Author-Oriented Link Management

Michael Creech
Broadband Information Systems Laboratory

HPL-96-44
March, 1996

Author-Oriented Link Management

Michael Creech
Broadband Information Systems Laboratory
HPL-96-44
March, 1996

author-oriented link management, WWW, web authoring, enterprise, CLT/WW

Currently, a major issue with the World Wide Web (WWW) is link management-keeping web structures consistent whenever pages are moved, deleted, or changed. This paper describes a link management technique that helps authors ensure the consistency of their content. This technique, called the change log table/web-walk (CLT/WW) approach, is based on web operations that log the gross changes made by authors to their web pages. We discuss the interesting properties of this approach, including: managing cross-site linking of many web sites, ease of integration into existing content management systems, and decoupling web operations from link management. The issues surrounding this technique and potential extensions to this approach are also discussed.

1 Introduction

Link management is the process of keeping web structures consistent whenever web resources (pages) are moved, deleted, or changed. Link management embodies two types of web structures—web pages created by authors, and links saved by browsers (e.g., hotlists). Author-oriented link management involves both informing authors whenever their pages have incorrect links, as well as automatically repairing these links whenever possible. This paper focuses on author-oriented link management, since author's pages are the source of all links and are therefore the most important to manage.

Link management is required whenever a page is modified in a way that affects the HyperText Markup Language (HTML) [1] links referring to that page. One simple example is whenever a page is moved, all the hypertext links to that page are broken until these links are updated. Other types of page modifications that can affect links include deleting, splitting, merging, and changing a page's topic.

Page modifications occur for a variety of reasons, including the restructuring of pages based on their growth and evolution, administrative changes related to disk space limitations and performance limitations, and page removal or archiving due to aging.

The absence of link management is a serious problem within the current World Wide Web (WWW). As Ingham, et. al. [2] aptly state:

> "Such broken links are the single most annoying problem faced by browsing users in the current Web. Broken links result in a tarnished reputation for the provider of the document containing the link, annoyance for the document user, and possible lost opportunity for the owner of the resource pointed to by the link."

Unlike small hypertext systems [3], which tend to have bidirectional links, the WWW has unidirectional links, which, in the large scale, make it impossible to know who references a specific resource—making repair of these references very difficult. To make matters worse, the exponential growth of the WWW—with ever increasing amounts of content—leads to more and more links being created, and to more and more people making changes.

The small amount of link management support that currently exists in the WWW only covers updating browsers concerning moved resources. By manually updating a server configuration table, the HyperText Transfer Protocol (HTTP) [4] redirect directive provides a forwarding pointer to the new location of the moved resource. A browser (e.g., Netscape) attempting to access the old resource receives a redirect request to update its hotlists, and other information. This minimal solution only handles the movement of resources, and only helps updating browsers—it is not author-oriented; and, therefore, does not repair any author's documents that point at the old resource location. There is also no way of determining when this forward

reference can ever be removed, and another resource can never be moved into its place.

This paper describes a link management technique that helps authors ensure the consistency of their content. This technique, called the change log table/web-walk (CLT/WW) approach, is based on web operations that log the gross web changes made by authors.

Since our focus is on helping authors, Section 2 begins by considering the link management needs of the author. This includes understanding the common web operations authors perform, the information they require to repair link inconsistencies, the constraints placed on link management by the content management systems they use, and when they need to have links automatically repaired.

Section 3 introduces the CLT/WW approach, including the characteristics of this approach, and the requirements for integrating this approach into existing content management systems. The section will end with a simple one-site example of how this approach works. Section 4 expands the discussion to enterprise link management—managing the cross-site linking of two or more content sites that are part of an organization. The remaining sections present related work, plans for future work, and concluding remarks.

2 Link Repair, Content Management, and Web Operations

The link management capabilities desired by an author include having links automatically updated, whenever possible, and when not possible, having the appropriate information to make reasonable decisions on how to update these links. The ability to automatically update links is strongly affected by the way authors publish their content. We will begin by considering the feasibility of automatic link repair based on the content management model employed.

When automatic link repair is not possible, authors need to know *what* has changed as well as *how* it has changed. For example, an author needs to know that page A was deleted, instead of finding a broken hypertext link to A, or, even worse, not knowing that the link to A is broken. In order to specify how a page has changed, we must define some web operations that affect an author's links. To this end, we will complete this section by identifying five such common web operations.

2.1 Content Management and Automatic Link Repair

The feasibility of automatically updating an author's links is strongly determined by the content management model employed. By content management model, we simply mean the process by which authors edit, test, and publish their pages. This may be as simple as an author directly editing a page on an HTTP server, or may involve an author submitting page changes into a multi-staged content management system, in

which multiple copies of the page may exist for testing, verification, and review before being published on a "hot" server for general consumption.

