WebDAV

A network protocol for remote collaborative authoring on the Web

E. James Whitehead, Jr.[†] & Yaron Y. Goland[‡]
[†]Dept. of Info. and Computer Science, U.C. Irvine, USA, *ejw@ics.uci.edu*
[‡]Microsoft Corporation, Redmond, WA, USA, *yarong@microsoft.com*

Abstract. Collaborative authoring tools generate network effects, where each tool's value depends not just on the tool itself, but on the number of other people who also have compatible tools. We hypothesize that the best way to generate network effects and to add collaborative authoring capability to existing tools is to focus on the network protocol. This paper explores a protocol-centric approach to collaborative authoring by examining the requirements and functionality of the WebDAV protocol. Key features of the protocol are non-connection-oriented concurrency control, providing an upward migration path for existing non-collaborative applications, support for remote manipulation of the namespace of documents, and simultaneous satisfaction of a wide range of functional requirements.

Introduction

Despite many compelling research examples of collaborative authoring, so far their impact on actual authoring practice has been limited. While BSCW (Bentley et al., 1997) and HYPER-G (Maurer, 1996) have developed communities of use, electronic mail remains the dominant technology used for collaborative authoring, mainly due to its ubiquity.

In order to perform collaborative authoring, all collaborators need to use compatible authoring tools—typically all collaborators use the *same* tools, tailor-made to support collaboration. To collaborate using PREP (Neuwirth et al., 1994), all collaborators need to use PREP, likewise for GROVE (Ellis et al., 1989) and DUPLEX (Pacull et al., 1994).

The economics of compatibility drive network effects (Rohlfs, 1974), where the utility of each tool depends not just on the tool itself, but on the number of other people who also own compatible tools. If there were just one telephone in the world, its utility would be small, with two or three not much more useful. But when

millions, or hundreds of millions of people own telephones, their utility is immense. Like the telephone, one collaborative authoring tool by itself has limited collaborative value, but when millions have compatible collaboration technology, the utility of each tool is great. When a common network protocol ensures interoperability between collaborative authoring tools and authoring servers, as each tool adopts the protocol, the value of servers that support the collaboration protocol increases, since *without any further investment in the server*, they can now support an additional tool. The reverse is true as well. As more servers become available, the value of the authoring tools increases without any further investment in the tools, since there are now more individuals and organizations capable of collaboration. By creating a network protocol with a focus on interoperability, it is possible to generate network effects among compatible collaborative authoring technologies.

Users of existing applications have invested significant time in gaining expertise in the applications they frequently use, hence the cost to users of switching to another tool is high. Yet users must switch tools to gain the advantages of existing collaborative authoring systems. Exceptions to this rule are BSCW and MESSIE (Sasse and Handley, 1993), which leverage standard network protocols to allow the use of existing non-collaborative tools with the system; the Hypertext Transfer Protocol, HTTP, (Fielding et al., 1997) is employed this way by BSCW, while MESSIE piggybacks on the Simple Mail Transfer Protocol (Postel, 1982). CONTACT (Kirby and Rodden, 1995) also supports collaborative authoring while remaining compatible with the heterogeneous non-collaborative tools in the user's environment. Like BSCW, CONTACT provides a Web forms interface to a collaboration support system, but does not provide BSCW's HTTP upload/download support. BSCW, MESSIE, and CONTACT share our view that collaborative authoring capability is not, by itself, sufficient to overcome the costs of switching authoring tools. Unlike these systems, we go further and assert that collaborative authoring capability must be added to existing tools.

Our hypothesis is *the best way to generate network effects and to add collaborative authoring capability to existing tools is to focus on the network protocol*, the mechanism by which collaborative tools communicate. There are many reasons for focusing attention on the network protocol that underlies a remote collaborative authoring system. From a technical viewpoint, focusing on the network protocol elevates the protocol's behavior across the Internet to a first-class concern. This ensures that thorny issues are addressed up-front, such as how to ensure efficient behavior under high-latency conditions, the provision of adequate security and authentication, and giving the protocol good scalability and extensibility characteristics. If addressed poorly, these factors can cripple the widespread adoption of a remote authoring technology.

The Web Distributed Authoring and Versioning (WEBDAV) working group of the Internet Engineering Task Force (IETF) has adopted this protocol-centric approach, and developed a novel network protocol, the WEBDAV DISTRIBUTED AUTHORING PROTOCOL (hereafter called the WEBDAV protocol), which supports interoperable remote, asynchronous, collaborative authoring. The WEBDAV protocol, a detailed specification of which can be found in Goland et al. (1999), is a set of extensions to HTTP which provide facilities for concurrency control, namespace operations, and property management. The protocol allows users to collaboratively author their content *directly* to an HTTP server, allowing the Web to be viewed not just as a read-only way to download information, but as a *writeable, collaborative medium*.

IETF protocol specifications foster interoperability by providing a concrete specification used to develop multiple applications. Indeed, for a specification to advance along the standards track within the IETF, it must demonstrate two interoperable implementations of every protocol feature. This requirement, combined with the group review process within the IETF, works to ensure that the semantics of the protocol are precise. Furthermore, the focus on interoperability across many applications helps limit the likelihood that the concerns of any one program will bias the protocol's design.

Since IETF specifications can be retrieved free of charge, and there are no licensing fees for using IETF technology, IETF network protocols are very attractive to both corporate and open-source developers. This freedom of the intellectual property rights aids in the adoption of the protocol by a wide variety of tools.

