing)
2xx success
3xx redirection
4xx client error
5xx server error

Each group has several subcodes to denote the reason.
2.4 Summary

What can one do with HTTP?
It is capable to up- and downloading files and there is a mechanism to authenticate a client.

What does it lack?
It is not possible to lock a file, there is no management of versions of resources and no mechanism
to specify who is allowed to do what (access control).

In the next section I will introduce extensions to HTTP that add these missing features (and
even more).

3 Extensions

3.1 WebDAV

The “Extensions for Distributed Authoring” are specified in RFC2518 [10]. WebDAV (Web
Distributed Authoring and Versioning) is the commonly used name for it.

It introduces the following new features:

properties are pairs made of a name and a value and belong to a resource. They may be
alive (the server enforces syntax and semantics) or dead (the client may specify arbitrary
values). Properties may be used to store metadata such as author or date of creation of
the resource.

collections are defined as “set of resources”. In an URL they act like directories, but on the
server they may be entries in a database.

namespace operations are methods to manipulate the namespace on the server (e.g., COPY,
MOVE).

locking is an optional feature. If it is supported, then resources must have a property “sup-
portedlock” that denotes the supported types of locks for it. There are only write locks
specified. These may be exclusive (one writer only) or shared (multiple writers).

XML structures for additional method parameters in request body

3.1.1 Details

The following methods were added:

PROPFIND retrieve properties

PROPPATCH set or remove properties

MKCOL create new collection

COPY copy resource to an other location

MOVE move resource between locations

LOCK acquire a lock

UNLOCK release a lock

To the returncodes were added:
102 processing  server is still calculating

207 multistatus  Requests may affect multiple resources. Multistatus indicates that the response body contains an XML structure having individual returncodes for each resource.

422 unprocessable entity  erroneous XML request

423 locked  source or destination resource is locked

424 failed dependency  requested action depended on another action and that failed

507 insufficient storage  server storage capacity exhausted

3.1.2 Examples

MKCOL /webdisc/xfiles/ HTTP/1.1
Host: www.server.org

HTTP/1.1 201 Created

Figure 3: WebDAV request (1)

Figure 4: WebDAV response (1)

MKCOL /webdisc/xfiles/ HTTP/1.1
Host: www.server.org

HTTP/1.1 201 Created

Figure 3: WebDAV request (1)

HTTP/1.1 201 Created

Figure 4: WebDAV response (1)

MKCOL /webdisc/xfiles/ HTTP/1.1
Host: www.server.org

HTTP/1.1 201 Created

Figure 3: WebDAV request (1)

HTTP/1.1 201 Created

Figure 4: WebDAV response (1)

The second example, shown in Figure 5, is more complex. It requests all names of properties on resource “/container/”. Note that the parameter for PROPFIND is not in the header, but in an XML-structured request body. The header field “Depth:” may be used to limit recursive traversal.

As response to our second request (Figure 6) we get a multi-status. This would be neccessary if we had requested recursive traversal. The response body contains an XML structure with a response for each resource, listing the names of all properties.
HTTP/1.1 207 Multi-Status
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<multistatus xmlns="DAV:"
    <response>
        <href>http://www.foo.bar/container/</href>
        <propstat>
            <prop>
                <creationdate/>
                ...
                <supportedlock/>
            </prop>
            <status>HTTP/1.1 200 OK</status>
        </propstat>
    </response>
</multistatus>

Figure 6: WebDAV response (2)

3.2 DeltaV

The management of versions was introduced in 2002 [5]. It declares that each version has its own resource and constant URL (see Figure 7). Versions reference their predecessor(s) and successor(s) with properties. Thus there are forks and merges possible. The resource itself is still available under its own address (e.g., /foo.html). It contains always the content of the up-to-date version and a property “DAV:checked-in” identifying this version.

Optional there may be a version history resource holding references to all versions of a resource. All resources may reside on different hosts.
3.2.1 Details

The following methods were added:

**VERSION-CONTROL** Put a resource under version control.

**CHECKOUT** Mark resource as being edited by the client.

**CHECKIN** Commit changes and create new version resource.

**UNCHECKOUT** Restore pre-CHECKOUT state.

**UPDATE** Change the up-to-date version.

**LABEL** Modify labels of versions.

**MERGE** Merge two (or more) version resources into one.

**MKWORKSPACE** A workspace is a special area on the server, where checked-out resources can be stored to be edited by multiple authors concurrently.