Automatic link repair is affected by the content management model because in order to do such repairs, the content must be changed "upstream" as far as possible in the content management process to avoid being re-released in its uncorrected form—potentially undoing link repairs that were just made. The best place to make such changes is in the author's originating source content, but this may not be possible. For example, if an author's source pages are stored on his PC, and he publishes pages by submitting them to a content management system, simply fixing the pages in the content management system is not enough, since we must also fix the source pages on his PC. When the originating source pages are not available, link repairs may simply consist of notifying an author of a needed change in her content and letting her manually change her source content at her convenience.

2.2 Moving Pages

Moving a page involves physically moving the page in question, and keeping consistent the web of links that refer to this page. It is possible to automatically update links that point to a moved page as long as the above restrictions on when links can be automatically repaired are honored.

Note that movement of pages can take place within the same server (intra-server) and between servers (inter-server). Inter-server moves are more complex and have other complications, such as whether the appropriate Common Gateway Interface (CGI) [5] based scripts, server-side directives and includes, and correct versions of software exist to correctly serve up this page on the new server.

2.3 Deleting Pages

Deleting pages is very similar to moving pages, in that the links to the pages being deleted need to be updated. When pages are deleted, it is generally not possible to automatically update links as with pages that have moved, because the deletion of a page often requires more than deleting the links that reference it. This can be seen in the example of an HTML subsection with one bulleted item that contains a link reference. If the page referred to by the link is deleted, the author may want to delete the whole subsection, or restructure his document in some other fashion. Thus, the author must be notified and decide how to change the content manually.

2.4 Splitting Pages

Splitting pages means dividing a page into two or more pages. Many times this maps to narrowing the scope of an existing page. Since it is very difficult to know which of the split pages to point to, it is generally not possible to automatically update links to split pages. Thus, the best we can do is to tell the author that a link may be incorrect and now points to a page that was split into the following set of pages. With this information, the author can determine how to update the link.

2.5 Merging Pages

Merging pages involves combining the contents of two or more pages into one page. Many times this translates into widening the scope of a page. It is possible to have all the links that pointed to the pre-merged pages be automatically updated to point to the merged page. However, authors may still want to be notified since the semantics of the referenced page may have changed.

2.6 Changing Page Titles

An HTML page title change is just that—changing the "title" of the page, not changing the page's Uniform Resource Locator (URL) [6][7]. This frequently occurs when the contents of a page are broadened or narrowed. A page title change is not equivalent to moving a page because all the referencing links are still correct, however each of the anchor names of these links may now be incorrect. Updating link anchors is not necessarily straightforward, because the link anchor title may be different from the title of the referenced page. In cases where the anchor is different, the author should be notified of this difference so that the anchor name can be changed. It may be possible to have link anchors changed automatically by having the author include some sort of extra attribute with their link. If this were the case, link management tools could automatically update the link anchor name.

3 Change Log Table/Web-Walk Approach

Since link management involves keeping links to a modified web page consistent, how do we obtain all the links to this page? The most obvious way is to focus on the links themselves and keep track of all links (their source and destination pages) by building an inverted index.[1] Each time a link is added or deleted, it is entered into the index. Then, we could search this index to retrieve all the pages that point to a modified page. There are two main problems with this approach. First, it is not scalable—it is very difficult to manage several autonomous content sites.[2] And, secondly, this solution is closed in the sense that all the tools used for creating or modifying links must be modified to update this index. For example, if we edit page A and remove a link to page B, this fact must be reflected in the index.

A less obvious approach which avoids these problems, focuses on the *pages themselves*. This approach builds a table containing a log of the operations performed on the pages (not the links). We call this a *change log table*[3] (*CLT*). A CLT might contain operations like *move* $A \rightarrow A'$, *delete* B, and *change-title* C *from* X *to* Y. The CLT is used in conjunction with a `web-walk` program[4] that walks the web structure(s) at

1. Or, storing all links in a database.
2. By site, we mean a group of authors that use one set of content management tools to publish content on one HTTP server.
3. This is not to be confused with HTTP server log files.
4. This program could be built upon an existing link validation web-walking robot, such as ChURL [8], Checkbot [9], or MOMSpider [10].

a site. `web-walk` checks each HTML page for links[5] that refer to pages in the CLT, and, if found, notifies the appropriate person and potentially repairs links.