One use scenario for the WEBDAV protocol is collaborative authoring of an academic paper by a geographically dispersed authoring team. When the project begins, one member of the project uses server-specific mechanisms to create username/password pairs for each collaborator, and then creates the shared document with a WEBDAV-enabled word processor by saving it to a URL. After giving each collaborator write access, they communicate the URL of the document to the rest of the team using email, or perhaps in a teleconference. The authors take turns working on the document, using email, telephone conversations, and other out-of-band communications to negotiate editing slots. During each editing pass, an author loads the document from the URL, causing it to be locked. They then work within the editor, saving directly back to the URL. When done, they perform a final save to the URL, then unlock the document. At all times, other authors can load the document directly from the URL using either their word processor (it will be read-only if locked) or their Web browser (if it can view the word processor format), to see its current status. When done, they can give the URL to colleagues interested in the paper, and create hyperlinks to the paper in other Web pages.

This paper reports on the WEBDAV protocol, presenting the requirements it

satisfies, and how these requirements, combined with the constraints of operating in the Internet environment, guided its design. Key contributions of the WEBDAV protocol include:

Protocol focus: The focus of this work is a network protocol, the abstractions and semantics which inform the syntax of octets transmitted during a TCP/IP connection. This contrasts with existing work on remote collaborative authoring which focus on user interface, awareness, and other concerns.

Providing an upward migration path for existing non-collaborative applications: There are many applications that have no support for remote collaborative authoring. The WEBDAV protocol provides these applications a way to add remote authoring support without dramatically modifying the input processing of the application, as would be necessary if the application were modified to support fine-grain synchronous authoring. Furthermore, its grain size for concurrency control is compatible with most existing file-oriented tools.

Non-connection-oriented protocol for concurrency control: The WEBDAV protocol provides long-duration exclusive and shared write locks, a well-known concurrency control technique. To achieve robust Internet-scale collaboration, where network connections may be disconnected arbitrarily, and for scalability, since each open connection consumes server resources, the duration of WEBDAV locks is independent of any individual network connection.

Support for remotely manipulating the namespace of documents: The WEBDAV protocol provides copy and move operations, and provides support for listing the members of Web collections, similar to a directory listing in a file system. Though these operations have appeared before in numerous systems, their appearance in remote collaborative authoring systems is unusual, reflecting the awareness that these systems need to provide mechanisms for navigating to the document which will be authored, and for maintaining logical groupings of documents in collections.

Simultaneous satisfaction of requirements: The WEBDAV protocol simultaneously satisfies a wide range of difficult requirements, a non-trivial engineering activity in which tradeoffs amongst goals impacted the protocol. WEBDAV addressed several concerns which are critical to adoption of a remote collaboration technology, but which are either not mentioned, or not addressed in the collaborative authoring literature, including connection security, strong authentication, internationalization, and independence from authoring environment. While WEBDAV benefited by adopting existing technologies such as Transport Layer Security (TLS) (Dierks and Allen, 1999), Digest Authentication (Franks et al., 1997), and XML (Bray et al., 1998), the identification of the requirements, and the integration of these technologies to satisfy those requirements is unique.

Extensibility: The WEBDAV protocol recognized that it would not be able to

address all features that an authoring system would need. Thus extensibility became a critical feature. The protocol has been designed with well defined feature namespaces that allow for new features to be added later on. Existing systems will always be able to recognize new commands and have sufficient information to determine if they should ignore the unknown command or fail.

The remainder of this paper presents requirements for remote collaborative authoring of Web content, then describes the WEBDAV protocol and how it meets these requirements. The key attributes of the WEBDAV protocol, discussed in the Introduction above, will be expanded in this section. Following sections will briefly describe the existing servers and clients that implement the WEBDAV protocol, and discuss the WEBDAV protocol in relation to other existing work.

Requirements in support of collaboration

Requirements given below are an abridged version of those found in Slein et al. (1998), the consensus requirements document of the WEBDAV working group.

Equal support for all content types

The Web is composed of documents, images, and objects of many content, or Internet media types (Freed et al., 1996). *A Web authoring protocol must treat all content types equally*.

A Web authoring protocol which only provides operations for resources of one preferred content type, such as HTML, or which requires authoring-specific extensions to support features such as versioning, would have limited applicability due to its lack of support for the wide variety of other content types. Furthermore, since many common content types are in constant evolution, in order to ensure stability of the protocol a strict separation between the protocol, and the format of the objects operated upon by the protocol, must be maintained. For example, during the development of the WEBDAV protocol itself, new standards for HTML 3.2, 4.0, and XML were issued, highlighting how quickly these document formats can develop and evolve. Tailoring an authoring protocol too closely to any one content type would rapidly make the protocol obsolete.

Concurrency control support

Since the Web is inherently multi-user, *a Web authoring protocol must mediate concurrent access by multiple authors.*

Early Web authoring tools encountered the “lost update problem” which occurs

when two or more simultaneous authors of a Web page clobber each other's work with successive saves to the same URL, without first merging changes by other authors. Although HTTP 1.1 added support for *detecting* lost updates through the use of unique identifiers associated with the document state, no support was provided for *preventing* lost updates in the first place.

Early on, WEBDAV decided to use long duration, whole resource locking as its concurrency control mechanism (the rationale for this choice is discussed in the section on Locking). This then allowed the WEBDAV requirements to focus on desired properties of locks:

Lock independence: the lock operation must be independent of other operations. For example, it should not be necessary to retrieve the resource to lock it (e.g., as would be the case with a GET with lock operation). The primary motivation for this requirement is to maintain a separation of concerns between locking and other HTTP operations.

Multi-resource locking: the protocol must support an atomic operation to lock multiple resources on the same server. Documents often consist of many separate files. For example, Web pages are composed of text, images, and executable content stored as separate Web resources. Hence an authoring tool needs the ability to prevent lost updates on all the components of these multi-part documents, so the document can be authored as a unit. A multi-resource lock guarantees this. The atomicity restriction ensures that if two people are trying to lock the same group of resources, only one will be granted a lock.