**MKACTIVITY** Make activity. Activities are a mechanism to track logical changes to a project.

**REPORT** Reports are an extensible mechanism for obtaining information about a resource.

**BASELINE-CONTROL** With baseline-control it is possible to manage consistent snapshots of projects.

There are also many new properties defined.

A version controlled resource can only be altered when it is in the checked-out-state. Each checkin creates a new version resource. The checked-out file may be stored in a workspace on the server, on the client's machine or - if the server supports “checkout-in-place” - it may be edited directly under the address of the version controlled resource.

For a DeltaV unaware client a write-request fails, if the server requires an explicit checkout. But a server may also do checkout and checkin automatically for each write-request. (This could result in an insane amount of versions.) Alternatively it may require the client to acquire a lock and does the checkout on LOCK and the checkin on UNLOCK. (This way the creation of versions is more under control.)

3.3 Ordered Collections

In late 2003 came up with RFC3648 [12] a specification of an interface to manage the sequence of resources in a collection. The server stores the sequence as it is told by a client. Server maintained orders are not specified, but may be possible in the future as extensions. The semantics of the order is identified by the property “DAV:ordering-type” that all collections must have.

A new method ORDERPATCH can be used to manipulate the order. With the header “Position:” it is possible to specify a place in the sequence for resources manipulated by other methods.
3.4 Access Control

The youngest of the protocols is RFC3744 [6]. It was released in May 2004 and specifies the extensions that can be used to control whoever is allowed to do what. The authentication schemes from RFC2617 [8] should be used to ensure the identity of the client.

Each resource gets a property “DAV:acl”, the Access Control List (acl). An acl contains the Access Control Entries (ace) structured in XML. Each ace identifies a principal and a set of denied and granted privileges. A principal can represent a user, a program, a server or a group of principals. It is identified by the URL of its principal-resource.

The permissions can be manipulated with the new method ACL.

3.5 more

There are more extensions under development:

- CalDAV (for exchanging calendar data)
- Binding Extensions (howto make resources available under multiple URLs, like symlinks)
- WebDAV-Quota (interface for getting information about storage-space usage)
- DASL (WebDAV SEARCH) (interface for server-side searches)
- Redirect Reference Resources
- Datatypes for WebDAV properties
- Locking Protocol (collision avoidance for WebDAV locking)
- Computing the CHECKIN URI in WebDAV versioning (on CHECKOUT)
- additional properties for PROPFIND/allprop requests
- URL constraints
- HTTP ADDMEMBER Method


4 Implementations

<table>
<thead>
<tr>
<th>server:</th>
<th>client:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMD Athlon 1GHz</td>
<td>AMD Athlon 800MHz</td>
</tr>
<tr>
<td>512MB RAM</td>
<td>512MB RAM</td>
</tr>
<tr>
<td>3com 100Mbit LAN</td>
<td>Intel 100Mbit LAN</td>
</tr>
<tr>
<td>Ubuntu Linux 5.04 or</td>
<td>Ubuntu Linux 5.04 or</td>
</tr>
<tr>
<td>Windows 2003 Server</td>
<td>Windows XP Professional</td>
</tr>
</tbody>
</table>

Figure 8: Test Setup

Figure 8 shows the setup used for testing the implementations. The network was able to transfer 10MB per second using NFS. This was measured with a 2.5GB large testfile generated with dd from /dev/urandom. A second file of 500MB was used for the WebDAV implementations.
4.1 Server-side

<table>
<thead>
<tr>
<th></th>
<th>WebDAV</th>
<th>DeltaV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache2 + mod_dav</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Apache2 + mod_svn</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Microsoft IIS 6.0 + WebDAV extension</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Microsoft Sharepoint 2.0</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Tamino</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>WebFile Server</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>GroupDrive Server</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

Figure 9: Server Implementations

“mod_dav” makes apache able to understand the WebDAV extensions. Locks are stored in a
global lock-file and properties are saved in subfolders “.DAV”.