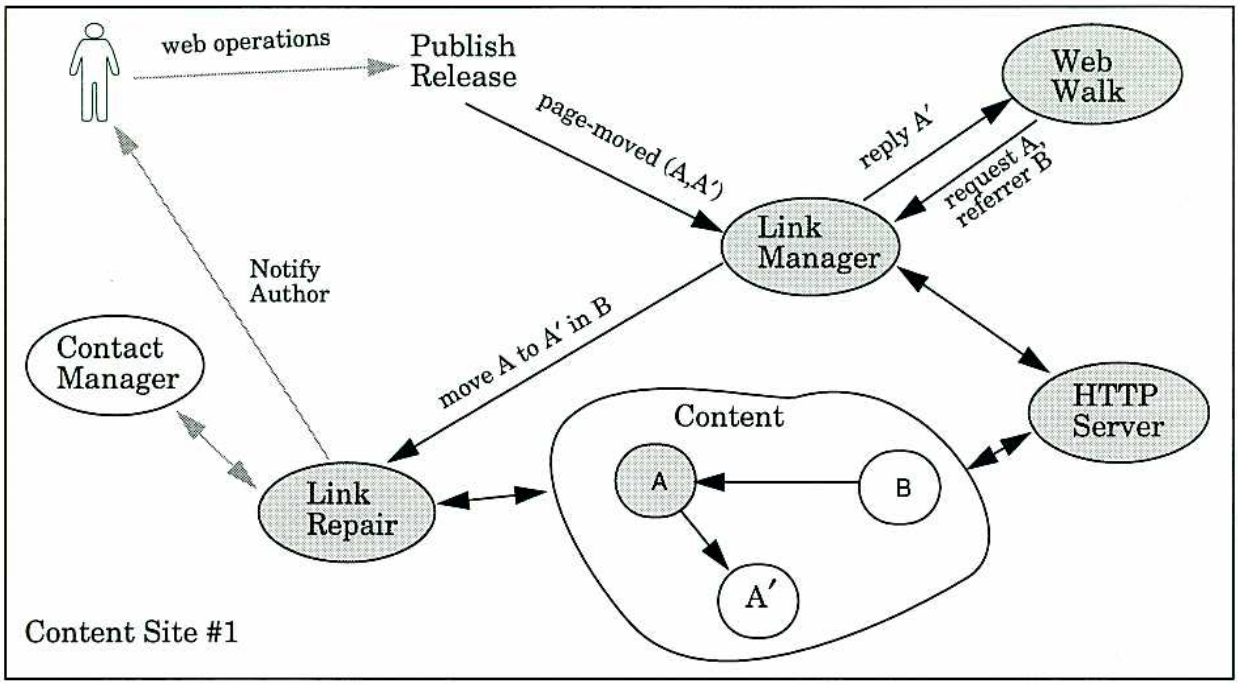

Figure 1 Content at one site

Figure 1 shows the major components necessary for link management using the CLT/WW approach on one content site. The gray arrows in the figure represent the areas that interface with the outside world. In the "Content" area of this figure, page A has been moved to A′, but the link from B to A has not yet been updated.

The major components of this approach are:

`web-walk`—brings a site back to a link-consistent state. It walks the local web structure (only pages served at this site) performing notifications and repairing links as it goes. It starts with a set of root pages and uses the `link-manager` server (an HTTP-based server) to retrieve pages.

`link-manager`—a server that is the central "brain" for ensuring consistent link structures. The `link-manager` performs three main operations:

1. It updates the CLT to correspond to changes in the web structure.
2. It returns the contents of a page or information about it, given its URL.
3. When it finds web inconsistencies, it calls `link-repair` to notify the author about the inconsistency and to potentially repair it.

5. Note that `web-walk` will not find links in dynamically generated pages, such as those produced by CGI scripts.

The CLT is contained within the `link-manager`. This table contains site web changes made since the last time the site's `web-walk` completed. The CLT houses the URL, operation performed, and any extra information required. Sample CLT entries are "A was deleted," and "B was moved to C."

`link-repair`—fixes web inconsistencies, and notifies authors about inconsistencies via e-mail.

`contact manager`—this is a component used by `link-repair` that determines whom to notify about links that have been repaired and links that need to be repaired. It also supplies permission information that controls whether a link can be automatically repaired. This component is assumed to exist in the content management environment.

3.1 Independence of Basic Web Operations from Link Management

With this technique, web operations are independent of link repairs. This means that the various web operations, as discussed in Section 2, perform all the operations that constitute their behavior, except for fixing links, which is performed by `web-walk`.

This web operation/link repair independence is essential for ensuring the scalability of this approach when handling multiple autonomous sites (see Section 4). This approach decouples content update from the complex and slow operation of fixing links. This is especially true when updating content cannot be done automatically, and the author is not available for a long time (e.g., an author goes on vacation).