Write locks: the protocol is only required to support a write lock, and no read lock is necessary (or provided by the WEBDAV protocol). On the Web, by default a resource is readable, although it may be protected by access control. Therefore, HTTP does not require that a Web browser obtain a lock in order to read a resource, as is the case with traditional database locking, and retrofitting the Web with this capability was not feasible, or desirable.

Lock discovery: it must be possible to discover whether a resource is locked. This allows a user interface to be constructed which indicates whether a resource is locked.

Support for properties (metadata)

Since there is a significant amount of information about Web resources which is not directly stored in the contents of the resource, *a Web authoring protocol must provide facilities to create, modify, read, and delete arbitrary properties (metadata) on all Web resources, irrespective of content type.*

Most Web resources have associated descriptive information, such as its author, title, publisher, keywords, copyright status, content rating, etc. While HTML, with its

META tags provides a way to embed this information within HTML documents, many Web resource types have no capability for storing metadata in their data formats. For those that do, their capabilities and storage locations vary by content type, making them awkward to use. Properties, essentially *name, value* pairs, are useful for recording bibliographic information, which can later be used in searches on property values, a capability currently being developed by the DAV Searching and Locating (DASL) working group within the IETF (DASL, 1999).

Properties can be viewed as an assertion about a resource. For example, an author property asserts that the author of the resource is given in the value of the property. In a distributed system, the issue of consistency maintenance for properties is important—if a new author starts editing a resource, the author property must be updated, or it will be inconsistent. A Web server can automatically maintain the consistency of property values that it can compute, such as a property that records the length of the resource. Other properties, such as copyright status, can only be maintained by the client. WEBDAV terms properties whose syntax and semantics are enforced by the server as "live properties", and properties which are stored without processing, and without automatic consistency maintenance, as "dead properties." This leads to the requirement that *a Web authoring protocol must support both live and dead properties*.

Support for content-type independent links

As Web resources have a wide variety of relationships between them, and since existing Web links are unidirectional, and are supported by only a few content types, *a Web authoring protocol must provide operations to create, modify, read, and delete typed links (relationships) between Web resources of any content type.* Links can be used to capture a wide variety of relationships, such as the print ordering of a set of HTML documents, or related documents such as a table of contents, an index, or a glossary.

Retrieval of unprocessed source for editing

When a Web resource is retrieved using HTTP, the transmitted octets (known as the response entity body) are a *representation* of the state of the resource, and there is no guarantee that the actual state of the resource is being returned. Many Web resources exploit this, providing a representation that differs greatly from the underlying state of the resource. Examples include HTML with server-side-include directives and active server pages, which are both processed by the server before creating the response entity body, as well as CGI scripts, where the response entity body typically

has no correlation with the persistent state of the resource, the code of the CGI script itself. Since HTTP GET always returns a response entity body *after* the server has performed its processing, *a Web authoring protocol must provide support to retrieve an authoring-suitable representation of the source of a Web resource*.

Namespace manipulation

Since resources may need to be copied, or moved as a Web site evolves, *a Web authoring protocol must provide support for copying and moving Web resources*.

There are ramifications beyond mere renaming—the behavior of a server's namespace may not be uniform, and policies such as freedom from automatic download by robot, or different cache expiration dates may vary across the namespace. This requirement also follows from our user metaphor of saving to a Web site just like any other filesystem, since most "Save As" dialog boxes provide the ability to list directories and manipulate names.

The key insight is that merely focusing on providing facilities for authoring a single resource is insufficient. Web resources exist within a space of URL names, and a Web authoring protocol needs to provide support for manipulating this environment.

Support for collections

A Web authoring protocol must provide support for creating and deleting a collection, adding and removing members to/from a collection, and for listing the members of a collection.

Collections are resources which act as containers of other resources, including other collections. They provide an abstraction that can be used to group resources, and thus help reduce the burden of authoring within a large corpus of documents. Collections are also used for navigation within large document spaces, as evidenced by the ubiquitous hierarchic file navigation windows within the "Save As" and "Open" dialog boxes of current tools. Web authoring protocols need to support this navigation style. With DASL, collections will support navigation-by-query as well.

The WebDAV protocol

This section describes in detail the major features of the WEBDAV protocol—properties, locking, and namespace management—that were designed to satisfy the Web collaborative authoring requirements given in the previous section.

Properties

WEBDAV properties are *name, value* pairs. A property name is a Uniform Resource Identifier (URI), as defined in Berners-Lee et al. (1998). Property values are expressed as XML elements. Two methods are provided to manipulate properties, PROPFIND and PROPPATCH.

The PROPFIND method is used to retrieve properties, and supports three operations: retrieve all property names and values, retrieve selected names and values, or retrieve only the property names. When applied to a collection resource PROPFIND can be instructed to act recursively against the collection resource and its members. By recursively we mean that if any of the members of the collection are themselves collections then PROPFIND will be executed against the members of the sub-collections, and so on. Note that even when used recursively, only a single request is sent, and only a single response, with property information for all affected resources, is returned.

The PROPPATCH method can set or remove multiple properties on a resource in an atomic operation. That is, all the set and remove commands in the PROPPATCH will either be successfully applied in the order submitted, or none will. To support dead properties, WEBDAV servers are capable of supporting arbitrary properties, as client-maintained "dead" values. Like PROPFIND, PROPPATCH can also be applied recursively against collection resources.

For example, *DAV:source* is one normative property defined in the WEBDAV protocol. If present, it provides a link from a resource to a URL where a representation suitable for authoring can be retrieved.

Design rationale for properties

Several different designs were considered for properties. The first design used typed links: a set of URLs that expressed a relationship between two or more resources. To express the relationship "is-author-of", a link would be defined from the document being authored to a separate resource that contained the name of the author. Though powerful, this link-based design was rejected for performance reasons, since it resulted in many network requests, and because consistency maintenance of linked metadata is difficult.