The subversion module “mod_svn” stores all data (resources and properties) in a subversion
database.

Microsoft Internet Information Server has its own WebDAV extension. It works well with
commercial clients. Linux clients could upload using PUT, but GET resulted in error 404 (not found). This may be caused by encoding problems and needs more investigation.

Sharepoint is a free add-on for the “Windows Server 2003”. It is an interactive website with
lots of Javascript. Resources and versions may be managed using the webinterface or any client.
Mozilla did not understand all Javascript elements. It supports the server-side workspace.

From Software AG [1] comes with “Tamino” an implementation that does not only completely
support WebDAV, and DeltaV, but also DASL and ACL. It is a java application that runs inside
an application server. Because of its commercial license I was unable to test it.

Another commercial product is the “WebFile Server” from xythos [3]. Resources are stored as
files on a filesystem and metadata resides in a database. It is also an application that needs an
application server. There was no possibility to test it.

Last but not least there is the “GroupDrive Server” from South River Technologies [2]. It
promises block-level access to the resources and also a block-level granular locking. It also
provides a webinterface.

4.2 Client-side

<table>
<thead>
<tr>
<th></th>
<th>WebDAV</th>
<th>DeltaV</th>
<th>up-/download (MB/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cadaver</td>
<td>yes</td>
<td>yes</td>
<td>6.0/6.5</td>
</tr>
<tr>
<td>DAVExplorer</td>
<td>yes</td>
<td>yes</td>
<td>6.0/-</td>
</tr>
<tr>
<td>mount.davfs</td>
<td>yes (no properties)</td>
<td>no</td>
<td>3.0/3.0</td>
</tr>
<tr>
<td>Gnome</td>
<td>yes (no properties)</td>
<td>no</td>
<td>2.5/2.0</td>
</tr>
<tr>
<td>BitKinex</td>
<td>yes (no properties)</td>
<td>no</td>
<td>8.0/2.5</td>
</tr>
<tr>
<td>OpenOffice.org</td>
<td>yes (no properties)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Microsoft Office XP</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Microsoft Internet Explorer 6</td>
<td>yes (no properties)</td>
<td>no</td>
<td>5.5/2.5</td>
</tr>
<tr>
<td>Xythos Drive</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>WebDrive</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>

Figure 10: Client Implementations
Cadaver is a command-line tool like “ftp”. It supports WebDAV and DeltaV. Uploads and downloads are done directly with no intermediate copy. A 2.5GB file is too large to be transferred.

DAVExplorer is a java GUI. It supports even ACL. Downloads are stored in RAM. A heap, increased to 800MB, is not sufficient to store the necessary data while downloading the 500MB testfile.

mount.davfs uses the coda kernel-module and mounts the web-folder directly into the filesystem. There is no support for properties. Down- and upstream use an intermediate copy in /tmp.

Gnome is capable to use WebDAV. It has no support for properties. Uploads use an intermediate copy in RAM, downloads cause an extremely high load on the client.

Bitkinex was originally a ftp-client. So it is naturally that it does not support properties. Uploads are efficient. A downstream may be split into several connections.

Openoffice can open and also save documents directly from a given URL. Microsoft Office is able to do the same as Openoffice. Additionally it locks documents on open and unlocks them when they are closed.

The Internet Explorer is able to open a URL “as Webfolder”. Downloads use an intermediate copy on the harddisk.

The “Xythos Drive” from xythos makes a webfolder available under a windows drive letter. Like the xythos server i was unable to test it.

From South River Technologies is the “WebDrive”. Like the “Xythos Drive” it uses a drive letter. I found no way to use the block-level access and locking, provided by their “GroupDrive Server”. All files are completely stored in a cache on the client.

5 Conclusion

WebDAV and its extensions are a powerful and versatile family of networking protocols. Not all implementations use all features. Implementations for special purposes do not need all features. Existing implementations are not usable as network-filesystem. They are too slow. Mostly because of the intermediate copy.

WebDAV is suitable as replacement for FTP (e.g., remote management of websites).

DeltaV is good for remote collaboration (e.g., SharePoint) and version management of projects (e.g., subversion).

Further investigations may look at large sets of small files and high-latency-low-bandwidth connections.
References