3.1.1 Drawback to Independence

Web operation/link repair independence poses a synchronization problem when pages are published during a `web-walk`. In this case, we must wait until the next scheduled `web-walk` to consider these pages. This ensures that no needed repairs or notifications are missed by the fact that the `web-walk` had partially completed before these pages were published. Note that we cannot solve this problem by simply waiting until the next `web-walk` completes, because this introduces the problem of renotifying authors and refixing links for the link references that were *not* missed by the previous `web-walk`. To solve this problem, any requests to the `link-manager` to add CLT information during the `web-walk` are queued until the `web-walk` completes. Queuing is easily accomplished by marking the CLT entry as "new," if a `web-walk` is in progress, and ignoring "new" entries during a `web-walk`. Note that we cannot simply lock out publishing during a `web-walk`, since a `web-walk` may take a considerable amount of time.

3.2 Requirements for Use with Content Management Systems

This approach requires the following capabilities from its associated content management system:

1. The ability to track the web operations defined in section 2. This requires some degree of modification to the content management process, or to the link

management tools used. For example, if a content management process requires authors to deposit their content changes by filling out a form, the form may need to be changed to gather information about the web operations performed (e.g., deleted B). If authors use simple scripts for performing link management operations (e.g., move-node), these scripts must be modified, or wrapped, to track these operations.

Tracked web operations must be saved until content is actually published. This is only required when content is not immediately published, as in a content management system where authors test their changes in a staging area before publishing their changes. No saving is necessary if web operations are performed directly on the content being served. Instead, this information can be directly passed to the `link-manager`.

2. A hook into the content management process at the point where content is published, that passes the tracked operations from the previous step to the `link-manager`.

3. The ability to perform a `web-walk`. This requires access to the root page(s) for the site, and a content administrator (or some time-based process) that starts `web-walk`.

4. A way to notify authors and to obtain permission to automatically repair links (e.g., a contact manager).

Given the requirement to track web operations, it might appear that the CLT/WW approach is no better than building an inverted index—each requires changes to link management tools, or to the content management process. However, the difference is in the granularity and frequency of changes that must be tracked. With the CLT/WW approach, we only need to track web operations such as those outlined in Section 2. For example, two important categories of changes that require no tracking by this approach are, creating new pages, and making link modifications to existing pages. Thus, an author could use any editing program to create new pages and to edit the text and links on existing pages.

3.3 How the Pieces Fit Together—an Example

The following is a simple example showing how the pieces in Figure 1 work together.

1. An author uses a link management tool to move page A to A′. This tool does not update all the pages that point to A. The tool logs the change (move A→A′).

2. The author's change is published, causing the information saved in tracking this change to now be handed to the `link-manager`, which adds it to the CLT. If a `web-walk` is taking place when this item is added, the item is marked as new and not processed by the current `web-walk`.

3. Sometime later, a `web-walk` is initiated. It traverses the local web structure, starting from all the root pages on this site. `web-walk` takes a root page's URL and passes it to the `link-manager` to return the contents of this root. Links to other pages at this site are found within this root page, and `web-walk` calls itself recursively, handing itself the URL of the destination for each of these links.[6]

4. When the `link-manager` receives a request for a URL, it first looks up the URL in the CLT, searching for a source argument that matches this URL. If the URL is found in the CLT (as in looking up page A from within page B):

   i. A change has occurred to this URL since the last time the local web was made consistent (as with page A). In this case, the `link-manager` passes this URL (A), the URL of the referencing page (B), and any other information from the CLT to `link-repair`.

   ii. `link-repair` contacts the `contact manager` to find the contact person for the reference page (B) and permission information for performing automatic link repair. With this information, `link-repair` notifies the appropriate person about the inconsistency, and may also automatically fix the links in the reference page (B), depending on the permission information and the type of the inconsistency.

   If no entry for the URL is found within the CLT:

   iii. The `link-manager` calls the site's HTTP server with the URL, and the results of this request are passed back to `web-walk`.

   The `link-manager` returns either the contents of a page, or information about the status of a page. Status information is returned when a page was deleted, split into several pages, or cannot be found. The contents of the page are returned for all other cases. When a page was moved, or merged into another page, this new page's contents are returned. For example, if page A were moved to A′, then the contents of A′ would be returned.

5. `web-walk` continues recursively scanning the HTML pages returned to it, as long as these pages haven't been seen before and the pages are within the site. `web-walk` handles replies from the HTTP server (passed back through the `link-manager`), such as the 404 response (Not Found), as well as `link-manager` information ("page deleted").

4 Enterprise Link Management

The CLT/WW approach can help solve the more complex problem of link management for an "enterprise." By enterprise, we mean two or more autonomous content sites each using potentially different processes for publishing content. Each site may also

6. The web-walker must be able to handle cycles in the web structure.

use different HTTP servers (e.g., NCSA HTTPd [11], OpenMarket WebServer [12]). What distinguishes an enterprise from random sites on the Internet is that enterprise content sites belong to an organization that can dictate link management processes or tools that each site must use when doing content management. This leads to a certain consistency across the enterprise and allows a specific approach, such as the CLT/WW approach, to be adopted throughout the enterprise. An enterprise is likely to have content linked across its different sites. In such an environment, the major link management problem becomes how to update links that point to pages on different sites (cross-site links).