The approach which was finally adopted flowed from the notion of allowing "links" to record metadata strings directly. Thus small properties could be recorded directly inside of the property's value, while large values could be accessed indirectly through a URL. This design better suits the preponderance of metadata which consists of "small" data items, such as an author's name.

The PROPFIND method allows multiple property values to be retrieved at once, thus satisfying performance requirements. XML is used for the default property syntax as a simple, yet structured format for property values. In addition, WEBDAV's additional rule that any XML elements not understood must be ignored allows property values to be extended while retaining backward compatibility. XML also allows properties to contain arbitrary data with less complexity than the dual method design.

The ultimate argument in favor of this design for PROPPATCH is the interdependence of property values and resource state. For example, if properties A & B have mutually dependent values and a client crashes between updating A and B, the resource would be left in an inconsistent state. This led to the inclusion of both set and delete commands in the PROPPATCH method, in conjunction with the atomicity requirement.

URIs are used to name WEBDAV properties, because many different groups and applications will create new properties. URIs form a controlled namespace that provides support for both centralized and decentralized extensions yet guarantees uniqueness. Registering a new URI scheme gains the benefits of a centrally controlled namespace, while using URLs allows any organization with a domain name to create new property names.

Locking

Concurrency control within the WEBDAV protocol is provided by write locks. The LOCK method supports two types of locks: "exclusive write locks" and "shared write locks". Since any kind of principal—human or Web robot—could write to a Web resource, a write lock prevents any principal other than the lock owner, from writing to a locked resource. An exclusive write lock prevents *all* principals other than the single lock owner from writing to the resource, while a shared write lock can be owned simultaneously by several users. This terminology differs from the typical database use, where a shared lock is typically taken out prior to reading a database cell.

The lock compatibility table for write locks is given in Table I.

CURRENT LOCK STATE	SHARED LOCK REQUEST	EXCLUSIVE LOCK REQUEST
None	Granted	Granted
Shared Lock	Granted	Not Granted
Exclusive Lock	Not Granted	Not Granted

Table I: Lock compatibility table. The current lock state of the resource is given in the leftmost column, and lock requests are listed in the first row. The intersection of row and column gives the results of a lock request, where "granted" means the lock was granted, and "not granted" means the lock was not granted.

Principals are identified using Digest Authentication, an extension to HTTP specified in Franks et al. (1997). This scheme, which uses multiple one-way hashes to encrypt a username, password pair, is substantially better than authentication schemes in existing remote authoring tools. For example, MESSIE (Sasse et al., 1993) only transmits an unencrypted username before granting write access to a repository. HYPER-G, whose Client-Server Protocol is described in Kappe and Pani (1996), has a much-stronger DES-encrypted username, password pair to authenticate users. Unfortunately, most remote authoring system papers do not describe how they authenticate their users, making it impossible to determine how secure they are—raising questions on how easily their concurrency control and access control schemes can be spoofed.

The WEBDAV locking mechanism is non-connection-oriented. When a lock is granted, the server returns a lock token, a globally unique identifier, to the client. So long as it stores the lock token, the client need not remain connected. When the client subsequently wants to perform write operations on the locked resource, it recreates the network connection, then passes authentication credentials and the lock token to the server along with the write request. The authentication credentials ensure the requesting principal is who they claim to be, and the lock token ensures the requesting principal's client software is aware of the lock. This eliminates cases where the same principal could have multiple applications running at the same time, only a subset of which are aware of a given lock.

Since any client can discover the lock token for an active lock by examining the *DAV:lockdiscovery* property, lock tokens are public knowledge. Thus, possession of a lock token does not, by itself, grant a client permission to write a locked resource. It is the combination of having correct authentication credentials and demonstrating knowledge of the lock, by passing the lock token, that grants the ability to write a locked resource.

Multiple resources can be locked with a single lock request by performing a recursive lock upon a collection and its children, receiving a single lock token in the

response to represent the entire lock.

Design rationale for locking

Internet connections are inherently unreliable: an application should be prepared for a connection to go down at any time. To achieve robust behavior, a protocol must be designed to gracefully handle broken connections. For robustness as well as efficiency, HTTP was designed as a "stateless" protocol, where no state is associated with a network connection. This is more robust than approaches which associate state with a network connection, since a dropped connection only affects a single request, and is more efficient, since it does not require connection state to be established every time a connection is created. It also eliminates the need for a "resynchronization" operation in the protocol to recover connection state after an interruption.

Concurrency control within the WEBDAV protocol was designed to satisfy the goals of keeping the protocol independent of content type, allowing merges to be avoided, providing robust, even disconnected, operation in the Internet environment, and offering an easy adoption path for existing non-collaborative applications like spreadsheets, word processors, and text editors.

Synchronous collaborative authoring applications such as GROVE (Ellis & Gibbs, 1989) and PREP (Neuwirth et al., 1994) willingly forgo content type independence to provide fine-grain merging down to the keystroke level. DUPLEX (Pacull et al., 1994) and ALLIANCE (Salcedo and Decouchant, 1997) both exploit special knowledge about the internal hierarchical structure of their documents to provide concurrency control on document subtrees.

While these approaches mitigate the major drawback of locking, the lack of availability of a locked document, the cost is too high: adding content-type specific knowledge to the protocol. The major problem with adding content-type knowledge to the protocol is the number, and variation among content types. There are currently 19 text/* and over 150 application/* Internet media types. Some of these are word processing formats where operations like "insert a character" and "delete a character" make sense, others are image formats with different editing semantics, such as "change pixel's color", and "draw a line". Furthermore, some content types are revised incompatibly, requiring the protocol to distinguish between—and provide support for—multiple versions of these content types.

The approach in GROVE and PREP of sending fine-grain update notifications requires a constant network connection to be held open between collaborating applications. This requires such systems to be fully network connected throughout collaborative authoring sessions, and easily reconnect accidentally disconnected

authors. In contrast, WEBDAV locking does not require a constant network connection, even allowing collaborators to completely disconnect from the network once a resource has been locked and downloaded to a local machine.

As Grinter (1996) noted in her study of configuration management system use, merging can be a complex activity, requiring coordination between authors to resolve overlapping changes, discussing the rationale behind the changes, and how the changes work together. The WEBDAV exclusive write lock is a tradeoff between preventing the need for merges and document unavailability during the lock.

Finally, WEBDAV locks have a granularity of an entire Web resource, which maps easily onto concepts like a spreadsheet, a word processing document, or an image. Since many existing applications work on files, these applications can more easily be converted to using WEBDAV than if they had adopted a locking model with finer granularity. Grain size smaller than an entire file would increase availability, but would require significant reworking of how the application accepts input, since it would now need to listen to update notifications from other collaborators. While it is conceivable that applications will eventually be restructured to provide fine-grain concurrency control for application-specific media types, the WEBDAV protocol initially aims to provide a stable intermediate level which allows applications to become collaboration-aware without a significant reengineering burden.

Namespace Management

WEBDAV provides five methods for manipulating the namespace, DELETE, MKCOL, COPY, MOVE and PROPFIND. The DELETE method, first defined in Fielding et al., (1997), takes a single argument, the name of a resource to be deleted. The WEBDAV protocol extends the DELETE method so it can be applied recursively against collection resources. The MKCOL method takes a single argument, the URL where a new collection resource is created.

The most basic form of the COPY method takes two arguments, the URL of a source resource and a destination URL. A successful COPY results in the source resource being copied and made available at the destination. A COPY method may also applied recursively against a collection resource, with best-effort (not atomic) semantics. By default any properties on the source are copied onto the destination. Live properties that cannot be supported as live properties on the destination are copied across as dead properties. For example, if the source had an iso:time_of_day property which returned the current time of day and the destination is on a different server that doesn't have the code to support iso:time_of_day as a live property then a dead property called iso:time_of_day will be created on the destination and set to the

value of the property at the time of the copy. The COPY method can be set to fail if all live properties are not copied as live or if a specified list of live properties are not copied across as live. A looser restriction is also available which specifies that the COPY should not fail for any property related reasons such as the inability to copy any properties across.

The MOVE method is defined as the *logical equivalent* of a COPY followed by a DELETE, performed as an atomic operation. Like COPY, MOVE may be recursively applied to a collection, again with best-effort semantics. Properties are handled in the same manner and with the same options as for the COPY method.

The PROPFIND method is relevant to namespace management since it is used to retrieve a listing of a collection's members. To retrieve a collection's membership list, either a recursive or single level PROPFIND is executed on the collection resource. By asking for one or more properties to be returned, the result lists every member of the collection, even if the result just states the property isn't defined on the member resource.

Design rationale for namespace operations

WEBDAV's namespace design was strongly motivated by the needs of existing authoring clients and servers, specifically ones who expect a strictly hierarchical namespace. A strictly hierarchical namespace is one in which if resource "a" and "a/b" both exist then "a" must be a collection and must contain "a/b" as a member. The "/" character specifies containment. Many systems, especially more advanced versioning environments, do not require a strictly hierarchical namespace. Their namespace is usually flat with versioning relationships used to express containment. However clients and servers who require a strictly hierarchical namespace, including all file system based programs, cannot operate properly if the hierarchical namespace requirements are not met.

For example, in a fully generic containment system, resource "a" and "a/b" could both be non-collection resources and both be contained by "a/b/c". In other words, in a generic containment system the "/" character has no special meaning. It is possible to layer a generic containment system on top of a strictly hierarchical namespace through the use of sufficient mapping information. For example, a WEBDAV server built on top of a strictly hierarchical namespace could store all files in a single container and then map WEBDAV names to those files.

The compromise solution was WEBDAV collections would enforce a strictly hierarchical namespace while a later specification (Slein et al., 1999) would provide for linking facilities that allow WEBDAV collections to refer to resources outside of their hierarchical namespace.

Example implementations

Several WEBDAV client and server applications have been implemented, providing valuable feedback during the development of the protocol and its extensions. Interoperability testing is ongoing, and while some implementations are not yet entirely compliant (normal for the early stages of deployment of a protocol), the existence of interoperable client/server pairs has demonstrated that the WEBDAV protocol is a firm basis for interoperable collaborative authoring on the Web. Current WEBDAV applications are briefly described below.

WebDAV Servers

PYDAV - The PYDAV server, by Jim Davis while at Xerox PARC, is a fully compliant WEBDAV server (Davis, 1999). It is written in PYTHON, (a scripting language created by Guido van Rossum, described in Lutz (1996)) and runs on UNIX and WINDOWS. Persistent storage for resources is the filesystem, while storage and retrieval of properties is handled by a DBM database.

The architecture of the server maintains a clean separation between the code that handles the WEBDAV Protocol, and the code that manages the repository of resources and properties. This architecture allows handlers for other protocols to access the same repository, and for the WEBDAV protocol to access the content of other repositories, such as a document management system, or a relational database. This architecture shows the promise of the WEBDAV protocol as an object integration technology, providing a standard, network-accessible front-end to a wide variety of persistent stores.

Internet Information Services 5™ (IIS 5) - IIS 5 is Microsoft™'s HTTP & Application Web Server for WINDOWS 2000™. IIS 5 is fully WEBDAV compliant and leverages the file management, property storage and locking features of the WINDOWS 2000 file system to provide WEBDAV storage, property management and locking. This integration ensures that a user cannot get around a WEBDAV lock by going directly to the file system, or lose WEBDAV properties just by copying a file somewhere else in the system. Thus users who access the server's file system directly or who go through WEBDAV will enjoy the same experience.