With the CLT/WW approach, content for an enterprise is treated in a fashion similar to content for one site, with a few major differences:

1. A new `enterprise-manager` server is needed that tracks what sites embody the enterprise. It notifies site `link-managers` about changes to the sites in the enterprise and is used by these `link-managers` to determine what other sites exist. When a `web-walk` begins, the `link-manager` at that site must first contact the `enterprise-manager` to see if any site information has changed. Note that if single point of failure is a major concern, replication of the `enterprise-manager` may be needed.

2. `web-walk` must now access the first level of pages outside the site (treated as leaf nodes). It contacts the `link-managers` at other sites to retrieve off-site page information.

3. When an incorrect cross-site link is found, the `link-manager` at the site pointed to by the link must return an update request, along with its normal return value, to the calling `web-walk`. When the calling `web-walk` receives this request, it passes the request to its own site's `link-manager`. Then, the `link-manager`, with `link-repair`, determines the appropriate action.

4. Each site broadcasts[7] the beginning and ending of its `web-walk` to the `link-managers` at all the other sites. Each site must keep track of whether another site is in the middle of performing a `web-walk`.

5. Two additional fields per site are added to all CLT entries for managing multi-site synchronization (see Table 1). These are discussed below in terms of the problems they address.

4.1 Multi-site Synchronization Issues

For the enterprise, three new synchronization issues emerge:

1. Ensuring that a CLT entry is active for at *least* one full `web-walk` by each and every site.

---

7. Although broadcasting tends to be associated with scalability limitations, each site only infrequently broadcasts a small amount of information.

Table 1: Sample Multi-Site CLT Entry

Op	Site #1		...	Site #n	
	new entry?	web-walk completed?	...	new entry?	web-walk completed?
move A→A′	Yes	No	...	No	Yes

Considering an entry for less than one full `web-walk` can lead to missed repairs and notifications—as described in the single site case. This presents a problem when CLT entries are added to a site concurrent with offsite `web-walks` that are looking at links to this site. What is required is that each site track all `web-walks` that are in progress. When a CLT entry is added to a site, it must be marked as a "new entry" for all sites that have a `web-walk` in progress. While marked as new, this entry will be ignored by the `link-manager` in lookups requested by the sites that have a `web-walk` in progress. For example, if a CLT entry at Site #C was marked as being a "new entry" for Sites #M and #N, any lookups by Site #C's `link-manager`, in answering `web-walk` requests, will ignore this entry if the requesting sites are #M or #N.

An entry will be unmarked as new for a particular site, when this site's `web-walk` completes. In Table 1, the entry shown was added while Site #1's `web-walk` was in progress.

2. Ensuring that a CLT entry is active for at *most* one full `web-walk` by each and every site.

   Since some sites may perform two or more `web-walks` before a CLT entry is removed, we must ensure that no attempts are made to refix links and renotify authors on these sites. To eliminate this problem, all non-new CLT entries are marked as "web-walk completed", when a site completes its `web-walk`. When the CLT is searched a second time by the same site, all entries marked as completed for that site are ignored. In Table 1, the entry shown has had a `web-walk` completed by Site #n. A CLT entry can be removed when all of its site entries are marked as "web-walk completed."

3. Sites may be added, deleted, or have their names changed during other sites' `web-walks`.

   To solve this problem, when a site is changed in one of these ways, it must send a message to the `enterprise-manager`, which relays it to all the enterprise sites. Each site then receives an *add-site*, *remove-site*, or *change-site* message. When an add-site message is received, a new site is added to each CLT entry. CLT entries will not be removed until this new site performs a `web-walk`. When a remove-site message is received, it removes that site from each CLT entry. A

check is then performed to see if any CLT entries can be deleted. When a change-site message is received, the site registers an internal name change for referring to this site.

Given the many interacting servers and components that are required for link management of an enterprise, it would be very difficult for this scheme to guarantee link repairs and notifications. But, even though servers may crash or be taken down for maintenance, network partitions may form, or authors may not perform needed link repairs, this approach should still move the web toward a state of consistency. It does imply, however, that under the worst conditions, authors will have to update their pages as they do today.