MOD_DAV - A module for the APACHE Web server (Fielding et al., 1997) which currently implements the Class 1 features of the WEBDAV Protocol (it does not support locking, an area of ongoing development). MOD_DAV (Stein, 1999) is an open source project led by Greg Stein that is written in C for the LINUX operating system, and handles resource and property persistence similar to PYDAV, with resources stored in the filesystem, and properties stored in GDBM.

WebDAV Clients

POSTIES - This application provides collaborative, shared notes, similar to POST-IT™ notes. Written in Java, POSTIES stores each note as a separate Web resource within a collection. A user modifies a note by clicking within a note, then starting to type. This causes a background thread to request an exclusive lock on the note. If the lock is granted, the note is locked without the user being aware of the activity. If the note is already locked, an error dialog appears, and modifications to the note are lost. Since HTTP, and hence the WEBDAV protocol, is not currently suitable for sending notification messages (a deficiency noted by many, including Trevor et al., 1997, and Dix, 1997), the DAV POSTIES application polls for changes to notes on a regular interval.

SITECOPY - Written in C by Joe Orton, this program uses the WEBDAV protocol to update a remote Web site from a local directory (Orton, 1999). SITECOPY synchronizes a remote site from a local directory, uploading, deleting and moving files on the server as necessary. Changes on the server are not reflected in the local directory; the synchronization is only one-way (two-way synchronization is an area of future work). SITECOPY demonstrates the flexibility of a protocol-based approach, since it is a type of application which was not originally considered when developing the protocol.

Internet Explorer 5™ (IE 5) – IE 5 provides support for WEBDAV through a namespace extension which allows a WEBDAV server to be displayed using the same WINDOWS EXPLORER™ interface provided for browsing the local file system. This interface provides for copy, move, delete, rename, create new file, create new collection and drag/drop functionality. Users can activate the namespace extension through the WINDOWS™ shell using a special wizard, through pre-defined shell shortcuts, through the file-open dialog or through scripting in an HTML page. In addition, the namespace extension provides access to a WEBDAV compliant off-line synchronization manager which can handle automatically downloading and synchronizing the off-line store with the server.

Office 2000™ – OFFICE 2000 fully integrates IE 5 and so supports all the WEBDAV features that IE 5 supports. In addition OFFICE 2000 also provides for rich resource management directly in its File-Open and File-Save dialogs. Thus all the WEBDAV functionality provided in the WINDOWS EXPLORER is now available directly through OFFICE 2000's File-Open/Save dialogs. This allows users to treat a WEBDAV server the same way they would treat any file system. OFFICE 2000 makes full use of WEBDAV exclusive write locks in order to enforce single access to edited files. While OFFICE 2000 supports merging, in general OFFICE users strongly prefer to avoid merges whenever possible.

Related Work

Similar network protocols to WEBDAV can be characterized as either an extension to HTTP, a non-HTTP hypertext authoring protocol, or a file transfer protocol.

To date, extensions to HTTP for remote authoring have been proprietary, implemented by just a single vendor, and have scant documentation. For example, the FRONTPAGE™ web authoring tool uses a set of remote procedure calls implemented using the HTTP POST method to perform remote authoring operations, and only work with Web servers which support them. WEBDAV differs from the FRONTPAGE extensions by using separate HTTP methods for each operation, an approach that has advantages for access control. When the name of an operation is specified both in the HTTP request line, and in the message body of an HTTP POST method request, a server implementing access control must look in two places for the name of the operation.

AOLPRESS/AOLSERVER (some details of the protocol can be found in America Online, 1998) and the Netscape Enterprise Server (Cunningham and Faizi, 1997) both extended HTTP by defining new HTTP methods. Since both of these efforts fed ideas and contributors to the WEBDAV effort, there are many similarities in approach. Both add locking, and namespace operations (copy, move, make directory, list directory) in separate HTTP methods. The Netscape extensions also have methods for properties, providing operations to set and remove name/value pairs, and for versioning, providing check-out, check-in, list history, destroy, and set default version methods. But, despite the similarities, the WEBDAV protocol differs from these extensions in many ways. The semantics of locking are well defined in the WEBDAV protocol, and are insufficiently specified in these other extensions. Furthermore, WEBDAV locks return a lock token, allowing multiple clients operated by the same principal to work within the same portion of the namespace free from overwrite conflicts caused by unexpected tool interactions. The WEBDAV lock, copy, and move operations can apply to an entire hierarchy of resources, providing support for Web pages composed of multiple resources. Unlike the Netscape extension's properties, which are simple ASCII strings, WEBDAV property values accommodate contents in multiple character sets, providing internationalization support. Since WEBDAV property values are XML, they offer better extensibility characteristics than strings. An important differentiator for WEBDAV is that it is a non-proprietary protocol, and hence more likely to be broadly adopted than any of these proprietary extensions.

Several systems have employed a non-HTTP network protocol to provide remote collaborative authoring. Open hypermedia systems, as discussed in the survey by Østerbye and Wiil (1996), employ protocols for authoring hypertext links and for transmitting link traversal events, and often also have a protocol for authoring and

retrieving hypermedia objects. Protocols in open hypermedia systems have many differences in their marshalling strategies, varying from the ASCII linearization of Lisp and Scheme data structures in HYPERDISCO (Wiil and Leggett, 1996), the use of XDR and RPC by CHIMERA (Anderson et al., 1994), to a different use of ASCII by the Open Hypermedia Protocol for Navigation (OHP-Nav), developed as a community effort by open hypermedia researchers (Davis et al., 1998). Although it does not label itself as such, the HYPER-G Client-Server Protocol (Kappe and Pani, 1996) can also be considered an open hypermedia protocol. Despite many differences, open hypermedia protocols do share some similarities in approach. All are client-server protocols, where a hypermedia-aware application is a client, and the open hypermedia system is the server. Unlike HTTP, open hypermedia protocols allow server-initiated asynchronous notification messages to be transferred to the client, where they signal a link traversal, or give collaboration awareness information. As a result, these protocols maintain a stateful, open connection during use sessions. The WEBDAV protocol differs from open hypermedia protocols by providing a stateless, non-connection- oriented protocol. The WEBDAV protocol does not treat hypertext links as first-class objects, instead modeling them as properties. This avoids the scalability bottleneck in open hypermedia systems of having a single point of control for links, and link consistency. An in-depth discussion of the control tradeoffs in the architectures of open hypermedia systems and the Web is presented in Whitehead (1999).

Finally, the WEBDAV protocol differs from the ubiquitous File Transfer Protocol (Postel and Reynolds, 1985) in several respects. Each FTP session opens two network connections, a control channel, used to send commands from client to server, and a data channel, used for transferring files from machine to machine. This differs from HTTP, which combines control and data information into a single channel, permitting even more simple client implementations. While FTP provides namespace operations equivalent to those of WEBDAV, it does not provide any form of overwrite prevention, thus making it unsuitable for authoring, and has no metadata facilities. Furthermore, an FTP session is stateful, which would lead to robustness problems for long-duration collaborative sessions.

Conclusions

This paper has explored a protocol-centric approach to collaborative authoring by examining the requirements and functionality of the WEBDAV protocol. Providing an upward migration path for existing applications has proven crucial. We have shown that freedom from licensing restrictions, large-grain locking, and namespace

management operations reduce reengineering barriers for collaborative authoring with existing applications. Several WEBDAV applications now exist, promising to generate network effects, accelerating widespread adoption of collaborative authoring.

Since compatibility among collaborative authoring tools is essential for the generation of network effects, our protocol-centric approach focused design attention on issues of interoperability, security, internationalization, extensibility, and protocol scalability. The resulting design separated several concerns: the behavior of locks from the behavior of network connections (and from other protocol operations); and the semantics of the protocol from the content types authored using the protocol. Simultaneously addressing these issues has yielded a well-engineered protocol adapted to the Internet environment.

Finally, our experience has validated the importance of designing collaborative authoring applications from the user down, and from the protocol up. Both approaches focus on important issues, different facets of the same design space.

Acknowledgements

Since the WEBDAV protocol is the consensus output of the WEBDAV working group, the authors wish to explicitly note that they are not the sole creators of the WEBDAV protocol, and gratefully acknowledge the numerous, substantial contributions made by the members of the WEBDAV working group. While a complete list of contributors to the WEBDAV protocol can be found in the acknowledgements section of Goland et al., 1999, we would like to specially thank the other authors of the WEBDAV protocol, Asad Faizi, Steve Carter, and Del Jensen, as well as working group members Larry Masinter, Jim Davis, Jim Amsden, and Judy Slein for their contributions. Similarly, the WEBDAV requirements document, Slein et al. (1998) is a consensus document of the working group and we would like to specially acknowledge the editors of this document, Judy Slein, Fabio Vitali, and David Durand for their contributions. Finally, we would like to thank David McDonald, Rohit Khare. and David Norris for their thoughtful reviews.

Jim Whitehead’s work on this project was sponsored by the Defense Advanced Research Projects Agency, and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-97-2-0021. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency, Air Force Research Laboratory or the U.S. Government.

References

America Online (1998): “AOL Server Administrator’s Guide”, America Online, 1998. http://www.aolserver.com/server/docs/2.3/html/admin.html

Anderson, K. M., Taylor, R. N., and Whitehead, Jr. E. J. (1994): “Chimera: Hypertext for Heterogeneous Software Environments”, in *Proc. 1994 European Conference on Hypermedia Technology (ECHT’94)*, Edinburgh, Scotland, Sept. 18-23, 1994, pp. 94-107.

Bentley, R., Horstmann, T., and Trevor, J. (1997): “The World Wide Web as enabling technology for CSCW: The case of BSCW”, in *Computer Supported Cooperative Work: The Journal of Collaborative Computing*, vol. 6, nos. 2-3, 1997, pp. 111-134.

Berners-Lee, T., Fielding, R., and Masinter, L. (1998): “Uniform Resource Identifiers (URI): Generic Syntax”, MIT/LCS, U.C. Irvine, Xerox. RFC 2396, August, 1996.

Bray, T., Paoli, J., and Sperberg-McQueen, C. M. (1998): “Extensible Markup Language (XML)”, World Wide Web Consortium Recommendation REC-xml-19980210, February, 1998.

Cunningham, J. and Faizi, A. (1997): “Distributed Authoring and Versioning Protocol”, Unpublished manuscript, 1997. http://www.ics.uci.edu/pub/ietf/webdav/ns_dav.html

DASL (1999): “DAV Searching and Locating Home Page”, http://www.ics.uci.edu/pub/ietf/dasl/

Davis, H., Reich, S., and Millard, D. (1998): “A Proposal for a Common Navigational Hypertext Protocol”, Open Hypermedia Systems Working Group draft, http://www.ecs.soton.ac.uk/~hcd/ohp/ohp35.htm

Davis, J. (1999): “PyDAV WebDAV Server”, http://sandbox.xerox.com/webdav/, 1999.

Dierks, T. and Allen, C. (1999): “The TLS Protocol Version 1.0” Certicom. RFC 2246, Jan., 1999.