4.2 How the Pieces Fit Together—a Multi-site Example

We conclude this section with an example of how enterprise link management is performed using the CLT/WW approach. This example augments the previous single-site example in Section 3.3, with a second site (see Figure 2). In this example, we have authors Andra, Rob, and Mike. Site #1 contains Andra's page A, and Mike's page C, that points at A. Site #2 houses Rob's page B, which also points at A (cross-site link). This example will show Andra moving A to A′, followed by a `web-walk` of Site #1, and then a `web-walk` of Site #2:

1. Andra uses a link management tool to move page A to A′. This tool does not update all the pages that point at A. The tool logs the change (move A→A′).
2. Andra publishes her change which causes the Site #1 `link-manager` to add her change as a new CLT entry. If a `web-walk` were in progress on either Site #1 or Site #2, the appropriate column of this entry would be marked as a "new entry." However, no `web-walk` is in progress.

Sometime later:

3. Site #1's `web-walk` program is initiated. It begins by checking with the `enterprise-manager` to see if any changes have occurred to the number or names of other sites (assume none have occurred). It then broadcasts to all `link-managers` that it has begun a `web-walk`. The Site #1 and Site #2 `link-managers` log this fact in order to catch any concurrently added CLT entries.
4. Site 1's `web-walk` begins walking Site 1's content. For local pages, Site 1's `link-manager` is called to return pages. For a nonlocal page, `web-walk` attempts to contact the `link-manager` at the site where the page resides. Assume that `web-walk` is now looking at Mike's page C and investigating the link to page A.
5. Site #1's `link-manager` is called to look up page A, referenced by page C. It finds the CLT entry for A moving to A′. It calls `link-repair`, which contacts Mike stating that page C's reference to page A was changed to page A′ (since