Dix, A. (1997): “Challenges for Cooperative Work on the Web: An Analytical Approach”, in *CSCW: The Journal of Collaborative Computing*, vol. 6, nos. 2-3, 1997, pp. 135-156.

Ellis, C.A., and Gibbs, S. J. (1989): “Concurrency control in groupware systems”, in *Proc. ACM SIGMOD ’89 Conference on the Management of Data*, Seattle, WA, May 2-4, 1989.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., and Berners-Lee, T. (1997): “Hypertext Transfer Protocol -- HTTP/1.1”, U.C. Irvine, DEC, MIT/LCS. RFC 2068, January, 1997.

Fielding, R., Kaiser, G. (1997): “The Apache HTTP Server Project”, *IEEE Internet Computing*, 1(4), July/August, 1997.

Franks, J., Hallam-Baker, P., Hostetler, J., Leach, P., Luotonen, A., Sink, E., and Stewart, L. (1997): “An Extension to HTTP: Digest Access Authentication”, Northwestern Univ., CERN, Spyglass, Microsoft, Netscape, Spyglass, Open Market. RFC 2069, January, 1997.

Freed, N., Borenstein, N. (1996): “Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies”, Innosoft, First Virtual. RFC 2045, November, 1996.

Goland, Y. Y., Whitehead, Jr., E. J., Faizi, A., S. R. Carter, and D. Jensen (1999): “HTTP Extensions for Distributed Authoring -- WEBDAV”, Microsoft, U.C. Irvine, Netscape, Novell. RFC 2518, February, 1999.

Grinter, R. (1996): “Supporting Articulation Work Using Software Configuration Management Systems”, in *CSCW: The Journal of Collaborative Computing*, vol. 5, 1996, pp. 447-465.

Kappe, F., and Pani, G. (1996): “Hyper-G Client-Server Protocol (HG-CSP)”, in Maurer, H. (ed.) 1996, pp. 550-591.

Kirby, A. and Rodden, T. (1995): “Contact: Support for Distributed Cooperative Writing”, in *Proc. Fourth European Conference on Computer Supported Cooperative Work (ECSCW’95)*, Stockholm, Sweden, September 10-14, 1995, pp. 101-116.

Lutz, M. (1996): *Programming Python*, O’Reilly & Associates, Cambridge, MA.

Maurer, H. ed. (1996): *Hyper-G, now Hyperwave: The next generation Web solution*, Addison-Wesley, Harlow, England.

Neuwirth, C. M., Kaufer, D. S., Chandhok, R., and Morris, J. H. (1994): “Computer Support for Distributed Collaborative Writing: Defining Parameters of Interaction”, in *Proc. ACM 1994 Conference on Computer Supported Cooperative Work (CSCW’94)*, Chapel Hill, NC, October 22-26, 1994, pp. 145-152.

Orton, J. (1999): “sitecopy Home Page”, http://www.lyra.org/sitecopy/

Østerbye, K., and Wiil, U. (1996): “The Flag Taxonomy of Open Hypermedia Systems”, in *Proc. Hypertext’96*, Washington, DC, March 16-20, 1996, pp. 129-139.

Pacull, F., Sandoz, A., and Schiper, A. (1994): “Duplex: A Distributed Collaborative Editing Environment in Large Scale”, in *Proc. ACM 1994 Conference on Computer Supported Cooperative Work (CSCW’94)*, Chapel Hill, NC, October 22-26, 1994, pp. 165-173.

Postel, J. (1982): “Simple Mail Transfer Protocol”, ISI. RFC 821, Standard 10, August, 1982.

Postel, J. and Reynolds, J. (1985): “File Transfer Protocol (FTP)”, ISI. RFC 959, October, 1985.

Rohlfs (1974): “A theory of interdependent demand for a communications service”, *Bell Journal of Economics*, vol. 5, no. 1, 1974, pp. 16-37.

Sasse, M. A., Handley, M. J. (1993): “Support for Collaborative Authoring via Email: The MESSIE Environment”, in *Proc. Third European Conference on Computer-Supported Cooperative Work (ECSCW’93)*, Milan, Italy, September 13-17, 1993, pp. 249-264.

Salcedo, M. R. & Decouchant, D. (1997): “Structured Cooperative Authoring for the World Wide Web”, in *Computer Supported Cooperative Work: The Journal of Collaborative Computing*, vol. 6, nos. 2-3, 1997, pp. 157-174.

Slein, J. A., Vitali, F., Whitehead, Jr., E. J., and Durand, D. (1998): “Requirements for a Distributed Authoring and Versioning Protocol for the World Wide Web”, Xerox, Univ. of Bologna, U.C. Irvine, Boston Univ. Informational RFC 2291, Feb., 1998.

Slein, J., Davis, J., Babich, A., Whitehead, Jr., E. J. (1999): “WebDAV Advanced Collections Protocol”, Internet-Draft, work-in-progress, draft-ietf-webdav-collection-protocol-03, Feb. 26, 1999.

Stein, G. (1999): “mod_dav: A DAV module for Apache”, http://www.webdav.org/mod_dav/

Trevor, J., Koch, T., and Woetzel, G. (1997): “MetaWeb: Bringing synchronous groupware to the World Wide Web”, in *Proc. of the Fifth European Conference on Computer Supported Cooperative Work (ECSCW’97)*, Lancaster, UK, September 7-11, 1997, pp. 65-80.

Whitehead, Jr., E. J. (1999): “Control Choices and Network Effects in Hypertext Systems”, in *Proc. Hypertext ’99*, Darmstadt, Germany, February 21-25, 1999, pp. 75-82.

Wiil, U., and Leggett, J. (1996): “The HyperDisco Approach to Open Hypermedia Systems”, in *Proc. Hypertext’96*, Washington, DC, March 16-20, 1996, pp. 140-148.