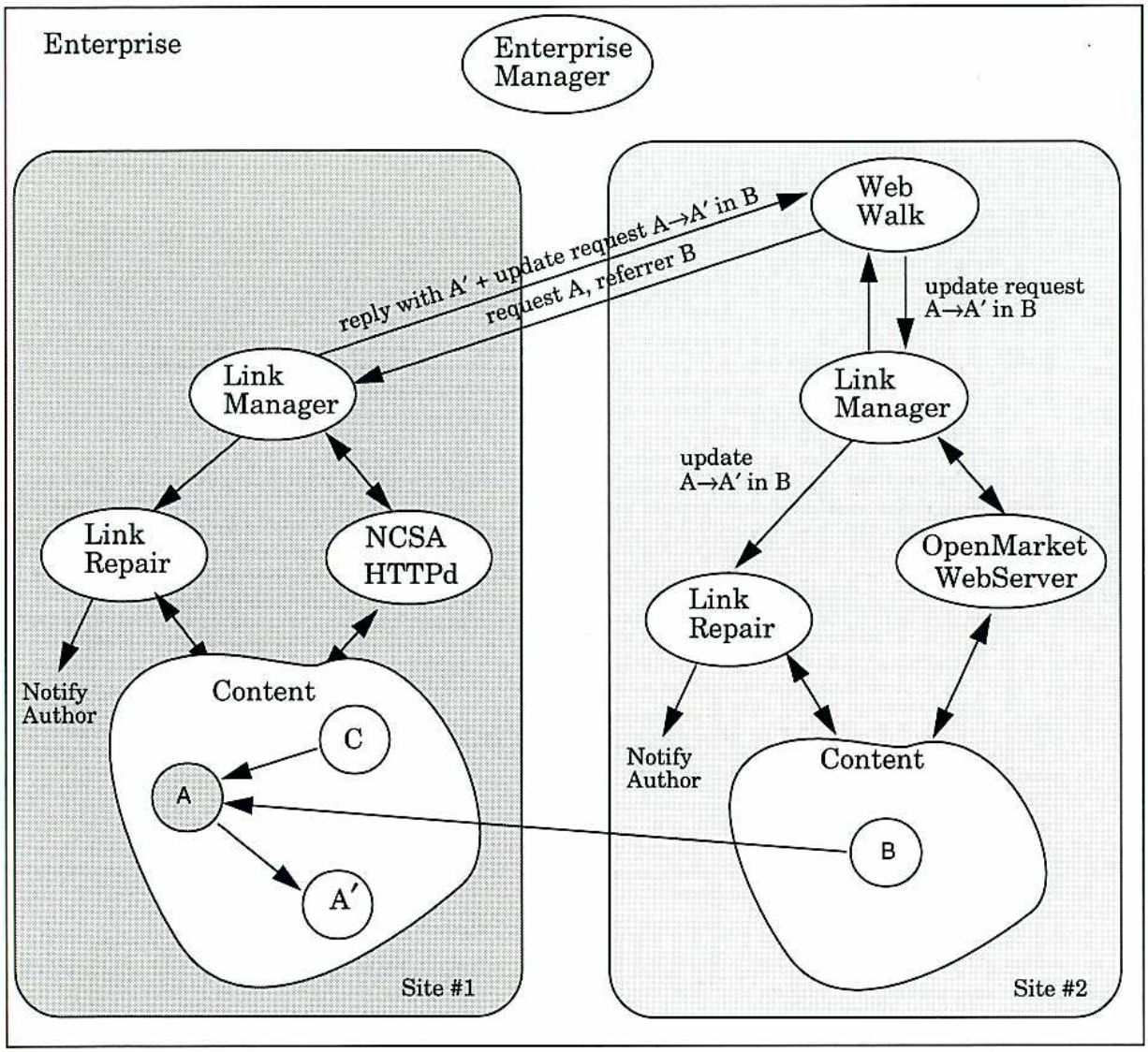

Figure 2 Content across multiple sites—across-site linking example, moving page A→A′

Mike allows links to be automatically repaired, `link-repair` changes the link reference in page C). The `link-manager` returns A′ to Site #1's `web-walk`.

6. Site #1's `web-walk` finishes and then broadcasts to all `link-managers` that it has ended its `web-walk`. Site #1's `link-manager` marks all non-new CLT entries as "web-walk completed" for Site #1 (this will mark our one entry, move A→A′). The `link-manager` then checks if any CLT entries can be deleted. Our one entry cannot be deleted, since Site #2 has not yet performed a `web-walk`.

Sometime later:

7. Site #2's `web-walk` program is initiated. It begins by checking with the `enterprise-manager` to see if any changes have occurred to the number or names of other sites (assume none have occurred). It then broadcasts to all `link-managers` that it has begun a `web-walk`; the Site #1 and Site #2 `link-managers` log this fact.

8. Site 2's `web-walk` begins walking Site 2's content. Assume it is now looking at Rob's page B and investigating the cross-site link to page A.

9. Site #1's `link-manager` is called to look up page A, referenced by page B. It finds the matching CLT entry (move A) and returns A′ to Site #2's `web-walk`. It also returns an update request, stating that B's link to A should be changed to A′. Upon receiving this request, Site #2's `web-walk` passes it to the Site #2 `link-manager` and continues walking the web, investigating page A′.

10. Given the update request, Site #2's `link-manager` calls Site #2's `link-repair` which contacts Rob, stating that page B's link to page A should be changed to page A′ (Rob forbids automatic link repair to his pages).

11. Site #2's `web-walk` finishes and broadcasts to all `link-managers` that it has ended its `web-walk`. Site #1's `link-manager` marks its one CLT entry as "web-walk completed" for Site #2. This entry is then deleted, since all sites have performed a `web-walk` for this entry.

5 Related Work

One recent approach to link management uses object-oriented technology to encapsulate WWW resources [2]. These objects have interesting properties, which include ensuring that pages continue to exist as long as another page links to them (referential integrity), as well as keeping links consistent when pages are moved (migration transparency). This environment is not targeted at authors and therefore provides no support for informing authors of important changes to the resources that affect their content (e.g., the title of a referenced page has changed).

Web validation tools, such as MOMspider [10] and Netscape's LiveWire Site Manager [13], can aid link management tasks by finding broken links, and helping administrators delete or change all the links at a site that point at a specific page. However, they do not tell authors *how* a page has changed. This means that an author can determine that a document is inaccessible, but will not be informed that the referenced document was deleted. These tools are also limited in that they only manage a single site; they do not handle cross-site linking within an enterprise.

The work of the Internet Engineering Task Force (IETF) on Uniform Resource Names (URN's) [14] defines a logical naming scheme that allows resources to be seamlessly moved, gives easy access to meta-information about a resource (through their closely associated URC's [15]), and allows resources to simultaneously exist in many

locations on the Internet. Unfortunately, URNs have their limitations and will not be completed and adopted for some time to come. For example, URN's do not handle page deletion, merging, splitting, or page title changes. Furthermore, URNs add complexity to the generation of page names (pages must be cleared by a naming authority), their name resolution is slower and more complex, and they are "opaque"—users cannot look directly at a URN and gain any insight about its meaning.

6 Future Work

Many questions must still be answered in order to determine the limitations of this approach. A prototype system must be constructed and tested in the hope of answering such key questions as: (1) the number of sites this approach can control; (2) the impact of large numbers of cross-site links; (3) how database technology can aid the CLT; (4) the number of pages that can be changed at a site between `web-walks`; (5) the effect of large numbers of pages referencing one page; (6) the rate at which `web-walks` can be performed, based on how much content is contained within a site; and (7) the "brittleness" of this approach, based on server and component failure.

Some other areas for research and extension involve:

1. Building an "incremental web walker" that improves the performance of the slow web-walking process. This web walker would only walk over heavily linked and heavily used areas of the web, thereby encouraging more frequent web walks and, therefore, more frequent link management, while reducing the need for full `web-walks`.
2. Extending the `enterprise-manager` so that it stores useful meta information for improving enterprise link management, such as content management policies, and site `web-walk` scheduling.
3. Expanding this approach to update client browsers. One approach would be to place an HTTP proxy [16] between each HTTP server and its clients. The proxy would simply pass requests and replies back-and-forth between the server and the clients, until it sees a server response of 404 (Not Found) or 301 (Permanently Moved). For these responses, the proxy calls the `link-manager` to return a result.
4. Building a web operation undo facility based on the record of web operations stored in the CLT.

7 Conclusion

The CLT/WW approach is an author-oriented link management technique that notifies authors of needed changes and automatically updates authors' documents.

Because it focuses on gross changes to pages, not link references, this approach is easier to integrate into existing content management systems, requires no changes to existing HTTP servers, and can manage the complexities of cross-site linking within an enterprise. Although link repairs and notifications cannot be guaranteed, the CLT/WW approach is a viable way of moving sites and enterprises toward a state of web consistency.

8 Acknowledgments

Thanks to John Dilley, Dennis Freeze, Gita Gopal, Gary Herman, Tai Jin, and Jeff Morgan for their comments and suggestions.

9 References

[1] Berners-Lee, T., and Connolly, D. "Hypertext Markup Language - 2.0". IETF, RFC: 1866. Nov 1995. URL: http://www.cis.ohio-state.edu/htbin/rfc/rfc1866.html

[2] Ingham, D., Little, M., Caughey, S., and Shrivastava, S. "W3Objects: Bringing Object-Oriented Technology to the Web." *Fourth International World Wide Web Conference*. Boston, Massachusetts, USA. Dec 1995.
URL: http://www.w3.org/pub/Conferences/WWW4/Papers2/141/

[3] Creech, M., Freeze, D., and Griss, M. "Using Hypertext in Selecting Reusable Software Components." *Proceedings of the Third ACM Conference on Hypertext*, pages 25-38, San Antonio, Texas, USA. Dec 1991.

[4] Berners-Lee, T., Fielding, R., and Frystyk, H. "Hypertext Transfer Protocol—HTTP/1.0". IETF Internet-draft, expires Aug 1996. Feb 1996. draft-ietf-http-v10-spec-05.html. URL: ftp://ietf.cnri.reston.va.us/internet-drafts/draft-ietf-http-v10-spec-05.txt

[5] The CGI Specification. URL: http://hoohoo.ncsa.uiuc.edu/cgi/interface.html

[6] Berners-Lee, T., et al. "Uniform Resource Locators (URL)." IETF, RFC: 1738. Dec 1994. URL: http://www.cis.ohio-state.edu/htbin/rfc/rfc1738.html

[7] Fielding, R. "Relative Uniform Resource Locators." IETF, RFC: 1808. Jun 1995. URL: http://www.cis.ohio-state.edu/htbin/rfc/rfc1808.html

[8] ChURL Home Page.
URL: http://www-personal.engin.umich.edu/~yunke/scripts/churl/

[9] de Graaff, H. Checkbot. URL:http://dutifp.twi.tudelft.nl:8000/checkbot/

[10] Fielding, R. "Maintaining Distributed Hypertext Infostructures: Welcome to MOMSpider's Web." *First International Conference on the World Wide Web*. Geneva, Switzerland. May 1994. URL: http://www.ics.uci.edu/WebSoft/MOMspider/WWW94/paper.html

[11] NCSA HTTPd Home Page. URL:http://hoohoo.ncsa.uiuc.edu/

[12]Open Market, Inc. "WebServer Product Description." URL: http://www.openmarket.com/products/servers/server.htm

[13]Netscape Communications Corp. "NETSCAPE INTRODUCES NETSCAPE LIVEWIRE AND LIVEWIRE PRO VISUAL ONLINE DEVELOPMENT ENVIRONMENT." URL: http://www.netscape.com/newsref/pr/newsrelease41.html

[14]Sollins, K. and Masinter, L. "Functional Requirements for Uniform Resource Names." IETF, RFC: 1737. Dec 1994. URL: http://www.cis.ohio-state.edu/htbin/rfc/rfc1737.html

[15]Hamilton, M. "UNIFORM RESOURCE IDENTIFIERS & THE SIMPLE DISCOVERY PROTOCOL." Loughborough University of Technology (LUT CS-TR 985). June 95. URL: http://gizmo.lut.ac.uk/~martin/uris/uris.html

[16]A. Luotonen, A. and Altis, K. "World-Wide Web Proxies." *First International Conference on the World Wide Web*. Geneva, Switzerland. May 1994. URL: http://www1.cern.ch/PapersWWW94/luotonen.ps